КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЮ «МАГІСТР»

на тему: «Інформаційна система моніторингу документообігу корпоративної компанії»

КАРАЧУН МАКСИМ СЕРГІЙОВИЧ

(прізвище, ім’я та по батькові студента повністю)

Київ 2023 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
к.т.н., доцент Гончаренко Т.А.

_____
„___” _____2023 року

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЮ «МАГІСТР»

на тему: "Інформаційна система моніторингу документообігу корпоративної компанії"

Виконав: студент ІІ-го курсу, групи КНм-ІІ

Спеціальності: 122 «Комп`ютерні науки» технології»
Освітня програма: Комп`ютерні науки
(шифр і назва напряму підготовки, спеціальності)

Карачун М.С.
(прізвище та ініціали)

Керівник __д.т.н., проф. Терентьєв О.О.
(прізвище та ініціали)

Рецензент __к.т.н., доц. Шабала Є.Є.
(прізвище та ініціали)

Київ, 2023 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: __автоматизації і інформаційних технологій
Кафедра:__ __інформаційних технологій
Освітній рівень: «магістр» за ОПП
Спеціальність: _122 «Комп`ютерні науки»
Освітня програма:__ Комп`ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
к.т.н., доцент Гончаренко Т.А.
_____
„___” _____2023 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЮ «МАГІСТР»

_____Карачун Максим Сергійович_____
(прізвище, ім’я та по батькові студента)

1. Тема роботи:__ Інформаційна система моніторингу документообігу корпоративної компанії
затверджена наказом ректора КНУБА № _____ від «____» _____2023 р.
2. Керівник роботи:Терентьєв Олександр Олександрович, д.т.н, професор кафедри інформаційних технологій проектування і прикладної математики
3. Строк подання студентом роботи до захисту: ____грудень 2023 р.
4. Зміст пояснювальної записки за розділами:
Р. 1._Аналіз системи документообігу
Р. 2._Розробка архітектури системи
Р. 3._Моделювання системи
Р. 4._Розробка програмного забезпечення
5. Інформаційні слайди
С. 1._ Основні функції системи електронного документообігу
С. 2._ Розробка архітектури системи
С. 3._Концептуальна модель
С. 4._Інфологічна модель
С. 5._Даталогічна модель

С. 6. Програмне забезпечення системи. Тестовий приклад програми

6. Календарний план виконання атестаційної випускної роботи

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз системи документообігу	Вересень 2023 р.
Р. 2. Розробка архітектури системи	Жовтень 2023 р.
Р. 3. Моделювання системи	Листопад 2023 р.
Р. 4. Розробка програмного забезпечення	Грудень 2023 р.
Остаточне оформлення роботи	Грудень 2023 р.
Направлення роботи на рецензування, перевірку на плагіат	02 грудня 2023 р.
Попередній захист роботи на кафедрі	05 грудня 2023 р.

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта, представника комісії	дата	підпис
Прийом програмного продукту	к.т.н. доц. Шабала Є.Є.		

8. Дата видачі завдання: 05 вересня 2023 року

Керівник _____ Терентьєв О.О.
(підпис) (прізвище та ініціали)

Магістрант _____ Карачун М.С.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Карачун М.С. «Розробка інформаційної системи моніторингу документообігу корпоративної компанії».

Атестаційна випускова робота магістра за спеціальністю: 122 «Комп`ютерні науки», освітня програма: «Комп`ютерні науки». – Київський національний університет будівництва та архітектури. – Київ, 2023.

Досліджено систему документообігу і способи її моніторингу в корпоративній компанії. При розробці системи моніторингу враховано особливості та недоліки системи документообігу, що сформувалась в компанії за попередній час існування. Вдосконалення спеціалізованої системи моніторингу здійснюється шляхом автоматизації процесів виявлення і обробки несистемних помилок.

Ключові слова. документообіг, запит, обробка, помилка, моніторинг, система.

SUMMARY

Karachun M.S. "Development of an information system for monitoring the document flow of a corporate company."

Certification master's thesis in the specialty: 122 "Computer Science", educational program: "Computer Science". - Kyiv National University of Construction and Architecture. - Kyiv, 2023.

The system of document circulation and methods of its monitoring in a corporate company are investigated. The development of the monitoring system takes into account the features and shortcomings of the document management system, which was formed in the company during its previous existence. Improvement of the specialized monitoring system is carried out by automation of processes of detection and processing of non-system errors.

Keywords. document flow, request, processing, error, monitoring, system.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

БД – База даних.

ЕД – Електронний документообіг.

ІТ – Інтелектуальні технології

КК – Корпоративна компанія.

ОТС – Організаційно-технічна система.

САД – Система автоматизації документообігу.

СЕД – Система електро́нного документоо́бігу.

СД – Система документообігу.

СМ – Система моніторингу.

ССМД – Спеціалізована систему моніторингу документообігу

ОНС – Об'єкти навколишнього середовища.

ОТС – Організаційно-технічна система.

РБД – Реляційнаі база даних .

РП – ринок посередників

КК – Корпоративна компанія.

Документообіг установи – це процес проходження документів в установі з моменту створення чи одержання до завершення виконання або відправлення [1].

Дистрибу́ція (дистриб'юція) – поняття логістики, яке означає комплекс взаємопов'язаних функцій, що реалізуються в процесі розподілення матеріального потоку між різними, як правило, гуртовими покупцями. Таким чином, розподільча логістика, або фізичний розподіл – це діяльність, що пов'язана з отриманням продукції, її зберіганням до моменту отримання замовлення і наступної доставки до клієнтів.

Систе́ма автоматиза́ції документоо́бігу (Систе́ма електро́нного документоо́бігу) – організаційно-технічна система, що забезпечує процес створення, управління доступом і поширення електронних документів в

комп'ютерних мережах, а також що забезпечує контроль над потоками документів в організації [4].

Організаційно-технічна система – Це множина взаємопов'язаних матеріальних об'єктів (технічних засобів і персоналу, який забезпечує їх функціонування і застосування за призначенням), призначених для безпосереднього виконання операції.

З системних позицій, ОТС – ієрархічний людино-машинний комплекс, цілеспрямовано функціонує з метою реалізації його властивостей відповідно до його цільового призначення. Найпростішою (елементарної) ОТС є робоче місце з штатним персоналом. У загальному випадку реальна ОТС являє собою ієрархічну систему, що утворена множиною елементарних ОТС.

Під *організацією ОТС* розуміється спосіб взаємозв'язку і взаємодії між її елементами, що забезпечують їх об'єднання в дану ОТС. Організація ОТС підрозділяється на постійну (інваріантну) та змінні частини. Перша частина визначає структуру ОТС, а друга – програму її функціонування. У загальному випадку ОТС складається з двох підсистем: керуючої і керованої, елементи якої реалізують функції перетворення ресурсів в результат.

Навколишнім середовищем називаються об'єкти, що не входять в ОТС. Об'єкти навколишнього середовища (ОНС) можуть впливати на ОТС, ресурси або результат її функціонування.

Логістикою називають будь-які процеси, що пов'язані з транспортуванням, зберіганням та обробкою будь-яких предметів.

Ланцюг поставок – це множина ланок логістичної системи, за допомогою якої можна простежити потік переміщення товару через ряд компаній, кожна з яких додає до нього додаткову цінність.

Повідомлення – це запит на виконання дії, доповнений набором аргументів, які можуть знадобитися при виконанні дії.

ВСТУП

Документи в будь-якій установі, незалежно від способу фіксації та відтворення інформації, проходять і опрацьовуються в установі на єдиних організаційних і правових засадах організації документообігу. Порядок документообігу регламентується інструкцією та регламентами роботи установи, положеннями про структурні підрозділи, посадовими інструкціями. Правильна організація документообігу сприяє оперативному проходженню документів в апараті управління, рівномірному завантаженню підрозділів і посадових осіб, позитивно впливає на управлінський процес загалом. [1].

Актуальність

Моніторинг системи документообігу (СД) був і лишається актуальним, оскільки майже всі компанії запроваджують системи документообігу. Для покращення аналізу данних про помилки та усунення подальшого виникнення таких помилок необхідно зберігати всю інформанію, та для покращення впровадження аналітики помилок зберігати повноту інформації.

Помилки в документообігу, та затягування вирішення помилок в документообізі можуть призвести до великих фінансових втрат. [3]

Існує багато систем документообігу, та не всі підходять для вирішення проблем документообігу в корпоративній компанії.

Саме тому, об’єктом дослідження в роботі є організаційно-технічна система документообігу корпоративної компанії.

Для покращення роботи корпоративної компанії та мінімізації фінансових втрат шляхом виправлення недоліуів уже існуючої системи документообігу, а також прискорення внесення нового функціоналу, зазвичай, здійснюється моніторинг СД. При чому, більшість СД включають у себе моніторинг створення, зміну та видалення існуючих документів.

Предметом дослідження є система моніторингу (СМ) документообігу корпоративної компанії (КК), що планує покращити роботу шляхом вдосконалення уже існуючої системи документообігу.

Основною метою цієї роботи є вдосконалення процесу документообігу корпоративної компанії шляхом автоматизації процесів виявлення і обробки помилок.

Для досягнення мети було сформовано такі задачі:

- дослідити структуру корпоративної компанії, що планує впровадження або вдосконалення існуючої системи моніторингу документообігу;
- проаналізувати методи пошуку рядка в рядку для їх впровадження в програмний продукт для реалізації функції ідентифікації поточних помилок з тими, що вже записані в базі даних системи;
- проаналізувати програмне забезпечення, що використовується в корпоративній компанії та інші існуючі системи моніторингу документообігу;
- дослідити системи виявлення помилок в системі документообігу;
- забезпечити збереження повноти інформації щодо запитів про помилки в документах.
- розробити спеціалізовану систему моніторингу документообігу (ССМД).

1. АНАЛІЗ СИСТЕМИ ДОКУМЕНТООБІГУ КОРПОРАТИВНИХ КОМПАНІЙ

1.1. Основні функції системи електронного документообігу

В даний час ринок програмного забезпечення України та інших країн насичений різними системами автоматизації діловодства, документообігу та інших бізнес-процесів підприємств і установ. Серед відомих СЕД постачальників можна назвати: Справа, БОС-Референт, CompanyMedia, DIRECTUM, DOCUMENTUM, DocsVision, ЕВФРАТ-Документообіг, Optima-Workflow, LanDocs, МОТИВ, Lotsia PDM Plus.

В Україні, в основному, використовуються СЕД таких виробників, як Атлас ДОК, Megapolis.Документообіг, ДОК ПРОФ, АСКОД і FossDoc, або інтеграторів зарубіжного програмного забезпечення, зокрема, на платформі Lotus Notes/Domino від компанії IBM. При цьому, функції, які пропонують різні СЕД своїм користувачам, досить різноманітні.

У першому наближенні ці функції можна розділити на такі категорії:

- зберігання і пошук документів;
- підтримка канцелярії;
- маршрутизація;
- контроль виконання документів;
- аналітичні звіти;
- інформаційна безпека;
- додаткові (специфічні) функції.

Далі надано коротку характеристику найбільш затребуваним функціям із зазначених категорій.

Зберігання і пошук документів

Централізоване зберігання документів – основна мета переходу на електронний документообіг маленьких компаній. У зв'язку з цим компанії варто звернути увагу на постачальника сховища даних, що використовується в СЕД, яка впроваджується.

Для централізованого зберігання документів маленьких компаній можуть використовуватися:

- сховища Lotus Notes/Domino (БОС-Референт, CompanyMedia);
- власні формати зберігання даних (ЕВФРАТ-Документообіг);
- Microsoft SQL Server в різних редакціях (Справа, DIRECTUM, DocsVision, LanDocs та ін.);
- Oracle (Атлас ДОК, ДОК ПРОФ и др.);
- одночасна підтримка MS SQL і Oracle (ЕВФРАТ-Документообіг, Справа, FossDoc та ін.).

Підтримка канцелярії та діловодства

Підтримка роботи канцелярії – важливий компонент СЕД, що орієнтовані на роботу в державних органах і в комерційних організаціях.

До основних "канцелярських" функцій можна віднести:

- представлення документа у виді електронної картки – аналога реєстраційної картки документа;
- підтримка введення документів в систему зі сканера;
- ведення номенклатури справ;
- реєстрація документів, в тому числі тих, що прийшли електронною поштою;
- повний цикл роботи з вхідними/вихідними документами;
- підтримка службових записок;
- робота зі зверненнями громадян;
- робота с заявками;
- ведення журналів реєстрації і обліку паперових оригіналів документів;
- підтримка ієрархічних довідників.

Маршрутизація і контроль виконання документів

Функції цієї категорії користуються попитом як у великих, так і дрібних організаціях і дозволяють управляти документопотоками на підприємстві та контролювати виконання робіт за документами.

До основних функцій даної категорії відносяться:

- проектування маршрутів документів з можливістю послідовно-паралельного їх виконання;
- підтримка різних дій над документами під час маршруту: візування, узгодження, накладення резолюції, підпис і т.п.;
- відправка документів як за типовими, раніше спроектованими, так і за вільними, визначеними користувачем в процесі виконання завдання, маршрутами;
- повідомлення співробітників про надходження до них на виконання нових документів;
- повідомлення про завершення етапів маршрутів;
- підтримка версійності документів (проектів документів);
- автоматичний контроль термінів виконання документів.

Аналітичні звіти

Як правило, звіти в СЕД створюються під конкретного замовника.

Однак існують і загальноприйняті звіти, такі як:

- звіт про поточну зайнятість співробітників;
- звіт про виконання робіт по документам (ретроспективний);
- звіт по прострочених дорученнях.

Інформаційна безпека

Функції зазначеної категорії забезпечують інформаційну безпеку підприємства наступними засобами:

- аутентифікація користувачів системи;
- розподіл прав доступу для співробітників-користувачів СЕД;
- підтримка електронного цифрового підпису документів;
- шифрування листів і документів;
- ведення історії і статистики роботи з документами;
- аудит роботи користувачів в системі.

Існують інші додаткові функції, візуальне оформлення програмного забезпечення для зручного користування програмним забезпеченням. Більшість з таблиць документів мають функції фільтрації документів у

таблицях. Позначення необхідних даних у таблицях.Також можуть мати ще багато користувацьких функцій таких як замітки, калькулятор і т.д. [7]

1.2. Критерії вибору системи документообігу

При виборі СЕД споживачам доводиться шукати компромісне рішення, що по найкраще задовольняє таким критеріям:

- забезпечення необхідної функціональності з можливістю подальшого розширення системи;
- мінімальна сукупна вартість володіння і швидка окупність системи;
- достатній рівень технічної підтримки;
- виробник, що зарекомендував себе, з реальними впровадженнями;
- облік вітчизняної законодавчої бази;

Але, якщо данні програмні продукти з різнх причин не можуть забеспечити роботу з документами на належному рівні, то пишутся самописні програми для роботи з документами та взаємодії з данними компанії.

Прикладом такої компанії є корпоративна компанія, що займається дистрибуцією деякої продукції.

Аналіз СД різних корпоративних компаній, показав, що найчастіше в системі спостерігаються такі особливості і недоліки, як:

- відсутність сповіщення відповідальних осіб про несистемні помилки;
- відсутність збереження несистемних помилок документообігу;
- відсутність збереження рішень помилок документообігу;
- відсутність зворотного зв’язку та контролю заходів, що спрямовані на усунення помилок і їх наслідків;
- відсутність повноти інформації про отримані помилки;
- моніторинг тільки апаратних можливостей СД;
- програмні продукти, що призначені забезпечувати моніторинг усіх систем корпоративної компанії, не отримують інформацію про несистемні помилки;

– Веб-розробка обмежується клієнтською стороною застосування, що призводить до того, що інтерфейс не включає деталі сервера чи браузера.

Проведений аналіз структури, процесів створення та зміни документів і програмного забезпечення однієї з корпоративних компаній виявив низку недоліків СД, які в першу чергу пов’язані з необхідність автоматизувати процес виявлення несистемних помилок.

Електронний документообіг потрібно впроваджувати для вирішення поставлених задач з структуризацією, зберіганням та обробкою документів. При цьому, ЕД повинен мати зрозумілий для користувача інтерфейс.

СЕД розміщає документи в певні папки і прийме на себе всі проблеми, що пов'язані з пошуком, доступом і зберіганням документів. Оптимальним рішенням при цьому може бути такий інтерфейс робочого місця користувача системи, в який інтегровані функції звичайного поштового клієнту (такого як Microsoft Outlook). Однак, якщо на підприємстві існують нижчезазначеніі проблеми, то просте сховище документів їх не вирішить.

Задача автоматизації роботи з документами і бізнес процесами постає, якщо:

– існує великий документопотік вхідних, вихідних і внутрішніх (службових) документів, розгляд яких значно збільшує терміни виконання робіт;
– оперативність прийняття та виконання рішень низька і знижується;
– знаходження винних у порушенні виконавчої дисципліни ускладнює чи затримує операційну діяльність компанії;
– велика кількість форм звітност;
– існує проблема витоку інформації та порушення комерційної таємниці.

1.3. Функції сучасних систем електронного документообігу

Дистрибуція є важливим поняттям маркетингу, де розглядаються проблеми забезпечення оптимального руху товару каналом розподілу до кінцевого споживача. Згідно з сучасною концепцією маркетингу дистрибуція є одним з елементів маркетинг-міксу [12].

До дистрибуції доцільно застосовувати методи управління ланцюгом поставок .

Система дистрибуції – складна економічна система, що об'єднує виробника готової продукції та різних посередників, які на засадах дистриб'юторського договору спільно здійснюють маркетингову, комерційну, логістичну діяльність з переміщення продукції до кінцевого споживача і її продажу відповідно до стратегії суб'єкта господарювання – організатора такої системи з дотриманням встановлених ним умов продажу, цін продажу, стандартів обслуговування і під його контролем.[10]

Система дистрибуції основана на поєднанні в процесах збуту готової продукції таких складових, як :

- стратегія підприємства – організатора системи дистрибуції на ринку;
- партнерство з комерційними посередниками, які на договірній основі об'єднуються в канали розподілу.

Ринок посередників – торгова діяльність організацій, фірм і інших юридичних осіб, що полягає в придбанні товарів і послуг з метою їх перепродажу .РП функціонує як біржі, аукціони, ярмарки, торги.

Предметом продажу може бути як реальний товар, так і документ на товар і послуги.

Ціноутворення має базуватись на єдиних для всіх учасників розподілу продукції підходах і передбачати справедливе і прозоре встановлення не лише роздрібної ціни, але й цін перепродажу у всьому каналі збуту.

Логістика має бути ефективною для забезпечення фізичного руху товару (обслуговування замовлень, транспортування, утримування складів, запасів і забезпечення наявності всього заявленого асортименту товарів).

Аналіз і контроль – насамперед контроль за роздрібними цінами, контроль наявності товарів у місцях продажу, контроль якості подання товару в кожному пункті продажу, контроль і аналіз діяльності партнерів виробничого підприємства з погляду дотримання домовленостей, стандартів обслуговування, демпінгування, недопущення внутрішньосистемної конкуренції, завдавання шкоди іміджу товаровиробника, а також аналіз дій конкурентів.

В цій роботі логíстика розглядається як наука про оптимальне управління інформаційними потоками в економічних адаптивних системах із синергічними зв'язками. Це уявлення про логістику значно відрізняється від оригінального.

Якщо раніше термін описував фізичний рух сировини і товарів, то тепер він включає планування, закупки, транспортування та зберігання. Подальшим розвитком логістики є управління ланцюгом поставок .[13]

Логістичні функції

В організації бізнесу логістика виконує ряд функцій, серед яких виділяють базисні, ключові і допоміжні.

До базисних логістичних функцій відносяться :

- постачання;
- виробництво;
- збут.

До ключових логістичних функцій відносяться:

- підтримка стандартів обслуговування споживача;
- управління закупівлями;
- транспортування;
- управління запасами;
- управління процедурами замовлень;

– управління виробничими процедурами;
– ціноутворення;
– фізичний розподіл.

До допоміжних (що підтримує) логістичних функцій відносяться:
– складування;
– вантажопереробка;
– захисна упаковка;
– підтримка повернення товарів;
– забезпечення запасними частинами і сервісом;
– збір зворотних відходів;
– інформаційна підтримка.[14]

Поряд з поняттям логістичної системи використовується термін логістичний ланцюг (ланцюг поставок). Іноді ланцюг поставок не закінчується на кінцевому споживачы, а додатково охоплює етап переробки і повторного використання матеріалів.

1.4 Оточення програмного забезпечення

Через те, що система документообігу дослыджуваноъ КК побудована у більшості на мові C# і працює під системою windows побудова реалізації системи моніторингу буде здыйснюватись на мові C#.

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників – мов C++,

Delphi, Модула і Smalltalk – C#, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем.

Наприклад: множинне спадкування класів (на відміну від C++).SQL це мова, орієнтована спеціально на реляційні бази даних (РБД). Він виконує велику роботу, яку ви повинні були б робити, якби використовували універсальну мову програмування, наприклад C. [15]

Щоб сформувати РБД на C, необхідно було б почати з нуля.

До числа принципово важливих рішень, які реалізовані корпорацією Microsoft у мові програмування C #, можна віднести такі:

- компонентно-орієнтований підхід до програмування (який характерний і для ідеології Microsoft. NET в цілому);
- властивості як засіб інкапсуляції даних (характерно також в цілому для ООП);
- обробка подій (маються розширення, в тому числі в частині обробки виключень, зокрема, оператор try);
- обробка подій (маються розширення, в тому числі в частині обробки виключень, зокрема – оператор try);
- делегати (delegate – розвиток покажчика на функцію в мовах C і C);
- індексатори (indexer – оператори індексу для звернення до елементів класу-контейнера);
- перевантажені оператори (розвиток ООП);
- оператор foreach (обробка всіх елементів класів-колекцій, аналог Visual Basic);
- механізми boxing і unboxing для перетворення типів;
- атрибути (засіб оперування метаданими в COM-моделі);
- прямокутні масиви (набір елементів з доступом за номером індексу і однаковою кількістю стовпців і рядків).

Об'єктно-орієнтоване програмування – це програмування, засноване на поданні програми у вигляді сукупності взаємодіючих об'єктів, кожен з яких є екземпляром певного класу, а класи є членами певної ієрархії наслідування.

Програмісти спочатку пишуть клас, а на його основі при виконанні програми створюються конкретні об'єкти (екземпляри класів). На основі класів створюються нові класи які розширюють базовий клас і таким чином створюється ієрархія класів.

На думку розробника мови Smalltalk, об'єктно-орієнтований підхід полягає в наступному наборі основних принципів:

- Все є об'єктами.
- Всі дії та розрахунки виконуються шляхом взаємодії (обміну даними) між об'єктами, при якій один об'єкт потребує, щоб інший об'єкт виконав деяку дію. Об'єкти взаємодіють, надсилаючи і отримуючи повідомлення.
- Кожен об'єкт має незалежну пам'ять, яка складається з інших об'єктів.
- Кожен об'єкт є представником (екземпляром, примірником) класу, який виражає загальні властивості об'єктів.
- У класі поведінка (функціональність) об'єкта визначена. Таким чином усі об'єкти, які є екземплярами одного класу, можуть виконувати однакові дії.
- Класи організовані в єдину деревоподібну структуру з загальним корінням, яка називається ієрархією успадкування.
- Пам'ять та поведінка, зв'язані з екземплярами деякого класу, автоматично доступні будь-якому класу, розташованому нижче в ієрархічному дереві.[20]

Таким чином, програма являє собою набір об'єктів, що мають стан та поведінку. Об'єкти взаємодіють використовуючи повідомлення. Будується ієрархія об'єктів: програма в цілому це об'єкт, для виконання своїх функцій вона звертається до об'єктів що містяться у ньому, які у свою чергу виконують запит шляхом звернення до інших об'єктів програми. Звісно, щоб уникнути безкінечної рекурсії у зверненнях, на якомусь етапі об'єкт трансформує запит

у повідомлення до стандартних системних об'єктів, що даються мовою та середовищем програмування.

Стійкість та керованість системи забезпечуються за рахунок чіткого розподілення відповідальності об'єктів (за кожну дію відповідає певний об'єкт), однозначного означення інтерфейсів міжоб'єктної взаємодії та повної ізольованості внутрішньої структури об'єкта від зовнішнього середовища (інкапсуляції).

Для розробки БД буде використовуватися MS SQL Server.

Microsoft SQL Server – комерційнасистема керування базами даних корпорації Microsoft.

Мова, що використовується для запитів – Transact-SQL, створена спільно Microsoft та Sybase.

Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Базовий код MS SQL Server (до версії 7.0) ґрунтувався на коді Sybase SQL Server Microsoft SQL Server як мову запитів використовує версію SQL, що отримала назву Transact-SQL (T-SQL), яка є реалізацією SQL-92 (стандарт ISO для SQL) з багатьма розширеннями.

T-SQL дозволяє використовувати додатковий синтаксис процедур, що зберігаються і забезпечує підтримку транзакцій (взаємодія бази даних з керуючим застосунком). Microsoft SQL Server та Sybase ASE для взаємодії з мережею використовують протокол рівня застосунка під назвою Tabular Data Stream (TDS, протокол передачі табличних даних).

Microsoft SQL Server також підтримує Open Database Connectivity (ODBC) – інтерфейс взаємодії застосунків з СУБД.

Версія SQL Server 2005 надає можливість підключення користувачів через веб-сервер-сервіси, що використовують протокол SOAP. Це дозволяє клієнтським програмам, які не призначені для Windows, кроссплатформенно з'єднуватися з SQL Server. Microsoft, а також випустила сертифікований

драйвер JDBC, що дозволяє застосункам під керування Java (таким як BEA і IBM Websphere) з'єднуватися з Microsoft SQL Server 2000 і 2005.

SQL Server підтримує дзеркалювання та кластеризацію баз даних.

Кластер серверу SQL – це сукупність однаково конфігурованих серверів; така схема допомагає розподілити робоче навантаження між декількома серверами. Усі сервери мають одне віртуальне ім'я, а дані розподіляються за IP-адресами машин кластеру протягом робочого циклу. Також у разі відмови або збою на одному з серверів кластеру доступне автоматичне перенесення навантаження на інший сервер.

SQL Server підтримує надлишкове дублювання даних за трьома сценаріями :

- Знімок: виконується «знімок» бази даних, який сервер відправляє одержувачам.
- Історія змін: всі зміни бази даних безперервно передаються користувачам.
- Синхронізація з іншими серверами: бази даних декількох серверів синхронізуються між собою. Зміни усіх баз даних відбуваються незалежно на кожному сервері, а під час синхронізації відбувається звірка даних. Дублювання такого типу передбачає можливість вирішення протиріч між базами даних.

SQL Server 2005 має вбудовану підтримку .NET Framework. Завдяки цьому, процедури бази даних, що зберігаються, можуть бути написані на будь-якій мові платформи .NET з використанням повного набору бібліотек, доступних для .NET Framework. На відміну від інших процесів, .NET Framework виділяє додаткову пам'ять і будує засоби керування SQL Server, не використовуючи вбудовані засоби Windows. Це підвищує продуктивність порівняно із загальними алгоритмами Windows, оскільки алгоритми розподілу ресурсів спеціально налагоджені для використання у структурах SQL Server.[19]

В якості фреймворка для реалізації обраний ASP.NET Core 2.0 (з використанням мови C #).

Серидовище розробки. Microsoft Visual Studio – серія продуктів, які включають інтегроване середовище розробки програмного забезпечення та інші інструментальні засоби. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби в рідному і керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

Компоненти

Visual Studio включає один або декілька з компонентів:

- Visual Basic .NET, а до його появи – Visual Basic;
- Visual C++;
- Visual C#;
- Visual F# (входить до складу Visual Studio 2010);
- Visual Studio Debugger.

Багато варіантів постачання також включають:

- Microsoft SQL Server;
- MSDE Visual Source Safe – файл-серверна система управління версіями;

У минулому до складу Visual Studio також входили продукти:

- Visual InterDev;
- Visual J++;
- Visual J#;
- Visual FoxPro;
- Visual Source Safe – файл-серверна система управління версіями.

В роботі використовується версія Visual Studio 2019 Community.

ASP.NET Core – вільне та відкрите програмне забезпечення каркасу веб застосунків, з продуктивністю вищою ніж у ASP.NET, розроблена

корпорацією Microsoft і співтовариством. Це модульна структура, яка працює як на повній платформі .NET Framework, так і на платформі .NET Core.

Фреймворк являє собою повний перепис, який об'єднує раніше окремі ASP.NET MVC та ASP.NET Web API у єдину модель.

Не зважаючи на те, що це є новим фреймворком, побудованим на новому веб-стеку, ASP.NET Core має високу ступінь сумісності концепцій з ASP.NET MVC, який об'єднує функціональність MVC, Web API та Web Pages.

В попередніх версіях платформи дані технології реалізовані окремо і тому містять багато дублюючої функціональності. Тепер це об'єднано в одну програмну модель ASP.NET Core MVC.

Програми ASP.NET Core підтримують програмні версії. Програми, що працюють на одному комп'ютері, можуть орієнтуватися на різні версії ASP.NET Core, що не можливо в попередніх версіях ASP.NET Core.[19]

Особливості ASP.NET Core :

- Відсутність досвіду розробника (до прикладу, компіляція неперервна, отже розробник не повинен додатково використовувати команду компіляції).
- Модульна структура розподіляється як NuGet пакунки.
- Cloud-optimized runtime (оптимізована для Інтернету).
- Хост-агностик за допомогою Відкритого Веб-Інтерфейсу для .Net (OWIN) підтримки, що працює в IIS або в автономному режимі.
- Єдина історія для створення веб UI і веб APIs (тобто обидва ті самі).
- Система створення конфігурації середовища на основі хмар.
- Легкий і модульний HTTP запит.
- Створення та запуск крос-платформних додатків ASP.NET Core у Windows, Mac та Linux.
- Відкрите джерело та орієнтоване на спільноту.
- Пряме прикріплення версії додатка при націлюванні на .NET Core.

В якості стека технологій для роботи з базою данних було обрано Entity Framework Core.

Entity Framework (EF) Core – це розширювана, відкрита версія та міжплатформна версія популярної технології доступу до даних Entity Framework.

EF Core може слугувати об'єктно-реляційним картографом (O / RM), дозволяючи розробникам .NET працювати з базою даних за допомогою .NET-об'єктів і виключати необхідність більшості коду доступу до даних, який їм зазвичай потрібно писати.

EF Core підтримує велику кількість баз даних. Спочатку з найпершої версії Entity Framework підтримував підхід Database First, який дозволяв по готовій базі даних згенерувати модель edmx. Потім ця модель використовувалася для підключення до бази даних. Пізніше був доданий підхід Model First.

Model First дозволяв створити вручну за допомогою візуального редактора модель edmx, і по ній створити базу даних. Починаючи з 5.0 кращим підходом стає Code First. Його суть полягає в тому, що спочатку пишеться код моделі на C #, а потім по ньому генерується база даних. При цьому модель edmx вже не використовується.

Рішення реалізовано за допомогою паттерна MVC , та веб – сервісами на основі Web Api. Концепція паттерна (Шаблону) MVC (model – view – controller) та передбачає поділ програми на три компонента:

Контролер (controller) представляє клас, що забезпечує зв'язок між користувачем і системою, поданням і сховищем даних. Він отримує дані, що вводяться користувачем, і обробляє їх. І, в залежності від результатів обробки, відправляє користувачеві певний висновок (у вигляді подання).

Подання (view) – це власне візуальна частина або призначений для користувача інтерфейс програми. Як правило, html-сторінка, яку користувач бачить, зайшовши на сайт.

Модель (model) представляє клас, що описує логіку використовуваних даних.

При такому підході модель є незалежним компонентом – будь-які зміни контролера або подання не зачіпають модель. контролер і уявлення є відносно незалежними компонентами, і нерідко їх можна змінювати незалежно один від одного.

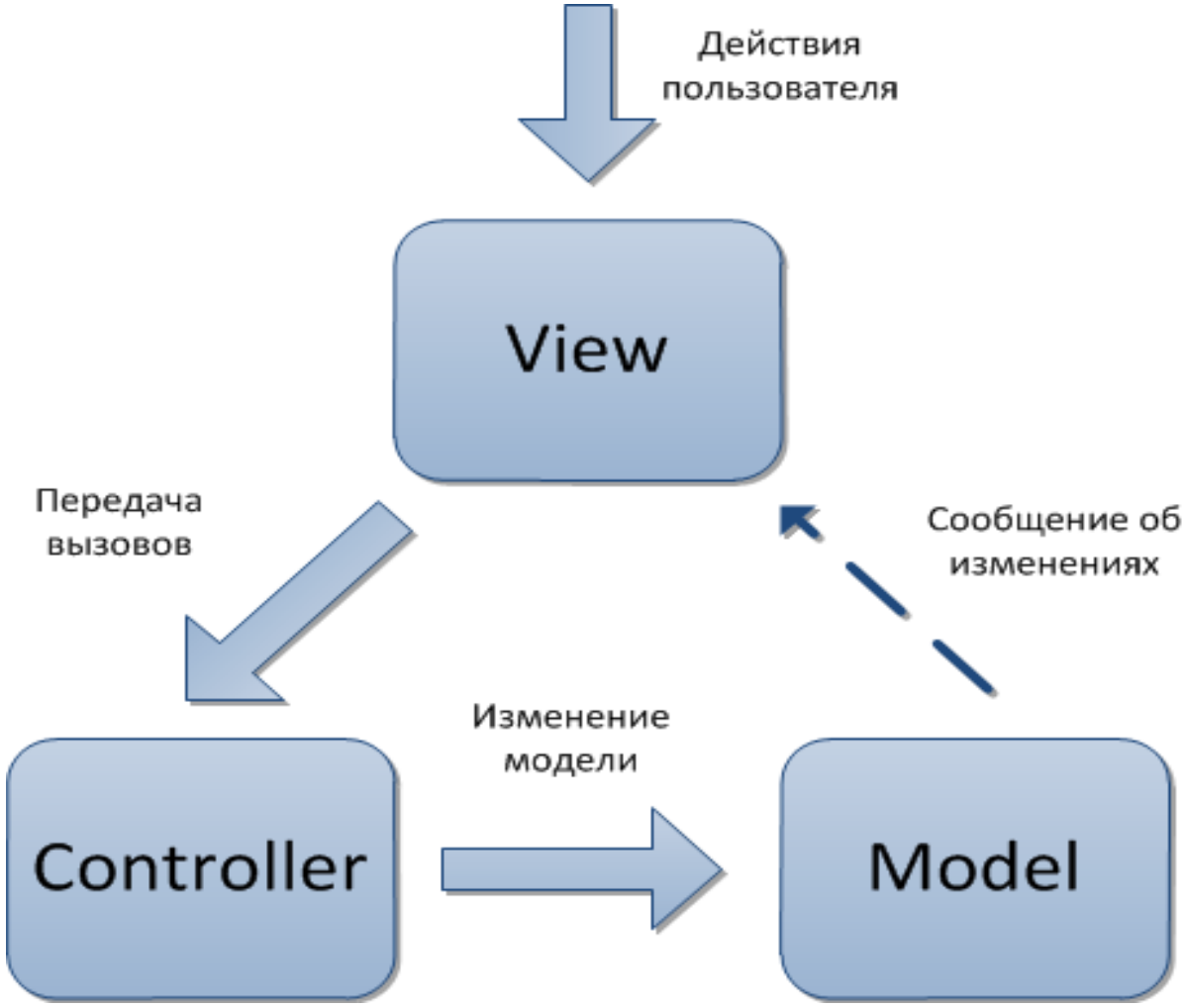

Рис.1.1 Конструкція MVC

Web Api – Application Programming Interface, або інтерфейс для програмування додатків.[18]

API може передавати інформацію, що представлена в форматі, відмінному від стандартного HTML. Завдяки чому ним зручно користуватися при написанні власних програм. Сторонні загальнодоступні API найчастіше віддають дані в одному з двох форматів: XML або JSON.

REST визначає ряд архітектурних принципів проектування Web-сервісів, орієнтованих на системні ресурси, включаючи способи обробки і передачі станів ресурсів по HTTP різноманітними клієнтськими додатками, написаними на різних мовах програмування.

Реалізація Web-сервісів REST слід чотирьом базовим принципам проектування:

- Явне використання HTTP-методів.
- Незбереження стану.
- Надання URI, аналогічних структурі каталогів.
- Передача даних в XML, JavaScript Object Notation (JSON) або в обох форматах.

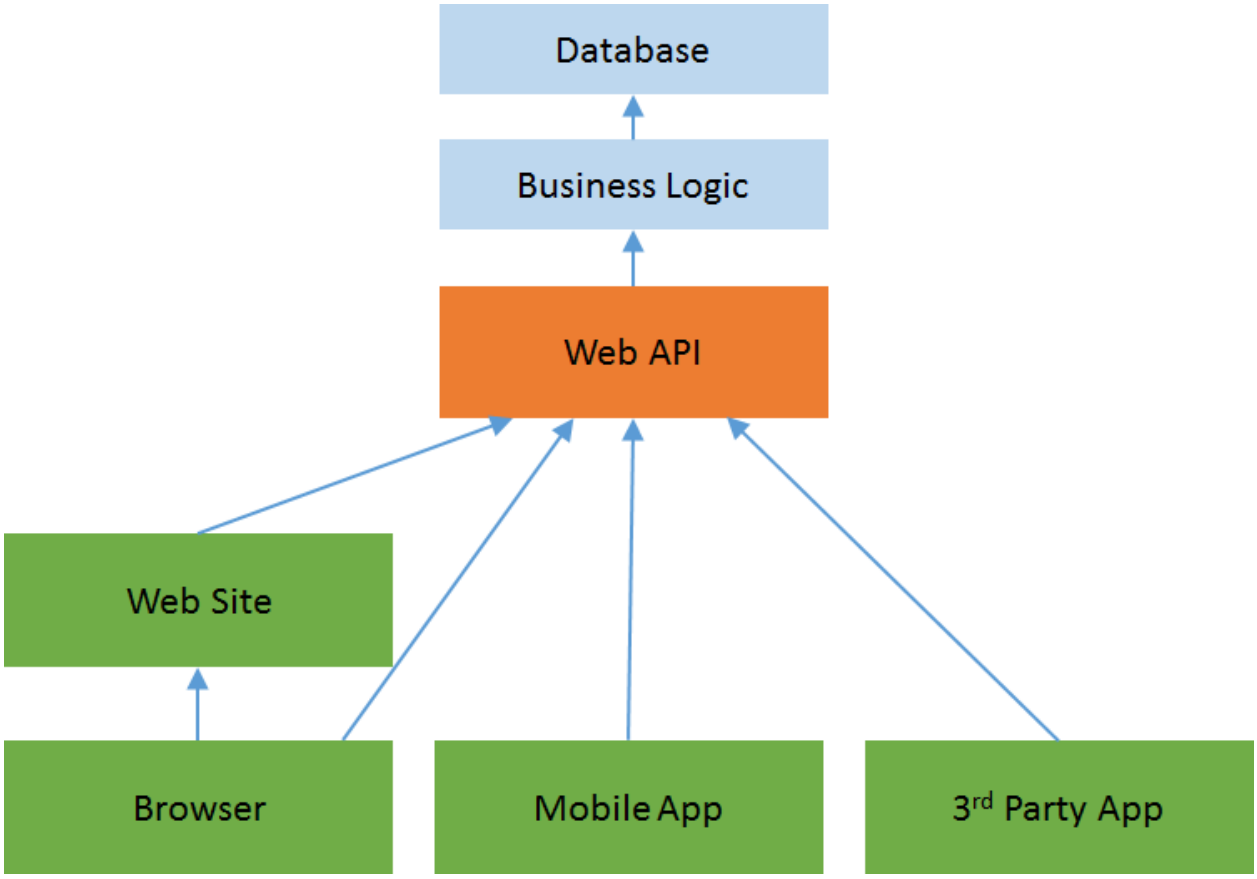

Рис.1.2 Конструкція сервісів

ASP.NET Core – це технологія від компанії Microsoft, призначену для створення різного роду веб-додатків: від невеликих веб-сайтів до великих веб-порталів і веб-сервісів.

ASP.NET Core може працювати поверх крос-платформної середовища. NET Core, яка може бути розгорнута на основних популярних операційних системах: Windows, Mac OS X, Linux. І таким чином, за допомогою ASP. NET Core ми можемо створювати крос-платформні додатки. І хоча Windows як середовище для розробки і розгортання програми досі превалює, але тепер вже ми не обмежені тільки цією операційною системою. Тобто є можливість запускати веб-додатки не тільки на ОС Windows, але і на Linux і Mac OS. А для розгортання веб-додатки можна використовувати традиційний IIS, або крос-платформний веб-сервер Kestrel.

ASP. NET Core надає наступні переваги:

- Єдине рішення для створення призначеного для користувача веб-інтерфейсу і веб-API.
- Інтеграція сучасних клієнтських платформ і робочих процесів розробки.
- Хмарна система конфігурації на основі середовища.
- Вбудоване введення залежностей.
- Спрощений високопродуктивний модульний конвеєр HTTP-запитів.
- Можливість розміщення в IIS, Nginx, Apache, Docker або у власному процесі.
- Паралельне управління версіями додатка, орієнтоване на .NET Core.
- Інструментарій, що спрощує процес сучасної веб-розробки.
- Можливість побудови та запуску в ОС Windows, macOS і Linux.
- Відкритий вихідний код і орієнтація на співтовариство.

ADO.NET Entity Framework (EF) – об'єктно-орієнтована технологія доступу до даних, є object-relational mapping (ORM) рішенням для .NET Framework від Microsoft. Надає можливість взаємодії з об'єктами як за допомогою LINQ у вигляді LINQ to Entities, так і з використанням Entity SQL. Для полегшення побудови web-рішень використовується як ADO.NET Data Services (Astoria), так і зв'язка з Windows Communication Foundation і Windows

Presentation Foundation, що дозволяє будувати багаторівневі додатки, реалізуючи один з шаблонів проектування MVC, MVP або MVVM.
Система буде розгортатися на IIS.

IIS було вибрано тому, що всі програмні продукти КК розміщені на IIS. Саме тому, це найзруніший сервер для розгортання веб додатків на мові C#. Також IIS був вибраний через екосистему Windows, та через те що більшість програмних продуктів корпоративної компанії створені спеціально під платформу Windows.

IIS (Internet Information Services, до версії 5.1 – Internet Information Server) – набір серверів для декількох служб Інтернету від компанії Майкрософт.

IIS поширюється з операційними системами родини Windows NT. Основний компонент IIS – веб-сервер, який дозволяє розміщувати в Інтернеті сайти.

IIS підтримує протоколи HTTP, HTTPS, FTP, POP3, SMTP, NNTP.

IIS другий за популярністю веб-сервер за кількістю сайтів, після Apache HTTP Server. За даними компанії Netcraft на 11.10.2007 понад 37.13% сайтів обслуговується веб-сервером IIS.

Перша версія IIS була випущена, як додатковий набір Інтернет сервісів для Windows NT 3.51. IIS 2.0 з'явився в операційній системі Windows NT 4.0. В IIS 3.0 вже була представлена Active Server Pages, технологія динамічного формування веб-сторінок. В IIS 4.0 відмовились від підтримки протоколу Gopher і розповсюджувався окремо на диску «Option Pack» CD-ROM for Windows NT.

Версії IIS, які підтримуються в цей час: 7.0 для Windows Vista, 6.0 для Windows Server 2003 та 5.1 для Windows XP Professional. Windows XP має обмежену версію IIS 5.1, яка підтримує тільки 10 одночасних підключень й один веб-сайт. В IIS 6.0 підтримує IPv6. FastCGI модуль також доступний для IIS5.1, IIS6 та IIS7.

У Windows Vista IIS 7.0 не встановлюється за замовчуванням, але можна вибрати як додатковий компонент для встановлення. IIS 7.0 на Vista не обмежує кількість підключень, але є обмеження при одночасному виконнані декількох запитів.

Ранні версії IIS містили багато вразливостей, головна з яких була CA-2001-19, вона призводила до Code Red черв'яка. 7.0 версія поки що не має задокументованих вразливостей..

В IIS 6.0 у Microsoft вирішили змінити поведінку попередньо встановлених ISAP обробників, багато з яких призводили до вразливостей у версіях 4.0 та 5.0, таким чином зменшили поверхню атаки ISS. Також була додана функція «Web Service Extensions», яка запобігає запуску будь-яких програм без явного дозволу адміністратора.

В IIS 7.0 компоненти розділені на модулі, таким чином встановлюються тільки необхідні компоненти, це також зменшує поверхню атаки. Ще одна функція додана для безпеки – це URLFiltering, з її допомогою відхиляються запити з підозрілих URL адрес, правила визначення підозрілих URL формуються користувачем.

У версії 5.1 та нижче усі веб-сайти запускалися в процесі та під системним (System) обліковим записом, стандартний обліковий запис Windows з розширеними правами.

У версії 6.0 усі процеси, що обробляють запити, запускаються під обліковим записом NETWORK SERVICE, який має значно менше привілеїв. IIS 6.0 має новий HTTP стек у ядрі(http.sys) з суворішим синтаксичним аналізатором HTTP запитів та кешем для відповідей з статичним та динамічним контентом.

IIS 5.0 та вищі підтримують такі механізми аутентифікації:

- Базова аутентифікація(Basic access authentication) – ім'я та пароль передаються в мережі відкритим текстом.
- Стисла аутентифікація (Digest access authentication) – пароль обробляється хеш-функцією перед відправленням по мережі, це

унеможливлює відтворення пароля у разі перехоплення зловмисником.

- Інтегрована аутентифікація Windows (Integrated Windows Authentication) – виконується спроба аутентифікації на сервері з тими обліковими даними, під якими працює браузер користувача.
- .NET Passport аутентифікація

Вперше з'явилась у Windows Vista а також включена до складу Windows Server 2008.

IIS 7.0 має модульну архітектуру. На відміну від монолітних серверів, які надають всі свої сервіси, IIS 7 має ядро двигуна веб-сервера. Модулі зі специфічною функціональністю можуть бути додані до двигуна. Переваги такої архітектури в тому, що тільки необхідні функції можуть бути увімкнені, а функціональність може бути розширена за рахунок використання спеціальних модулів.

IIS 7 розповсюджується з невеликою кількістю модулів, але Microsoft обіцяє зробити інші модулі доступними онлайн. Наступний набір модулів розповсюджується з сервером:

1. HTTP модулі
2. Модулі безпеки
3. Модулі контенту
4. Модулі стиснення
5. Модулі кешування
6. Модулі для протоколювання та діагностики

Значна зміна в порівнянні з попередніми версіями IIS полягає в тому що вся конфігураційна інформація зберігається виключно в конфігураційних XML файлах, а не в метабазах. Сервер має глобальний кофігураційний файл, який містить налаштування за замовчуванням, і кожна коренева віртуальна веб директорія (так само, як і її піддиректорії) можуть містити web.config файл, в якому глобальні налаштування розширюються або заміщуються. Внесені зміни в ці файли починають діяти відразу.

IIS 7 має повністю змінений адміністративний інтерфейс, який використовує такі переваги сучасної MMC, як панель задач та асинхронні операції.

Налаштування ASP.NET ще більше інтегроване в адміністративний інтерфейс [17].

1.5 Аналізи алгоритмів пошуку в рядку

Алгоритм прямого пошуку рядка.

Нехай задано масив **a** із **N** елементів та масив **b** із **M** елементів, які називаються відповідно базовим рядком та підрядком або образом, причому $0<M\leq N$:

a : array [1..N] of basetype ; b : array [1..M] of basetype **;**

Пошук рядка передбачає встановлення першого входження образа **b** в базовий рядок **a**.

Прямий пошук полягає в повторюваному порівнянні елементів двох масивів. У випадку неспівпадання чергової пари елементів образ зсувається на одну позицію вздовж базового масиву і процес порівняння проводиться знову починаючи з першого елемента шуканого підрядка.

Процес порівняння елементів може припинитися при виконанні однієї із двох умов*:*

1) всі елементи образа співпали з відповідними елементами бази – це означає, що позиція входження встановлена ;

2) після чергового неспівпадання елементів та зсуву образа відбувся вихід підрядка за межі бази – це означає, що шуканий підрядок не входить в базовий.

Якщо позначити положення початку образа відносно бази через індекс **k**, а біжучий індекс по образу – через **j**, то ці умови можна записати у вигляді

диз’юнкції логічних виразів **(j>M) or (k>N-M+1)**. Позначимо також біжучий індекс порівнюваних елементів бази через **i**.

Таким чином програмна реалізація прямого пошуку підрядка в рядку матиме вигляд процедури :

```
Var i, j, k : integer;
Begin
j:=1; k:=1;
while (j<=M) and (k<=N-M+1) do
begin
j:=1; i:=k;
while (j<=M) and (a[i]=b[j]) do
begin i:=i+1; j:=j+1 end;
k:=k+1
end;
if j>M then writeln (‘номер позиції·входження’, k-1)
else writeln ('не має входження образу в базу’);
End.[4]
```

Алгоритм Кнута-Морріса-Пратта (КМП) – один із алгоритмів пошуку рядка, що шукає входження слова `W` у рядку `S`, використовуючи спостереження, що коли відбувається невідповідність, то слово містить у собі достатньо інформації для того, щоб визначити, де наступне входження може початися, таким чином пропускаючи кількаразову перевірку попередньо порівняних символів.

Алгоритм повинен знайти початковий індекс `m` рядка `W[]` в рядку `S[]`. Найпростіший алгоритм пробігає по всьому рядку `S[m]`, де `m` – індекс.

Якщо індекс `m` досягне кінця рядка, то `W[]` не знайдено і алгоритм поверне результат «fail». На кожній позиції перевіряється рівність елемента на позиції `m` з `S[]` й елемента на першій позиції з `W[]`, тобто `S[m] =? W[0]`.

Якщо індекси рівні, то алгоритм перевіряє наступні відповідні елементи в рядках за індексом `i`.

Алгоритм перевіряє всі вирази `S[m+i] =? W[i]`. Якщо всі елементи з `W` знайдені, то алгоритм поверне позицію `m`.

Зазвичай, пробна перевірка відкидає можливість збігу.

Якщо рядки складаються з рівномірно розподілених елементів, то шанс, що перші елементи дорівнюють один одному, буде 1 до 26. Отже, в більшості випадків пробна перевірка відкидатиме початкові елементи. Шанс, що перші два елементи будуть рівними, дорівнює 1 до 26^2 (тобто, 1 до 676). Тобто, якщо елементи рівномірно розподілені, очікувана складність пошуку в рядку `S[]` довжини *k* буде порядку *k* порівнянь або *O*(*k*).

Якщо `S[]` має 1.000.000.000 елементів і `W[]` має 1000 елементів, то пошук рядка займе приблизно 1.000.000.000 порівнянь. Проте очікувана продуктивність не є гарантованою.

Якщо рядки не випадкові, то на кожному кроці `m` може знадобитися багато порівнянь. В найгіршому випадку два рядки збігаються майже за всіма літерами. Якщо рядок `S[]` має 1.000.000.000 елементів, що рівні *A* і рядок `W[]` складається з 999 елементів *A* і останній елемент *B*. Тоді найпростіший алгоритм на кожному кроці виконуватиме 1000 перевірок, а всіх перевірок буде 1 трильйон. Отже, якщо довжина `W[]` – *n*, то в найгіршому випадку складність дорівнюватиме *O*(*k*·*n*).

Алгоритм КМП має кращий показник продуктивності у найгіршому випадку. КМП витрачає небагато часу (за порядком розміру `W[]`, *O*(*n*)) на попереднє обчислення таблиці, і потім використовує таблицю для швидкого пошуку рядка за час *O*(*k*).

Швидкодія алгоритму пошуку.

Припускаючи існування таблиці `T`, пошукова складова алгоритму КМП має складність O(k), де `k` – це довжина `S`. За винятком сталих витрат на виклик

функції всі обчислення виконуються циклом **while**, обчислимо граничну кількість ітерацій циклу; для цього спочатку зробимо спостереження про природу T.

За визначенням вона побудована так, що якщо частковий збіг, що почався в $S[m]$, провалився під час порівняння $S[m + i]$ з $W[i]$, тоді наступний можлививй збіг має початись в $S[m + (i – T[i])]$.

Наступний можливий збіг може трапитись лише на більшому індексі ніж m, з цього випливає, що $T[i] < i$.

Знаючи цей факт, покажемо, що цикл може виконатись не більше $2k$ разів.

Для кожної ітерації, виконується одна з двох гілок тіла циклу. У першій гілці збільшується i та не змінюється m, так що $m + i$ індекс символу S, що ми розглядаємо збільшується.

Друга гілка додає $i – T[i]$ до m, і це завжди додатне число.

Отже, збільшується початок поточного збігу m.

Тепер, цикл завершується, якщо $m + i = k$.

З того що кожна гілка циклу може бути відвідана не більш, ніж k разів, бо вони збільшують відповідно або $m + i$ або m, і $m \leq m + i$: якщо $m = k$, тоді напевно $m + i \geq k$, з того що воно збільшується щонайбільше на одиницю кожного разу, ми мали мати $m + i = k$ в якийсь момент у минулому, яким би способом ми не просувались.

Так, як цикл виконується не більш, ніж $2k$ разів, ми показали, що складність алгоритму пошуку $O(k)$.

Далі надано інший спосіб розгляду часу виконання.

Починаємо зіставляти W і S в позиціях і та р, якщо W існує як підрядок S в р, тоді W[0 до m] == S[p до p+m].

За умови успіху (W[i] == S[p+i]), ми збільшуємо і на 1 (i++).

При невдачі (W[i] != S[p+i]), вказівник у тексті не змінюється, а вказівник в шуканому слові відкочується на певну кількість символів (i = T[i], де T це таблиця переходів), зіставляються W[T[i]] з S[p+i].

Найбільший відкіт обмежений і, тобто, для будь-якого незбігу, ми можемо відкотитись щонайбільше на наш поступ до невдачі. Звідси й випливає час 2k.[5]

Алгоритм пошуку рядка Бойера-Мура, – ефективний алгоритм пошуку рядка, який є еталоном при практичних дослідженнях алгоритмів пошуку рядка. Перевага цього алгоритму в тому, що ціною деякої кількості попередніх обчислень над зразком або шаблоном (але не над рядком, в якому ведеться пошук) шаблон порівнюється з вихідним текстом не у всіх позиціях. Чстина перевірок пропускаються як такі, що не дадуть результату.

Цей алгоритм швидко працює у ситуаціях коли зразок набагато коротший від тексту пошуку, або коли відбувається пошук в декількох документах. Зазвичай, чим довше зразок, тим швидше працює алгоритм.

Загальна оцінка обчислювальної складності алгоритму дорівнює $O(|haystack|+|needle|+|\Sigma|)$ на неперіодичних шаблонах і $O(|haystack|\cdot|needle|+|\Sigma|)$ на періодичних, де *haystack* – рядок, в якому виконується пошук, *needl* – шаблон пошуку, Σ – алфавіт, на якому проводиться порівняння.

У 1991 році Коул показав , що на неперіодичних шаблонах за повний прохід по рядку алгоритм зробить не більше $3\cdot|haystack|$ порівнян.

Алгоритм базується на трьох ідеях.

Сканування зліва направо, порівняння справа наліво.

Поєднується початок тексту (рядки) і шаблону, перевірка починається з останнього символу шаблону. Якщо символи збігаються, проводиться порівняння передостаннього символу шаблону і т. д. Якщо всі символи шаблону збіглися з накладеними символами рядка, значить, підрядок знайдений, і пошук закінчено.

Якщо ж якийсь символ шаблону не збігається з відповідним символом рядка, шаблон зсувається на кілька символів вправо, і перевірка знову починається з останнього символу.

Ці «декілька», згадані в попередньому абзаці, обчислюються за двома евристиками .

Евристика стоп-символу.

Припустимо, що ми проводимо пошук слова «колокол».

Перша ж буква не збіглася – «к» (назвемо цю букву *стоп-символом*). Тоді можна зсунути шаблон вправо до останньої його букви «к».

Стрічка: * * * * * * к * * * * * **

Шаблон: к о л о к о л

Наступний крок: к о л о к о л

Якщо стоп-символу в шаблоні взагалі немає, шаблон зміщується за цей стоп-символ.

Стрічка: * * * * * а л * * * * * * * *

Шаблон: к о л о к о л

Наступний крок: к о л о к о л

В даному випадку стоп-символ – «а», і шаблон зсувається так, щоб він виявився прямо за цією буквою.

В алгоритмі Бойера-Мура евристика стоп-символу взагалі не дивиться на співпавший суфікс, так що перша буква шаблону («к») опиниться під «л», і буде проведена одна завідома холоста перевірка.

Якщо стоп-символ «к» опинився за іншою буквою «к», евристика стоп-символу не працює.

Стрічка: * * * * к к о л * * * * *

Шаблон: к о л о к о л

Наступний крок: к о л о к о л ?????

У таких ситуаціях виручає третя ідея АБМ – евристика співпавшого суфікса.

*Евристика співпавшого суфікса***.**

Якщо при порівнянні рядка і шаблону збіглося один або більше символів, шаблон зсувається в залежності від того, який суфікс збігся.

Стрічка: * * т о к о л * * * * *

Шаблон: к о л о к о л

Наступний крок: к о л о к о л

В даному випадку збігся суфікс «ок», і шаблон зсувається вправо до найближчого «окол». Якщо підрядка «окол» в шаблоні більше немає, але він починається на «кол», зрушується до «кол», і т. д.

Обидві евристики вимагають попередніх обчислень – залежно від шаблону пошуку заповнюються дві таблиці.

Таблиця стоп-символів за розміром відповідає алфавіту (наприклад: якщо алфавіт складається з 256 символів, то її довжина 256); таблиця суфіксів – шуканому шаблону.

Саме через це алгоритм Бойера-Мура не враховує співпавший суфікс і неспівпавший символ одночасно – це вимагало б занадто багато попередніх обчислень.

Опишемо докладніше обидві таблиці.

Таблиця стоп-символів

У таблиці стоп-символів вказується остання позиція в *needle* (виключаючи останню букву) кожного з символів алфавіту. Для всіх символів, що не увійшли в needle, пишемо 0 (для нумерації з 0 – відповідно, −1).

Наприклад:, якщо needle=«abcdadcd», таблиця стоп-символів буде виглядати так.

Символ	a	b	c	d	[всі інші]
Остання позиція	5	2	7	6	0

Для стоп-символу «d» остання позиція буде 6, а не 8 – остання буква не враховується. Це відома помилка, що приводить до неоптимальності. Для АБМ вона не фатальна («витягує» евристика суфікса), але фатальна для спрощеної версії АБМ – алгоритму Хорспула.

Якщо розбіжність сталася на позиції `i`, а стоп-символ `c`, то зсув буде `i-StopTable[c]` [6].

Алгоритм Рабіна-Карпа – алгоритм пошуку рядка.

Алгоритм показує високу продуктивність на практиці, а також дозволяє узагальнення на інші споріднені задачі.

Ідея алгоритму полягає в заміні текстових рядків числами, порівняння яких можна виконувати значно швидше.

Алгоритм рідко використовується для пошуку одиночного шаблону, але має значну теоретичну важливість і дуже ефективний в пошуку збігів множинних шаблонів однакової довжини.

Для тексту довжини n і шаблону довжини m його середнє і кращий час виконання одно O(n) при правильному виборі хеш-функції (дивіться нижче), але в гіршому випадку він має ефективність O(nm), що є однією з причин того, чому він не дуже широко використовується.

Для додатків, в яких допустимі помилкові спрацьовування при пошуку, тобто, коли деякі зі знайдених входжень шаблону насправді можуть не відповідати шаблоном, алгоритм Рабина-Карпа працює за гарантований час O(n) і при відповідному виборі рандомизированной хеш-функції ймовірність помилки можна зробити дуже малою.

Також алгоритм має унікальну особливість знаходити будь-яку із заданих k рядків однакової довжини в середньому (при правильному виборі хеш-функції) за час O (n) незалежно від розміру k.

Одне з найпростіших практичних застосувань алгоритму Рабіна-Карпа полягає у визначенні плагіату.

Для усунення чутливості алгоритму до невеликих розбіжностей можна ігнорувати деталі, такі як регістр або пунктуація, за допомогою їх видалення. Оскільки кількість рядків, які ми шукаємо, k, дуже велика, звичайні алгоритми пошуку одиночних рядків стають неефективними. Замість того, щоб використовувати більш розумний пропуск, алгоритм Рабина-Карпа

намагається прискорити перевірку еквівалентності зразка з підрядками в тексті, використовуючи хеш-функцію.

Хеш-функція – це функція, яка перетворює кожну рядок в числове значення, зване хеш-значенням (хеш); наприклад, ми можемо мати хеш від рядка «hello» рівним 5.

Алгоритм використовує той факт, що якщо два рядки однакові, то і їх хеш-значення також однакові. Таким чином, все що нам потрібно, це порахувати хеш-значення шуканої підрядка і потім знайти підрядок з таким же хеш-значенням.

Однак існують дві проблеми, пов'язані з цим.

Перша проблема полягає в тому, що, так як існує дуже багато різних рядків, між двома різними рядками може статися колізія – збіг їх хешів. У таких випадках необхідно посимвольний перевіряти збіг самих подстрок, що займає досить багато часу, якщо дані подстроки мають велику довжину (цю перевірку робити не потрібно, якщо додаток допускає помилкові спрацьовування). При використанні досить хороших хеш-функцій колізії трапляються вкрай рідко, і в результаті середній час пошуку виявляється невеликим.

Приклад алгоритму (вихідного коду програми):

```
function RabinKarp(string s[1..n], string sub[1..m])
    hsub := hash(sub[1..m])
    hs := hash(s[1..m])
    for i from 1 to (n-m+1)
        if hs = hsub
            if s[i..i+m-1] = sub
                return i
        hs := hash(s[i+1..i+m])
    return not found
```

Рядки 2, 3, і 6 витрачають для виконання час Omega (m) кожна. Однак рядки 2 і 3 виконуються тільки один раз, а рядок 6 виконується тільки коли хеш-

значення збігаються, що відбувається нечасто. Рядок 5 виконується n раз, але завжди вимагає постійного часу.

Друга проблема полягає в перераховування хешу.

При наївному перерахунку хеш-значення підрядка s [i + 1..i + m] витрачається час O (m). І, так як це робиться в кожному циклі, алгоритм буде витрачати час Omega (mn), тобто таке ж, яке витрачають і найбільш прості алгоритми. Метод вирішення даної проблеми полягає в припущенні того, що змінна hs вже містить хеш-значення підрядка s [i..i + m-1]. Якщо використовувати його для підрахунку наступного хеш-значення за постійний час, тоді проблема буде вирішена.

Це досягається використанням так званого кільцевого хешу. Найпростішим прикладом кільцевого хешу є додавання значень кожного наступного символу в підрядку і подальше використання цієї формули для підрахунку кожного наступного хеш-значення за фіксований час:

$s[i+1..i+m] = s[i..i+m-1] – s[i] + s[i+m]$

Така формула не дає ніяких гарантій нечастого виникнення колізій, і дійсно нескладно переконатися, що в більшості додатків при її використанні вираз в 6 рядку буде виконуватися частіше, ніж при використанні інших, більш «розумних» кільцевих хеш-функцій.

При умові поганої хеш-функції, (наприклад:, hash = const), рядок 6 з високою ймовірністю буде виконуватися n разів, тобто при кожній ітерації циклу. Так як вона витрачає час Omega (m), сам алгоритм буде вимагати час Omega (mn).[11].

Було проаналізовано чотири класичних алгоритми пошуку рядка у рядку. У кожного з алгоритмів є свої сильні та слабкі сторони та використовуються вони у різних задачах.

Було реалізовано два з найбільш підходящих алгоритмів Карпа-Рабіна і Кнута-Морріса-Пратта. Також булв реалізований пошук рядка в рядці з використанням внутрішнього функціоналу серидовища розробки. Тобто методу Contains.

Результат дослідження виконання кожного з методів 100000 разів. (рис.1.3). Так як при виконанні невеликої кількості функцій не можна було б замірити час виконання (рис. 1.3).

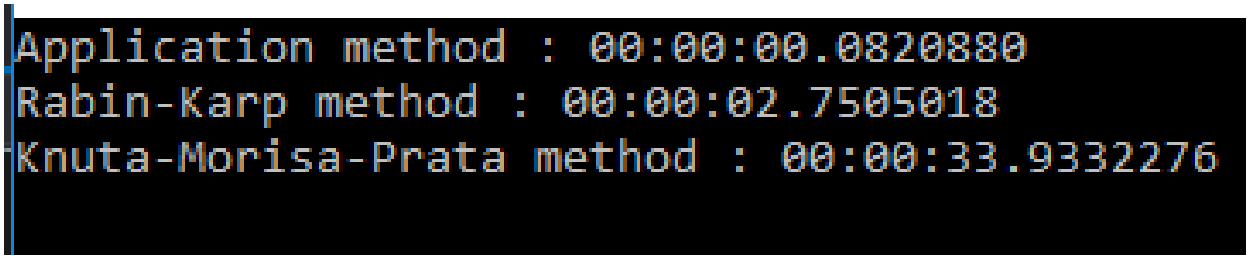

Рис.1.3Результат тестування алгоритмів пошуку на швидкість

Виконання методу який реалізований в серидовищі розробки (тобто методу Contains) було 0.082с.

Виконання методу в якому реалізований алгоритм Карпа-Рабіна було 2.75с.

Виконання методу в якому реалізований алгоритм Кнута-Морріса-Пратта було 33.93с.

На основі цих даних та простоти реалізації алгоритмів було вибрано використати в реалізації системи моніторингу документообороту функціоналу реалізованого в серидовищі розробки.

Метод Contains – має закритий внутрішній код тому дізнатися який алгоритм використовується для перевірки входження рядка неможливо.Так як в системі моніторингу не буде відбуватися пошук у великих рядках то най актуальнішим для реалізації пошуку у системі моніторингу буде використання вбудований в серидовище розробки функціонал. Реалізація будьякого іншого алгоритму у системі.

Через важкість покриття всіх крайніх випадків (наприклад шуканий рядок більший за основний) та можливість реалізації деякого використаного методу на нижчому рівні абстракцій та використанням коду який займається контролем пам’яті данний метод буде виконуватися швидше ніж будь яка реалізація алгоритмів вище.

Contains має атрибут в який передається рядок який шукається у даному рядку. Та повертає булеву змінну вказуючу є шукатий рядок в даному чи ні.

1.6 Функціонування компанії та системи документообігу

В реальних умовах завод-постачальник передає КК, певну кількість виробленої продукції, місцезнаходження, данні про вироблення наступної необхідної кількості продукції, залишки продукції на заводі (рис. 1.7).

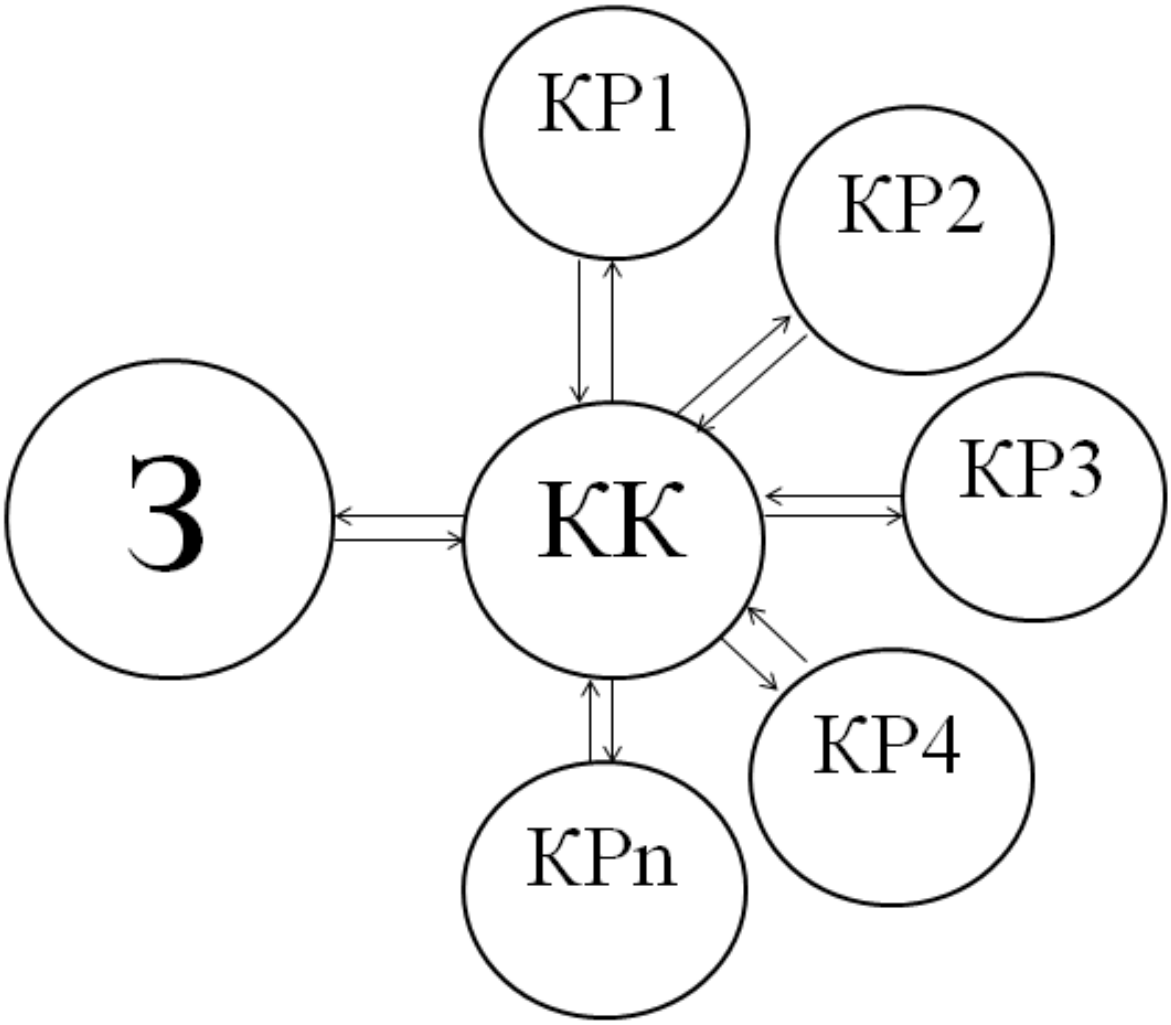

Рис.1.4 Структура навколо корпоративної компанії

На рисунку 1.4 прийняті такі скорочення:

З – завод; КК – Корпоративна компанія; КР – Кінечний реалізатор.

В функції корпоративної компанії входять:

- логістичнні питання для заводу;
- передача данних про к-ть необхідної продукції;
- данні про місце кінцевої доставки продукції;
- договори про замовлення продукції;
- в деяких випадках функції реалізатора;
- збір данних про залишки продукції на складах;
- маркетинг продукції.

Кінцевий реалізатор зв'язується з КК з допомогою послуг торгових агентів чи інших представників, що займають позиції відповідальних за розповсюдження продуктів, договори які необхідні для виконання продажів продукцій та інших дій, залишки продукцій на складах КР. Також для статистики, та інших потреб маркетингу передаються данні про продажу.

Корпоративна компанія та передає різноманітні данні кінцевому реалізатору. Наприклад: данні про залишки, списки продукції можливі для закупівлі, види договорів для можливості продажів продукції (рис. 1.5).

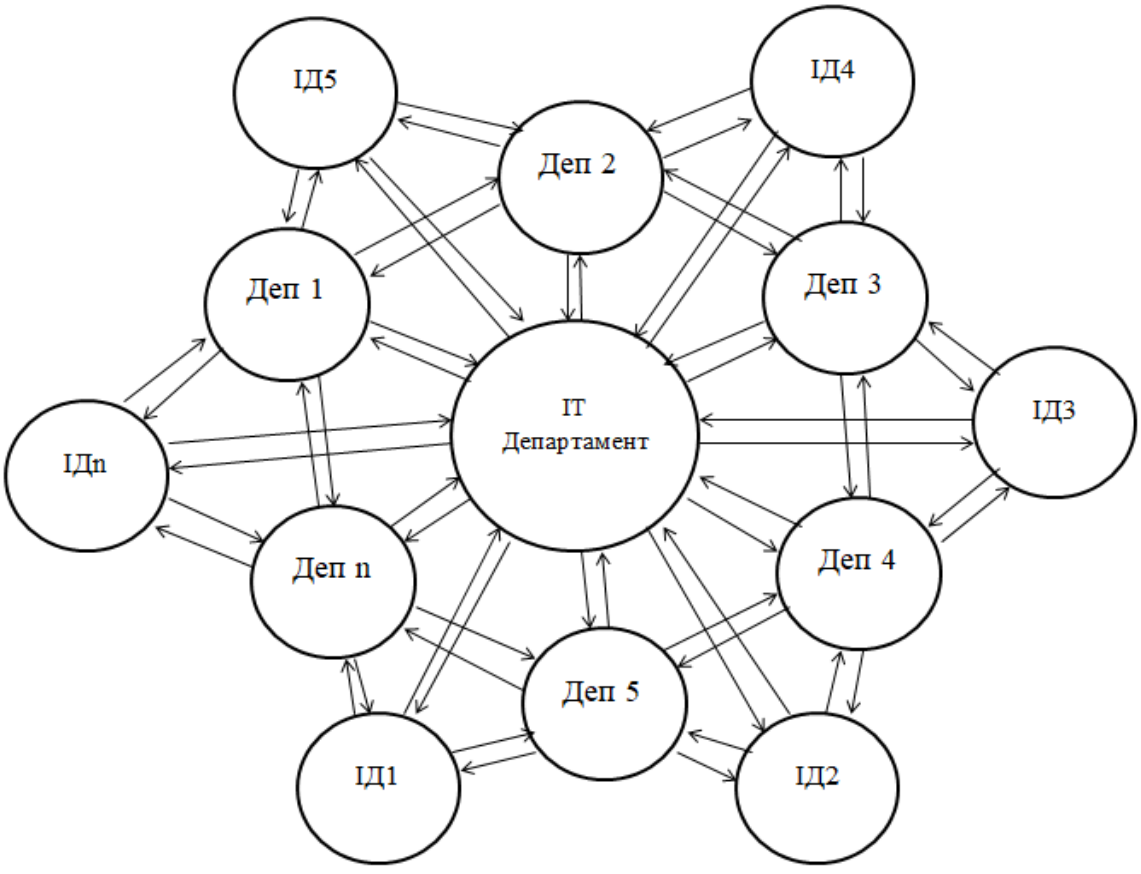

Рис.1.5 Структура обміну інформації всередині КК

На рисунку 1.5 прийняті такі скорочення: ІТ – ІТ департамент, Деп1….n - департаменти корпоративної компанії, ІД1….n – інші джерела інформації

Між департаментаментами та іншими джерелами відбувається обмін багатьма видами інформації документами про замовлення , залишками і т.д.

В даній компанії є багато програмних продуктів для взаємодій з документами , а також даними які зберігаються в базах даних.

На рівні з десктопними програмами є також інтернет магазин, веб сервіси та програми для систем IOS та Android.

В роботі буде розглянуто 3 програми для працівників різних рівнів доступу, та різними завданнями.

Одна з програм працює під системою IOS, та створена в основному для роботи торгових агентів та інших працівників, які займаются росповсюдженям товарів.

Та двома програмами працюючими під системою Windows .

Програма під системою IOS (Програма 1) , розроблялась тільки для працівників, що займаються реалізацією продукції в кінцеві точки, перевіркою правильності виставлення продукції на поличках (зумовленого в договорах), перевірці зумовлених цін на продукцію, показ існуючих акцій для клієнтів, показ залишків продукції на складах , створення замовлення та передачі її у внутрішню систему.

Ця програма має достатньо недоліків, що зумовлені підключенням сервісів інших компаній (наприклад сервісу розпізнаючого частини фотографій). Також одним з недоліків може бути людський фактор при користуванні програмами чи неспроможність створення необхідної логіки у данній архітектурі.

Одна з програм під Windows (Програма 2). розроблялась також для працівників займаючихся реалізацією продукції до кінцевих реалізаторів, але ця програма має більш обширний спектр можливостей, такі як огляд

інформації про торгових агентів (створений для їх головуючих). Можливість огляду компаній, які виконують функції доставки товарів. Так як програма активно дороблюється, утворюються проблеми пов’язані з сумісністю нових можливостей з вже створеною архітектурою. Системних помилок. Помилок пов’язаних з інтеграцією продуктів інших компаній. Та помилок створених «людським фактором» .

Інша програма під Windows (Програма 3). Створена ця програма для маніпуляцій з інформацією різних членів компанії. (Наприклад редактуваня та доповнення різних договорів бухгалтерами). Ця програма має такі ж проблеми як і попередня.

В двох останніх програмах виникають також проблеми пов’язані з доступом різних членів компанії до різних вкладинок інформації та різних видів маніпуляцій з ними. Різний рівень доступу реалізується за допомогою дозволу доступу до програм тільки з внутрішньої мережі. Це дозволяє не тільки виключити доступ людей не пов’язаних з компанією , а також повністю ідентифікувати користувача який відкрив програму. Доступ до різних рівнів інформації створюється за допомогою отримання даних користувача , за допомогою мережі, а також присвоєних груп доступів для окремих членів компанії, та утворення їх наслідування.

Також є інтернет магазин та веб сервіси для внутрішнього користування та для користування інших компаній. Веб сервіси для інших компаній тісно зв’язаний з інтернет магазином. Інтернет магазин виконує всі необхідні функції для замовлення продукції, список документів клієнта, та новини про нові продукти та акції у компанії.

Сервіс пов’язаний з інтернет магазином приймає данні про продажі сторонніми компаніями продукції данної компанії, залишки продукції на складах сторонніх компаній та інші данні необхідні для аналітики.

Також є внутрішні сервіси у яких реалізований весь функціонал необхідний для функціонування документообороту, запису даних у базу даних. Ці сервіси закриті для викликів з інших мереж. Графік обміну інформації в корпоративній

компанії між серверами та програмними продуктами знаходячимися на інших компьютерах відображено на рис.1.6

Основні функції:

- Прийняття необхідних даних для функцій документообороту.
- Відправлення відповідей у серидовище виклику сервісів.
- Запис даних у бази даних.
- Доповнення даних для заповнення в бази данних
- Видалення інформації з бази даних.

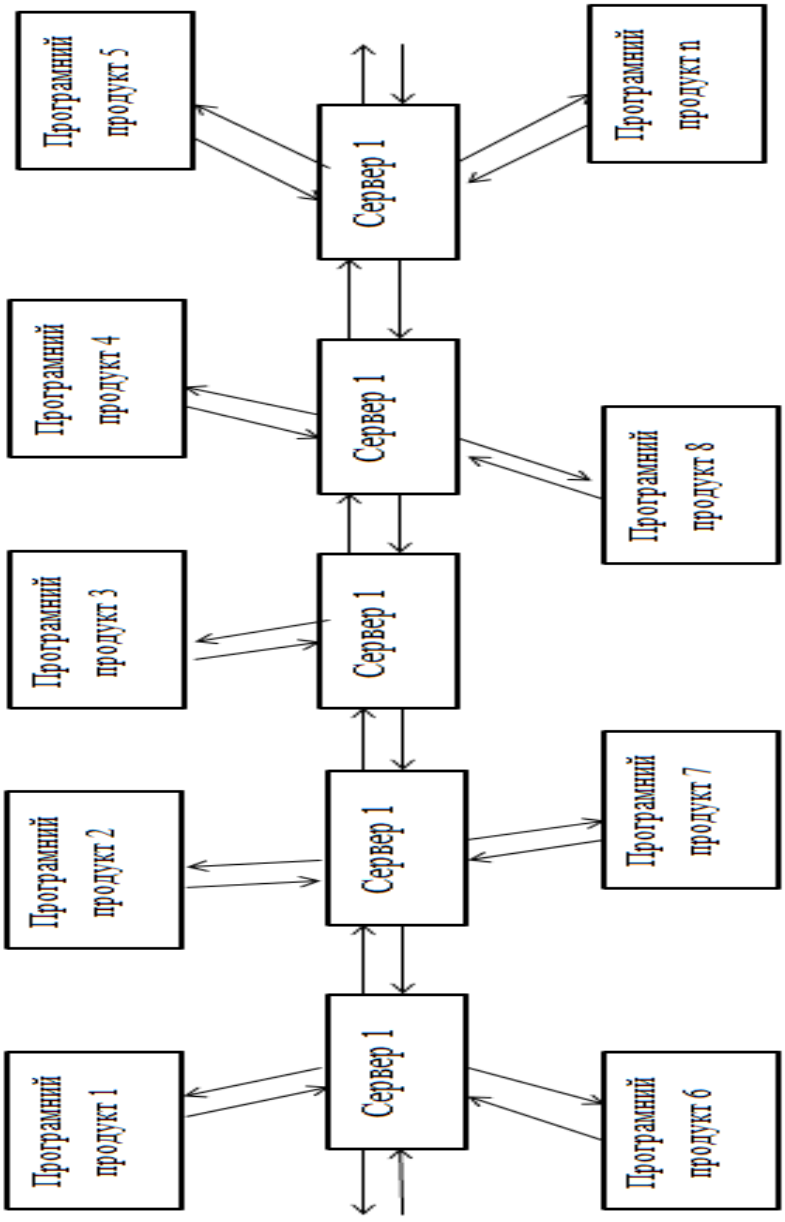

Рис.1.6 Сервери і програмні продукти

1.7 Існуючі системи моніторингу

Однією з систем моніторингу, що функціонує в КК, є програма, яка моніторить сервери, сервіси бази данних.

Ця програма, отримує пінг баз даних, та підсвічує назву даної БД якщо пінг надто високий. Також перевіряє чи відповіда БД .Також програма отримує версію сервісів , та відмічає працює сервіс чи ні.

Цей сервіс дуже зручний для системних адміністраторів та СПО , але не досить корисний для розробників. Для них цей сервіс дає змогу швидко дізнатися про утворення проблеми та почати дізнаватися про причини їх утворення. Також є веб сервіс який показує ту ж саму інформацію , але також моніторить сайти. Та виводить текст внутрішніх помилок деяких програм.

Сервіс досить зручний, для моніторингу деяких сайтів. Також досить непогана ідея виводу помилок у роботі деяких програм. Але не досить зручне відображення , так як всі помилки виводяться списком і немає розділення на помилки з різних програмних продуктів. Також через велике навантаження розробники не можуть своєчасно виявити помилку та своєчасно відреагувати на неї.

Також є моніторинг сервісу для обміну дистрибуційної інформації між дебітором та нашою компанією. Це інформація про продажи, залишки продукції та отримання замовлень від дебіторів та інше.

Моніторинг створює можливість оглядати інформацію відправлену від дебітора, та помилку яка утворилась. А також вітповідь дестрибутору. Цей моніторинг береться за основу сервісу для моніторингу описаних програмних продуктів.

1.8 Потоки інформації в системі документообороту

Система документообігу корпоративної компанії складається з багатьох програмних продуктів.

Програмні продукти поділяються на користувацькі програми, з якими працюють:

- бухгалтери;
- менеджери;
- інші працівники компанії, які беруть участь в процесі документообігу.

Також до користувацького програмного забезпечення відносяться веб сервіси для обміну інформацією з іншими компаніями, оскільки в них зосереджена мінімальна частина бізнес логіки для документообігу.

Сервіси, в яких зосереджена основна частина бізнес логіки.

Під сервісами будемо розуміти програмні продукти, які опрацьовують данні, виконують необхідні операції над інформацією, приймають данні з програмних продуктів користувачів та інших програмних продуктів.

Так як у інтерфейсах зосереджена основна частина бізнес логіки, програмні продукти для кінцевих користувачів є інтерфейсами для роботи з сервісами.

В кінцевих програмних продуктах реалізована система доступу до інформації основана на корпоративних даних компанії. Доступ до сервісів мають тільки працівники IT через достатню кількість необхідних технічних знать.

Для кінцевих програмних продуктів та користувачів сервіси є чорними скриньками.

В сервісах та кінцевих програмних продуктах реалізованаі системи вилову помилок як системних (помилок в алгоритмах та самому коді програми) так і користувацьких, помилки в документах (такі як не правильно визначені продукти, ціни, покупці і т.д.). Ці системи не є цілісними (не

працюють як єдина система) система вилову помилок реалізована як розособлена система, тобто складається з багатьох невеликих систем спостерігаючих кожна за свою частину функцій бізнеспроцесів документообігу. Дані про помилки не зберігаються в одному місці, в деяких частинах взагалі не зберігаються. В різних сервісах помилки визначаються навіть у різних програмних оточеннях.

Сервіси отримують дані з програмних продуктів для кінцевих користувачів. В ККї існує багато видів документів. Наприклад: накладні на продаж товару, виписки з банків про оплати, транспортні накладні, дані про продажі, транспортні накладні продукції корпоративної компанії з інших компаній (ця інформація може отримуватися за допомогою веб сервісів).

Кожен з документів може мати всередині рвзні несистемні помилки, що утворюються при неправильному заповненні інформації.

Наприклад: неправильно виставлена адреса доставки.

Несистемні помилки виявляються внутрішніми системами сервісів, а також відповідальними людьми. Несистемні помилки в основному позначаються в статусі документа і позначають документ як не правильний з зазначенням помилки. Після цього стан документа користувачі та відповідальні за бізнес процес можуть дізнатися з користувацьких програмних продуктів у рідких випадках відповідальні повідомляються за допомогою імейлу.

Система моніторингу документообігу буде отримувати несистемні помилки з сервісів та кінцевими програмними продуктами. Так як бізнес логіка для кожної функції процесу документообігу визначена в визначеному методі сервісу , то не системні помилки (помилки у обробці документів за зазначеною бізнес логікою) будуть визначатися у самих методах відповідаючих за обробку даних.

Данні будуть відправлятися за допомогою веб запитів до сервісів.
Також несистемними помилками можуть буди помилки доступу чи не відповідність прав для відповідних функцій бізнеспроцесів.

Для системи моніторингу документообігу сервіси та програмні продукти для працівників компанії є чорними скриньками.

В систему моніторингу будуть відправлятись:

- данні про програмний продукт в якому виникла помилка (назва сервісу у якому відбулася помилка чи назва програмного продукту для працівника компанії);
- текст помилки (текст помилки визначається функціями бізнес логіки відповідаючими за обробку інформації в якій виникла помилка);
- тип документу (транспортна накладна, накладна з продажами, залишки продукції і т.д.);
- ідинтифікатор особи при дії якої виникла помилка (у програмному продукті для кінцевого користувача, у сервісах особу яка внесла інформацію в якій виникла помилка) дані про особу отримуються з внутрішньої бази даних, так як до більшісті програмних продуктів мають доступ тільки авторизовані у корпоративній мережі особи.

Для сервісів обміну даних з іншими компаніями відправником буде відповідальна особа. Також у систему моніторингу будуть відправлятися адреси електронної пошти осіб у яких виникли помилки. Ця інформація зберігається у базі даних корпоративної компанії і її можна отримати за ідентифікатором особи.

Схема обміну інформації показана на рис. 1.7.

На рис. 1.7 цифрами позначені процеси:

1 – запиту необхідної інформації до сервісу реалізуючого систему моніторингу документообороту;

2 – передачі неохідної для документообігу інформації від користувачів програмних продуктів для користувачів та від сервісів обміну інформації з іншими компаніями. Інформація для утворення документів, інформації необхідної для запису у БД;

3 – обміну інформацією про документи мід сервісами. Необхідний для виповнення всіх функцій документообігу відповідний для кожного з типів документів.

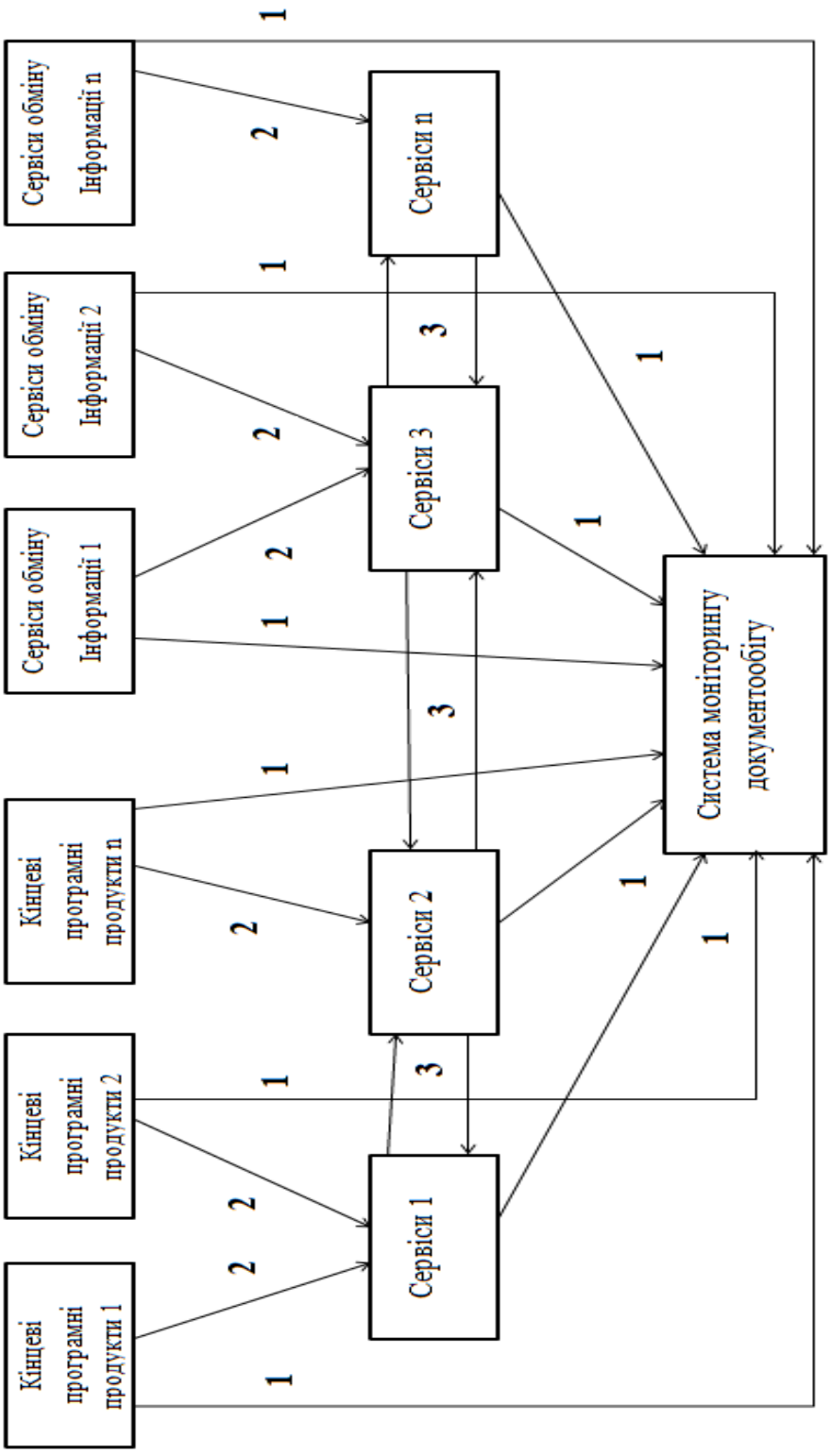

Рис.1.7.Схема обміну інформації між програмними продуктами

1.9 Спосіб пошуку помилок у документах

Перед внесенням данних з документів, в документах проходить аналіз данних на помилки. Перевірки документів проходять проходять за таким типом алгоритмів рис.1.8

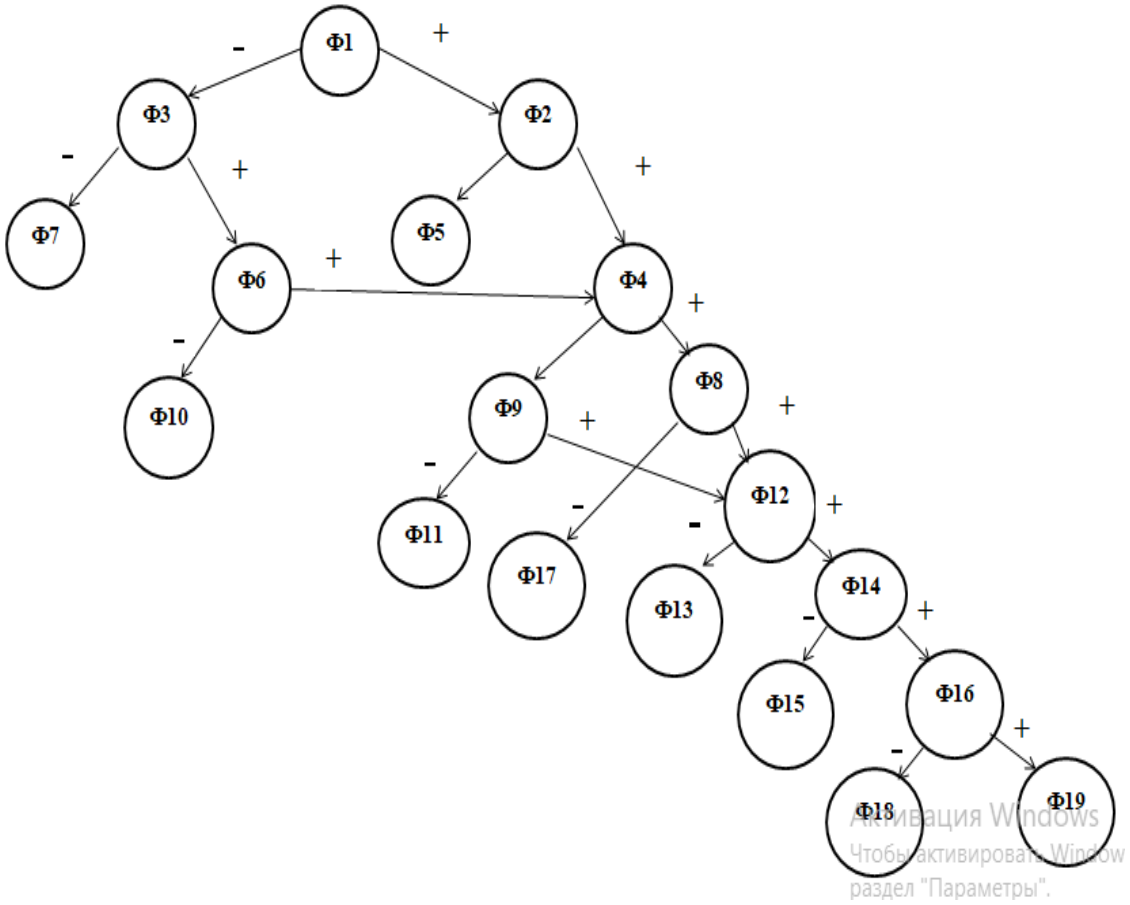

Рис.1.8 Дерево функцій визначення помилок

Транспортна накладна

Ланні волія: XXXXXXXXXXXXXXXXXX
Адреса доставки :XXXXXXXXXXXXXXXXXXXX

Адреса складу: XXXXXXXXXXXXXXXXXXXX

Номер машини: XXXXXX

Продукція:

Ф1.Заповнені данні водія і правильно заповнені.

Ф2.Співпада номер телефону отриманий з бази по данним водія з телефоном с документу.

Ф3.Заповнені данні телефону в документі і заповнені правильно.

Ф4.Номер машини в документі заповнений і заповнений правильно

Ф5.Вивід помилки через не співпадання номеру телефону

Ф6.Є данні водія з бази

Ф7.Вивід помилки про не вірно заповнені номер телефону і данні водія.

Ф8.Співпадають данні номеру машини з бази з номером машини з документу.

Ф9.Є данні номеру машини в базі .

Ф10.Вивід помилки немає данних водія в базі.

Ф11.Вивід помилки про відсутність номеру машини в базі і в документі.

Ф12.Співпадає номер машини з бази з номером машини з документу

Ф13.Вивід помилки не співпадають номери машини

Ф14.Внесена в документ адреса складу.

Ф15.Вивід помилки не внесена в документ адреса складу

Ф16.Внесена в документ алреса доставки

Ф17.Вивід помилки номер машини з бази і з документу не співпадають

Ф18.Вивід помилки не внесена адреса доставки

Ф19. Данні транспортної накладної вносяться в базу.

Замовлення:

Данні замовника: XXXXXXXXXXXXXXXXXX

Номер телефона замовника: XXXXXXXXXXXX

Адреса доставки :XXXXXXXXXXXXXXXXXXXX

Адреса складу: XXXXXXXXXXXXXXXXXXXX

ІНН замовника: XXXXXX

Продукція:

Ф1. Заповнені данні замовника і правильно заповнені.

Ф2.Співпада номер телефону отриманий з бази по данним замовника з телефоном с документу.

Ф3.Заповнені данні телефону в документі і заповнені правильно.

Ф4.ІНН замовника в документі заповнений і заповнений правильно

Ф5.Вивід помилки через не співпадання номеру телефону

Ф6.Є данні замовника з бази

Ф7.Вивід помилки про не вірно заповнені номер телефону і данні водія.

Ф8.Співпадають данні ІНН замовника з бази з ІНН замовника з документу.

Ф9.Є данні ІНН замовника в базі .

Ф10.Вивід помилки немає данних замовника в базі.

Ф11.Вивід помилки про відсутність ІНН замовника в базі і в документі.

Ф12.Співпадає ІНН замовника з бази з ІНН замовника з документу

Ф13.Вивід помилки не співпадають ІНН замовника

Ф14.Внесена в документ адреса складу.

Ф15.Вивід помилки не внесена в документ адреса складу

Ф16.Внесена в документ алреса доставки

Ф17.Вивід помилки ІНН замовника з бази і з документу не співпадають

Ф18.Вивід помилки не внесена адреса доставки

Ф19. Данні замовлення вносяться в базу.

Повернення:

Данні повертача : XXXXXXXXXXXXXXXXXX

Номер телефона повертача: XXXXXXXXXXXX

Адреса магазину :XXXXXXXXXXXXXXXXXXXX

Адреса складу: XXXXXXXXXXXXXXXXXXXX

ІНН повертача: XXXXXX

Продукція:

Ф1.Заповнені данні повертача і правильно заповнені.

Ф2.Співпада номер телефону отриманий з бази по данним повертача з телефоном з документу.

Ф3.Заповнені данні телефону в документі і заповнені правильно.

Ф4.ІНН повертача в документі заповнений і заповнений правильно

Ф5.Вивід помилки через не співпадання номеру телефону

Ф6.Є данні повертача з бази

Ф7.Вивід помилки про не вірно заповнені номер телефону і данні повертача.

Ф8.Співпадають данні ІНН повертача з бази з ІНН повертача з документу.

Ф9.Є данні ІНН повертача в базі .

Ф10.Вивід помилки немає данних повертача в базі.

Ф11.Вивід помилки про відсутність ІНН повертача в базі і в документі.

Ф12.Співпадає ІНН повертача з бази з ІНН повертача з документу

Ф13.Вивід помилки не співпадають ІНН повертача

Ф14.Внесена в документ адреса складу.

Ф15.Вивід помилки не внесена в документ адреса складу

Ф16.Внесена в документ алреса магазину

Ф17.Вивід помилки ІНН повертача з бази і з документу не співпадають

Ф18.Вивід помилки не внесена адреса магазину

Ф19. Данні повернення вносяться в базу.

Після проходження алгоритму помилки будуть відправлятись в програмний продукт реалізуючий функціонал системи моніторингу документообігу.

Вся інформація,що необхідна для запиту про помилки отримується з системи.

Особа, що формує документ, приходить у сервіс з продуктів для створення документу разом з інформацією документа. З системи також отримується електронна адреса Також в сервіс приходить назва програмного продукту де був стрворений документ.

2. РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ МОНІТОРИНГУ СИСТЕМИ ДОКУМЕНТООБІГУ

2.1 Дерево цілей

Дерево цілей – графічне зображення взаємозв'язку і підпорядкованості цілей, що відображає розподіл місії і мети на цілі, під цілі, завдання та окремі дії.

Дерево цілей з кількісними показниками, що використовуються в якості одного із засобів при прийнятті рішень, і носить назву "дерева рішень". Головна перевага "дерева рішень" перед іншими методами – можливість пов'язати ставлення цілі з діями, що підлягають реалізації в сьогоденні.

Основна ідея щодо побудови "дерева цілей" – декомпозиція – розкриття структури системи, при якому за однією ознакою її поділяють на окремі складові. Декомпозиція використовується для побудови "дерева цілей", щоб пов'язати генеральну мету зі способами її досягнення, що сформульовані у вигляді завдань окремим виконавцям.

Не існує універсальних методів побудови "дерева цілей". Способи його побудови залежать від характеру цілі, обраного методологічного підходу, а також від того, хто розробляє "дерево цілей", як він представляє собі поставлені перед ним завдання, як він бачить їхній взаємозв'язок.

Основне правило побудови "дерева цілей" – "повнота редукції" – процес зведення складного явища, процесу або системи до більш простих складових.

Для реалізації цього правила використовують такий системний підхід:

- ціль вищого рівня є орієнтиром, основою для розробки (декомпозиції) цілей нижчого рівня;
- цілі нижчого рівня є способами досягнення мети вищого рівня і мають бути представлені так, щоб їхня сукупність зумовлювала досягнення початкової цілі.

Вимоги до побудови "дерева цілей"::

1. Цілі кожного рівня повинні бути порівнянні по масштабу і значенню.

2. Формулювання цілей повинне забезпечувати можливість кількісної і якісної оцінки досягнення мети.

3. Основним принципом побудови дерева цілей є повнота редукції, тобто кожна мета певного рівня повинна бути зображена у вигляді під цілей наступного рівня так, щоб сукупність під цілей давала повне уявлення про початкову ціль.

4. Формулюючи цілі різних рівнів необхідно описати бажані результати, а не способи їх отримання.

5. Під цілі кожного рівня повинні бути незалежні одна від однієї і не повинні виходити одна з іншої.

6. Ознакою завершення побудови дерева цілей є формулювання таких понять, які визначають альтернативні способи досягнення цілі. Самі вони не є цілями, це заходи щодо досягнення цілі вищого рівня.

7. Відсутність суперечностей між цілями, що знаходяться на різних рівнях "дерева цілей".

8. Декомпозицію місії і цілі на всіх рівнях слід проводити за одним і тим же методологічним підходом є

9. Цілі усіх рівнів мають бути виражені в конкретних обсягах, строках з визначенням конкретних виконавців (відповідальних).

10. Забезпечення узгодженості, зв'язку між цілями різного порядку. При цьому слід враховувати наявність двох видів зв'язків між цілями - горизонтальних і вертикальних.

В дереві функцій відображені всі необхідні ціли для побудови системи моніторингу документообігурис.2.1.

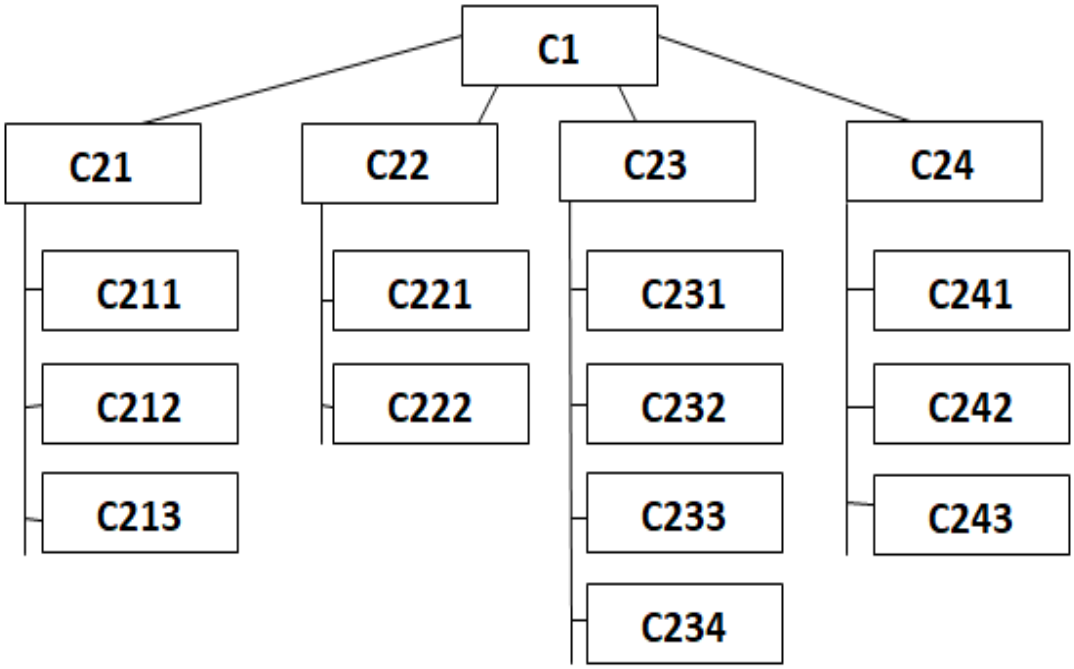

Рис.2.1 Дерево цілей

- С1- створення системи моніторингу документообігу
  - С21 – проаналізувати програмне забезпеченя корпоративної компанії
    - С211 – ознайомитись з програмними продуктами виконуючі функції моніторингу
    - С212 – ознайомитись з структурою документообігу
    - С213 – ознайомитися з проблемами документообігу даної корпоративної компанії
  - С22 – обрати середовище розробки
    - С221 – визначитися з операційною системою сервера
    - С222 - встановити середовище розробки
  - С23 – підключити технології та розробити алгоритми
    - С231 – автоматизувати роботу системи
    - С232 – розробити алгоритм пошуку рішення проблеми з вже внесених

        - С233 – розробити систему оповіщення відповідальних осіб за допомогою електронної пошти.
        - С234– автоматизувати визначення відповідального
    - С23 – підключити технології та розробити алгоритми
        - С241 – прослуховування порту для прийняття помилок з систем документообігу
        - С242 – реалізація авторизації з допомогою cookie
        - С243 – реалізація системи зміни статусу помилки
        - С244 – реалізація виводу даних з бази користувачам.

2.2 Концептуальна модель

Концептуальна модель – це абстрактна модель, що визначає структуру системи, властивості її елементів і причинно-наслідкові зв'язки, властиві системі і суттєві для досягнення мети моделювання.

Основні елементи концептуальної моделі:

- умови функціонування об'єкта, визначені характером взаємодії між об'єктом і його оточенням, а також між елементами об'єкта
- мета дослідження об'єкта та напрямок покращення його функціонування
- можливості керування об'єктом, визначення складу керованих змінних об'єкта

Концептуальна модель відображена у вигляді чорної скриньки рис 2.2.

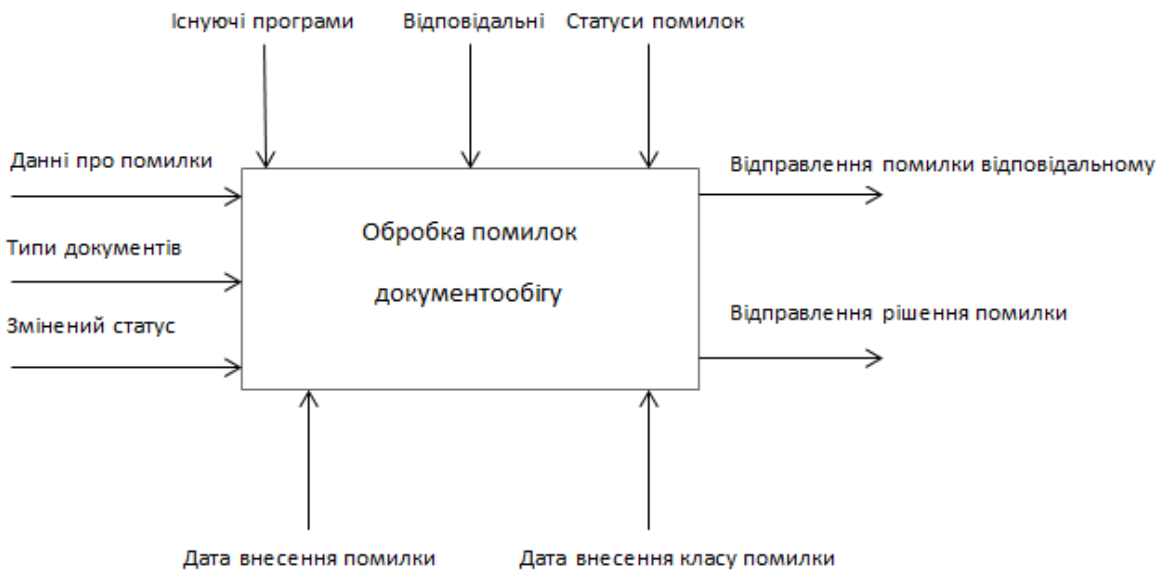

Рис.2.2 Концептуальна модель

Вхідні данні:

Данні про помилки в документах відправлені відправлені з систем документообігу, типи документів існуючих в системі, змінаений статус після розгляду помилки.

Нормативні данні:

Список існуючих програм виконуючих функцій документообігу, список відповідальних за кожен з видів документів, статуси помилок.

Данні системи:

Данни внесення користувацьких даних в систему.

Вихідні дані:

Внесення в систему даних про виправлення помилки, відправка повідомлення про виправлення відповідальним.

2.3 Дерево функцій

Дерево функцій – ієрархічна модель видів діяльності підприємства, що забезпечують досягнення дерева цілей.

Вершиною дерева функцій є головна мета підприємства, гілки дерева являють собою функції (або роботи), які необхідно реалізувати для досягнення головної мети підприємства і підлеглих їй цілей нижнього рівня.

Дерево функцій є основою для побудови бізнес-процесів організації (рис. 2.3).

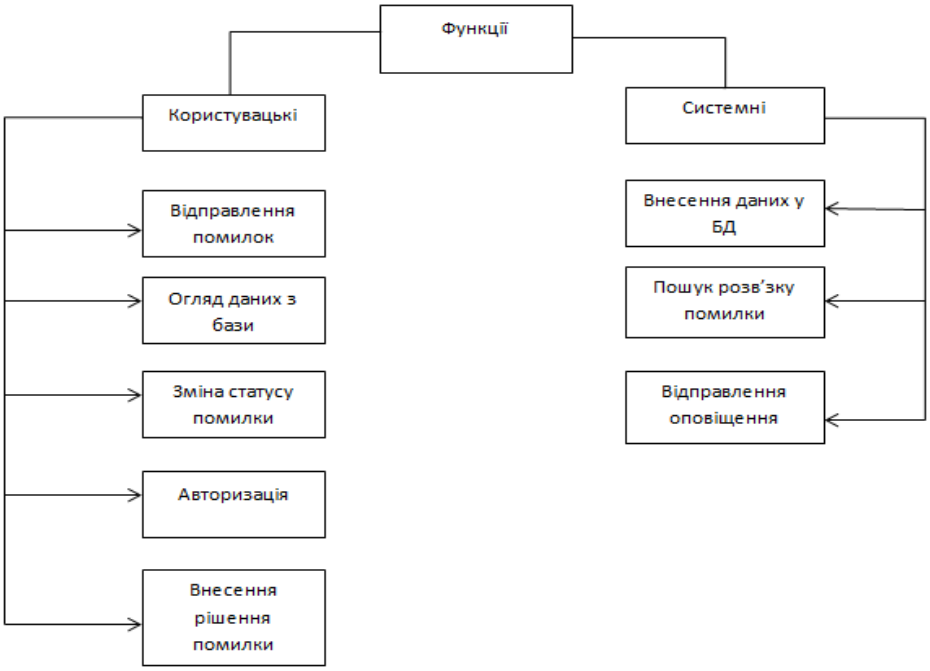

Рис.2.3 Дерево функцій

Користувачеві дозволяється відправляти сповіщення в систему про утворені помилки в системах документооборотів. Огляд стану розгляду помилки. Зміна статусу розгляду помилки. Авторизація у системі моніторингу. Внесення в базу рішення проблеми.

Система виконує функції внесення повних даних у базу даних збираючи дані з системи. На основі прийнятих даних про помилки відшукувати аналогічні , та вписувати рішення на основі рішення аналогічної проблеми.Відправлення оповіщень відповідальним особам.

3. МОДЕЛЮВАННЯ СИСТЕМИ ДОКУМЕНТООБІГУ КОРПОРАТИВНИХ КОМПАНІЙ

3.1 Обґрунтування вибору системи управління базою даних

Створення бази даних для web-системи миттєвих повідомлень буде виконуватися на СУБД MS SQL Server. SQL Server - це добре масштабуємий, повністю реляційний, швидкодіючий розрахований на багато користувачів сервер баз даних масштабу підприємства, здатний обробляти великі обсяги даних для клієнт-серверних додатків.

Одним з переваг SQL Server є простота його застосування, зокрема адміністрування. SQL Server Management Studio, що входить до складу всіх редакцій Microsoft SQL Server є повнофункціональним і досить простим засобом для адміністрування цієї СУБД.

Також має такі перевеги:

- високий ступінь захисту даних;
- потужні засоби роботи з даними;
- висока продуктивність;
- зберігання великих масивів даних;
- зберігання даних, що вимагають дотримання режиму секретності та не допустимість їх втрати.

Перед початком проектування баз даних необхідно зазначити, що будь-яка база даних є складовою частиною якоїсь інформаційної системи (ІС), яка має на увазі не тільки зберігання даних, але і їх обробку. Тому, проектування даних завжди супроводжує (а частіше передує) проектування алгоритмів їх використання.

У результаті проектування має бути визначена структура бази, тобто склад таблиць, їхня структура та логічні зв'язки. Структура реляційної таблиці визначається складом стовпців, їхньою послідовністю, типом даних кожного

стовпця та їхнім розміром, а також ключем таблиці. основні етапи, на які розбивається процес проектування бази даних інформаційної системи:

1. Побудова інфологічної моделі
2. Логічне проектування
3. Фізичне проектування

3.2 Розробка інфологічної моделі системи

Інфологічна модель бази даних являє собою опис об'єктів з набором атрибутів і зв'язків між ними, які виявляються в процесі дослідження як вхідних, так і вихідних даних. Вона призначається для структурного створення предметної області, з орієнтуванням на інформаційну увагу користувачів, що розробляється. Так само інфологічна модель повинна бути стабільною і незмінною, і бути поданням аспекту користувача на описану раніше предметну область.

Однак, при проектуванні інфологічної моделі, має бути присутня можливість для її збільшення і вставки допоміжних даних. Найпоширеніша модель в інфологічному моделюванні це модель "сутність-зв'язок", до головних компонентів відносяться - сутності і зв'язки. Під поняттям сутності трактується зміст об'єкта, про який набирають необхідну інформацію.

Для побудови здійснюються наступні заходи:

– обстеження предметної області, вивчення її інформаційної структури;

– виявлення всіх фрагментів, кожен з яких характеризується призначеним для користувача поданням, інформаційними об'єктами і зв'язками між ними, процесами над об'єктами;

– моделювання та інтеграція всіх моделей.

На рис.3.1 відображена інфологічна модель бази данних системи моніторингу документообігу.

Сутності: Запит, Програма, Відповідальний, Тип документу, Статус запиту.

Первинний ключ – матимуть усі сутності.

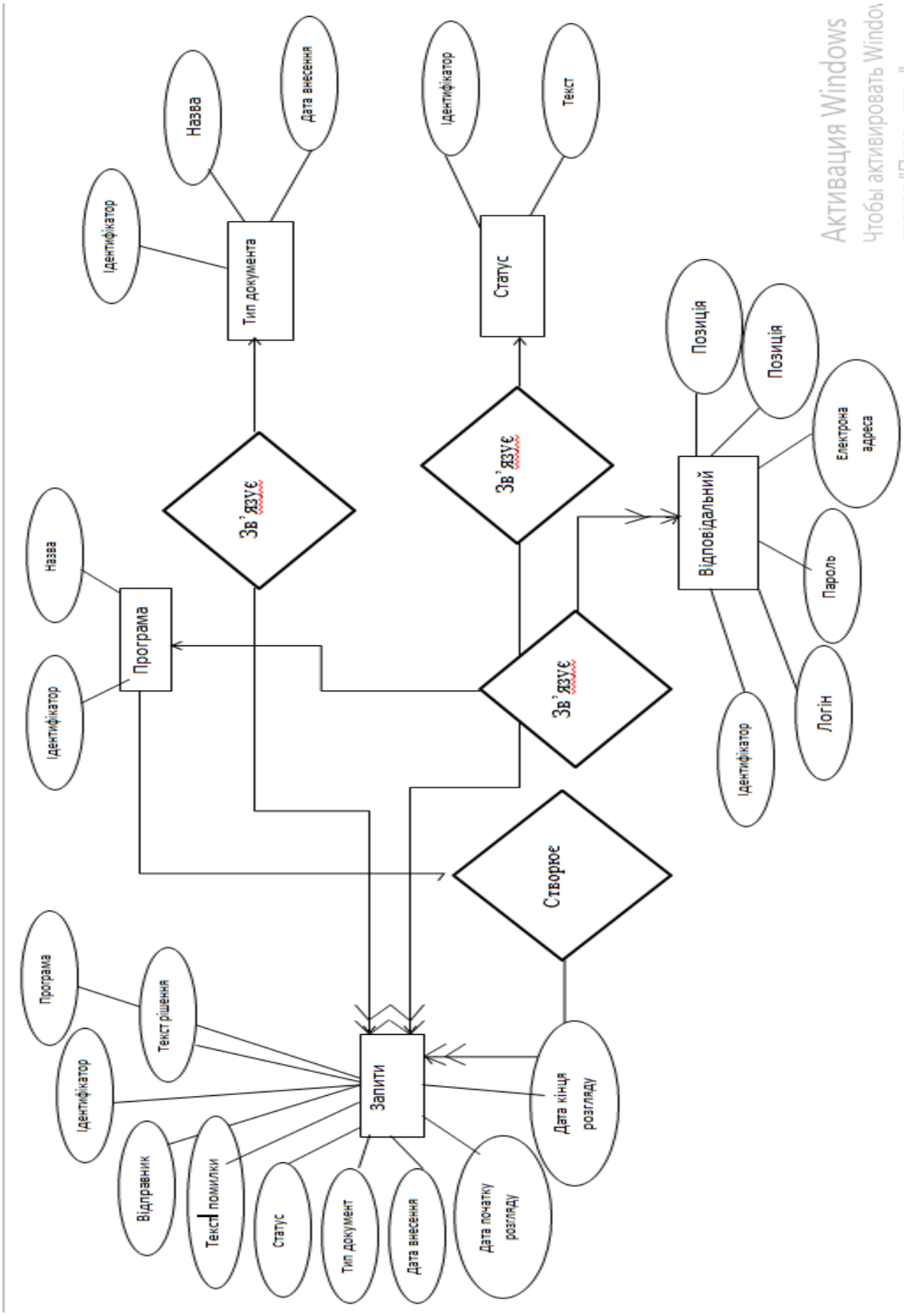

Рис.3.1 Інфологічна модель

У табл. 2.1 представлено назви сутності та їх ідентифікатори.

№ Таблиці	Назва таблиці	Ідентифікатор
1	Запит	Inquiries
2	Програми	Aplications
3	Тип документа	DocumentType
4	Відповідальні	Resipients
5	Статуси запитів	StateErrors

У табл. 2.1 Надано Словник даних бази даних підсистеми.

Таблиця 2.2 Словник даних бази даних підсистеми автоматизованої обробки замовлень КК

№ Атрибуту	Ідентифікатор	Назва атрибуту
1	I_ID	Ідентифікатор
2	I_Sender	Відправник
3	I_SenderEmail	Імейл відправника
4	I_StateError	Статус
5	I_AplicationID	Програма
6	I_DocumentTypeID	Тип документа
7	I_ErrorResponce	Текст рішення
8	I_ErrorText	Текст помилки
9	I_StartOfErrorConsideration	Дата початку розгляду
10	I_EndOfErrorConsideration	Дата кінця розгляду
11	I_DateCreate	Дата внесення
12	A_ID	Ідентифікатор
13	A_Name	Назва
14	D_Id	Ідентифікатор
15	D_Name	Назва
16	D_DateCreate	Дата внесення
17	R_ID	Ідентифікатор
18	R_Login	Логін
19	R_Password	Пароль
20	R_Email	Електрона адреса
21	R_Position	Позиція
22	R_AplicationID	Програма
23	S_ID	Ідентифікатор
24	S_Text	Текст

3.3 Даталогічна модель

Даталогічна модель (ДЛМ) являє собою структуру даних: структурні одиниці даних та їх елементи і зв'язки між елементами даних незалежно від їх змісту та середовища зберігання, а особливості обраної СКБД враховуються у фізичній моделі. Таким чином, ДЛМ є подальшою формалізацією інфологічної моделі і являє собою по модель даних (рис. 3.2).

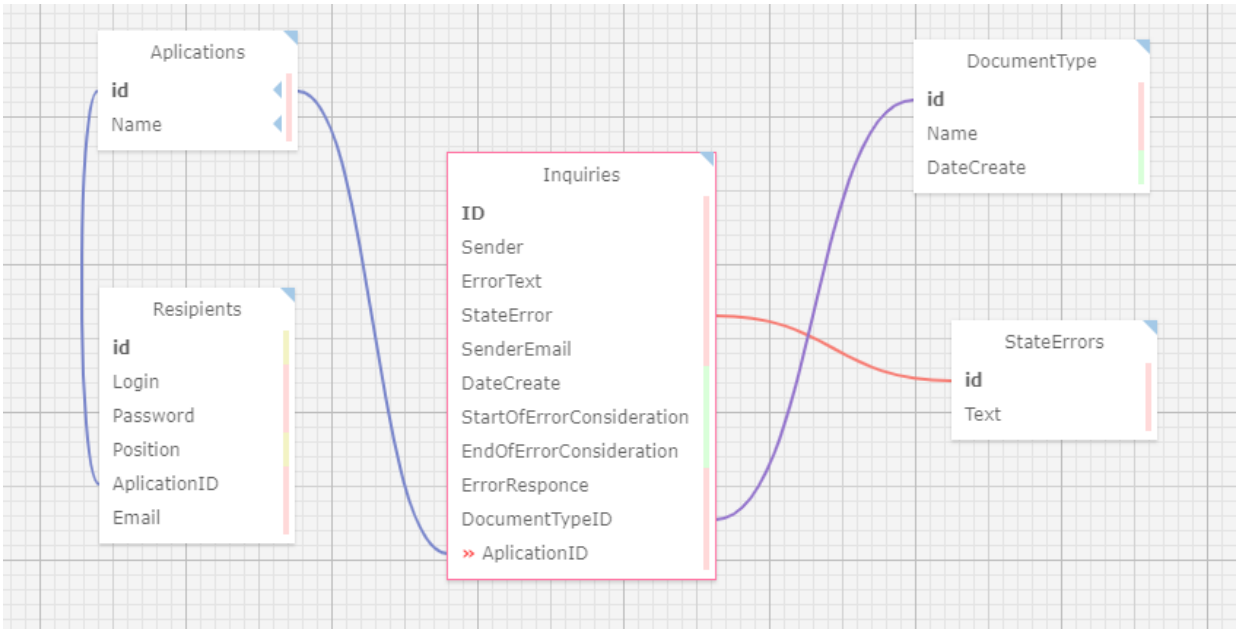

Рис.3.2 Даталогічна модель

Програма містить назву програми та ідентифікатор.

Відповідальний містить ідентифікатор, логін, пароль, електрону адресу, корпоративну позицію, ідентифікатор програми.

Запит містить ідентифікатор, відправника, текст помилки, ідентифікатор статусу запиту, електронну адресу відправника, дату запису запиту, початок розгляду запиту, кінець розгляду запиту, рішення помилки у запиті, тип документу, ідентифікатор програми з якої прийшла помилка.

Тип документа містить ідентифікатор, назву типу, дату створення запису.

Статус помилки містить ідентифікатор, текст статусу.

3.4 Фізична модель

Фізична модель бази даних містить всі деталі, необхідні конкретній СУБД для створення бази (найменування таблиць і стовпців, типи даних, визначення первинних і зовнішніх ключів і т. п.).

Фізична модель (рис. 3.3) будується на основі логічної з урахуванням обмежень, що накладаються можливостями обраної СУБД (в нашому випадку – MS SQL).

Фізична модель відображена на рис.3.3.

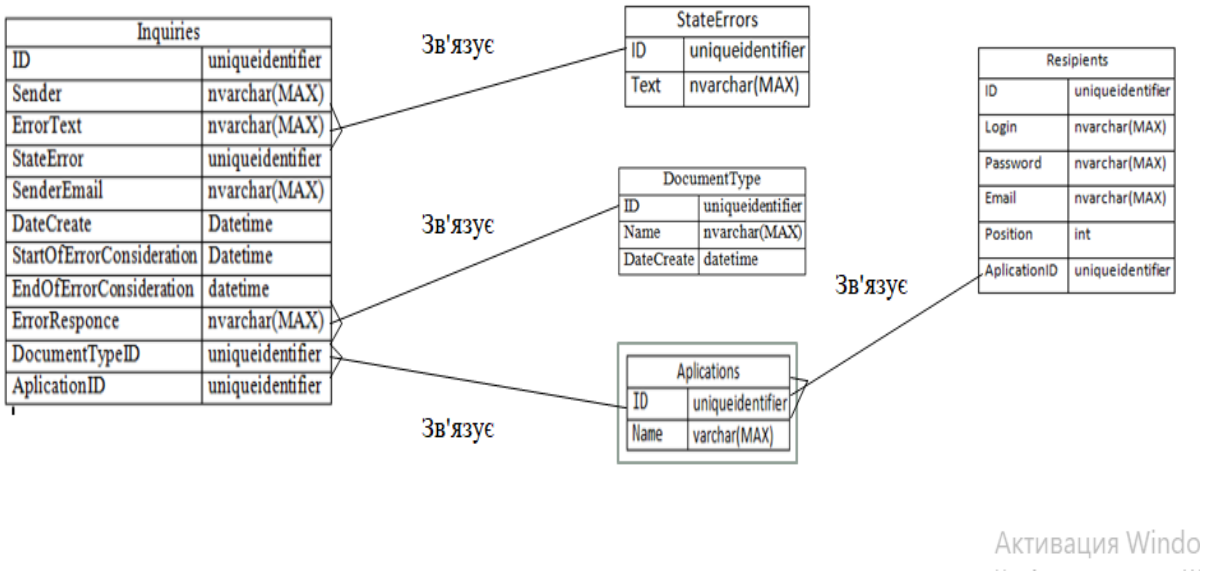

Рис.3.3 Фізична модель

Програма створює запит.

Тип документа звязується з класом помилки.

Відповідальний змінює статус запиту

Клас помилки звязується з запитом

3.5 Цілісність та захист бази даних системи

Захист інформації – комплекс заходів, що спрямовані на забезпечення найважливіших аспектів інформаційної безпеки (цілісності, доступності та, якщо потрібно, конфіденційності інформації і ресурсів, які використовуються для введення, зберігання, обробки і передачі даних).

Система буде безпечною, якщо вона, використовуючи відповідні апаратні і програмні засоби, управляє доступом до інформації так, що тільки належним чином авторизовані особи або особи, що діють від їхнього імені, процеси отримують право читати, писати, створювати і видаляти інформацію.

Система вважається надійною, якщо вона з використанням достатніх апаратних і програмних засобів забезпечує одночасну обробку інформації різного ступеня секретності групою користувачів без порушення прав доступу .

Основними критеріями оцінки надійності є політика безпеки та гарантованість.

Політика безпеки, включає в себе аналіз можливих загроз і вибір відповідних заходів протидії, відображає той набір законів, правил і норм поведінки, яким користується при обробці, захисті та поширенні інформації.

Вибір механізмів забезпечення безпеки системи здійснюється відповідно до сформульованої політик безпеки.

У сучасних СУБД підтримується один з двох найбільш загальних підходів до питання забезпечення безпеки даних: виборчий підхід і обов'язковий підхід. В обох підходах одиницею даних або "об'єктом даних", для яких повинна бути створена система безпеки, може бути як вся база даних цілком, так і будь-який об'єкт всередині бази даних.

Ці два підходи відрізняються таими властивостями:

– у разі виборчого управління деякий користувач володіє різними правами (привілеями чи повноваженнями) при роботі з цими

об'єктами. Різні користувачі можуть мати різні правами доступу до одного і того ж об'єкту.

– у разі виборчого управління, навпаки, кожному об'єкту даних присвоюється певний класифікаційний рівень, а кожен користувач має деяким рівнем допуску. При такому підході доступом до певного об'єкту даних мають тільки користувачі з відповідним рівнем допуску.

Для реалізації виборчого принципу передбачені такі методи.

У базу даних вводиться новий тип об'єктів БД – це користувачі. Кожному користувачеві в БД присвоюється унікальний ідентифікатор. Для додаткового захисту кожен користувач крім унікального ідентифікатора постачається унікальним паролем, причому якщо ідентифікатори користувачів у системі доступні системного адміністратора, то паролі користувачів зберігаються частіше за все в спеціальному кодованому вигляді і відомі тільки самим користувачам. Користувачі можуть бути об'єднані в спеціальні групи користувачів.

Один користувач може входити в кілька груп. У стандарті вводиться поняття групи PUBLIC, для якої повинен бути визначений мінімальний стандартний набір прав. За умовчанням передбачається, що кожен новостворюваний користувач, якщо спеціально не вказано інше, належить до групи PUBLIC. Привілеї або повноваження користувачів або груп – це набір дій (операцій), які вони можуть виконувати над об'єктами БД. В останніх версіях ряду комерційних СУБД з'явилося поняття "ролі".

Роль – поіменований набір повноважень.

Існує ряд стандартних ролей, які визначені в момент встановлення сервера баз даних. І є можливість створювати нові ролі, групуючи в них довільні повноваження. Введення ролей дозволяє спростити управління привілеями користувачів, структурувати цей процес.

Окрім того, введення ролей не пов'язане з конкретними користувачами, тому ролі можуть бути визначені і сконфігуровані до того, як визначені

користувачі системи. Користувачеві може бути призначена одна або декілька ролей.

Об'єктами БД, які підлягають захисту, є всі об'єкти, що зберігаються в БД: таблиці, подання, збережені процедури і тригери.

Для кожного типу об'єктів є свої дії, тому для кожного типу об'єктів можуть бути визначені різні права доступу. На самому елементарному рівні концепції забезпечення безпеки баз даних виключно прості. Необхідно підтримувати два фундаментальних принципи: перевірку повноважень і перевірку автентичності (аутентифікацію).

Перевірка повноважень заснована на тому, що кожному користувачеві або процесу інформаційної системи відповідає набір дій, які він може виконувати по відношенню до певних об'єктів. Перевірка автентичності означає достовірне підтвердження того, що користувач або процес, який намагається виконати санкціонована дія, дійсно той, за кого він себе видає.

Система призначення повноважень має в деякому роді ієрархічний характер. Найвищими правами і повноваженнями має системний адміністратор або адміністратор сервера БД. Традиційно тільки цей тип користувачів може створювати інших користувачів і наділяти їх певними повноваженнями.

СУБД в своїх системних каталогах зберігає опис самих користувачів, і опис їх привілеїв по відношенню до всіх об'єктів.

Схема надання повноважень будується за наступним принципом.

1) Кожен об'єкт в БД має власника – користувача, який створив цей об'єкт.
2) Власник об'єкта має всі права-повноваженнями на даний об'єкт, у тому числі він має право надавати іншим користувачам повноваження по роботі з даним об'єктом або забирати у користувачів раніше надані повноваження.

У ряді СУБД вводиться наступний рівень ієрархії користувачів – це адміністратор БД. У цих СУБД один сервер може управляти іншими СУБД (наприклад: MS SQL Server, Sybase).

У СУБД Oracle застосовується однобазовая архітектура, тому для них вводиться поняття подсхеми – частини загальної схеми БД і вводиться користувач, який має доступ до подсхеме.

У стандарті SQL не визначена команда створення користувача, але практично у всіх комерційних СУБД створити користувача можна не тільки в інтерактивному режимі, але і програмно з використанням спеціальних процедур, що зберігаються. Проте для виконання цієї операції користувач повинен мати право на запуск відповідної системної процедури.

У стандарті SQL визначені два оператори: GRANT і REVOKE відповідно надання та скасування привілеїв.

Оператор надання привілеїв має наступний формат:
GRANT {<список дій | ALL PRIVILEGES}
ON <імя_об'єкта> TO (<ім'я користувача>] PUBLIC} [WITH GRANT OPTION]

Тут список дій визначає набір дій з общедопустімого переліку дій над об'єктомданого типу.

Параметр ALL PRIVILEGES вказує, що дозволені всі дії з допустимих для об'єктів даного типу.

<Імя_ об’єкта> - задає ім'я конкретного об'єкта: таблиці, подання, збереженої процедури, тригера.

<Ім'я користувача> або PUBLIC визначає, кому надаються дані привілеї.

Параметр WITH GRANT OPTION є необов'язковим і визначає режим, при якому передаються не тільки права на зазначені дії, а й право передавати ці права іншим користувачам.

4. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ МОНІТОРИНГУ ДОКУМЕНТООБІГУ

4.1 Опис роботи програмного продукту

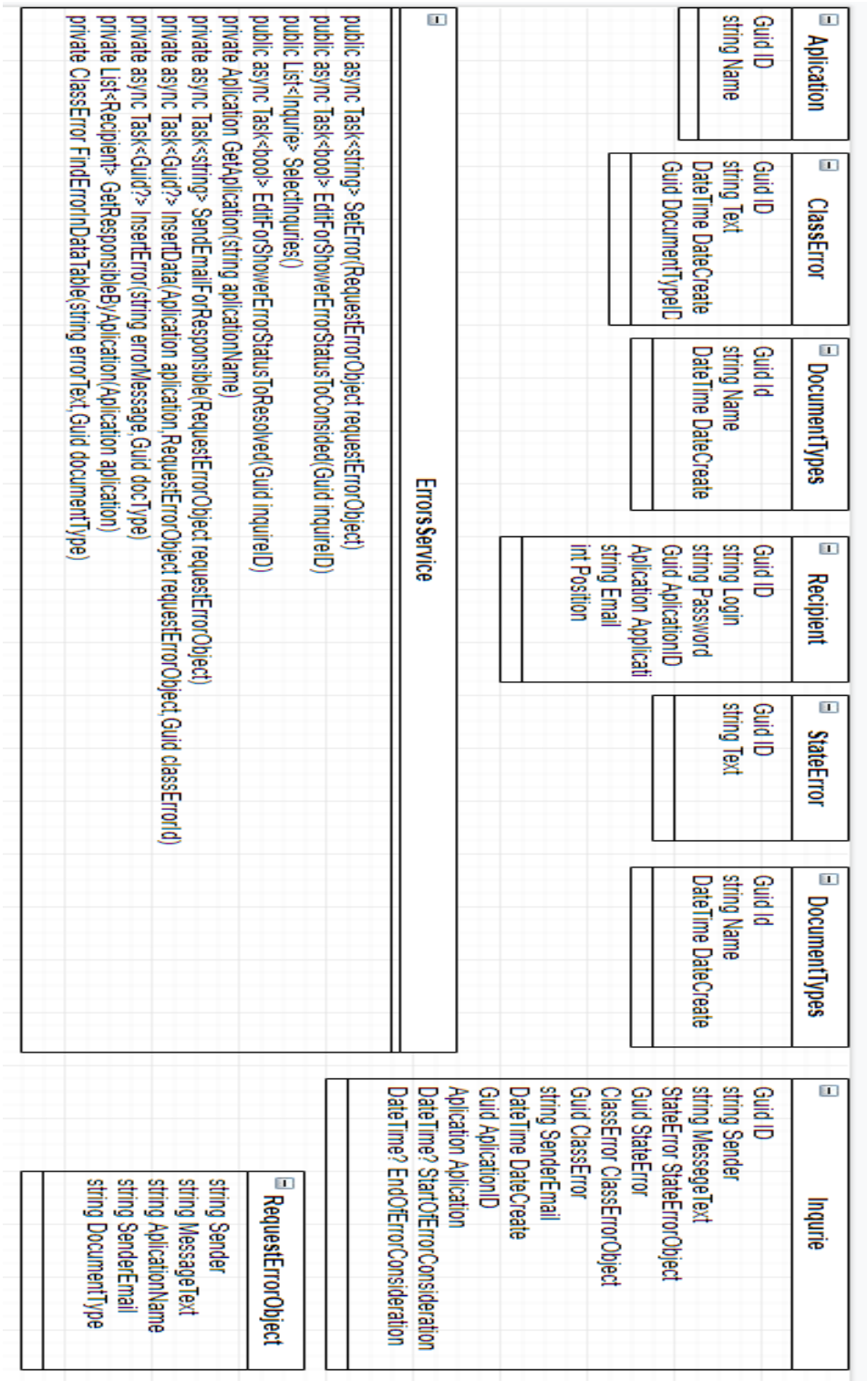

Рис.4.1 Діаграма класів

Всі класи програмного продукта відображені на рис.4.1.

Функції програмного забеспечення:

1. Авторизуватися користувачеві
2. Огляд запитів на основі корпоративного доступу до інформації
3. Зміна статусу запиту на в обробці
4. Зміна статусу запиту на вирішино
5. Пошук вирішення помилки з вже існуючих у БД , на основі тексту помилки і типу документа
6. Повідомлення відповідальних про помилку

Програмний продукт складається з:

1. Контролерів (веб контролер для прийняття даних про запити про помилки і для всіх користувацьких функцій авторизація , відображення списку запитів).
2. Сервіс ErrorsService для всіх функцій з інформацією про запити.
3. Інтерфейс для сервісів IErrorService
4. Моделі для функцій з інформацією для бази даних
5. Моделі прийняття даних веб сервісом.
6. Веб сторінки для авторизації і для відображення даних про запити.

Контроллери зв’язують представлення та моделі чи сервіси для роботи з даними з бази даних.У програмному продукті є 2 контролери ApiErrorsController і HomeController.

ApiErrorsController – контролер який реалізує веб інтерфейс для прийняття даних за допомогою HTTP.

HomeController – контролер для відтворення інформації необхідною користувачу.

ErrorsService – сервіс для відтворення логіки необхідної для відтворення функцій моніторингу. В сервісі відшукуються програми за допомогою

ідентифікатора програми з запиту, відповідальних. Відправлення електронних листів відповідальних і т.д.

IerrorService – інтерфейс класу для опису функцій сервісу(використовується для чистоти коду).

Моделі для баз даних – Aplication, DocumentTypes, Inqurie, Recipient, StateError

Моделі для прийняття даних веб сервісами – RequestErrorObject

Веб сторінки – cshtml файли.

Так как програмний продукт REST API для тестування веб інтерфейсу та внутрішньої логіки використовується Postman .

Postman – це клієнтський інструмент API, який допомагає перевірити API. Це дозволяє протестувати один і той же запит на різних середовищах із змінними для оточення.

На рис.4.2 описано алгоритм який реалізовано в програмному продукті моніторингу системи документообігу. Алгоритм описує основні функції системи моніторингу , та на ньому показано основний функціонал системи моніторингу. Алгоритм на рис. 4.2 реалізований в контролені ApiErrorsController та сервісі ErrorsService.

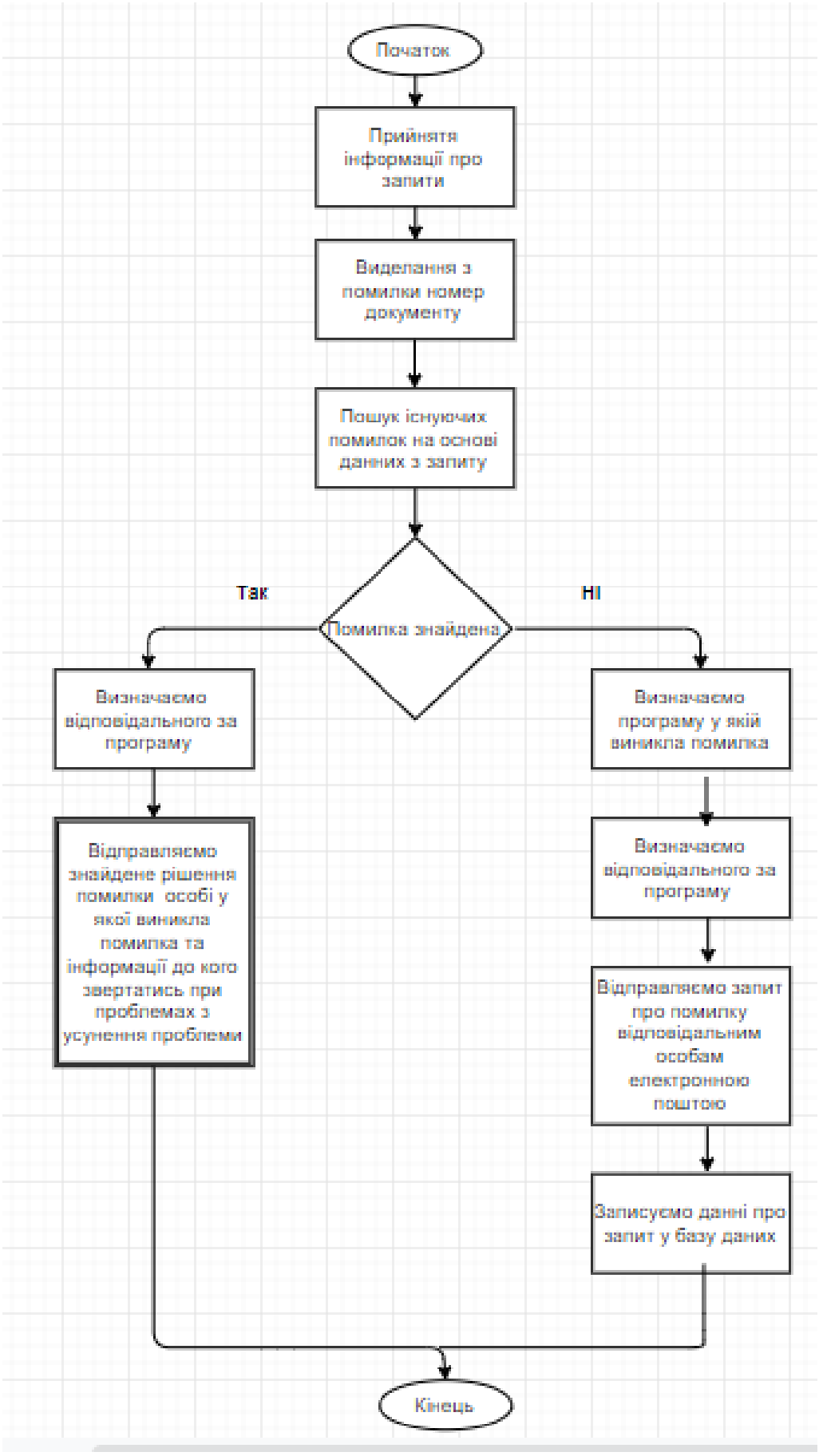

Рис.4.2 Алгоритм системи моніторингу документообороту

4.2 Інтерфейс програми.

Інтерфейс (від англ. Interface — поверхня розділу, перегородка) — сукупність засобів, методів і правил взаємодії (управління, контролю і т. д.) між елементами системи. Цей термін використовують у багатьох галузях науки й техніки. Його значення належить до будь-якої сполуки взаємочинних сутностей (як природничих, так апаратних і людино-машинних).

Під інтерфейсом розуміють не тільки пристрої, але й правила (протокол) взаємодії цих пристроїв. У контексті окремого елемента інтерфейс елемента протилежний до реалізації елементу (внутрішнього устрою та функціювання). Користувачеві елемента немає потреби знати, як реалізовано уживаний елемент, щоб керувати ним, але використовуваний елемент має надати інтерфейс керування. Наприклад, водієві зовсім не обов'язково знати, як влаштовано двигун, щоб керувати автомобілем, досить користуватися інтерфейсом автомобіля (кермом і педалями).

Інтерфейси є основою взаємодії в сьогочасних інформаційних системах. Якщо інтерфейс якого-небудь об'єкту (персонального комп'ютера, програми, функції) не змінюється (стабільний, стандартизований), це дає можливість модифікувати сам об'єкт, не перероблюючи його принципи взаємодії з іншими об'єктами.

В обчислювальній системі взаємодія може здійснюватися на користувацькому, програмному й апаратному рівнях. Відповідно до цього інтерфейси можуть існувати як:

- Спосіб взаємодії фізичних пристроїв;
- Спосіб взаємодії віртуальних пристроїв;
- Спосіб взаємодії людина-машина.

Веб-інтерфейс - це сукупність засобів, за допомогою яких користувач взаємодіє з веб-сайтом або веб-застосунком через браузер.

Веб-інтерфейси отримали широке поширення у зв'язку із зростанням

популярності всесвітньої павутини і відповідно повсюдного розповсюдження веб-браузерів.[9]

Однією з основних вимог до веб-інтерфейсів є їх однаковий зовнішній вигляд і однакова функціональність при роботі в різних браузерах.

Інтерфейсна частина дозволяє користувачеві авторизуватися на основі корпоративних даних .рис 4.3 та продивлятися таблицю з запитами рис.4.4. Авторизація відбувається на основі корпоративних даних отриманих з бази даних компанії за допомогою сервісів, на основі куки аутентифікації. Сервісами отримуються дані про логін і пароль користувача.

Дані для таблиці запитів беруться зі створеної для програмного продукту моніторингу бази даних.

Будується таблиця за допомогою функціоналу NonFactors.Grid.Core.Mvc6 це бібліотека для побудови користувацьких таблиць на веб сторінці.

Веб інтерфейс Web api має методи:

- SetError – для прийняття запитів,
- EditStatusErrorShowerResolved – для зміни статусу запиту на вирішений, приймає ідентифікатор запиту.
- EditStatusErrorShowerConsided – для зміни статусу запиту на у розгляді , приймає ідентифікатор запиту..

Для тестування цих методів , використовується програмний продукт постман рис.4.5.

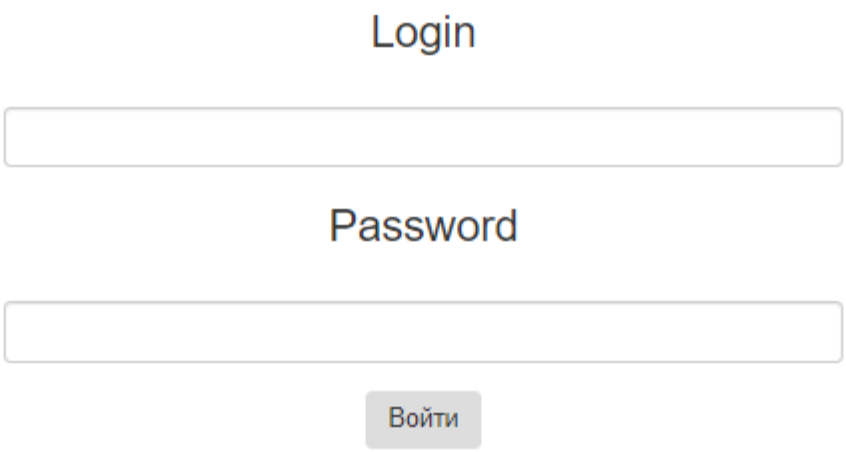

Рис.4.3 Форма для авторизації

ID	Відправник	Програма	Дата запиту	Статус помилки	Тип документу	Відповідальні	Текст помилки	Текст рішення	Початок розгляду	Кінець розгляду
eb14e8d4-31b3-4284-addd-220293971721	Nataliya Igovska	Programa3	28.11.2019	Error resolved	Transport Invoice	Tereschenko Vlad;Duein Dgons	Не виставлений водій №81542	Внесіть водія в документ	29.11.2019 09.26.35	30.11.2019 03.26.35
56e4c178-9bf0-4f06-b382-2b96bcf5564b	Ekaterina Grishko	Web Site	28.11.2019	The error is considered	Prices	Sergeev Sergey	Не правильно виставлені ціни №19542		05.12.2019 01.26.35	
90a908c8-010c-4b6d-8569-2ed15d9164f1	Elizaveta Ashaeva	Programa2	28.11.2019	Error sent	Order	Taram igus	Неправильно заповнений отримувач №41242			
2d4d728b-973c-4986-bde4-354d2e11d68a	Ageev Valentin	Web Site	28.11.2019	Error resolved	Mobile Order	Sergeev Sergey	Не виставлені критерії доставки №31502	Внесіть торгову точку для доставки	04.12.2019 09.26.35	05.12.2019 09.26.35
75a70f01-6262-4565-8bd4-5318ba9a7d65	Elizaveta Ashaeva	Programa2	28.11.2019	Error sent	Order	Taram igus	Помилка в цінах в замовлені 12345			
d33599f7-d6ce-4ee1-bc19-93859541565a	Petrov Petr	Programa1	22.11.2019	Error sent	Order	Ivanov Ivan;Valentin Ivan	Test Error4			
3cce4139-b1ea-4696-8b6e-a9b714529fa1	Ekaterina Grishko	Web Site	28.11.2019	Error sent	Prices	Sergeev Sergey	Не правильно виставлені скидки №11542			
99465558-45bb-426e-9bf1-c27a0300fb93	Pavel Ashaev	Sales Services	28.11.2019	The error is considered	Mobile Order	Antonina Ivashenko	Цінності внесені неправильно №81542		07.12.2019 01.26.35	
9c3afd63-deec-480f-a9cc-c6ce82843049	Nataliya Igovska	Programa3	28.11.2019	Error sent	Transport Invoice	Tereschenko Vlad;Duein Dgons	Не внесений номер машини №89542			
364db3b9-187e-4f1d-9355-c902c611a0d1	Ageev Valentin	Programa1	28.11.2019	Error resolved	Sale	Ivanov Ivan;Valentin Ivan	Не роспізнані позиції №11502	Внесіть данні по продукції з бази данних клієнта	04.12.2019 12.26.35	08.12.2019 09.26.35
e6f7ebca-fa44-4582-9fcb-fae32019ab95	Elizaveta Ashaeva	Sales Services	28.11.2019	The error is considered	Mobile Order	Antonina Ivashenko	Неправильна дата продажів №81542		02.12.2019 09.30.35	

« ‹ 1 › » All ▼

Рис.4.4 Таблиця з данними з бази даних

Рис.4.5 Запит з програми для тестування веб сервісів

ВИСНОВКИ

1.На основі дослідження структури КК було визначено що систему документообігу необхідно моніторити так як помилки можуть призвести до великих фінансових втрат.Так як моніторинг документообігу допоможе суттєво зменшити фінансові витрати або взагалі їх усунути.

2.Аналіз методів пошуку рядка в рядці показав що най оптимальнішим для порівняння нових помилок з вже записаних буде метод взятий з серидовища розробки.

3.Проаналізувшувавши існуюче програмне забезпечення було визначено що існуючі програмні продукти забезпечуючі функціонування системи моніторингу у КК не виконують функцій моніторингу помилок у документах.Також було визначено що в системі документообігу не вистача своєчасного повідомлення відповідальних осіб проведення документів про виникнення помилок.

4.Огляд системи вилову помилок у системі документообігу показав що вся необхідна інформація для збереження повноти інформації про помилки може отримуватися з системи визначення помилок.

5. Для збереження інформації про помилки була розроблена база даних необхідна, з якою взаємодіє система моніторингу документообігу

6. Було розроблено систему моніторингу системи документообігу , з най оптимальнішим алгоритмом пошуку помилки в існуючій базі даних зв’язаної з системою моніторингу документообігу

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Документообіг як складова документного забезпечення управлінської діяльності організацій [Електронний ресурс] – Режим доступу до ресурсу: https://sites.google.com/site/dokumentoobigvustanovi/home/dokumentoobig-ak-skladova-dokumentnogo-zabezpecenna-upravlinskoie-dialnosti-organizacij.

2. Web API [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Web_API.

3. Система автоматизації документообігу [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Система_автоматизації_документообігу.

4. Прямий пошук рядка [Електронний ресурс] – Режим доступу до ресурсу: http://studepedia.org/index.php?vol=1&post=111536.

5. Алгоритм Кнута — Морріса — Пратта [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Алгоритм_Кнута_—_Морріса_—_Пратта.

6. Алгоритм Бойєра — Мур [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Алгоритм_Бойєра_—_Мура.

7. Матвієнко О. Основи організації електронного документообігу / О. Матвієнко, Ц. Михайко. – Київ: Центр навчальної літератури. – К.: 2011. – 112 с.

8. Testing with Postman [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://wiki.onap.org/display/DW/Testing+with+Postman.

9. Веб-інтерфейс [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Веб-інтерфейс.

10. Липчук В. В. Маркетинг: навч. пос. / В. В. Липчук, Р. П. Дудяк, С. Я. Бугіль, Я. С. Янишин. - Львів: Магнолія 2006”, 2012. - 456с

11. Алгоритм Рабіна — Карпа [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Алгоритм_Рабіна_—_Карпа.

12. Microsoft Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio.

13. Логістика : [навч. посіб. для студ. вищ. навч. закл.] / Бержанір А. Л., Рибчак В. І., Слободяник Н. П. ; Уман. держ. аграр. ун-т. — Умань (Черкас. обл.) ; Уман. вид.-поліграф. п-во, 2009. — 347 с. : іл., табл. ; 21 см. — Присвяч. 165-річчю заснування ун-ту. — Бібліогр.: с. 344—347 (56 назв). — 200 пр.

14. Що таке логістика: визначення [Електронний ресурс] – Режим доступу до ресурсу: https://moyaosvita.com.ua/menedzhment/shho-take-logistika-viznachennya/.

15. Албахарі Д. C# 7.0. Справочник. Полное описание языка / Д. Албахарі, Б. Албахарі., 2016. – 1024 с.

16. Прайс М. C# 7 и. NET Core. Кросс-платформенная разработка для профессионалов / М. Прайс., 2018. – 640 с.

17. Internet Information Services [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Internet_Information_Services.

18. Введение в Web API [Електронний ресурс] // 2015 – Режим доступу до ресурсу: https://metanit.com/sharp/mvc/12.1.php.

19. Microsoft SQL Server [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server.

20. Об'єктно-орієнтоване програмування [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Об%27єктно-орієнтоване_програмування.