МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

П. П. ЛІЗУНОВ, В. О. НЕДІН, І. Д. КАРА

КОМП'ЮТЕРНІ МЕРЕЖІ ТА ТЕЛЕКОМУНІКАЦІЇ

Конспект лекцій
для студентів спеціальності
073 «Менеджмент»
освітньої програми «Менеджмент організацій і адміністрування»

Київ 2025

УДК 658:681.518
Л55

Рецензент: Г. М. Іванченко, доктор технічних наук, професор

Затверджено на засіданні навчально-методичної ради КНУБА, протокол № 1 від 26 вересня 2024 року.

Лізунов П. П.

Л55 Комп’ютерні мережі та телекомунікації : конспект лекцій / П. П. Лізунов, В. О. Недін, І. Д Кара. – Київ : КНУБА, 2025. – 240 с.

Розглянуто основні положення та питання застосування комп’ютерних мереж у практичних задачах менеджменту, які потребують здійснення управління підприємством і ресурсами підприємства, а також ведення неперервної роботи зі співробітниками та клієнтами з використанням мережевих засобів.

Призначено для студентів спеціальності 073 «Менеджмент» освітньої програми «Менеджмент організацій і адміністрування».

УДК 658:681.518

ЗМІСТ

Вступ

Дисципліна «Комп'ютерні мережі та телекомунікації» є розділом загальної дисципліни «Інформаційні технології». Теми лекцій та їх зміст розраховані на студентів спеціальності «Менеджмент» освітньої програми «Менеджмент організацій і адміністрування».

У процесі вивчення курсу студент має ознайомитись із різновидами комп'ютерних мереж, мережевими службами, системами управління інформаційними ресурсами, мовами web-програмування, що використовуються для створення різноманітних інтернет-систем.

Під час вивчення матеріалу студент має оволодіти знаннями щодо використання систем управління взаємовідносинами з клієнтами, систем управління людськими ресурсами, систем планування ресурсів підприємства, а саме тих, що використовуються із застосуванням мережевих інтернет-технологій. Також студентам потрібно навчитися створювати web-сайти з використанням систем управління вмістом та інтернет-магазини з використанням інтернет-систем, що дають можливість керувати такими ресурсами.

Для вміння рекламувати діяльність компанії, просувати продукти чи послуги підприємства на ринку студент має ознайомитись з інструментами створення та проведення рекламних кампаній в мережі Інтернет.

Наприкінці кожної лекції є контрольні запитання, які допомагатимуть студентам провести самоконтроль засвоєння матеріалу.

За результатами вивчення викладеного в конспекті лекцій матеріалу студент отримає теоретичні та практичні знання щодо використання сучасних мережевих інформаційних технологій у практиці управління підприємством і ресурсами підприємства, а також ведення неперервної роботи зі співробітниками та клієнтами.

Лекція 1

КОМП'ЮТЕРНІ МЕРЕЖІ, ТЕЛЕКОМУНІКАЦІЇ, МЕРЕЖА INTERNET

1.1. Комп'ютерні мережі

За фізичного з'єднання двох або декількох комп'ютерів утворюється *комп'ютерна мережа* (рис. 1.1).

Рис. 1.1

Комп'ютерна мережа (*Computer Network*) – це сукупність *електронно-обчислювальних машин* (*ЕОМ)*, об'єднаних між собою засобами передачі даних.

Засоби передачі даних у загальному випадку складаються з таких елементів, як *комп'ютери*, *канали зв'язку*, *комунікаційні пристрої*.

Для з'єднання комп'ютерних мереж потрібні спеціальне апаратне обладнання (*мережеве обладнання*) і спеціальне програмне середовище (*мережеві програмні засоби*).

Мережеве обладнання *(телекомунікаційне обладнання)* – пристрої, необхідні для роботи комп'ютерної мережі, наприклад: *маршрутизатор*, *комутатор*, *концентратор* та ін.

Мережеві програмні засоби – це програмне забезпечення, яке дає змогу організувати роботу користувача в мережі. Воно представлено *загальним*, *системним* і *спеціальним* програмним забезпеченням.

Комп'ютерні мережі належать до розподілених обчислювальних систем. Поряд із *комп'ютерними мережами* до

розподілених систем відносять також *мультипроцесорні* та *багатомашинні обчислювальні комплекси*.

Мультипроцесорні системи використовують декілька процесорів, кожен з яких може незалежно від інших виконувати окрему програму. У мультипроцесорній системі існує спільна для всіх процесорів *операційна система*.

Багатомашинна система (***кластер***) – це обчислювальний комплекс, що складається з декількох комп'ютерів, а також програмних та апаратних засобів, що забезпечують роботу всіх комп'ютерів комплексу як одного цілого. У такій системі кожен із комп'ютерів працює під управлінням власної операційної системи.

Основною ознакою *розподіленої обчислювальної системи* є наявність декількох *центрів обробки даних*.

Центр обробки даних – це приміщення у споруді або окрема будівля чи група будівель, де розташовані комп'ютерні системи (сервери), що призначені для збирання та зберігання великих обсягів інформації.

Усе *мережеве обладнання* працює під керуванням *мережевого* і *прикладного* програмного забезпечення.

Згідно з моделлю ***OSI*** (*Open Systems Interconnection*) архітектуру комп'ютерних мереж потрібно розглядати на різних рівнях (загальна кількість рівнів – сім).

Верхній рівень – ***прикладний*** (користувач взаємодіє з обчислювальною системою).

Нижній рівень – ***фізичний*** (забезпечує обмін сигналами між пристроями).

Рівні

1. Прикладний рівень.
2. Рівень представлення.
3. Сеансовий рівень.
4. Транспортний рівень.
5. Мережевий рівень.
6. Рівень з'єднання.
7. Фізичний рівень.

Прикладний рівень – створення документа користувачем.

Рівень представлення – фіксація положення документа й забезпечення взаємодії з іншими рівнями.

Сеансовий рівень – взаємодія комп'ютера користувача з локальною або глобальною мережею.

Транспортний рівень – перетворення документа у форму передачі даних, яка використовується в мережі.

Мережевий рівень – визначення маршруту передачі даних у мережі.

Рівень з'єднання – модулювання сигналів для фізичного рівня відповідно до даних, отриманих із мережевого рівня.

Фізичний рівень – безпосередня передача даних сигналами.

Для забезпечення необхідної сумісності на кожному рівні діють спеціальні стандарти, що мають назву ***протоколи***. Існують:

апаратні протоколи – визначають характер апаратної взаємодії;

програмні протоколи – визначають характер взаємодії програм і даних.

1.2. Локальні та глобальні мережі

Відповідно до *протоколів*, що використовуються, комп'ютерні мережі поділяються на:

- ***локальні*** *(LAN – Local Area Network)*;
- ***регіональні*** *(MAN – Metropolitan Area Network)*;
- ***глобальні*** *(WAN – Wide Area Network).*

Локальна мережа – мережа, яка прив'язана до однієї певної організації. Вона об'єднує комп'ютерні системи й периферійні пристрої в групи, які сумісно використовують пристрої та дані.

Регіональна мережа – мережа, яка охоплює місто, регіон, область. Має багато спільного з *LAN*.

Глобальна мережа – мережа, яка розповсюджується на країни і в багатьох випадках створюється шляхом об'єднання *LAN* і *MAN*.

У разі утворення комп'ютерної мережі виникає проблема вибору конфігурації зв'язку, або ***топології***.

Під ***топологією мережі*** розуміють конфігурацію пристроїв мережі і комунікаційного обладнання.

1.3. Мережеві топології

Ієрархічна топологія. Кожен із пристроїв забезпечує безпосереднє керування пристроями, нижчими за ієрархією (рис. 1.2).

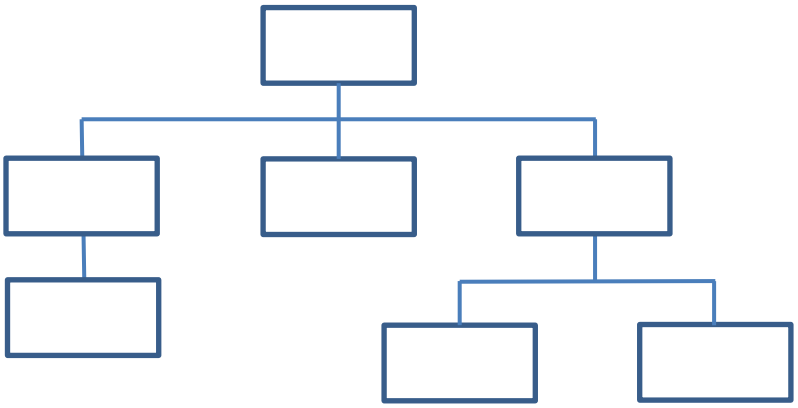


Рис. 1.2

У наш час така топологія є найпоширенішим типом топології зв'язку як у локальних, так і глобальних мережах.

Ієрархічна топологія також називається *деревоподібною*. У більшості випадків мережею керує комп'ютер на верхньому рівні ієрархії і поширення трафіку між іншими комп'ютерами ініціюється головним комп'ютером ієрархії.

Топологія шини. Кожен із пристроїв під'єднаний до магістралі й не залежить від іншого. Така топологія може використовуватись у мережі *Ethernet* (рис. 1.3).

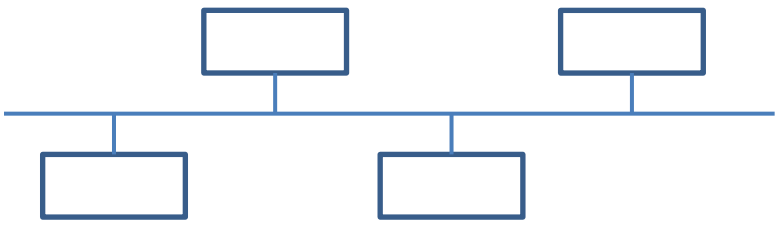


Рис. 1.3

Як центральний елемент у топології шини виступає пасивний кабель, до якого підключається низка комп'ютерів. Інформація, що

передається, розповсюджується по кабелю й доступна одночасно всім комп'ютерам, які приєднані до цього кабелю.

Ethernet – сімейство технологій пакетної передачі даних між пристроями для локальних дротових комп'ютерних мереж. Дає змогу пристроям взаємодіяти між собою за протоколом, який є спільною мережевою мовою.

Мережа *Ethernet* є локальною мережею комп'ютерів та інших електронних пристроїв, яка охоплює невелику площу в кімнаті, офісі, будинку або будівлі. За способом з'єднання може поділятися на *ThinNet, ThickNet, Twisted Pair*.

ThinNet – невелика локальна мережа *Ethernet* у межах кімнати чи офісу, у якій для з'єднання комп'ютерів використовується тонкий коаксіальний кабель, у вузлах з'єднань якого використовуються Т-подібні конектори.

ThickNet – локальна мережа, у якій об'єднується декілька невеликих локальних мереж, використовуючи для з'єднання як магістраль товстий коаксіальний кабель.

Twisted Pair – мережа, у якій для з'єднання комп'ютерів використовуються кабельні дроти типу «вита пара».

Кільцева топологія. Маршрутизація даних здійснюється по замкненому контуру. Мережами з такою топологією є *Token Ring* (рис. 1.4).

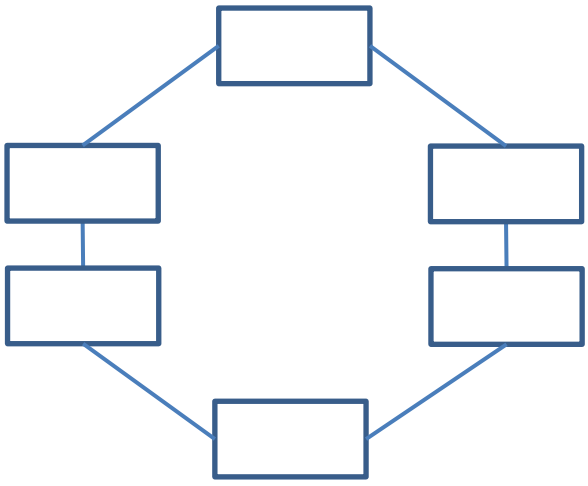


Рис. 1.4

Основною перевагою кільця є те, що будь-яка пара вузлів поєднана двома шляхами – за годинниковою стрілкою і проти неї.

Кільце є дуже зручною конфігурацією для організації зворотного зв'язку – дані, зробивши повне коло, повертаються до вузла-джерела. Тому відправник може контролювати процес доставки даних адресату.

Token Ring – технологія передачі даних у локальній обчислювальній мережі з топологією кільця. У системі постійно передається ***маркер*** – символ права доступу до мережі. Обладнання може розпочати передачу даних, лише захопивши *маркер*. Після того як адресат отримав надіслані йому дані, він запускає у мережу новий *маркер*.

Зіркова топологія. Центральний вузол відповідає за маршрутизацію даних через себе. Така топологія може використовуватись у мережах *Ethernet* і *Token Ring* (рис. 1.5).

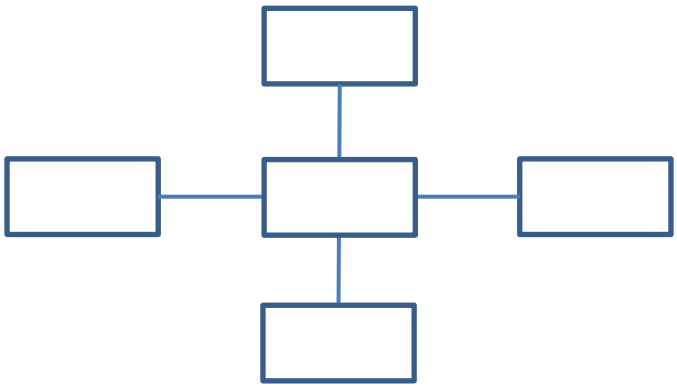


Рис. 1.5

До функцій концентратора належить спрямування інформації, що передається від одного комп'ютера до іншого або всім іншим комп'ютерам мережі. Концентратором може бути комп'ютер або спеціалізований пристрій, як-от багатовходовий повторювач, комутатор чи маршрутизатор.

Комірчаста топологія. Маршрутизація даних між двома пристроями здійснюється найкоротшим можливим шляхом (рис. 1.6). У таких мережах можна виділити окремі, довільно пов'язані фрагменти (підмережі), що мають типову топологію, тому їх можна віднести до мереж зі змішаною (гібридною) топологією.

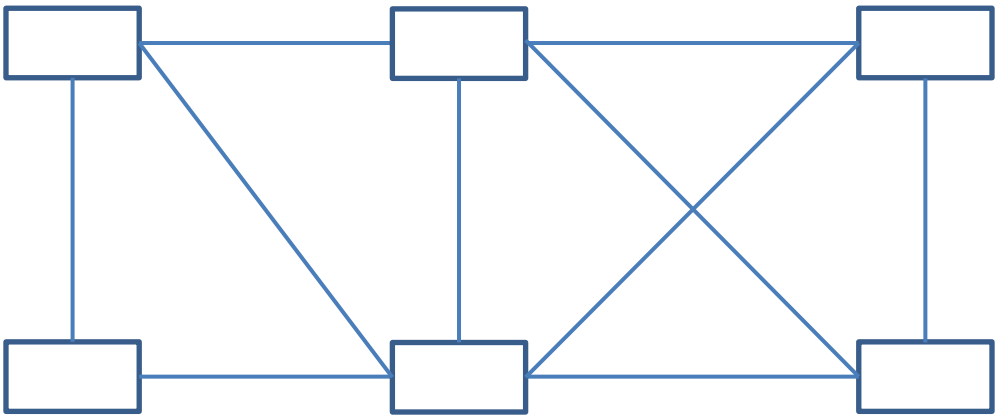


Рис. 1.6

1.4. Телекомунікації

Телекомунікації – це сукупність засобів, які забезпечують перенесення інформації, поданій у відповідній формі, на значну відстань за допомогою поширення сигналів в інформаційному середовищі або сукупності середовищ.

Процес передачі, отримання й обробки інформації з використанням *інформаційних технологій* має назву ***телекомунікація***.

До засобів телекомунікації належать:

- кабельний зв'язок,
- оптоволоконний зв'язок,
- супутниковий зв'язок,
- мобільний зв'язок,
- радіозв'язок,
- телебачення.

Телекомунікаційна мережа (*Telecommunication Network*) – це сукупність засобів телекомунікацій, що надає віддаленим об'єктам можливість інформаційної взаємодії шляхом обміну сигналами.

До телекомунікаційних мереж належать:

- комп'ютерні мережі,
- пошукові системи,
- системи електронної комерції.

У комп'ютерних мережах *мережевим інтерфейсом* (рис. 1.7) називають точку з'єднання між комп'ютером користувача та

мережею, також точку з’єднання комутованої телефонної мережі та телефону, а також точку з’єднання двох мереж між собою.

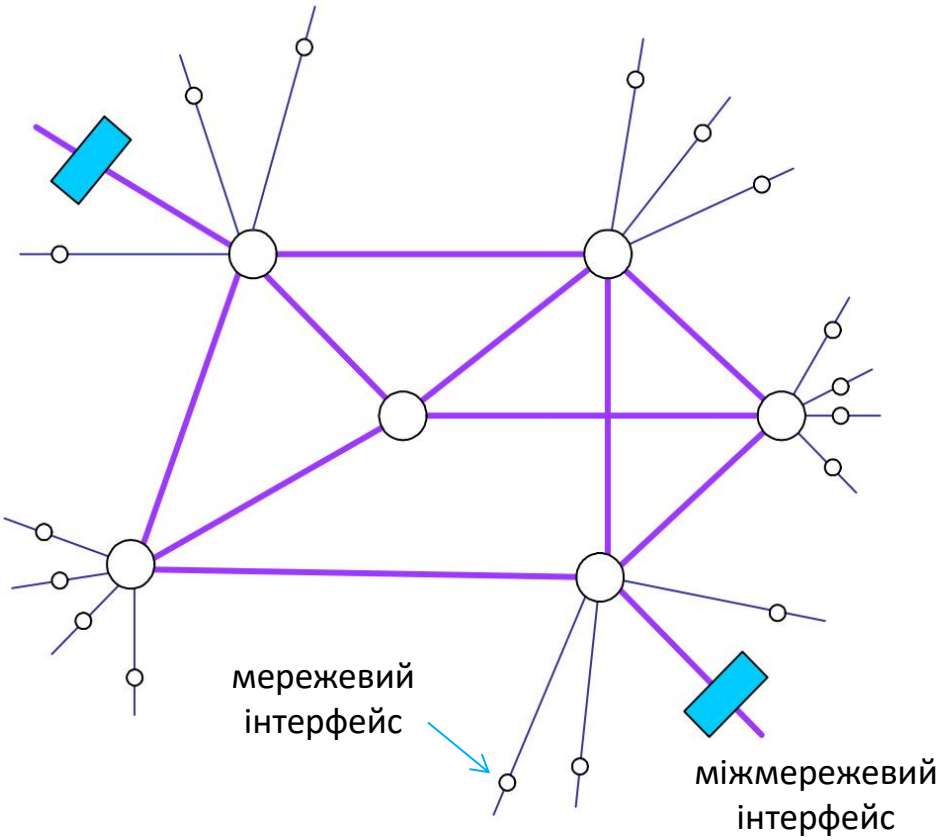


Рис. 1.7

Мережевий інтерфейс – це системний інтерфейс між двома частинами обладнання або шарами протоколів у комп’ютерній мережі.

Міжмережевий інтерфейс – функціональний блок, за допомогою якого здійснюється зв’язок між двома комп’ютерними системами.

Кабельний зв’язок – це тип телекомунікації з використанням кабельного з’єднання для передачі сигналів від джерела до приймача.

Оптоволоконний зв’язок – це тип телекомунікації із використанням імпульсів світлового випромінювання через оптичне волокно.

Супутниковий зв’язок – вид космічного радіозв’язку, що базується на використанні штучних супутників Землі, на яких змонтовані ретранслятори. Супутниковий зв’язок здійснюється між

земними станціями, які можуть бути як стаціонарними, так і мобільними.

Мобільний зв'язок – зв'язок із застосуванням радіотехнологій, під час якого кінцеве обладнання споживача може вільно переміщуватися в межах телекомунікаційної мережі, зберігаючи єдиний унікальний ідентифікаційний номер мобільної станції.

Радіозв'язок – це електротехнічний зв'язок, який здійснюється за допомогою радіохвиль із використанням трансляторів і приймачів сигналу, що передають інформацію в реальному часі.

Телебачення – різновид радіозв'язку, сукупність методів і засобів трансляції та приймання візуальної інформації в реальному часі.

1.5. Глобальна мережа Internet

Internet – це комунікаційна всесвітня система об'єднаних комп'ютерних мереж для зберігання та передачі інформації.

Internet – це:

- засіб розповсюдження інформації,
- засіб для спілкування людей.

Мережа Internet – це Всесвітня міжнародна система об'єднаних комп'ютерних мереж, що будується на комплексі *Internet*-протоколів (*IP*) та маршрутизації пакетів даних.

Internet утворює глобальний інформаційний простір, є фізичною основою для Всесвітньої павутини *World Wide Web* і великої кількості інших систем передачі даних.

Структуру ***Internet*** складає низка організацій.

IANA *(Internet Assigned Numbers Authority – управління призначенням адрес в Internet)* – організація, яка виконує контроль за розподілом всього простору *Internet*-адрес, включно з *IP*-адресами. *IANA* виділяє адресний простір *регіональним реєстратурам* відповідно до їх потреб.

RIR *(Regional Internet Registry – регіональна реєстратура Internet)* – організація, яка займається розподілом адресного простору в межах одного з трьох регіонів (Америка, Європа, Азія).

Регіональні реєстратури виконують координацію діяльності *локальних реєстратур*.

LIR *(Local Internet Registry – локальна реєстратура Internet)* – організація, яка займається розподілом адресного простору користувачам мереж (сервіс-провайдерам та їхнім абонентам) і наданням супутніх реєстраційних послуг. Отримання статусу *LIR* означає отримання незалежного блоку адрес і номера *автономної системи* (*AS*).

AS *(Autonomous System – автономна система)* – група маршрутизаторів (шлюзів) одної адміністративної області, що взаємодіє з іншими автономними системами через зовнішній протокол маршрутизації (*External Gateway Protocol – EGP*) з використанням усередині *AS* внутрішнього протоколу маршрутизації (*Interior Gateway Protocol – IGP*). Схему взаємодії автономних систем показано на рис. 1.8.

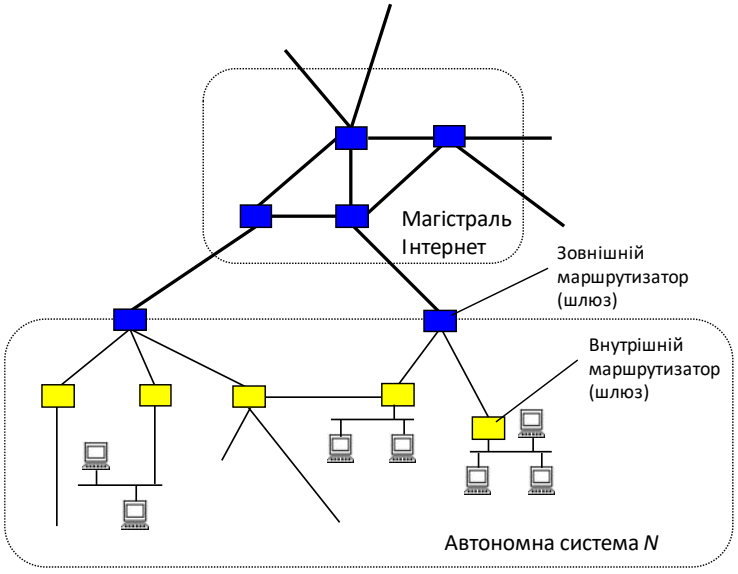


Рис. 1.8

Маршрутизатор – спеціальний комп'ютер із відповідним *програмним забезпеченням* (*ПЗ*), який слугує для визначення найкоротшого маршруту в мережі до місця призначення.

Шлюз – спеціальний комп'ютер із відповідним *ПЗ*, який слугує для об'єднання мереж із різною будовою і протоколами обміну даних.

Міст – спеціальний комп'ютер з відповідним *ПЗ*, який слугує для об'єднання мереж з однаковою будовою і протоколами обміну даних.

У структурі *глобальної мережі* можна виділити три рівні.

Перший – внутрішній рівень. Становить мережу передачі даних і складається з вузлів зв'язку. Кожен вузол зв'язку являє собою сукупність засобів передачі даних і складається з комутаційної *ЕОМ* та апаратури передачі даних.

Другий рівень становлять різноманітні сервери, так звані *хост-ЕОМ* (*host computer*), які виконують в мережі задачі зі зберігання й обробки даних. Такими серверами можуть бути, наприклад, сервери різних локальних мереж.

Третій рівень – термінальний, складається зі звичайних клієнтських робочих станцій, які користуються послугами глобальної мережі *Internet*.

1.5.1. Способи адресації в Internet

У мережі ***Internet*** використовуються три способи адресації:

- апаратна адреса (*MAC*-адреса): 00:E0:29:78:96:FF;
- числова адреса (*IP*-адреса): 194.85.160.50;
- символьна адреса (доменне ім'я): de.info.ua.

MAC-адреса *(Media Access Control – управління доступом до носія)* – адреса, яка дає змогу унікально ідентифікувати кожен пристрій у мережі й доставляти дані тільки на цей пристрій.

IP-адреса *(Internet Protocol)* – адреса комп'ютера, яка складається із чотирьох чисел. Ліва частина вказує, до якої ділянки мережі належить комп'ютер, права – містить номер комп'ютера, який повинен отримати інформацію.

Доменне ім'я – частина простору ієрархічних імен мережі *Internet*, що обслуговується групою серверів системи доменних імен (*DNS*-серверів).

DNS-сервер (*Domain Name Service – служба доменних імен*) – сервер, що зберігає інформацію про вузли, імена яких належать домену, і виконує переведення їх імен в *IP*-адреси. Кожний домен

має унікальне ім'я, а кожен комп'ютер, під'єднаний до мережі *Інтернет*, має зазвичай доменне ім'я.

Служба доменних імен (*Domain Name Service, DNS*) виконує перетворення доменного імені в числову *IP*-адресу і навпаки. Служба *DNS* для перетворення доменного імені в *IP*-адресу використовує *DNS*-сервери. На них у базах даних зберігається інформація відповідності доменного імені *IP*-адресі.

1.5.2. Формування *IP*-пакета даних

IP-пакет – форматований блок інформації, що передається через комп'ютерну мережу (рис. 1.9), структуру якого визначено протоколом *IP (Internet Protocol)*. За використання пакетного форматування мережа може надсилати довгі повідомлення більш надійно й ефективно.

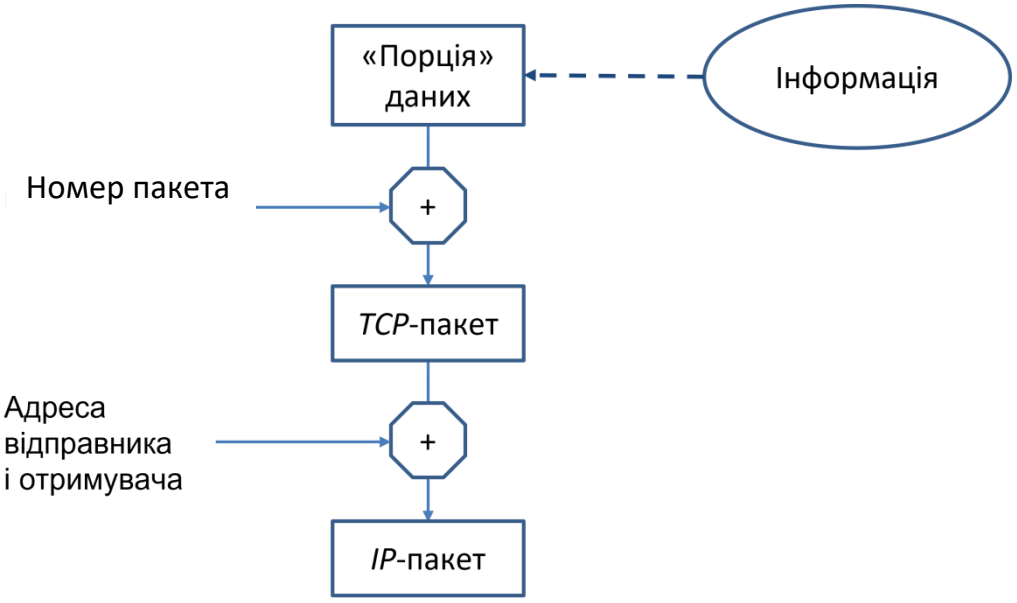


Рис. 1.9

TCP *(Transmission Control Protocol)* – протокол управління передачею.

Протокол ***TCP*** розбиває передану інформацію на «порції» та нумерує їх. За допомогою протоколу ***IP*** ці частини передаються отримувачу. Далі ***TCP*** перевіряє, чи всі частини отримані, а також розміщує їх в потрібному порядку і збирає у єдине ціле. ***TCP/IP*** – це

набір протоколів, які використовуються для передачі даних між комп'ютерами в мережі.

Трансфер пакетів від одного мережного вузла до іншого забезпечується протоколом мережного рівня. Щоб дані передалися від *хоста*-відправника до *хоста*-отримувача, потрібно повідомити *IP*-адресу отримувача та відправника, а також вказати номер порту. Поєднання *IP*-адреси та номера порту, за яким здійснюється передача даних, називають *сокетом*.

Хост – це будь-який пристрій, що працює у форматі «клієнт-сервер» у режимі сервера. У більш широкому розумінні *хостом* може бути будь-який комп'ютер, підключений до локальної чи глобальної мережі.

1.5.3. Маршрутизація даних

Маршрутизація даних – процес визначення маршруту передачі даних. *Маршрутизація* здійснюється маршрутизаторами, які використовуються для отримання *IP*-пакета від одного пристрою та передачі його іншому (рис. 1.10).

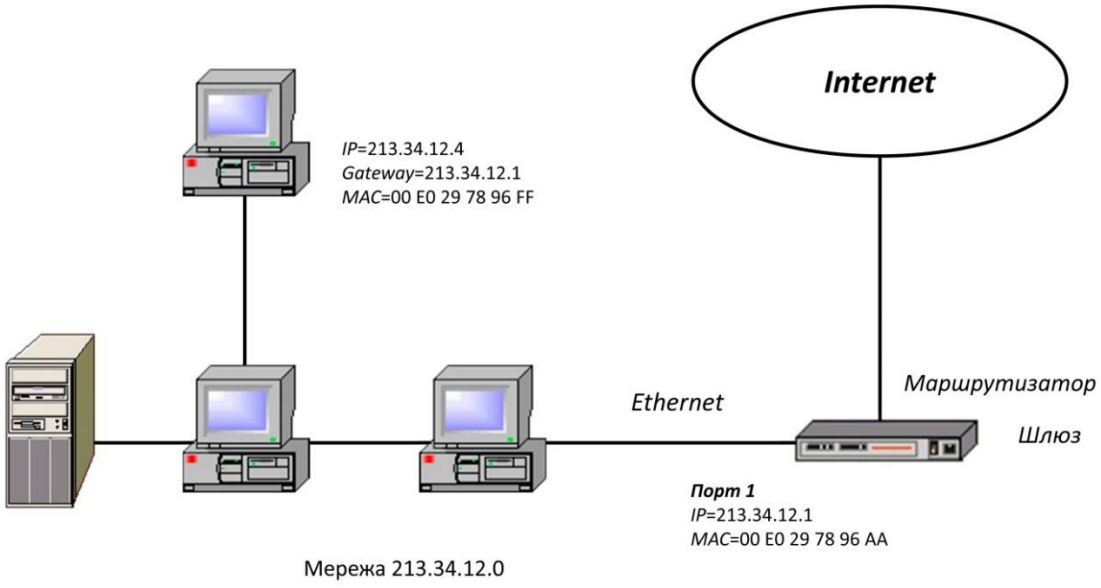


Рис. 1.10

1.5.4. Відтворення доменних імен

Доменне ім'я – символьна адреса *web*-вузла (сайту), що слугує для ідентифікації областей, які є одиницями адміністративної

автономії в мережі *Інтернет*, у складі вищої за ієрархією області. Кожна з таких областей називається *доменом* або *доменною областю*.

Як показано на рис. 1.11, якщо *web*-вузол з назвою домену `ns` перебуває в домені 1-го рівня `com`, його доменне ім'я буде `ns.com`, і це буде означати, що *web*-вузол `ns` розташований у доменній області 1-го рівня `com`. Аналогічно, *web*-вузол з такою ж самою назвою домену `ns` може перебувати в доменній області 1-го рівня `ua`, у такому разі його доменне ім'я буде `ns.ua`, причому ці обидва *web*-вузла будуть незалежні один від одного, лише матимуть спільну назву домену *web*-вузла `ns`.

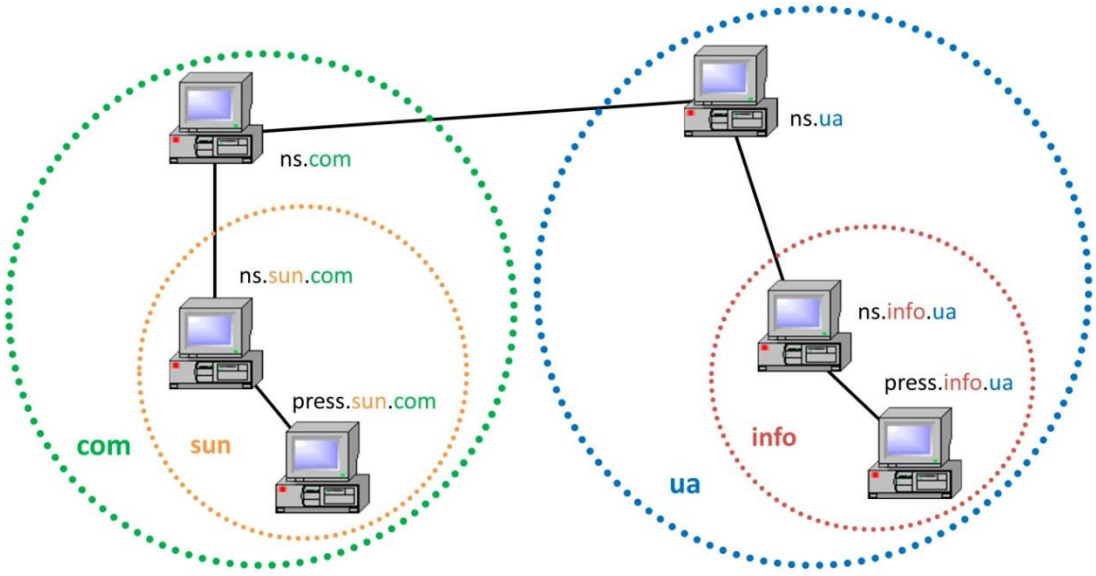


Рис. 1.11

У доменній області 1-го рівня може бути розташована підлегла доменна область, яка вважається доменною областю або доменом 2-го рівня. Наприклад, як показано на рис. 1.11, у доменній області 1-го рівня `com` розташована доменна область `sun`, яка буде доменом 2-го рівня `sun.com`, а в доменній області 1-го рівня `ua` розташована доменна область `info`, яка буде доменом 2-го рівня `info.ua`. Зі свого боку, *web*-вузол, який належить відповідній доменній області 2-го рівня, матиме доменне ім'я, у якому відображена належність до певної доменної області, а саме, як показано на рис. 1.11, `ns.sun.com`, `press.sun.com`, `ns.info.ua`, `press.info.ua`.

1.5.5. Структура доменного імені

Повна адреса сайту складається з декількох частин – протоколу передачі даних і доменного імені (рис. 1.12). Доменне ім’я, зі свого боку, складається з трьох частин у послідовності ієрархії, починаючи з кінця: доменна область, домен сайту, піддомен.

Рис. 1.12

Ієрархія доменних областей зазначається з кінця адресного рядка й починається з домену 1-го рівня, за ним вказується домен наступного підлеглого рівня.

Як показано на рис. 1.12, домен `webnode` належить доменній області 1-го рівня `com`. Якщо б, наприклад, була зазначена доменна область `com.ua`, у такому випадку домен `webnode` належав би доменній області 2-го рівня `com`, яка, зі свого боку, належить доменній області 1-го рівня `ua`.

Якщо *web*-сайт використовує піддомен, його назва вказується ліворуч від домену сайту. Інакше в цій частині може зазначатись акронім `www`, який вказує на те, що вузол `webnode` є кореневим цього *web*-сайту.

Контрольні запитання

1. Що називається комп’ютерною мережею?
2. Що таке мережеве обладнання?
3. Що таке мережеві програмні засоби?
4. Що називають мультипроцесорною системою?

5. Назвіть рівні архітектури комп’ютерних мереж.
6. Що таке локальна комп’ютерна мережа?
7. Що таке глобальна комп’ютерна мережа?
8. Які ви знаєте мережеві топології?
9. Що таке телекомунікація?
10. Що належить до засобів телекомунікації?
11. Що таке телекомунікаційна мережа?
12. З яких організацій складається глобальна мережа Інтернет?
13. Що таке автономна система?
14. Які ви знаєте способи адресації в мережі Інтернет?
15. Як відбувається формування IP-пакета даних?
16. Що таке маршрутизатор, шлюз, міст?
17. Що таке хост?
18. Як здійснюється маршрутизація даних?
19. Що таке доменне ім’я?
20. Із чого складається структура доменного імені?

Лекція 2

ДОСТУП ДО МЕРЕЖІ INTERNET, СЛУЖБИ INTERNET

2.1. Доступ до Internet

Поставкою послуг мережі *Інтернет* займається *провайдер*.

Провайдер – організація, яка надає доступ до мережі *Internet* іншим організаціям або приватним особам. У загальному випадку структура доступу до мережі має вигляд, як показано на рис. 2.1.

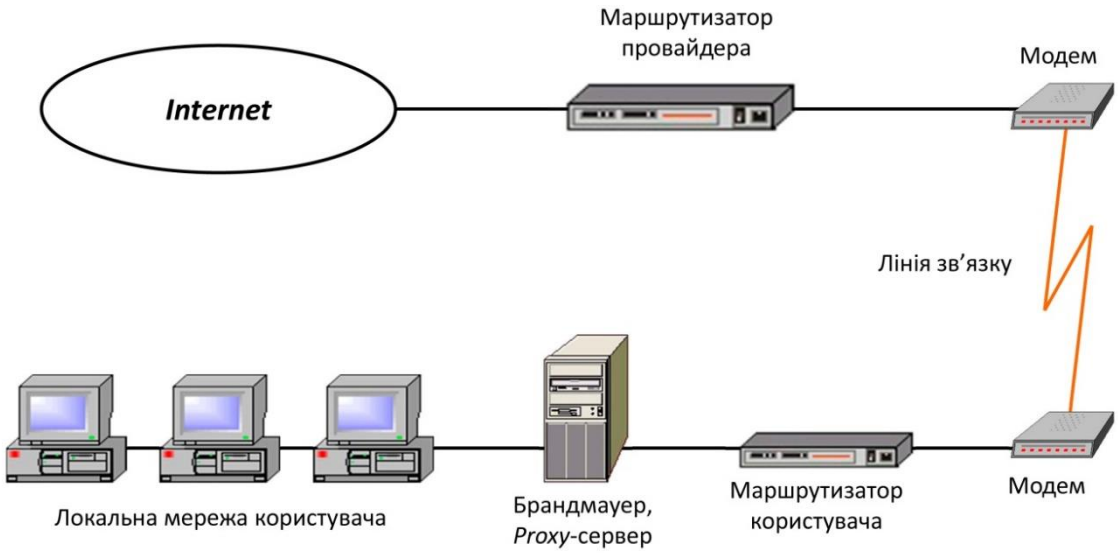


Рис. 2.1

Дані з мережі *Інтернет* передаються через маршрутизатор провайдера, далі через модем провайдера, далі по лінії зв'язку до модему користувача, далі через маршрутизатор користувача до комп'ютерів локальної мережі користувача. На шляху може бути встановлений *брандмауер* та *проксі-сервер*.

Брандмауер – система, яка з метою захисту створює кордон між мережами, застерігаючи від несанкціонованого потрапляння до мережі непотрібних пакетів даних.

Проксі-сервер – сервер, який включається між *локальною мережею* та мережею *Internet*.

Призначення проксі-сервера: кешування *web*-документів для групи комп'ютерів; виконання контролюючих, дозвільних та облікових функцій.

Лінія зв'язку – лінія, що постійно з'єднує абонентів між собою.

2.2. Способи підключення до Internet

З використанням різних технологій доступу до мережі *Internet*, що постійно розвиваються та вдосконалюються, застосовують різні способи підключення. Ці способи залежать від багатьох факторів, головними з яких є швидкість передачі даних і стабільність сигналу.

2.2.1. Комутоване підключення через АТС

Комутативне з'єднання не потребує спеціальної лінії зв'язку і може використовувати телефонні лінії (рис. 2.2). Комутацію виконує АТС (автоматична телефонна станція). Залежно від сигналу розрізняють аналогові й цифрові телефонні лінії. Для передачі цифрової інформації використовується спеціальний пристрій – *модем* (**мо**дулятор **дем**одулятор).

Рис. 2.2

Модем – це пристрій, що застосовується в системах зв'язку для фізичного сполучення інформаційного сигналу із середовищем його поширення, у якому він не може існувати без адаптації.

2.2.2. Комутоване підключення через ISDN

ISDN *(Integrated Services Digital Network – цифрова мережа інтегрованих послуг)* – технологія одночасної передачі цифрових та аналогових даних із використанням однієї лінії зв'язку шляхом розподілення та відокремлення частот передачі (рис. 2.3). Умовою є забезпечення безперервного використання сигналів, що використовують аналоговий зв'язок, із сигналами, що використовують цифровий зв'язок.

Основне призначення *ISDN* – передача даних дротовою лінією і забезпечення інтегрованих телекомунікаційних послуг.

ISDN використовує телефонні дроти. Це має такі переваги: вони вже існують і можуть використатися для подачі живлення на термінальне обладнання.

Рис. 2.3

Адаптер – це пристрій, який перетворює сигнал формату одного електричного пристрою або системи на сигнал формату іншого несумісного пристрою або системи.

2.2.3. Підключення через Frame Relay

Frame Relay – технологія передачі даних із комутацією пакетів у загальній мережі (рис. 2.4).

Рис. 2.4

Робота *Frame Relay* заснована на надсиланні пакетів даних, які називаються «фреймами». Ці *фрейми* містять інформацію про адреси джерела й отримувача, а також інформацію про дані, які мають бути передані. Замість встановлення виділеного каналу між двома точками *Frame Relay* використовує віртуальні канали для надсилання пакетів даних. *Frame Relay* забезпечує гарантовану пропускну здатність і якість обслуговування. Крім того, *Frame Relay* є економічно ефективною технологією, оскільки вона використовує виділені лінії замість комутованих каналів, що може призвести до значної економії витрат на підключення.

2.2.4. Підключення за технологією DSL

DSL *(Digital Subscriber Line – цифрова абонентська лінія)* – технологія високошвидкісного *Інтернету*, яка дає змогу передавати цифрові дані по дротах телефонної мережі (рис. 2.5). *DSL* не заважає телефонній лінії. Та сама лінія може використовуватися як для *Інтернету*, так і для звичайних телефонних служб.

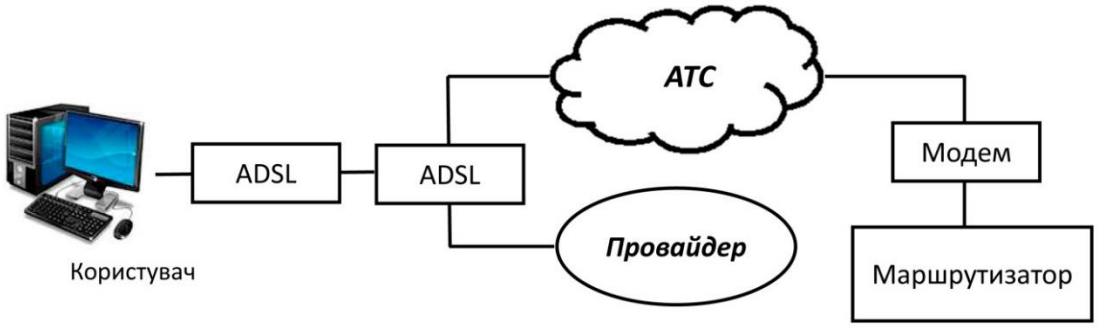


Рис. 2.5

Найпопулярнішою реалізацією *DSL* сьогодні є асиметрична цифрова абонентська лінія *ADSL*.

2.2.5. Підключення по виділеній лінії

Підключення по виділеній лінії – це доступ до мережі *Інтернет* за допомогою фізичного каналу зв'язку у вигляді дротової лінії (рис. 2.6) або у вигляді радіорелейного каналу чи ізольованої смуги частот.

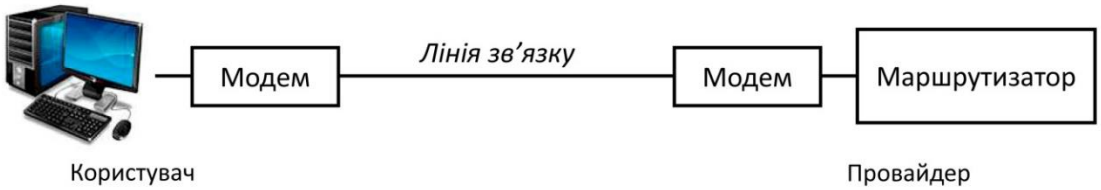


Рис. 2.6

Кінцеве обладнання, що використовується з боку абонента, залежить від конкретного типу використовуваної виділеної лінії.

2.2.6. Підключення за технологіями мобільного зв'язку

Мобільний інтернет – це технологія підключення до мережі *Інтернет* із використанням мереж операторів мобільного зв'язку.

Така технологія надає користувачеві свободу доступу до *глобальної мережі* з різних місць, де є покриття мобільного оператора.

Для встановлення зв'язку з мобільними пристроями використовуються *базові станції* (рис. 2.7).

Рис. 2.7

Базова станція – це системний комплекс апаратури для прийому та передачі сигналу, який здійснює централізоване обслуговування групи кінцевих споживачів. У разі потреби вона здійснює ретрансляцію сигналу, а її оператор контролює ситуацію в ефірі.

Для доступу до мережі *Internet* використовуються технології *мобільного зв'язку* різних поколінь: *2G*, *3G*, *4G*.

Покоління мобільного зв'язку *2G* використовує технологію *GSM*.

GSM *(Global System for Mobile)* – глобальна система мобільного зв'язку, технологія для мобільного цифрового зв'язку з розділенням каналу та високим рівнем безпеки завдяки шифруванню з ключем.

Покоління мобільного зв'язку *3G* використовує технології *UMTS* та *HSPA*.

UMTS *(Universal Mobile Telecommunications System)* – універсальна мобільна телекомунікаційна система, технологія

мобільного зв’язку, що розроблена Європейським інститутом телекомунікаційних стандартів для впровадження *3G* у Європі. Як спосіб передачі даних використовується технологія *W-CDMA*.

W-CDMA *(Wideband Code Division Multiple Access)* – широкосмуговий багаторазовий доступ із кодовим розподілом, технологія радіоінтерфейсу, що використовує широкосмуговий доступ із кодовим розподілом каналів.

HSPA *(High Speed Packet data Access)* – система високошвидкісного пакетного доступу, технологія, яка забезпечує швидкий канал передачі та приймання даних із мобільного пристрою. *HSPA* поділяється на *HSUPA (High-Speed Uplink Packet Access)* для високошвидкісного каналу передачі даних із мобільного пристрою до мережі та *HSDPA (High-Speed Downlink Packet Access)* для високошвидкісного приймання даних із мережі на мобільний пристрій.

Покоління мобільного зв’язку *4G* використовує технологію *LTE*.

LTE *(Long Term Evolution)* – «довготерміновий розвиток», технологія мобільного зв’язку, яка є вдосконаленням технології *UMTS* з метою задоволення потреб у швидкості.

2.3. Wi-Fi-мережі

Останнім часом особливу популярність для підключення до мереж набули мережі *Wi-Fi*, де передача даних здійснюється радіоканалом від *Wi-Fi* маршрутизатора до мобільного пристрою користувача. Сучасні *Wi-Fi* мережі забезпечують високу швидкість обміну даними, гарну захищеність від несанкціонованого доступу та низку різних параметрів налаштування.

Wi-Fi є технологією бездротової локальної мережі. Під абревіатурою *Wi-Fi (Wireless Fidelity)* на цей час розвивається ціла низка стандартів передачі цифрових потоків даних по радіоканалах.

Wi-Fi-мережа – це бездротова мережа, яка дає змогу підключатися до *Інтернету* бездротовим пристроям за технологією *WLAN*.

***WLAN** (Wireless LAN)* – це бездротова локальна мережа, у якій мобільний користувач підключається до локальної мережі *LAN* за допомогою бездротового з'єднання (рис. 2.8).

Рис. 2.8

Переваги *WLAN*:

- гнучкість розгортання,
- мобільність робочого місця,
- швидкий монтаж мережі,
- легкість використання.

Недоліки *WLAN*:

- недостатньо стійка якість передачі даних,
- невеликий радіус дії,
- недостатня захищеність мережі.

2.4. Служби Internet

Служби ***Internet*** – це системи, які надають послуги користувачам *Інтернету,* а також інші продукти, які використовують *Інтернет* як середовище передачі інформації.

Послуги, що надаються *Інтернетом*, можна поділити на дві основні категорії.

1. Відкладені *(off-line)* – основною ознакою цієї групи є наявність тимчасової перерви між запитом та отриманням інформації.

2. Прямі *(on-line)* – для них характерно те, що інформація на запит повертається негайно. Якщо від одержувача інформації потрібна негайна реакція на неї, то така послуга має інтерактивний характер.

До служб мережі *Інтернет* належать:

- служба *FTP*,
- служба *WWW*,
- служба *E-Mail*,
- служби телеконференцій,
- служби інтерактивного спілкування,
- служби пошуку інформації.

2.4.1. Служба FTP

Служба ***FTP*** – це засіб, що дає змогу здійснювати операції з файлами документів і програм в мережі *Internet* за протоколом ***FTP***.

FTP *(File Transfer Protocol – протокол передачі файлів)* – протокол, призначений для передачі файлів у комп'ютерних мережах. Протокол ***FTP*** дає змогу підключатися до серверів ***FTP***, переглядати зміст каталогів та здійснювати операції над файлами на сервері (копіювання, переміщення, видалення тощо).

Сервер ***FTP*** є сховищем файлів. Ці файли користувач може прочитати або скопіювати. *DNS*-адреси таких серверів починаються з *ftp* (наприклад, *ftp.microsoft.com)*. Інформація на *FTP*-серверах організована у вигляді традиційних каталогів.

Вузли *FTP*-служби використовуються розробниками програмного забезпечення для його поширення.

Префікс протоколу *ftp://*, структура адреси:

ftp://ім'я_користувача:пароль@доменне_ім'я.

Приклад:

ftp://name: password@ftp.microsoft.com/services/.

На рис. 2.9 показано схему взаємодії користувача з *FTP*-сервером. Коли користувач підключається до *FTP*-сервера, встановлюється *керуюче з'єднання*.

Керуюче з'єднання встановлюється як з'єднання клієнт-сервер і існує весь час, поки клієнт працює із сервером. *Керуюче з'єднання* використовується для передачі команд від клієнта до сервера й отримання відгуків від нього.

Рис. 2.9

У той момент, коли користувач здійснює передачу даних, встановлюється *інформаційне з'єднання*.

Інформаційне з'єднання відкривається щоразу, коли здійснюється передача файлу між клієнтом і сервером. Після закінчення передачі-отримання файлу *інформаційне з'єднання* закривається.

2.4.2. Служба WWW

WWW *(World Wide Web – Всесвітнє павутиння)* – це система організації інформації в *Internet*, єдиний інформаційний простір, який складається із сотень мільйонів взаємозв'язаних електронних документів, що зберігаються на *web-серверах*. Окремі документи, що становлять *web*-простір, мають назву *web-сторінки*. Групи тематичних *web-сторінок* мають назву *web-вузли* (*web-сайти* або просто *сайти*). Один фізичний *web-сервер* може містити достатньо багато *web-вузлів*, кожному з яких відводиться окремий каталог на жорсткому диску сервера.

WWW поєднує в собі безліч можливостей (наприклад, передачу текстів, графічних зображень, звуків, відео). Усі інформаційні об'єкти зв'язуються єдиною структурою, в основу якої покладено поняття *гіпертексту*.

Служба ***WWW*** використовує протокол під назвою *HTTP*.

HTTP *(Hyper Text Transfer Protocol – протокол передачі гіпертексту)* – мережевий протокол прикладного рівня для передачі інформації. Основним призначенням ***HTTP*** є передача *web-сторінок*.

Служба ***WWW*** працює за принципом *клієнт-сервер*. Існує велика кількість серверів, які за запитом клієнта надають йому *гіпермедійний документ*.

Гіпермедійний документ – це документ, що складається із частин з різним представленням інформації (текст, звук, графіка, тривимірні об'єкти тощо).

У *гіпермедійному документі* кожний елемент може бути *гіперпосиланням* на інший документ чи його частину. Такі посилання організовані таким чином, що кожний інформаційний ресурс у *глобальній мережі Internet* однозначно адресується, і документ, який зчитується в цей момент, може посилатися як на інші документи на цьому ж сервері, так і на документи, які розташовані на інших комп'ютерах в мережі *Internet*.

Гіперпосилання – це «активний» текст, зображення чи кнопка на *web-сторінці*, натиснення на яку ініціює перехід на іншу сторінку чи іншу частину поточної сторінки.

Адреса, за якою відбувається перехід на той чи інший *web-ресурс*, має таку структуру:

http://www.доменне_ім'я.

Приклад:

http://www. google.com.

Перегляд *web-сторінок* і робота з ними здійснюється за допомогою *браузерів*.

Браузер *(Browser)* – це програма для інтерактивного перегляду *гіпертекстових документів*, надає можливість здійснювати

навігацію по каталогах інформаційних ресурсів *web-сайтів* мережі *Internet*.

На рис. 2.10 показано схему взаємодії користувача з *web*-сервером. Користувач за допомогою *web*-браузера строює і надсилає *http*-запит, який засобами служби *WWW* оброблюється й адресується до певного *web*-сервера.

Рис. 2.10

Отриманий *web*-сервером *http*-запит виконується, після чого результат виконання надсилається клієнту у вигляді *http*-відповіді, яка, зі свого боку, оброблюється на комп'ютері користувача *web*-браузером і відображається у вікні браузера у вигляді кінцевого результату як *web*-сторінка.

HTTP-запит зазвичай створюється в рядку адреси вікна браузера або може бути адресою гіперпосилання на *web*-сторінці та має певну структуру (рис. 2.11).

Рис. 2.11

Цей запит є ідентифікатором ресурсу *URI*, який складається безпосередньо з адреси ресурсу в мережі *URL*, після якої у разі потреби зазначається перелік змінних та їх значень у вигляді рядка даних `datastring`. Частиною структури *URL* є *URN*.

URI *(Universal Resource Identifier – універсальний ідентифікатор ресурсу)* – символьний рядок, що ідентифікує ресурс у мережі *Інтернет* із передачею до нього рядку даних запиту.

URL *(Uniform Resource Locator – уніфікований покажчик ресурсу)* – адреса ресурсу в мережі.

URN *(Uniform Resource Name – уніфікована назва ресурсу)* – частина *URL*, що безпосередньо містить назву ресурсу в мережі *Інтернет*.

2.4.3. Служба E-Mail

Служба ***E-Mail*** *(Electronic mail – електронна пошта)* – сервіс відкладеного зчитування *(off-line)*. Після відправлення повідомлення (зазвичай у вигляді звичайного тексту) адресат отримує його на свій комп'ютер через деякий період часу й ознайомлюється з ним, коли йому зручно.

Електронна пошта – універсальний сервіс: безліч мереж у всьому світі, побудованих на зовсім різних принципах й протоколах, можуть обмінюватися електронними листами через *Internet*, отримуючи тим самим доступ до інших його ресурсів. Практично всі сервіси *Internet*, які використовуються як сервіси прямого доступу (*on-line*), мають інтерфейс до *електронної пошти*. Користувач, не маючи доступу до інформації, що зберігається в *Internet* в режимі *on-*

line, може отримувати більшу її частину за допомогою електронної пошти.

Структура адреси:

логін@доменне_ім'я_сервера.

Приклад:

name@gmail.com.

Для роботи з *електронною поштою* використовуються протоколи ***SMTP***, ***POP*** та ***IMAP***.

SMTP *(Simple Mail Transfer Protocol – простий протокол пересилання пошти)* – це протокол, який використовується для пересилання електронної пошти до поштового сервера або з комп'ютера-клієнта, або між поштовими серверами.

POP *(Post Office Protocol – поштовий офісний протокол)* – це протокол, що використовується клієнтом для доступу до повідомлень електронної пошти на сервері.

IMAP *(Internet Message Access Protocol – протокол доступу до інтернет-повідомлень)* – це протокол, що використовується клієнтом для доступу й керування повідомленнями електронної пошти на сервері.

За своєю функціональністю *IMAP* є набагато ширшим за *POP*. *IMAP* не розрахований на відсилання пошти, ця функція делегована іншим протоколам, зокрема *SMTP*.

Електронна пошта є найпоширенішим видом мережевих послуг. За допомогою спеціальної програми *(поштового клієнта)* користувач створює електронний лист і надсилає його іншому абоненту.

Поштовий клієнт – це програма для роботи з поштовим сервером, що дає можливість надсилати й отримувати електронну пошту.

Схему взаємодії користувачів служби *E-Mail* показано на рис. 2.12.

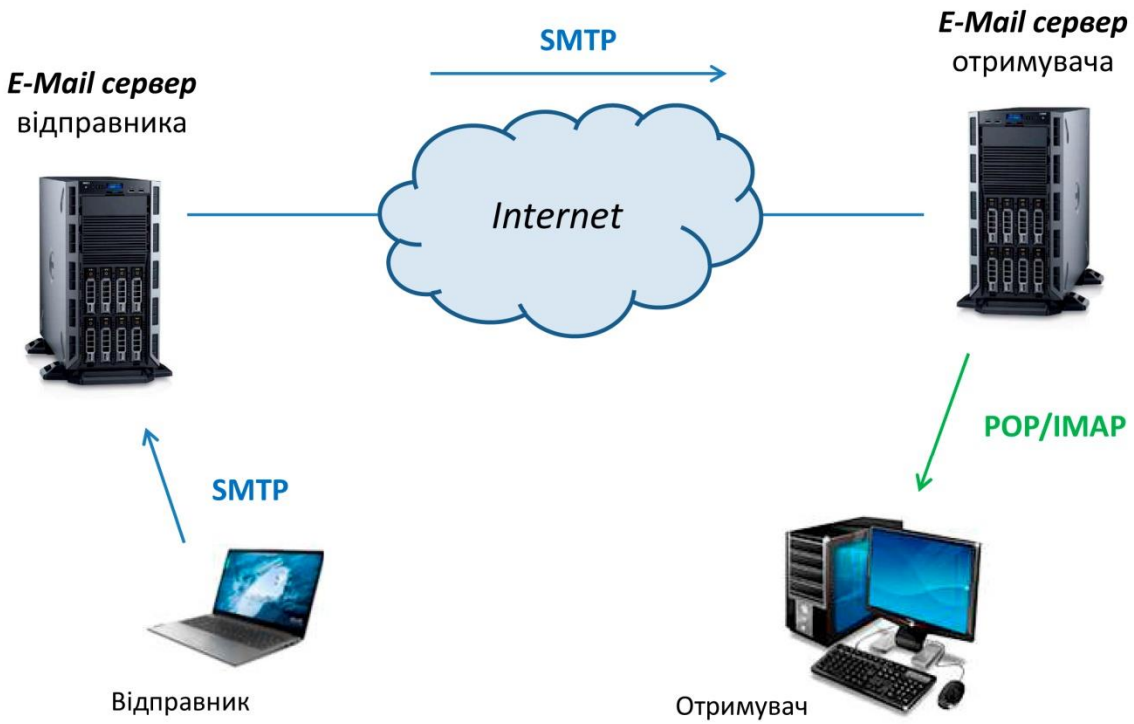


Рис. 2.12

2.4.4. Служби телеконференцій

Телеконференції – друга за поширеністю служба *Інтернету*, яка надає відкладені послуги *(off-line)*.

Служба ***телеконференцій*** складається з багатьох тематичних ***телеконференцій*** – *груп новин*, що підтримуються *серверами новин*.

Сервер новин – це комп'ютер, який може містити тисячі груп новин найрізноманітніших тем. Кожен *сервер новин*, який отримав нове повідомлення, передає його всім вузлам, з якими обмінюється новинами.

Група новин – це набір повідомлень з певної теми. Новини розділені за ієрархічно організованими тематичними групами, ім'я кожної групи складається з імен підрівнів.

У наш час широкого поширення набули *web-телеконференції*, які мають назву *форум* або *web-форум*. ***Форум*** працює через *web-інтерфейс* і розміщується не централізовано на *сервері новин*, а на сервері поточного *web-сайту*.

Окремим видом *телеконференцій* є *відеоконференції*, які, на відміну від форумів чи груп новин, працюють у режимі прямого доступу *(on-line)*.

Відеоконференція – це сеанс зв'язку, що передбачає аудіовізуальний контакт учасників із використанням спеціалізованого *програмного забезпечення* та обладнання.

2.4.5. Служби інтерактивного спілкування

Служби ***інтерактивного спілкування*** призначені для прямого *(on-line)* обміну повідомленнями через *Інтернет* між користувачами в реальному часі.

До служб *інтерактивного спілкування* належать, наприклад, *ICQ, Skype, Viber, Telegram, WhatsApp* та інші.

Більшість служб ***інтерактивного спілкування*** побудовано за *клієнт-серверною* технологією.

Для обміну повідомленнями користувачів у реальному часі з використанням служб *інтерактивного спілкування* потрібно, щоб обидва користувачі були зареєстровані в тій самій службі.

Для спілкування групи користувачів у реальному часі використовується форма спілкування під назвою ***чат.***

Чат *(chat – бесіда)* – служба *Інтернету*, що дає можливість здійснювати текстові дискусії в режимі реального часу. Найпопулярнішим відкритим стандартом, що лежить в основі чатів, є ***IRC*** *(Internet Relay Chat – бесіда через Internet).*

У свій час чати, в основі яких лежав стандарт *IRC*, набули досить широкого поширення. Однак сьогодні все більш популярними є чати, що організуються в межах окремих *web-сайтів* із вбудованим у їх інтерфейс функціоналом. Це дає змогу користувачам спілкуватися без встановлення додаткового програмного забезпечення.

2.4.6. Служби пошуку інформації

Служби ***пошуку інформації,*** або ***пошукові системи****, – онлайн-*служби, що надають можливість пошуку інформації в *Інтернеті*.

Пошукова система – спеціальний сервер, що постійно досліджує *Інтернет*, аналізуючи його певні сегменти з метою збору

та збереження даних про *web-сайти* за допомогою *каталогів* та *індексів*.

Каталоги *пошукової системи* організовані у вигляді ієрархії розділів та підрозділів.

Індекси *пошукової системи* автоматично формуються за допомогою потужних комп'ютерів, що постійно сканують *Інтернет* і дають змогу користувачам виконувати пошук інформації за ключовими словами.

Індекс *пошукової системи* – структуровані дані, що зберігаються на сервері пошукової системи у вигляді стислої копії вихідної *web-сторінки* та посилання на неї.

Введення пошукового запиту здійснюється через *web-інтерфейс* браузера. Сформований запит надсилається на сервер пошукової системи, де за ним із каталогів баз даних здійснюється вибір шуканої інформації, після чого результат запиту надсилається назад на комп'ютер клієнта й відображається у вікні браузера у вигляді *web-сторінки* зі списком результатів пошуку.

У наш час існує велика кількість *пошукових систем*, але найпопулярнішими з них є *Google*, *Bing*, *Yahoo*!

Схему роботи *пошукової системи* показано на рис. 2.13. Користувач на *web*-сторінці *пошукової системи* створює запит і відправляє його натисканням кнопки «Знайти». *Пошукова система* обробляє синтаксис запиту та зіставляє ключові слова зі словами в індексі, порівнює список сайтів, які містять ключові слова, ранжує список за релевантністю, відправляє результат користувачу.

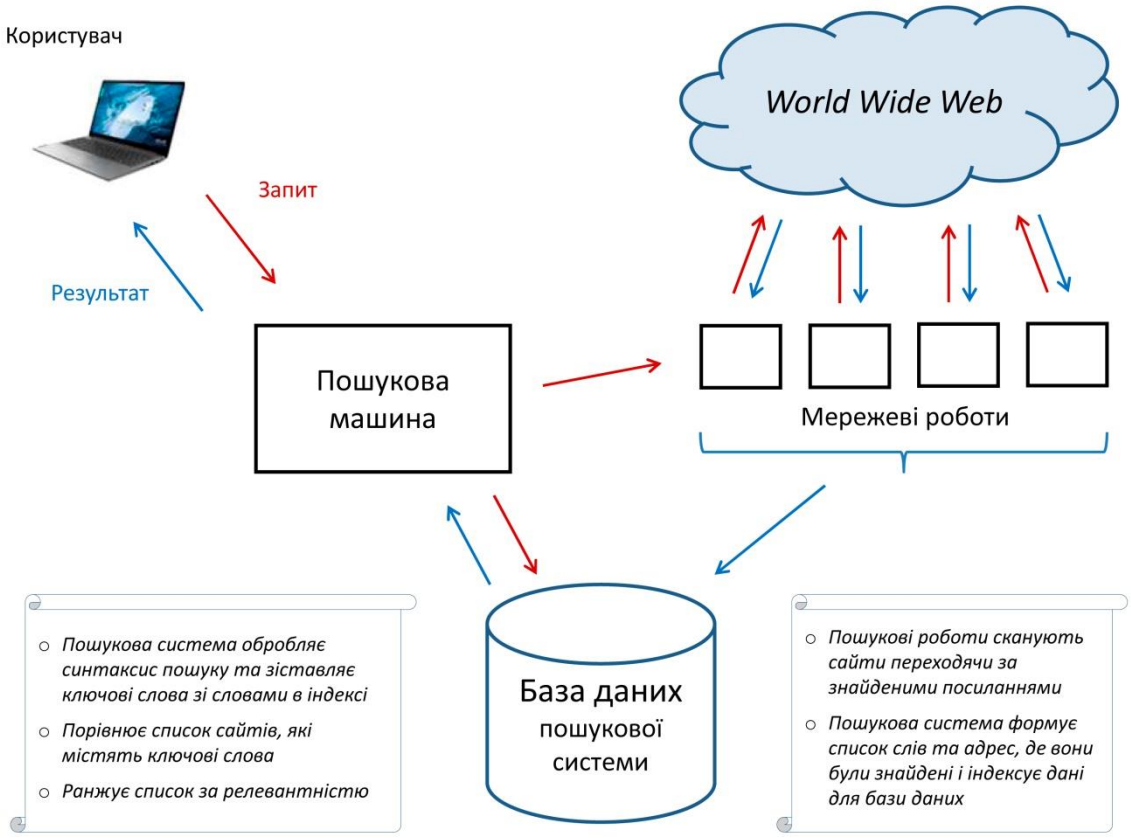


Рис. 2.13

До складу пошукової системи входять: *пошукова машина*, *мережеві роботи*, *база даних пошукової системи*.

Пошукова машина – програмно-апаратний комплекс, який складається із сервера, що забезпечує функціональність пошукової системи, здійснює автоматизацію пошуку в комп'ютерній мережі й паралельно виконує оперативне оновлення інформації у своїй базі даних без участі людей.

Мережеві роботи (або *пошукові роботи*, або *спайдери*) – це комп'ютери зі спеціальним програмним забезпеченням, що здійснюють загальний пошук інформації в мережі *Інтернет*.

Пошукові роботи повідомляють про зміст знайденого документа, індексують його і добувають підсумкову інформацію. Вони також переглядають заголовки *web-сторінок*, посилання і відправляють проіндексовану інформацію до бази даних пошукової системи, сканують сайти переходячи за знайденими посиланнями, далі *пошукова система* формує список слів та адрес, де вони були знайдені, індексує дані для *бази даних пошукової системи*.

База даних пошукової системи – це сховище проіндексованої і упорядкованої каталогами інформації.

Контрольні запитання

1. Хто такий провайдер?
2. Що таке брандмауер?
3. Що таке проксі-сервер?
4. Які ви знаєте способи підключення до мережі Інтернет?
5. Що таке модем?
6. Що є мобільним інтернетом?
7. Що таке базова станція?
8. Що таке Wi-Fi-мережа?
9. Які Ви знаєте служби мережі Інтернет?
10. Що називається гіпермедійним документом?
11. Що таке гіперпосилання?
12. Що таке браузер?
13. Що таке URI та URL?
14. Що таке поштовий клієнт?
15. Що таке пошукова система?
16. Для чого призначені мережеві роботи?
17. Для чого використовується база даних пошукової системи?

Лекція 3
УПРАВЛІННЯ ІНФОРМАЦІЙНИМИ РЕСУРСАМИ У WEB-ПРОСТОРІ

З розвитком мережі *Internet* і використанням сучасних програмно-апаратних засобів, що забезпечують великі швидкості передачі й обробки даних, а також високий рівень захисту даних під час передачі, поширеним стало застосування *інформаційних систем* із використанням *web*-технологій у мережі *Internet*.

Таке застосування *інформаційних систем* здійснюється з використанням *web*-серверів, баз даних на них, *web*-платформ, систем управління, що встановлені і працюють на комп'ютері-сервері з реалізацією інтерфейсу для забезпечення обміну даними між сервером і користувачем через *web*-браузери, які працюють на комп'ютері-клієнті.

Такий підхід забезпечує можливість вести бізнес і здійснювати керування підприємством чи ресурсами підприємства, а також вести неперервну роботу зі співробітниками та клієнтами в будь-який зручний час, перебуваючи в будь-якому місті, у будь-якій країні, де є доступ до *Internet*, за допомогою стаціонарних комп'ютерів, ноутбуків чи мобільних пристроїв.

Для здійснення такого керування, а також для вдалого ведення справ підприємства потрібно володіти знаннями про сучасні *інтернет-технології*, *інтернет-системи*, інструменти, у яких ці технології реалізовані, їх різновиди та призначення для застосування, уміти ефективно вибирати необхідні програмні засоби для вирішення певних задач або виконавців, які професійно володіють знаннями у своїй спеціалізованій галузі та допоможуть швидко і якісно виконати поставлені перед ними завдання.

Управління інформаційними ресурсами – процес цілеспрямованого, економічно обґрунтованого формування й перетворення *інформаційних ресурсів* засобами *інформаційних систем* і *технологій*.

Інформаційний ресурс – організована сукупність документованої інформації, що містить бази даних та інші масиви

інформації в *інформаційних системах*.

Інформаційна система – це комплекс програмно-технічних засобів, що здійснюють процеси перетворення *інформаційних ресурсів* методами, способами і прийомами *інформаційних технологій*.

Інформаційна технологія – це сукупність методів, способів, прийомів і засобів доступу, збору, збереження, обробки й передачі інформаційних ресурсів, включно з прикладними програмними засобами. Мета *інформаційної технології* – генерування людиною інформації для аналізу і прийняття на її основі рішення з виконанням будь-якої дії.

У наш час у *глобальній мережі Internet* перебуває великий обсяг інформації, що являє собою *інформаційні ресурси* з різних напрямів діяльності. Основний обсяг *інформаційних ресурсів* у вигляді *web-сторінок*, *файлів*, *баз даних*, розташовано на *web-сайтах*, розміщених на *web-серверах Інтернету*. Управління цими ресурсами, а також передача, обмін, обробка, пошук здійснюються за допомогою систем управління та інших *Internet*-систем.

3.1. Сайти та Internet-системи

Web-сайт – це сукупність програмних, інформаційних, а також медійних засобів, логічно пов’язаних між собою, доступних користувачам за допомогою мережі *Internet*.

Професійно розроблений сайт може слугувати як високоефективним інструментом ведення бізнесу, так і інформаційним або іміджевим ресурсом, який розповідає про діяльність компанії.

Класифікація сайтів та їх ділення за типами є досить умовним. Бажання класифікувати сайти пояснюється прагненням розробника пояснити клієнту, що ж вийде в підсумку. Для того щоб визначити, який саме тип сайту потрібний компанії, мало використовувати загальні рекомендації. Лише чітке визначення цілей і завдань сайту, що планується, дасть чітке розуміння того, яким він повинен бути і на яку аудиторію буде розрахований. Отже, перед тим як почати

розробку *web-сайту* для свого підприємства, потрібно зрозуміти, який саме тип сайту потрібен.

Взагалі сайти можна поділити на чотири основні типи: *інформаційні сайти, корпоративні сайти, інтернет-магазини, спеціалізовані сайти*.

Інформаційні сайти – це сайти, які надають найбільш важливу інформацію про компанію в Інтернеті, її товари та послуги, зі швидким і якісним інформуванням користувача. Інформаційний сайт може представляти не тільки компанію, що займається комерційною діяльністю, а й громадську організацію або окрему людину.

Корпоративні сайти – це сайти, які виконують функції як сайту-візитки, так і електронного магазину або каталогу. Такі сайти зазвичай містять новини, інформаційні сторінки, каталог товарів або послуг, форму заявки або форму для зв'язку. Корпоративні сайти можуть містити системи, які дають змогу взаємодіяти з партнерами. Цей тип сайтів підходить середнім і великим компаніям, що бажають розширити сфери своєї діяльності.

Інтернет-магазини (електронні магазини) – електронні каталоги товарів із можливістю здійснення їх замовлення або покупки через мережу Інтернет. Це є ресурси, що спрямовані на продаж товарів в Інтернеті.

Окрему групу становлять *спеціалізовані сайти*, до яких можна віднести: портали новин, електронні бібліотеки, різноманітні сервіси для зберігання інформації (аудіо, відео), кінозали, соціальні мережі тощо.

Портали новин – це сайти, на яких у хронологічному порядку представлена інформація у вигляді новин і статей на різні теми (політика, бізнес, освіта, культура, спорт, охорона здоров'я тощо). На таких сайтах також можуть бути присутні коментарі для обговорювання подій відвідувачами.

Електронні бібліотеки – це сайти, які пропонують відвідувачам електронні книги, журнали й іншу публікацію в

електронному вигляді для безплатного перегляду або скачування в зручний час.

За аналогією з електронними бібліотеками існують сервіси для зберігання різноманітної інформації у вигляді файлів в аудіо- та відеоформатах, графічних або текстових форматах чи у вигляді архівів на різні теми. У них можуть бути представлені різноманітні фільми (документальні, художні, мультиплікаційні), записи концертів, виступів, подій, аудіозаписи пісень у вигляді окремих саундтреків чи альбомів, аудіокниги, фотоальбоми, а також електронні копії чи електронні версії книжок. Уся ця інформація зберігається в упорядкованому за тематикою вигляді.

У наш час велику популярність набули сервіси зберігання відеофільмів у вигляді *віртуальних кінозалів*. На таких сайтах відвідувач в online-режимі може здійснювати перегляд різноманітних фільмів не тільки на персональному комп'ютері чи планшеті, але й за допомогою телевізорів, у яких реалізована технологія *SmartTV*.

У сучасному суспільстві поширеним стало спілкування між людьми через соціальні мережі, особливо коли співбесідники мешкають у різних містах світу.

Соціальні мережі – це *web*-сайти, які призначені для побудови, відображення й організації соціальних взаємовідносин між людьми в Інтернеті. На таких сайтах люди зазвичай шукають нові знайомства, обмінюються новинами, опубліковують власні фото та відеоматеріали та обговорюють їх за допомогою коментарів, ведуть електронні щоденники. Також можуть здійснювати листування чи аудіо- та відеозв'язки, якщо це передбачено інструментарієм такого сайту.

Окрім соціальних мереж для спілкування між людьми, також існують такі сайти, як *форуми, чати, блоги*.

Web-форуми та ***чати*** – це місця в мережі Internet для спілкування між користувачами за інтересами. Вони працюють за дискусійним принципом: користувач створює тему, інші користувачі можуть приєднатися до її обговорення. Чат, на відміну від форуму,

має онлайновий характер спілкування, тоді як на форумі користувач може залишити повідомлення, а відповідь на нього переглянути в будь-який інший час.

Блоги (або *online*-щоденники) – різновид персонального сайту, особистої сторінки в Інтернеті, де розташована стрічка новин у хронологічному порядку однієї людини. Так само, як і на форумах, кожне повідомлення, що залишено користувачем, який веде блог, може бути обговорено у вигляді коментарів іншими користувачами.

Internet-системи – це інформаційні системи, що являють собою програмні комплекси, які призначені для виконання ряду певних задач у *web*-просторі з використанням *web*-технологій. До них належать *системи управління контентом*, *системи online-замовлень і бронювань*, *торговельні системи*, *пошукові системи*, *платіжні системи* тощо.

Системи управління контентом CMS *(Content Management System)* – це інформаційні системи, які використовуються для забезпечення й організації спільного процесу створення, редагування та управління вмістом (контентом) сайту. Основним завданням такої системи є збір і об'єднання у єдине ціле різних джерел інформації. У таких системах управління контентом визначає все різноманіття існуючих даних: стандартні документи, мультимедійний контент, каталоги всілякої інформації тощо.

Системи online-замовлень і бронювань – це системи управління замовленнями, що забезпечують взаємодію між постачальниками товарів і послуг та клієнтами (замовниками, покупцями), які мають широку мережу філій як у межах країни, так і по всьому світу.

Торговельні системи – це системи, що забезпечують можливість здійснення угод, а також підтримку, зберігання, обробку інформації, необхідної для здійснення й виконання угод із цінними паперами та фінансовими інструментами.

Пошукові системи – це складні програмно-апаратні комплекси, що призначені для здійснення пошуку ресурсів в Інтернеті, збереження відомостей про них у своїх базах даних і

надання користувачеві переліку посилань відповідно до його пошукового запиту.

Платіжні системи – це системи розрахунків між фінансовими організаціями та Internet-користувачами за різні послуги через мережу Інтернет.

Також до *Internet*-систем можна віднести *системи управління взаємовідносинами з клієнтами*, *системи управління людськими ресурсами*, *системи планування ресурсів підприємств* у разі, якщо вони реалізовуються на *web*-платформі з організацією зв'язку з користувачами за допомогою *web*-інтерфейсу, що надає можливість доступу до них з будь-якої точки Земної кулі через *Інтернет*.

Система управління взаємовідносинами з клієнтами CRM *(Customer Relationship Management)* – система, яка призначена для автоматизації взаємодії організації із замовниками (клієнтами) з метою підвищення рівня продажу, оптимізації маркетингу й поліпшення обслуговування клієнтів шляхом збереження інформації про клієнтів та історії взаємин з ними, встановлення й поліпшення бізнес-процесів і подальшого аналізу результатів.

Система управління людськими ресурсами HRM *(Human Resources Management)* – система, що спрямована на оптимізацію використання людських ресурсів і забезпечення організацій якісним персоналом, здатним виконувати покладені на нього трудові функції.

Система планування ресурсів підприємств ERP *(Enterprise Resource Planning)* – система, яка реалізує стратегію інтеграції виробництва й операцій управління трудовими ресурсами, фінансового менеджменту та управління активами, орієнтована на безперервне балансування й оптимізацію ресурсів підприємства за допомогою спеціалізованого інтегрованого пакета прикладного програмного забезпечення, що забезпечує загальну модель даних і процесів для всіх сфер діяльності.

3.2. Технологія створення сайтів

Створення сайту – складний і відносно тривалий процес, який протікає в декілька етапів, у міру проходження яких ідея перетворюється в реальний функціонуючий ресурс. У цьому процесі беруть участь різні фахівці, кожен з яких відповідає за свою ділянку робіт.

До початку створення сайту, щоб проєкт був успішним, потрібно визначити, які завдання покладатимуться на сайт, на яку аудиторію він буде розрахований, яка функціональність у нього буде закладена, і насамперед розглянути маркетингову сторону питання.

Зазвичай після маркетингового планування та складання технічного завдання на розробку приступають безпосередньо до створення самого сайту, процес якого складається з таких етапів:

- розробка структури сайту;
- розробка інтерфейсу та дизайну сайту загалом та його сторінок окремо;
- верстка шаблонів сторінок, втілення додаткових скриптів;
- хостинг сайту, реєстрація домену, встановлення сайту на сервер хостинг-провайдера, встановлення та налаштування системи управління контентом;
- створення структури сайту в *CMS*, підключення шаблонів сторінок до структурних елементів, наповнення сайту в *CMS* контентом, у разі потреби – додаткова розробка та підключення необхідних програм (скриптів);
- остаточне налаштування, тестовий запуск;
- тестування, виправлення помилок (у разі потреби);
- остаточний запуск, презентація.

На етапі розробки структури сайту приділяється увага архітектурі майбутнього проєкту: з яких сторінок він складатиметься, як вони будуть пов’язані між собою, з яких елементів структурно складатимуться самі сторінки сайту. Якщо сайт має спеціалізований характер і потребує роботи з окремо

визначеною структурою бази даних, на цьому етапі також розробляється структура додаткових таблиць бази даних і схеми їх взаємодії зі сторінками сайту. Отриманий на цьому етапі результат є основою для розробки дизайну сторінок *web*-дизайнерами та розробки додаткових скриптів у разі потреби *web*-програмістами.

На етапі розробки інтерфейсу й дизайну сайту *web*-дизайнери розробляють ескізи макетів сторінок. Вони визначають, яким чином кінцевий споживач отримуватиме доступ до інформації та послуг сайту, тобто займаються безпосередньо розробкою інтерфейсу користувача. Здебільшого сторінки включають містять графічні елементи. Їх підготовкою займаються художники, ілюстратори, фотографи, технічні дизайнери, шрифтовики та ін. Результатом роботи на етапі розробки дизайну стануть ескізи макетів усіх типових сторінок *web*-сайту, які демонструються замовнику. Якщо замовник задоволений зовнішнім виглядом цих ескізів, настає наступний етап – верстка сторінок сайту.

На етапі верстки здійснюється верстка макетів і підготовка шаблонів сторінок за їх ескізами до підключення до *CMS*. Верстальник на цьому етапі за допомогою *html*-верстки та *css*-стилів перетворює отримані зображення ескізів у *html*-шаблони сторінок, а програмісти в разі потреби втілюють у них скрипти для забезпечення інтерактивності.

Паралельно з процесом верстки й підготовки шаблонів сторінок здійснюється хостинг сайту, реєстрація та отримання домену для нього, встановлення та налаштування сайту на сервері хостинг-провайдера, а також встановлення та налаштування *CMS*. Якщо сайт встановлюється на власному сервері, потрібно також здійснити налаштування самого сервера та його програмного забезпечення.

На наступному етапі, після встановлення та налаштування *CMS*, уже безпосередньо в ній, а саме в панелі адміністратора системи управління, здійснюється побудова структури сайту, підключення шаблонів і наповнення сайту контентом. На цьому етапі відбувається завершальна фаза створення сайту, після якої

виконується остаточне налаштування необхідних параметрів і тестовий запуск.

Одним із найважливіших етапів, після якого здійснюється остаточний запуск і опублікування сайту, є його тестування. Цей етап передбачає перевірку міжбраузерності (тобто чи у всіх браузерах однаково коректно відображається інформація), перевірку злагодженості та чіткості роботи програмних компонентів, тестування зручності користування сайтом, інтерфейсу користувача, тестування продуктивності за великих навантажень, а також тестування сайту на уразливість. Якщо сайт є *інтернет*-магазином або використовує взаємодію з платіжними системами, обов'язково здійснюється тестування процесу оплати, транзакцій і складання звітів. У разі виникнення помилок вони виправляються і здійснюється повторне тестування доти, доки сайт не запрацює бездоганно.

Останнім етапом (і самим приємним у всьому процесі) є остаточний запуск створеного сайту і його презентація. Однак у зв'язку з тим, що в сучасному *web*-просторі існує безліч різноманітних сайтів, для того щоб він набув популярності й відвідуваності, потрібно здійснити його просування та розкрутку. *Просування сайту* – це комплекс заходів щодо забезпечення відвідуваності сайту цільовою аудиторією. Здійснюється це за допомогою інструментів пошукової оптимізації *SEO*.

3.3. SEO-оптимізація

SEO *(Search Engine Optimization – пошукова оптимізація)* – комплекс заходів щодо внутрішньої та зовнішньої оптимізації для підвищення позицій сайтів в результатах видачі результатів запиту пошукових систем за певними запитами користувачів із метою збільшення мережевого трафіку, потенційних клієнтів і подальшої монетизації цього трафіку.

Зазвичай чим вища позиція сайту в результатах пошуку, тим більше зацікавлених відвідувачів переходить на нього з пошукових систем.

Під час аналізу ефективності пошукової оптимізації оцінюється показник *конверсії сайту*.

Конверсія сайту – це відношення числа відвідувачів, які виконали на сайті будь-які цільові дії, до загальної кількості відвідувачів сайту, що виражене у відсотках.

Вдала конверсія по-різному трактується продавцями, рекламодавцями та постачальниками контенту.

Для продавця успішна конверсія означає операцію купівлі. Для постачальника контенту успішною конверсією може бути реєстрація відвідувачів на сайті, а також підписка на розсилку або завантаження програмного забезпечення.

Пошукові системи враховують безліч внутрішніх і зовнішніх параметрів сайту для обчислення його *релевантності* (ступеня відповідності введеному запиту):

- ***щільність ключових слів*** – складні алгоритми сучасних пошукових систем дають змогу проводити семантичний аналіз тексту, щоб відсіяти пошуковий спам, у якому ключове слово зустрічається занадто часто;
- ***індекс цитування*** – залежить від кількості і авторитетності *web*-ресурсів, що посилаються на цей сайт;
- ***«водність» тексту*** – показник, що визначає наявність слів, які не несуть жодної корисної інформації та слугують для розведення тексту;
- ***внутрішні фактори поведінки*** – низка різноманітних дій користувачів, які вони можуть зробити на сайті: вхід, загальний час відвідування сайту, кількість сесій одного користувача на сайті, кількість переглянутих сторінок, факт повернення на сайт, переходи за посиланнями в тексті та за посиланнями в меню;
- ***зовнішні фактори поведінки*** – основним зовнішнім показником якості поведінки користувача під час взаємодії із сайтом є відмова від подальшого пошуку за ключовою фразою у пошуковій системі;
- ***індекс якості сайту*** – показник того, наскільки корисний певний сайт користувачам;

- ***внутрішня перелінковка*** – посилання всередині сайту, тобто посилання для переходу між сторінками безпосередньо в тексті на сторінці, а не через меню;
- ***швидкість завантаження сайту*** – показник швидкості завантаження сайту.

Для характеристики швидкості використовується кілька параметрів завантаження сайту: завантаження до появи першого контенту; завантаження першого контенту до взаємодії; швидкість відповіді сервера на запит; довжина *html*-коду.

Усі фактори, що впливають на положення сайту у видачі пошукової системи, можна поділити на *зовнішні* та *внутрішні*.

До *внутрішньої оптимізації* належить робота, що спрямована на загальне підвищення якості сайту та користі, яку він приносить відвідувачу.

До *внутрішньої оптимізації* можна віднести роботу над структурою проєкту, над полегшенням сприйняття контенту та безпосередньо над якістю цього контенту.

Складові внутрішньої пошукової оптимізації:

- *HTML, CSS*-валідація;
- збільшення швидкості роботи сайту;
- адаптація до мобільних пристроїв;
- складання семантичного ядра;
- аналіз сайтів-конкурентів;
- написання якісного контенту та створення правильної структури сайту;
- коригування текстової релевантності всієї сторінки;
- оптимізація метатегів і тегів заголовку;
- внутрішня перелінковка.

Зовнішні фактори оптимізації поділяються на *статичні* та *динамічні*.

Статичні *зовнішні фактори* визначають релевантність сайту на основі цитування його зовнішніми *web*-ресурсами, а також їхньої авторитетності незалежно від тексту цитування.

Динамічні *зовнішні фактори* визначають релевантність сайту на підставі цитування його зовнішніми *web*-ресурсами та їхньої авторитетності залежно від тексту цитування.

Складові зовнішньої пошукової оптимізації:

- реєстрація у незалежних каталогах;
- реєстрація у каталогах пошукових систем;
- обмін посиланнями;
- реєстрація в сервісах;
- розміщення статей;
- соціальні мережі;
- пресрелізи;
- наявність блогів;
- наявність мережі сайтів «сателітів».

3.4. Використання систем управління контентом

Якщо дивитися з позиції звичайного замовника, то розробка сайту на основі будь-якої *CMS* повинна мати такі переваги:

- у роботі використовується найбільш ефективний інструмент для вирішення конкретного завдання (залежно від виду сайту та вимог до його функціонала підбирають оптимальну *CMS*);
- використання *CMS* дає змогу власникові сайту самостійно створювати й видаляти розділи сайту, редагувати різну інформацію без залучення стороннього фахівця – це одна з переваг над статичними сайтами;
- робота сайту постійно тестується великою кількістю користувачів, а знайдені помилки й уразливості досить оперативно усуваються, при цьому сайт працює на найпередовіших і перевірених технічних рішеннях;
- тимчасові витрати на розробку сайту істотно знижуються, оскільки розробникові не потрібно фіксувати свою увагу на чисто технічних завданнях: «як зробити стрічку з новинами» або «як навчити CMS шукати товари в каталозі», а можна зосередитися на інформаційній і візуальній складових майбутнього сайту.

CMS бувають різні. Деякі системи орієнтовані тільки на вирішення конкретних завдань (ведення блогів, інтернет-магазини, форуми), інші є універсальними й надають розробникам зручне середовище проєктування та програмування для розробки різноманітних сайтів і *web*-додатків. Частина *CMS* складається з безлічі функціональних блоків модулів, інші є монолітні та ще й зашифровані. Одні системи поставляються безоплатно і з можливістю доопрацювання, а деякі надаються за гроші й не допускають можливості редагування ядра «движка».

Досі немає єдиної і чіткої класифікації, прийнятої ринком, існуючих *CMS*, проте це не заважає виділити лідерів у цій сфері, з яких першу п'ятірку становлять *Joomla, Wordpress, Drupal, MODX, 1С-Бітрікс*.

Joomla – одна з найбільш потужних *CMS* з відкритим кодом (Open Source CMS) на планеті. Яскрава, сучасна, постійно оновлюється, легко встановлюється, досить проста в управлінні й використанні, надійна. Вона використовується по всьому світу для всього: від простих сайтів до комплексних корпоративних додатків.

Joomla є безплатним програмним забезпеченням, захищеним ліцензією GPL. На ній можна робити сайти будь-якої складності і для різних цілей, як-от інформаційні або корпоративні сайти, сайти для малого бізнесу, персональні або домашні сторінки.

Joomla дуже зручна у використанні як для дизайнера, так і для розробника. Вони можуть швидко розробляти сайти для своїх клієнтів, а потім з мінімальними інструкціями навчати їх управляти своїми сайтами самостійно.

Якщо клієнти потребують спеціалізованих функціональних можливостей, то стандартні можливості *Joomla* легко розширюються. Уже написано понад 8 тисяч додаткових розширень, більшість з яких розповсюджуються безплатно та доступні в каталозі розширень.

У багатьох компаній і організацій є вимоги, які не підтримуються стандартним функціоналом *Joomla*. У цих випадках вбудоване середовище розробки значно полегшує роботу

розробника для створення складного додатка, який допоможе розширити основні можливості системи.

Wordpress – система управління вмістом, яка має програмний функціонал, що використовує так звані віджети. Особливістю віджета є те, що він не розширює можливості самої платформи, а працює окремо. Завдяки ньому можна вносити істотні зміни в дизайн і оформлення сайту. Звичайно, віджети повною мірою не зможуть замінити *HTML, CSS, PHP* і *MySQL*, але і того, що вони пропонують, вистачає для вирішення переважної кількості задач. Для вивчення вказаних мов програмування, структуризації даних і розмітки потрібно зануритися в читання книг, що потребує часу. А з віджетами таке налаштування в гіршому разі забирає години. Причому рівень виконання буде такий, що результат усі визнають професійною роботою.

Ця система популярна і серед професійних *web*-дизайнерів і програмістів. Завдяки *Wordpress* здійснюється наповнення сайту безліччю необхідних скриптів і нових стилів. І професіоналам часто просто необхідно підправити окремі елементи, щоб вони відповідали бажаному результату. Це повноцінна система програм, яку можна настроювати під конкретні цілі. Вона без проблем постійно працює на сайті. Безперервно здійснюється підтримка його функціональності, від початку і до кінця.

Спочатку технологія *Wordpress* була орієнтована на те, щоб створювати персональні блоги: людина пише собі все що захоче, читачі коментують – от і все. Але надалі система набула нових можливостей, чому немало посприяв її безплатний статус. Як результат, зараз у ній можна створити і каталог, і *інтернет*-магазин, і блог. Причому під безплатністю розуміється не надання доменного імені та хостингу, а теми, плагіни, скрипти й інше програмне забезпечення.

У *Wordpress* є режим роботи *WordpressMultisite*, який дає змогу використовувати файли ядра системи та наявну базу даних для побудови мережі із сайтів, що створені на базі самої системи. При цьому на кожному ресурсі можна здійснювати власні налаштування.

Також у *Wordpress* існують десятки тисяч шаблонів із різними темами оформлення дизайну сайту. Їх встановлення просте – досить завантажити архів вибраного варіанта на сервер, а вже *CMS* його розпакує, встановить і змусить працювати. Наявність готових скриптів і віджетів дає змогу значно розширити можливості розробника сайту в його роботі.

Окрім описаних вище можливостей системи *Wordpress*, для неї можна виділити такі переваги: безоплатність, легкість користування, міжплатформеність, наявність графічного й текстового редактора, популярність. Якщо говорити про цифри, то існує як мінімум 60 мільйонів сайтів, на які кожен місяць заходить 350 мільйонів чоловік, і вони переглядають за цей термін приблизно 3 мільярди сторінок. І це все на Wordpress!

Drupal – один із представників *CMS*. Ця система може бути використана для побудови різних типів сайтів. У базовій поставці можливості системи найкраще підходять для побудови новинних і форумних сайтів, персональних і колективних блогів тощо, де основне завдання – не загубитися в купі матеріалів, що регулярно надходять, мати можливість їх структурувати й архівувати для легкого доступу в майбутньому. Функціонал нарощується додатковими модулями, які можна додавати в разі потреби.

Drupal – це не просто повноцінне функціональне середовище для розробки і створення всіляких сайтів, яке надається безплатно і має гнучкі можливості, але й у якомусь сенсі – стиль програмування і навіть життя.

Кожен документ сайту, що працює на *CMS Drupal*, може входити в одну або кілька рубрик. Самі ж рубрики можуть складати списки або складні ієрархічні структури довільної вкладеності (з множинними пращурами і перехресними посиланнями елементів). Також можлива наскрізна рубрикація за всіма типами документів сайту (наприклад, список ключових слів, загальний для форумів і блогів). Форум із виведенням цікавих новин на головну сторінку або сайт новин із блогами і відеопрезентаціями – усе це можна вкласти у єдиний рубрикатор (або кілька рубрикаторів) і це буде частинами

єдиного сайту, а не розрізненими сторінками, лише об’єднаними загальним дизайном.

У системі існують готові варіанти вирішення типових завдань. Сайт новин, сайт-візитівка компанії, блог або форум – такі сайти можна побудувати, користуючись тільки модулями системи, що йдуть у постачанні, потрібно тільки включити відповідні модулі, налаштувати їх за своїм смаком і перенести сайт на хостинг. Так, у стандартній комплектації *Drupal* легко можна отримати сайт-візитівку компанії, якщо користувача влаштують наявні шаблони тем оформлення (лише в налаштуваннях потрібно буде змінити кольори тем оформлення й логотип).

Для зручності доступу до архівних матеріалів використовується рубрикація контенту та пошук з урахуванням видів контенту, рубрик і вмісту. Документи зберігають незмінні посилання весь час свого життя (перманентні посилання). Також за допомогою коротких посилань і псевдонімів сайт набуває імена розділів, що запам’ятовуються, і окремих сторінок, які не використовують спеціальних символів і тому добре індексуються пошуковими системами.

Для побудови структури сайту в *Drupal* використовується оригінальна методика під назвою «Таксономія» (Taxonomy). За допомогою таксономії можна визначити довільне число рубрик, у яких надалі розміщатимуться матеріали сайту. Ці рубрики можуть бути представлені як лінійні списки або ієрархічні структури довільної вкладеності (як деревовидні, коли елемент має тільки один батьківський елемент в ієрархії, так і довільні, коли елемент може мати відразу декілька батьківських елементів). У підсумку отримуємо таку схему: документи (nodes) різних типів (nodetypes) асоціюються з рубриками (terms), рубрики, зі свого боку, розбиваються на приналежність до словників (vocabularies). Така схема дає змогу вибудовувати на сайті декілька незалежних структур, асоціюючи ті самі документи (ніби листя на структурному «дереві» сайту) з різними структурами.

Вміст сайту в *Drupal* відокремлений від дизайну. За допомогою змінних тем можна дуже значно змінювати вигляд сайту, не чіпаючи при цьому його вміст і структуру. *Drupal* не прив'язаний до якогось одного механізму реалізації тем, розробник сайту може вибирати найбільш зручні йому способи формування дизайну сайту та налаштовувати вигляд сторінок за власним смаком. Це передбачає вибір тем із ряду готових рішень (themeengines), які пропонує *Drupal,* як-от xtemplate (теми з шаблонами в *XML*) або phptemplate (теми з шаблонами на *PHP*).

У *Drupal* можливе включення *php*-скриптів у будь-яких документах сайту. У скриптах можна звертатися до функцій *API Drupal* – це дає змогу оперативно створювати динамічні сторінки з функціональністю, не передбаченою системою, не прибігаючи до написання окремих модулів. Окрім цього, до ядра *Drupal* можна додавати нові можливості за допомогою додаткових наявних готових модулів для різних застосувань – від фільтрів для імпорту даних і галерей зображень до систем ведення проєктів і електронної комерції.

Система *Drupal* надає різні механізми для інтернаціоналізації та локалізації інтерфейсу, в тому числі, можливість виправлення перекладів через *web*-інтерфейс (для перекладів, що зберігаються в базах даних). Також вдалою особливістю архітектури *Drupal* є повсюдне використання в ньому Unicode – усі тексти зберігаються в UTF-8, у цьому ж кодуванні відправляються новини й поштові повідомлення. Така уніфікація дає змогу публікувати на сайті матеріали різними мовами без перемикання кодувань у браузері – на одній сторінці мирно уживуться тексти англійською, німецькою і китайською. Слід зазначити, що для отримання повної багатомовності (синхронне ведення утримуваного сайту, а не тільки інтерфейсних написів різними мовами) вбудованого рішення немає.

Drupal також є безплатним програмним забезпеченням, поширюваним за ліцензією GPL, багатоплатформеною системою, яка може бути встановлена на більшості *web*-серверів (наприклад,

Apache і *MS IIS*). Для зберігання даних використовує *СУБД MySQL, PostgreSQL, MS SQL*.

MODX («модэкс») – це безплатна професійна система управління контентом для *web*-додатків, призначена для забезпечення й організації спільного процесу створення, редагування та управління контентом. *MODX* є досить сучасною розробкою, яка останнім часом впевнено набирає обертів і збільшує свою аудиторію шанувальників. До того ж вона є надзвичайно гнучкою і ефективною, при цьому безплатно поставляється, але й поріг входження в число розробників на цій *CMS* в декілька разів вищий, ніж на інших системах.

MODX розповсюджується безплатно за ліцензією GPL з відкритим вихідним програмним кодом. Це означає, що систему *MODX* може використовувати кожен як для особистого використання, так і для комерційного поширення сайтів, побудованих на цій системі управління.

MODX написана мовою *PHP* і може використовувати для зберігання даних *СУБД MySQL* або *MS SQL*. Система може бути встановлена на більшості *web*-серверів (наприклад, *IIS, Apache, Lighttpd, nginx* і *Zeus*), а панель адміністратора системи працює практично у всіх сучасних браузерах.

За допомогою *MODX* можна створити як простий блог, так і багатосторінковий інтернет-магазин. Розробники запевняють: «*CMS* ніколи не повинна стояти на шляху дизайну». Для кожної сторінки *MODX* пропонує кілька оригінальних шаблонів.

MODX дає змогу легко налаштовувати ключові слова, робить сайт більш «видимим» для пошукових машин, скорочує час просування. Платформа підтримує платіжні системи й безліч інших корисних модулів. Незважаючи на широту можливостей, вона відрізняється гнучкістю і зручністю адміністрування.

Перевагами системи є підтримка *RSS*, *web*-статистики, підписок, контроль усіх елементів сторінки (від платіжної системи до документообігу та чату). У разі хорошого володіння мовою *web*-

програмування *PHP* можна вільно «переписати» функціонал системи під себе.

Недоліками цієї системи є мала кількість готових шаблонів, потреба в спеціальних плагінах для відновлення видалених об'єктів.

MODX відмінно підійде для створення інтернет-магазину або іншого ресурсу, який передбачає проведення фінансових операцій. Однак, незважаючи на наявність вбудованих платіжних сервісів, платформа вважається не такою безпечною, як закриті *CMS*.

1С-Бітрікс – професійна система управління *web*-проєктами, універсальний програмний продукт для створення, підтримки й успішного розвитку корпоративних сайтів, *інтернет*-магазинів, інформаційних порталів, сайтів спільнот, соціальних мереж та інших *web*-проєктів.

Сайти під управлінням *1С-Бітрікс* надійно працюють за високих навантажень, навіть за вкрай високої відвідуваності сайт буде завжди в мережі, а отже, ваш бізнес ні на хвилину не перестане працювати.

Одною з особливих переваг системи *1С-Бітрікс* є безпека та надійність створюваного сайту. Безпека завжди повинна стояти на чолі кута під час розробки *інтернет*-проєкту. Адже чим надійніше захищений сайт (особливо *інтернет*-магазин), тим більша довіра до нього з боку клієнтів. Якщо сайт зламаний і дані були втрачені або, що гірше, потрапили в руки зловмисників, це може завдати непоправної шкоди репутації компанії.

Успішність сайту, можливість приносити більше прибутку, зручність для користувача – від цього безпосередньо залежить ваш прибуток. Завдяки широкому функціоналу аналітики можна виявити сильні і слабкі сторони свого сайту, виправити виявлені недоліки, підвищити конверсію та рентабельність свого бізнесу.

Сьогодні мати сайт під управлінням *1С-Бітрікс* не тільки розумно і вигідно, але й модно. Сайт, розроблений на кращій *CMS*, стане показником статусу і престижу. Але для використання системи *1С-Бітрікс* потрібно придбати ліцензію. Крім того, у разі замовлення

індивідуального дизайну сайту потрібно сплатити повний процес розробки.

1С-Бітрікс – потужна й об'ємна система управління сайтом, а отже, ресурсномістка. Для коректної роботи *CMS* потрібно вибрати досить потужний хостинг, оскільки до складу програмного продукту входить понад 40 модулів для управління інформаційним наповненням і структурою, продажами через Інтернет, соціальною мережею, медіафайлами та фотогалереями, форумами, блогами, рекламою тощо.

1С-Бітрікс забезпечує високий рівень захищеності від злому сайтів, що підтверджує незалежний аудит компанії Positive Technologies. Система автоматично проводить діагностику роботи та видає рекомендації щодо поліпшення продуктивності.

Продукт, створений на *1С-Бітрікс*, дає змогу досягти чудових результатів щодо швидкодії навіть в умовах обмеженості ресурсів, а також будувати високопродуктивні системи для *web*-сайтів із дуже великою відвідуваністю й високими піковими навантаженнями.

Система *1С-Бітрікс* повністю сумісна з 1С: Підприємство 8.2, завдяки чому можна створити інтернет-магазин, інтегрований в інформаційне середовище компанії: автоматично публікувати на сайті каталоги товарів з 1С, прайс-листи, вивантажувати замовлення, їх статуси, а також дані щодо залишків на складі із сайту 1С і назад. Покрокове вивантаження каталогу з 1С в *інтернет*-магазин знижує навантаження на ваш сайт, що особливо важливо для каталогів у десятки і сотні тисяч найменувань.

Також у ядро системи вбудований універсальний Framework для адаптивної верстки, що полегшує виготовлення сайтів для мобільних пристроїв будь-якого типу, на яких можна реалізувати можливість управління замовленнями й отримування основних звітів про роботу магазину зі звичайних мобільних пристроїв.

Контрольні запитання

1. Що таке web-сайт?
2. Які типи сайтів ви знаєте?

3. Що таке Internet-система?

4. Які Internet-системи ви знаєте?

5. Що таке система управління контентом (вмістом) сайту?

6. Що таке система управління взаємовідносинами з клієнтами?

7. Що таке система управління людськими ресурсами?

8. Що таке система планування ресурсів підприємств?

9. Які системи управління існують та чим вони відрізняються?

10. З яких етапів складається процес створення сайту?

11. Для чого використовуються система управління контентом?

Лекція 4

ОГЛЯД МОВ WEB-ПРОГРАМУВАННЯ (HTML, PHP)

Web-програмування – це розділ програмування, орієнтований на розробку *web*-додатків (програм-скриптів, що забезпечують функціонування динамічних *web*-сайтів).

Мови *web*-програмування – це мови, які призначені для роботи з *web*-технологіями. *Web*-програми, що створюються за допомогою таких мов, можна розділити на дві групи: *клієнтські* та *серверні*.

Клієнтські *програми (скрипти)* виконуються на комп'ютері користувача переважно браузерами. Такі скрипти дають можливість спростити роботу *серверних програм*, тим самим суттєво зменшивши навантаження на сервер.

Серверні *програми (скрипти)* виконуються на сервері, де розташований сайт. Коли користувач дає запит на певну сторінку *web*-сайту, а саме переходить на неї за посиланням або вводить в адресному рядку браузера *URI*-адресу, сторінка спочатку генерується на сервері, обробляється відповідними програми, що з нею пов'язані, і тільки потім прямує до користувача по мережі у вигляді *html*-документа.

Робота серверних програм повністю залежить від сервера, на якому розташований *web*-сайт, і від того, яка версія тієї чи іншої мови підтримується.

Важливою рисою роботи серверних мов є можливість організації безпосередньої взаємодії із *системою управління базами даних СУБД* – сервером бази даних, де упорядковано зберігається інформація, що належить до поточного сайту.

З поняттям *web*-програмування тісно пов'язане поняття *web*-розробка.

Web-розробка – це процес створення *web*-сайту або *web*-додатка.

Як вже зазначалось, основними етапами процесу створення сайту є *web*-дизайн, верстка сторінок, написання програм-скриптів, а також налаштування *web*-сервера.

На етапі верстки сторінок сайту використовується мова

гіпертекстової розмітки *HTML*, а також формальна мова опису зовнішнього вигляду *html*-документів *CSS*.

На етапі написання скриптів, для подій і виконання операцій на *web*-сторінках з боку комп'ютера-клієнта використовуються такі мови, як *JavaScript, Visual Basic Script, Java*, а для виконання операцій з боку сервера, використовуються такі мови, як препроцесор гіпертексту *PHP, Perl, Ruby, Python, Java, C#*.

Якщо сайт здійснює роботу із *СУБД*, для звернення до баз даних та роботи з їх таблицями використовується мова запитів *SQL (Select Query Language)*.

В особливих випадках для передачі й обміну даними між *web*-додатками може використовуватися розширена мова розмітки *XML (eXtensible Markup Language)*.

З огляду роботи будь яких *web*-додатків із використанням тих чи інших мов *web*-програмування, усе, що відбувається у *web*-просторі, спрямоване на те, щоб згенерувати *web*-сторінку у вигляді *html*-документа та відправити результат кінцевому споживачу. Отже, ключовим фактором у всьому цьому процесі є робота з *html*-кодом.

4.1. Мова розмітки HTML. Відображення web-сторінок

Мова HTML *(HyperText Markup Language)* – це стандартизована мова розмітки *гіпертекстових документів*, що відповідає міжнародному стандарту *ISO 8879*. Вона інтерпретується браузерами, і отриманий за результатами інтерпретації відформатований текст відображається на екрані монітора комп'ютера або мобільного пристрою. У Всесвітній павутині *html*-сторінки зазвичай передаються за протоколами ***http*** або ***https*** у вигляді простого чи шифрованого тексту відповідно.

Мова ***HTML*** була розроблена британським вченим Тімом Бернерс-Лі в 1991 році. Вона створювалася як мова для обміну науковою і технічною документацією, придатною для використання людьми, які не є фахівцями у сфері верстки. Основною ідеєю було те, що за допомогою *HTML* можна було легко створити відносно прості, але красиво оформлені документи. Розробкою *HTML* у версії

2.0 і стандартизованої специфікації до неї зайнявся консорціум *W3C (World Wide Web Consortium)* під керівництвом Тіма Бернер-Лі в 1995 році. На сьогодні консорціумом *W3C* розроблена і впроваджена 5-та версія *HTML*.

Переважно *html*-документи зберігаються у файлах із розширенням `*.html` або `*.htm`. Браузери й операційні системи за цим розширенням розпізнають документи як *web*-сторінки. Усі *html*-документи записуються у форматі *ASCII (American Standard Code for Information Interchange)* і тому можуть бути створені та відредаговані в будь-якому текстовому редакторі.

4.1.1. Теги в HTML

Будь-який гіпертекстовий документ поділяється на окремі структурні елементи, що є компонентами його структури, за допомогою *тегів*. Кожний *тег* складається з лівої дужки «<», імені *тегу* та правої дужки «>». Теги поділяються на *теги* відкриття (наприклад, `<H1>`) і *теги* закриття (наприклад, `</H1>`). Слеш «/» вказує на те, що це тег закриття. Між тегами відкриття й закриття розміщено частину інформації, що відображається у браузері у вигляді тексту, графіки чи мультимедіа.

Самі *теги* є дескрипторами, за допомогою яких браузер ідентифікує той чи інший елемент. Для кожного з них у *HTML* використовуються *атрибути*, що визначають властивості елемента і *стилі*, що описують, у якому вигляді користувач побачить інформацію на екрані.

Загальний синтаксис написання тегів такий:

```
<ТЕГ атрибут1="значення" атрибут2="значення">...</ТЕГ>
```

У *тегів* допустимі різні *атрибути*, які розділяються між собою проміжком. Утім, є *теги* без *атрибутів*. Умовно *атрибути* можна поділити на обов’язкові, вони неодмінно повинні бути присутні, і необов’язкові, їх додавання залежить від мети застосування тегу.

Теги поділяються на два типи: парні (контейнери) і одиночні. Парні можуть містити інші теги або текст, тому вони є контейнерами того вмісту, який вони охоплюють. Вони обов’язково складаються з

двох однойменних тегів відкриття і закриття. Допускається вкладати до контейнера інші теги, однак слід дотримуватися їх послідовності (рис. 4.1).

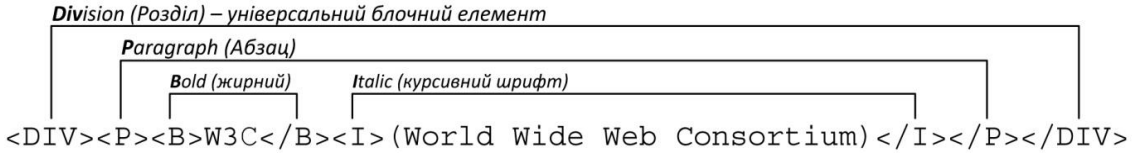


Рис. 4.1

Одиночні теги використовуються самостійно, наприклад:

```
<IMG src="drawing.jpg" height="350" width="400">
```

Кожний *тег* належить до певної групи: наприклад, *табличні теги* спрямовані на формування таблиць і не можуть застосовуватися для форматування фрагментів тексту, але можуть слугувати інструментом для логічної розмітки структури якогось певного блоку на частини. Таким чином, *теги* поділяються на групи: *теги верхнього рівня*; *теги заголовка документа*; *теги блочних елементів*; *теги рядкових елементів*; *теги списків*; *табличні теги*; *елементи форм*; *фрейми*; *теги форматування фрагментів тексту*.

Теги верхнього рівня призначені для формування структури *web*-сторінки та визначають розділ заголовка й тіла документа. До них належать такі теги, як <HTML>, <HEAD>, <BODY>.

Кожний *html*-документ починається і закінчується парою тегів <HTML>…</HTML>. Тег <HTML> пояснює браузеру, що документ містить інформацію мовою HTML. Подальший текст, до тегу закриття </HTML>, браузер сприймає як інструкцію мови розмітки. Структурно *html*-документ поділяється на дві основні частини: заголовок документа, який відділяється парою тегів <HEAD>…</HEAD>, і тіло документа, що відділяється парою тегів <BODY>…</BODY>.

Теги заголовка документа призначені для інформування браузера й пошукових систем, а також для підключення додаткових зовнішніх файлів, які можна використовувати в роботі поточної *web*-сторінки. У заголовку документа за допомогою тегу <TITLE> вказується текст заголовка документа, який виводиться в рядку

назви вікна браузера. За допомогою метатегів <META> вказується група інструкцій для браузера й пошукових систем, яка використовується для ідентифікації документа. Також у разі потреби в частині заголовка документа відповідними тегами <SCRIPT> і <LINK> підключаються файли скриптів і стилів за допомогою атрибутів src та href відповідно.

Решта тегів із вищенаведених груп використовується для розташування елементів гіпертексту в тілі документа, які відображаються в робочій області вікна браузера під час перегляду *web*-сторінки. Так, до тегів блочних елементів належать такі теги, як заголовки розділів різних рівнів від 1 до 6 – <H1>, <H2>, … , <H6>, абзаци – <P>, універсальні блочні елементи – <DIV>, які мають багатофункціональне призначення. До тегів рядкових елементів належать такі теги, як зображення – <IMG> з обов'язковим атрибутом src, гіперпосилання – <A> з обов'язковим атрибутом href, а також теги форматування шрифту в рядку. До тегів списків належать теги, які утворюють самі списки: ненумеровані й нумеровані – <UL> і <OL>, списки меню – <MENU>, а також теги, що утворюють складові елементи списків – <LI>. До табличних тегів належать теги, що утворюють таблицю – <TABLE>, <THEAD> <TBODY>, <TFOOT>, теги, що утворюють рядки, клітинки заголовків стовпців і клітинки стовпців таблиці в межах <TABLE>…</TABLE> відповідно <TR>, <TH> і <TD>.

Окремою групою є теги елементів форми, які використовуються для організації обміну даними. У межах форми <FORM>…</FORM> розташовуються елементи керування для передачі даних. Серед них – текстові поля <TEXTAREA>, кнопки <BUTTON>, списки вибору <SELECT> і елементи вводу багатотипного призначення <INPUT> з обов'язковим атрибутом type, яким задається тип елемента, а також з атрибутами name і value.

Особливою групою є теги для утворення фреймів. Існують фрейми <FRAME>, що розділяють вікно браузера на окремі області в

межах контейнера <FRAMESET>…</FRAMESET>, за допомогою яких *web*-сторінка поділяється на декілька документів, і фрейми <IFRAME>, що утворюють область втіленої *web*-сторінки в тілі певного документа. У кожну з таких областей завантажується самостійна *web*-сторінка, адреса якої визначається атрибутом src.

Наведені вище теги є основними та найбільш використовуваними елементами тіла документа. Більш повний список елементів мови гіпертекстової розмітки можна знайти в будь-якому довіднику з HTML, наприклад [11].

У лістингу 4.1 наведено приклад коду *html*-документа в розмітці *html*-тегами, який відобразиться в браузері у вигляді *web*-сторінки (рис. 4.2).

```
<HTML>
 <HEAD>
 <TITLE>Hypertext markup language – Мова
 гіпертекстової розмітки</TITLE>
 <META http-equiv="Content-Type" content="text/html";
 charset="Windows-1251">
 <META name="keywords" content="meta, html, www, web">
 </HEAD>
 <BODY text="black" link="blue" vlink="darkblue"
alink="red">
 <H1>Мова Html</H1><H2>Створення документа в HTML</H2>
  <P align="justify"><U>HyperText Markup Language</U>
  <B>(HTML)</B> є стандартною мовою, що призначена для
  створення <I>гіпертекстових</I> документів в
  середовищі Internet.<BR>Всі <B>HTML</B> документи
  записуються в форматі <I>ASCII</I>.<BR>
  Будь-який <I>гіпертекст</I> може бути розбитий на
  окремі структурні елементи:</P>
  <UL type="circle">
   <LI>Власне документ</LI>
   <LI>Глави, параграфи, пункти, підпункти</LI>
   <LI>Абзаци</LI><LI>Таблиці</LI><LI>Малюнки</LI>
  </UL>
  <DIV>Пошук інформації в мережі <b>Internet</b>
  здійснюється
```

```
  за допомогою пошукових систем:<BR>
  <A href="http://google.com">Google</A>,
  <A href="http://yandex.ua">Yandex</A>,
  <A href="http://rambler.ru">Rambler</A>,
  <A href="http://yahoo.com">Yahoo</A>,
  <A href="http://msn.com">MSN</A>.
  </DIV>
 </BODY>
</HTML>
```

Лістинг 4.1. Приклад коду *html*-документа

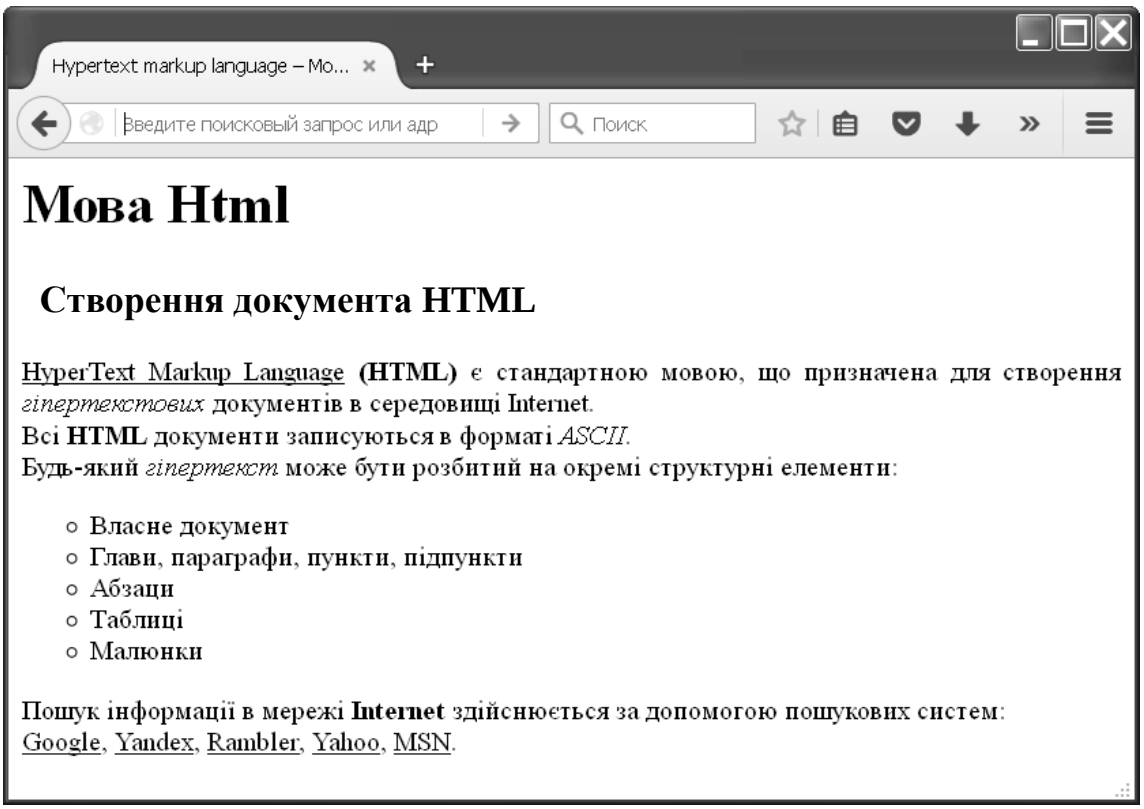


Рис. 4.2

4.1.2. Атрибути тегів

Щоб розширити можливості тегів і більш гнучко управляти вмістом контейнерів, застосовуються атрибути тегів. За призначенням вони поділяються на ідентифікатори (`id`, `class`, `name`, `type`, `accesskey`), атрибути стилю (`style`, також `id`, `class`), атрибути вмісту значень параметрів (`value`, `content`), що задаються атрибутом `name`, атрибути подій (`onload`, `onfocus`,

onblur, onclick, onchange, onmouseover, onmousemove тощо), атрибути, що містять адреси посилання на ресурс чи адреси файлів для відкриття або відображення (href, src, action, background), атрибути відображення елементів (hidden, width, height, border, size, align, title, alt, color, bgcolor тощо).

Основними атрибутами тегу елемента, що використовуються для всіх елементів, є: унікальний ідентифікатор елемента – id, ідентифікатор класу *css*-стилю елемента – class, атрибут *css*-стилю елемента – style. Атрибути type, name, value використовуються переважно для тегів елементів керування. Усі інші атрибути та їх набір використовуються залежно від призначення тегу елемента. Атрибути відображення елементів, що формують зовнішній вигляд елемента, останнім часом використовуються рідше, оскільки стилі відображення набагато зручніше задавати за допомогою атрибута стилю style чи у файлі *css*-стилів. Повний список атрибутів для тегів залежно від їх призначення можна знайти в будь-якому довіднику з HTML, наприклад [11].

4.1.3. Елементи керування формою

Оскільки *HTML* є мовою розмітки гіпертекстового документа для його відображення, він не є повноцінною мовою програмування, у якій здійснюється робота зі змінними, обробка даних тощо. Ця задача відводиться мовам *web*-програмування, а саме: *PHP*, *PERL*, *PYTHON*, *ASP*, *VBScript*, *JavaScript* і деяким іншим. Але для того щоб передати необхідні дані для обробки, їх потрібно ввести або задати. Здійснюється це на *web*-сторінці за допомогою елементів керування формою, які утворюються відповідними тегами. У лістингу 4.2 наведено приклад такої форми, а на рис. 4.3 показано, як вона відобразиться на *web*-сторінці. У ньому продемонстровані такі елементи, як приховане поле введення (елемент, що утворюється тегом <INPUT> з типом hidden), текстові поля (елементи, що утворюється тегом <INPUT> з типом text), багаторядкове текстове

поле (елемент, що утворюється тегом <TEXTAREA>), кнопка для здійснення відправлення даних форми до сервера (елемент, що утворюється тегом <INPUT> з типом submit), кнопка для виконання дії з подією onclick (елемент, що утворюється тегом <INPUT> з типом button). Типи для елемента керування <INPUT> задаються атрибутом type.

```
<HTML>
 <HEAD>
  <TITLE>Форма відправлення повідомлення</TITLE>
  <META http-equiv="Content-Type" content="text/html";
  charset="Windows-1251">
 </HEAD>
 <BODY>
  <FORM action="index.php" method="post">
   <INPUT type="hidden" name="cmd" value="sendmail">
   Тема повідомлення:
   <INPUT type="text" name="subject" value=""><BR>
   e-mail відправника:
   <INPUT type="text" name="email" value=""><BR>
   Ім’я відправника:
   <INPUT type="text" name="username" value=""><BR>
   Текст повідомлення:<BR>
   <TEXTAREA name="message"></TEXTAREA><BR>
   <INPUT type="submit" value="Відправити">
   <INPUT type="button" value="Відмінити"
   onclick="cancel();">
  </FORM>
 </BODY>
</HTML>
```

Лістинг 4.2. Приклад *html*-документа з формою

Як правило, приховані елементи введення використовуються для встановлення сталого значення змінної, а всі інші поля введення (рис. 4.3) – для введення даних, що зберігатимуться в атрибуті value змінної, ім’я якої задається атрибутом name. У цьому прикладі з натисканням на кнопку «Відмінити» виконується функція cancel(). Після натискання на кнопку «Відправити» відбудеться

передача даних форми до сервера за методом POST з виконанням файлу `index.php`.

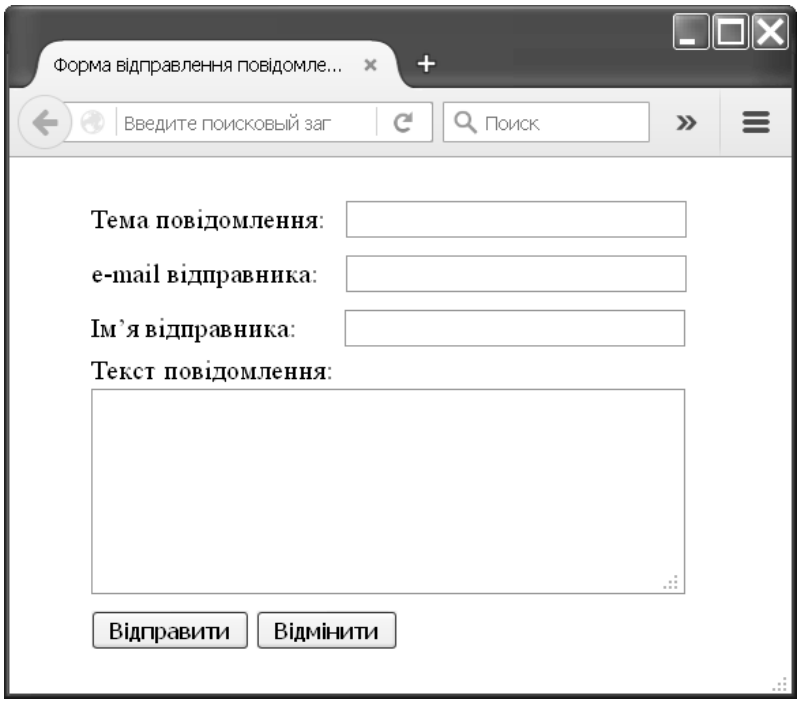


Рис. 4.3

Обробка даних на сервері здійснюється за допомогою програм (скриптів), що створені за серверними мовами програмування. Серед них – *PERL, JAVA, RUBY, PYTHON, ASP, PHP*. Однак майже всі з них є мовами загального призначення, які створювалися для вирішення загальносистемних задач. Мова сценаріїв *PHP* насамперед створювалася як вбудована мова розробки динамічних *web*-сторінок, скрипт якої легко втілюється в *html*-код *web*-сторінки, що й робить її найбільш популярною та вживаною мовою *web*-програмування, оскільки вона володіє великим набором вбудованих засобів для розробки *web*-додатків і має багато переваг порівняно з іншими мовами.

4.2. Мова програмування PHP

Мова PHP *(Hypertext Preprocessor)* – скриптова мова, яка спеціально розроблена для написання *web*-додатків (сценаріїв), що виконуються на *web*-сервері препроцесором гіпертексту.

Препроцесор *PHP* працює на *комп'ютері-сервері*. Таким чином, він здатний виконувати все те, що виконує будь-яка інша

програма стандарту *CGI (Common Gateway Interface)*, і використовується для організації взаємодії програми *web*-сервера із зовнішньою програмою.

Препроцесор здійснює генерування динамічних *web*-сторінок, обробку даних форм, відправку та прийом даних cookies (фрагментів даних, які зберігає *web*-браузер у спеціальних файлах із метою їх обміну із *cgi*-програмою *web*-сервера), а також роботу із сесіями. Схему генерування *html*-документа з використанням препроцесора *PHP* показано на рис. 4.4.

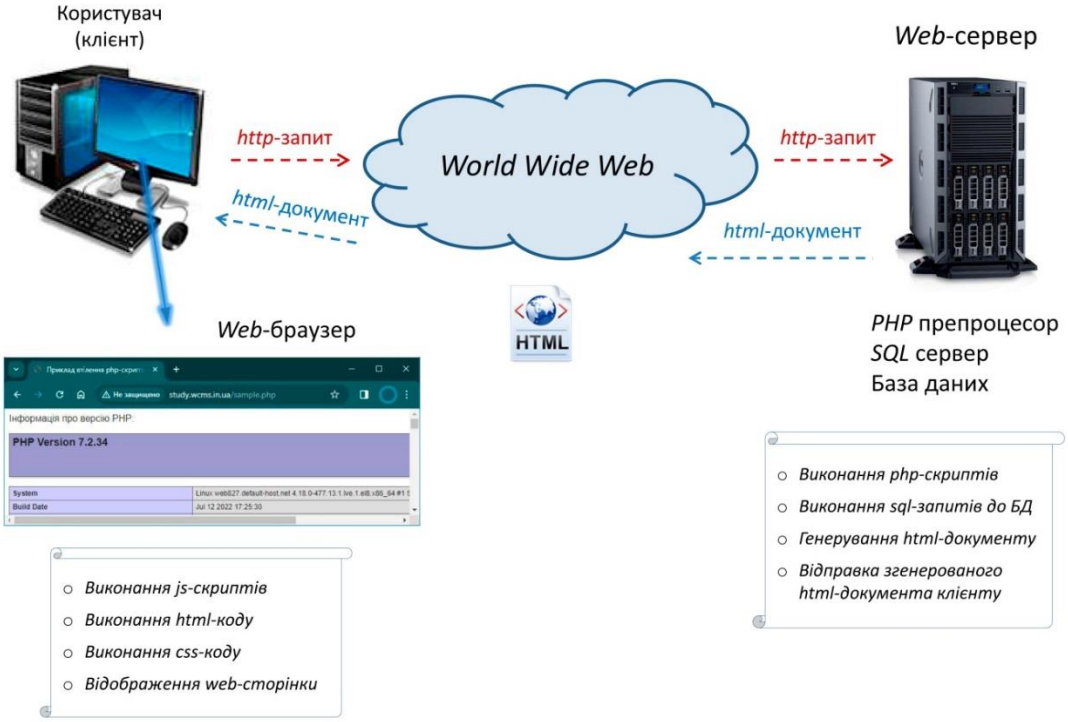


Рис. 4.4

Препроцесор *PHP* є інтерпретатором, який обробляє сценарії, що написані мовою *PHP*, транслює їх у байт-код, виконує його, а наприкінці виконання повертає результат у вигляді тексту, придатного для обробки іншими програмами, а саме *web*-браузерами.

Синтаксис мови *PHP* базується на синтаксисі таких мов, як *C, JAVA, PERL*. Основною метою цієї мови є надання *web*-розробникам можливості швидкого та легкого створення ґенеруємих *web*-сторінок.

PHP у наш час підтримується переважною більшістю хостинг-провайдерів і є одним із лідерів серед мов, що застосовуються для створення *динамічних web-сайтів*.

Для роботи програми не потрібно описувати будь-які змінні, використовувані модулі тощо. Будь-яка програма може починатися безпосередньо з оператора *PHP*, у межах *php*-блоку, який утворюється тегами «<? … ?>». У лістингу 4.3 наведено приклад втілення *php*-скрипту в *html*-документ.

```
<HTML>
 <HEAD>
  <TITLE>
  <? print "Приклад втілення php-скрипту" ?>
  </TITLE>
 </HEAD>
<BODY>
<?
  print "Інформація про версію PHP:<br>";
  phpinfo();
?>
</BODY>
</HTML>
```

Лістинг 4.3. Приклад втілення *php*-скрипта в *html*-документі

У цьому прикладі в *html*-код сторінки вставлені дві області *php*-скрипту. Під час генерування такої сторінки в першій області командою `print` буде виведено текст «`Приклад втілення php-скрипта`». У другій – першою дією також командою `print` буде виведено текстовий рядок «`Інформація про версію PHP:<br>`», другою – функцією `phpinfo()` на сторінці у браузері буде виведено інформацію про версію *PHP* та її налаштування.

Дуже зручною особливістю мови, що робить її гнучкою й найбільш популярною мовою для створення динамічних *web*-сторінок, є те, що *php*-скрипт можна як втілювати в *html*-код, так і частину *html*-коду у вигляді тексту втілювати у вміст текстової змінної в *php*-скрипті, тим самим конструюючи текст генерованої

кінцевої *web*-сторінки залежно від різних умов. Приклад такого конструювання наведений в лістингу 4.4.

```
<HTML>
 <HEAD>
 <?
  print "<title>Приклад конструювання html-
  документа</title>";
 ?>
 </HEAD>
 <BODY>
 <?
 if($font=='bold') $font="<b>напівжирний</b>";
 elseif($font=='italic') $font="<i>курсивний</i>";
 else $font="звичайний";
 print "<div>Обрано накреслення шрифту - $font</div>";
 ?>
 </BODY>
</HTML>
```

Лістинг 4.4. Приклад конструювання *html*-документа з використанням *PHP*

У цьому прикладі в *html*-код сторінки також вставлені дві області *php*-скрипту. У разі генерування такої сторінки в першій області командою `print` буде вдруковано текстовий рядок, що зазначений у лапках. У другій – спочатку виконається перевірка умови відповідності вмісту змінної `$font` тому чи іншому значенню з присвоєнням нового значення залежно від результату перевірки умови. Наступною дією командою `print` буде вдруковано текстовий рядок, що зазначений у лапках, причому текст у ньому буде об’єднаний із вмістом змінної `$font`.

Крім того що *php*-скрипт може втілюватися в *html*-документ, тим самим використовуючись у файлах із розширенням `*.html` і `*.htm`, він також може існувати як окрема програма, викладена в файлі з розширенням `*.php`.

4.2.1. Змінні в PHP

Змінні у PHP представлені знаком «$», після якого вказується ім'я змінної, наприклад `$value`. Ім'я змінної чутливе до регістру. Імена змінних відповідають тим самим правилам, що й інші назви у *PHP*. Правильне ім'я змінної повинно починатися з букви або символу підкреслення і складатися з букв, цифр і символів підкреслення в будь-якій кількості.

У *PHP* використовуються зумовлені чи суперглобальні змінні. Будь-якому скрипту, що запускається, *PHP* надає велику кількість зумовлених змінних. Однак багато з них не можуть бути повністю задокументовані, оскільки вони залежать від сервера, що запускає скрипт, його версії та налаштувань, а також від інших факторів. Деякі з цих змінних недоступні. Повний список зарезервованих зумовлених змінних можна переглянути в online-довіднику мови *PHP* на сайті http://php.net.

Область видимості змінної визначається тим, де і як вона була оголошена. Поділяють змінні на *локальні*, *глобальні* та *статичні*. Змінна, оголошена в межах функції, без ключових слів, вважається *локальною*. Така змінна доступна тільки всередині функції, у якій вона визначена. За будь-якого присвоєння поза функцією використовуватиметься зовсім інша змінна, яка не має нічого спільного (крім імені) зі змінною, оголошеною всередині функції.

Глобальні змінні – це змінні, які доступні всій програмі, включно з підпрограмами та функціями.

Статичні змінні існують тільки в локальній області видимості функції, але не втрачають свого значення, коли виконання програми виходить із цієї області видимості.

Оголошення *локальної змінної* здійснюється першим присвоюванням їй значення в межах функції.

Оголошення глобальної змінної здійснюється першим присвоюванням їй значення в межах усієї програми, а її виклик у межах будь-якої функції у версіях *PHP* до 5-ї включно здійснюється ключовим словом `global` або у версіях *PHP*, починаючи з 4-ї, – за

допомогою суперглобального масиву `$GLOBALS`, у якому зберігаються всі глобальні змінні.

Оголошення статичної змінної в межах функції здійснюється за допомогою ключового слова `static`.

Іноді буває зручно мати змінним ім'я змінної, тобто ім'я, яке може бути визначено і змінено динамічно. Для цього у *PHP* використовуються так звані мінливі змінні. Такі змінні починаються з двох знаків «`$$`». Наприклад, якщо звичайна змінна `$a` набуде значення «`string`», тобто `$a = «string»`, а мінлива змінна `$$a` набуде значення «`text line`», тобто `$$a = «text line»`, то динамічно утвориться змінна `$string`, що міститиме значення «`text line`», тобто `$string = «text line»`.

Разом із цим у *PHP* використовуються зовнішні змінні. Це змінні, які приходять на сервер з *web*-сторінки користувача зі списку параметрів запиту *URL* або в разі відправлення даних із форми. Ці змінні відразу є глобальними й окремого оголошення не потребують. Вони також містяться в суперглобальному масиві `$_GET`, якщо дані були передані через *URL* або відправлені з форми за методом GET, чи в суперглобальному масиві `$_POST`, якщо дані були відправлені з форми за методом POST.

4.2.2. Масиви в PHP

Масив у PHP – це впорядкований набір даних, що являє собою список однотипних елементів, у яких встановлюється відповідність між значенням і ключем. Він оптимізований у декількох напрямах, тому його можна використовувати як масив, список, хеш-таблицю, словник, колекцію, стек, чергу тощо. Оскільки значенням масиву може бути інший масив, можна також створювати дерева й багатовимірні масиви.

Існує два типи масивів, що розрізняються за способом ідентифікації елементів. У масивах першого типу елемент визначається індексом. Такі масиви називаються *індексованими*. Масиви другого типу мають асоціативну природу, і для звертання до елементів використовуються ключі (у вигляді ключових слів), які

логічно пов’язані зі значеннями. Такі масиви називають *асоціативними*.

Важливою особливістю мови *PHP* є те, що в ній, на відміну від інших мов, можна створювати масиви будь-якої складності безпосередньо в тілі програми (скрипту).

У *PHP* використовується декілька способів оголошення (створення) масивів. Перший з них – присвоєння значення будь-якому елементу масиву, наприклад: `$a[]='Мова php'`. Другий спосіб – за допомогою конструкції `array()`, наприклад: `$a=array('html','php','js')`. Третій спосіб, починаючи з версії PHP 5.4, – за допомогою квадратних дужок, наприклад: `$a=['c','perl','php']`. Для цих способів, якщо не вказувати індексу елемента масиву, він буде визначатися автоматично, починаючи з 0 і до останнього елемента. Наприклад, якщо елементам масиву послідовно присвоїмо значення:

`<? $a[]='Фундаменти'; $a[]='Стіни'; $a[]='Перекриття'; ?>`,

то автоматично елемент `$a[0]` набуде значення «`Фундаменти`», елемент `$a[1]` набуде значення «`Стіни`», елемент `$a[2]` набуде значення «`Перекриття`». У випадках, коли потрібно задати певний індекс, для першого способу він вказується в дужках, наприклад:

`<? $a[3]='Колони'; $a[4]='Ригелі'; $a[5]='Балки'; ?>`,

для другого та третього, відповідно, за допомогою конструкцій:

```
$name=array(0=>value, 1=>value1, 2=>value2, …)
```

та

```
$name=[0=>value, 1=>value1, 2=>value2, …].
```

Наприклад:

`<? $a=array(1=>'Пісок', 2=>'Щебінь', 3=>'Цемент'); ?>.`

У разі використання асоціативних масивів замість індексу застосовується ключ у вигляді ключового слова. Конструкція має вигляд: `$name[key]=value`, або

`$name=array(key1=>value1,key2=>value2,key3=>value3,…).`

Наприклад:

```
<?
$a['матеріал']="залізобетон";
$a['бетон']="C25/30";
$a['арматура']="A400C";
$b=array('колони'=>"залізобетонні",'стіни'=>"цегляні");
?>
```

За застосування багатовимірних індексованих чи асоціативних масивів у *PHP* використовуються прості багатовимірні масиви, які мають конструкцію `$name[key1][key2][key3]…[keyN]=value` та складні багатовимірні масиви, які мають конструкцію

```
$name[key1]=array(key11=>value11,key12=>value12, …);
$name[key2]=array(key21=>value21,key22=>value22, …);
…
$name[keyN]=array(keyN1=>valueN1,keyN2=>valueN2, …).
```

Також може використовуватися конструкція такого складу:

```
array(key1=>array(key11=>value11,key12=>value12, …),
      key2=>array(key21=>value21,key22=>value22, …),
…
      keyN=>array(keyN1=>valueN1,keyN2=>valueN2, …).
```

4.2.3. Типи даних у PHP

PHP є мовою програмування з динамічною типізацією. Це не вимагає вказувати тип змінної під час її оголошення. Перетворення між скалярними типами часто здійснюються неявно, без додаткових зусиль, утім *PHP* надає широкі можливості для явного перетворення типів. *PHP* підтримує вісім простих типів даних: чотири скалярні – `boolean`, `integer`, `float`, `string`; два змішані – `array`, `object`; два спеціальні – `resource`, `NULL`. Також використовуються декілька псевдотипів – `mixed`, `number`, `callback`, `void`.

Логічний тип `boolean` здатний приймати тільки два значення `TRUE` (істина) і `FALSE` (фальш). З перетворенням у логічний тип число 0, порожній рядок, нуль у рядку «0», `NULL` і порожній масив вважаються рівними `FALSE`. Усі інші символи автоматично перетворюються в `TRUE`.

Діапазон цілих чисел (integer) залежить від платформи. Зазвичай це діапазон 32-бітних знакових цілих чисел від –2 147 483 648 до 2 147 483 647. Діапазон дійсних чисел (double) також залежить від платформи. Для 32-бітної архітектури діапазон дозволяє оперувати числами від $\pm 1{,}7 \times 10^{-308}$ до $\pm 1{,}7 \times 10^{+308}$.

Рядок (string) – це довільна послідовність символів будь-якої довжини. Символ у рядку має розмір 1 байт. Це означає, що в рядку можливі 256 різних символів. Рядок може бути визначений декількома різними способами: одинарними лапками – 'текст'; подвійними лапками – "текст". На відміну від другого способу, змінні й екрануючі послідовності для спеціальних символів, що зустрічаються в рядках, взятих в одинарні лапки, не обробляються. Наприклад, якщо змінна $a набуде значення 'показник $in', а змінна $b набуде значення "Відомий $a", тобто:

```
<? $a='показник $in'; $b="Відомий $a"; ?>,
```

то внаслідок обробки рядка, що містить змінна $b, вона буде містити рядок 'Відомий показник $in'. Тобто '$in' (в одинарних лапках) сприйматиметься не як змінна, а як звичайний текст, на відміну від того, що "$a" (у подвійних лапках) обробиться як змінна. Ця особливість мови має дуже широке використання.

Тип даних array представляє масиви, які були розглянуті вище, тип даних object – об'єкти, які будуть детальніше розглянуті в окремому підрозділі «Об'єкти і класи».

Спеціальні змінні з типом resource містять посилання на зовнішні ресурси. Ресурси створюються й використовуються спеціальними функціями. Повний перелік цих функцій і відповідних типів ресурсів можна переглянути в online-довіднику мови *PHP* на сайті http://php.net.

Змінні можуть набувати спеціальне значення NULL, яке говорить про те, що ця змінна не має значення. NULL – це єдине можливе значення типу NULL. Існує тільки одне значення типу NULL – регістронезалежна константа NULL.

Псевдотип `mixed` говорить про те, що параметр може приймати багато типів (але не обов'язково всі). Псевдотип `number` говорить про те, що параметр може бути `integer` або `float`. Псевдотип `callback` приймають як параметр `callback`-функції, яку визначає користувач. `Callback`-функція – це функція зворотного виклику, в якій відбувається передача виконавчого коду як одного з параметрів іншого коду. Псевдотип `void`, як тип результату, означає, що повернене значення відсутнє. `Void` у списку параметрів означає, що функція не отримує параметри.

4.2.4. Керуючі конструкції

Будь-який сценарій *PHP* складається з ряду конструкцій (набір інструкцій). Конструкцією можуть бути оператори, функції, цикли, умовні оператори, навіть конструкції, що не роблять нічого (порожні конструкції). Конструкції зазвичай закінчуються символом крапка з комою – «`;`». Крім того, конструкції можуть бути згруповані в групу, формуючи групу конструкцій, взятих у фігурні дужки «`{...}`». Група конструкцій – це також окрема конструкція. Групи використовуються переважно разом із керуючими конструкціями. Керуючі конструкції мови – це набори інструкцій, що дають змогу змінювати перебіг виконання програми. У *PHP* існує чотири основні групи керуючих конструкцій: конструкції умови (`if…elseif…else`), конструкції циклів (`while`, `do…while`, `for`, `foreach`), конструкції вибору (`switch…case`), конструкції включень (`require`, `include`, `require_once`, `include_once`).

Конструкції умови є найбільш поширеними конструкціями в усіх алгоритмічних мовах програмування. Основною з них є конструкція з використанням оператору `if`. Вона має вигляд:

```
if(умова) вираз;
```

або

```
if(умова) {вираз1; вираз2; …; виразN;}.
```

За виконання умови здійсниться виконання виразу чи інструкції (набір виразів), інакше вираз ігнорується.

Часто потрібно виконати один вираз, якщо певна умова правильна, та інший вираз, якщо умова неправильна. Для цього конструкція умови розширюється оператором `else`. Така конструкція має вигляд:

```
if(умова) вираз1; else вираз2;
```

або

```
if(умова) {вираз1; вираз2; …; виразN;}
 else {вираз_A; вираз_B; …; вираз_Z;}.
```

Також у багатьох випадках потрібно здійснити перевірку різних умов, для кожної з яких виконуватиметься своя інструкція. Для цього конструкція умови розширюється оператором `elseif`. Така конструкція має вигляд:

```
if(умова1) вираз1; elseif(умова2) вираз2; else вираз3;
```

або

```
if(умова1) {вираз11; вираз12; …;}
elseif(умова2) {вираз21; вираз22; …;}
else {вираз_A; вираз_B; …;}.
```

Приклад:

```
<?
  $font='білий';
  if($bgcolor=='red') $background='червоний';
  elseif($bgcolor=='green') $background='зелений';
  elseif($bgcolor=='blue') $background='синій';
  else {$background='сірий'; $font='чорний';}
  print "Колір фону - $background, шрифт - $font.";
?>
```

На другому місці за частотою використання після конструкцій умов перебувають конструкції циклів. Цикли дають можливість повторювати певну кількість разів певний набір інструкцій, який має назву «*тіло циклу*». Один прохід циклу має назву «*ітерація*». *PHP* підтримує чотири види циклів: цикл із передумовою (`while`), цикл

з постумовою (do…while), цикл із лічильником (for), спеціальний цикл перебору масивів (foreach). У разі використання циклів є можливість використання операторів break і continue. Перший із них перериває роботу всього циклу, а другий – тільки поточної ітерації. Конструкція циклу з передумовою має вигляд:

```
while(логічний вираз) {тіло циклу;}
```

Цикл while працює за таким принципом: обчислюється значення логічного виразу, якщо значення є істинне (TRUE), виконується тіло циклу, в іншому випадку – переходимо на наступний за циклом оператор. Якщо тіло циклу складається з одного виразу, фігурні дужки можна опустити.

Приклад:

```
<?
 $x=0;
 while($x<100){$y=2*$x; print "$x $y<br>"; $x++;}
?>
```

Конструкція циклу з постумовою має вигляд:

```
do{тіло циклу;} while(логічний вираз)
```

На відміну від циклу while, цикл do…while перевіряє значення виразу після кожної ітерації. Таким чином, тіло циклу виконується хоча б один раз.

Цикл з лічильником використовується для виконання тіла циклу певну кількість разів. Конструкція циклу for має вигляд:

```
for(ініціалізація; умова; приростення) {тіло циклу;}
```

Цикл for починає свою роботу з ініціалізації змінної лічильника. Після цього перевіряється умова, якщо вона виконується (TRUE), то виконується тіло циклу. Після того як буде виконаний останній оператор тіла, виконуються приростення лічильника й перевіряється умова. Якщо умова виконується, знову виконується тіло циклу і наступне приростення. Якщо – ні, виконання циклу припиняється.

Приклад:

```
<?
 print "№<br>";
 for($i=0;$i<9;$i++)
 {if($i==0) continue; else print "$i<br>";}
?>
```

У *PHP* використовується спеціальний цикл для перебору масивів `foreach`. Конструкція цього циклу має вигляд:

```
foreach(array as key=>value) {тіло циклу;}
```

Тіло циклу циклічно виконується для кожного елемента масиву `array`, де за кожної ітерації змінній `$key` присвоюється ключ, а змінній `$value` – значення поточного елемента масиву.

Приклад:

```
<?
//Перебір масиву $Names з виведенням значень елементів масиву
  foreach($Names as $key=>$value)
  {print "$key. $value.<br>";}
?>
```

Часто замість декількох розташованих поспіль інструкцій `if…else` доцільно скористатися спеціальною конструкцією вибору `switch…case`. Ця конструкція призначена для вибору дій залежно від значення змінної, що перевіряється. Конструкцію вибору можна використовувати, якщо передбачуваних варіантів багато і для кожного варіанта потрібно виконати специфічні дії. Така конструкція має вигляд:

```
switch(змінна) {
case значення1: вираз1; break;
case значення2: вираз2; break;
...
case значенняN: виразN; break;
[default: вираз за замовчуванням; break;]
}
```

Конструкція вибору працює за таким принципом: змінна, що зазначена в `switch`, порівнюється із вказаними в `case` значеннями. Якщо вона дорівнює першому значенню, виконується перший вираз, якщо другому – другий тощо. Якщо жодне значення з набору не

збігається зі значенням змінної, виконується блок `default`, якщо він вказаний.

Приклад:

```
<?
 switch($found) {
  case 1: $str='стовпчастий'; break;
  case 2: $str='стрічковий'; break;
  case 3: $str='плитний'; break;
  default: $str='пальовий';
 }
 print "Для будівлі вибрано $str фундамент.";
?>
```

Однією з особливостей мови *PHP* є використання в ній конструкцій включення. Вони дають змогу збирати *PHP* програму (скрипт) із декількох окремих файлів. Основними з них є `require` і `include`. Конструкція `require` дає можливість додавати файли до сценарію *PHP* до його виконання. Конструкція `include` – додавати файли до коду *php*-скрипту під час виконання сценарію. Синтаксис конструкцій `require` і `include` має вигляд:

```
require Ім'я_файлу;
include Ім'я_файлу;.
```

Ці конструкції можуть зібрати сценарій *PHP* із декількох окремих файлів, які можуть бути як *html*-сторінками, так і *php*-скриптами. За допомогою конструкції `include` можна включати до сценарію один файл безліч разів із різними умовами виконання блоку сценарію у ньому.

У великих *php*-сценаріях інструкції `require` і `include` застосовуються дуже часто. Тому інколи стає досить складно контролювати, як би випадково не включити той самий файл декілька разів у випадках, коли це не потрібно, що найчастіше призводить до помилки, яку складно виявити. Для цього передбачено використання конструкції одноразового включення: `require_once` і `include_once`, що дає впевненість, що один файл не буде включений двічі. Працюють ці конструкції так само, як

і `require` і `include` відповідно. Різниця в їх роботі лише в тому, що перед включенням файлу інтерпретатор перевіряє, чи включався указаний файл раніше, чи ні. Якщо так, то файл не буде включений знову.

4.2.5. Створення та використання функцій

Мовою *PHP* використовується багато вбудованих функцій, повний список і призначення яких можна переглянути на сайті http://php.net. Але часто виникає потреба у створенні функцій, що виконують певні особисті задачі. Такі функції мають назву *функції користувача*.

Функції користувача в *PHP* мають ряд особливостей: доступні попередньо усталені параметри; можливість викликати певну функцію зі змінним числом параметрів; функції мають можливість повертати значення різних типів; область видимості змінних усередині функції є ієрархічною – є можливість змінювати змінні, передані як аргумент.

Функція користувача може бути оголошена в будь-якій частині скрипту, до місця її першого використання. Їй не потрібно ніякого попереднього оголошення, як в інших мовах програмування. Синтаксис оголошення функцій користувача має вигляд:

```
function Ім'я_функції(аргумент1,аргумент2,...)
{тіло функції;}
```

Оголошення функції починається службовим словом `function`, потім вказується ім'я функції, після імені функції – список аргументів у дужках. Тіло функції береться у фігурні дужки і може містити будь-яку кількість операторів.

До імен функцій висуваються декілька вимог: імена функцій можуть містити символи кирилиці, але надавати функціям імена, що складаються з таких символів, не рекомендовано; імена функцій не повинні містити проміжків; ім'я кожної функції повинно бути унікальним; функціям можна надавати такі ж самі імена, як і змінним, тільки без знака «$» на початку імені. При цьому потрібно пам'ятати, що регістр під час оголошення функцій і звернення до них

не враховується.

Типи значень, що повертає функція користувача, можуть бути будь-якими. Для повернення результату роботи функції до основної програми використовується оператор `return`. Якщо функція не повинна нічого повертати, оператора `return` не вказують. Оператор `return` може повертати все що завгодно, у тому числі й масиви.

Приклад оголошення і створення функції:

```
<?
 //Функція переводу дати "dd.mm.yyyy"
 //в формат бази даних "yyyy-mm-dd"
 function datefordb($date)
 {
  $d=substr($date,0,2);
  $m=substr($date,3,2);
  $y=substr($date,6,4);
  return "$y-$m-$d";
 } ?>
```

Використовується декілька способів виклику функції. Якщо функція виконується з поверненням значення, її виклик здійснюється шляхом присвоєння. Наприклад:

```
<? $date='08.09.2016'; $dbdate=datefordb($date); ?>
```

У випадку якщо функція виконується без повернення значення, вона викликається лише її застосуванням. Наприклад:

```
<? mainmenu(0,5); epath($eid); curpage(); ?>
```

Це дає дуже зручну можливість будувати фрагменти *html*-тексту у функціях, а потім вставляти їх як частини *html*-документа викликом певної функції.

4.2.6. Об’єкти і класи

Останнім часом ідея *об’єктно-орієнтованого програмування* (*ООП*) набула найбільш поширеного застосування у *web*-програмуванні. Об’єктно-орієнтовані програми більш прості та мобільні, їх легше модифікувати й супроводжувати. Крім того, сама

ідея об'єктної орієнтованості за грамотного її використання дає можливість програмі бути більш захищеною від різного роду помилок.

PHP включно з 4-ю версією забезпечував лише деяку підтримку *ООП*. Однак після виходу 5-ї версії підтримка *ООП* у *PHP* стала майже повною. Технологія *ООП* має три головні переваги: вона проста для розуміння (дає змогу мислити категоріями повсякденних об'єктів); підвищено надійна та проста для супроводу (правильне проєктування забезпечує простоту розширення й модифікації об'єктно-орієнтованих програм); прискорює цикл розробки. Специфіка *ООП* помітно підвищує ефективність праці програмістів і дає їм можливість створювати більш потужні, масштабовані й ефективні програми.

В об'єктно-орієнтованому програмуванні базовим поняттям є *клас*. ***Класи*** утворюють синтаксичну базу *ООП*. Їх можна розглядати як свого роду контейнери для логічно пов'язаних даних і функцій (що мають назву *методи класу*). Екземпляром класу є *об'єкт*. ***Об'єкт*** – це сукупність змінних (*властивостей*) і функцій (*методів*) для їх обробки. Властивості й методи називаються членами класу.

Синтаксис оголошення класів має вигляд:

```
class І'мя_класу {тіло класу;}
```

Оголошення класу починаються службовим словом `class`, після нього вказується ім'я класу. У тілі класу здійснюється опис членів класу – властивостей і методів для їх обробки. Для оголошення об'єкта, створюваного на базі класу, використовується оператор `new`:

```
Об'єкт = new І'мя_класу;
```

Властивості класу описуються за допомогою службового слова `var`. Методи класу описується так само, як і звичайні функції користувача, а саме за допомогою службового слова `function`.

Приклад оголошення класу, створення об'єкта, використання його властивостей і методів:

```
<?
 class Plate
 {
  var $length; var $width; var $height;
  function area(){
   $l=$this->length; $w=$this->width; $a=$l*$w;
   return $a;
  }
  function volume(){
   $a=$this->area(); $h=$this->height; $v=$a*$h;
   return $v;
  }
 }
 $plate = new Plate;
 $plate->length=6;   # м.
 $plate->width=1.5;  # м.
 $plate->height=0.2; # м.
 $A=$plate->area(); $V=$plate->volume();
 print "Площа плити $A м2, об'єм плити $V м3.";
?>
```

У наведеному прикладі створюється об'єкт `$plate` (плита) як примірник класу `Plate` з властивостями `length` (довжина), `width` (ширина), `height` (висота), методами `area()` і `volume()`, що визначають площу й об'єм плити відповідно.

У тілі класу під час його опису для доступу до членів класу з його методів використовується змінна `$this`. Звертання до властивостей і методів об'єкта здійснюється за допомогою послідовності символів «–>».

Під час оголошення членів класу можуть використовуватися модифікатори доступу, які дають змогу керувати доступом до методів і властивостей. Видимість членів класу може визначатися ключовими словами: `public`, `private`, `protected`. Модифікатор `public` дає можливість звертатися до властивостей і методів звідусіль. Модифікатор `private` – звертатися до властивостей і методів тільки в межах поточного класу. Модифікатор `protected` дає змогу звертатися до властивостей і

методів тільки поточного класу та класу, який наслідує його властивості й методи.

У світі *web*-розробки *PHP* давно стала однією з найпопулярніших мов *web*-програмування. Понад 78 % сайтів у світі використовують цю мову для виконання сценаріїв на сервері.

Зі свого боку, в об'єктно-орієнтованій парадигмі програмування програма поділяється на набір *об'єктів*, кожен з яких має свої *властивості* та *методи*. Такий підхід забезпечує можливість створювати програмне забезпечення, яке є більш модульним, гнучким, масштабованим і легко підтримуваним. Цей підхід до програмування дає змогу розділити функціональність програми на невеликі, логічно пов'язані модулі, що полегшує розуміння коду, його тестування та підтримку.

Об'єктно-орієнтоване програмування – це підхід до розробки програмного забезпечення, що ґрунтується на понятті *об'єктів* та їх взаємодії. *ООП* засновано на трьох ключових принципах: це *інкапсуляція*, *успадкування* та *поліморфізм*.

Інкапсуляція – це здатність приховувати деталі реалізації і надавати лише необхідний інтерфейс. Це означає, що можна мати *властивості* й *методи об'єкта*, але при цьому можемо вирішити, хто має доступ до них. Можна встановити деякі *властивості* та *методи* як публічні (доступні всім *об'єктам*) або як приватні (доступні лише всередині самого *об'єкта*).

Такий принцип *ООП* допомагає створювати більш безпечний та організований код.

Успадкування – це концепція, згідно з якою одні *класи* можуть приймати (успадковувати) властивості інших – *батьківських* класів. В *ООП* такі *класи* називають *дочірніми*, такими, що успадковують властивості й поведінку своїх *батьківських* класів.

Так, наприклад, може існувати клас «Седан», який успадковує *властивості* та *методи* від класу «Легкові автомобілі». Таким чином, клас «Седан» отримує всі основні характеристики класу «Легкові автомобілі» і може додати свої власні унікальні *властивості* та *методи*.

Такий принцип *ООП* дає змогу повторно використовувати код та створювати ієрархії *класів* для кращої організації.

Поліморфізм – це можливість *об'єкта* використовувати *методи похідного класу*, якого не існує на момент створення базового. Це означає, що *об'єкт* може проявляти різні форми, тобто використовувати *методи* з тим самим ім'ям, але з різною реалізацією, залежно від контексту.

Такий принцип *ООП* дозволяє працювати з *об'єктами* різних *класів* однаково, що спрощує кодування та підвищує його гнучкість.

Об'єкти мають свої *властивості* та *методи*, які можуть бути використані для взаємодії з іншими об'єктами.

Об'єкти, як і звичайні змінні, що створює і використовує препроцесор *PHP* під час виконання сценарію, є повністю віртуальними сутностями, які існують в оперативній пам'яті протягом виконання програми, а в момент її завершення руйнуються. Але результатом їх виконання можуть стати видимі об'єкти, що належать *web*-сторінкам, які відображаються у вікні браузера на комп'ютері-клієнті зі своїми властивостями та методами, якими можна керувати. І вже на цьому етапі обробник *PHP* є неспроможний керувати цими елементами, оскільки він працює на комп'ютері-сервері. Ця задача повинна виконуватися обробником на комп'ютері-клієнті. Такими обробниками є вбудовані до *Internet*-браузерів інтерпретатори. Вони виконують сценарії, що написані відповідними мовами *web*-програмування. Однією з них, найбільш популярною та розвинутою, є мова *JavaScript*.

Контрольні запитання

1. Для чого призначена мова HTML?
2. Що таке тег?
3. На які типи та групи поділяються html-теги?
4. Що таке атрибут тегу?
5. Для чого використовується мова web-програмування PHP?

6. Як представлена змінна у PHP та як оголосити змінну?
7. Скільки типів масивів існує у PHP?
8. Які способи оголошення масивів використовуються в PHP та як оголосити масив?
9. Які ви знаєте керуючі конструкції мови PHP?
10. Як створюються й використовуються функції у PHP?
11. Що таке об’єкт, що таке клас? Як створити об’єкт у PHP?

Лекція 5

ОГЛЯД МОВ WEB-ПРОГРАМУВАННЯ (JavaScript, SQL, CSS)

5.1. Мова програмування JavaScript

JavaScript – це *об'єктно-орієнтована мова програмування* з використанням *прототипно-орієнтованого* стилю. В *прототипно-орієнтованому програмуванні* поняття *клас* відсутнє, а успадкування здійснюється шляхом клонування наявного примірника *об'єкта* – ***прототипу***. Зазвичай *JavaScript* використовується як вбудована мова для програмного доступу до об'єктів. Найбільш широке застосування знаходить у браузерах як мова сценаріїв для додання *web*-сторінкам *інтерактивності*. Сучасний *JavaScript* є безпечною мовою програмування загального призначення. Вона не надає низькорівневих засобів роботи з пам'яттю і процесором, оскільки із самого початку була орієнтована на браузери, у яких це не потрібно.

Втілення скриптів мовою *JavaScript* (*js*-скриптів) до *html*-документа здійснюється за допомогою тегу `<SCRIPT>`. Він дає змогу розташовувати код скрипту в тілі *html*-документа в межах пари тегів `<SCRIPT>…</SCRIPT>`, а також підключати зовнішні файли скриптів, вказавши *URL* файлу за допомогою атрибута `src`. Щоб скрипт визначався саме як скрипт мовою *JavaScript*, використовується атрибут `type`, у якому вказується мова скрипту `"text/javascript"` в межах дескриптора, наприклад:

```
<SCRIPT type="text/javascript">…</SCRIPT>
```

При підключенні зовнішнього файлу *js*-скриптів, наприклад з ім'ям `common.js`, конструкція матиме вигляд:

```
<SCRIPT type="text/javascript" src="common.js"></SCRIPT>
```

5.1.1. Змінні, типи даних, керуючі конструкції

Синтаксис мови *JavaScript* подібний до синтаксису *C*. Але, оскільки *JavaScript* є мовою з динамічною типізацією даних, для роботи програми, як і в *PHP*, не потрібно описувати змінні, також будь-який скрипт може починатися безпосередньо з оператора. Для оголошення змінної в *JavaScript* можна лише присвоїти їй значення.

Але й можна оголосити змінну за допомогою ключового слова `var`. Якщо змінна оголошена першим присвоюванням значення чи за допомогою ключового слова `var` у межах скрипту, але за межами будь-якої з функцій, вона вважається глобальною. Якщо змінна оголошена в межах функції шляхом присвоєння значення, вона також буде глобальною. Але, якщо змінна оголошена в межах функції за допомогою ключового слова `var`, вона буде локальною, тобто доступною у межах поточної функції, і не буде мати нічого спільного з однойменною змінною, що була оголошена в межах іншої функції чи в межах усього скрипту, за її межами. Ця особливість є специфічною для *JavaScript*.

JavaScript визначає тип змінної за її контентом. Але масиви в *JavaScript* вважаються об'єктами і їх оголошення здійснюється наслідуванням певного класу, наприклад:

```
var Map = new Array();
```

Якщо змінній присвоюється чисельне значення, для неї автоматично встановлюється тип `number`. Якщо змінній присвоюється значення у вигляді тексту в лапках, для неї автоматично встановлюється тип `string`. Якщо змінна оголошується як об'єкт, її тип буде `object`. Окрім цих типів, у *JavaScript* використовується ще й такий тип, як `boolean`, який набуває значення `true` (істина) або `false` (фальш).

Так само як і *PHP*, *JavaScript* може змінювати тип змінної за перебігом виконання програми. Ця особливість скриптових мов *web*-програмування робить їх гнучкими у використанні. Розглянемо приклад:

```
<script type="text/javascript">
  var a = new Array();
  a[0]=1; a[1]=2; a[2]=a[0]+a[1];
  a='Hello!';
</script>
```

У цьому прикладі змінна `a` створена як масив з типом `object`, далі кожен член масиву з індексами 0, 1, 2 набуде відповідних

значень, що присвоюються в наступному рядку. Кожен член масиву набуде тип `number`. Далі, в момент, коли змінній `a` буде присвоєно текстове значення у вигляді рядку, масив зруйнується, а тип змінної `a` стане `string`.

Керуючі конструкції *JavaScript* майже такі самі, як і у *PHP*, але деякі з них мають відмінні риси в синтаксисі. Конструкції також зазвичай закінчуються символом крапка з комою – «`;`», можуть бути згруповані в групу, формуючи групу конструкцій, взятих в фігурні дужки «`{...}`». У *JavaScript* існує три основні групи керуючих конструкцій: конструкції умови (`if…else if…else`), конструкції циклів (`while`, `do…while`, `for`, `for…in`), конструкції вибору (`switch…case`).

Конструкції умови мають такий синтаксис:

```
if(умова) вираз;
if(умова) {вираз1; вираз2; …; виразN;}
```

Якщо потрібно виконати дію у разі невиконання умови, синтаксис має вигляд:

```
if(умова) вираз1; else вираз2;
```

або

```
if(умова) {вираз1; вираз2; …; виразN;}
else {вираз_A; вираз_B; …; вираз_Z;}
```

Якщо перевіряється декілька умов, синтаксис має вигляд:

```
if(умова1) вираз1; else if(умова2) вираз2; else вираз3;
```

або

```
if(умова1) {вираз11; вираз12; …;}
else if(умова2) {вираз21; вираз22; …;}
else {вираз_A; вираз_B; …;}
```

Потрібно звернути увагу на те, що, на відміну від *PHP*, оператор `else if` пишеться з проміжком між `else` та `if`.

Приклад:

```
<script type="text/javascript">
  elem=document.getElementById('canvas');
  var font='#ffffff';
```

```
  if(bgcolor=='red') bgcolor='#ff0000';
  else if(bgcolor=='green') bgcolor='#008000';
  else if(bgcolor=='blue') bgcolor='#0000ff';
  else {bgcolor='808080'; font='#000000';};
  elem.style.color=font;       //Колір шрифту
  elem.style.background=bgcolor;   //Колір фону
</script>
```

JavaScript підтримує чотири види циклів: цикл із передумовою (`while`), цикл із постумовою (`do…while`), цикл із лічильником (`for`), спеціальний цикл перебору властивостей об'єкта (`for…in`). У разі використання циклів є можливість використання операторів `break` і `continue`. Перший із них перериває роботу всього циклу, а другий – тільки поточної ітерації. Синтаксис циклу з передумовою має вигляд:

```
while(логічний вираз) {тіло циклу;}
```

Приклад:

```
<script type="text/javascript">
  var x=0;
  while(x<100) {y=2*x; document.write(x+'
'+y+'<br>'); x++;}
</script>
```

Синтаксис циклу з постумовою має вигляд:

```
do{тіло циклу;} while(логічний вираз);
```

На відміну від циклу `while`, цикл `do…while` перевіряє значення виразу після кожної ітерації.

Цикл із лічильником використовується для виконання тіла циклу певну кількість разів. Синтаксис циклу `for` має вигляд:

```
for(ініціалізація; умова; приростення) {тіло циклу;}
```

Цикл `for` починає свою роботу з ініціалізації змінної лічильника. Після цього перевіряється умова, якщо вона виконується (`true`), то виконується тіло циклу. Після того як буде виконаний останній оператор тіла, виконуються приростення лічильника й перевіряється умова. Якщо умова виконується, знову виконується

тіло циклу і наступне прирощення. Якщо – ні, виконання циклу припиняється.

Приклад:

```
<script type="text/javascript">
  document.write('№<br>');
  for(var i=0; i<9; i++)
  {if(i==0) continue; else
document.write(i+'<br>');}
</script>
```

У *JavaScript* використовується спеціальний цикл для перебору властивостей об'єкта `for…in`. Синтаксис цього циклу має вигляд:

```
for(властивість in об'єкт) {тіло циклу;}
```

Цей цикл перебирає по черзі кожну властивість об'єкта та в кожній ітерації для поточної властивості виконує тіло циклу.

Приклад:

```
<script type="text/javascript">
  var a={a:1, b:2, c:3}
  for(p in a)
{document.write(p+'='+a[p]+'<br>');}
</script>
```

Для вибору дій залежно від значення змінної, що перевіряється, так само як і в *PHP*, використовується конструкція `switch…case`. Синтаксис має вигляд:

```
switch(змінна) {
case значення1: вираз1; break;
case значення2: вираз2; break;
...
case значенняN: виразN; break;
[default: вираз за замовчанням; break;]
}
```

Приклад:

```
<script type="text/javascript">
var found =
Number(document.frmData.found.value);
 switch(found) {
```

```
  case 1: str='стовпчастий'; break;
  case 2: str='стрічковий'; break;
  case 3: str='плитний'; break;
  default: str='пальовий';
 }
 alert('Для будівлі обрано '+str+' фундамент.');
</script>
```

5.1.2. Функції, об’єкти та методи, властивості об’єктів

JavaScript-функція – це відокремлений від решти програми сценарій під власним ім’ям. Використовуючи це ім’я, інший сценарій може викликати його в будь-який момент будь-яку кількість разів. У разі виклику функції їй можна передати значення, які називаються *аргументами*. Аргументи можна використовувати як змінні вмежах поточної функції.

Функція *JavaScript* може бути оголошена в будь-якій частині скрипту до місця її першого використання. Їй не потрібно ніякого попереднього оголошення, як і в *PHP* або *C*. Синтаксис оголошення функцій має вигляд:

```
function Ім’я_функції(аргумент1,аргумент2,...)
{тіло функції;}
```

Оголошення функції починається службовим словом `function`, потім вказується ім’я функції, після імені функції – список аргументів у дужках. Тіло функції береться у фігурні дужки і може містити будь-яку кількість операторів. Отже, якщо функція повинна повертати результат роботи до основної програми, використовується оператор `return`. Також оператор `return` може застосовуватися в разі виходу з функції без повернення результату. У цьому випадку після оператора нічого не вказується.

Як і в усіх мовах програмування, основаних на C, використовується декілька способів виклику функції. Якщо функція виконується з поверненням результату, її виклик здійснюється через присвоєння, якщо функція виконується без повернення результату, вона викликається лише її застосуванням.

Приклад:

```
<script type="text/javascript">
  function area(l,b) {a=l*b; return a;}
  A=area(5,2);
  alert('Площа A='+A+'м2');
</script>
```

У *JavaScript* використовується ряд вбудованих функцій (методів), однією з них є функція `alert()`, вона виводить модальне вікно з повідомленням, текст якого вказаний як аргумент. Повний список цих функцій (методів) можна знайти в будь-якому довіднику з *JavaScript*, наприклад [9].

Кожна функція в *JavaScript* є об'єктом класу `Function`. Усі аргументи, що передані функції, інтерпретуються як імена параметрів для створюваної функції в порядку їх перерахування. Синтаксис створення функції з використанням об'єкта `Function` має вигляд:

```
var Ім'я_функції = new
Function(арг1[,арг2,...],тіло_функції);
```

Тут для оголошення об'єкта (нової функції), створюваного на базі класу `Function`, використовується оператор `new`.

Приклад:

```
<script type="text/javascript">
  var area = new Function('a', 'b', 'return
a*b');
  A=area(5,2);
  alert('Площа A='+A+'м2');
</script>
```

Окрім створення й використання самих функцій як об'єктів, у *JavaScript* передбачена можливість створення інших об'єктів користувача та їх методів за допомогою оператора `function`. Цей підхід має назву об'єктно-орієнтоване програмування в функціональному стилі.

Приклад:

```
<script type="text/javascript">
  function Plate(l,w,h)
```

```
    {
     this.area=function() {a=l*w; return a;}
     this.volume=function() {v=a*h; return v;}
    }
    plate = new Plate(6,2,0.2);
    A=plate.area(); V=plate.volume();
    alert('Площа плити '+A+' м2, об'єм плити '+V+'
  м3.');
  </script>
```

У наведеному прикладі службовим словом `function` оголошується клас `Plate()` з параметрами `l`, `w`, `h`. Оператором `new` створюється об'єкт `plate` (плита) як екземпляр класу `Plate()` зі значеннями параметрів `l=6`, `w=2`, `h=0.2`. За допомогою методів `area()` і `volume()` об'єкта `plate` визначається площа плити та її об'єм. Методи `area()` і `volume()` описані в класі `Plate()` також за допомогою оператора `function`. Для опису методів використовується оператор `this`, він повертає посилання на об'єкт, який є поточним контекстом виклику. Звертання до методів і властивостей здійснюється через крапку.

Під час опису класу об'єкта можна також задавати його властивості. Це здійснюється за допомогою службового слова `var`. Розглянемо приклад оголошення класу `Plate()` з використанням властивостей:

```
<script type="text/javascript">
  function Plate()
  {
   var length; var width; var height;
   this.area=function()
   {l=this.length; w=this.width; a=l*w; return a;}
   this.volume=function()
   {a=this.area(); h=this.height; v=a*h; return v;}
  }
  plate = new Plate();
  plate.length=6;    // м.
  plate.width=1;     // м.
```

```
  plate.height=0.2; // м.
  A=plate.area(); V=plate.volume();
  alert('Площа плити '+A+' м2, об'єм плити '+V+' м3.');
</script>
```

У наведеному прикладі параметри об'єкта задаються властивостями `length`, `width`, `height`. У цьому випадку, на відміну від попереднього прикладу, об'єкт `plate` створюється з властивостями, значення яких можна змінювати в процесі виконання скрипту, тобто зі змінними параметрами. У попередньому прикладі об'єкт `plate` створюється зі сталими значеннями параметрів.

В об'єктно-орієнтованому програмуванні методами об'єктів є їх функції, а властивостями – їх змінні.

Окрім функціонального стилю, що дає змогу створювати класи об'єктів за допомогою `function`, у *JavaScript* використовується прототипний стиль, де замість класу створюється примірник об'єкта. Синтаксис створення такого об'єкта має вигляд:

```
var ім'я_об'єкта = {
  властивість1: значення1,
  властивість2: значення2,
...
  метод1() {тіло методу;}
  метод2() {тіло методу;}
...
}
```

Приклад:

```
<script type="text/javascript">
  var cylinder = {
   radius: 0,
   height: 0,
   area()
   {r=this.radius; a=Math.PI*Math.pow(r,2); return a;},
   volume()
   {a=this.area(); h=this.height; v=a*h; return v;}
  }
  cylinder.radius=0.2; //м.
  cylinder.height=2;  //м.
```

```
  V=cylinder.volume();
  document.write('Об’єм циліндру V='+V+' м3.');
</script>
```

У наведеному прикладі створюється примірник об’єкта `cylinder` (циліндр), у якому описуються властивості `radius` (радіус), `height` (вистота) і методи `area()`, `volume()`. Для ініціалізації властивостей їм присвоюються початкові значення, які потім змінюються в міру виконання скрипту. Метод `area()` визначає площину основи циліндру, метод `volume()` – об’єм циліндра за заданих властивостей `radius` і `height`.

5.1.3. Об’єкти JavaScript

У *JavaScript* майже все є об’єктом. Усі примітивні типи, за винятком `null` і `undefined`, обробляються як об’єкти. *JavaScript* використовує три види об’єктів: об’єкти користувача (створення яких розглянуто вище), базові об’єкти мови й об’єкти з боку клієнта – об’єкти браузера та об’єкти, які представляють елементи *html*-документа *web*-сторінки у браузері. Є десять основних типів базових об’єктів мови. Серед них такі об’єкти, як `Global`, `Object`, `String`, `RegExp`, `Array`, `Date`, `Math`, `Boolean`, `Number`, `Function`. Для кожного з них існує свій набір методів і властивостей. На основі багатьох із них можна створювати нові об’єкти за допомогою оператора `new`.

Об’єкт `Global` призначений для збирання глобальних функцій і констант до одного об’єкта. Він ніколи не використовується безпосередньо, і його неможливо створити за допомогою оператора `new`. Він створюється за ініціалізації обробника скриптів, і його методи та властивості стають доступними негайно.

Об’єкт `Object` надає функції, загальні для всіх об’єктів *JavaScript*. Крім того, він використовується як асоціативний масив. Синтаксис створення об’єкта на базі класу `Object` має вигляд:

```
var obj = new Object( [value] );
```

Необов’язковий параметр `value` може бути як звичайною змінною, так і підпорядкованим об’єктом.

Об’єкт `String` дає змогу керувати текстовими рядками, форматувати їх і виконувати пошук підрядків у рядках. Він зазвичай створюється неявно, за допомогою текстових літералів. Синтаксис створення об’єкта на базі класу `String` має вигляд:

```
var str = new String(['текстовий рядок']);
```

Об’єкт `RegExp` – це вбудований глобальний об’єкт, у якому зберігається інформація про результати збігів шаблону регулярного виразу. Синтаксис створення об’єкта на базі класу `RegExp` має вигляд:

```
var expr = new RegExp(pattern[,flags]);
```

`pattern` задає шаблон пошуку (текст регулярного виразу), `flags` задає спосіб пошуку за шаблоном.

Об’єкт `Array` підтримує створення масивів будь-якого типу даних. Масив пронумерованих елементів також може слугувати стеком або чергою. Клас `Array` рекомендується використовувати тільки для масивів із числовими індексами. Для асоціативних масивів *JavaScript* застосовує тип `Object`. Синтаксис створення об’єкта на базі класу `Array` має вигляд:

```
var array = new Array([size]);
```

або

```
var array = new Array([elem0[,elem1,elem2, ...]);
```

У першому випадку створюється масив встановленого розміру `size`, у другому – масив із набором елементів, що вказані в дужках. Отже, якщо в дужках нічого не вказувати, створиться порожній масив.

Об’єкт `Date` забезпечує основні можливості роботи з датою та часом. Він містить число, що представляє певний момент часу з точністю до мілісекунди. Діапазон дат, які можуть бути представлені об’єктом `Date`, становить близько 285 616 у кожний бік від 1 січня 1970 р. Синтаксис створення об’єкта на базі класу `Date` має вигляд:

```
var date = new Date();
var date = new Date(dateVal);
var date = new Date(year,month,day[,h,m,s,ms]);
```

У першому випадку об'єкт `date` набуде значення поточної дати. У другому, якщо параметр `dateVal` є числовим, він інтерпретується як кількість мілісекунд у форматі UTC, якщо `dateVal` є рядком, то він розбирається й інтерпретується як дата за стандартними правилами методу `Date.parse()`. У третьому – створюється об'єкт із переданими значеннями року, місяця, дня, часу, хвилини, секунди, мілісекунди.

Об'єкт `Math` надає основні математичні функції та константи. На відміну від інших глобальних об'єктів, `Math` не є конструктором і на його основі не можна створити інші об'єкти, використовуючи оператор `new`, але всі його методи й властивості доступні скрипту в будь-який момент часу. Усі методи й властивості `Math` статичні. Синтаксис використання об'єкта `Math` має вигляд:

```
Math.[method | property]
```

Наприклад, для визначення синуса кута (у радіанах) використовується метод `sin()`, вираз матиме вигляд:

```
var a = Math.sin(3.14);
```

для визначення значення числа π використовується властивість `PI`, вираз матиме вигляд:

```
var pi = Math.PI;
```

Об'єкт `Boolean` створює нове логічне значення. Він є оболонкою типу даних `boolean`. *JavaScript* неявно використовує цей об'єкт за перетворення логічного типу даних boolean в об'єкт `Boolean`. Синтаксис створення об'єкта на базі класу `Boolean` має вигляд:

```
var bool = new Boolean([value]);
```

Якщо аргумент `value` відсутній, має значення false, 0, null, NaN або є символом нового рядка, то початковим значенням об'єкта

`Boolean` є false. В іншому випадку початковим значенням є значення true.

Об'єкт `Number` представляє число будь-якого роду. У разі потреби *JavaScript* сам створює об'єкти `Number` із числових значень. Основне призначення об'єктів `Number` полягає в збиранні властивостей до одного об'єкта, а також у перетворенні до числа чисельного рядка, якщо він є у текстовому форматі. Синтаксис створення об'єкта на базі класу `Number` має вигляд:

```
var num = new Number(value);
```

Обов'язковий параметр `value` має чисельне значення.

Об'єкт `Function` створює нову функцію. Кожна функція в *JavaScript* є об'єктом класу `Function`. Усі аргументи, передані функції, інтерпретуються як імена параметрів для створюваної функції в порядку їх перерахування. Виклик конструктора `Function` як функції здійснюється за допомогою оператора `new`. Синтаксис створення об'єкта на базі класу `Function` має вигляд:

```
var fun = new Function(arg1[,arg2,...],function_body);
```

Список аргументів `arg1,arg2,...`, які приймаються функцією, є необов'язковими параметрами, що використовуються в тілі функції `function_body`.

Повний список усіх методів і властивостей розглянутих базових об'єктів і їх призначення наведений у довіднику [9].

Програми зазвичай розробляються у спеціальних комерційних цілях. Наприклад, система менеджменту призначається для обробки замовлень з боку клієнтів. Комерційні прикладні задачі, на додаток до вимог кінцевого користувача, формують «простір проблем» або те, на що посилаються як на предметну область програмного забезпечення. Ця область також містить і таке поняття, як графічний інтерфейс користувача (GUI).

У конструкції об'єктно-орієнтованих додатків об'єкти є центральними логічними блоками. Найчастіше об'єкти в коді програми – це уявлення реальних об'єктів, що розташовані в просторі проблем. Об'єктом може бути будь-яке поняття:

матеріальна або видима річ у предметній області; абстрактна концепція, що існує тільки в розумі розробника; видимі об'єкти графічного інтерфейсу користувача (GUI-об'єкти), наприклад вікна, фрейми, поля введення, кнопки тощо. У браузері такими об'єктами є елементи інтерфейсу як самого браузера, так і ті, з яких складається *web*-сторінка у його вікні.

У *JavaScript* вікно браузера, документ і всі його елементи представлені відповідними об'єктами для їх керування. Їх поділяють на об'єкти верхнього, другого, третього і четвертого рівнів.

До об'єктів верхнього рівня належать такі об'єкти, як `Window`, `Navigator`, `Screen`, `History`, `Location`, `Document`, `Frame`.

Об'єкт `Window` є глобальним об'єктом і вважається головним в ієрархії об'єктів з боку клієнта у JavaSrcipt. Він містить усі інші клієнтські об'єкти, крім об'єкта `Navigator`. Цей об'єкт, власне, являє собою вікно браузера. Звертання чи виклик методів і властивостей здійснюється за правилами звертання *JavaScript* до методів і об'єктів, тобто через крапку. Наприклад, якщо потрібно викликати метод відкриття нового екземпляра вікна `open()`, звертання матиме вигляд:

```
window.open();
```

Об'єкт `Navigator` містить інформацію про браузер відвідувача. За допомогою цього об'єкта можна отримати інформацію про ім'я, версію браузера, а також іншу додаткову інформацію. Звертання до об'єкта `Navigator` здійснюється таким чином:

```
window.navigator,
```

а може мати й скорочену форму, без використання префіксу `window`. Наприклад, для звертання до властивості `appName`, яка визначає ім'я браузера, звертання матиме вигляд:

```
navigator.appName
```

Об'єкт `Screen` містить загальну інформацію про екран монітору комп'ютера чи мобільного пристрою. Він просто надає

відомості про графічні параметри клієнтської системи за межами вікна браузера, як-от ширина, висота (у пікселях). Наприклад, для визначення ширини та висоти екрана звертання до відповідних властивостей матиме такий вигляд:

```
w=screen.width; h=screen.height;
```

Об’єкт `History` надає журнал навігації за весь час роботи з певним вікном, а також дає змогу здійснювати перехід назад чи вперед за історією відвідування сторінок. Наприклад, якщо потрібно повернутися до попередньої сторінки за допомогою метода `back()`, виклик методу матиме вигляд:

```
history.back();
```

Об’єкт `Location` надає інформацію про поточний *URL* і дає змогу *JavaScript* перенаправити відвідувача на іншу сторінку за її *URL*. Звертання до об’єкта `Location` здійснюється таким чином:

```
window.location
```

У разі якщо потрібно перезавантажити сторінку, використовується метод `reload()`. Виклик методу матиме вигляд:

```
window.location.reload();
```

Для визначення *URL* поточної сторінки використовується властивість `href`. Звертання до неї матиме вигляд:

```
href=window.location.href;
```

Об’єкт `Document` використовується для отримання, зміни, додавання вмісту в *html*-документі, а також для обробки подій у ньому. Він забезпечує доступ до всіх елементів *web*-сторінки і слугує точкою входу для доступу до них. Наприклад, для доступу до елемента сторінки `<BODY>` використовується звертання:

```
document.body
```

Для доступу до інших елементів сторінки, які можуть бути в кількості, відмінної від одиниці, використовуються відповідні масиви. Наприклад, якщо на сторінці буде розміщено декілька зображень (об’єктів `Image`), а нам потрібно звернутися до другого з них, то звертання матиме вигляд:

```
document.images[1]
```

Оскільки в *JavaScript*, як і в *PHP*, нумерація починається з 0, то індекс другого елемента буде 1. Якщо потрібно зібрати масив усіх зображень сторінки у змінну, що зберігатиме цей масив, здійснюється присвоєння:

```
imgs=document.images;
```

Також до будь-якого елемента можна звернутися, використовуючи його ідентифікатор (якщо встановлений), за допомогою методу `getElementById()`, а зібрати список однотипних елементів у масив можна за допомогою методів: `getElementsByTagName()` – збирає масив елементів з однаковим тегом; `getElementsByClassName()` – збирає масив елементів з однаковим ім'ям класу *css*-стилю. Наприклад, якщо потрібно зібрати всі елементи `<DIV>` у масив, здійснюється така дія:

```
divs=document.getElementsByTagName('div');
```

а потім, якщо потрібно отримати другий із них, здійснюється присвоєння:

```
curdiv=divs[1];
```

Отримавши таким чином елемент, далі можна продовжити керувати ним за допомогою методів і властивостей псевдооб'єкта `Element`. Цей об'єкт надає методи та властивості для роботи з усіма видами елементів.

Об'єкт `Frame` – це набір фреймів об'єкта `Window`, у яких паралельно можуть виводитися різні *html*-документи. Цей фрейм відповідає елементу `<FRAMESET>`.

Повний список методів і властивостей об'єктів верхнього рівня та їх призначення наведений у довіднику [9].

До об'єктів другого рівня належать об'єкти, які відповідають елементам тіла документа і є підлеглими об'єкта верхнього рівня `Document`. Основними з них є об'єкти: `Anchor`, `Applet`, `Embed`, `Form`, `Image`, `Link`. Вони відповідають *html*-елементам `<A>`, `<APPLET>`, `<EMBED>`, `<FORM>`, `<IMG>`, `<LINK>`. Усі інші елементи ідентифікуються як об'єкти за допомогою пошукових методів

об’єкта Document getElementById(), getElementsByTagName(), getElementsByClassName(). Повний список усіх методів і властивостей основних об’єктів другого рівня та їх призначення наведений у довіднику [9]. Для інших застосовуються методи та властивості псевдооб’єкта Element.

До об’єктів третього рівня належать об’єкти, які відповідають елементам, що належать формі <FORM>, тобто є підпорядкованими об’єктами об’єкта Form. Серед них такі об’єкти, як Button, Checkbox, FileUpload, Hidden, Password, Radio, Reset, Submit, Text, Select, Textarea. Об’єкти Button, Checkbox, FileUpload, Hidden, Password, Radio, Reset, Submit, Text відповідають елементу керування <INPUT type=""> зі значенням атрибута type, відповідно button, checkbox, file, hidden, password, radio, reset, submit, text. Об’єкт Select відповідає елементу керування <SELECT>, об’єкт Textarea – елементу керування <TEXTAREA>. Усі методи та властивості цих об’єктів наведені в довіднику [9].

До об’єктів четвертого рівня входить об’єкт Option, що належить об’єкту Select. Він відповідає елементу <OPTION> списку елемента <SELECT>.

На відміну від інших об’єктів, об’єкт Form і всі підлеглі йому об’єкти третього рівня мають властивість name, яка задає ім’я змінної відповідного елемента керування форми. Також майже всі вони, окрім Select, мають властивість value, що містить значення змінної name. Для об’єкта Select значення змінної name визначається властивістю value підлеглих об’єктів Option. Усі властивості об’єкта Option також наведені в довіднику [9].

5.1.4. Об’єктна модель браузера та документа

Браузер надає доступ до ієрархії об’єктів, що представлена у вигляді об’єктної моделі, на вершині якої перебуває глобальний об’єкт Window. Усі інші об’єкти поділяються на 3 групи (рис. 5.1).

Перша група – базові об’єкти *JavaScript*, друга група – об’єкти об’єктної моделі браузера *BOM (Browser Object Model)*, третя група – об’єкти об’єктної моделі документа *DOM (Document Object Model)*.

BOM – це об’єкти для роботи із чим завгодно, крім документа. Доступ до фреймів, запитів сервера, функцій `alert()`, `confirm()`, `prompt()` – усе це *BOM*. Усі можливості *BOM* стандартизовані в *HTML5*.

Основним інструментом роботи та динамічних змін на сторінці є об’єктна модель документа *DOM*. Відповідно до *DOM*-моделі, документ є деревом. Кожний *html*-тег утворює вузол дерева з типом *«Елемент»*. Вкладені до нього теги стають дочірніми вузлами. Для представлення тексту створюються вузли з типом *«Текст»*. Інакше кажучи, *DOM* – це подання документа у вигляді дерева тегів, яке доступне для зміни через *JavaScript*. Для того щоб змінити вузол *DOM*, потрібно спочатку його отримати. Доступ до *DOM*-у починається з `document`. Звідти можна дістатися до будь-яких інших вузлів.

Увійти в корінь дерева елементів сторінки можна двома шляхами: за допомогою `document.documentElement` – ця властивість посилається на *DOM*-об’єкт для тегу `<HTML>`, або за допомогою `document.body`, що відповідає тегу `<BODY>`.

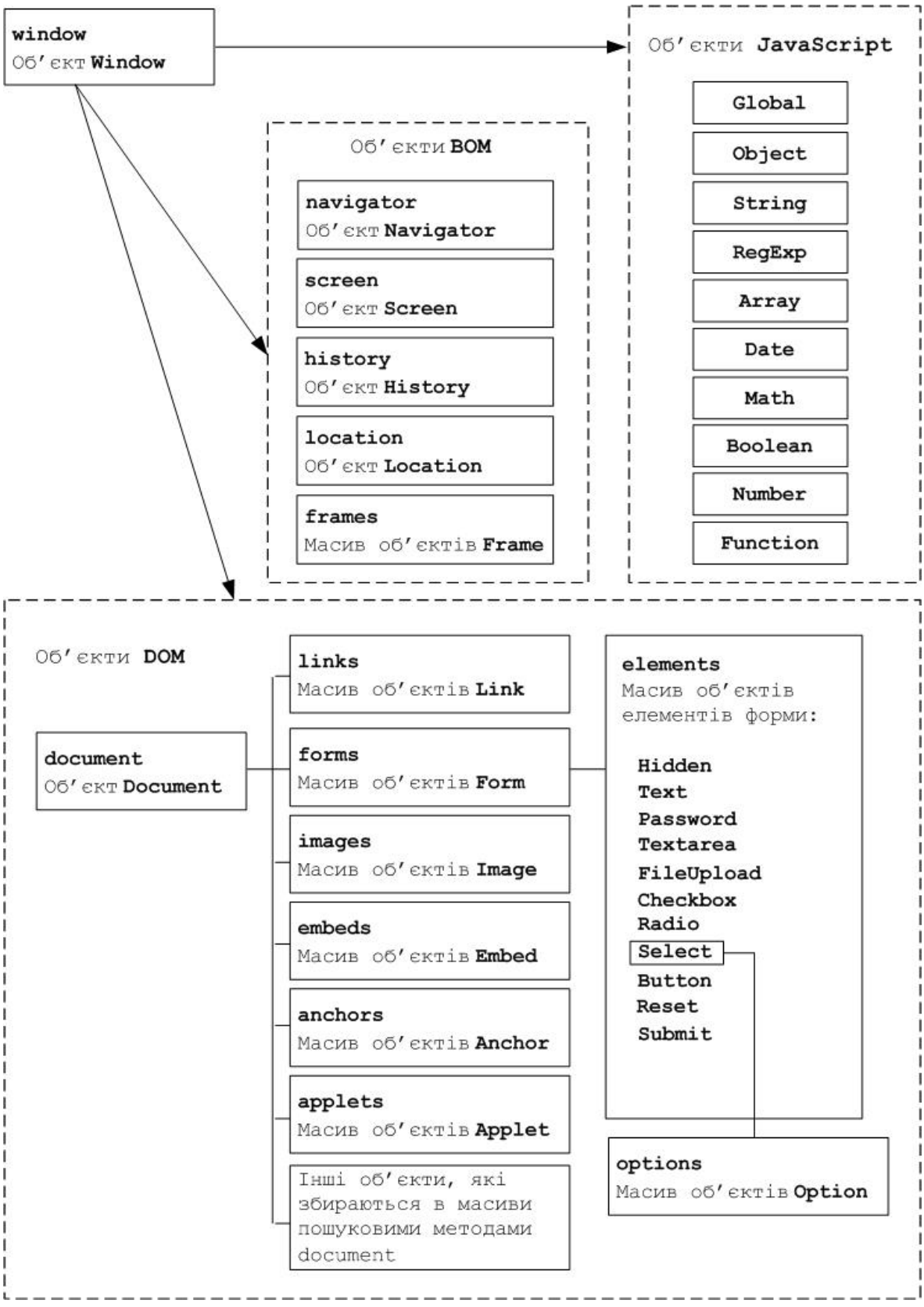


Рис. 5.1

Об'єкт `Window` містить всі інші клієнтські об'єкти, які поділяються на групи.

Перша група – базові об’єкти *JavaSrcipt*, а саме:

`Global` – призначений для збирання глобальних функцій і констант до одного об’єкта;

`Object` – надає функції, загальні для всіх об’єктів *JavaScript*;

`String` – дає змогу керувати текстовими рядками;

`RegExp` – це вбудований глобальний об’єкт, у якому зберігається інформація про результати збігів шаблону регулярного виразу;

`Array` – підтримує створення масивів будь-якого типу даних;

`Date` – забезпечує основні можливості роботи з датою та часом;

`Math` – надає основні математичні функції та константи;

`Boolean` – створює нове логічне значення;

`Number` – представляє число будь-якого роду;

`Function` – створює нову функцію.

Друга група – об’єкти верхнього рівня об’єктної моделі браузера *BOM (Browser Object Model)*:

`Navigator` – містить інформацію про браузер відвідувача;

`Screen` – містить загальну інформацію про екран монітору комп’ютера чи мобільного пристрою;

`History` – надає журнал навігації за весь час роботи з певним вікном, а також дає змогу здійснювати перехід за історією відвідування сторінок;

`Location` – надає інформацію про поточний *URL* і дає можливість *JavaScript* перенаправити відвідувача на іншу сторінку за її *URL*;

`frames` – має набір фреймів **Frame** об’єкта **Window**, у яких паралельно можуть виводитися різні *html*-документи;

`Document` – використовується для отримання, зміни, додавання вмісту в *html*-документі, а також для обробки подій у ньому.

Об’єкт `Document` є головним об’єктом об’єктної моделі документа *DOM (Document Object Model)*, об’єкти якої становлять третю групу,та є, зі свого боку, об’єктами другого рівня.

Третя група – об’єкти об’єктної моделі документа *DOM*:

`links` – масив об’єктів `Link`, тобто набір *html*-елементів `link`;

`forms` – масив об’єктів `Form`, тобто набір *html*-елементів `form`;

`images` – масив об’єктів `Image`, тобто набір *html*-елементів `img`;

`embeds` – масив об’єктів `Embed`, тобто набір *html*-елементів `embed`;

`anchors` – масив об’єктів `Anchor`, тобто набір *html*-елементів `a`;

`applets` – масив об’єктів `Applet`, тобто набір *html*-елементів `applet`.

Зі свого боку, об’єкт `Form` складається з масиву елементів форми `elements`, які є об’єктами третього рівня, до яких належать об’єкти: `Hidden`, `Text`, …, `Select`, …, `Submit`.

Об’єкт `Select` складається з масиву елементів вибору `options`, які є об’єктами четвертого рівня, до яких належать об’єкти `Option`.

За належністю один до одного вузли дерева *DOM* відносно поточного вузла поділяються на власне поточний – `Node`, батьківські – `parentNode`, дочірні – `childNodes`, `children`, `firstChild`, `lastChild`, сусідні – `previousSibling`, `nextSibling`. За допомогою цих псевдомасивів і навігаційних посилань можна здійснювати переміщення по вузлах дерева DOM, а також визначати й керувати властивостями тих чи інших елементів. Усі навігаційні посилання доступні тільки для читання.

Наприклад, якщо потрібно отримати назву тегу батьківського елемента, вираз матиме вигляд:

```
tag=node.parentNode.tagName;
```

Якщо потрібно отримати ім’я класу *css*-стилю попереднього сусіднього елемента, у цьому випадку вираз матиме вигляд:

```
name=node.previousSibling.className;
```

5.1.5. Події у JavaScript. Обробка подій

Майже все, що відбувається у вікні браузера, є *подією*. Практично всі *JavaScript*-додатки виконують ті чи інші дії, реагуючи на різні події. ***Подія*** – це сигнал від браузера про те, що щось сталося. Існує багато видів подій. Серед них – події вікна та тіла документа, події форм і елементів керування форми, події миші, події клавіатури, події доторкання. Назви подій та їх призначення наведені в таблиці 5.1.

Основна цінність *JavaScript* у його інтеграції зі сторінкою. Будь-який документ або *DOM*-елемент вміє ініціювати різні події, а на подію, знаючи його ім’я, можна призначити *обробник*. Обробники дають можливість організувати реакцію на виникнення подій зі сценарію. При цьому відповідний обробник вказується як атрибут елемента *html*-документа. Обробником події є *функція-обробник*, що призначається для виконання обробки. Для того щоб скрипт реагував на подію, потрібно призначити хоча б одну *функцію-обробник*. Зауважимо, що *JavaScript* – це однопоточна мова, тому обробники завжди виконуються послідовно.

Таблиця 5.1

Основні події та їх призначення

Атрибут або властивість	Назва події, метода	Призначення
Події вікна (`window`), **тіла документа** (`document.body`)		
`onload`	`load`	Завершення відкриття вікна, завантаження сторінки
`onunload`	`unload`	Закриття вікна чи завантаження документа
`onresize`	`resize`	Зміна розміру вікна
`onscroll`	`scroll`	Прокрутка сторінки у вікні чи в контейнері
`onmove`	`move`	Переміщення вікна

Таблиця 5.1 (закінчення)

Події форм і елементів керування форми		
onfocus	focus	Отримання елементом фокуса
onblur	blur	Втрата поточним елементом фокуса
onchange	change	Зміна значення елемента форми в разі втрати фокуса
onselect	select	Виділення тексту в поточному елементі
onsubmit	submit	Відправлення даних форми
onreset	reset	Очищення даних форми
Події миші		
onclick	click	Натискання лівою клавішею миші на елементі
ondblclick	dblclick	Подвійне натискання лівою клавішею миші на елементі
onmousedown	mousedown	Натиснута кнопка миші в межах поточного елемента
onmouseup	mouseup	Відпущена кнопка миші в межах поточного елемента
onmousemove	mousemove	Переміщення курсора миші в межах поточного елемента
onmouseover	mouseover	Курсор миші наведений на поточний елемент
onmouseout	mouseout	Курсор миші виведений за межі поточного елемента
Події клавіатури		
onkeydown	keydown	Натиснута клавіша на клавіатурі
onkeyup	keyup	Відпущена клавіша на клавіатурі
onkeypress	keypress	Натиснута і відпущена клавіша на клавіатурі
Події доторкання		
ontouchstart	touchstart	Початок доторкання пальцем до екрана
ontouchend	touchend	Закінчення доторкання пальцем до екрана
ontouchmove	touchmove	Переміщення пальцем по екрану

Існує декілька способів призначення обробника на конкретну подію елемента. Перший із них – за допомогою відповідного атрибута події тегу. Наприклад, якщо з натисканням на кнопку, яка належить формі, має виконуватися запит цієї форми, запис у *html*-коді кнопки матиме вигляд:

```
<input type="button" value="Виконати"
    onclick="this.form.submit();">
```

Другий спосіб – через властивості об'єкта. Можна призначити обробник, використовуючи властивість *DOM*-елемента, яка відповідає певній події. Тут, на відміну від попереднього способу, все відбувається в межах скрипту. Але потрібно спочатку ідентифікувати елемент, над яким виконуватиметься дія, як об'єкт *JavaScript*, а потім скористатися його властивістю. Розглянемо попередній приклад у цій постановці. У *html*-коді кнопка матиме такий вигляд:

```
<input id="todo" type="button" value="Виконати">
```

У сценарії *js*-скрипту фрагмент коду буде такий:

```
…
btn=document.getElementById('todo');
btn.onclick = function() {this.form.submit();}
…
```

У цьому прикладі ідентифікація елемента здійснюється за допомогою ідентифікатора `id`, але можна ідентифікувати й іншим шляхом.

Третій спосіб призначення обробника – за допомогою методів `addEventListener` і `removeEventListener`. Цей спосіб дає змогу використовувати скільки завгодно будь-яких обробників однієї події. Призначення обробника події здійснюється через виклик методу `addEventListener()` з трьома аргументами `event`, `handler`, `phase`:

```
element.addEventListener(event,handler[,phase]);
```

`event` – ім'я події; `handler` – посилання на функцію-обробник; `phase` – необов'язковий аргумент «фаза», на якій обробник повинен спрацювати.

Видалення обробника події здійснюється через виклик метода `removeEventListener()` з тими самими аргументами:

```
element.removeEventListener(event,handler[,phase]);
```

Для видалення обробника події потрібно передати саме ту функцію-обробник, яка була призначена. У цій постановці попередньо розглянутий приклад набуде вигляду:

```
…
btn=document.getElementById('todo');
btn.addEventListener('click', function()
{this.form.submit();});
…
```

Методи, які використовуються в третьому способі, відповідають стандарту *W3C* і працюють у браузерах, що його підтримують. Серед них – *Mozilla Firefox, Google Chrome, Opera, Safari, Internet Explorer 10* і вище. Для браузера *Microsoft Internet Explorer*, до 9-ї версії включно, використовуються аналогічні методи `attachEvent`, `detachEvent`.

Обробники подій можуть бути прив'язаними до об'єктів `Window`, `Document` тощо. У той момент, коли відбувається будь-яка подія, створюється об'єкт `Event`, який передається як аргумент обробнику події. Інтерфейс події об'єктної моделі документа *DOM* доступний тільки через об'єкт `Event`, який передається як аргумент до обробника події зі своїми методами та властивостями. В Internet Explorer 8 і ранніх версіях об'єкт `Event` є доступний у вигляді глобальної змінної `window.event`.

5.2. Мова запитів SQL. Робота з базами даних

Збереження й обробка інформації сучасними *інтернет*-системами не відбувається без використання баз даних. Майже всі системи управління контентом сайту для збереження інформації також використовують бази даних. Для управління даними, що зберігаються в базах даних, використовується структурована мова запитів *SQL (Structured Query Language)*, яка застосовується в контексті серверної мови *web*-програмування, де взаємодія із *системою управління базами даних (СУБД)* здійснюється за допомогою спеціальних бібліотек, а обмін інформацією – за допомогою певних функцій для роботи з базами даних.

Мова *PHP*, що розглянута вище, підтримує взаємодію з великою кількістю різних *СУБД*. Серед них – *MySQL*, *Oracle*, *PostgresSQL*, *SQLite* тощо. Традиційно *PHP* використовують спільно з *MySQL*, однак використання інших *СУБД* може бути доцільним для вирішення певних задач, якщо функціональні можливості *MySQL* недостатні.

У контексті *PHP sql*-запит конструюється у текстовій змінній, а потім передається відповідній функції для його виконання як аргумент.

Приклад:

```
<?
  $sql='UPDATE USERS SET VISITS=VISITS+1 WHERE
ID=210';
  mysql_query($sql);
  $sql='SELECT NAME FROM USERS WHERE ID=210';
  $result=mysql_query($sql);
  $name=mysql_result($result,0);
?>
```

У наведеному прикладі перший раз текстова змінна `$sql` набуде значення у вигляді sql-запиту на оновлення даних, який виконається функцією `mysql_query()`. Після цього змінна `$sql` набуде значення у вигляді sql-запиту на вибірку, що виконається функцією `mysql_query()` з поверненням результату запиту у змінну `$result`. В останньому рядку змінна `$name` отримає значення вказаного поля, вилученого з результату запиту функцією `mysql_result()`.

Список усіх функцій для роботи з базами даних, їх призначення і приклади використання наведені в online-довіднику мови *PHP* на сайті http://php.net.

Усі *sql*-запити складаються за правилами, які відповідають певній конструкції, що призначена для виконання тієї чи іншої дії. *SQL* є непроцедурною мовою програмування, яка застосовується для створення, модифікації та управління даними, що зберігаються в реляційних базах даних. Вона являє собою сукупність команд,

операторів, інструкцій, обчислюваних функцій. Згідно із загальноприйнятим стилем програмування, команди (та інші зарезервовані слова) в *SQL* рекомендується писати прописними літерами, хоча вони є нечутливими до регістру.

Команди мови *SQL* поділяються на: команди визначення даних *DDL (Data Definition Language)*; маніпуляції даними *DML (Data Manipulation Language)*; визначення доступу до даних *DCL (Data Control Language)*; управління транзакціями *TCL (Тгansaction Control Language)*. До команд *DDL* належать команди створення, зміни, знищення об'єктів – `CREATE`, `ALTER`, `DROP`. До команд *DML* – команди додавання нових даних, зміни існуючих даних, вибору даних, видалення даних – `INSERT`, `UPDATE`, `SELECT`, `DELETE`. До команд *DCL* належать команди надання дозволу на певні операції з об'єктами, відклику раніше виданих дозволів, дозволу або заборони ролі для поточного сеансу – `GRANT`, `REVOKE`, `SET ROLE`, `DENY`. До команд *TCL* належать команди підтвердження транзакції, скасування всіх змін, зроблених у контексті поточної транзакції, поділу транзакції на більш дрібні ділянки – `COMMIT`, `ROLLBACK`, `SAVEPOINT`. Кожна з команд має певну конструкцію. Розглянемо конструкції основних команд *sql*-запитів.

Створення бази даних здійснюється командою `CREATE DATABASE`. Синтаксис команди має вигляд:

```
CREATE DATABASE db_name,
```

де `db_name` – ім'я нової бази даних, що створюється. Для того щоб почати роботу зі створеною базою даних, потрібно повідомити *СУБД*, з якою базою даних ми маємо намір працювати. Здійснюється це за допомогою команди `USE`. Синтаксис команди має вигляд:

```
USE db_name
```

Створення таблиць в базі даних здійснюється командою `CREATE TABLE`. Її синтаксис має вигляд:

```
CREATE TABLE table_name
[(field type[ definition], ...)],
```

де `table_name` – ім'я нової таблиці, `field` – ім'я поля таблиці, що

створюється, type – тип поля, definition – визначення.

Для зміни структури таблиці в базі даних використовується команда ALTER TABLE. За допомогою цієї команди можна додати до таблиці нове поле (оператор ADD), змінити ім'я та тип поля (оператор CHANGE), видалити поле (оператор DROP). Синтаксис таких команд має вигляд:

```
ALTER TABLE table_name ADD new_field AFTER field;
ALTER TABLE table_name CHANGE field new_field type;
ALTER TABLE table_name DROP field;
```

Для додавання нового поля до таблиці, після оператора ADD вказується його ім'я new_field, а після ключового слова AFTER – ім'я поля field, після якого потрібно додати нове. Для зміни імені та типу певного поля після оператора CHANGE вказується поточне ім'я поля field, за ним, після проміжку, вказується нове ім'я new_field і, якщо потрібно змінити тип, новий тип – type. Для видалення з таблиці певного поля його ім'я field вказується після оператора DROP. Щоб видалити декілька полів, їх потрібно вказати через кому, з використанням перед іменем кожного з них оператора DROP.

```
ALTER TABLE tab_name DROP fld1, DROP fld2, DROP fld3
```

Для видалення таблиці з бази даних використовується команда DROP TABLE. Синтаксис команди має вигляд:

```
DROP TABLE table_name
```

Щоб видалити декілька таблиць, їх потрібно вказати через кому.

Знищення бази даних здійснюється через команду DROP DATABASE. Синтаксис має вигляд:

```
DROP DATABASE db_name
```

Додавання нових записів до таблиці в базі даних здійснюється через команду INSERT. Синтаксис має вигляд:

```
INSERT INTO table_name
[(field1[,field2,field3,...])]
```

```
VALUES (value1[,value2,value3,...])
```

Оператор INTO визначає ім'я таблиці table_name, до якої здійснюватиметься додавання даних. Після імені таблиці в дужках через кому можна вказати імена полів таблиці, але цей параметр необов'язковий. Рядок значень нового запису, що додається до таблиці, вказується після ключового слова VALUES в дужках, через кому. Варто звернути увагу на те, що кількість значень у рядку VALUES повинна збігатися з кількістю полів таблиці, інакше такий запит не виконається.

Окрім конструкції INSERT…VALUES команди додавання записів INSERT до таблиці, ще існує конструкція INSERT…SET. Вона дає змогу додавати новий запис у вигляді присвоєння полю значення. Такий синтаксис має вигляд:

```
INSERT INTO table_name
SET field1=value1[, field2=value2,
field3=value3, ...]
```

Для цієї конструкції також обов'язково вказати значення всіх полів нового запису таблиці. Після ключового слова SET через кому вказується список полів з присвоєнням їм значень нового запису.

Оновлення даних у полях записів таблиці здійснюється за допомогою команди UPDATE. Синтаксис такої команди має вигляд:

```
UPDATE table_name
SET field1=value1[, field2=value2,
field3=value3, ...]
[WHERE definition]
[LIMIT rows]
```

Після ключового слова SET через кому вказується список полів таблиці table_name з присвоєнням їм нових значень запису. Якщо потрібно задати умову, використовується ключове слово WHERE, після якого визначається умова запиту definition. Якщо треба обмежити кількість записів rows, які будуть змінені, використовується ключове слово LIMIT. На відміну від

INSERT…SET, в UPDATE для SET вказується список лише тих полів, значення яких буде змінено.

Вибір даних із таблиці здійснюється за допомогою команди SELECT. Синтаксис команди має вигляд:

```
SELECT field1[,field2,field3,...]
FROM table_name WHERE definition
[ORDER BY field1[ ASC|DESC][, field2[
ASC|DESC]], ...]
[LIMIT [offset,] rows]
```

Після оператора SELECT вказується список полів, з яких треба вибрати дані. Якщо потрібно вказати всі поля, замість списку вказується символ «*». Після ключового слова FROM вказується ім'я таблиці, з якої слід вибрати дані. Якщо потрібно задати умову відбору, використовується ключове слово WHERE, після якого визначається умова definition. Якщо потрібно здійснити сортування даних результату відбору, використовується ключова фраза ORDER BY. Після неї вказуються поля, за якими треба здійснити сортування, та напрям сортування: ASC – у прямому напрямку (від А до Я), DESC – у зворотному (від Я до А). Якщо потрібно обмежити кількість записів rows, що вибираються, та почати вибірку з якоїсь певної offset, використовується ключове слово LIMIT, після якого вказуються ці параметри.

Попередньо розглянута конструкція команди SELECT дає змогу здійснювати вибір даних з однієї таблиці. Але за допомогою SELECT можна здійснювати вибір даних відразу з декількох взаємозв'язаних таблиць. У цьому випадку конструкція команди SELECT матиме вигляд:

```
SELECT T1.field1 AS fld1[, T1.field2 AS fld2, ...]
[,T2.field1 AS fldA[, T2.field2 AS fldB,...]]
FROM table1 T1, table2 T2[,...]
WHERE definition
[ORDER BY fld1[ ASC|DESC][, fd2[ ASC|DESC]], ...]
[LIMIT [offset,] rows]
```

Тут, на відміну від попередньої конструкції, таблицям присвоюються псевдоімена T1, T2, …, які вказуються в рядку FROM після проміжку від вихідного імені таблиці. Ці псевдоімена використовуються для визначення певних полів таблиць, яким також присвоюються псевдоімена, що вказуються після ключового слова AS. Визначення умови в цій конструкції є обов'язковим параметром, оскільки в ній задаються зв'язки між таблицями за ключовими полями. Розглянемо приклад такого вибору.

Нехай існує таблиця користувачів users з полями id, name і таблиця замовлень orders з полями id, user_id, goods_id, order_date, а нам потрібно вибрати імена користувачів, які замовляли товари 4 жовтня 2016 року. Такий запит матиме вигляд:

```
SELECT T1.name AS uname
FROM users T1, orders T2
WHERE T1.id=T2.user_id AND T2.order_date='2016-10-04'
ORDER BY uname ASC
```

Цей запит вибере список імен користувачів за вказаною умовою як список псевдополя uname.

За допомогою такої конструкції можна здійснювати запит на вибірку відразу з двох і більше таблиць, що пов'язані між собою, на відміну від конструкції SELECT…JOIN, яка дає змогу здійснювати вибірку тільки з двох таблиць. Синтаксис конструкції SELECT…JOIN має такий вигляд:

```
SELECT T1.field1 AS fld1[, T1.field2 AS fld2, ...]
[T2.field1 AS fldA[, T2.field2 AS fldB,...]]
FROM table1 T1 JOIN table2 T2
ON condition
WHERE definition
```

У ній, на відміну від попередньої, зв'язок між таблицями вказується в області condition після ключового слова ON, а інша умова – після ключового слова WHERE.

Попередньо розглянутий приклад у конструкції SELECT…JOIN матиме вигляд:

```
SELECT T1.name AS uname
FROM users T1
JOIN orders T2
ON T1.id=T2.user_id
WHERE T2.order_date='2016-10-04'
ORDER BY uname ASC
```

Для видалення записів з таблиці використовується команда `DELETE`. Синтаксис цієї команди має вигляд:

```
DELETE FROM table_name [WHERE definition]
```

Ця команда видаляє всі записи з таблиці `table_name` за вказаною умовою.

Для конструювання умови можуть використовуватися ключові слова: `AND`, `BETWEEN`, `OR`. Вони визначають діапазони значень умови вибору. Повний список усіх команд, операторів, ключових слів, інструкцій, обчислюваних функцій мови *SQL*, а також приклади їх застосування наведені в довіднику щодо *SQL* [1].

Для побудови таблиць баз даних і роботи з ними невід'ємною складовою є визначення типів даних, що зберігаються в таблиці. У разі проєктуванні таблиць потрібно завчасно визначитися з розміром полів, форматом кодування символів, типом самої таблиці, оскільки все це напряму впливає на розмір бази даних і ефективність її використання.

Типи даних (полів) таблиць можуть бути числові, дати і часу, символьні або текстові, бінарні. У таблицях 5.2–5.4 наведено список типів даних, їх розмір, опис і діапазон.

Таблиця 5.2

Числові типи даних

Тип	Довжина поля	Розмір, байт	Опис
`tinyint(n)`	n = 1…3	1	Цілі числа від –128 до 127
`smallint(n)`	n = 1…5	2	Цілі числа від –32 768 до 32 767
`mediumint(n)`	n = 1…8	3	Цілі числа від –8 388 608 до 8 388 607

Таблиця 5.2 (закінчення)

`int(n)`	n = 1…10	4	Цілі числа від –2 147 483 648 до 2 147 483 647
`bigint(n)`	n = 1…20	8	Цілі числа від – 9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
`float(n,d)`	n – до крапки d – після крапки	4	Число з плаваючою крапкою від $\pm 3.4028 \times 10^{-38}$ до $\pm 3.4028 \times 10^{+38}$
`double(n,d)`	n – до крапки d – після крапки	8	Число подвійної точності з плаваючою крапкою від $\pm 1.7977 \times 10^{-308}$ до $\pm 1.7977 \times 10^{+308}$
`decimal(n,2)`	n – до крапки	5	Точне число з десятинною крапкою від $\pm 9.99 \times 10^{-6}$ до $\pm 9.99 \times 10^{+6}$

Таблиця 5.3

Типи даних дати та часу

Тип	Формат (довжина)	Розмір, байт	Опис
`datetime`	yyyy-mm-dd hh:mm:ss	8	Формат дати та часу від '1000-01-01 00:00:00' до '9999-12-31 23:59:59'
`date`	yyyy-mm-dd	3	Формат дати від '1000-01-01' до '9999-12-31'
`Time`	hh:mm:ss	3	Формат часу від '-838:59:59' до '838:59:59'
`timestamp(n)`	yyyymmddhhmmss (n = 1…14)	4	Формат дати та часу. Зберігаються як кількість секунд з початку епохи Unix ('1970-01-01 00:00:00' UTC).
`year(n)`	yyyy (n = 2, 4)	1	Формат року від 1 901 до 2 155

Таблиця 5.4

Текстові та бінарні типи даних

Тип	Довжина поля	Розмір, байт	Опис
`char(n)`	n = 1…255	n	Текстовий рядок фіксованої довжини. Фіксована довжина задається в дужках. Може зберігати до 255 символів
`varchar(n)`	n = 1…255	n + 1	Текстовий рядок змінної довжини. Найбільша довжина задається в дужках. Може зберігати до 255 символів
`tinytext`	1…255	1…255	Текстовий рядок довжиною до 255 символів
`text`	1…65 535	1…65 535	Текстовий рядок довжиною до 65 535 символів
`mediumtext`	1…16 777 215	1…16 777 215	Текстовий рядок довжиною до 16 777 215 символів
`longtext`	1…4 294 967 295	1…4 294 967 295	Текстовий рядок довжиною до 4 294 967 295 символів
`binary(n)`	n = 1…255	n	Бінарний об’єкт фіксованої довжини. Фіксована довжина задається в дужках. Може зберігати до 255 символів
`varbinary(n)`	n = 1…255	n + 1	Бінарний об’єкт змінної довжини. Найбільша довжина задається в дужках. Може зберігати до 255 символів
`tinyblob`	1…255	1…255	Бінарний об’єкт довжиною до 255 символів
`blob`	1…65 535	1…65 535	Бінарний об’єкт довжиною до 65 535 символів
`mediumblob`	1…16 777 215	1…16 777 215	Бінарний об’єкт довжиною до 16 777 215 символів
`longblob`	1…4 294 967 295	1…4 294 967 295	Бінарний об’єкт довжиною до 4 294 967 295 символів

5.3. Правила CSS-стилів

У сучасному *web*-просторі важливим і визначним аспектом є зовнішній вигляд і візуальне оформлення та стиль сторінок *web*-сайту. Від цього суттєво залежить, скільки часу відвідувач перебуватиме на сайті, зайде він ще раз або стане його постійним відвідувачем, чи зручно буде йому користуватися інструментами інтерфейсу і взагалі який візуальний вплив на психологічному рівні здійснюватиме його вигляд. Під час розроблення будь-якого сайту, особливо комерційного, створенням художнього образу займаються насамперед *web*-дизайнери. Але створені ними картинки, щоб стали живими сторінками, потрібно ожвавити. Здійснюється таке оживлення *html*-верстkою за безпосередньою участю *CSS*.

CSS (Cascading Style Sheets – каскадні таблиці стилів) – формальна мова опису зовнішнього вигляду *html*-документів. Основною метою розробки *CSS* було відокремлення опису логічної структури *web*-сторінки (яке здійснюється за допомогою *HTML*) від опису її зовнішнього вигляду (що здійснюється за допомогою *CSS*). Такий поділ надає гнучкості та можливості управління його вмістом, а також зменшує складність і повторюваність у структурному вмісті. Крім того, *CSS* дає змогу представляти той самий документ у різних стилях.

Правила *CSS* розташовуються у так званих таблицях стилів. Ці таблиці стилів, зі свого боку, можуть розташовуватися як у самому *html*-документі, зовнішній вигляд якого вони описують, так і в окремих файлах *css*-стилів, що мають розширення `*.css`. У файлі *css*-стилів не міститься нічого, крім переліку правил *CSS* і коментарів до них. Ці таблиці стилів можуть бути підключені до *html*-документа чотирма різними способами.

Перший спосіб – коли таблиця стилів розміщена в окремому файлі. Вона може бути підключена до *html*-документа за допомогою тегу `<LINK>`, розташованого в цьому документі в області заголовку `<HEAD>…</HEAD>`. За такого підключення шлях до зовнішнього файлу стилів вказується в атрибуті `href`, наприклад:

```
<LINK rel="stylesheet" type="text/css" href="style.css">
```

Другий спосіб – коли таблиця стилів розміщена в окремому файлі і може бути підключена за допомогою директиви `@import`, що розташовується в іншому файлі стилів, або в *html*-документі в межах пари тегів `<STYLE>…</STYLE>`, наприклад:

```
<STYLE media="all">@import url(style.css);</STYLE>
```

Третій спосіб – коли таблиця стилів описана в самому *html*-документі, вона може розташовуватися в межах пари тегів `<STYLE>…</STYLE>`, наприклад:

```
<STYLE>
  * {padding: 0; margin: 0;}
  html, body {width: 100%; height: 100%;}
  img, table {border: none;}
  table {border-collapse: collapse;}
</STYLE>
```

Четвертий спосіб підключення, коли таблиця стилів описана в самому *html*-документі, застосовується безпосередньо до певного елемента за допомогою атрибута `style`, наприклад:

```
<DIV style="position: relative;">...</DIV>
```

У перших трьох способах підключення таблиці *CSS* до документа кожне правило *CSS* із таблиці стилів має дві основні частини – *селектор* і *блок оголошень*. Селектор, розташований у лівій частині правила, визначає, на які частини документа поширюється правило. Блок оголошень розташовується в правій частині правила. Він поміщається у фігурні дужки «`{...}`» і складається з одного чи більш оголошень, розділених між собою символом «`;`». Кожне оголошення є поєднанням властивості *CSS* і значення, розділених символом «:». Селектори можуть групуватися в одному рядку через кому. У такому випадку властивість застосовується до кожного з них. Синтаксис визначення *css*-правила має вигляд:

```
селектор1[, селектор2, ...] {
  властивість1: значення1;
  властивість2: значення2;
…
```

```
    властивістьN: значенняN;
}
```

У четвертому способі перелік оголошень, розділених між собою символом крапка з комою, задається в атрибуті `style`.

Селектори поділяються на такі основні види, як універсальний селектор, селектори за тегом, селектори за ідентифікатором, селектори за класом, контекстні селектори.

Універсальний селектор позначається символом «*». У цьому випадку описані правила застосовуються для всіх елементів *web*-сторінки. Опис правила з універсальним селектором має вигляд:

```
* {padding: 0; margin: 0;}
```

Якщо селектором є ім'я тегу, правила застосовуються до всіх елементів *web*-сторінки з указаним ім'ям тегу. Опис правил із селектором за тегом має вигляд:

```
table {border: none; border-collapse: collapse;}
```

Правило буде застосовано для всіх таблиць `<TABLE>…</TABLE>`.

Якщо селектором є ідентифікатор елемента, правило застосовується до елемента з указаним ідентифікатором. Опис правила із селектором за ідентифікатором має вигляд:

```
#mainmenu {position: relative; height: 100px;}
```

Це правило застосовуватиметься до елемента з `id="mainmenu"`.

Якщо селектором є ім'я класу елемента, правило застосовується до елементів з указаним класом. Опис правила із селектором за класом має вигляд:

```
.prompt {text-align: justify; font-size: 11px;}
```

Правило застосовується до всіх елементів із класом *css*-стилю `prompt` (`class="prompt"`). Якщо клас вказати після імені тегу, правило застосовуватиметься для всіх елементів з вказаним іменем тегу та класу, наприклад:

```
div.prompt {font-size: 12px; color: green;}
```

Описане таким чином правило застосовуватиметься до всіх блочних елементів <DIV class="prompt">…</DIV>.

Контекстний селектор застосовується для підпорядкованих елементів. Він може складатися з комбінацій перелічених селекторів, але правила застосовуються до вказаних елементів, що належать батьківському. Опис правила таких селекторів має вигляд:

```
div.tile img {border: none; height: 100px;}
```

Таке правило застосовуватиметься до всіх елементів зображень <IMG>, які розташовані в межах елементів

<DIV class="tile">…</DIV>.

Також використовуються селектори за атрибутом, за дочірнім і сусіднім елементами, за псевдокласом і псевдоелементом, але вони підтримуються не в усіх браузерах. Опис правил із використання цих селекторів і приклади їх застосування наведені в довіднику [10].

Властивості *CSS* поділяються на групи залежно від призначення. Існує дев'ять основних груп: властивості тексту і шрифту, кольору і фону, блочної моделі, позиціонування, візуального відображення, смуг прокрутки, таблиць, списків, фільтрів.

До властивостей тексту і шрифту належать властивості, призначені для форматування блоків тексту і шрифту в ньому. Серед них – text-indent, text-align, text-decoration, text-transform, text-shadow, letter-spacing, word-spacing, white-space, line-height, vertical-align, font-family, font-style, font-weight, font-size, font. Вони дають змогу визначати довжину відступу в першому рядку текстового блока, вирівнювати текст в елементі, оформляти текст, змінювати регістр тексту, задавати ефект затінення, інтервал між символами та словами, забороняти перенесення тексту на наступний рядок, визначати відстань між базовими лініями сусідніх рядків, вирівнювати текст за вертикаллю в межах елемента, визначати сімейство, стиль, товщину, розмір шрифту, а також підключати шрифти з файлів шрифтів за допомогою властивості @font-face.

Приклад:

```
div {
  text-indent: 20px;
  text-align: justify;
  line-height: 22px;
  font-family: Arial, Tahoma, Geneva, sans-serif;
  font-size: 18px;
}
```

Візуальне оформлення, яке дає змогу зробити видовищну та привабливу *web*-сторінку, здійснюється за допомогою властивостей кольору і фону. Серед них – `color`, `background` (`background-color`, `background-image`, `background-repeat`, `background-attachment`, `background-position`). Вони дають змогу визначити: колір тексту, колір фону, фонове зображення елемента, напрямок заповнення, фіксацію, позицію зображення.

Приклад:

```
body {
  color: green;
  background:white url(bgimage.jpg) center no-repeat;
}
```

Кожен елемент розташовується в блоці певної ширини (`width`) та висоти (`height`). Цей блок має контейнер вмісту елемента, контур елемента (`outline`), рамку елемента (`border`), внутрішні поля елемента (`padding`), а також зовнішні поля елемента (`margin`). Схему такої блочної моделі наведено на рис. 5.2.

Для властивостей `margin` і `padding`, що встановлюють ширину зовнішніх і внутрішніх полів, параметр може мати від одного до чотирьох значень. Якщо вказано тільки одне значення, воно буде присвоєно відразу усім полям.

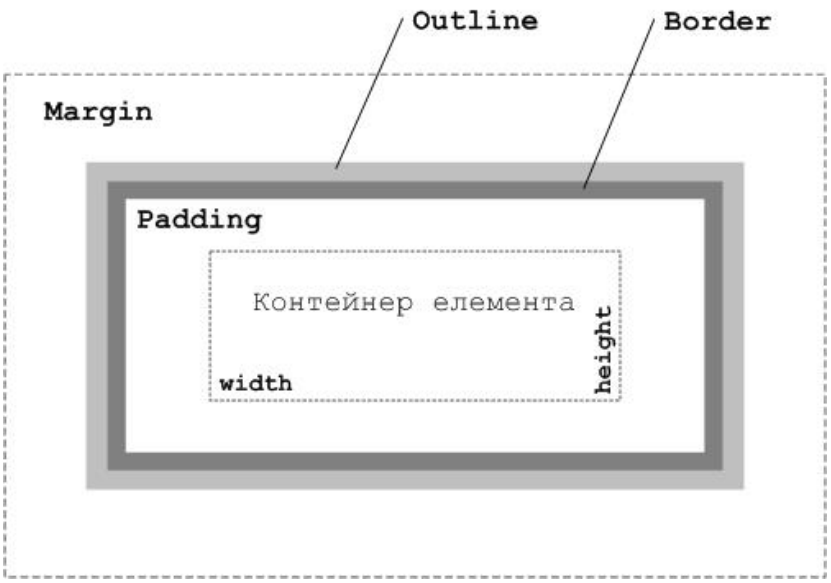


Рис. 5.2

Якщо – два значення, перше з них присвоюється верхньому і нижньому полям, а друге – правому та лівому. Якщо – три, перше значення присвоюється верхньому полю, друге – правому та лівому, третє – нижньому. Якщо – чотири, перше присвоюється верхньому полю, друге – правому, третє – нижньому, четверте – лівому, наприклад:

```
div {margin: 20px 30px 10px 30px; padding: 10px 5px;}
```

Ширину кожного поля також можна визначити окремо відповідними властивостями `margin-top`, `margin-right`, `margin-bottom`, `margin-left`, `padding-top`, `padding-right`, `padding-bottom`, `padding-left`.

Для властивостей `outline` і `border` параметри (товщина лінії, стиль, колір) вказуються в рядок через проміжок, наприклад:

```
div {outline: 2px solid red; border: 1px solid blue;}
```

Також їх можна визначити окремо через відповідні властивості: `outline-width`, `outline-style`, `outline-color`, `border-width`, `border-style`, `border-color`.

Ширина та висота елемента можуть визначатися в пікселях або відсотках, наприклад:

```
div {width: 200px; height: 100px;}
```

CSS дає змогу визначати положення *html*-елементів *web*-

сторінки у вікні браузера стосовно інших елементів. Здійснюється це за допомогою властивостей позиціонування. Серед них – position, top, right, bottom, left, які визначають метод позиціонування та положення елемента стосовно певної межі контейнера z-index, що визначає порядок розташування елементів за верстами, float і clear, які визначають положення елемента з обтіканням іншими елементами й заборону обтікання елемента з боків іншими елементами, відповідно.

Приклад:

```
div {
  position: absolute;
  top: 20px; left: 25px;
  z-index: 1;
}
div img {float: left;}
```

За допомогою властивостей *css*-стилів можна керувати візуальним відображенням елементів. Серед них – display, visibility, opacity, overflow, cursor. Вони дають змогу визначати відображення, видимість, прозорість елементів, відображення частини вмісту елемента у випадку, коли ця частина виходить за його рамки, та який вигляд має курсор миші в разі наведення на елемент.

Приклад:

```
div {
  display: block;
  visibility: visible;
  opacity: 0.8;
  cursor: pointer;
}
```

Для елементів *web*-сторінки, які використовують смуги прокрутки, їх стиль можна визначати за допомогою відповідних властивостей: scrollbar-3dlight-color, scrollbar-base-color, scrollbar-face-color, scrollbar-track-color, scrollbar-arrow-color, scrollbar-shadow-

color, scrollbar-darkshadow-color, scrollbar-highlight-color. За їх допомогою можна встановити колір смуги, повзунка, кнопок зі стрілками, стрілок і задати візуальні ефекти.

Окрему групу становлять властивості *css*-стилів для таблиць. До них належать: caption-side, table-layout, border-collapse, border-spacing, empty-cells, speak-header, column-span, row-span. Вони визначають зовнішній вигляд таблиці та її складових елементів. Окрему групу становлять властивості для визначення стилів списків. До них належать: list-style-type, list-style-image, list-style-position, list-style.

Для отримання різних візуальних ефектів для елементів *web*-сторінки застосовуються *css*-фільтри, які описуються властивостями: alpha, blur, chroma, dropshadow, flipH, flipV, glow, gradient, gray, invert, mask, shadow, wave, xray. Вони визначають прозорість, розмитість, затінення, світіння та інші ефекти.

Повний список усіх властивостей *css*-стилів, їх призначення та приклади застосування можна знайти в довіднику з *CSS* [10].

Контрольні запитання

1. Для чого використовується мова web-програмування JavaScript?
2. Як оголошуються глобальні та локальні змінні?
3. Які ви знаєте керуючі конструкції JavaScript?
4. Як у JavaScript створюється об’єкт?
5. Які базові об’єкти JavaScript ви знаєте?
6. Що таке подія? Як у JavaScript обробляється подія?
7. Для чого призначена мова SQL?
8. Як у php-скрипті можна виконати sql-запит?
9. Для чого використовується формальна мова CSS?
10. Якими способами можна підключати таблиці css-стилів до html-документів?

Лекція 6

СИСТЕМА УПРАВЛІННЯ КОНТЕНТОМ W#CMS

6.1. Про систему

Система управління контентом *W#CMS* є платформою для створення й адміністрування сайтів та *інтернет*-систем різної складності (*інтернет-магазинів, електронних каталогів, електронних бібліотек, систем обліку, систем управління* тощо).

Особливістю системи є те, що в ній можна реалізувати різні способи створення сайтів та *інтернет*-систем людьми, які володіють різним рівнем знань у цій сфері. Так, можна виділити три основні способи: *простий, середньої складності* та *спосіб із розширеними можливостями*.

Перший спосіб створення сайту (*простий*) не потребує спеціалізованих знань у сфері *web*-дизайну, верстки, *web*-програмування. Увесь сайт можна створити, сконструювавши його в панелі адміністратора системи, створивши при цьому структуру сайту, і для кожного його елемента в полі «*Текст сторінки*» задати текст для сторінки елемента, оформивши його зовнішній вигляд за допомогою вбудованого текстового *html*-редактора. Для цього потрібні мінімальні навички роботи з текстовим редактором, інтерфейс якого дуже схожий і близький за функціональністю з редактором *MS Word*. У підсумку можна отримати повноцінний *web*-сайт. Такий спосіб дає змогу створювати нескладні інформаційні сайти різної структури.

Другий спосіб створення сайту (*середньої складності*) потребує вміння роботи із *шаблонами* сторінок і знання функцій системи для *фрагментування* та *верстки шаблонів*, які, зі свого боку, підключаються до елементів структури сайту в панелі адміністратора системи й виводяться як вміст поточної сторінки елемента. Цей спосіб потребує мінімальних знань мови програмування *PHP* у роботі зі *змінними* та *функціями*, а також у разі потреби знань мови *JavaScript* та елементарних навичок роботи із *sql*-запитами й таблицями баз даних. Такий спосіб дає змогу

створювати різні сайти різної структури з динамічними сторінками: *корпоративні сайти*; нескладні *інтернет-магазини*; *електронні каталоги*; *електронні бібліотеки*; нескладні *сайти новин*.

Однак цей спосіб обмежується роботою з однією базою даних, з основною таблицею, де зберігається структура сайту з інформацією про її елементи, та допоміжними таблицями, якими можна управляти в панелі адміністратора.

Третій спосіб – це *спосіб із розширеними можливостями*. Система дає змогу створювати складні сайти з різними можливостями: підключення до зовнішніх серверів баз даних, з використанням усіх можливостей програмування й застосуванням мов *PHP*, *JavaScript* і *SQL*, підключенням зовнішніх фреймворків, а також створювати інші системи управління, використовуючи себе як платформу. Такий спосіб дає змогу створювати різні сайти різної структури з використанням багатьох *баз даних* і зберіганням інформації на різних *серверах*, використовуючи *базу даних* системи як основну. Панель адміністратора основної системи використовується для створення й управління структурою сайту або створюваної *інтернет*-системою, а в шаблонах сторінок і додаткових файлах скриптів виконується все інше конструювання сторінок із застосуванням різних інструментів та вживленням блоків з *php*- та *js*-скриптів. До таких сайтів і систем належать: ексклюзивні *інтернет-магазини* зі складною структурою і спеціалізованими можливостями; *інтернет-торговельні центри*; складні *електронні каталоги* й *бібліотеки*, що працюють з окремими незалежними таблицями; *системи обліку*, *портали новин* зі своєю, незалежною від основної, панеллю адміністратора; *соціальні мережі*; *електронні торговельні мережі* та інші системи управління.

Більшість відомих систем управління вмістом мають можливості для створення сайтів першим, а деякі – першим і другим способами з вищенаведених. Більшість із них спрямовані тільки на те, щоб сайт легко міг створювати користувач, який не володіє особливими знаннями та досвідом роботи з *html* і *web*-програмування. Це й робить систему *W#CMS* актуальною для

вирішення різного роду поставлених задач, пов’язаних зі створенням *web*-сайтів та інших *web*-систем різної складності.

У системі структурно сайт будується з елементів, які загалом утворюють структуру дерева з вузлами, розгалуженнями, кінцевими елементами і зв’язками. Кожному елементу присвоюється тип: *вузол*, *кінцева сторінка*, *зображення*, *файл*, *новина*, *акція*, *товар*. У разі потреби тип елемента можна змінити. Для кожного з елементів задається його належність до іншого елемента, або *кореневого вузла*.

Кожен елемент у системі має основні параметри: *унікальний id*, *назва елемента*, *адресна назва* (унікальна), у разі потреби – *примітка*, *опис*, *текст сторінки*. Також у разі потреби можна задати *картинку прев’ю* та *альтернативну картинку*, прикріпити *файл завантаження*.

Кожному елементу можна привласнити *html*-шаблон. *Шаблон* – це попередньо розмічений тегами і скриптами файл, вміст якого виводиться із зверненням до елемента. Якщо шаблон для елемента не встановлено і він відкривається як сторінка, то за стандартом виводитиметься текст, що міститься в полі «*Текст сторінки*».

Крім цього, можна вказати додаткові параметри елемента, які залежать від його типу: *дата створення*, *дата завантаження*, *дата початку*, *дата закінчення*, *величина*, *розмір*, *ціна*, *розмірність*, *одиниця виміру*, *валюта*, *унікальне значення*, *рядок*.

Параметр «*Дата створення*» застосовується для таких типів, як *вузол*, *кінцева сторінка*, *новина*, *товар*. Для типів *зображення* і *файл* він змінюється на «*Дата завантаження*», для типу *акція* – на «*Дата початку*». Параметр «*Дата закінчення*» застосовується для типів *новина*, *акція*, *товар*. Параметр «*Величина*» застосовується для типу *кінцева сторінка*, для типів *зображення* і *файл* він змінюється на «*Розмір*», для типів *акція* і *товар* – на «*Ціна*». Параметр «*Розмірність*» застосовується для типу *кінцева сторінка*, а для типів *зображення* і *файл* він змінюється на «*Одиниця виміру*», для типів *акція* і *товар* – на «*Валюта*». Параметри «*Унікальне значення*» та

«*Рядок*» застосовуються для всіх типів. Відповідності додаткових параметрів до типів представлені в таблиці 6.1.

Таблиця 6.1

Відповідності додаткових параметрів до типів

Параметр	**Тип**						
	Вузол	Кінцева сторінка	Зображення	Файл	Новина	Акція	Товар
	1	2	3	4	5	6	7
Дата створення	+	+	Дата завантаження	Дата завантаження	+	Дата початку	+
Дата закінчення	-	-	-	-	+	+	+
Величина	-	+	Розмір	Розмір	-	Ціна	Ціна
Розмірність	-	+	Одиниця виміру	Одиниця виміру	-	Валюта	Валюта
Унікальне значення	+	+	+	+	+	+	+
Рядок	+	+	+	+	+	+	+

У цій таблиці знак «–» означає, що для цього типу параметр не використовується, знак «+» означає, що параметр використовується під оригінальною назвою, зміна назви означає, що параметр використовується під назвою, яка зазначена в певній клітинці.

У загальному вигляді система складається з *процесора* (движка), *бази даних*, де зберігається інформація про кожен елемент і його зв’язки, та *панелі адміністратора*, у якій здійснюється наповнення сайту контентом (рис. 6.1).

У системі використовується технологія конструювання структури сторінок сайту (основного *макета* та *шаблонів*) *фрагментами*. Кожен *фрагмент* визначається відповідною

функцією фрагмента або *класом*, для нього стандартно визначений свій стиль, який, зі свого боку, можна налаштувати у файлі стилів.

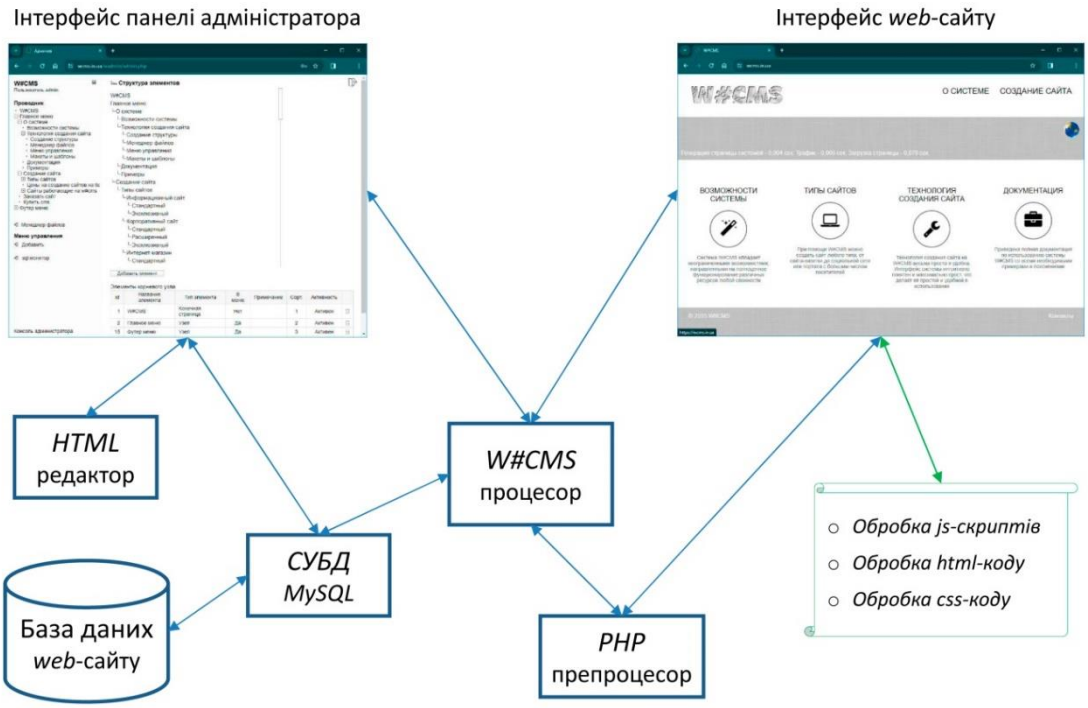


Рис. 6.1

Наприклад, для побудови головного меню використовується *фрагмент* під назвою `mainmenu`, що задається функцією `mainmenu()`, з відповідним стилем, який визначається ідентифікатором `mainmenu` (рис. 6.2).

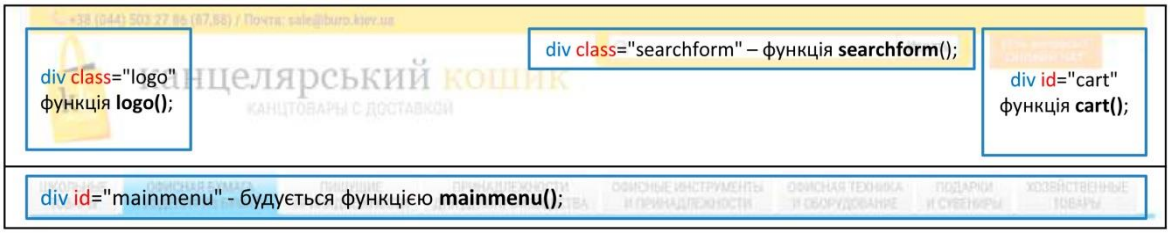


Рис. 6.2

Для побудови області виведення логотипу використовується *фрагмент* під назвою `logo`, що задається функцією `logo()`, з відповідним стилем, який визначається класом `logo`. Для побудови області виведення рядка пошуку використовується *фрагмент* під назвою `searchform`, задається функцією `searchform()`, з

відповідним стилем. Для кожного з *фрагментів* можна також задати свій власний стиль.

6.2. Можливості системи W#CMS

Можливості системи *W#CMS*, спрямовані на повноцінне функціонування різних *Internet*-ресурсів будь-якої складності, багатогранні. У ній застосований унікальний алгоритм роботи з базами даних, а також методи виведення інформації, що гарантують високу швидкість роботи кінцевого продукту. Движок системи має надійний «механізм» захисту. Швидкий і гнучкий він має можливість роботи за технологією *Ajax* як повністю, так і частково або комбіновано. У разі потреби його можна модифікувати під будь-які примхи.

Система дає змогу створювати й обслуговувати сайти зі структурою будь-якої складності:

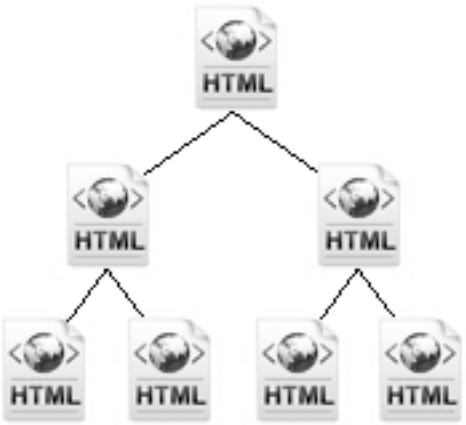
- лінійної, коли матеріал впорядковується в логічний ланцюжок (рис. 6.3, *а*);
- деревовидної, якщо кожен елемент належить іншому елементу або є батьківським до підлеглого йому елемента (рис. 6.3, *б*);
- гібридної, як сукупності деревовидної та послідовної структури в одному дереві (рис. 6.3, *в*);
- ґратчастої, коли між різними елементами (сторінками) можна встановити взаємний зв’язок і можливість швидкого переходу з однієї сторінки на іншу без потреби у відвідуванні проміжних сторінок (рис. 6.3, *г*).

Система також дає змогу створювати й обслуговувати сайти різних типів і призначень із використанням додаткових програмних модулів, підключених плагінів, власних скриптів, додаткових таблиць баз даних, декількох баз даних водночас із можливістю незалежного управління контентом сайту, який складається з декількох піддоменів і баз даних, що належать їм.

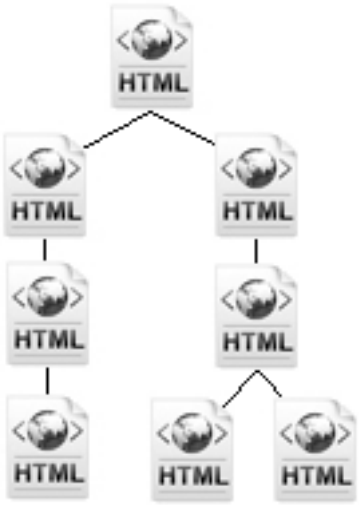
Система *W#CMS* надає можливість використовувати її для створення й обслуговування електронних мереж із незалежними піддоменами,

оскільки в ній реалізована можливість роботи з декількома незалежними базами даних. Кожний із піддоменів може обслуговуватися окремо, за допомогою панелі адміністратора *W#CMS*, встановленої на комп'ютері-сервері піддомену.

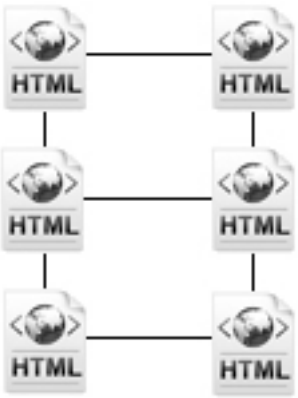
а – лінійна структура



б – деревоподібна структура



в – гібридна структура



г – ґратчаста структура

Рис. 6.3

6.3. Структура системи W#CMS

Під час створення сайту в системі *W#CMS* і наповненні його контентом інформація, що зазначається в панелі адміністратора, зберігається в *базі даних*. *Процесор* зчитує дані з *бази даних* і на основі *макета* (описаного в *файлі макета*) та попередньо підготовлених *шаблонів* (якщо використовуються) генерує вміст поточної сторінки (рис. 6.4).

У таблиці `ETREE` зберігається структура сайту і вся інформація про елементи, з яких він складається. У таблиці `ETYPE` зберігається список типів елементів, з яких може складатися сайт. У таблиці `ETREE_REL` зберігаються зв'язки елементів таблиці `ETREE`.

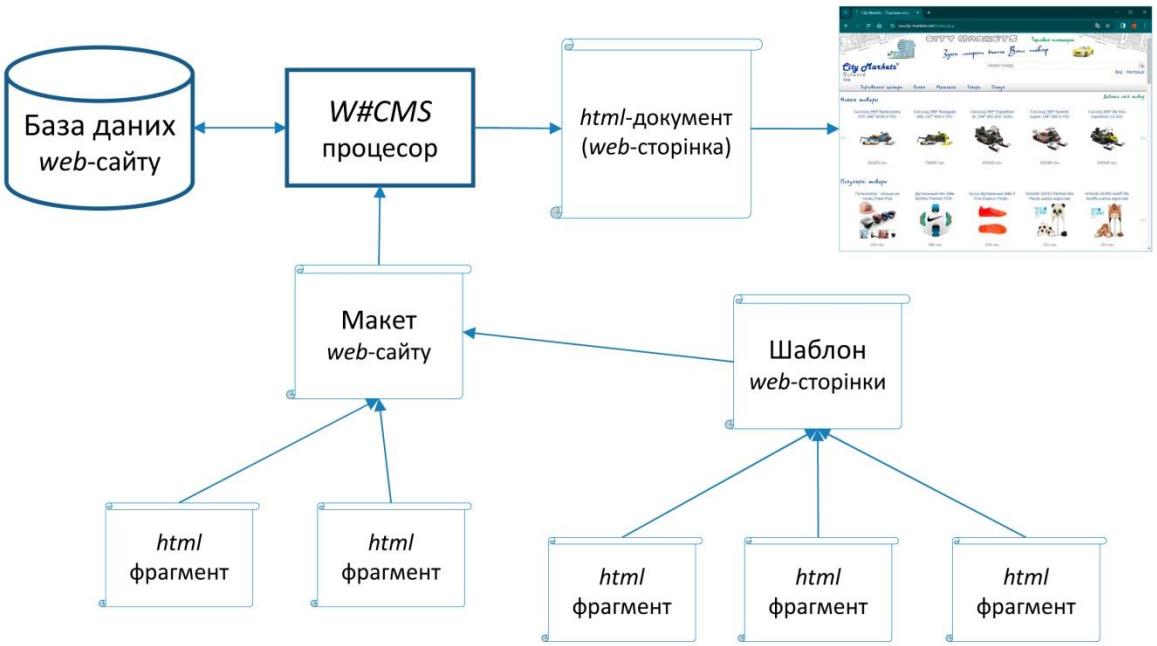


Рис. 6.4

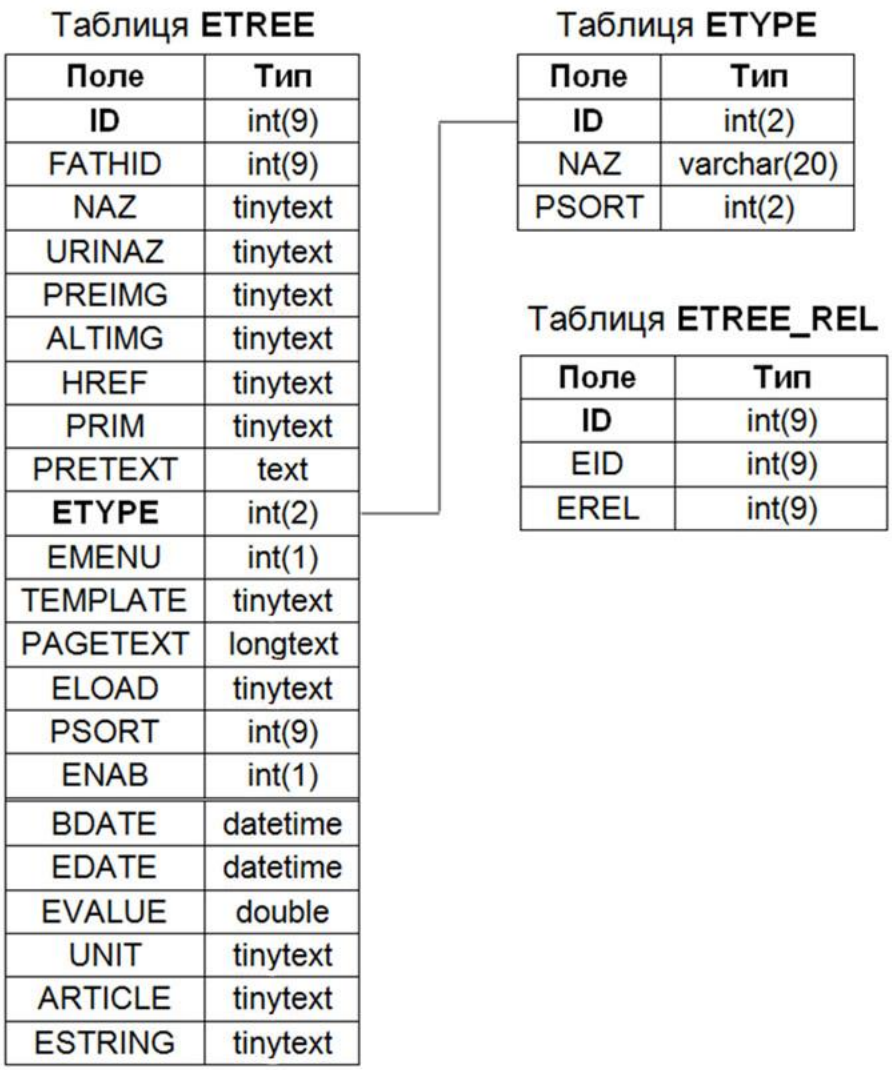


Рис. 6.5

Поля основної таблиці системи `ETREE` мають призначення, описані в таблиці 6.2.

Кожному полю таблиці `ETREE` відповідає зарезервована в системі змінна, що містить значення відповідного поля. Також для кожного елемента може бути створений об’єкт класу **`wcms`**, у якому зберігатиметься вся інформація про елемент, зчитана з таблиці. При цьому для поточної сторінки він створюється самостійно і зберігається в змінній `$wcms`.

Таблиця 6.2

Призначення полів таблиці ETREE

Поле	Назва
ID	id елемента
FATHID	Належність елемента до батьківського
NAZ	Назва елемента
URINAZ	Адресна назва елемента
PREIMG	Картинка прев'ю
ALTIMG	Альтернативна картинка
HREF	Гіперпосилання на іншу сторінку або зовнішній ресурс
PRIM	Примітка
PRETEXT	Опис
ETYPE	Тип елемента
EMENU	Відображення в меню
TEMPLATE	Шаблон елемента
PAGETEXT	Текст сторінки
ELOAD	Файл завантаження
PSORT	Порядок сортування
ENAB	Активність елемента
BDATE	Дата створення
EDATE	Дата закінчення
EVALUE	Величина
UNIT	Розмірність
ARTICLE	Унікальне значення
ESTRING	Рядок

Для кожного з елементів його тип зберігається в полі `ETYPE` значенням `id` таблиці `ETYPE`, у якій розміщено список типів елементів за id (у полі `ID`) та назвою (в полі `NAZ`). Типи елементів таблиці `ETYPE` та їх ідентифікатор наведені в таблиці 6.3.

У полях таблиці `ETREE_REL` зберігається інформація про зв'язок елемента, `id` якого записується в поле `EID`, з елементом, `id` якого записується у поле `EREL`, за створення зв'язку.

Таблиця 6.3

id	Тип
1	Вузол
2	Кінцева сторінка
3	Зображення
4	Файл
5	Новина
6	Акція
7	Товар

У разі потреби використання додаткових таблиць для сайту або системи, їх можна легко додати в панелі адміністратора системи. Інформація про додані додаткові таблиці зберігатиметься в системній таблиці `SYSMENU`. Таблиця `SYSMENU` містить список пунктів меню управління панелі адміністратора, а також імена таблиць, шаблонів і файлів стилю, що належать до певного пункту меню.

Системна таблиця `SYSFILE` містить список файлів, завантажених у менеджері файлів панелі адміністратора, а також деяких системних файлів і папок, які дозволяється переглядати, редагувати, перезавантажувати. Також ця таблиця містить інформацію про розташування цих об’єктів, їх опис, призначення або стан. У таблиці 6.4 наведено список призначень і станів об’єктів і їх ідентифікатори.

Таблиця 6.4

Призначення або стан об’єкта	id
Об’єкт закритий	0
Папка тільки для читання	1
Частково відкрита папка. Дозволено перегляд, завантаження, видалення тільки відкритих файлів	2
Відкрита папка з обмеженнями. Дозволено перегляд усього змісту, завантаження і видалення тільки відкритих файлів	3
Відкрита папка без обмежень. Дозволено перегляд усього змісту, завантаження і видалення всіх файлів	4
Відкритий файл тільки для читання	5
Відкритий файл. Дозволено читання, перезавантаження	6
Відкритий файл, завантажений адміністратором. Дозволено читання, перезавантаження, видалення	7

Інформація про адміністраторів поточного сайту та дані про них зберігаються в системній таблиці `SYSADM`. Оскільки керування структурою системних таблиць заборонено, їх структура не висвітлюється.

У системі W#CMS формування вмісту сторінки на основі інформації з бази даних, визначення змінних виведення, обробка полів злиття, виконання функцій і команд здійснюються процесором. У ньому за `$eid` поточної сторінки чи `$id` іншого елемента з бази даних зчитуються параметри та створюється об'єкт класу **`wcms`**, у якому зберігаються всі зчитані параметри елемента. Для поточної сторінки за `$eid` автоматично створюється об'єкт `$wcms` і на основі макета та шаблонів генерується її вміст. Для будь-яких інших елементів за `$id` елемента можна додатково створити власні об'єкти класу **`wcms`** та використовувати їх з різною метою.

Процесор викликається з файлу `index.php` або `router.php` (якщо сайт працює за технологією *Ajax*). Основний файл процесора – `processor.php`. Функції процесора описані в файлах: `getfuns.php` – функції визначення; `outfuns.php` – функції виведення. Команди процесора – у файлі `commands.php`. Зазвичай усі ці файли редагувати заборонено, але, за надзвичайної потреби їх можна модифікувати або доповнити. Процесором також створюється об'єкт `$wcms` класу **`wcms`**, у якому зберігаються всі параметри поточної сторінки.

Ключовим файлом системи є `index.php`. У ньому відбувається підключення файлу конфігурації, модуля управління sql-запитами, з'єднання із сервером і вибір основної бази даних, підключення процесора, підключення макета. Якщо сайт працює за технологією *Ajax*, використовується файл `router.php`, у якому відбуваються всі необхідні підключення, а саме: файлу конфігурації, процесора, підключення до основної бази даних.

Усі файли системи розкладені за своїми папками залежно від їх призначення.

Файли процесора розташовані в папці `wprocessor`. Вони є системними файлами, їх модифікація заборонена, оскільки некоректні зміни в цих файлах можуть привести до некоректної роботи системи. У менеджері файлів панелі адміністратора ці файли недоступні для перегляду й редагування.

У папці `wincludes` перебувають файли *php*-скриптів, функцій і класів, що підключаються. Основними з них є: `common.php` – файл, у якому містяться функції загального призначення; `current.php` – файл, у якому прописуються функції, що застосовуються тільки до поточного сайту. Також у разі потреби можна додати до цієї папки додаткові файли *php*-скриптів. У менеджері файлів панелі адміністратора файл `common.php` є доступним тільки для перегляду, а файл `current.php` – для перегляду, редагування й перезапису.

У папці `wjs` розміщені файли *js*-скриптів процесора, функцій *Ajax*, *js*-функцій загального призначення і функцій, що застосовуються тільки до поточного сайту. Файли з *js*-функціями, що забезпечують роботу системи, є системними, їх модифікація заборонена. Інші файли є відкритими. До них належать: `common.js` – файл з *js*-функціями загального призначення; `current.js` – файл, у якому прописуються *js*-функції, що застосовуються тільки для поточного сайту. Також у разі потреби можна додати до цієї папки додаткові файли *js*-скриптів. У менеджері файлів панелі адміністратора файл `common.js` є доступним тільки для перегляду, а файл `current.js` – для перегляду, редагування й перезапису.

У папці `wstyles` розташовані файли *css*-стилів. Основними з них є: `general.css` – основний файл стилів елементів, що генеруються процесором; `styles.css` – файл стилів елементів поточного сайту. Ці файли доступні як для перегляду, так і для перезапису. Також у разі потреби можна додати додаткові файли *css*-стилів.

Папка `wimages` призначена для файлів зображень, що є незмінною складовою дизайну поточного сайту. У менеджері файлів

панелі адміністратора цими файлами можна управляти як завгодно, а саме переглядати, перезаписувати, видаляти.

Для збереження завантажених шаблонів сторінок призначена папка `wpages`. Основним файлом у цій папці є файл макета `layout.htm`. Усі інші шаблони завантажуються в панелі адміністратора системи зі сторінки редагування елемента. У менеджері файлів файл макета `layout.htm` доступний для перегляду і перезапису. Файли шаблонів, що завантажуються зі сторінки редагування елемента, у менеджері файлів доступні тільки для перегляду, однак для перезапису й видалення вони доступні на сторінках перегляду і редагування елемента. Це пов'язано з тим, що шаблони, що зберігаються в цій папці, прив'язані до елементів, які їх використовують.

Для збереження завантажених з панелі адміністратора файлів різних типів, відповідних тому чи іншому елементу, призначена папка `wloads`. Ця папка в панелі адміністратора відкрита тільки для перегляду. Управління файлами, що входять до неї, здійснюється зі сторінок перегляду й редагування елемента.

Усі необхідні налаштування системи здійснюються у файлі `configs.php` (рис. 6.6). У ньому задаються дані доступу до сервера: хост сервера баз даних, ім'я користувача, пароль, ім'я основної бази даних, кодування виведення.

У ньому також задаються: ім'я хоста сайту, ім'я основного файлу (`$indexfile`), ім'я файлу макета (`$layout`), налаштування виду посилань (`$linktype`). Налаштування роботи з http-сервером у разі потреби задаються в інших файлах конфігурації. Наприклад, під час роботи із сервером Apache вони задаються у файлі `.htaccess`.

configs.php

```php
<?
//Дані для роботи з сервером
$module='mysql';
$dbhost="site.mysql.in.ua"; //Хост
$dbuser="root"; //Им'я користувача
$dbpassword="1234"; //Пароль
$dbname="dbwcms"; //Им'я бази даних
$charset="utf8"; //Кодування виводу (utf8/cp1251).

//Налаштування шляхів
$hostname="site.in.ua";
$indexfile="index.php";
$layout="wpages/layout.htm";

//Налаштування виду посилань
$linktype=0; // 0 - href з $indexfile та змінних запиту;
             // 1 - подія onclick;
             // 2 - href з назв сторінок
?>
```

Рис. 6.6

Система W#CMS працює у двох основних загальноприйнятих форматах кодування: UTF-8 (utf8) і Windows-1251 (cp1251). Змінною `$charset` файлу конфігурації задається формат кодування виведення, який використовується як для роботи з базою даних, так і для кодування виведення *html*-сторінок у браузерах.

У системі також використовуються три різні види гіперпосилань, які задаються змінною `$linktype`. Перший із них (задається значенням 0) – це гіперпосилання на сторінки, що складається з основного файлу й переліку змінних, перша з яких містить id сторінку, на яку створюється посилання. Наприклад:

```
<a href="index.php?eid=3">
```

При цьому повна адреса такої сторінки матиме вигляд:

```
http://site.in.ua/index.php?eid=3
```

Другий вид (задається значенням 1) – це посилання з подією onclick, тобто за натискання на гіперпосилання завантаження певної сторінки відбудеться без перезавантаження всього сайту. Це, зі свого боку, заощаджує інтернет-трафік і час на завантаження, що суттєво підвищує швидкість роботи сайту. Наприклад:

```
<a href="#" onclick="doLoadPage('3');">
```

Оскільки в такому випадку система працює за технологією *Ajax,* повна адреса сторінки матиме вигляд як у першому виді.

Третій вид (задається значенням 2) – це гіперпосилання, що складається з адресної назви сторінки, на яку утворюється посилання. Наприклад:

```
<a href="Pro-sustemy">
```

При цьому повна адреса такої сторінки буде мати вигляд:

```
http://site.in.ua/Pro-sustemy
```

6.4. Панель адміністратора системи W#CMS

Створення структури сайту, наповнення його контентом і управління ним здійснюється в панелі адміністратора. Вхід до панелі адміністратора й робота з нею здійснюються через будь-який інтернет-браузер, у якому інформація виводиться у вигляді вікна з робочими сторінками панелі адміністратора (рис. 6.7). Вікно панелі адміністратора системи W#CMS розділяється на декілька частин.

У лівій частині вікна розташовується панель управління. У ній розташовані провідник, менеджер файлів, меню управління. Провідник використовується для навігації за елементами структури сайту. Менеджер файлів використовується для перегляду й управління файлами, що входять до складу сайту. Меню управління використовується для додавання й управління додатковими таблицями бази даних, а також для підключення сторінки інтерфейсу управління додатковими таблицями.

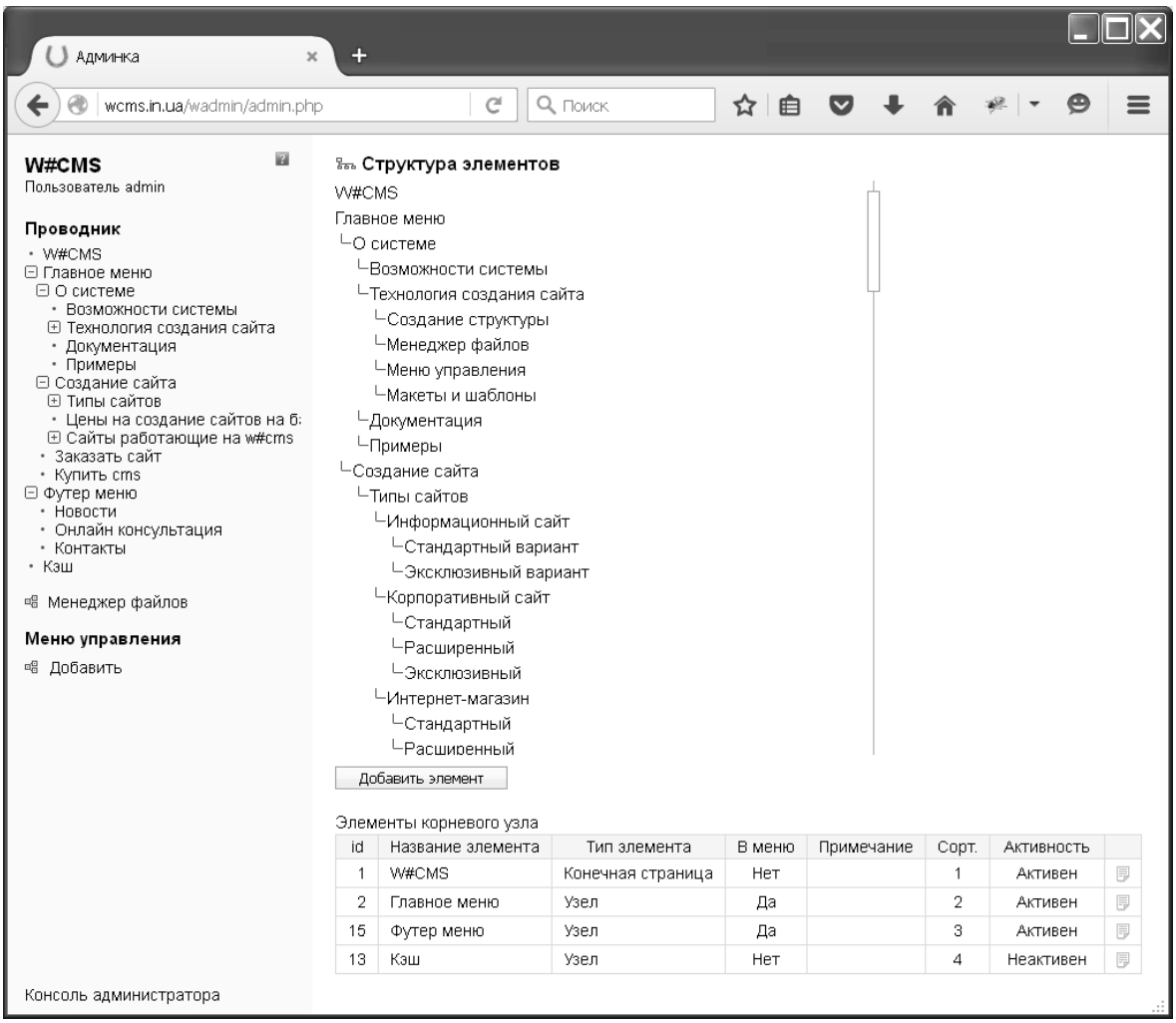


Рис. 6.7

У центральній частині вікна розміщено робочу область, у якій виводяться сторінки для роботи з елементами й об’єктами. Знизу центральної частини розташована таблиця елементів поточного вузла з основною інформацією про підлеглі елементи.

На головній сторінці панелі адміністратора виводиться структура елементів сайту у вигляді дерева. За його допомогою можна перейти до будь-якого елемента, переглянути його властивості та параметри, здійснити редагування. Перехід до будь-якого іншого елемента здійснюється за допомогою провідника. Якщо елемент є вузлом, ліворуч від нього є позначення вузла (іконка зі знаком «+» чи «–»), за допомогою якого можна згорнути або розгорнути вузол.

Додавання (створення) нових елементів ініціюється кнопкою «Добавить элемент», яка розташована під деревом елементів на

головній сторінці чи під таблицею властивостей на сторінці властивостей елемента (рис. 6.8).

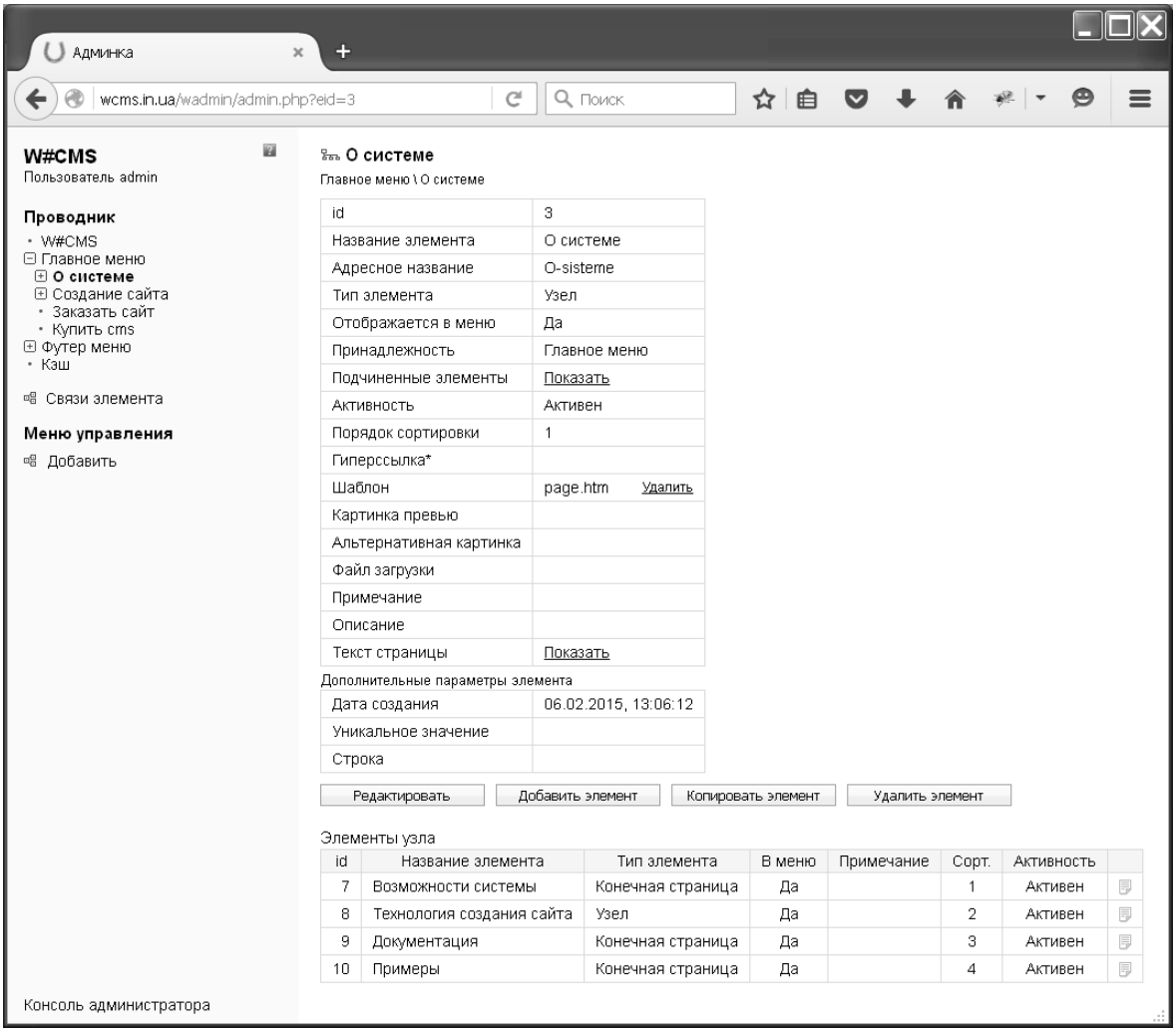


Рис. 6.8

Редагування, копіювання, видалення елементів ініціюються кнопками: «Редактировать», «Копировать элемент», «Удалить элемент» відповідно. Ці кнопки розташовані на сторінці властивостей елемента. На ній у вигляді таблиці виводиться список основних і додаткових параметрів елемента.

Якщо елемент є вузлом і має підлеглі елементи, їх ієрархію можна вивести в окремому віконці, натиснувши на кнопку «Показать» у рядку «*Подчиненные элементы*» таблиці властивостей (рис. 6.9). За допомогою цього списку елементів можна швидко перейти до того чи іншого елемента для перегляду його властивостей на відповідній сторінці.

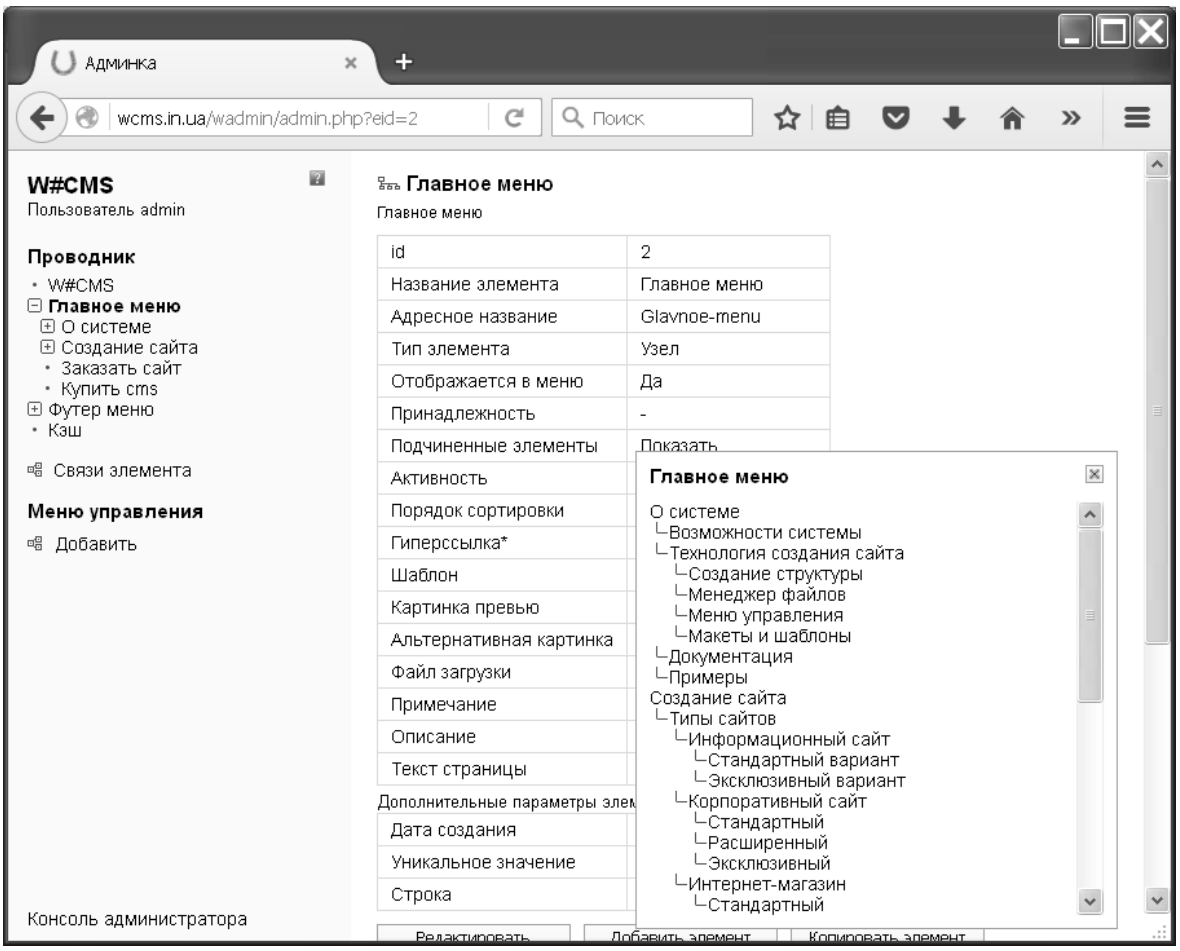


Рис. 6.9

Якщо для елемента встановлено *html*-шаблон, його вміст можна переглянути у вікні перегляду вмісту файлу шаблону, клацнувши по імені файлу шаблону в рядку «*Шаблон*» таблиці властивостей (рис. 6.10). Вміст шаблону для зручності перегляду виводиться у вигляді попередньо відформатованого тексту, розміченого різним кольором, шрифтом, накресленням. Цим вікном можна керувати за всіма правилами роботи з вікнами у Windows, але редагування тексту в ньому є неможливим. Якщо до елемента підключені зображення, що призначені для графічного відображення елемента на сторінках сайту, а також прикріплений будь-який файл зображення, вони виводяться у відповідних рядках таблиці властивостей: «*Картинка превью*», «*Альтернативная картинка*», «*Файл загрузки*» у вигляді зменшених зображень, які можна переглянути в оригінальному розмірі в окремому віконці (рис. 6.11), клацнувши мишею по відповідному зображенню.

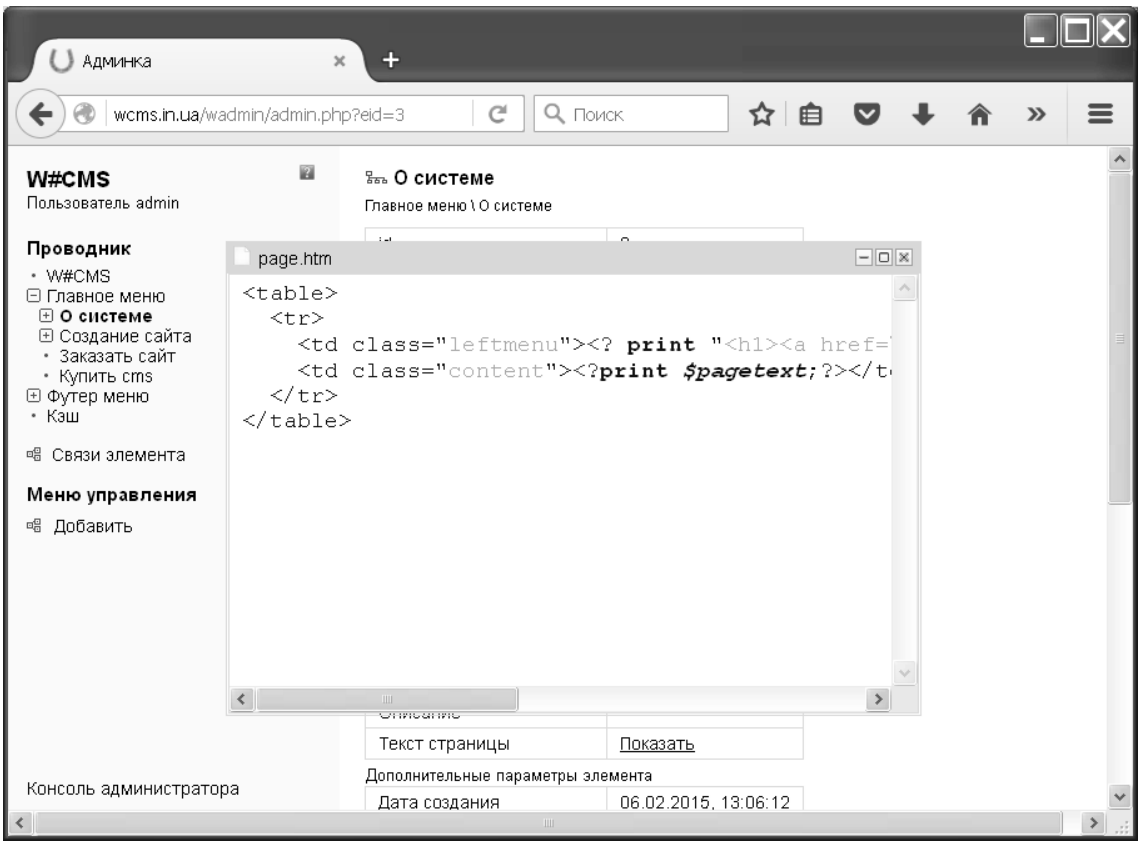


Рис. 6.10

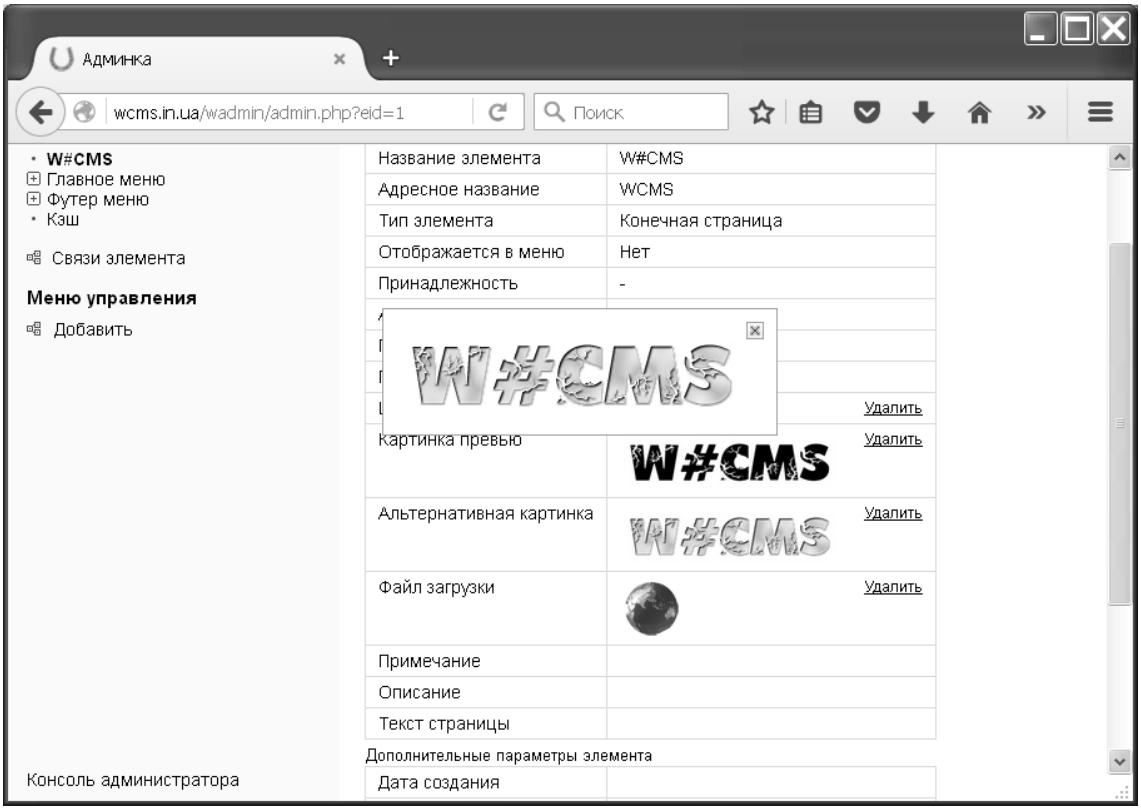


Рис. 6.11

Для кожного з елементів використовуються текстові параметри: «*Примечание*», «*Описание*», «*Текст страницы*». Зміст параметрів «*Примечание*» і «*Описание*» виводиться у відповідних рядках таблиці властивостей. Зміст параметра «*Текст страницы*», якщо потрібно переглянути, виводиться в окремому віконці (рис. 6.12), що відкривається за натискання на кнопку «Показать» у відповідному рядку таблиці властивостей.

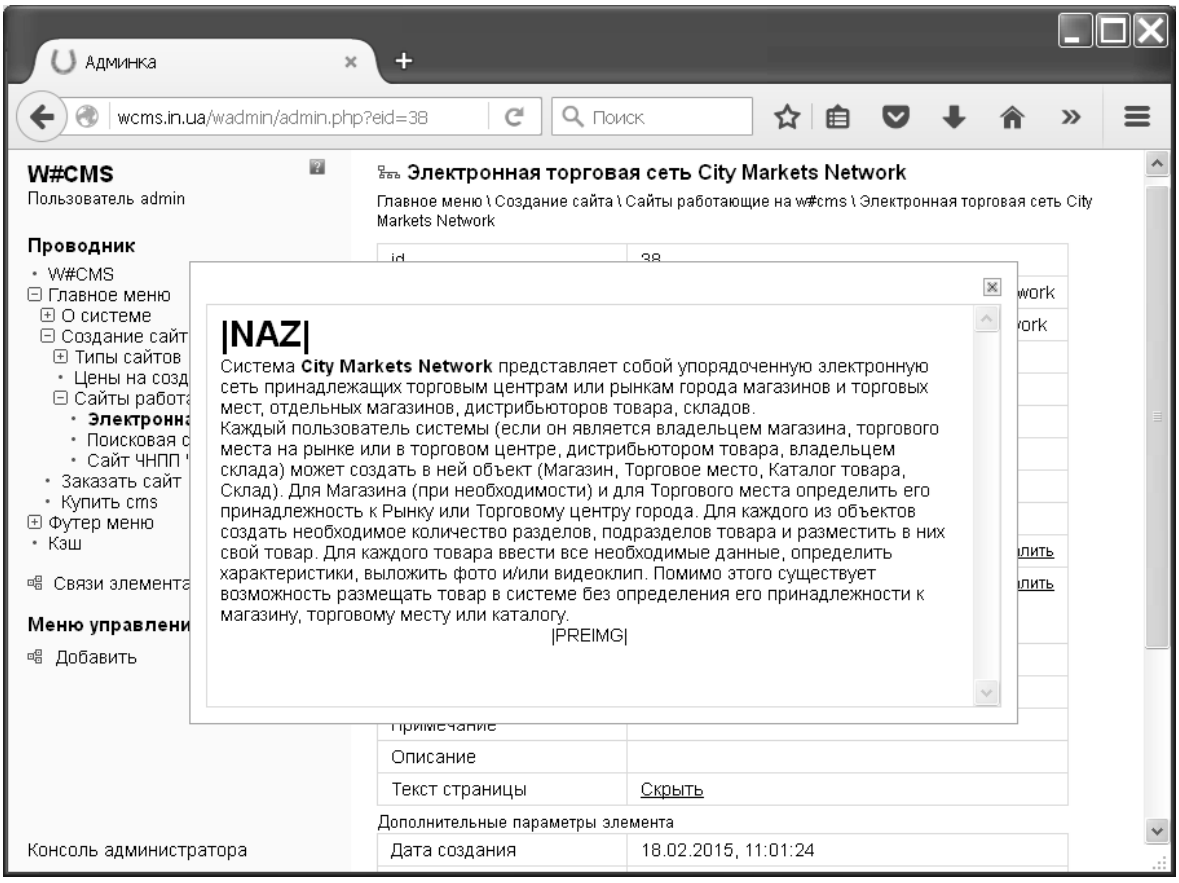


Рис. 6.12

Кожний елемент структури сайту можна представити окремою *html*-сторінкою, вміст якої задається параметром «*Текст страницы*», що може мати повний вміст готової *html*-сторінки сайту. При цьому за допомогою полів злиття в межах *html*-фрагменту «*Текст страницы*» можна виводити вміст інших параметрів поточного елемента. Також вміст параметра «*Текст страницы*» можна виводити як *html*-фрагмент у межах будь-якої іншої *html*-сторінки.

Якщо елемент є вузлом, для нього на сторінці властивостей у розділі «Элементы узла» виводиться таблиця підлеглих елементів

(рис. 6.9). У цій таблиці можна змінювати відображення елемента в меню (тобто показувати його чи ні), порядок сортування, змінювати стан активності. Також можна здійснювати перехід на сторінку властивостей підлеглого елемента або відкривати його в режимі редагування.

Інтерфейс панелі адміністратора системи максимально простий, що робить її простою та зручною у використанні, тим самим виносячи всі свої складності за дужки й позбавляючи від них користувача, залишаючи йому простір для творчої роботи. Так, для редагування або додавання нового елемента потрібно лише натиснути відповідну кнопку, а далі на відповідній сторінці ввести всі необхідні дані. Редагування або додавання нового елемента здійснюється на сторінці редагування елемента (рис. 6.13). На ній задаються назва елемента й адресна назва, тобто назва, за допомогою якої можна перейти до сторінки, якщо елемент є сторінкою, з рядка адреси браузера. Для кожного елемента обов'язково задається його тип, за яким система визначає, як поводитись із цим елементом. Якщо потрібно, щоб посилання на елемент відображувалося в списку меню, можна вказати на це за допомогою параметра «*Отображается в меню*».

У системі *W#CMS* легко змінювати належність елемента до іншого або до кореневого вузла. Для цього потрібно натиснути на кнопку «Изменить» у відповідному рядку, а далі у віконці, що з'явилося, вибрати необхідний батьківський елемент (рис. 6.14).

Існують випадки необхідності використання гіперпосилання на зовнішній ресурс чи на іншу сторінку поточного сайту. Якщо в полі «*Гиперссылка*» вказати таке посилання, то зі звертанням до цього елемента здійснюватиметься переадресація на ресурс, указаний у цьому полі.

Для приєднання шаблону до елемента існує два способи. Перший з них – це вибір шаблону зі списку вже завантажених (рис. 6.15), що дає можливість використовувати один шаблон для декількох однотипних елементів. Другий – завантаження шаблону, який ще не використовується іншими елементами, в полі «*Загрузить*

другой шаблон». Вибір файлу шаблону здійснюється за допомогою діалогового вікна відкриття файлу браузера.

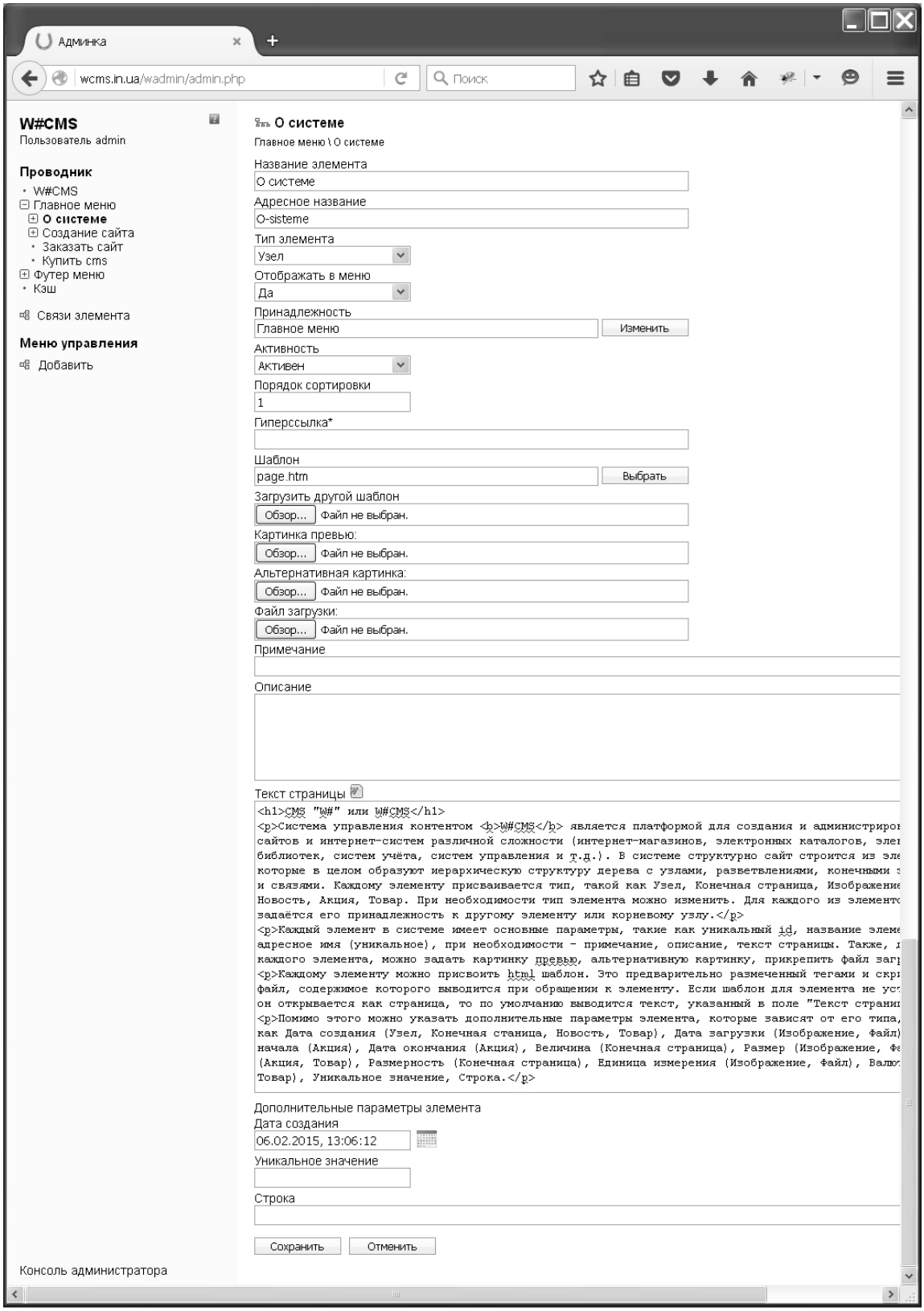


Рис. 6.13

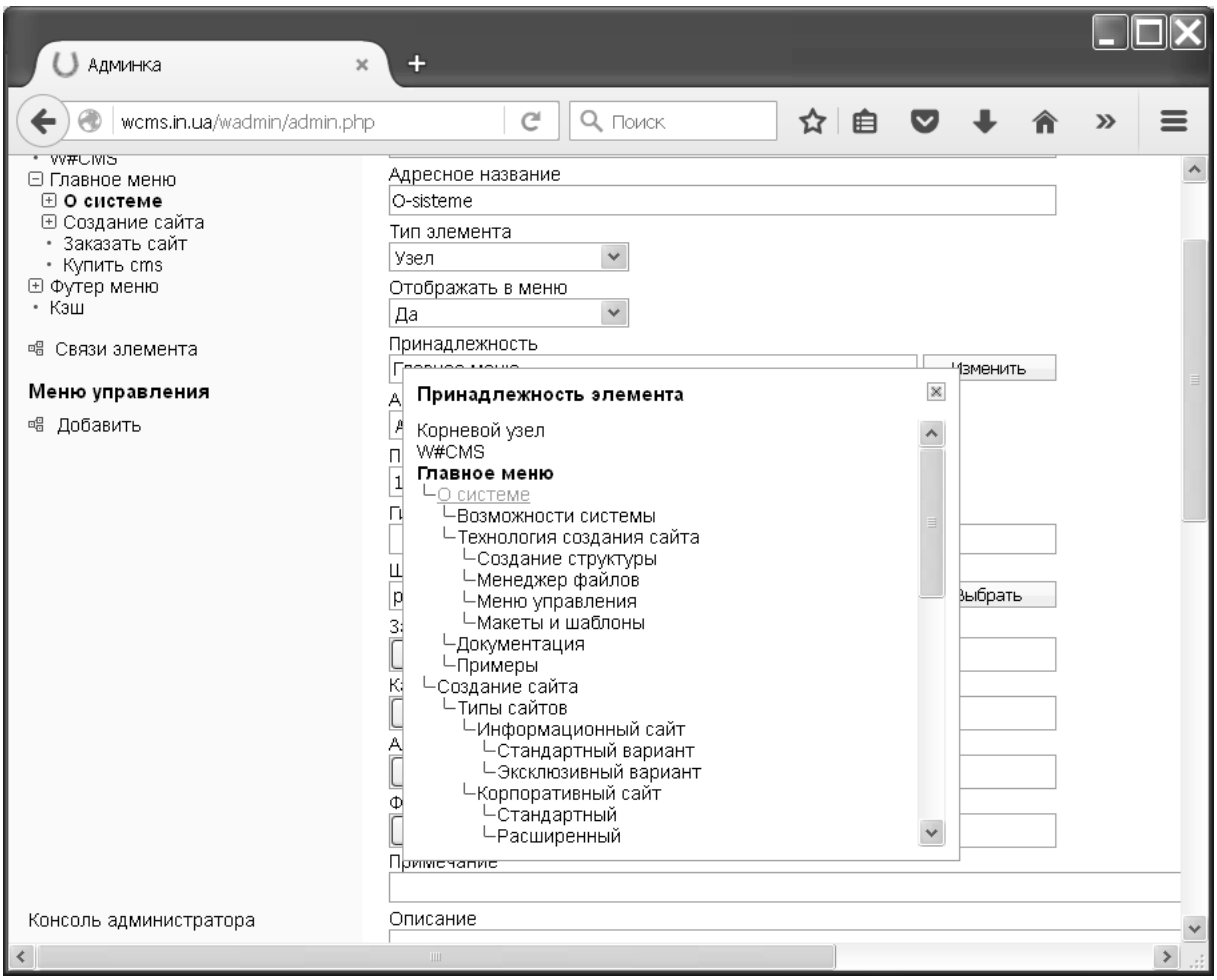


Рис. 6.14

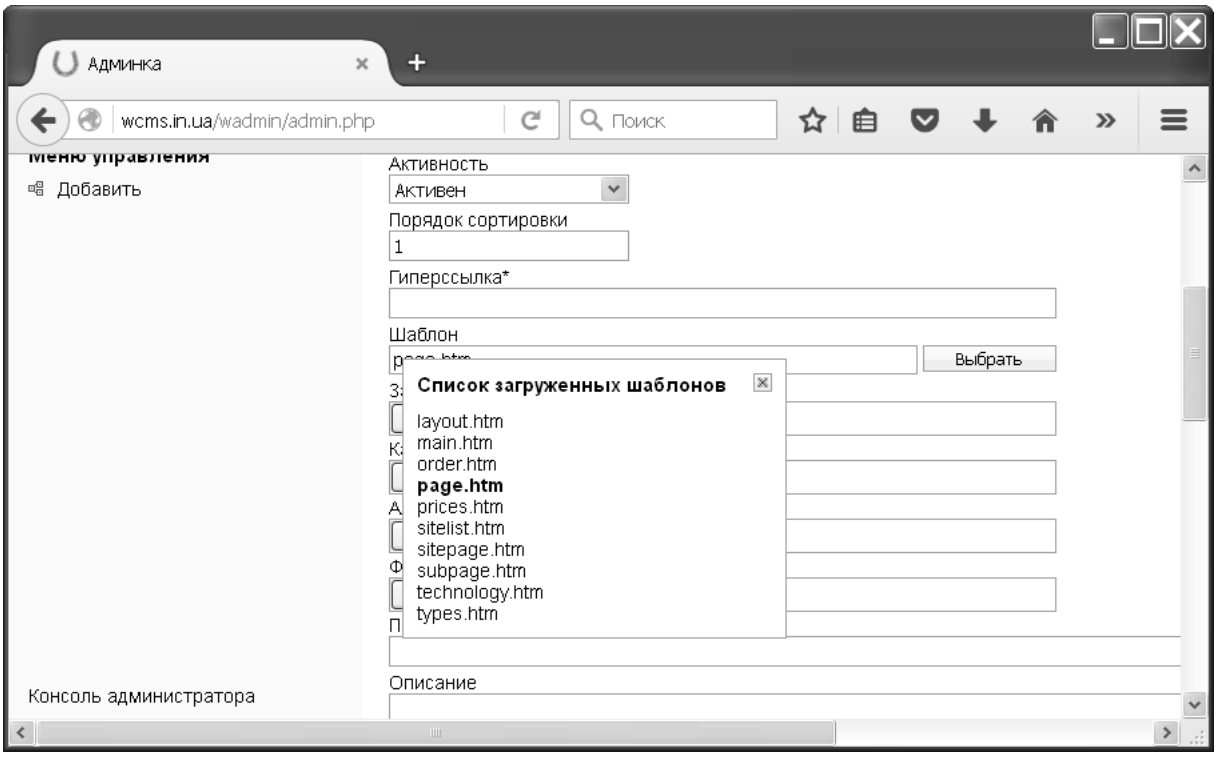


Рис. 6.15

Для зручності роботи з *html*-текстом сторінки, поле для введення тексту сторінки забезпечено текстовим редактором FCKeditor, за допомогою якого можна самостійно скласти повноцінну *web*-сторінку, використовуючи надані інструменти на панелі інструментів. Для завантаження редактора й початку роботи з ним потрібно клацнути лівою клавішею миші по іконці , яка розташована праворуч від назви поля «*Текст страницы*». Редактор завантажується замість відповідного текстового поля «*Текст страницы*» на сторінці редагування і в ньому відображається весь його вміст (рис. 6.16). У разі потреби редактор можна перевести у повноекранний режим відповідною кнопкою на панелі інструментів.

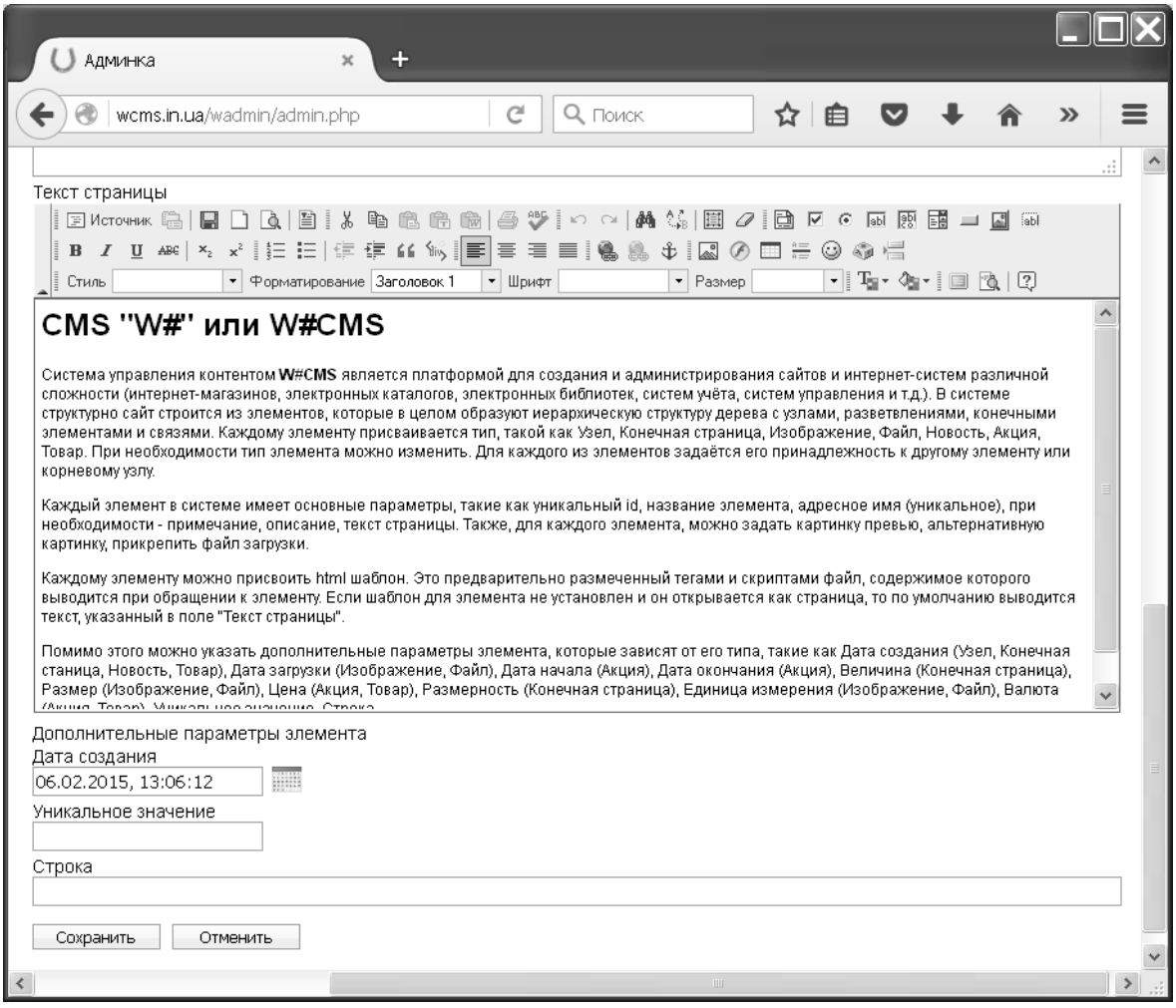


Рис. 6.16

У редакторі використовуються два режими відображення тексту – режим верстки (основний) і режим джерела (*html*-код). У режимі верстки текст відображається таким, яким його побачать

відвідувачі сайту на поточній сторінці. Форматування тексту, редагування, вставка фрагментів і елементів *html*-тексту, об’єктів *html*-сторінки та інші дії здійснюються за допомогою кнопок, представлених на панелі інструментів. У режимі джерела текст відображається в розмітці *html*-тегами. У цьому режимі редагування тексту здійснюється з урахуванням правил роботи з тегами html.

Для кожного з елементів структури сайту на сторінці редагування елемента також задаються додаткові параметри, зміст яких залежить від типу елемента. Так, для елемента з типом *вузол* можна використовувати додаткові параметри, що відображені на рис. 6.13. Для елемента з типом *кінцева сторінка*, *зображення* або *файл*, *новина*, *акція* – додаткові параметри, що наведені на рисунках 6.17, 6.18, 6.19, 6.20 відповідно.

Для елемента з типом *товар* можна встановити додаткові параметри, що наведені на рис. 6.21. Залежності додаткових параметрів від типів елементів наведено в таблиці 6.1.

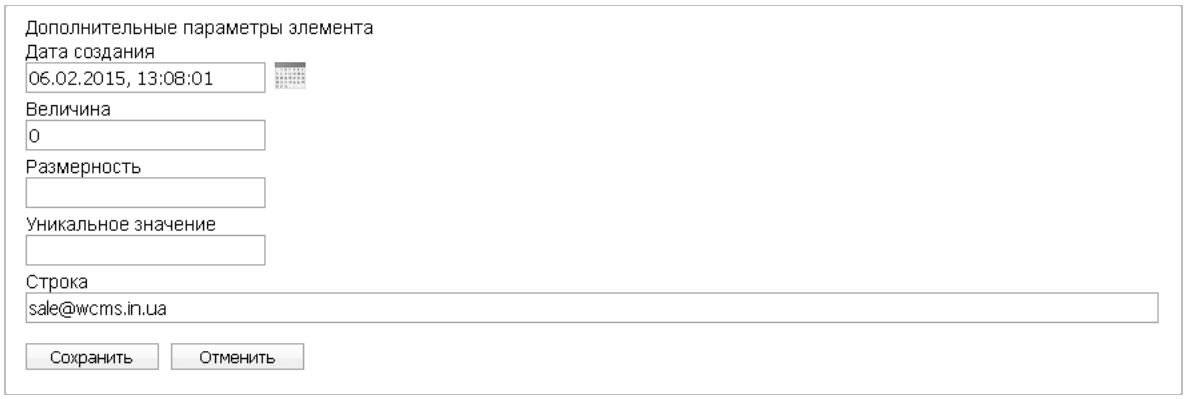


Рис. 6.17

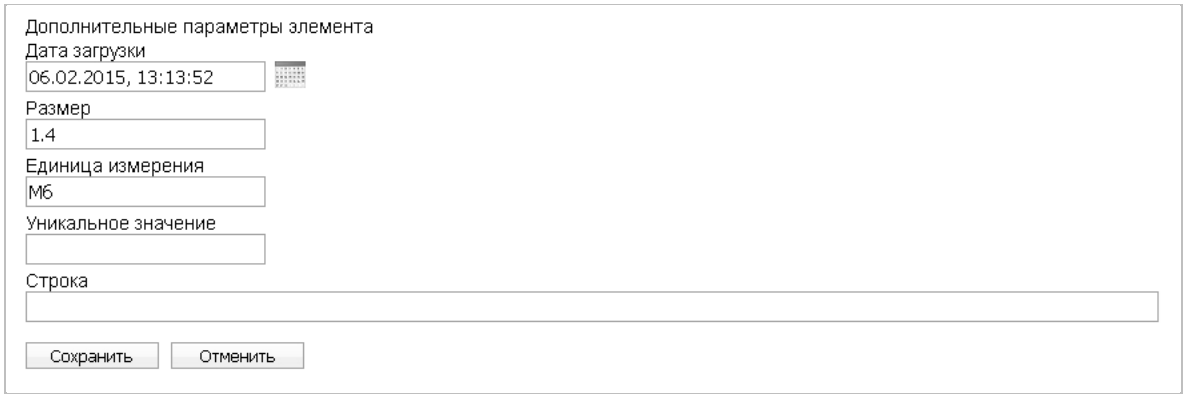


Рис. 6.18

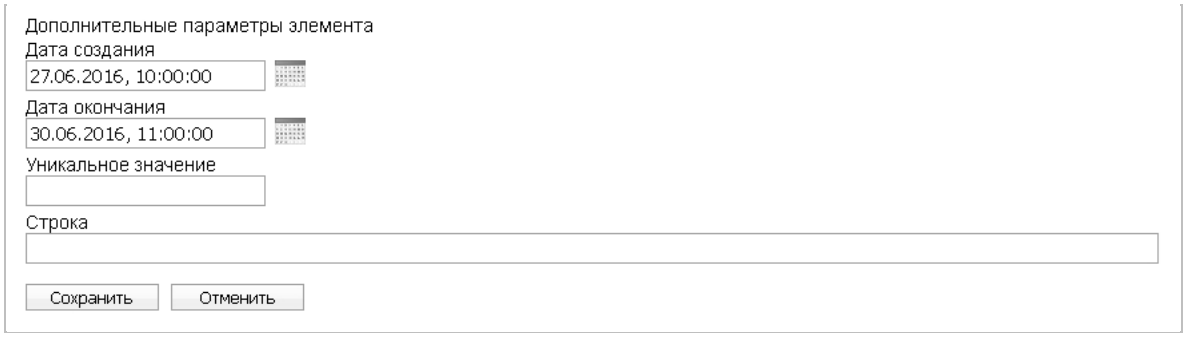


Рис. 6.19

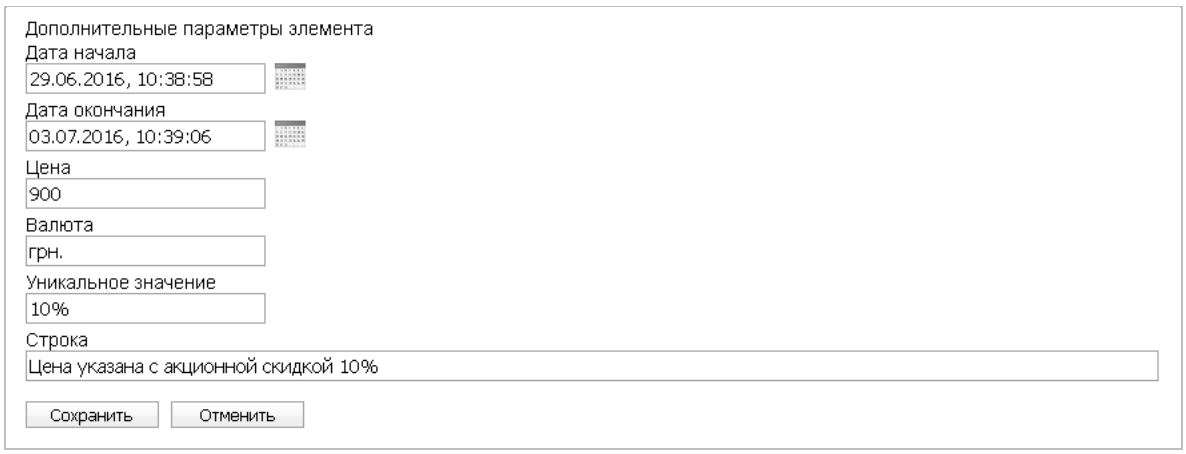


Рис. 6.20

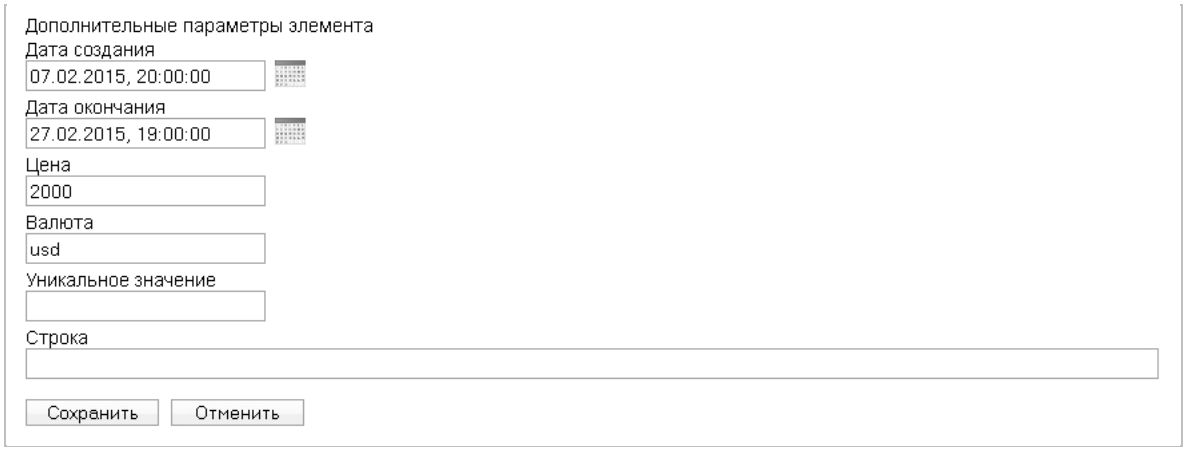


Рис. 6.21

Параметри «*Уникальное значение*» та «*Строка*» є універсальними. Їх можна використовувати в різних призначеннях

залежно від мети застосування поточного елемента. Параметр «*Уникальное значение*» може бути або числом, або текстом. Параметр «*Строка*» може містити будь-яке рядкове значення в рамках розмірності поля, довжина якого становить 255 символів.

Дати у відповідних полях вводяться в форматах `dd.mm.yyyy` та `hh:mm:ss` або вибираються з календаря, який відображається після натискання на відповідну кнопку праворуч від поля введення дати.

Контрольні запитання

1. Для чого призначена система управління контентом W#CMS?
2. Які особливості системи W#CMS відрізняють її від інших подібних систем?
3. Яка технологія конструювання структури сторінок сайту використовується в системі W#CMS?
4. Якими можливості є у системи W#CMS? Що можна створювати на основі цієї системи?
5. Яку структуру має система W#CMS?
6. Які типи елементів використовуються в системі?
7. Для чого призначена панель адміністратора системи та як її завантажити?
8. Як здійснюється додавання нових елементів сайту?
9. Як здійснюється редагування елементів сайту?
10. Які параметри встановлюються для елемента під час його створення та редагування?
11. Як переглянути дерево підлеглих елементів?
12. Як підключити html-шаблон?
13. Як переглянути підключений html-шаблон?
14. Які додаткові параметри елемента використовуються залежно від його типу?

Лекція 7

ТЕХНОЛОГІЯ СТВОРЕННЯ САЙТІВ НА БАЗІ W#CMS

Технологія створення сайту з використанням *W#CMS* досить проста і зручна. Вона поділяється на дві основні частини: *створення структури сайту* з наповненням його контенту в панелі адміністратора і *фрагментування* макета та шаблонів. Перш ніж приступити до роботи з *CMS,* безпосереднього створення структури сайту й наповнення його контентом, потрібно визначитися із самою структурою нового сайту та розробити дизайн його сторінок. Після цього визначити *макет* сайту, *шаблони* сторінок (у разі потреби), здійснити верстку. Зверстані *шаблони* підключаються до відповідних елементів дерева сайту, тим самим утворюючи його робочу структуру. Якщо до елемента не підключений шаблон, а весь його контент заданий у полі «*Текст сторінки*», то замість шаблону виводитиметься весь вміст цього поля. Також вміст поля «*Текст сторінки*», як і всіх інших параметрів, можна виводити в рамках підключеного *шаблону*, розміщуючи їх у різних місцях, у різній послідовності, як завгодно і скільки завгодно разів. Це робить систему, поряд з іншими її можливостями, гнучкою і універсальною.

Структура сайту, всі параметри й налаштування зберігаються в базі даних, інформація з неї зчитується процесором, який на основі *макета* і *шаблонів* генерує вміст сторінок сайту. *Макет* і *шаблони*, зі свого боку, верстаються з використанням спеціальних функцій, що генерують *фрагменти* сторінок за вказаними параметрами й визначеним стилем. Стилі налаштовуються у файлі стилів. Здійснюється таке моделювання за допомогою текстових редакторів або програм для роботи з *html* і *css*.

7.1. Створення структури сайту

Створення структури сайту здійснюється в панелі адміністратора, на головній сторінці якої виводиться структура елементів сайту у вигляді дерева. Додавання нових елементів здійснюється за допомогою кнопки «Добавить элемент». Під час додавання (створення) нового елемента обов’язково вказується його

належність до іншого елемента або кореневого вузла, а також тип елемента. Система залежно від типу елемента надає йому ті чи інші можливості та права, але основний набір параметрів для кожного типу залишається однаковим.

Якщо елемент є вузлом, він у структурі елементів сайту є вузловим і може бути представлений у головному меню у вигляді пункту меню, повноцінною *web*-сторінкою, може виконувати функції зв'язку між підлеглими елементами й батьківським елементом (якщо він належить до якогось елемента). Якщо елемент є кінцевою сторінкою, він має такі ж самі властивості, як і вузол, але є кінцевим елементом.

Якщо елемент є зображенням чи файлом, він має властивості, відповідні до його типу. Ці властивості характеризуються додатковими параметрами елемента, що наведені в таблиці 6.1. Характерним є те, що для елемента з типом *кінцева сторінка* параметр «*Файл загрузки*» містить прикріплений до сторінки файл, який завантажуватиметься за гіперпосиланням, що буде його представляти на сторінці, а для елемента з типом *зображення* параметр «*Файл загрузки*» містить саме файл зображення, для елемента з типом *файл* параметр «*Файл загрузки*» містить файл будь-якого типу, дозволеного системою для завантаження на сервер.

Якщо типом елемента *новина*, він володіє відповідними властивостями, які також наведені в таблиці 6.1, і система оброблює цей елемент як *web*-сторінку сайту новин. Якщо типом елемента *акція* або *товар*, цей елемент у системі обробляється, як *web*-сторінка сайту інтернет-магазину і має відповідні властивості (табл. 6.1).

Однією з особливостей системи також є те, що для кожного з елементів окремо можна встановити незалежний від основної структури зв'язок з будь-яким іншим елементом структури без встановлення підлеглості, причому кількість таких зв'язків може бути необмеженою. Це здійснюється в розділі «Связи элемента», посилання на який з'являється в панелі управління з відкриттям сторінки властивостей елемента (рис. 6.8).

7.2. Створення макета та шаблонів

Під час створення макета сайту можна керуватися однією із структур із розташуванням *головного меню*. Зазвичай використовуються структура з горизонтальним розташуванням головного меню та структура з вертикальним розташуванням головного меню.

Рис. 7.1

Структурно макет сайту поділяється на декілька областей: *«Заголовок сайту» (Header), «Рядок меню» (Menu bar), «Вміст сторінки» (Page content), «Відвал» (Footer).*

В області *«Заголовок сайту»* зазвичай виводяться логотип (ліворуч), форма швидкого пошуку, область додаткового меню (за потреби), якщо сайт є інтернет-магазином – область кошика (праворуч). Також у заголовку можуть виводитися фонові зображення (рис. 7.2).

В області *«Рядок меню»* виводиться головне меню сайту. Воно складається з пунктів меню, також підпунктів, через які можна здійснювати навігацію по сторінках поточного сайту.

В області *«Вміст сторінки»* виводиться вміст поточної сторінки сайту.

У нижній області, що має назву *«Підвал»*, зазвичай виводиться додаткова інформація, що стосується сайту та пов’язаною з ним інформації, а також може виводитись область додаткового меню.

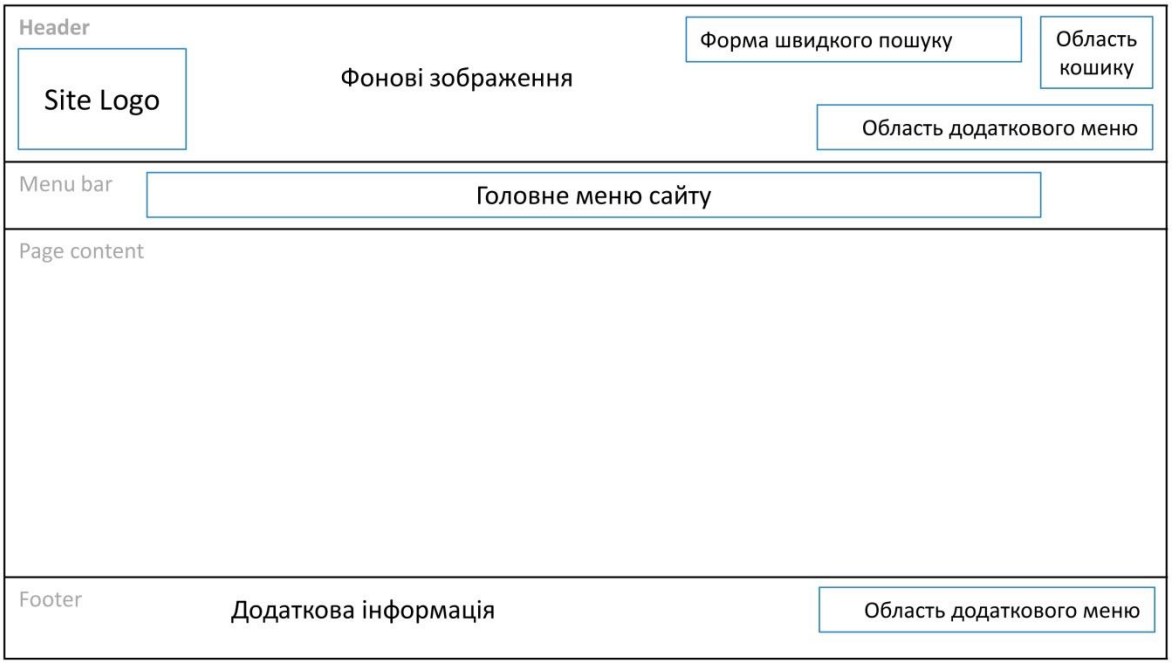


Рис. 7.2

Макет сайту зазвичай моделюється у файлі макета `layout.htm` (рис. 7.3). У ньому здійснюється основна *html*-розмітка, вказуються файли підключення *css*-стилів і *js*-скриптів, а також основні функції виведення, як-от `mainmenu()` і `curpage()`. Функція `mainmenu` виводить головне меню сайту. Функція `curpage` – вміст шаблону поточної сторінки або вміст поля «*Текст страницы*», якщо шаблон не встановлено.

```
1  <html>
2      <head>
3          <title><? print $title; ?></title>
4          <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5          <meta http-equiv="cache-control" content="no-cache">
6          <meta http-equiv="pragma" content="no-cache">
7          <meta name="robots" content="index,follow">
8          <meta name="description" content="">
9          <meta name="keywords" content="">
10         <link rel="stylesheet" type="text/css" href="wstyles/general.css">
11         <link rel="stylesheet" type="text/css" href="wstyles/styles.css">
12         <script type="text/javascript" src="wjs/ajax.js"></script>
13         <script type="text/javascript" src="wjs/processor.js"></script>
14         <script type="text/javascript" src="wjs/common.js"></script>
15         <script type="text/javascript" src="wjs/current.js"></script>
16     </head>
17     <body><? mainmenu(0,5); curpage(); ?></body>
18 </html>
```

Рис. 7.3

У кожному із шаблонів моделюється структура тієї чи іншої сторінки з використанням *html*-розмітки, функцій фрагментів

системи *W#CMS*, а також у разі потреби блоків *php*- і *js*-скриптів. Використання функцій фрагментів системи повинно здійснюватися за правилами організації роботи *php*-скрипту, тобто функції і змінні системи повинні перебувати в межах *php*-блока.

На рис. 7.4 наведено приклад шаблону сторінки товару для розширення системи *W#CMS.shop*.

```
<?dialogwin(); epath($eid);?>
<table class="igoods">
  <tr>
   <td>
    <h1><?print $naz;?></h1>
    <input type="button" class="btn" value="Заказать"
    onclick="commondialog('22&gid=<?print $eid;?>');">
    <div class="image"><?if($preimg) print "<img src=\"$preimg\">";?></div>
    <div class="price">Цена: <?print $evalue;?><span><?print $unit;?></span></div>
    <div class="about"><?print $pagetext;?></div>
   </td>
  </tr>
  <tr><td><input type="button" value="Назад" class="btn" onclick="history.back();"></td></tr>
</table>
<?
elim(12);
//Prepare for pager
global $pgno,$lim,$n_elem;
$psort=getValue('PSORT',$eid);
$pgno=ceil($psort/$lim);
$res=wcms_query(0,"SELECT COUNT(ID) FROM ETREE WHERE FATHID=$fathid AND ENAB=1");
$n_elem=wcms_result($res);
$eid=$fathid;
pager(10);
?>
```

Рис. 7.4

Тут показано використання таких функцій системи: `dialogwin()`, `epath()`, `elim()`, `pager()`, `getValue()`, `wcms_query()`, `wcms_result()`, а також змінних виведення системи: `$eid`, `$fathid`, `$naz`, `$preimg`, `$evalue`, `$unit`, `$pagetext`, спеціальних змінних системи: `$pgno`, `$lim`, `$n_elem`. Опис змінних і функцій системи представлений у наступному розділі.

Після створення макет підключається до сайту через менеджер файлів у відповідному розділі. Кожний створений шаблон підключається до елемента структури сайту на сторінці редагування елемента.

7.3. Використання менеджера файлів системи

Для управління файлами конфігурації, налаштувань, додаткових скриптів, а також іншими додатковими файлами, необхідними для роботи створюваного сайту, в системі існує менеджер файлів (рис. 7.5). Він складається з трьох вікон: вікна «Файлы конфигурации, настроек и дополнительных скриптов», вікна провідника по файловій структурі системи, вікна перегляду вмісту файлу.

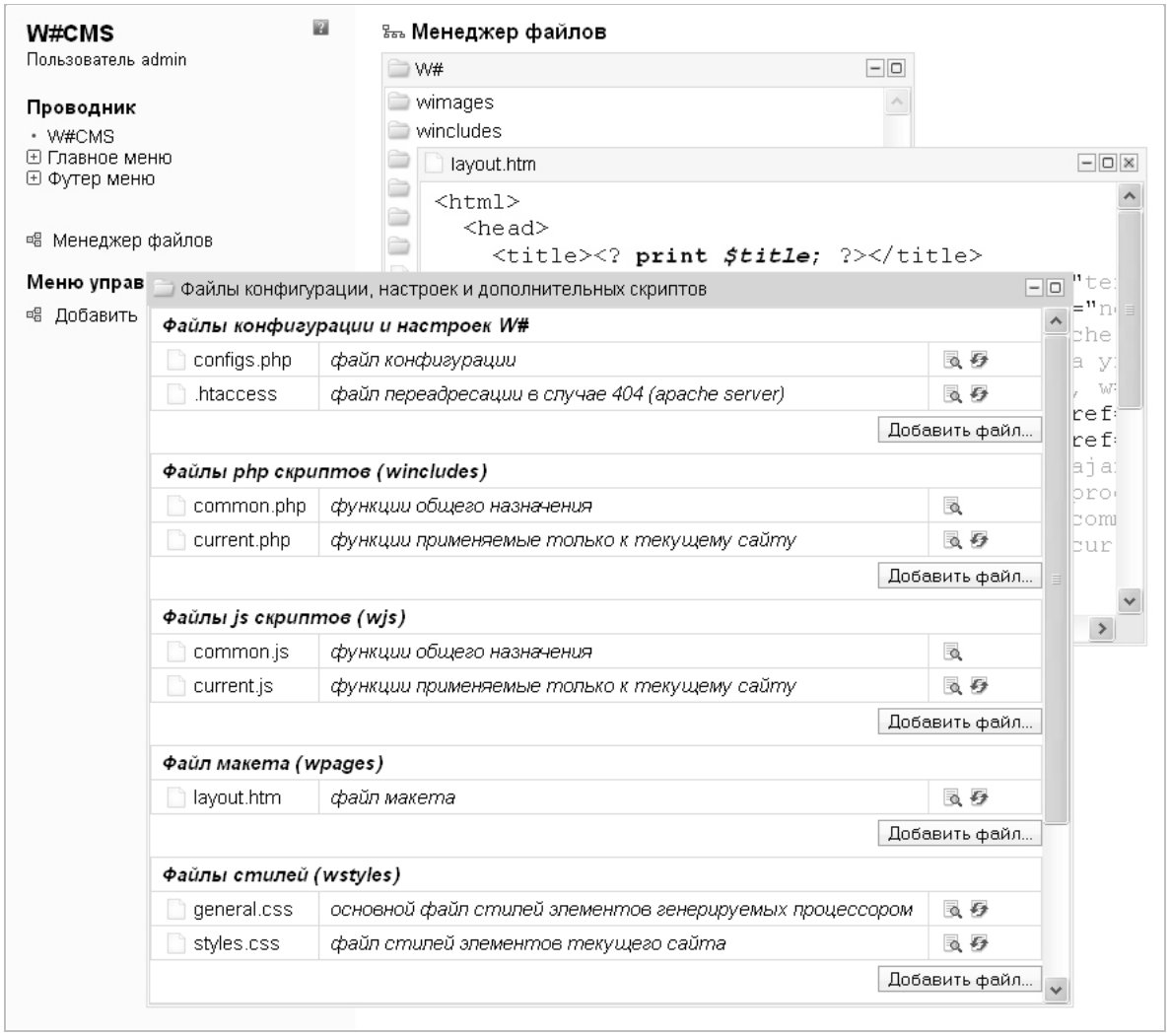


Рис. 7.5

Додавання необхідних файлів, оновлення наявних і видалення тих, видаляти які дозволено, здійснюються у вікні «Файлы конфигурации, настроек и дополнительных скриптов» (рис. 7.6). Структурно це вікно поділяється на шість частин. У кожній з них

групами за призначенням виводиться список файлів конфігурації і налаштувань, *php*-скриптів, *js*-скриптів, макета, файлів стилів, файлів графіки інтерфейсу.

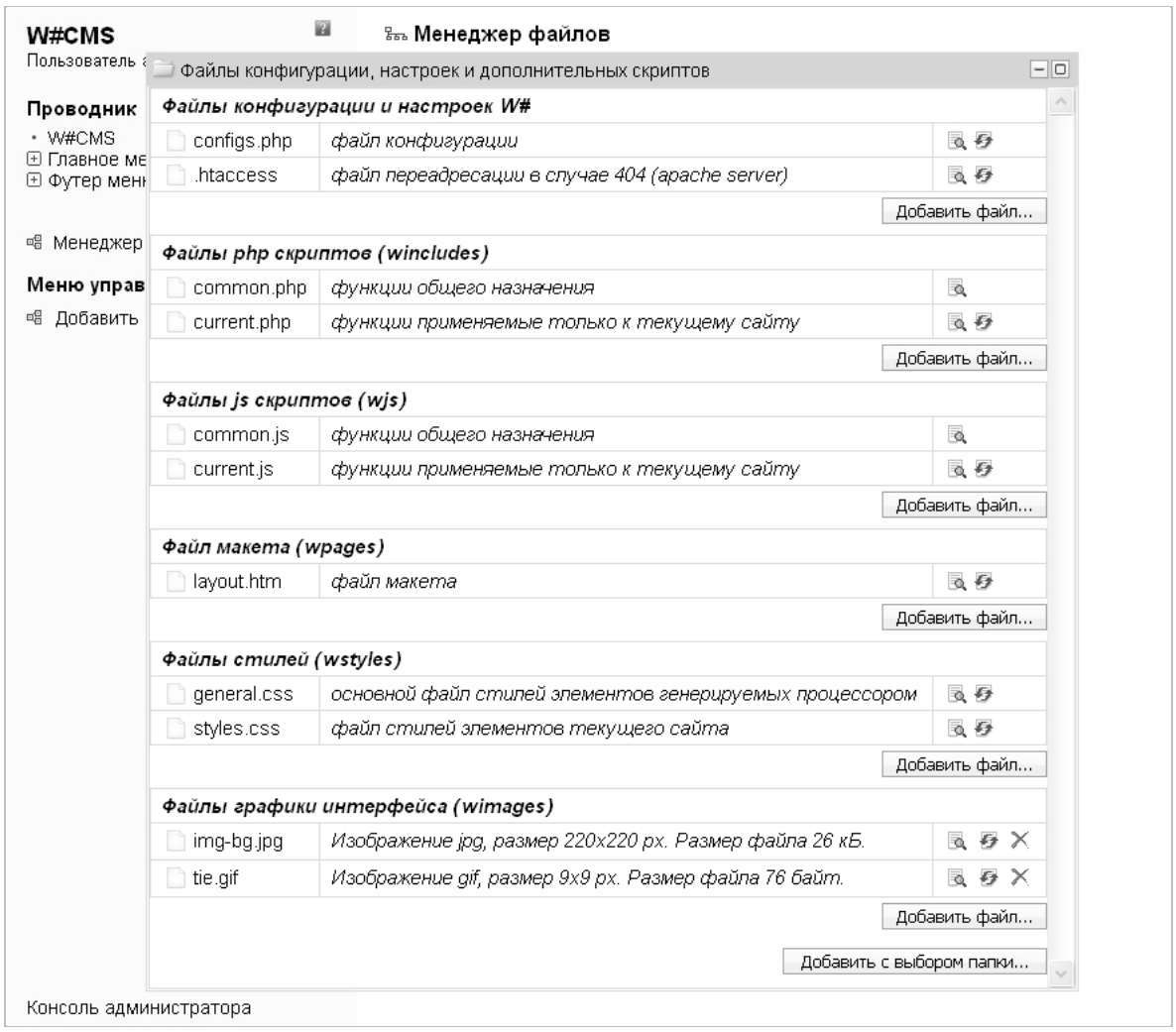


Рис. 7.6

Для кожного з файлів виводиться його ім’я та короткий опис, а також у вигляді іконок – кнопки операцій «Просмотреть» , «Обновить» , «Удалить» . Окремо до кожної групи можна додати новий файл (кнопка «Добавить файл…»), що буде завантажено в папку, ім’я якої вказано в дужках у рядку назви групи. Також здійснити додавання файлу можна за вибором папки, у яку додається файл, за допомогою кнопки «Добавить с выбором папки…». Вибір файлу для додавання здійснюється за допомогою діалогового вікна відкриття файлу браузера.

У вікні файлів і папок (рис. 7.7) відображаються дозволені для перегляду файли та папки системи. До них належать ті файли та папки, які можна перезаписувати, редагувати й видаляти. Файли самої системи, до яких доступ заборонено, не відображаються. У рядку назви вікна виводиться шлях до поточної папки. Коренева папка системи в менеджері файлів позначається ім'ям W#.

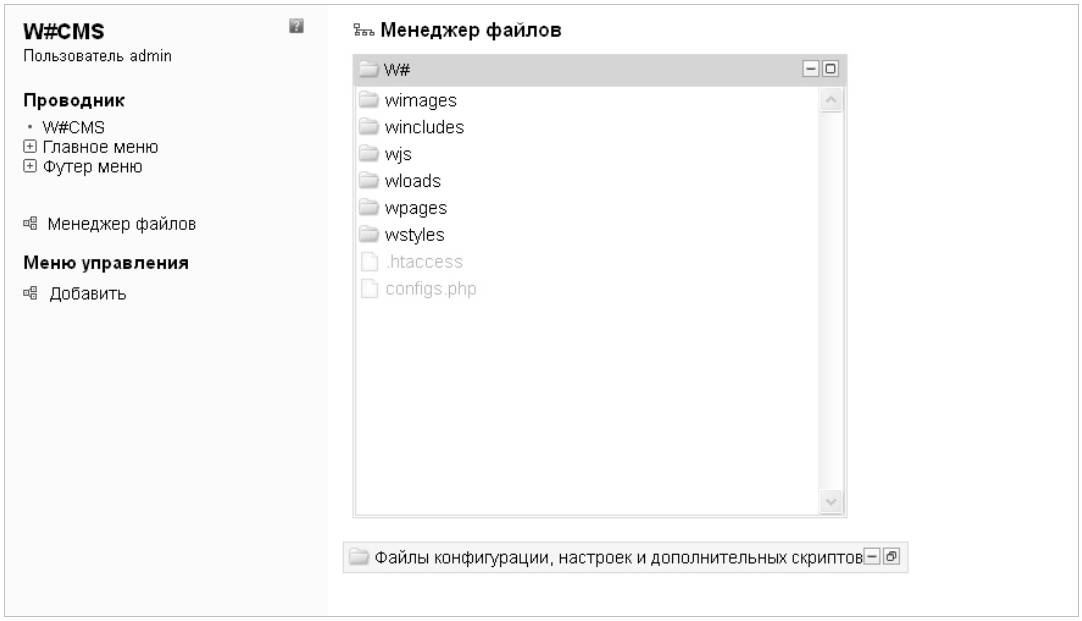


Рис. 7.7

У вікні перегляду файлів (рис. 7.8) виводиться вміст вибраного файлу. Якщо файл текстовий, формату *html*, *php* або *js*-скриптів, стилів *css*, його вміст для зручності перегляду виводиться у вигляді попередньо відформатованого тексту, розміченого різним кольором і шрифтом.

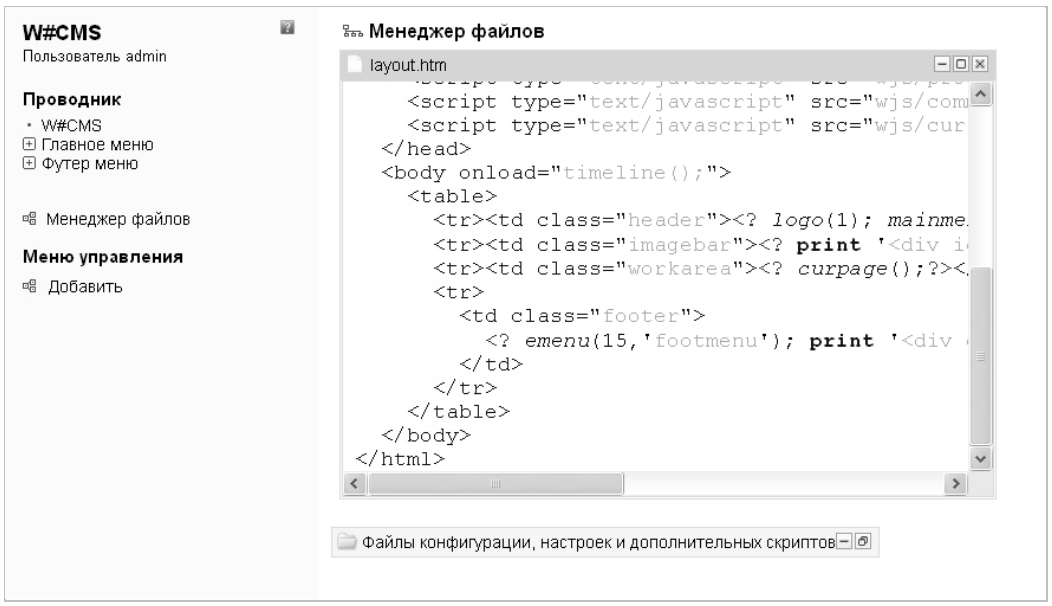


Рис. 7.8

Якщо файл графічний, формату *jpg*, *gif*, *png*, у вікні перегляду виводиться графічний вміст вибраного файлу.

7.4. Використання меню управління системи

У системі передбачена можливість додавання додаткових таблиць бази даних і управління їх вмістом за допомогою меню управління. Для цього створюється окремий пункт меню управління, до якого приєднуються нова таблиця бази даних і файл її обробки адміністратором сайту в панелі адміністратора. Для створення такого пункту меню потрібно в розділі «Меню управления» панелі управління вибрати пункт «Добавить» (рис. 6.7). Під час створення кожного пункту меню вказується його назва, ім’я таблиці, файл обробки й управління таблицею в панелі адміністратора. Також у разі потреби можна задати файл стилю (рис. 7.9).

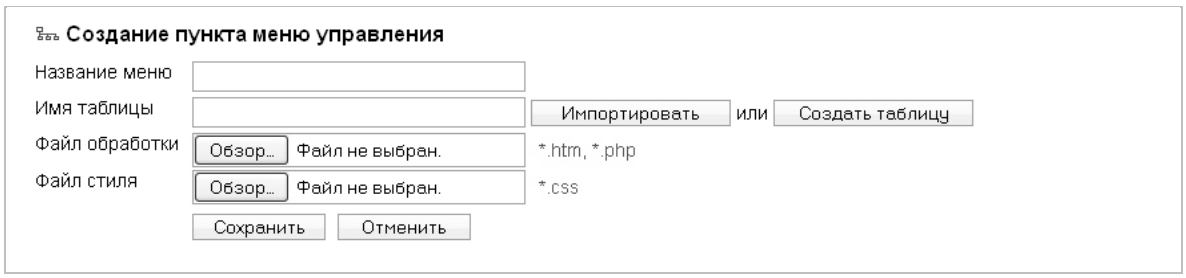


Рис. 7.9

Таблицю бази даних під час створення пункту меню управління можна або імпортувати з `*.sql` файлу, або створити. Імпорт здійснюється завантаженням файлу за допомогою відповідного поля завантаження (рис. 7.10), яке з'являється після натискання на кнопку «Импортировать».

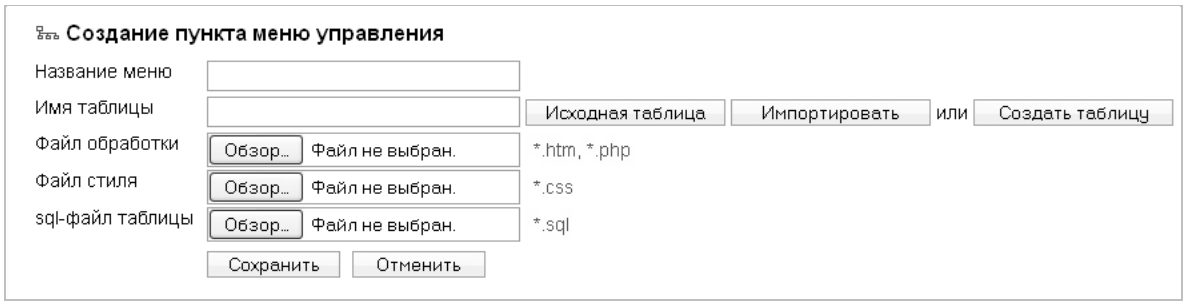


Рис. 7.10

Для створення таблиці потрібно після натискання кнопки «Создать таблицу» ввести назви й тип полів таблиці, у разі потреби – довжину, а також задати основні параметри створюваної таблиці: тип таблиці, порівняння, первинний ключ, `auto_increment` (рис. 7.11). Під час створення таблиці кожне нове поле легко додається за допомогою кнопки «Добавить поле».

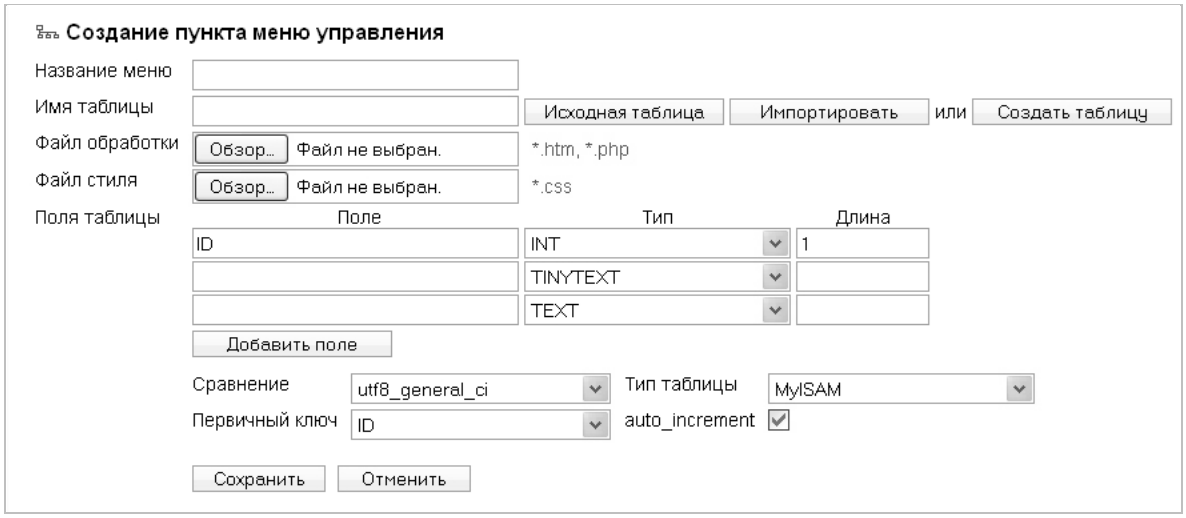


Рис. 7.11

Після створення пункту меню управління він з'являється у списку пунктів меню в розділі «Меню управления». Керування

доступом до пунктів меню здійснюється зі сторінки «Управление доступом к пунктам меню управления» (рис. 7.12), яку можна відкрити, клацнувши мишею по назві розділу «Меню управления». На цій сторінці у вигляді таблиці виводиться список пунктів меню, імена таблиць, що прикріплені до пунктів меню, імена файлів обробки, файлів стилів. У разі потреби за допомогою прапорців можна вимикати доступ до того чи іншого пункту меню. Вимкнений пункт меню не відображатиметься в списку пунктів меню лівої панелі управління й буде недоступним для роботи з ним другорядними адміністраторами сайту.

Для кожного з пунктів меню на сторінці «Настройки пункта меню управления» можна переглянути відповідні налаштування: назва меню, ім’я таблиці, її порівняння, файл обробки й управління таблицею адміністратором у панелі адміністратора і файл стилю, а також структуру таблиці з назвою полів, їх типом і довжиною (рис. 7.13). Ця сторінка відкривається зі сторінки пункту меню кнопкою «Свойства меню», яка з іншими кнопками управління з’являється в нижній частині панелі адміністратора з відкриттям відповідної сторінки.

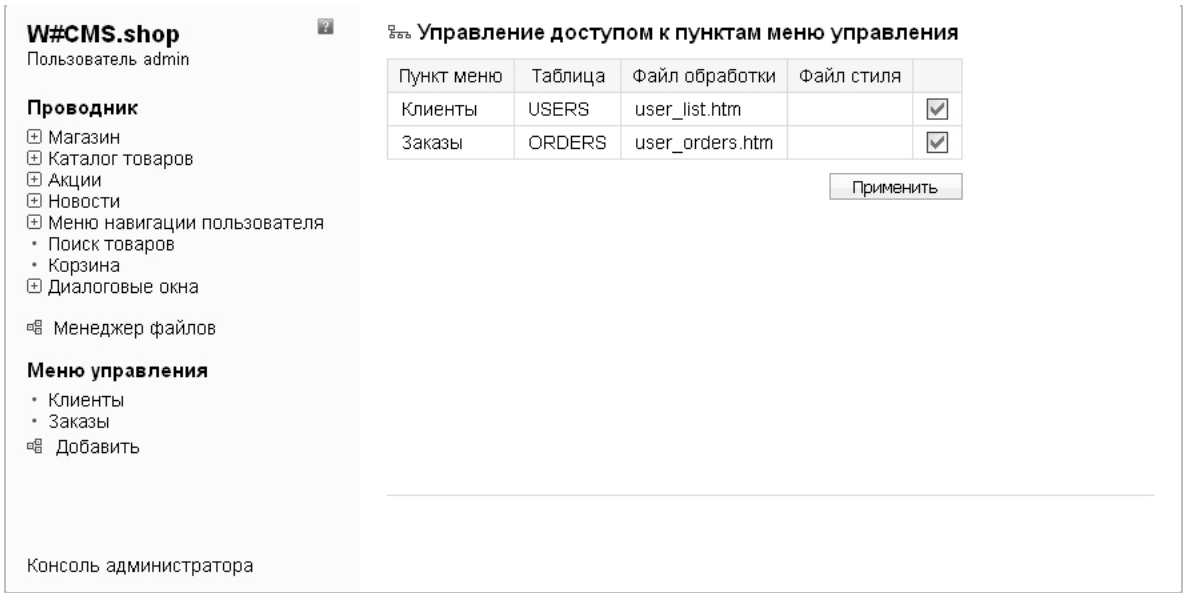


Рис. 7.12

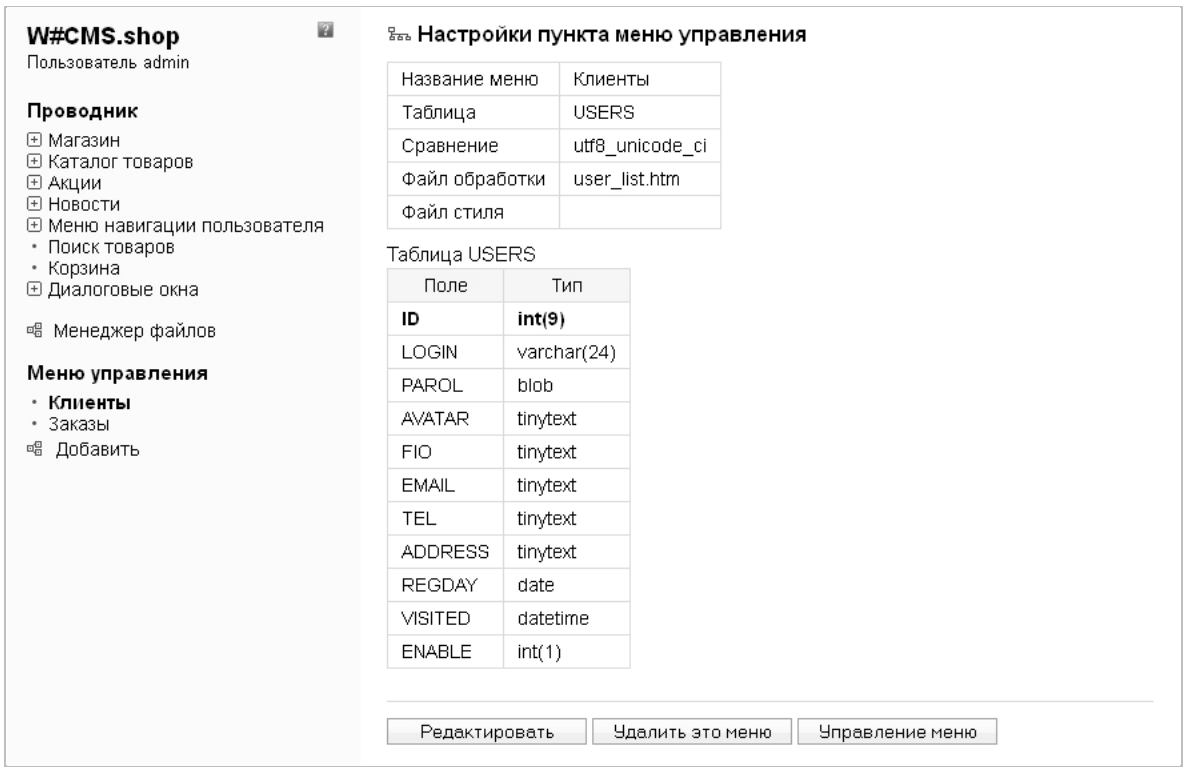


Рис. 7.13

У разі потреби всі налаштування пункту меню управляння можна змінити. Для цього слід натиснути кнопку «Редактировать» і на сторінці «Редактирование пункта меню управления» здійснити всі необхідні зміни (рис. 7.14). На цій сторінці можна змінити ім’я таблиці, у разі потреби імпортувати нову таблицю замість існуючої або змінити структуру існуючої таблиці, перезавантажити оновлений файл обробки у форматі `*.htm` чи `*.php`, файл стилю у форматі `*.css`. Для зміни структури таблиці потрібно натиснути на кнопку «Изменить таблицу», після чого відкриються поля таблиці для редагування, де можна змінити назву поля, його тип, довжину, додати нові поля або видалити непотрібні, а також змінити тип таблиці, її порівняння, первинний ключ.

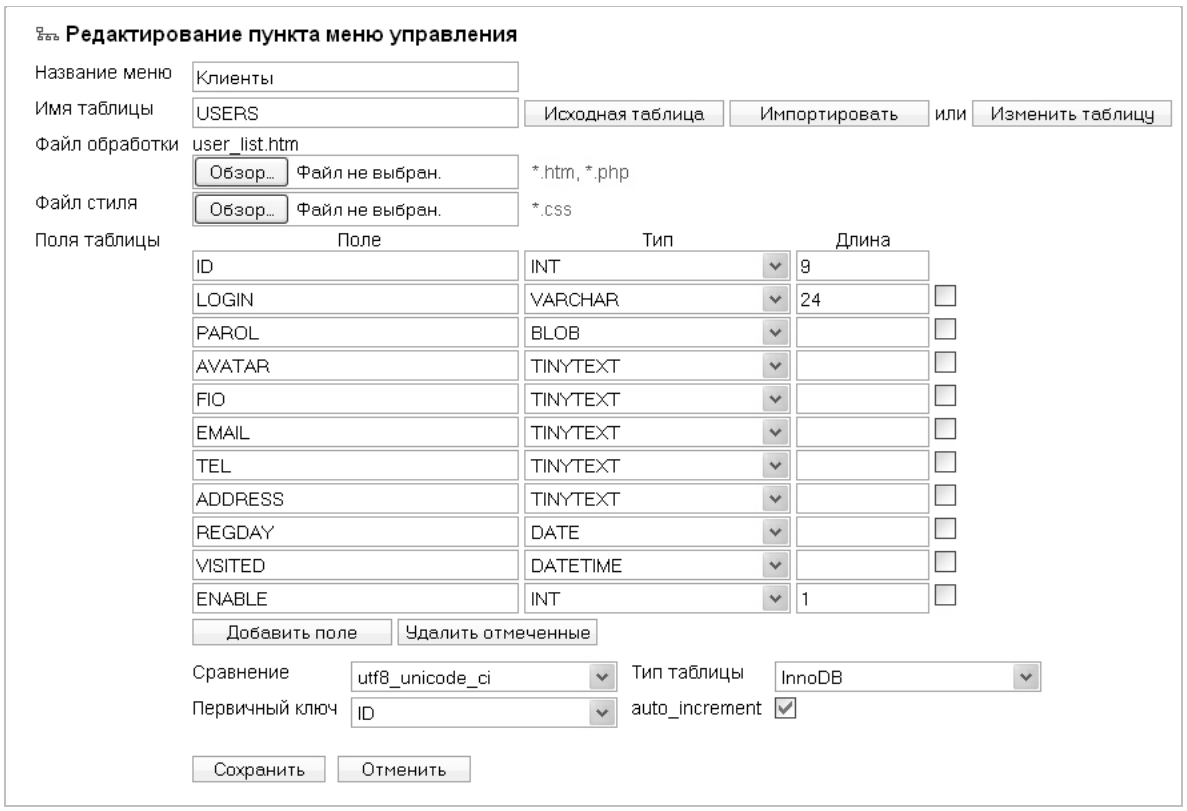


Рис. 7.14

7.5. Використання функцій-фрагментів

Функції фрагментів, по суті, є функціями-конструкторами *html*-сторінок, тобто кожна з них займається формуванням тієї чи іншої частини (фрагмента) або блока сторінки. Вони використовуються в лейауті і шаблонах і є їх складовими частинами, спільно формують *web*-сторінку.

Виведення вмісту шаблону поточної сторінки:

```
<? curpage(); ?>
```

Цей фрейм виведе такий *html*-фрагмент:

```
<div id="curpage">
[вміст шаблону поточної сторінки і тексту сторінки]
</div>
```

Примітка: ця функція у файлі макета (`layout.htm`) визначає те місце, куди буде виведений вміст шаблону поточної сторінки або вміст параметра «*Текст страницы*» (поле PAGETEXT, змінна `$pagetext`).

Виведення області побудови діалогового вікна:

```
<? dialogwin(); ?>
```

Цей фрейм виведе такий *html*-фрагмент:

```
<script type="text/javascript" src="wjs/drag.js">
</script>
<div id="dialogwin" class="dialogwin"></div>
```

Примітка: ця функція визначає область, у яку буде виведений вміст діалогового вікна, що викликається *js*-функцією `commondialog()` або `doShowWin()`.

Виведення логотипу:

```
<? logo(1); ?>
```

Цей фрейм в обгортці `<div class="logo">...</div>` виведе *html*-фрагмент у вигляді зображення логотипу «*Картинка превью*» елемента `id=1`. Для нього автоматично буде сформоване гіперпосилання. Також, якщо для елемента буде вказана «*Альтернативная картинка*», то з наведенням курсора миші на логотип його зображення змінюватиметься.

Виведення шляху до елемента:

```
<? epath(33); ?>
```

Цей фрейм в обгортці `<div class="epath">...</div>` виведе *html*-фрагмент у вигляді шляху до елемента із зазначеним id. Кожен складовий елемент шляху, крім самого елемента, має гіперпосилання.

Побудова дерева з підлеглих елементів певного вузла:

```
<? etree(2,3); ?>
```

Цей фрейм в обгортці `<div class="etree">...</div>` виведе *html*-фрагмент у вигляді дерева підлеглих елементів вузла `$id` до рівня глибиною, зазначеною параметром `$depth`. У цьому прикладі `$id=2`, `$depth=3`. Якщо `$id=0`, то побудова починається від кореня. Глибина є необмеженою. Кожен елемент дерева має гіперпосилання.

Побудова головного меню з підлеглих елементів певного вузла:

```
<? mainmenu(2,3); ?>
```

Цей фрейм в обгортці `<div id="mainmenu">...</div>` виведе *html*-фрагмент у вигляді головного меню сайту, що складається з підлеглих елементів вузла `$id` до рівня глибиною, зазначеною параметром `$depth`. У цьому прикладі `$id=2`, `$depth=3`. Якщо `$id=0`, то побудова починається від кореня. Глибина необмежена.

Побудова меню з підлеглих елементів 1-го рівня певного вузла:

```
<? emenu(2); ?>
```

Цей фрейм виведе *html*-фрагмент `<menu class="emenu">... </menu>` у вигляді меню, що складається з підлеглих елементів 1-го рівня вузла `$id`. Для цієї функції можна використовувати необов'язковий параметр `$style`, у якому можна задати особливий стиль елемента:

```
<?
  emenu(3,'own');
  emenu(4,'class=sec');
  emenu(5,'style="float:left;"');
?>
```

Побудова меню з елементів, перелічених у списку:

```
<? smenu('2,3,5,8,9'); ?>
```

Цей фрейм виведе *html*-фрагмент `<menu class="smenu">... </menu>` у вигляді меню, що складається з елементів, перелічених в параметрі `$list` через кому. Для цієї функції також можна використовувати необов'язковий параметр `$style`, у якому можна задати особливий стиль елемента.

Виведення списку елементів 1-го рівня певного вузла:

```
<? elist(2); ?>
```

Цей фрейм виведе *html*-фрагмент `<ul class="elist">... </ul>` у вигляді списку з гіперпосиланнями, що складається з підлеглих елементів 1-го рівня вузла `$id`. Для цієї функції також можна використовувати необов'язковий параметр `$style`, у якому можна задати особливий стиль елемента.

Виведення списку перелічених елементів:

```
<? slist('2,3,5,8,9'); ?>
```

Цей фрейм виведе *html*-фрагмент `<ul class="slist">...</ul>` у вигляді списку з гіперпосиланнями, що складається з елементів, перелічених у параметрі `$list` через кому. Для цієї функції також можна використовувати необов'язковий параметр `$style`, у якому можна задати особливий стиль елемента.

Виведення таблиці, що складається зі списку елементів 1-го рівня певного вузла:

```
<?
  $heads='№,назва,опис,примітка';
  $fields='#,NAZ,PRETEXT,PRIM';
  etable(2,$heads,$fields);
?>
```

Цей фрейм виведе *html*-фрагмент `<table class="etable"> ...</table>` у вигляді таблиці, що складається з рядків, зазначених у `$fields` полів підлеглих елементів 1-го рівня вузла `$id`, із зазначеними в `$heads` заголовками. Якщо `$heads=''`, то рядок заголовка буде відсутній.

Виведення таблиці, що складається зі списку перелічених елементів:

```
<?
  $heads=' №,назва,опис,примітка,дата';
  $fields='#,NAZ,PRETEXT,PRIM,BDATE
  _@orderby(BDATE DESC)@format(BDATE,1)';
  stable('2,3,5,8,9',$heads,$fields);
?>
```

Цей фрейм виведе *html*-фрагмент `<table class="stable"> ...</table>` у вигляді таблиці, що складається з рядків, зазначених у `$fields` полів, перелічених у параметрі `$list` елементів, через кому, із зазначеними в `$heads` заголовками.

Виведення списку елементів 1-го рівня певного вузла у вигляді рядів таблиці:

```
<?
  $heads='№,назва,опис,примітка';
  $fields='#,NAZ,PRETEXT,PRIM';
  erows(2,$heads,$fields);
?>
```

Цей фрейм виведе *html*-фрагмент `<table class="erows">...</table>` у вигляді таблиці, що складається з рядків, зазначених у `$fields` полів підлеглих елементів 1-го рівня вузла `$id`, із зазначеними в `$heads` заголовками. Якщо `$heads=''`, то рядок заголовка буде відсутній.

Виведення списку перелічених елементів у вигляді рядів таблиці:

```
<?
  $heads=' №,назва,опис,примітка,дата';
  $fields='#,NAZ,PRETEXT,PRIM,BDATE
  _@orderby(BDATE DESC)@format(BDATE,1)';
  srows('2,3,5,8,9',$heads,$fields);
?>
```

Цей фрейм виведе *html*-фрагмент `<table class="srows">...</table>` у вигляді таблиці, що складається з рядків, зазначених у `$fields` полів, перелічених у параметрі `$list` елементів, через кому, із зазначеними в `$heads` заголовками.

Виведення списку елементів 1-го рівня певного вузла у вигляді «плитки»:

```
<?
  $pattern='|#|<h1>|NAZ|</h1><div>|PREIMG|</div>';
  etile(2,4,'',$pattern);
?>
```

Цей фрейм в обгортці `<div class="etile">...</div>` виведе *html*-фрагмент у вигляді списку «плиткою», що складається з підлеглих елементів 1-го рівня вузла `$id`. Кожен з елементів буде поміщений у контейнер `<div class="tile">...</div>`. У параметрі `$pattern` задається шаблон виведення полів елемента в контейнері `tile`. У цьому прикладі буде виведений список

підлеглих елементів вузла $id=2 у 4 стовпці ($ncols=4). $style='' означає, що стиль буде використаний за стандартним налаштуванням.

Виведення списку перелічених елементів у вигляді «плитки»:

```
<?
  $pattern='|#|<h1>|NAZ|</h1><div>|PREIMG|</div>
  <div>|BDATE|</div>_@orderby(BDATE DESC)
  @format(BDATE,1)';
  stile('2,5,8,9',4,'',$pattern);
?>
```

Цей фрейм в обгортці ... виведе *html*-фрагмент у вигляді списку «плиткою», що складається з елементів, перелічених у параметрі $list. Кожен з елементів буде поміщений до контейнера ... . У параметрі $pattern задається шаблон виведення полів елемента в контейнері tile. У цьому прикладі буде виведений список елементів з ID 2,5,8,9 у 4 стовпці.

Виведення картинки прев'ю елемента:

```
<? preimg(2); ?>
```

Цей фрейм виведе *html*-фрагмент <img class="preimg" src="..."> для зображення «*Картинка превью*» елемента $id=2. Для цієї функції використовується необов'язковий параметр $style, у якому можна задати особливий стиль елемента.

Виведення альтернативної картинки елемента:

```
<? altimg(2); ?>
```

Цей фрейм виведе *html*-фрагмент <img class="altimg" src="..."> для зображення «*Альтернативная картинка*» елемента $id=2. Для цієї функції також використовується необов'язковий параметр $style.

Виведення картинки прев'ю елемента зі зміною за наведення на неї курсора миші:

```
<? rollimg(2); ?>
```

Цей фрейм виведе *html*-фрагмент `<img class="rollimg" src="...">` для зображення «*Картинка превью*» елемента `$id=2`, яке змінюватиметься на зображення «*Альтернативная картинка*» з наведенням на нього курсора миші. Для цієї функції також використовується необов'язковий параметр `$style`. Для цієї функції фрагмент `img` буде обгорнутий у тег гіперпосилання елемента.

Виведення зображення файлу завантаження елемента:

```
<? loadimg(2); ?>
```

Цей фрейм виведе *html*-фрагмент `<img class="loadimg" src="...">` для зображення «*Файл загрузки*» елемента `$id=2`. Для цієї функції також використовується необов'язковий параметр `$style`.

Виведення назви елемента:

```
<? naz(2); ?>
```

Цей фрейм в обгортці `<div class="naz">...</div>` виведе назву елемента `$id=2`. Для цієї функції також використовується необов'язковий параметр `$style`.

Виведення примітки елемента:

```
<? prim(2); ?>
```

Цей фрейм в обгортці `<div class="prim">...</div>` виведе примітку елемента `$id=2`. Для цієї функції також використовується необов'язковий параметр `$style`.

Виведення опису (претексту) елемента:

```
<? pretext(2); ?>
```

Цей фрейм в обгортці `<div class="pretext">...</div>` виведе опис (претекст) елемента `$id=2`. Для цієї функції також використовується необов'язковий параметр `$style`.

Виведення тексту сторінки елемента:

```
<? pagetext(2); ?>
```

Цей фрейм в обгортці ... виведе *html*-текст сторінки елемента $id=2. Для цієї функції також використовується необов'язковий параметр $style.

Виведення пейджера сторінок:

```
<?
  elim(8);
  etile(2,4,'','<h1>|NAZ|</h1><div>|PREIMG|</div>');
  pager(5);
?>
```

Функція pager() в обгортці ... виведе *html*-фрагмент у вигляді ярликів сторінок, кількість яких задається параметром $ntabs. У цьому прикладі $ntabs=5. Перед цією функцією потрібно викликати функцію elim(), яка визначає максимальну кількість елементів на сторінці, а також вона використовується спільно з функціями etile(), stile() або searchresult() після їх виклику.

Виведення форми пошуку:

```
<?
  elim(10);
  searchform($eid,'Шукати',1);
  searchresult();
  pager(5);
?>
```

Функція searchform() в обгортці ... виведе *html*-фрагмент у вигляді форми для введення пошукового запиту з текстовим полем і кнопкою. У параметрі $id вказується ID сторінки виведення результату запиту. У цьому прикладі – це ID поточної сторінки, який міститься в змінній $eid. Назва кнопки задається параметром $btn, індекс пошукової форми (0 – quick; 1 – string) задається параметром $ind. У цьому прикладі $btn='', $ind=0.

Виведення результату пошуку

```
<?
  elim(10);
  searchform($eid,'Шукати',1);
  searchresult('1,2,5,7');
  pager(5);
?>
```

Функція `searchresult()` виведе *html*-фрагмент результату виконання пошукового запиту `<table class="searchresult"> ...</table>` у вигляді таблиці, що складається з рядків з результатом пошуку. У необов'язковому параметрі `$typelist` через кому задається список типів елементів, що беруть участь у пошуку. Перед цією функцією потрібно викликати функцію `elim()`, яка визначає максимальну кількість елементів на сторінці, після неї – функцію `pager()`.

Виведення форми для відправлення повідомлення:

```
<? mailform('Відправити'); ?>
```

Цей фрейм виведе *html*-фрагмент `<form class="mailform"> ...</form>` у вигляді форми введення даних для відправлення повідомлення. За допомогою параметра `$btn` задається назва кнопки. У цьому прикладі – `$btn='Відправити'`.

Контрольні запитання

1. Як у W#CMS створюється структура сайту? Як додаються елементи, задаються їх властивості та інші параметри?
2. З яких частин структурно складається web-сторінка?
3. Для чого призначаються макет і шаблони сторінок?
4. Як у W#CMS підключаються шаблони сторінок до елементів дерева сайту?
5. Для чого та як у панелі адміністратора W#CMS використовується менеджер файлів?
6. Як у менеджері файлів здійснюється перегляд і додавання файлів php- та js-скриптів, css-стилів, файлів графіки інтерфейсу?

7. Для чого та як у панелі адміністратора W#CMS використовується меню управління?

8. Як здійснюється підключення додаткових таблиць до бази даних системи та керування ними?

9. Для чого призначені та як використовуються функції-фрагменти?

10. Якою функцією та як можна побудувати область головного меню сайту?

11. Яка функція використовується для побудови області логотипу?

Лекція 8

СИСТЕМИ УПРАВЛІННЯ CRM, HRM, ERP

8.1. Системи управління взаємовідносинами з клієнтами CRM

CRM (Customer Relationship Management) – система, яка призначена для автоматизації взаємодії організації із замовниками (клієнтами) з метою підвищення рівня продажу, оптимізації маркетингу й поліпшення обслуговування клієнтів.

CRM-система збирає дані кожного клієнта в одній панелі управління. За допомогою цієї інформації можна відстежувати шлях покупця та робити релевантні пропозиції на кожному етапі. Унаслідок такого застосування цикл продажів товарів і послуг скорочується на 8–14 %.

Існує безліч сервісів для малих, середніх і великих компаній із різних сфер бізнесу. Проте є низка завдань *CRM*, а саме:

- **консолідація даних клієнтів**. *CRM*-система збирає контакти лідів і покупців, їх демографічні дані та іншу інформацію забезпечуючи легкий доступ до неї;
- **відстеження взаємодій та активності**. *CRM*-системи дають змогу відстежувати комунікацію з клієнтами в чатах через месенджери, телефон, email та інші канали;
- **оцінка продуктивності**. Професійна *CRM*-система дає змогу отримувати звіти з детальними даними щодо ефективності взаємодії компанії з клієнтами;
- **автоматизація рутинних процесів**. Автоматизація маркетингу та продажу, що є основою будь-якої *CRM*-системи.

Уся інформація про клієнтів представлена у вигляді наочних карток. Для кожного клієнта фіксуються: контактні дані, історія звернень (листи, звернення у чатах чи соціальних мережах), дзвінки разом з їх записами, подальші задачі, файли (наприклад, договори чи чеки про оплату).

Базу клієнтів у *CRM* можна сегментувати за різними параметрами для персоналізації пропозицій та комунікації – окрім

очевидного параметру імені, можна персоналізувати комунікацію по посадах, індустріях, поточних проблемах тощо.

Робота з *CRM*-системою скорочує час і допомагає автоматизувати рутинні задачі. З використанням *CRM* можна автоматизувати повну *воронку продажу* – від збору лідів, призначення відповідального менеджера до серій автоматичних листів для прогріву лідів, сповіщень, коли щось важливе відбувається з угодою або лідом, створення задач тощо. До популярних функцій автоматизації також належать масові розсилки, створення листів і документів за шаблоном.

Воронка продажу – це маркетингова модель, що описує передбачуваний шлях майбутнього покупця від першого знайомства із пропозицією чи товаром до реальної покупки.

Сучасні системи управління взаємовідносинами з клієнтами мають базові функції *CRM*, а також розширені функції, які далеко виходять за рамки колишніх процедур сортування та ведення картотек.

Усі процеси, що орієнтовані на клієнтів, можна оптимізувати та зміцнювати відносини з ними, зробивши клієнта центром усіх операцій продажу, маркетингу, комерції та обслуговування.

CRM-системи можуть вирішувати задачі, з якими весь час мають справи різні компанії.

Залежно від своїх можливостей *CRM* можна поділити на декілька видів.

- **Операційні системи *CRM*** – допомагають виконувати повсякденні процеси компанії та автоматизувати рутинні завдання.
- **Аналітичні системи *CRM*** – це величезні бази даних із детальною інформацією про клієнтів та бізнес-процеси.
- **Колективні системи *CRM*** – допомагають підвищувати ефективність взаємодії між різними відділами компанії.

Кожна бізнес-задача потребує певного рішення. Наприклад, те, що працює для досягнення маркетингових цілей, може не підходити для обслуговування клієнтів. Через це варто ретельно ставитися до

вибору платформи, яка зможе задовольнити всі бажання й вирішувати нагальні питання.

Розглянемо низку *CRM*-систем, що використовуються для бізнесу на території України та у світі. Серед них *SendPulse, KeepinCRM, KeyCRM, NetHunt, Zendesk Sell, Zoho*.

SendPulse – інструмент з ефективним набором функцій автоматизації маркетингу. Для залучення лідів можна створювати професійні мультиканальні форми підписки та чат-боти в соціальних мережах.

У системі використовується можливість конструювання автоматизованих ланцюжків повідомлень для своєчасної реакції на них із метою успішного розвитку взаємодії з лідами. Також сервіс допомагає створювати сценарії повідомлень чат-ботів для комунікації з користувачами в *Telegram, Facebook Messenger, WhatsApp, Viber* та *Instagram*.

У *CRM SendPulse* можна зберігати й оновлювати інформацію про клієнтів, створювати угоди, додавати потрібні статуси, візуалізувати процеси. Якщо здійснювати *email*-розсилки через *SendPulse* та використовувати чат-боти, у цій *CRM* можна створювати автоматичні розсилки.

У системі можна додавати членів команди і призначати відповідальних за створювані угоди. Колеги, яких користувач додає до сервісу, зможуть бачити та працювати з усіма угодами.

CRM SendPulse є безплатною системою. У ній можна вільно зберігати всю інформацію, створювати ефективну *воронку продажу,* щоб працювати з клієнтами за певним сценарієм, користуватися різними каналами для спілкування, закривати угоди.

KeepinCRM – сервіс, який дає змогу вести комунікацію з клієнтами, керувати продажами, підрядниками та складами, вести фінансовий облік і документообіг.

KeepinCRM надає можливість інтеграції з «*Новою поштою»* та *УкрПоштою*. Сервіс дає змогу автоматично заповнювати дані про клієнтів і товари, формувати та друкувати ТТН (товарно-транспортні накладні), відстежувати статуси доставки, формувати списки

кур’єра. У системі щомісяця проводять оновлення та додають нові функції для підвищення ефективності роботи.

KeepinCRM забезпечує контроль залишків на складах, надає можливість передавати товари між філіями, налаштовувати автоматичне повернення товарів на склад, робити розсилку повідомлень клієнтам через *SMS* та багато іншого.

Система інтегрується з *Instagram* та *Facebook Leads*, *Rozetka*, *Prom*, *WordPress*, *TurboSMS*. Також є можливість інтеграцій з *Telegram*, *Viber*, *ПриватБанк*, *Monobank*, *Liqpay*, *Google Sheets*, *MailChimp* та *SendPulse*.

KeyCRM підходить різним видам бізнесу. Сервіс допомагає здійснювати *онлайн*-продажі на маркетплейсах, в *Instagram* та в *інтернет*-магазинах, а також допомагає тим, хто займається бізнесом у сфері послуг.

За допомогою цієї платформи можна збирати ліди, заявки та замовлення з різних маркетплейсів і месенджерів. Використовуються вбудовані модулі *eBay*, *Amazon*, *Rozetka*, *Prom*, *Allo* та *Shopify*. Також перевагою системи є те, що сервіс швидко оформлює ТТН і автоматично відправляє дані на маркетплейси.

Система *KeyCRM* веде підрахунок товарів на складах і здійснює облік їх залишків. *KeyCRM* має величезний список інтеграцій, тому можна швидко здійснювати підключення необхідних сервісів.

За допомогою цієї *CRM* можна збирати всі заявки в одному вікні, користуватися календарем записів на послуги, збирати запити з *Telegram*, *Viber*, *Instagram*, *Facebook*, форм на сайті та вести облік витрат за матеріалами.

NetHunt допомагає автоматизувати бізнес-процеси, задачі та контроль за виконанням завдань командою.

За допомогою платформи можна відповідати на повідомлення клієнтів з різних месенджерів в одному вікні. Нові чати автоматично додаються до *CRM* та до картки клієнта. Усі дані клієнтів і чати з ними будуть надійно збережені в системі. До того ж у цій системі можна структурувати всю інформацію у зручній формі.

Якщо потрібно зосередитися на генерації лідів, система допоможе зібрати нових лідів, налаштувати сповіщення про них і «підігріти» потенційних клієнтів шляхом відправлення автоматичних листів. Завдяки цьому з *NetHunt* можна звільнити своїх співробітників від рутинних задач і дати їм можливість зосередитися на більш серйозних проблемах клієнтів.

У *CRM*-системі *NetHunt* можна призначати задачі, розподіляти їх між виконавцями, відстежувати перебіг виконання, встановлювати пріоритетність завдань.

Уніфікований спосіб постановки завдань дає змогу самим співробітникам нічого не забути і не випустити з уваги.

Zendesk Sell – це модель операційної *CRM*-системи, яка дає змогу телефонувати клієнтам, надсилати їм листи та планувати зустрічі в рамках одного сервісу.

У системі можна автоматизувати обмін даними прямо всередині за допомогою зовнішніх сервісів, наприклад *Mailchimp, Pandadoc,* і також власних додатків, створених за допомогою платформи *Zendesk Apps*. Цей інструмент надає можливості для керування лідами, починаючи від пошуку клієнтів за кількома каналами і закінчуючи створенням їх точних профілів.

Zendesk Sell має вбудовану базу даних із понад 20 мільйонів компаній і 200 мільйонів професіоналів, тому в ній можна знаходити потенційних клієнтів на основі створених профілів покупців.

Найважливішою перевагою *Zendesk Sell* є його аналітика, яка налічує понад 30 готових до використання звітів та інформаційних панелей, що налаштовуються. Останні дають змогу відстежувати активність продажів, кількість дзвінків, тривалість укладання угод, результативність та багато іншого. У звітах відображаються найважливіші показники, які можна порівнювати з фінансовими цілями реального часу.

Zoho CRM – багатофункціональна система для автоматизації робочих процесів, яку завдяки своїй гнучкості можна налаштувати під будь-які бізнес-завдання.

Програмне забезпечення *Zoho* надає користувачам багатоканальний зв'язок. Це дає змогу автоматизувати процес зв'язку з клієнтами за допомогою телефону, електронної пошти, чату та соціальних мереж.

Коли клієнт зв'язується з компанією, одразу приходить сповіщення. У процесі роботи отримується звіт про ефективність комунікації з потенційними клієнтами та покупцями.

За допомогою розмовного помічника можна швидко знайти необхідну інформацію у *CRM* і отримати прогнози щодо угод з певними клієнтами. Таким чином можна визначити потенційних клієнтів, які з більшою імовірністю принесуть прибуток.

Сервіс допомагає створювати інтерфейс користувача *CRM* відповідно до вподобань, який можна персоналізувати так, щоб програмне забезпечення відповідало вимогам і потребам.

8.2. Системи управління людськими ресурсами HRM

HRM (Human Resources Management) – система, спрямована на оптимізацію використання людських ресурсів і забезпечення організацій якісним персоналом, здатним виконувати покладені на нього трудові функції.

Управління персоналом є невід'ємною частиною автоматизованих систем управління підприємством.

Основні задачі, які вирішуються *HRM*-системою:

- комплектація штату підприємства відповідно до стратегії його розвитку в коротко-, середньо- та довгостроковій перспективах. Залучення, утримання та мотивація найбільш кваліфікованого персоналу;
- створення системи підготовки керівного резерву, забезпечення наступництва керівництва і зниження ризику кадрових втрат;
- орієнтація служби управління персоналом на досягнення виробничих результатів;
- забезпечення розвитку та навчання персоналу відповідно до цілей діяльності підприємства і його підрозділів;

- реалізація оптимальної та з низькими витратами функції обліку у сфері управління персоналом.

Про важливість вирішення цих задач для підприємства свідчить те, що втрати, які пов'язані із заміною втраченого співробітника, можуть становити від 30 до 150 % його річного окладу, залежно від рівня його знань і навичок.

Ротація кадрів виявляється серйозною проблемою, яка може істотно погіршити показники загальної ефективності підприємства. Розв'язати її, як і інші проблеми, пов'язані з персоналом, здатна *HRM*-система.

Сучасні інтегровані *HRM*-системи містять функціональні блоки, які відповідають за:

- ✓ розрахунок заробітної плати;
- ✓ облік співробітників;
- ✓ рекрутинг;
- ✓ управління ефективністю та навчанням;
- ✓ взаємодію користувачів із системою.

Вибір програми для *HR*-менеджерів та правильної системи для управління персоналом – важливий процес для компанії. Оскільки недостатньо мати списки співробітників та їх зарплат, а також *Excel* таблиці з відпустками та лікарняними, потрібна складніша система, у якій можна бачити інформацію щодо кожного кандидата, співробітника, відділу чи підрозділу в рамках однієї системи та ще такої, яка може працювати з використанням *інтернет*-технологій.

Сучасні технології змінили ситуацію в управлінні *HR*-процесами, надаючи зручне програмне забезпечення для *HR*-менеджерів та співробітників компаній.

У пересічній сучасній *HRM*-системі є всі необхідні інструменти для управління співробітниками підприємства (компанії), а саме:

Управління персоналом. Сюди входять наповнення бази даних співробітників для швидкого ознайомлення з їх профілем, базова комунікація у вигляді оголошень і новин, календар із важливими подіями команди, а також можливість залишати запити,

наприклад популярні запити на відсутність для економії часу *HR*-фахівців.

Управління рекрутингом. *Рекрутинг* – це процес виявлення, залучення, відбору, співбесіди, найму й адаптації кандидатів до компанії. Як відомо, у процес рекрутингу входить кілька субпроцесів, від аналізу вакансій до найму й адаптації нових фахівців. *HRM*-система дає можливість розвантажити зайнятість рекрутерів переймаючи на себе всю тяганину з призначенням інтерв'ю, розсилкою листів, розсилкою вакансій і збором відгуків. Сюди ж додаються вакансії, за якими рекрутери можуть просто орієнтуватися, та багато іншого.

Онбординг та офбординг. *Онбординг* – це процес адаптації співробітника на новому робочому місці, *офбординг* – це процес, який відбувається, коли співробітник залишає компанію (звільнюється). Процеси, які не можна переоцінити, тому що вони відіграють ключові ролі в побудові бренду роботодавця. Співробітники, які приходять на нове місце роботи, очікують, що компанія швидко та якісно введе їх у курс справи. Аналогічно і під час звільнення – компанія має докласти зусиль, щоб розійтися зі співробітником на позитивній ноті. За допомогою *HRM*-системи обидва процеси можна автоматизувати та провести ефективніше, з користю для обох сторін.

Оцінка персоналу. За допомогою оцінки компанія бачить, наскільки ефективними є співробітники на своїх місцях і чи потрібне втручання, якщо хтось із них не виявляє свій талант і потенціал повноцінно. Найчастіше *HRM*-системи пропонують оцінку за поширеною та простою у підрахунку та розумінні методикою *Performance Review*.

Performance Review (аналіз роботи) – це формальна оцінка, під час якої керівник оцінює результати роботи працівника. Її також називають оцінкою ефективності або оцінкою співробітників. Вона може бути структурована різними способами, щоб ефективно визначити сильні та слабкі сторони працівника, запропонувати конструктивний зворотний зв'язок і встановити цілі на майбутнє.

Отримання фідбеку й оцінка залученості / лояльності. *Фідбек* – це відгук, критичний коментар до чогось. *Фідбек* від співробітників може бути автоматизований як за допомогою опитувань, так і за допомогою індивідуальних зустрічей тет-а-тет. Чим частіше власник (керівник) занурюється у спілкування зі співробітниками або здійснює оцінку ситуації на основі анонімних опитувань, тим краще він розуміє потреби своєї команди. Сучасні системи пропонують опитування для виявлення показника *eNPS* та інших маркерів залучення до процесу роботи.

eNPS (Employee Net Promoter Score) – це метод оцінки лояльності співробітників, що виражається індексом задоволеності персоналу.

Аналітика та звітність. Чим більше показників може автоматизувати та самостійно підраховувати *HRM*-система, тим краще для *HR*-фахівця.

Інтеграція з популярними сервісами. *HRM*-система може інтегруватися з месенджерами, програмами для таск-менеджменту, продуктами *Google* та *Microsoft, job*-сайтами для швидкого обміну даними та зручності щоденного ведення справ.

Розглянемо низку *HRM*-систем, що використовуються для управління персоналом на території України та у світі. Серед них *Hurma System, PeopleForce, BambooHR, Zoho People, Quinyx WorkForce, Orange HRM*.

Hurma System – хмарний сервіс, що автоматизує всі *HR*-процеси, рекрутинг та управління цілями.

Система звільняє *HR*-фахівців від рутини, оптимізує залучення талантів, адаптацію та утримання співробітників, збирає статистику й аналітику. Допомагає керувати цілями та ключовими результатами компанії.

HRM платформа *Hurma* автоматизує нарахування відпусток, лікарняних та інших видів відсутності. Кожен співробітник бачить цю інформацію в особистому кабінеті та може створити запит на відсутність. *HR*-менеджер моментально отримує сповіщення й

опрацьовує інформацію. За допомогою штучного інтелекту *Hurma* вміє прогнозувати ризик звільнення працівників.

Система дає змогу створювати ланцюжки завдань для управління персоналом і керувати ними в напівавтоматичному режимі.

PeopleForce – *HRM*-система для управління ресурсами, часом, ефективністю та культурою компанії. Дає можливість вирішувати всі *HR* завдання: рекрутинг, контроль ефективності співробітників, контроль залучення та задоволеності, обробка звернень від співробітників, тайм-трекінг.

PeopleForce пропонує *HRM*-рішення для автоматизації *HR* та рекрутинг-процесів: найму, обліку робочого часу, управління продуктивністю, оцінки залучення співробітників та інших завдань.

Система забезпечує інтеграцію з інструментами підвищення продуктивності: *Google*, *Slack*, *Zoom* та *Teams*.

База даних співробітників у *PeopleForce* є центром, де зберігається й обробляється вся інформація про них. Контактні й особисті дані, оплачуваний і неоплачуваний баланс відпусток, історія кар'єрного росту, відпрацьований час, історія компенсацій та багато іншого.

PeopleForce використовують клієнти із 14 країн для управління понад 60 000 співробітниками. Серед них такі відомі бренди, як *BMW, Rakuten, Ajax Systems, Parimatch, SkyUp*.

PeopleForce використовується також для малого та середнього бізнесу, складається із 6 модулів: *PeopleHR, PeopleRecruit, PeopleTime, PeoplePerform, PeoplePulse, PeopleDesk*, які зручно налаштовуються під потреби компанії залежно від завдань.

BambooHR найпопулярніша система *HRM* для управління персоналом у світі. Це хмарне програмне рішення для управління персоналом для малого та середнього бізнесу.

BambooHR пропонує невеликим і зростаючим компаніям інформаційну систему для роботи з персоналом, яка виконує такі функції, як відстеження кандидатів, облік робочого часу, розрахунок заробітної плати, адміністрування пільг, залучення співробітників,

автоматичне нагадування й аналітика даних персоналу для управління всіма аспектами життєвого циклу працівника.

За допомогою *BambooHR* будь-яка *HR*-команда або відділ кадрів може покращити процес найму, залучити нових співробітників, керувати компенсацією, підтримувати й аналізувати дані про співробітників і розвивати корпоративну культуру.

Використовуючи інструменти для адаптації, система оптимізує весь процес набору: від подання заявок до планування першого дня.

Система також надає послуги налаштування, кілька ролей адміністратора, підтримку користувачів і керування неактивними співробітниками.

Під час ціноутворення в *BambooHR* використовується модель на основі підписки, яка залежить від кількості співробітників плюс разова плата за впровадження.

Zoho People – це *online*-система, що містить базу даних із повною інформацією про співробітників компанії. Дає змогу створити реєстр посад, складати ієрархічну структуру компанії, вести облік зарплат, премій, витрат на відрядження, планувати процеси прийому на роботу нових співробітників, вести базу вакансій і резюме.

Zoho People є потужною *HRM*-системою для *HR*-відділів у компаніях з понад 50 співробітниками. Під час найму нового співробітника сервіс дає можливість створити чек-лист (які процедури потрібно зробити – організувати робоче місце, купити комп’ютер тощо).

Крім того, *Zoho People* містить модуль Self-service для співробітників, у якому вони можуть отримувати інформацію про компанію та її новини. Цією інформацією керує *HR*-відділ.

Zoho People підтримує інтеграцію з іншими програмами ,забезпечуючи безперебійний обмін даними та підвищуючи загальну продуктивність. Мінімізація ручної роботи зменшує ймовірність помилок та упущень.

Quinyx WorkForce – система управління персоналом і розрахунку заробітної плати. Має повний функціонал системи

управління людським капіталом. Використовує сучасні *HR*-технології та високий рівень кастомізації.

Quinyx WorkForce слугує для автоматизації та уніфікації процесів кадрового обліку, підбору, оцінки, розвитку та мотивації персоналу. Ключовою можливістю системи є підтримка одночасної роботи користувачів в усіх географічно віддалених філіях компанії з актуальними даними щодо *HR*-процесів у режимі *online* в рамках єдиної методології. У системі впроваджено уніфіковану технологію автоматизації процесів управління персоналом, що дає змогу значно зменшувати трудовитрати на консолідацію та облік даних.

OrangeHRM – безплатна open-source система для управління персоналом із модулем для рекрутингу. *OrangeHRM* – перша у світі система управління персоналом із відкритим вихідним кодом. Система працює як під *Linux*, так і під *Windows,* має можливість використання як *онлайн*-сервіс.

OrangeHRM – модульне програмне рішення. Серед основних можливостей системи – адміністрування системи та рольове розмежування доступу, управління інформацією про персонал, управління відпустками та відгулами, облік робочого часу, управління продуктивністю й ефективністю, підтримка рекрутингу, облік відряджень і ділових поїздок, атестація персоналу, управління соціальним пакетом навчання персоналу.

8.3. Системи планування ресурсів підприємств ERP

ERP (Enterprise Resource Planning) – система, яка реалізує стратегію інтеграції виробництва й операцій управління трудовими ресурсами, фінансового менеджменту та управління активами, орієнтована на безперервне балансування й оптимізацію ресурсів підприємства за допомогою спеціалізованого інтегрованого пакета прикладного програмного забезпечення.

Система планування ресурсів підприємства допомагає підприємствам автоматизувати основні бізнес-процеси та керувати ними для досягнення оптимальної продуктивності.

ERP-система координує потік даних між корпоративними бізнес-процесами, надає єдине джерело достовірних даних та оптимізує бізнес-процеси по всьому підприємству. Вона поєднує фінанси, ланцюжки поставок, бізнес-процеси, комерцію, звітність, виробництво й управління персоналом на єдиній платформі.

У більшості компаній уже впроваджено системи для управління фінансами, проте ізольовані системи обмежені поточними операціями та не здатні допомогти з подальшим розвитком.

Коли компанії розширюються, їх потреби змінюються, тому системи управління мають розвиватися разом із ними, через це є актуальним мати програмне забезпечення, здатне зробити діяльність підприємства більш гнучким та ефективним.

Раніше *ERP*-системи були програмними пакетами, які працювали окремо та не обмінювалися даними з іншими системами. Кожну систему доводилося дорого, складно й унікально доопрацьовувати, щоб задовольнити вимоги конкретної компанії, що уповільнювало чи зовсім стримувало впровадження нових технологій та оптимізацію процесів.

Відмінність сучасних *ERP*-систем у тому, що вони об’єднують усі процеси в одну гнучку систему. Вони роблять дані доступними не тільки в *ERP*-системі, але й в офісних додатках, *інтернет*-магазинах і навіть у рішеннях для взаємодії з клієнтами. Усі дані зводяться докупи, щоб поліпшити аналітику й допомогти користувачу оптимізувати процеси по всій компанії.

Крім того, сучасні *ERP*-системи пропонують гнучкі варіанти розгортання, підвищену безпеку та конфіденційність, сталий розвиток і налаштування з мінімумом програмування. Також, що найважливіше, вони забезпечують стійкість і безперервність бізнес-процесів завдяки аналітиці, яка допомагає швидше впроваджувати інновації.

Оскільки не існує універсального програмного забезпечення для всіх бізнес-процесів, технології *ERP* мають за мету об’єднувати їх. Звівши разом процеси, системи та дані, користувач отримує

аналітику, прискорення й адаптивність, необхідні для початку оптимізації бізнес-процесів.

ERP здатна підвищити ефективність компанії в трьох напрямах.

1. **Забезпечує оптимальну результативність**. Рішення на основі штучного інтелекту надають аналітику, яка допомагає приймати більш виважені рішення та підказує напрями для подальшого підвищення операційної ефективності.

2. **Пришвидшує операційні рішення**. Об'єднавши процеси та дані, користувач підвищує їх видимість і гнучкість для співробітників, допомагаючи їм швидше вживати заходів і досягати більших результатів.

3. **Забезпечує гнучкість бізнесу**. Багато *ERP*-функцій адаптуються та масштабуються відповідно до потреб компанії. Це допомагає завчасно підготуватися до будь-яких перебоїв у бізнес-процесах чи змін на ринку, а також оперативно реагувати на них.

ERP-система допомагає усувати бар'єри між адміністрацією, виробництвом і відділами обслуговування клієнтів. При цьому вона здатна адаптуватися до нових бізнес-пріоритетів.

ERP-системи використовують штучний інтелект. Використання штучного інтелекту дає можливість ефективно оптимізувати рутинні завдання, знизити операційні витрати й усунути помилки, пов'язані з людським фактором. Штучний інтелект вивчає та оптимізує процеси підприємства, тоді як основне програмне забезпечення системи *ERP* надає необхідну кількість структурованих даних.

Наведемо ключові відділи компанії, які охоплює *ERP*-система.

Торгівля. Торговельні мережі стикаються з багатьма викликами. *ERP*-система надає повноцінне рішення для багатоканальної торгівлі, що поєднує бек-офіс, звичайний і *інтернет*-магазин. Рекомендації від штучного інтелекту роблять процес купівлі більш персоналізованим і зручним для клієнтів, а торговельні мережі зможуть підвищити продуктивність співробітників.

Фінанси. Сучасна *ERP*-система допомагає підвищити прибутковість і покращити дотримання нормативних вимог. Панелі моніторингу й аналітика на базі штучного інтелекту надають користувачу фінансове зведення та інші дані в реальному часі з будь-якого пристрою. Система допомагає мінімізувати введення даних вручну, автоматизуючи щоденні завдання та відстежувати зміни для дотримання нормативних вимог.

Управління персоналом. Сучасні рішення допомагають керувати корпоративними даними й оптимізують кадрові завдання, наприклад нарахування зарплати, наймання тощо. Власник зможе ефективніше утримувати, наймати та розширювати можливості співробітників, а також відстежувати їхню результативність і заздалегідь виявляти проблеми з персоналом.

Виробництво. *ERP* полегшує ділове спілкування, автоматизує щоденні процеси за допомогою автоматизації процесів, а також надає дані в реальному часі, щоб допомагати виробникам задовольняти запити клієнтів і керувати ресурсами. Крім того, це рішення оптимізує управління проєктами та витратами, а також планування виробництва.

Ланцюжок поставок. Якщо співробітники компанії досі вводять дані вручну та не можуть точно назвати обсяг складських запасів, власник зможе заощадити час і гроші, автоматизувавши ці процеси за допомогою *ERP*. Сучасні рішення для ланцюжків постачання також спрощують управління запасами завдяки панелям моніторингу та бізнес-аналітики.

Модульний принцип організації дає змогу впроваджувати *ERP*-системи поетапно, послідовно перетворюючи на експлуатацію один чи кілька функціональних модулів, вибравши актуальні. Крім того, модульність *ERP*-систем дає можливість будувати рішення на основі декількох *ERP*-систем, вибираючи з кожної найкращі модулі у своєму класі. У більшості основних постачальників виділяються групи модулів: *фінанси, персонал, операції*.

Модуль «Фінанси»

Фінансові модулі вважаються центральними компонентами *ERP*-системи, а формування фінансової звітності засобами *ERP*-системи вважається однією з фактично обов'язкових умов позитивних результатів процедури *Due Diligence*.

Due Diligence – це процедура незалежної перевірки компанії, складання об'єктивного уявлення про об'єкт інвестування, що передбачає оцінку інвестиційних ризиків, незалежну оцінку об'єкта інвестування, всебічне дослідження діяльності компанії, комплексну перевірку її фінансового стану й положення на ринку.

Серед фінансових модулів *ERP* фігурує безліч різних функціональних блоків. У різних системах і різних версіях виділяються різні їх компонування, серед найпоширеніших (за організаційними підрозділами):

– **бухгалтерські**: рахунки до одержання (дебітори), рахунки до оплати (кредитори), консолідація;

– **обліково-управлінські**: облік витрат і доходів за місцями виникнення, за продуктами, проєктами, калькуляція собівартості;

– **казначейські**: управління ліквідністю, управління рухом коштів (включаючи банківські рахунки та касу), взаємодія з банками, управління боргом і запозиченнями;

– **фінансово-управлінські**: управління основними коштами, інвестиційний менеджмент, фінансовий контроль та управління ризиками.

Також іноді до складу фінансових модулів *ERP*-систем включено фінансове планування й управління ключовими показниками ефективності, але основні розробники постачають для цих функцій окремі спеціалізовані програмні продукти.

Модуль «Персонал»

Однією з принципових відмінностей *ERP* як стратегії від використання окремих додатків для автоматизації розрахунку зарплати є уявлення про тісну інтеграцію інформації про трудові ресурси для можливості оперативного планування й управління операціями з урахуванням інформації про доступність персоналу,

можливості точно розраховувати витрати за місцями виникнення відповідно до інформації про компенсацію задіяного персоналу.

Серед модулів управління персоналом в *ERP*-системах використовують: кадровий облік, облік робочого часу (табельний облік), управління нарядами на роботи, управління відрядженнями, розрахунок продуктивності трудових ресурсів, управління оплатою праці, преміями, компенсаціями та розрахунок заробітної плати, пенсійний облік, оцінка персоналу, управління кваліфікацією (професійними навичками, навчанням), підбір персоналу.

Модуль «Операції»

Модулі операційного блоку покривають діяльність організації зі створення товарів та послуг і необхідні функції із забезпечення цих процесів. Якщо кадрові та фінансові модулі досить універсальні для різних організацій, то багато операційних модулів більш специфічні для різних галузей, оскільки підходи до перетворення ресурсів у різних галузях суттєво відрізняються.

У більшості систем сформувалися такі групи операційних модулів: *логістичні*, *виробничі*, *забезпечувальні*, *збутові*.

Логістичні: постачання, управління взаємовідносинами з постачальниками, управління ланцюжками поставок та транспортуванням, управління запасами, складами, інвентаризацією.

Виробничі: управління специфікаціями та рецептурами (у процесних виробництвах – хімічних, металургійних, харчових та інших), виробниче планування, облік продукції, управління виробничими програмами.

Забезпечувальні: керування технічним обслуговуванням і ремонтами обладнання, планування потужностей, керування транспортом.

Збутові: ціноутворення, обробка та конфігурування замовлень, продаж, післяпродажне обслуговування.

Окремі функції операційного блоку найчастіше виносяться у спеціалізовані програмні продукти й фігурують як виділені класи прикладного програмного забезпечення. Такими є *EAM* для

технічного обслуговування та ремонтів, *CRM* для продажу та дистрибуції, *PLM* для управління специфікаціями, *APS* і *MES* для управління виробництвом.

Серед *ERP*-систем, що використовуються на території України, знайшли широке застосування *UGLA, Perfectum, BAS, Microsoft Dynamics 365, Oracle ERP*.

UGLA ERP – гнучка та безпечна система для управління компанією, доступна з будь-якої точки світу без додаткового обладнання. У системі завдяки взаємоінтеграції модулів *ERP* та *CRM* налагоджена ефективна комунікація з усіма підрозділами. Є можливість отримання будь-яких показників діяльності компанії. Технологія *UGLA* використовує сервери *Amazon*.

Perfectum – *ERP*-система, призначена для автоматизації планування ресурсів і процесів підприємств, включно з управлінням фінансами, кадрами, складами, замовленнями, проєктами, закупками, продажами.

BAS – *ERP*-система від компанії 1С для України. Призначена для автоматизації великих підприємств зі складними технологічними процесами, позиціонується як система, що здатна забезпечити потреби будь-яких масштабів, починаючи від 50 робочих місць. У системі використовуються мобільні та хмарні технології.

Microsoft Dynamics 365 – *ERP*-система зі звичним інтерфейсом від *«Майкрософт»*. Повна інтеграція з платформою та програмами *MS*. Містить інструменти для управління продажами, маркетингом, сервісом і бізнес-процесами. Можлива робота із системою просто з *Outlook*.

Oracle ERP – інтегрований хмарний комплекс додатків для управління фінансами, закупівлями та портфелем проєктів для малого, середнього та великого бізнесу.

Контрольні запитання

1. Що таке CRM-система та для чого вона використовується?

2. На які види поділяються CRM-системи залежно від своїх можливостей?

3. Які ви знаєте сучасні CRM-системи, що використовуються на території України?

4. Що таке HRM-система та для чого вона використовується?

5. Назвіть основні задачі, які вирішуються HRM-системою?

6. Які функціональні блоки містять сучасні HRM-системи?

7. Які ви знаєте сучасні HRM-системи, що використовуються на території України?

8. Що таке ERP-система та для чого вона використовується?

9. З яких модулів складається ERP-система?

10. Які ви знаєте сучасні ERP-системи, що використовуються на території України?

Лекція 9
РЕКЛАМА В МЕРЕЖІ INTERNET

Реклама в мережі Інтернет *(Internet advertising)* – це комплекс інструментів, які компанії використовують для просування бренду й підвищення продажів.

У процесі розвитку *web*-технологій з початку 90-х років реклама в мережі *Інтернет* стала віртуальним еквівалентом традиційних методів маркетингу, як-от *ролики на телебаченні, радіо, оголошення в газетах, білборди* тощо.

У наш час найпопулярнішими видами реклами в мережі *Інтернет* стали: *пошуковий маркетинг, email-маркетинг, реклама в соціальних мережах, медійна реклама, нативна реклама*.

Реклама діяльності чи послуг в мережі *Інтернет* має низку переваг, а саме:

- ✓ масштабність,
- ✓ бюджетність,
- ✓ збільшення трафіку для сайту,
- ✓ можливість таргетингу,
- ✓ використання ретаргетингових кампаній,
- ✓ налагодження точок взаємодії з аудиторією,
- ✓ ведення статистики.

Масштабність. Люди шукають інформацію про продукти та послуги за допомогою пошукових систем, наприклад *Google, Yandex, Yahoo* та ін.

Отже, через це реклама в мережі *Інтернет* – це спосіб розповісти про бренд понад 4 мільярдам користувачів по всьому світу. Завдяки *Інтернету* можна легко охопити цільову аудиторію в різних країнах світу.

Бюджетність. Будь-яка компанія, від невеликого сімейного бізнесу до величезного підприємства, може використовувати онлайн-рекламу й отримувати максимальний результат від фінансових вкладень, оскільки такі види реклами, як *пошуковий маркетинг* та *email-маркетинг*, взагалі не потребують інвестицій на початковому етапі.

Збільшення трафіку для сайту. Чим більше відвідувачів компанія залучає, тим більше клієнтів вона отримає і тим вищий буде рівень продажів.

Мета реклами в мережі Інтернет – привернути увагу користувачів і привести їх на свій сайт. Тому оголошення повинні викликати інтерес і надавати людям вагомі підстави для переходу.

Можливість таргетингу. На відміну від традиційної реклами, яку можуть бачити всі без винятку, оголошення в мережі Інтернет можна таргетувати.

Використання ретаргетингових кампаній. Якщо покупці відвідують певний магазин, відкривають картки товарів і нічого не купують, слід нагадати їм про свій бренд за допомогою рекламних банерів, які вони побачать під час перегляду сайтів.

Налагодження точок взаємодії з аудиторією. *Реклама в мережі Інтернет* дає змогу компаніям з'являтися в потрібний час у потрібному місці.

Щоб продемонструвати свої продукти, використовуються різні *соціальні мережі*. Для розповсюдження новин і побудови довгострокових взаємин з клієнтами – *email-маркетинг*.

Слід об'єднувати різні види *онлайн*-реклами, щоб показати, що компанія завжди поруч і готова допомогти.

Ведення статистики. На відміну від *офлайн*-маркетингу, де ефективність виміряти досить важко, *реклама в мережі Інтернет* дає змогу точно відстежувати результати за допомогою платформ *web*-аналітики, наприклад *Google Analytics*.

На сьогодні не виникає жодних сумнівів про те, що *реклама в мережі Інтернет* – це швидкий і гнучкий спосіб доставки *промо*повідомлень людям по всьому світу, оскільки сучасна тенденція така, що люди, починаючи з *міленіалів,* звикли спілкуватися *онлайн*.

9.1. Види інтернет-реклами

Ефективність реклами в мережі *Інтернет* у сучасному світі важко переоцінити. Практично кожен із нас щодня бачить її на своїх

гаджетах і навіть не замислюється над тим, який вплив вона на нас здійснює.

Зважаючи на величезну кількість користувачів *World Wide Web, онлайн*-реклама вже не є додатковою можливістю розвитку бізнесу, а слугує повноцінною необхідністю сучасних реалій.

Мережа *Інтернет* відкриває багато можливостей для взаємодії з потенційними клієнтами. Завдяки цьому в наш час існує багато видів *інтернет*-реклами, серед яких:

- пошуковий маркетинг,
- *email*-маркетинг,
- реклама в соціальних мережах,
- банерна реклама,
- нативна реклама,
- відеореклама,
- *push*-повідомлення,
- мобільна реклама.

Пошуковий маркетинг

Пошуковий маркетинг – це просування *web*-сайту для залучення на нього людей, які цікавляться конкретними продуктами чи послугами.

Основною метою використання інструментів *інтернет*-маркетингу є залучення покупця на сайт, який виступає як рекламний майданчик з просування товарів і послуг на ринку.

Зазвичай пошук будь-якого товару в мережі *Інтернет* починається із запиту в пошуковій системі.

Рядок пошуку є відправною точкою шляху покупця. Після введення ключового слова користувачі здебільшого переглядають результати першої сторінки видачі.

Пошуковий маркетинг спрямований на те, щоб сайт компанії потрапив на верхні позиції в пошуковій видачі як органічним, так і платним способом.

Google AdWords відображає результати, що ґрунтуються на рейтингу оголошень в аукціоні. Компанії називають свою ціну для

певного ключового слова, тоді як *Google* аналізує якість і релевантність оголошення.

Формула розрахунку рейтингу оголошення:

$$R = P_{\max} \times i_{aq},$$

де R – рейтинг оголошення, $P_{\max}$ – максимальна ціна за клік, i_{aq} – показник якості рекламного оголошення.

Email-маркетинг

Email-маркетинг – це маркетинговий інструмент, який дозволяє комунікувати з аудиторією за допомогою *email*-листів. Він є одним із найстаріших каналів на ринку. Останнім нововведенням стала поява інтерактивних *AMP-листів*.

AMP-листи *(Accelerated Mobile Pages)* – технологія прискорених мобільних сторінок із відкритим вихідним кодом. Дає змогу за низької швидкості мережі виконати швидке завантаження *web*-сторінок. Технологія створена *Google* у 2015 році для економії часу користувачів.

Email-маркетинг – найменш нав'язливий канал реклами, оскільки клієнти добровільно підписуються на розсилку, щоб отримувати інформаційні листи і промоматеріали, від яких можуть відписатися в будь-який момент. Основна мета такої взаємодії полягає в тому, щоб приводити на сайт нові ліди, які вирощують шляхом чергування рекламного й некомерційного контенту.

У світі налічується 4 мільярди користувачів електронної пошти. Тому, якщо є бажання встановити контакт зі своїми клієнтами, електронна пошта стає ідеальним способом зв'язку для розповсюдження рекламних *email*-розсилок.

Для створення списків *email*-розсилок використовуються так звані *лід-магніти*. *Лід-магніт* зазвичай створюється у вигляді безкоштовної пропозиції, яка може мати різні формати, в обмін на надання адреси електронної пошти.

Реклама в соціальних мережах

Реклама в соціальних мережах – це просування товарів і послуг свого бренду засобами мереж для спілкування.

Компанії використовують два способи, щоб налаштувати свої новини та промоматеріали під цільову аудиторію:

Органічний. Публікується такий контент, як *лайфхак, рецепти* або цікаві способи використання товарів, для підвищення залученості.

Платний. У цьому випадку використовують бізнес-функціонал соціальних мереж, налаштовуючи показ рекламних оголошень на підставі віку, статі, улюблених занять та інших спільних характеристик потенційних клієнтів.

Реклама в соціальних мережах дає змогу:

✓ побудувати охоплення поміж цільової аудиторії і підвищити популярність бренду;

✓ збільшити кількість підписників, підвищити залученість для публікацій;

✓ залучити цільову аудиторію на *web*-сайт або в додаток;

✓ зібрати контактну інформацію користувачів для подальшої взаємодії з ними.

Банерна реклама

Банерна реклама – це інструмент, який використовують для залучення користувачів, що використовується на цільових ресурсах. Вона є картинкою у графічному вигляді, основною метою якої є просування певної сфери обслуговування або товару. Це один із найстаріших видів реклами в мережі *Інтернет*, який ще називають *медійним*. Як і інші типи реклами, банер відсилає користувача на конкретний сайт, підвищуючи таким чином популярність рекламованого контенту.

40 % людей уникають спливаючих вікон, банерів, *flash*-реклами та інших оголошень за допомогою спеціальних розширень. До того ж багато користувачів, які не використовують блокування, підсвідомо ігнорують повідомлення через психологічний феномен, який називається «банерна сліпота».

Із середини 1990-х років банерна реклама отримала репутацію нав’язливого засобу масової інформації, але не у випадку, коли вона містить високо релевантні пропозиції для користувача.

Власники сайтів із великим обсягом трафіку хочуть монетизувати його, тому продають частину свого *інтернет*-простору за допомогою *Google AdSense*, а рекламодавці купують його через *Google Ads*. При цьому *Google* показує релевантні оголошення на основі двох цінових підходів: *ціна за клік* і *ціна за тисячу показів*, а також дає можливість брендам налаштовувати ремаркетингові кампанії.

Нативна реклама

Нативна реклама – спосіб, яким рекламодавець привертає до себе увагу в контексті якогось сайту й інтересів користувача. В оригіналі вона сприймається як частина сайту, що переглядається, враховує особливості майданчика, не ідентифікується як реклама й не викликає в аудиторії відторгнення.

Нативна реклама «вплітається» в основний потік інформаційного вмісту сайту та відповідає йому за форматом і змістом, на відміну від традиційних рекламних розміщень, щодо яких велика ймовірність «банерної сліпоти».

Мета *нативної реклами* – створення менш нав'язливого рекламного звернення для збільшення числа кліків, як наслідок – продажу та інших цільових дій.

По суті, *нативна реклама* ненав'язливо «чекає» користувача там, де він сам шукає матеріал, що його цікавить, і так само відповідає на його питання, як і неспонсоровані повідомлення.

У певному сенсі це альтернатива банерній рекламі. Компанії платять таким популярним ресурсам, як *AdMe, ЛігаБізнесІнформ, Fishki.net* тощо, за розміщення *промо*матеріалів у своїх публікаціях. Поки це відбувається в розважальній манері, читачі не усвідомлюють, що їм щось рекламують, тому цей підхід називається «нативним».

Відеореклама

Відеореклама – форма реклами, що зазвичай розміщується в *Інтернеті* і спрямована на створення іміджу компанії, просунення послуг чи товарів, представлення інформації з метою підвищення продажу.

Цей вид реклами в мережі *Інтернет* дуже швидко набирає популярності. Рекламу створюють у відеоформаті й розміщують на таких сервісах, як *YouTube, Vimeo, DailyMotion* і *Vine*. Відеоролики коштують достатньо дорого, проте їх ефективність може бути дуже висока, оскільки кращі з них швидко стають впізнаваними.

Відмінною рисою *відеореклами* в мережі *Інтернет* є взаємодія між компанією, що розміщує рекламу, і аудиторією, що дивиться рекламний відеоролик.

Оскільки *відеореклама* часто розміщується на популярних відеохостингах, аудиторія отримує можливість залишати відгуки й формувати свій власний рейтинг відео.

Представники компанії отримують інформацію про кількість переглядів запису, географічну статистику – місце проживання глядачів, статистику переглядів по днях. Ці дані дають змогу оцінювати ефективність рекламної діяльності, давати прогнози про конверсії. Також популярність відеореклами в мережі *Інтернет* зумовлена відносно низькими цінами на розміщення того чи іншого ролика.

Push-повідомлення

Push-повідомлення – це коротке спливаюче повідомлення в додатках або браузерах. Його відправляють користувачам, щоб розповісти про оновлення, новини й акції. Головна ціль *push-повідомлень* – доставити клієнту релевантну інформацію для підтримання залученості.

Ця технологія дає змогу привертати увагу користувачів, коли вони онлайн. Повідомлення з’являються в правому нижньому кутку екрані й після натискання на них користувачів перенаправляють на певну сторінку.

Мобільні push-повідомлення доступні для користувачів після встановлення мобільних додатків. Мобільні повідомлення (*in-app* повідомлення) допомагають вчасно інформувати про оновлення, спрямовувати підписників в потрібні розділи додатку й надавати короткі інструкції. Основними платформами для отримання *push-повідомлень* у додатках є *Android* і *iOS*.

Web push-повідомлення – це браузерні повідомлення, які з'являються на екрані комп'ютера або мобільного телефона у браузері. Компанії використовують *web push-повідомлення* переважно в маркетингових цілях. Вони відправляють інформацію про акції та знижки, повідомляють про нові товари на складах або діляться корисним контентом.

Мобільна реклама

Мобільна реклама – інформування споживачів про продукт чи послугу та заохочення до їх покупки через мобільні пристрої (смартфони, планшети). Мобільна реклама є частиною ширшого поняття – *мобільний маркетинг.*

Мобільна реклама – це маркетинговий інструмент, який відображає оголошення на смартфонах і планшетах через платні канали. Це завжди симбіоз технічних інструментів та креативних рішень. Лише в поєднанні вони дають можливість брендам досягати маркетингових цілей. У процесі створення рекламної кампанії потрібно визначитися з метою, вибрати формати, не забуваючи про комфорт аудиторії, і відстежувати перебіг після запуску.

Мобільна реклама на сьогодні це здебільшого графічні банери та посилання на сайтах в мобільному *інтернеті*, тобто на сайтах, що переглядаються з мобільного телефона. Також до мобільної реклами належить реклама, що розповсюджується через *SMS* розсилки.

Компанії можуть рекламувати себе за допомогою *SMS* після того, як користувач підпишеться, або за допомогою медійної реклами в браузері, оптимізованої під мобільні пристрої.

9.2. Вартість реклами

Пошукова реклама. Середня ціна за клік у всіх галузях становить $2,32. Вартість залежить від кількості пошукових запитів на місяць, рівня конкуренції, тобто числа компаній, які використовують це ключове слово, і розміру ставки на аукціоні оголошень.

Email-маркетинг. Вартість такої реклами залежить від кількості підписників. Ціна за 2000 підписників починається від $20, за 5000 – від $30, за 10 000 – від $40.

Реклама в соціальних мережах. Залежно від галузі середня ціна за *клік* у *Facebook* становить $1,86, а ціна за тисячу показів – $11,20. В *Instagram* ціна за *клік* становить $0,50 – $1,00. У більш конкурентних галузях, як-от юриспруденція, вартість реклами досягає $3,00 за клік.

Медійна реклама. Ціна залежить від розміру оголошення, місця його розміщення, тривалості показу на сторінці й середньої кількості відвідувачів. Для такої реклами можна вибирати *Google Ads*. Вартість медійної реклами в *Google* (оплата за роботи агенції) починається від $230 на місяць, або середня ціна за *клік* – $0,09.

Нативна реклама. Основною причиною використання нативної реклами є великий обсяг трафіку, який компанія, що розміщує рекламу, отримує на свій сайт. Така реклама потребує від рекламодавця високоякісного контенту, а отже, професійного копірайтингу, редагування та дизайну. Вартість нативної реклами починається від $160 на місяць.

Відеореклама. Компанії платять *YouTube* за кожен перегляд. Зазвичай відеооголошення коштують від $0,10 до $0,30 за перегляд. Це означає, що за 10 000 переглядів оголошення треба буде сплатити $1000. Іншим важливим фактором є створення відео. Вартість тут залежить від того, скільки людей працює над відео і скільки коштують їхні послуги.

Web push. За допомогою системи *SendPulse* можна використовувати *Web push* для швидкої розсилки рекламних повідомлень абсолютно безплатно, якщо клієнт цієї системи має ~10 000 підписників. А взагалі вартість такої реклами суттєво залежить від країни розповсюдження (наприклад, для України становить $0,04 за клік).

Мобільна реклама. Вартість такого виду реклами становить від $0,1 до $1 за клік, вона залежить від багатьох параметрів.

9.3. Контекстна реклама

Контекстна реклама – це ефективний інструмент залучення нових клієнтів і повернення старих. *Контекстну рекламу* можна використовувати на будь-якому етапі просування в *Інтернеті*. Ідеально підходить як для невеликого локального бізнесу, націленого на вирішення проблем клієнтів тут і зараз, так і для великого *інтернет*-магазину з доставкою в будь-яку точку країни.

Контекстна реклама дає змогу залучити цільову аудиторію на свій сайт чи в *інтернет*-магазин, підвищити впізнаваність бренду чи продукту, продавати товари чи послуги в *Інтернеті*, отримувати контактні дані потенційних клієнтів і збільшувати лояльність наявних клієнтів.

Контекстна реклама показується зацікавленим користувачам. Замовник сплачує лише за реальні кліки з реклами, вказавши граничну вартість кліка та бюджет. Налаштування реклами також дає змогу вибирати різні параметри показів. Завдяки детальній аналітиці можна відстежувати та коригувати ефективність у реальному часі.

Результати роботи *контекстної реклами* найчастіше видно вже в перші дні після старту. Замовити рекламу в *Google* і отримати стабільний потік запитів від потенційних клієнтів можна відразу після запуску, тому реклама в *Google* є швидким і універсальним інструментом для залучення клієнтів.

Контекстна реклама – тип *інтернет*-реклами, за якого рекламне оголошення відображається відповідно до змісту, вибраного аудиторією, місця, часу або іншого контексту *інтернет*-сторінок.

Контекстна реклама діє вибірково й відображається відвідувачам *інтернет*-сторінки, сфера інтересів яких потенційно збігається з тематикою рекламованого товару чи послуги цільової аудиторії, що підвищує імовірність їхнього відгуку на рекламу.

Пошукові машини в *Інтернеті* широко використовують системи контекстної реклами, що є для них основним джерелом отримання доходу.

Найчастіше контекстну рекламу можна зустріти на сторінках із результатами пошуку, на сайтах, які встановили блоки контекстної реклами на своїх сторінках, у мобільних додатках. За типом контекстну рекламу розділяють на *пошукову* та *тематичну*.

Пошукова реклама демонструється серед результатів пошуку, у тому числі на окремому сайті. Орієнтація тематики здійснюється за характером пошукового запиту.

Тематична реклама демонструється на сторінках сайтів і в мобільних додатках, що входять до рекламної мережі. Рекламні блоки є доповненням до змісту сторінок. Тематика залежить від тематики сайту або орієнтується на раніше виявлений користувачем інтерес.

9.4. Таргетингова реклама

Таргетингова реклама (цільова або вибіркова реклама) – це вид *онлайн*-реклами, у якому використовуються методи й налаштування пошуку цільової аудиторії відповідно до заданих параметрів (характеристики та інтереси) людей, які можуть цікавитися рекламованим товаром або послугою. Така реклама показується лише обраній (цільовій) аудиторії і дає змогу ефективніше використовувати рекламний бюджет компанії.

Існує механізм вибірки – *націлення*, з набором певних параметрів, який використовується для налаштування реклами, що таргетується.

Виділяють такі параметри налаштувань:

- ***демографічні*** (орієнтовані на національність, економічний статус, стать, вік, рівень освіти, рівень доходу та зайнятість);
- ***психографічні*** (засновані на цінностях споживача, його особистості, відносинах, думці, способі життя й інтересах);
- ***орієнтовані на поведінку*** (відображаються в історії браузера, історії покупок та інших дій користувача на сайті);
- ***тимчасові*** (відповідно до годин або тижнів, у які продаж товару чи послуг можливий);

- ***географічні*** (поширюється на людей, що в певний проміжок часу перебувають у певному районі або на тій чи іншій території).

При цьому недостатня вибірка або відносно широкі налаштування можуть охоплювати частину користувачів зі схожими характеристиками, але не зацікавленими в товарі або послузі, що рекламується.

У зв'язку із цим цей вид реклами потребує тестового періоду налаштування кампанії, глибокого аналізу отриманих результатів і подальшої роботи з налаштуваннями.

Існують різні види таргетингової реклами, кожен з яких дає змогу таргетологам знайти в *Інтернеті* необхідну аудиторію відповідно до завдань рекламної кампанії.

Кожен вид таргетингової реклами – це процес, що розвивається й постійно доопрацьовується, заснований на пошуку аудиторії за заданими критеріями та параметрами.

Види таргетингової реклами:

- таргетингова реклама в соціальних мережах,
- мобільний таргетинг,
- контентний таргетинг,
- таргетинг у режимі реального часу,
- контекстно-медійна мережа *Google*.

Таргетингова реклама в соціальних мережах

Цей вид реклами налаштовується в рекламних кабінетах і демонструє рекламні оголошення (текстові, графічні та текстово-графічні блоки) користувачам, які переглядають сторінки соціальних мереж.

Мобільний таргетинг

Мобільна реклама, що таргетує, дає можливість передавати більше інформації про споживача, включно з його місцем перебування та часом.

Мобільна реклама є перспективним напрямом таргетованої реклами, що активно розвивається, оскільки мобільний трафік постійно зростає.

Контентний таргетинг

Один із найпростіших з погляду аналітики методів таргетингу – контентний. Інша назва контентного таргетингу – це *контентно-орієнтована реклама*. Вона може розміщуватися на різних *онлайн*-майданчиках. Рекламна кампанія з таргетингом на контент передбачає механізм зворотного зв'язку з користувачем.

Таргентинг у режимі реального часу

RTB (Real Time Bidding) – технологія в індустрії онлайн-реклами, що являє собою аукціон рекламних оголошень у реальному часі.

Реклама в режимі реального часу – технологія продажу та купівлі показів реклами в *Інтернеті* на основі автоматизованого аукціону, що дає змогу рекламодавцю розміщувати рекламу не на конкретному майданчику, а демонструвати її на різних майданчиках лише спеціально відібраної за напрямом аудиторії.

Необхідна рекламодавцю аудиторія побачить рекламне оголошення в разі, якщо рекламодавець вибере ставку з найбільшою вартістю за принципом аукціону в реальному часі.

Технологія *RTB* допомагає як рекламодавцям провести рекламну кампанію з максимальними результатами, так і *онлайн*-майданчикам із певною аудиторією клієнтів отримати прибуток.

Контекстно-медійна мережа *Google*

Контекстно-медійна мережа *Google Display Network* надаж можливість звертатися до потенційних клієнтів, коли вони переглядають улюблені сайти або відео на *YouTube*, перевіряють пошту в *Gmail* або використовують мобільні сайти та програми. Цей вид реклами представляє адаптивні медійні оголошення й передбачає розміщення на пошукових сторінках *Google* та інших ресурсах, які стосуються *Google*.

Таргетингова реклама застосовує широкі можливості порівняно з іншими видами реклами. Багато в чому ефективність таргетингової реклами пояснюється чисельністю користувачів, які надають про себе досить достовірну інформацію в соціальних

мережах, а також дають можливість використовувати дані про їх покупки та деякі інші дії в *Інтернеті*.

9.5. Реклама в Google Ads

Google Ads пропонує багато можливостей привернути увагу до бізнесу. Серед них:

- ✓ пошукові оголошення,
- ✓ медійні оголошення,
- ✓ товарні оголошення,
- ✓ відеореклама,
- ✓ реклама додатків.

За допомогою пошукових оголошень можна збільшити обсяг продажів, кількість потенційних клієнтів або відвідувачів сайту, рекламуючи свою компанію людям, які шукають у *Google* товари чи послуги із запропонованого асортименту.

Використовуючи медійні оголошення, можна підвищити інформованість про компанію за допомогою привабливих оголошень, які привертають увагу людей, коли вони переглядають сайти, перевіряють пошту в *Gmail* або використовують мобільні додатки.

Товарні оголошення показують покупцям привабливу інформацію про товари, що дає змогу тримати покупців у курсі наявного асортименту.

Відеореклама підвищує інформованість про бренд, показує рекламу користувачам повторно, а також коли вони переглядають або шукають відео на *YouTube*.

Щодня *Google* реєструє мільярди пошукових запитів. Щоб ознайомити потенційних клієнтів зі своїм брендом, зацікавити їх пропозиціями й отримати бажаний результат, варто скористатися оголошеннями саме в пошуковій мережі.

Користувачі постійно використовують пошук *Google*, щоб отримати відповіді на запитання: чим зайнятися, куди піти, що купити, як заробити. Ключові слова допомагають показувати рекламу, коли користувачі шукають пропозиції рекламодавця.

Google Ads використовує ставки під час аукціону, щоб визначити, які саме оголошення показувати. Встановлюючи ставки, які доцільні для компанії, рекламодавцю можна залучати більше потенційних клієнтів.

Завдяки оголошенням у пошуковій мережі можна налаштувати своє повідомлення та демонструвати пропозиції користувачам, що здійснюють пошук у *Google*.

За допомогою медійних оголошень можна рекламувати свій бізнес, коли користувачі шукають потрібний контент в *Інтернеті*, переглядають відео на *YouTube*, перевіряють пошту *Gmail* або користуються мобільними пристроями та додатками.

Медійні рекламні кампанії мають широке охоплення й можуть допомогти досягти поставлених цілей, показуючи рекламу клієнтам, де б вони не перебували.

Сервіс *Google Ads* допомагає продавати товари найважливішим покупцям – тим, хто цікавиться пропозиціями, перебуваючи вдома, у дорозі або в магазині.

За допомогою реклами в *Google Ads* можна просувати товари, які продаються у звичайних або *інтернет*-магазинах, залучати більш зацікавлених потенційних клієнтів і збільшувати трафік до свого *web*-сайту або звичайного магазину.

Під час створення товарних оголошень можна додавати фото свого товару, його назву, ціну, назву магазину, інші параметри.

Завдяки інструментам *Google Ads* і детальним звітам можна дізнатися, що працює ефективно, а що варто покращити.

Реклама в *YouTube* використовує дані *Google Ads* з метою показу оголошень потрібній аудиторії в слушний момент, тим самим перетворюючі глядачів на клієнтів за будь-якого бюджету.

Завдяки рекламі в *YouTube* користувачам буде легше впізнати компанію рекламодавця серед інших.

Тим, хто займається розробкою додатків, за допомогою рекламних кампаній в *Google Ads* можна рекламувати свої розробки і привертати увагу великої кількості людей, які зацікавлені в таких програмних продуктах, у пошуку *Google*, *Google Play*, на *YouTube*.

9.6. Створення рекламної кампанії в Google Ads

За допомогою рекламних кампаній можна рекламувати свої товари та послуги в мережі *Google*, яка складається із чисельних сервісів і ресурсів, як-от сторінки з результатами пошуку, сайти, відео, мобільні програми, карти та відомості про товари.

Тип кампанії слід вибирати, виходячи з мети реклами, стратегії бренду та кількості часу, яку користувач готовий приділити цій кампанії.

Для кожного типу кампанії використовується окремий набір критеріїв націлювання й оголошень. Наприклад, кампанія в пошуковій мережі дає можливість розміщувати текстові оголошення в результатах пошуку, а за допомогою відеокампанії можна показувати відеорекламу на *YouTube*.

Для створення нової рекламної кампанії першим кроком потрібно вибрати рекламну мету та мету конверсії. Далі буде запропоновано вказати цілі конверсії. Це допоможе визначити найбільш підходящий тип кампанії для охоплення потрібної аудиторії.

Під час створення кампанії користувач може отримувати сповіщення відповідно до налаштувань. За їхньою допомогою можна дізнаватися про проблеми, які можуть негативно вплинути на ефективність кампанії або перешкодити її публікації.

У меню навігації можна переглянути етап налаштування кампанії, у разі потреби – повернутися на попередні етапи, щоб усунути потенційні проблеми з націленістю, призначенням ставок, бюджетом та інше.

Другим кроком для створення рекламної кампанії потрібно вибрати її тип. Тип кампанії визначає, де клієнти побачать оголошення рекламодавця. Процес налаштування та рекомендації щодо використання залежать від типу кампанії.

На цьому етапі створення рекламної кампанії відкриється сторінка, де можна встановити налаштування, створити об’яви та групи об’яв.

Нижче зазначені типи кампанії, що можна створити.

Пошукова мережа – текстові оголошення в результатах пошуку.

КММ – графічні оголошення на сайтах.

Відео – відеореклама на *YouTube*.

Торгова – інформація про товари в *Google*.

Discovery – реклама в *онлайн*-фідах.

Додаток – просування програми різними каналами.

Локальна – просування місцевих компаній різними каналами.

Розумна – спрощення кампаній.

Максимальна ефективність – пошук найцінніших клієнтів за всіма каналами.

В обліковому записі можуть з'являтися повідомлення, які допоможуть встановити налаштування так, щоб нічого не заважало показу реклами.

Якщо в користувача ще немає облікового запису *Google Ads*, потрібно його створити на головній сторінці сервісу, причому процес створення складається з 3 етапів:

1. Додавання інформації про компанію.
2. Вибір мети та бюджету кампанії.
3. Додавання відомостей про спосіб оплати.

Додавання інформації про компанію починається із запрошення вказати назву компанії та *URL* її сайту. Це дає змогу системі отримати потрібну інформацію та максимально спростити створення облікового запису. Вказані відомості використовуватимуться на наступних етапах.

Також можна встановити зв'язок з іншими своїми обліковими записами, наприклад з профілем компанії в *Google* або каналом *YouTube*. Це дає змогу системі підібрати для кампанії найбільш підходящі рішення.

Вибір мети та бюджету кампанії. Після додавання інформації про компанію з'являється запрошення вказати мету кампанії. Система автоматично порекомендує тип кампанії, який найкраще підходить для мети.

Додавання відомостей про спосіб оплати. На останньому етапі потрібно вказати свої платіжні дані.

Для створення успішної рекламної кампанії в *Google* потрібно виконати 8 кроків:

1. **Позначити маркетингову мету**. Створення кампанії починається з вибору мети. Вона визначає, який результат отримується завдяки рекламі.

2. **Вибрати тип кампанії**. Коли відбувається вибір мети, з'явиться список типів кампаній, що рекомендуються для неї. Від типу кампанії залежить те, де саме з'являтимуться оголошення та який вигляд вони матимуть.

Кампанії з максимальною ефективністю ґрунтуються на штучному інтелекті від *Google*. Вони дають змогу отримувати більш високі результати, реагувати на зміни в поведінці споживачів у реальному часі та знаходити клієнтів, які здійснюють конверсії, у всіх каналах *Google*, включно з *YouTube*, медійною мережею, пошуковою мережею та рекомендаціями.

3. **Задати бюджет**. Від зазначеного середнього денного бюджету залежать витрати на рекламу та ставки в автоматичних аукціонах оголошень. Змінити бюджет можна будь-якої миті.

4. **Вибрати спосіб призначення ставок**. Якщо під час налаштування кампанії вибрати мету, буде запропоновано стратегію призначення ставок з урахуванням цієї мети. Завдяки цьому кампанія зможе ефективно досягати потрібних результатів. Для деяких типів кампаній замість способу призначення ставок можна вибрати одну зі стратегій автоматичного призначення ставок.

5. **Додати об'єкти до оголошення.** У кампаніях типу «Пошукова мережа», «Відео», «Створення попиту» та «Максимальна ефективність» можна додавати до оголошення додаткову інформацію, наприклад посилання на сайт, маршрут або номер телефону.

6. **Створити групи оголошень.** У всіх кампаніях, окрім торгових і кампаній із максимальною ефективністю, можна

об’єднувати схожі оголошення в групи та задавати налаштування націлювання на рівні груп.

7. **Налаштувати націлювання**. Налаштування націлювання дає змогу визначити цільову аудиторію. Якщо не налаштувати націлювання, оголошення показуватимуться загальній аудиторії відвідувачів.

Таргетинг дає змогу обмежити охоплення та показувати рекламу лише тим користувачам, яким цікаві ці товари чи послуги. Найчастіше використовується націлювання на ключові слова, аудиторію, розташування, теми, а також ремаркетинг. Доступні види націлювання залежать від типу кампанії.

8. **Налаштувати відстеження конверсій.** За допомогою відстеження конверсій можна отримувати відомості про важливі дії користувачів, які здійснюються на сайті. Це допоможе точніше оцінювати ефективність об’яв, налаштувань націлювання та рекламних кампаній загалом.

Контрольні запитання

1. Які переваги має реклама в мережі Інтернет?
2. Які Ви знаєте види інтернет-реклами?
3. Чим нативна реклама відрізняється від банерної?
4. Що таке мобільна реклама?
5. Що таке контекстна реклама?
6. Що таке таргетингова реклама?
7. Які параметри налаштувань застосовують для налаштування таргетингової реклами?
8. Які ви знаєте види таргетингової реклами?
9. Як створюється рекламна кампанія в Google Ads?
10. Які кроки потрібно виконати для створення успішної рекламної кампанії в Google?

Лекція 10
НЕЙРОМЕРЕЖІ

Нейромережа *(штучна нейронна мережа)* – це математична модель, а також її програмне чи апаратне втілення, що імітує структуру та функціонування біологічних нейронних мереж із метою вирішення різноманітних задач, як-от класифікація, регресія, прогнозування та генерація.

Поняття «Нейромережа» виникло під час вивчення процесів, що виникають у мозку людини, під час спроби змоделювати ці процеси. Після розробки алгоритмів навчання отримані моделі почали використовувати в практичних цілях: у задачах прогнозування, для розпізнавання образів, у задачах управління тощо.

В основі *нейромереж* лежать штучні *нейрони*, які об'єднуються в графові структури й передають сигнали один одному через *ваги зв'язків*.

Завдяки процесу навчання, під час якого *ваги* та зміщення між *нейронами* оптимізуються, *нейромережі* стають здатними до виявлення закономірностей і залежностей у *вихідних даних*. У наш час *нейромережі* активно використовуються в таких галузях, як *комп'ютерний зір*, *машинний переклад*, *розпізнавання мови*.

Нейромережа являє собою комп'ютерну систему, яка намагається відтворити роботу людського мозку як процес сприйняття інформації та навчання, самостійно вчиться та вдосконалюється, намагаючись стати схожою на людський мозок у вирішенні різних задач.

10.1. Базова структура та компоненти нейромережі

Структура *нейромережі* складається з трьох основних шарів: *вхідний шар*, *приховані шари* та *вихідний шар*.

Шар у *нейромережі* – це група нейронів, які працюють разом і виконують певну функцію в мережі. *Шари* між собою з'єднані. Кожен *шар* має свою роль у процесі обробки інформації.

Вхідний шар приймає дані ззовні, наприклад зображення або

текст, і передає їх у наступні шари. *Вхідний шар* не змінює дані, а лише слугує точкою входу для них.

Приховані шари забезпечують обробку вхідних даних і передачу інформації між шарами. Вони називаються «прихованими», оскільки їхні результати не відображаються напряму на виході мережі. Кількість прихованих шарів та *нейронів* у них може варіюватися залежно від складності задачі й архітектури мережі.

Вихідний шар формує результат, який *нейромережа* передбачає на основі вхідних даних. Результатом може бути класифікація, числове значення або інша інформація залежно від типу задачі.

Отже, *шари* в *нейромережі* – це *групи нейронів*, які працюють разом і відповідають за різні етапи обробки інформації. Вони забезпечують *нейромережі* можливість адаптації до різних задач.

Усі *нейрони* в *шарах* з'єднані між собою через *ваги зв'язків*. *Ваги* відіграють важливу роль у навчанні *нейромережі*, оскільки вони визначають силу впливу одного *нейрона* на інший. У процесі навчання *ваги* оптимізуються, щоб мінімізувати *помилку передбачення* мережі.

Крім *ваг зв'язків*, кожен *нейрон* має так зване *зміщення*, яке дає змогу регулювати активацію *нейрона* незалежно від вхідного сигналу. *Зміщення* допомагає *нейромережі* легше адаптуватися до різних даних і виконувати більш гнучкі перетворення на вхідних даних.

Ще одним ключовим компонентом *нейромережі* є *функція активації*. Вона застосовується до кожного *нейрона* у *прихованих* і *вихідних шарах*, щоб визначити його активність на основі суми вхідних сигналів, помножених на відповідні *ваги* з додаванням *зміщення*. *Функція активації* може бути *лінійною* або *нелінійною* залежно від типу задачі й архітектури мережі.

Навчання *нейромережі* полягає в оптимізації *ваг зв'язків* та *зміщень* на основі навчального набору даних. Для цього зазвичай використовують *метод зворотного поширення помилки*, який

базується на *градієнтному спуску*. Процес навчання може тривати довгий час, залежно від розміру набору даних, архітектури мережі та складності задачі.

Отже, базова структура та компоненти нейромережі мають вхідний, прихований і вихідний шари, нейрони з вагами зв'язків та зміщеннями, а також функції активації.

Разом вони сприяють адаптації нейромережі до вихідних даних і вирішенню складних задач.

10.2. Відмінності між біологічними та штучними нейронними мережами

Біологічні нейронні мережі складаються з біологічних клітин – *нейронів*, які передають імпульси через синаптичні зв'язки. А *штучні нейромережі*, навпаки, базуються на *математичних моделях* і *комп'ютерних алгоритмах*, імітуючи функціонування біологічної мережі. Однак *штучні нейромережі* мають менш складну структуру й обмежену здатність до навчання порівняно з біологічними аналогами. Водночас *штучні нейромережі* демонструють вражаючі результати в різних галузях науки та техніки, включно з комп'ютерним зором, машинним перекладом, розпізнаванням мови й автономним рухом транспортних засобів.

Однією з ключових відмінностей між *біологічними* та *штучними нейронними мережами* є швидкість передачі сигналів і навчання. Біологічні нейронні мережі можуть передавати імпульси зі швидкістю до 120 метрів за секунду, тоді як штучні мережі передають інформацію зі швидкістю сучасних комп'ютерних процесорів. Також *штучні нейромережі* навчаються значно швидше, ніж біологічні, завдяки можливості використовувати паралельні алгоритми й оптимізацію обчислень.

Водночас біологічні нейронні мережі мають значно більшу кількість *нейронів* та *зв'язків*, що забезпечують їм перевагу в аналізі складних ситуацій і формуванні адаптивних стратегій. *Штучні нейромережі* ж, незважаючи на їх прогрес, поки що не можуть повністю відтворити всі функції людського мозку.

10.3. Роль нейромереж у вирішенні задач

Нейромережі відіграють важливу роль у вирішенні проблем, оскільки вони можуть вивчати й адаптуватися до різних типів даних і вирішувати складні завдання. Їх головна мета полягає в тому, щоб знайти шаблони й застосовувати ці знання для передбачення або класифікації нових даних.

Нейромережі можуть виконувати різні типи завдань, залежно від архітектури мережі та навчальних даних. До основних видів завдань належать:

класифікація – визначення категорії, до якої належить певний об’єкт або подія, на основі його характеристик;

регресія – прогнозування числового значення на основі вихідних даних;

генерація тексту – створення тексту на основі навчальних даних, зазвичай використовується для автоматичного створення описів, статей або відповідей на запитання;

обробка зображень – розпізнавання об’єктів, тексту або об’єктів на зображеннях.

Нейромережі відіграють важливу роль у вирішенні проблем, виконуючи різні типи завдань у багатьох галузях промисловості та сферах життя. Завдяки своїй спроможності навчатися й адаптуватися до різних даних, нейромережі стають все більш популярними та корисними інструментами для сучасного світу.

10.4. Застосування нейромереж

Нейромережі знаходять широке застосування в різних галузях промисловості та сферах життя:

Медицина – діагностика захворювань, аналіз медичних зображень, передбачення ефективності лікування.

Фінанси – виявлення шахрайства, прогнозування цін на акції, оптимізація портфелів інвестицій.

Маркетинг – прогнозування вподобань клієнтів, автоматичне створення рекламних матеріалів, аналіз споживчої поведінки.

Автоматичний переклад – нейромережі можуть навчатися перекладати тексти між різними мовами, забезпечуючи швидкий і точний переклад.

Розпізнавання мови – розпізнавання та розуміння голосових команд для керування різними пристроями, як-от смартфони та побутова техніка.

Автономні транспортні засоби – навігація, ухилення від перешкод і безпечне керування автономними автомобілями або дронами.

Безпека – аналіз відео та зображень для виявлення підозрілої діяльності, захист від кібератак або виявлення шахрайства.

Рекомендаційні системи – здатні аналізувати історію перегляду користувача та рекомендувати продукти, фільми, музику тощо на основі його вподобань.

Нейромережі використовуються в різних галузях і сферах діяльності завдяки своїй спроможності навчатися, адаптуватися та вирішувати складні проблеми.

У сфері обробки природної мови *нейромережі* допомагають у розпізнаванні мови, аналізі емоцій, генерації тексту й автоматичному перекладі. Прикладом може слугувати *Google Translate*, який використовує *нейромережі* для перекладу текстів між різними мовами з високою точністю.

У сфері розпізнавання образів та комп'ютерного зору *нейромережі* широко використовуються для розпізнавання образів: від класифікації зображень до виявлення об'єктів на відео. Такі нейромережі можна зустріти в системах безпеки, автономних автомобілях та медичній діагностиці.

У фінансовій сфері *нейромережі* застосовуються для прогнозування курсів валют, ринкових трендів та оцінки кредитоспроможності клієнтів. Це допомагає фінансовим установам приймати обґрунтовані рішення та знижувати рівень ризику.

У медичній галузі *нейромережі* використовуються для розпізнавання патологічних змін на зображеннях, отриманих за допомогою МРТ, КТ, рентгену й інших діагностичних методів. Вони

допомагають в аналізі генетичних даних, прогнозуванні результатів лікування та розробці нових лікарських засобів.

У геології та кліматології *нейромережі* застосовуються для аналізу та прогнозування землетрусів, повеней та інших природних катаклізмів. Це допомагає науковцям і організаціям планувати заходи безпеки та зменшувати наслідки стихійних лих.

У відеоіграх та віртуальній реальності *нейромережі* використовуються для створення реалістичного штучного інтелекту, який контролює персонажі ігор та інші аспекти ігрового середовища. Це забезпечує глибше занурення та вищу якість ігрового процесу.

У маркетингу та рекламі *нейромережі* допомагають аналізувати дані про поведінку споживачів, прогнозувати тенденції та розробляти ефективні рекламні кампанії. Вони також можуть оптимізувати розміщення реклами на *web*-сайтах і в соціальних медіа, що збільшує конверсію та віддачу від рекламних інвестицій.

Нейромережі допомагають рекомендаційним системам пропонувати користувачам товари та послуги на основі їх інтересів і взаємодії з платформами.

У смарт-містах *нейромережі* використовуються для управління розумним освітленням, контролю транспорту, розподілу енергії та забезпечення безпеки. Це допомагає оптимізувати ресурси, підвищити енергоефективність і забезпечити високий рівень комфорту для мешканців міст.

Нейромережі використовуються в робототехніці для навчання роботів розуміти та інтерпретувати дії людей, навігації в незнайомому середовищі й виконання складних завдань. Це відкриває можливості створення роботів, які можуть працювати поряд з людьми, надавати допомогу та виконувати різні завдання в промисловості й домашньому середовищі.

У біотехнологіях *нейромережі* використовуються для розуміння складних хімічних реакцій, моделювання біологічних процесів і розробки нових лікарських засобів. Вони можуть прискорити процес відкриття нових препаратів і допомогти науковцям знаходити інноваційні рішення в лікуванні хвороб.

Завдяки своїй гнучкості, здатності адаптуватися й вирішувати складні проблеми *нейромережі* продовжують знаходити нові застосування в різних галузях науки, промисловості та повсякденного життя.

10.5. Тонкощі навчання нейромереж

Навчання нейромережі – процес, під час якого мережа вчиться адаптуватися й розпізнавати закономірності в наданих їй даних. Цей процес допомагає *нейромережі* забезпечувати коректні висновки та передбачення, що засновані на нових даних, які не були використані під час навчання.

10.5.1. Процес налаштування ваг і зсувів

Основою навчання *нейромережі* є налаштування *ваг* (*зв'язків* між нейронами) і *зсувів* (порогів активації *нейронів*). Під час процесу навчання *нейромережа* постійно коригує *ваги* та *зсуви*, щоб мінімізувати похибку між передбаченнями мережі й реальними результатами.

10.5.2. Роль швидкості навчання й алгоритмів оптимізації

Швидкість навчання визначає рівень зміни *ваг* і *зсувів* під час кожної ітерації процесу навчання. Занадто висока швидкість навчання може призвести до пропуску оптимальних значень, а низька швидкість навчання може забезпечити повільний процес тренування.

Алгоритми оптимізації, як-от *градієнтний спуск* або *адаптивні методи*, використовуються для знаходження оптимальних значень *ваг* і *зсувів*, які мінімізують *функцію втрат* (похибку між передбаченнями та реальними даними). Знання тонкощів тренування *нейромереж* дає можливість створювати ефективні та точні моделі для різноманітних завдань і застосунків.

Щоб використовувати *нейромережі* для вирішення конкретних задач, спочатку потрібно підготувати набір даних для навчання. Цей набір даних може містити приклади, які відображають відносини між вихідними даними та бажаними результатами. Зазвичай набір

даних поділяється на *навчальний*, *валідаційний* та *тестовий*, щоб контролювати процес навчання й оцінювати його результати.

Під час навчання *нейромережі* демонструються приклади з навчального набору даних, алгоритми оптимізації налаштовують *ваги* та *зсуви*, враховуючи швидкість навчання. *Валідаційний набір* даних використовується для оцінки якості моделі під час навчання, надаючи можливість виявити *перенавчання* або *недонавчання*.

Після завершення процесу навчання *нейромережа* готова до застосування в реальних ситуаціях, де вона може використовуватися для передбачення результатів, класифікації, розпізнавання образів, обробки мови та інших задач.

Отже, навчання *нейромережі* передбачає ряд важливих аспектів, як-от налаштування *ваг* і *зсувів*, *швидкість навчання*, *алгоритми оптимізації* та робота з наборами даних. Розуміння цих тонкощів допоможе створювати та застосовувати ефективні *нейромережі* для вирішення складних задач і підвищення продуктивності різних галузей промисловості.

10.6. Глибоке навчання

Глибоке навчання – це розвиток і розширення класичних *нейромереж*, що забезпечує здатність розпізнавати, аналізувати та класифікувати більш складні шаблони та ієрархії. Перевагами *глибокого навчання* є його висока точність і здатність обробляти великі обсяги даних.

Глибокі нейромережі складаються з багатьох шарів нейронів, які взаємодіють між собою, передаючи інформацію від входу до виходу.

Глибокі нейромережі демонструють надзвичайну ефективність у розпізнаванні образів, обробці мови та інших задачах, які класичні алгоритми вирішують із труднощами. Завдяки *глибокому навчанню* стали можливі прориви в таких сферах, як комп’ютерний зір, автономні автомобілі та машинний переклад.

Глибоке навчання має свої обмеження.

По-перше, воно потребує величезних обсягів даних та

обчислювальної потужності для навчання й оптимізації моделей.

По-друге, *глибокі нейромережі* можуть бути складними для інтерпретації та пояснення, що ускладнює розуміння того, як вони приймають рішення.

По-третє, може виникнути проблема *перенавчання*, коли модель надто добре «запам'ятовує» навчальні дані, але погано узагальнює нову інформацію.

Попри обмеження, *глибоке навчання* продовжує розвиватися та вносити значні зміни в багатьох галузях науки та техніки. Дослідники активно працюють над удосконаленням алгоритмів *глибокого навчання*, зменшенням вимог до обчислювальної потужності й поліпшенням зрозумілості та інтерпретації моделей.

Однією з таких тенденцій є використання технік *передачі навчання,* які дають змогу використовувати знання, отримані під час навчання однієї моделі, для швидкого навчання інших моделей у схожих задачах. Це може сприяти ефективному використанню ресурсів і підвищенню загальної продуктивності *глибокого навчання*.

Також з'являються нові архітектури *нейромереж*, які намагаються відтворити більш точні моделі людського мозку та його роботи, наприклад *капсульні мережі (capsule networks)* та *спайкові нейронні мережі (spiking neural networks)*. Вони можуть привести до створення ще більш потужних та ефективних систем *глибокого навчання*.

10.7. Робота з нейромережами

Процес роботи з нейромережами починається з вибору відповідної архітектури, що найкраще підходить для рішення конкретної задачі. Вибір залежить від типу даних, обсягу даних, а також від складності проблеми, яку потрібно вирішити. Різні архітектури нейромереж, як-от *згорткові*, *рекурентні* та *глибокі*, пропонують різні можливості для різних сценаріїв застосування.

Існує декілька основних архітектур нейромереж, кожна з яких призначена для різних типів задач та їх застосувань.

Штучні нейронні мережі прямого поширення *(Feedforward*

Neural Networks) – найпростіша архітектура нейромережі, у якій інформація передається в одному напрямі від входу до виходу через різні шари. Вони не мають циклів або зворотних зв'язків і складаються з одного або кількох прихованих шарів.

Згорткові нейронні мережі *(Convolutional Neural Networks)* розроблені спеціально для роботи з даними, що мають просторову структуру, як-от зображення. Вони використовують *згорткові шари* для автоматичного виявлення особливостей зображень замість ручного інженерного проектування.

Рекурентні нейронні мережі *(Recurrent Neural Networks)* розроблені для роботи з послідовними даними, як-от текст або числові ряди. Вони мають зворотні зв'язки, які дають змогу пам'ятати інформацію з попередніх кроків.

Довга короткочасна пам'ять *(Long Short-Term Memory)* – різновид *рекурентної нейромережі*, який вирішує проблему затухання градієнта, яка виникає під час навчання традиційних *рекурентних нейромереж*. *LSTM* має спеціальну структуру вузлів, які дають змогу моделі «пам'ятати» або «забувати» інформацію протягом тривалого періоду часу.

Мережі згорткового автоенкодера *(Convolutional Autoencoders)* – нейромережі, що навчаються кодувати вхідні дані в компактному представленні, а потім реконструювати вихідні дані з цього представлення. *Згорткові автоенкодери* використовують згорткові шари для роботи з просторовими даними, наприклад із зображеннями, і зазвичай використовуються для виявлення особливостей або відтворення зображень.

Генеративно-змагальні мережі *(Generative Adversarial Networks)* складаються з двох окремих нейромереж, які працюють разом: *генератора*, який створює *синтетичні* дані, і *дискримінатора*, який навчається відрізняти справжні дані від *синтетичних*. Такі мережі зазвичай використовуються для генерації зображень і текстів.

Капсульні мережі *(Capsule Networks)* – тип *згорткових нейромереж*, які використовують спеціальні *капсульні шари* для

збереження інформації про просторові відносини між об’єктами на зображенні. Вони можуть краще розуміти ієрархічні структури даних і зазвичай працюють краще, ніж традиційні згорткові мережі, для задач розпізнавання об’єктів.

Мережі з увагою *(Attention Networks)* – архітектура, яка дає можливість моделям приділяти більше уваги важливим частинам вхідних даних. Механізм *уваги* зазвичай використовується в комбінації з *рекурентними* мережами, особливо в задачах обробки природного мовлення, як-от машинний переклад і генерація тексту.

Мережі граф-нейронів *(Graph Neural Networks)* – архітектура нейромереж, яка працює з графічними структурами даних. Вони здатні обробляти відносини між об’єктами й агрегувати інформацію із сусідніх вузлів. Такі мережі часто використовуються в рекомендаційних системах, аналізі соціальних мереж, хімічному моделюванні.

Спайкові нейронні мережі – це клас мереж, які намагаються більш точно моделювати динаміку *спайкових нейронів* біологічного мозку. Вони роблять це шляхом впровадження темпоральної компоненти в активаційні функції нейронів.

Передпроцесування даних та *інженерія ознак* – це критично важливі етапи роботи з нейромережами. Вони забезпечують очищення та перетворення вхідних даних у формат, який може бути легко оброблений нейромережею.

Передпроцесування даних здійснює нормалізацію, заповнення пропущених значень і видалення шуму.

Інженерія ознак передбачає вибір найважливіших ознак і створення нових ознак, що можуть поліпшити ефективність моделі.

Під час навчання нейромережі слід дотримуватися кількох основних принципів.

По-перше, розділити дані на *навчальний*, *валідаційний* і *тестовий* набори, щоб мати можливість оцінити ефективність моделі й уникнути перенавчання.

По-друге, використати методи оптимізації, зокрема *градієнтний спуск* або *адаптивні методи*, які допомагають

знаходити оптимальні параметри моделі.

По-третє, під час налаштування *гіперпараметрів*, як-от *швидкість навчання, кількість шарів* або *кількість нейронів*, використати такі методи, як *перехресна перевірка (cross-validation)* або *пошук на сітці (grid search)*, щоб знайти оптимальні значення для поточної моделі.

Також важливо контролювати процес навчання нейромережі, використовуючи такі інструменти, як *візуалізація функції втрат (loss function)* і *метрик точності (accuracy metrics)*. Це дасть змогу слідкувати за успішністю навчання та виявляти проблеми з *перенавчанням* або *недонавчанням*.

У разі потреби можна зупинити навчання або змінити *гіперпараметри* для поліпшення результатів.

Враховуючи всі ці аспекти роботи з нейромережами, можна побудувати ефективні й надійні моделі, які допоможуть вирішити складні задачі та принести цінні результати в різних сферах застосування.

З практичним досвідом і застосуванням кращих практик навчання та налаштування нейромереж можна вдосконалити свої навички та зробити значний внесок у розвиток *штучного інтелекту*.

10.8. Нейромережа і штучний інтелект

Нейронна мережа та *штучний інтелект* є тісно пов’язаними поняттями, але вони мають деякі відмінності.

Штучний інтелект – це галузь комп’ютерних наук, яка досліджує методи створення машин, здатних до інтелектуальної діяльності, аналогічної людському розуму. Це передбачає навчання, мовне розпізнавання, сприйняття, уміння розв’язувати проблеми, адаптацію та інші аспекти людської інтелектуальної діяльності.

Штучний інтелект охоплює різні методи та техніки для досягнення цих цілей, серед яких *нейронні мережі* є одним із підходів.

Нейронні мережі використовуються в галузі *штучного інтелекту* для навчання комп’ютерів розпізнавати шаблони,

класифікувати дані й виконувати інші завдання, що потребують інтелектуального аналізу, через те, що нейронні мережі складаються з великої кількості взаємопов'язаних вузлів (нейронів), які обробляють інформацію паралельно, адаптуючись до вихідних даних.

Отже, основна відмінність між *штучним інтелектом* і *нейронними мережами* полягає в тому, що *штучний інтелект* є широкою галуззю, яка охоплює різні методи та техніки для досягнення інтелектуальної діяльності, тоді як *нейронні мережі* – це один із підходів до створення *штучного інтелекту*, який зосереджується на імітації біологічних нейронних мереж для обробки інформації та навчання.

Деякі підходи до *штучного інтелекту*, окрім *нейронних мереж*, охоплюють інші системи.

Експертні системи – комп'ютерні програми, які використовують бази знань і правила для моделювання рішень експертів у певній предметній сфері. Вони здатні відповідати на запитання, поставлені користувачем, і надавати рекомендації аналогічно тому, як це робить людина-експерт.

Методи пошуку й оптимізації – алгоритми, які використовуються для розв'язання складних проблем і пошуку найкращих можливих рішень. Приклади містять *алгоритми градієнтного спуску, генетичні алгоритми* та *інтелектуальний пошук*.

Машинне навчання – підгалузь штучного інтелекту, яка досліджує методи навчання комп'ютерів безпосередньо з даних, без явного програмування. Машинне навчання передбачає ряд алгоритмів, як-от регресія, класифікація та кластеризація, які можуть бути застосовані для різних задач, від розпізнавання образів до прогнозування.

Методи символічного обчислення – підхід, який використовує символічні представлення знань і маніпуляції з ними для розв'язання проблем. Приклади містять *автоматичне доведення теорем, символічне обчислення* й *інтелектуальний аналіз*.

Формальна логіка та методи виведення – підходи, які використовують логічні форми і правила для представлення знань і виведення нових фактів або рішень.

Ці методи передбачають *класичну логіку першого порядку, теорію множин, нечітку логіку* та багато інших. Вони можуть бути застосовані для різних задач, як-от автоматичне міркування, планування, управління знаннями та виявлення залежностей.

Інтелектуальні агенти та розподілені системи – підходи, які використовують кілька незалежних агентів або суб'єктів, які взаємодіють один з одним для вирішення задач.

Ці системи можуть використовувати алгоритми координації, переговорів, аукціонів та інші механізми для досягнення спільної мети.

У різних сферах штучного інтелекту нейронні мережі зазвичай використовуються разом з іншими методами та техніками для покращення результатів. Наприклад, у задачах розпізнавання мови або образів можуть використовуватися комбінації нейронних мереж, машинного навчання й алгоритмів оптимізації.

Комплексні системи штучного інтелекту можуть містити елементи з різних підходів, надаючи можливість розв'язувати складні проблеми й адаптуватися до нових ситуацій.

10.9. Нейромережі й обладнання для їх роботи

Щоб забезпечити ефективну роботу нейромереж, потрібно використовувати відповідне обладнання. Розглянемо основні типи обладнання, на якому працюють нейромережі.

Центральний процесор *(CPU)* – це загальноприйнятий компонент комп'ютера, який виконує арифметичні та логічні операції. Нейромережі можуть працювати на *CPU,* але через велику кількість одночасних операцій, які потрібно виконати, робота нейромереж на *CPU* може бути повільною.

Графічний процесор *(GPU)* були розроблені спеціально для обробки великої кількості графічних даних. Вони мають більше ядер, ніж *CPU,* і відрізняються паралельною архітектурою, що дає

змогу виконувати велику кількість операцій одночасно. Завдяки цьому *GPU* є відмінним вибором для роботи з нейромережами, оскільки вони прискорюють навчання й обробку даних.

Tensor Processing Unit *(TPU)* – це спеціалізований вид обладнання, розроблений компанією *Google* спеціально для роботи з нейромережами. *TPU* оптимізовані для виконання операцій з *тензорами*, які є основними компонентами нейромереж. Вони забезпечують значно вищу продуктивність порівняно з *CPU* та *GPU* під час обробки великої кількості даних.

Відповідне програмне забезпечення. Важливим аспектом роботи нейромереж є наявність відповідного програмного забезпечення. Це можуть бути бібліотеки та фреймворки, зокрема *TensorFlow, PyTorch, Keras* та інші, які допомагають створювати, навчати й розгортати нейромережі. Вони містять попередньо підготовлені модулі, алгоритми оптимізації та інші інструменти, які спрощують розробку та реалізацію нейромереж на різних типах обладнання.

Обчислювальні хмари та пристрої Edge AI. Нейромережі можуть працювати не тільки на персональних комп'ютерах, але й на серверах у хмарних сервісах (наприклад, *AWS, Google Cloud, Microsoft Azure*) або пристроях *Edge AI*, які розраховані на обробку даних безпосередньо на пристроях, що розташовані ближче до джерела даних (наприклад, *IoT*-пристрої, дрони, роботи тощо). Це дає змогу оптимізувати швидкість обробки даних і заощадити ресурси.

Нейромережі можуть працювати на різних типах обладнання, зокрема *CPU, GPU, TPU*, хмарних серверах і пристроях *Edge AI*. Вибір конкретного типу обладнання залежить від потреб проєкту, наявності ресурсів і специфіки завдань, які повинна виконати нейромережа.

Незалежно від вибору обладнання важливо мати відповідне програмне забезпечення для створення, навчання та розгортання нейромережі.

Нейромережі відіграють важливу роль у розвитку сучасних

технологій, оскільки вони дають можливість моделювати складні процеси та розв’язувати задачі, які раніше були не під силу класичним алгоритмам. Вони знайшли застосування в різних галузях.

Розв’язання складних задач. Нейромережі дають можливість розглядати й аналізувати велику кількість даних, враховуючи їх нестандартність і нелінійність. Це сприяє вирішенню складних проблем, які традиційними методами не можуть бути розв’язані.

Адаптація та навчання. Однією з ключових переваг нейромереж є можливість навчання й адаптації до нових даних. Це дає можливість системам із нейромережами покращувати свої рішення та пристосовуватися до змін у світі.

Багатогранність застосувань. Нейромережі можуть бути використані в різних галузях, оскільки вони легко адаптуються до різних задач і можуть працювати з різними типами даних. Це робить їх універсальним інструментом для розв’язання широкого спектра задач.

Отже, нейромережі є важливим елементом сучасних технологій, що відкривають нові можливості в різних галузях і дають змогу розв’язувати складні задачі шляхом адаптації та навчання.

Контрольні запитання

1. Що таке нейромережа?
2. Із чого складається базова структура нейромережі та які її компоненти?
3. У яких галузях застосовуються нейромережі?
4. Що таке процес навчання нейромережі?
5. Що таке глибоке навчання нейромережі?
6. Які обмеження має глибоке навчання нейромережі
7. Які архітектури нейромереж існують і для яких типів задач кожна з них призначена?
8. Що таке штучний інтелект?

9. Яке обладнання використовується для роботи нейромереж?

10. Що таке тензорний процесор?

Список літератури

1. *Ален Тейлор*. SQL для чайників. 8-ме видання. Пер. з англ. Діалектика, 2022. – 544 с.
2. *Блозва А. І.* Комп’ютерні мережі : навчальний посібник / А. І. Блозва, Ю. В. Матус, В. В. Смолій, Б. С. Гусєв, Д. Ю. Касаткін, Т. Ю. Осипова, Я. А. Савицька. – Київ : Компрінт, 2017. – 821 с.
3. *Вайк Ален и др*. JavaScript в примерах. Пер. с англ. – Киев : Издательство «Дик Софт», 2000. – 304 с.
4. *Джош Локхарт*. Современный PHP: новые возможности и передовой опыт. – Пер. с англ. – Киев : ДМК ПРЕСС, 2016. – 303 с.
5. *Жураковський Б. Ю.* Комп’ютерні мережі. Частина 1 : навчальний посібник [Електронний ресурс] / Б. Ю. Жураковський, І. О. Зенів. – Київ : КПІ ім. Ігоря Сікорського, 2020. – 336 с.
6. *Задерейко О. В.* Комп’ютерні мережі : навчально-методичний посібник [Електронне видання] / О. В. Задерейко, Н. В. Багнюк, А. А. Толокнов. – Одеса : Фенікс, 2023. – 210 с.
7. *Казакова Н. Ф.* Інформаційні системи і технології в менеджменті : навчальний посібник / Н. Ф. Казакова, О. О. Фразе-Фразенко. – Одеса : Фенікс, 2017. – 215 с.
8. *Лізунов П. П.* Інформаційні системи і технології в управлінні організацією : навчальний посібник / П. П. Лізунов, М. В. Коханович, В. О. Недін. – Київ : КНУБА, 2018. – 156 с.
9. *Фрімен Ерік.* Програмування на JavaScript / Ерік Фрімен, Елізабет Робсон. – Київ : Фабула, 2022. – 672 с.
10. *McFarland David*. CSS: The Missing Manual. 4th Edition. – O'Reilly Media, Inc., 2015. – 715 p.
11. *Robbins Jennifer.* Pocket Reference. 5th Edition. – O'Reilly Media, Inc., 2013. – 104 p.

ДЛЯ НОТАТОК

Навчальне видання

ЛІЗУНОВ Петро Петрович,
НЕДІН Валентин Олегович,
КАРА Ірина Дмитрівна

КОМП'ЮТЕРНІ МЕРЕЖІ ТА ТЕЛЕКОМУНІКАЦІЇ

Конспект лекцій

Редагування та коректура Т. В. Івченко
Комп'ютерне верстання Л. В. Лабунець

Підписано до друку 13.03.2025. Формат $60\times84_{1/16}$
Ум. друк. арк. 13,95. Обл.-вид. акр. 15,0.
Електронний документ. Вид. № 6/І-25.

Видавець і виготовлювач
Київський національний університет будівництва і архітектури

Проспект Повітряних Сил, 31, Київ, 03037

Свідоцтво про внесення до Державного реєстру суб'єктів
видавничої справи ДК № 808 від 13.02.2002