КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій

_____

(факультет)

Інформаційних технологій

_____

(кафедра)

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ

НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Система обліку працівників і заробітньої плати для комунальників»

Щербіна Костянтин Віталійович

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

Київ 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій

_____
(факультет)

Інформаційних технологій

_____
(кафедра)

ЗАТВЕРДЖУЮ
Завідувач кафедри Інформаційних технологій
Хроленко В. М.
_____
„___” _____2024 року

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ

НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Автоматизована інформаційна система обліку та управління особистими фінансами»

Виконав: студент 4–го курсу, групи КНс–21
Спеціальності: 122 «Комп`ютерні науки»
Спеціалізація: _« Система обліку працівників і заробітньої плати для комунальників»
(шифр і назва напряму підготовки, спеціальності)
Щербіна К. В.
(прізвище та ініціали)
Керівник Хроленко В. М.
(прізвище та ініціали)
Рецензент _____
(прізвище та ініціали)

Київ 2024 р

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: __автоматизації і інформаційних технологій
Кафедра:__ __інформаційних технологій
Освітній рівень: «бакалавр» за ОП
Спеціальність:_122 «Комп`ютерні науки»
Спеціалізація:_Інформаційні управляючі системи і технології.

ЗАТВЕРДЖУЮ

Завідувач кафедри Інформаційних технологій
Хроленко В. М.

_____
„_19_” ___січня___2024 року

ЗАВДАННЯ
ДО ВИКОНАННЯ АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

Щербіна Костянтин Віталійович

Тема роботи: Система обліку працівників і заробітньої плати для комунальників

затверджена наказом ректора КНУБА № ____ від «17» листопада 2024 р.

2. Керівник роботи: Хроленко Володимир Миколаєвич
кафедри інформаційних технологій проектування і прикладної математики

3. Строк подання студентом роботи до захисту:_____

4. Зміст пояснювальної записки за розділами:

Р.1. Аналіз та дослідження проблеми

Р.2. Проєктуванняінформаційного забезпечення

Р.3. Практична реалізація

5. Інформаційні слайди:

- TDB

6. Календарний план виконання атестаційної випускної роботи

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз та дослідження проблем	Лютий 2024 р.
Р. 2. Проєктування інформаційного забезпечення забезпечення	Квітень 2024 р.
Р. 3. Практична реалізація	Квітень 2024 р.
Остаточне оформлення роботи	Травень 2024 р.
Направлення роботи на рецензування	Червень 2024 р.
Попередній захист роботи на кафедрі	Червень 2024 р.

8. Дата видачі завдання: 19 січня 2024 р.

Керівник		Хроленко В. М.
	(підпис)	(прізвище та ініціали)
Бакалавр		Щербіна К. В.
	(підпис)	(прізвище та ініціали)

АНОТАЦІЯ

Щербіна К.В. Система обліку працівників і заробітньої плати для комунальників.

Дипломна бакалаврська робота за спеціальністю – «Комп'ютерні науки» – Київський національний університет будівництва та архітектури, Київ, 2024 рік.

Бакалаврську роботу присвячено дослідженню та застосуванню основних тенденцій, принципів та реалізацій інформаційних систем. В рамках дипломного проекту було розроблено систему обліку працівників та їх контроль заробітньої плати, та база даних для цієї системи.

У роботі розглянуто та використано основні принципи та технології розробки автоматизованих інформаційних систем обліку та управління особистими фінансами. Було створено дизайн програмного продукту, розроблено базу даних для зберігання інформації та написано програмний код для обробки і візуалізації збережених даних

Ключові слова є: "інформаційне забезпечення", "особисті фінанси", "база даних", "управління фінансами", "функціональні можливості".

SUMMARY

Shcherbina K.V. System of providing workers and wages for public utilities.

Bachelor's degree work for the specialty - "Computer Science" - Kyiv National University of Civil Engineering and Architecture, Kyiv, 2024.

The bachelor's work is dedicated to the investigation and determination of the main trends, principles and implementation of information systems. As part of the thesis project, a system for the employment of workers, their salary control, and a database for this system was developed.

The work examines the basic principles and technologies of development of automated information systems in the area and management of special finances. A software product design was created, a database was created to save information, and program code was written to process and visualize the saved data.

Key words: “information security”, “finance features”, “database”, “financial management”, “functional capabilities”.

ЗМІСТ

1. Аналіз та дослідження проблем

1.1. Загальні та теоретичні відомості, концепції автоматизованої системи для обліку та управління особистими фінансами

1.2. Постановка та аналіз проблеми

1.3. Дерево основних цілей

1.4. Вимоги та особливості проєктування системи

1.5. Аналіз існуючих рішень

1.5.1. Аналіз існуючих розробок автоматизованих інформаційних систем для обліку та управління особистими фінансами

1.5.2. Аналіз існуючих баз даних

1.6. Постановка задачі

2. Проєктування інформаційної системи

2.1. Ключові питання постановки завдання проєктування інформаційного забезпечення

2.2. Тип програмного забезпечення для інформаційної системи

2.3. Вибір технології та мови програмування для розробки автоматизованої інформаційної системи

2.4. Забезпечення безпеки

2.5. Архітектура інформаційної системи

2.6. Проєктування та розробка графічного інтерфейсу користувача

2.6.1. Вимоги до інтерфейсу

2.6.2. Загальні можливості структури інтерфейсу

2.7. Проєктування моделі бази даних

2.7.1. Створення концептуальної моделі

2.7.2. Створення логічної моделі

2.7.3. Створення фізичної моделі

2.7.4. Загальний вигляд сутностей бази даних

3. Практична реалізація

3.1. Вибір програмного інструментарію

3.1.1. Бази даних

3.1.2. Програмне середовище

3.2. Файлова структура проекту

3.3. Шар бізнес-логіки

3.4. Шар представлення

4. Бізнес-план

4.1. Опис продукту

4.1.1. Вступ

4.1.2. Основні функції та переваги

4.1.3. Цільова аудиторія

4.2. Аналіз ринку

4.2.1. Огляд ринку

4.2.2. Оцінка попиту

4.3. Маркетингова стратегія

4.3.1. Ціноутворення

4.3.2. Канали розповсюдження

4.4. Фінансовий план

4.4.1. Прогноз доходів

4.4.2. Початкові витрати

4.5. 5.

Висновок

ВСТУП

Актуальність дослідження. Тема "Система обліку працівників і заробітньої плати для комунальників" є дуже актуальною в сучасному світі, де технології стають все більш інтегрованими в повсякденне життя людей.

Облік фінансів працівників необхідний для контролю витрат компанї, прибутків, змін у бюджеті. Ці інструменти допомагають планувати бюджет компанії та враховувати робочі потужності, виявляти. Розробка системи обліку працівників і заробітньої плати для комунальників є рішенням для допомоги в управлінні бюджетом комунальних служб.

Отже, система обліку працівників і заробітньої плати для комунальників має великий потенціал для покращення якості виконання своїх обов’язків людей які працюють на комунальних службах. Така тема дипломного проєкту є дуже актуальною та має потенціал для розвитку в майбутньому.

Для визначення напрямку даної роботи, необхідно спочатку визначити об’єкт дослідження і предмет дослідження, а також надати опис, за допомогою яких методів планується проводити дослідження і вирішення проблеми.

Отже, для дослідження і побудови імітаційної моделі підсистеми виділяємо наступні сутності:

- об’єктом дослідження даної роботи є інформаційне забезпечення автоматизованої система обліку працівників і заробітньої плати комунальників.
- предмет дослідження роботи є розробка інформаційного забезпечення автоматизованої системи обліку працівників і заробітньої плати для комунальників. В основі інформаційного забезпечення лежить модель обліку та управління фінансами комунальних служб.

- методи дослідження – на початку необхідно провести дослідження існуючих систем для обліку та управління працівниками та їх заробітньою платою, які є не надто розповсюдженним у використанні. Провести аналіз найкращих практик для побудови зручного, інформаційного та інтуїтивного інтерфейсу користувача. Виділити основі критерії

функціональності інформаційної системи для забезпечення потреб у обліку та управління працівниками та їх фінансами.

Практична значимість. Результати отримані під час дослідження можуть бути використані для реалізації програмного продукту для системи обліку працівників і заробітньої плати для комунальників найкращих практик і тенденцій.

Результати дослідження. Розробка та реалізація автоматизовної інформаційної Система обліку працівників і заробітньої плати для комунальників, що дозволяє легко вести облік фінансів комунальних служб і отримувати звітну інформацію у зручному, зрозумілому графічному вигляді. Результатом є також розробка оптимальної структури баз даних, що забезпечують ефективну обробку, зберігання та взаємодію різних даних в системі.

1 АНАЛІЗ ТА ДОСЛІДЖЕННЯ ПРОБЛЕМ

1.1 Загальні та теоретичні відомості, концепції автоматизованої система обліку працівників і заробітньої плати для комунальників

Зв'язок інформаційних технологій зі змінами в суспільстві та побуті людей необхідно розглядати у контексті тенденцій, що склалися в останні роки. Питання фінансів у комунальних службах не є новим. Люди, що вміють вести облік фінансів комунальників зазвичай вокористовувала традиційні інструменти для обліку різного роду інформації: ручка і папір. На ранніх етапах розвитику інформаційних технологій для виконання обліку певного типу даних була необхідність у наявності кваліфікованого оператора, тобто людини, що безпосередньо виконує облік за допомогою інформаційної системи. З часом інформаційні системи конкретно для обліку фінансів перейшли у, як мінімум, два шляхи - одні для корпорацій, де все ще є необхідність кваліфіковного робітника, бухгалтера, який за допомогою автоматизованої системи повинен вести облік фінансів для робітників. Інформаційні системи для обліку фінансів комунальних робітників повинні мати, перш за все, інтуїтивний інтерфейс, що є простим і зрозумілим для пересічного користувача. Сам облік фінансів теж не повинен створювати складнощів для його виконання. Звіти по фінансовим рухам мають бути зображені у графічному вигляді та надавати мінімальний, але повний об’єм інформації.

Автоматизована інформаційна система для обліку робітників та керування фінансами – це технологія, яка надає можливість автоматизувати та контролювати рух коштів комунальних мереж у максимально спрощеному вигляді.

Однак, під час використання інформаційних систем для керування робітниками та фінансами комунальних робітників гостро стає питання безпеки даних. Дані комунальних мереж, їх фінансові операції, що зберігаються інформаційною системою, можуть бути використані зловмисниками для різних цілей. Ці аспекти програмного продукту повинні бути розглянуті детально.

Тому, метою даної дипломної роботи є детальний аналіз вимог щодо системи для управління та обліку для комунальних мереж та робітників які там працюють та розробка відповідної інформаційної системи.

Для досягнення поставленої мети, в роботі є намір розглянути основні аспекти обліку та управління фінансами комунальних мереж, архітектурні особливості, системи управління, типи та реалізації баз даних, комунікації та безпеки. Також проаналізовані вже існуючі інформаційні системи для обліку працівниками та керуванням фінансами комунальних мереж, що призначені як для маленької мережі, так і для великих комунальних мереж.

Загальна мета дипломної роботи – дослідити методики та тенденції для реалізації системи для обліку та управління особистими фінансами, а саме її інформаційного забезпечення, проаналізувати її особливості та можливості.

1.2 Постановка та аналіз проблеми.

Облік фінансів комунальних робітників необіхдний для контролю надходженню грошей, прибутків, змін у плану виконання. Ці інструменти допомагають розподіляти бюджет, виявляти витрати які були не доцільними і скорочувати їхню кількість і поліпшувати роботу робітників які сильніше заохоченні. Розробка автоматизованої інформаційної системи обліку працівників і заробітньої плати для комунальників є рішенням для допомоги в управлінні бюджетів комунальних робітників.

Поетапний процес розробки інформаційного забезпечення автоматизованої системи обліку працівників і заробітньої плати для комунальників можливо представити у вигляді життєвого циклу, серед яких виділяють наступні основні етапи:

1. Етап підготовки та аналізу;
2. Етап збору та аналізу вимог;
3. Етап архітектурного проєктування;
4. Етап детального проєктування та моделювання;
5. Етап розробки програмного забезпечення;
6. Етап інтеграції та тестування;
7. Етап впровадження та навчання персоналу.
8. Етап супроводу та технічного обслуговування

На першому етапі, підготовки та аналізу, узгоджуються всі деталі майбутньої автоматизованої системи обліку працівників і заробітної плати. Основна мета цього етапу - зрозуміти потреби підприємства, визначити основні проблеми та завдання, які повинна вирішувати система, і сформувати загальне бачення проєкту. Це включає оцінку поточного стану облікових систем, виявлення їхніх недоліків і визначення стратегічних цілей підприємства, які нова система може підтримати. Результатом цього етапу є єдине бачення системи, що значно полегшує подальшу розробку та мінімізує ризики.

Етап збору та аналізу вимог є одним із найважливіших у життєвому циклі розробки системи. На цьому етапі збираються всі необхідні дані для успішної розробки та впровадження системи. Визначаються вимоги користувачів і

функціональні вимоги, проводяться інтерв'ю та опитування з майбутніми користувачами системи. Окремо вивчається нормативно-правова база, щоб система відповідала всім законодавчим вимогам. Результатом цього етапу є сформоване технічне завдання, яке узгоджується з усіма зацікавленими сторонами.

Після того, як стали зрозумілими цілі та завдання системи, можна переходити до етапу архітектурного проєктування. На цьому етапі розробляється загальна архітектура системи, визначаються основні компоненти та їх взаємодія. Обираються технології та інструменти для розробки системи, визначаються мови програмування, бази даних та інші технічні аспекти. Проєктуються бази даних та користувацький інтерфейс, створюються моделі даних і структури бази даних. Архітектурне проєктування завершується розробкою архітектурної схеми системи.

Наступний етап - детальне проєктування та моделювання. На цьому етапі створюються детальні технічні специфікації для кожного функціонального модуля, моделюються бізнес-процеси, які підтримуватиме система, і створюються логічні та фізичні моделі даних. Створюються прототипи ключових компонентів системи, які дозволяють оцінити зручність та функціональність майбутньої системи. Результатом цього етапу є детально опрацьовані технічні специфікації та прототипи, готові до реалізації.

Етап розробки програмного забезпечення включає створення програмного продукту на основі проєктної документації, прототипів, дизайну та архітектури. Розробники пишуть програмний код для кожного функціонального модуля, інтегрують компоненти та підсистеми, забезпечують коректну взаємодію між модулями. Розробка ділиться на дві частини: розробка інтерфейсу користувача та розробка серверної частини. Адміністратори бази даних створюють та наповнюють базу даних. Результатом цього етапу є готовий програмний продукт, який відповідає вимогам, визначеним на попередніх етапах.

На етапі інтеграції та тестування команда перевіряє, чи відповідає розроблена система вимогам, визначеним на етапі збору вимог. Тестування включає перевірку функціональності системи, ефективності її роботи та безпеки.

Виконуються інтенсивні випробування для виявлення та виправлення помилок. Тестування завершується перевіркою готовності системи до випуску.

Після завершення тестування система випускається на ринок і впроваджується на підприємстві. На цьому етапі система розгортається на реальних робочих місцях, налаштовується відповідно до специфіки підприємства, і проводиться навчання персоналу. Організовуються навчальні сесії для користувачів, підготовка навчальних матеріалів та тренінгів. Впровадження системи супроводжується збиранням відгуків від користувачів для подальшого вдосконалення системи.

Останнім етапом є супровід та технічне обслуговування системи. На цьому етапі система підтримується в робочому стані, постійно перевіряється на наявність помилок і проблем. Надається технічна підтримка користувачам, вирішуються технічні проблеми, впроваджуються оновлення та вдосконалення системи на основі зворотного зв'язку від користувачів. Це може включати розширення функціональності, покращення інтерфейсу користувача, підвищення продуктивності та покращення безпеки системи.

1.3 Дерево основних цілей

Основною метою є розробка інформаційного забезпечення автоматизованої системи обліку працівників і заробітньої плати для комунальників. Дерево цілей опису даної мети наведено на рисунку 1.1.

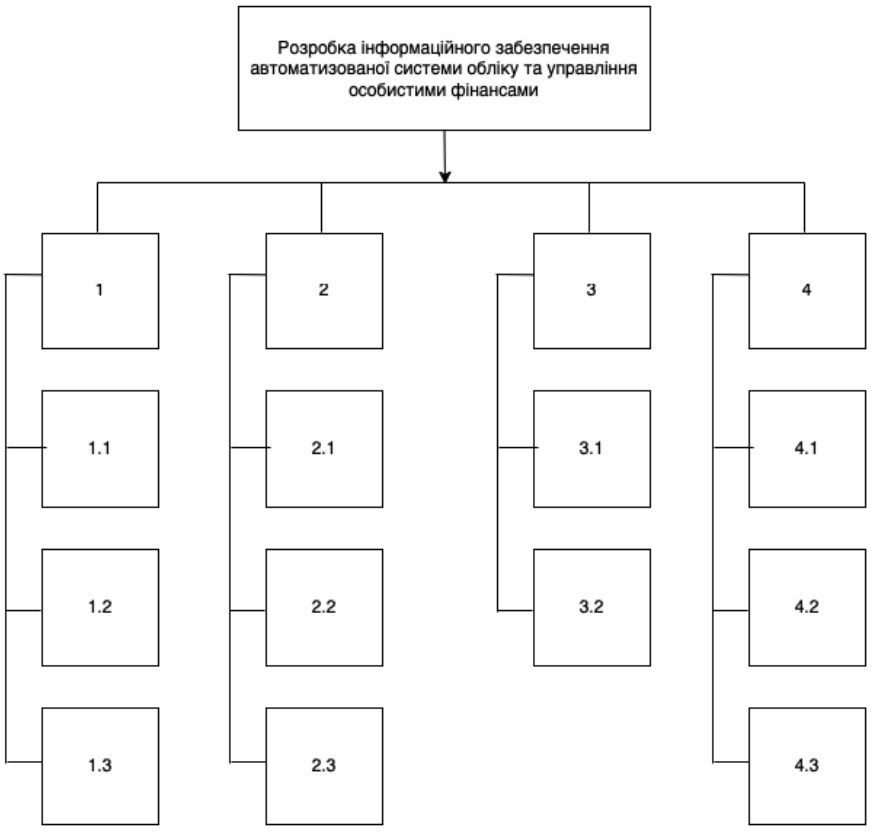

Рисунок 1.1 – Дерево цілей

Функціональні вимоги:

1.1. Зберігання інформації про працівників: електронна пошта, дані для авторизації, ім'я, прізвище, номер телефону, адреса.

1.2. Зберігання інформації про заробітну плату та фінансові операції працівників.

1.3. Аналіз фінансових показників для формування динамічних звітів із застосуванням певних параметрів для інформаційного відображення стану зарплатного фонду.

1.4. Функціональна можливість розподілу фінансових операцій за категоріями (відділами, проектами тощо).

Надійність та безпека:

2.1. Розробка системи автентифікації та авторизації користувачів з високим рівнем доступу.

2.2. Захист даних працівників та інформація про заробітну плату.

2.3. Виявлення та виправлення можливих вразливостей систем безпеки та захисту даних.

Масштабованість та розширюваність:

3.1. Розширення функціональності бази даних для підтримки нових вимог та функцій.

3.2. Розширення кількості збережених даних у міру збільшення кількості працівників та фінансових операцій.

Підтримка та обслуговування:

4.1. Забезпечення підтримки та обслуговування системи обліку всього її життєвого циклу.

4.2. Забезпечення можливості оновлення програмного забезпечення системи обліку.

4.3. Вирішення помилок у разі їх вирішення, оперативне реагування на проблему.

1.4 Вимоги та особливості проєктування системи

Даний проєкт має на меті розробку інформаційного забезпечення для системи обліку працівників і заробітної плати для комунальних служб. Ця система повинна включати функціонал для управління даними про працівників, відображення фінансових даних у вигляді графіків для проведення аналізу. Вона також має забезпечувати відображення інформації за різними критеріями, такими як дата, відділ або категорія витрат. Основна мета системи – надання даних у зрозумілому вигляді для кращого розуміння загального стану заробітної плати та управління персоналом.

Основним елементом системи є база даних, яка забезпечує збереження, обробку та доступ до інформації. Важливі завдання щодо інформаційного забезпечення включають:

Збереження текстових та числових даних про працівників і фінансові операції.

1. Представлення даних у зручному та структурованому вигляді.
2. Швидкий доступ до даних.
3. Можливість внесення змін та доповнень.
4. Забезпечення цілісності та конфіденційності даних.

Для вирішення цих завдань обрана реляційна база даних. Відмінність між реляційними і нереляційними базами даних полягає в способі зберігання інформації. Реляційна база даних зберігає інформацію у вигляді таблиць з чіткою структурою. Вона дозволяє створювати зв’язки між таблицями, що спрощує реалізацію зв’язків між сутностями на рівні бази даних.

Основною причиною вибору реляційної бази даних є те, що вони добре зарекомендували себе і підходять для зберігання структурованих даних. Використання реляційної бази даних у системі обліку працівників і заробітної плати забезпечить ефективне та оптимізоване зберігання й обробку даних, що є важливим для успішної роботи системи.

Для забезпечення конфіденційності даних впровадження системи автентифікації та авторизації є ключовим елементом. Це гарантує, що доступ до конфіденційної інформації матимуть лише авторизовані користувачі.

Такий підхід до проєктування інформаційної системи спрямований на створення ефективної системи для обліку працівників і заробітної плати, використовуючи реляційну базу даних, яка задовольняє всі необхідні вимоги для забезпечення надійного зберігання інформації.

1.5 Аналіз існуючих рішень

1.5.1 Аналіз існуючих розробок автоматизованих інформаційних система обліку працівників і заробітньої плати для комунальників

У сучасному світі автоматизовані інформаційні системи для обліку працівників і заробітної плати займають важливу нишу серед програмних продуктів, що спрямовані на оптимізацію роботи підприємств. Такі системи дозволяють швидко і зручно вести облік працівників, їхніх фінансових операцій і отримувати зрозумілу та чітку картину стану фонду оплати праці. У даному аналізі розглянуто найбільш відомі системи обліку працівників і заробітної плати для комунальних служб.

Система «Workforce»

Система «Workforce» - це інформаційна система, що дозволяє зберігати дані про працівників та їхні заробітні плати за певними категоріями для подальшого відображення даних у графічному вигляді. Система дає можливість аналізувати фонд оплати праці за певний період часу або за певними категоріями. Вона представлена у вигляді веб-додатку та розрахована на середньостатистичне підприємство.

Система «PayrollPlus»

Система «PayrollPlus» представляє собою більш просунутий інструмент для обліку працівників і заробітної плати. Вона надає більш детальний графічний вигляд загального стану фонду оплати праці та дозволяє інтегруватися з іншими системами для синхронізації фінансових даних.

Система «HRManagerPro»

Система «HRManagerPro» розроблена для професійного використання у великих підприємствах. Вона надає можливість розподіляти фінансові операції за відділами, враховувати майбутні виплати, прив’язувати банківські рахунки для синхронізації фінансів. Також система надає зовнішній програмний інтерфейс (API), що дозволяє іншим розробникам використовувати певний спектр готових рішень для обліку працівників і управління заробітною платою.

Ці системи є прикладами успішного впровадження автоматизованих рішень для управління працівниками та їхніми заробітними платами, що сприяє підвищенню ефективності та зручності роботи комунальних служб.

Таблиця 1.1 – опис існуючих розробок автоматизованих інформаційних систем обліку та управління особистими фінансами.

Назва	Країна	Аудиторія	Особливості
Workforce	США	Середнє підприємство	Веб-додаток, просте управління даними про працівників, графічний вигляд заробітної плати.
PayrollPlus	Німеччина	Великі підприємства	Просунутий інструмент для управління заробітною платою. Детальний графічний вигляд фонду.

HRManagerPro	США, Німеччина, Польща, Литва	Професійне використання	Інструмент для обліку працівників та заробітної плати, використовується великими підприємствами.

Порівнюючи різноманітні системи обліку працівників і заробітної плати, можна визначити декілька ключових аспектів, що варто враховувати при розробці власної автоматизованої системи. Майбутня система повинна в повній мірі задовольняти потреби підприємства у зручному та швидкому обліку працівників та їх заробітної плати. Графічний інтерфейс повинен бути інтуїтивно зрозумілим та нести в собі максимально корисну інформацію для ефективного здійснення аналізу фонду оплати праці. Система повинна мати можливість створювати власні категорії фінансових операцій, здійснювати пошук і відображати запитані дані у зрозумілому форматі.

Для обіймання більш широкої аудиторії, система повинна бути кросплатформенною, тому було прийнято рішення розробляти систему у вигляді вебдодатку.

1.5.2 Аналіз існуючих баз даних

У даному розділі розглянуті деякі з найбільш поширених та ефективних рішень баз даних, доступних на ринку. Бази даних є невід'ємною частиною багатьох програмних продуктів та інформаційних систем, і різні системи баз даних надають користувачам можливість зберігати, організовувати та оптимізовувати дані. Ось огляд декількох з них:

Oracle Database: Це потужна та розширювана система управління базами даних, яка підтримує широкий спектр додатків, включаючи бізнес-аналітику та високошвидкісні транзакційні системи. Вона володіє можливостями масштабування та високої доступності. Oracle Database добре підходить для великих підприємств, які потребують надійного та продуктивного рішення для обробки великих обсягів даних.

MySQL: Це одна з найпопулярніших відкритих реляційних систем управління базами даних, яка підтримує широкий спектр додатків, включаючи веб-додатки та електронну комерцію. MySQL відзначається масштабованістю та високою доступністю. Вона часто використовується в середніх та малих підприємствах завдяки своїй простоті у використанні та надійності.

MongoDB: Це нереляційна система управління базами даних, яка дозволяє зберігати та обробляти нереляційні дані. MongoDB підтримує масштабування та високу доступність з використанням вбудованого механізму шарування. Вона ідеально підходить для додатків, які потребують гнучкості у структурі даних, таких як соціальні мережі або інтернет-магазини.

PostgreSQL: Це відкрита реляційна система управління базами даних з розширюваним архітектурним підходом. Вона підтримує масштабування, високу доступність та надає різноманітні інструменти для розробки та управління базами даних. PostgreSQL відома своєю надійністю та багатофункціональністю, що робить її популярною серед розробників і великих підприємств, які потребують комплексних рішень для обробки даних.

Ці системи баз даних представляють собою найкращі практики у сфері управління даними та можуть бути застосовані для розробки інформаційного забезпечення автоматизованої системи обліку працівників і заробітної плати для комунальників. Вибір конкретного рішення залежить від специфіки потреб підприємства та вимог до системи.

Таблиця 1.2 – опис рішень баз даних для розробки автоматизованої інформаційної системи обліку та управління особистими фінансами.

Назва бази даних	Тип бази даних	Мова запитів	Підтримувані операційні системи	Особливості
Oracle Database	Relational	SQL	Windows, Linux, macOS	Надійна платформа, підтримка великих обсягів даних, можливість використовувати більшість функцій SQL,
MySQL	Relational	SQL	Windows, Linux, macOS	Відкритий код, підтримка транзакцій, гарна підтримка індексів
MongoDB	NoSQL	MongoDB	Windows, Linux, macOS	Гнучкість у роботі зі структурованими та неструктурованими даними, горизонтальне масштабування
PostgreSQL	Relational	SQL	Windows, Linux, macOS	Відкритий код, підтримка транзакцій, можливість використовувати різні типи даних, розширені можливості безпеки

Ці рішення баз даних надають різноманітні можливості для розробки автоматизованої системи обліку працівників і заробітної плати. Вибір конкретної

бази даних залежить від потреб підприємства, вимог до масштабованості, надійності, безпеки та обсягів даних, які необхідно обробляти.

За результатами порівняльної характеристики баз даних, що можуть використовуватися в розробці автоматизованої системи обліку працівників і заробітної плати для комунальників, можна зробити висновок, що кожна база даних має свої переваги та недоліки і обирається в залежності від конкретних потреб проекту.

Наприклад, якщо потрібна висока продуктивність та підтримка великої кількості користувачів, то доцільно використовувати PostgreSQL або MongoDB. PostgreSQL пропонує багатий набір функцій, включаючи підтримку транзакцій, розширені можливості безпеки та роботу з різними типами даних, що робить її ідеальною для складних корпоративних систем. MongoDB, зі свого боку, забезпечує гнучкість у роботі зі структурованими та неструктурованими даними і дозволяє легко масштабувати систему горизонтально, що важливо для проектів з великою кількістю змінних даних.

Якщо важливо максимально спростити розробку та налаштування бази даних, можна обрати MySQL. Вона має відкритий код, підтримує транзакції та індекси, що робить її зручною для невеликих і середніх проектів. Oracle Database може бути вибором для великих підприємств, де важлива надійність платформи та підтримка великих обсягів даних, а також можливість використовувати всі функції SQL.

Крім того, вибір бази даних залежить від характеру даних, що зберігаються, та вимог до безпеки та зберігання історії змін. Наприклад, для систем, де потрібно зберігати багато історичних даних або забезпечувати високу безпеку, PostgreSQL або Oracle Database можуть бути кращими варіантами.

Отже, при розробці автоматизованої системи управління та обліку працівників і заробітної плати важливо враховувати всі ці фактори і обирати базу даних, яка краще підходить для конкретного проекту.

1.6 Постановка задачі

Метою дипломного проекту є створення інформаційної системи та бази даних для автоматизованої системи управління та обліку працівників і заробітної плати для комунальних служб. Система дозволить користувачам легко і швидко зберігати дані про працівників та їх фінансові операції. На основі цих даних система будує інформативні графіки з урахуванням різних критеріїв. Реалізація проекту передбачає розробку бізнес-моделі системи, проектування сутностей для зв’язків між ними, розробку сервісного шару, який надає методи для роботи із системою, а також проектування схеми бази даних для зберігання інформації про працівників і заробітну плату. База даних повинна містити дані про працівників, їх фінансові операції, категорії витрат, допоміжні структури для зберігання статистичних даних, що використовуються при побудові графіків. Очікуваними результатами є розроблена автоматизована система для управління та обліку працівників і заробітної плати.

Основні завдання дипломного проекту:

1. Аналіз вимог користувачів у галузі управління працівниками та заробітною платою для визначення потреб і функціональності системи, визначення типів даних, які потрібно зберігати в базі даних.
2. Вибір технології, вивчення особливостей та можливостей обраної технології.
3. Проектування архітектури системи, створення бізнес-моделі системи.
4. Проектування схеми бази даних залежно від моделі, визначення структури таблиць, які будуть зберігатися в базі даних.
5. Створення бази даних.
6. Розробка серверної частини системи з урахуванням архітектурних вимог.
7. Розробка графічного веб-інтерфейсу користувача.

8. Оптимізація, налаштування індексів для підвищення швидкодії пошуку даних.
9. Забезпечення безпеки.
10. Тестування системи.
11. Впровадження системи.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Ключові питання постановки завдання проєктування інформаційного забезпечення

Для написання інформаційного забезпечення автоматизованої системи обліку працівників і заробітної плати необхідно розглянути декілька ключових питань:

Визначити тип розроблюваної системи:

Обраний тип системи – веб-додаток. Вибір веб-додатку обґрунтовується тим, що такий додаток може працювати на смартфонах, планшетах і персональних комп'ютерах. Система може бути використана на будь-якому пристрої, що підтримує роботу веб-браузера.

Вибір технології та мови програмування:

Потрібно визначити технології та мови програмування, які будуть використовуватися для розробки системи. Сюди входить також вибір фреймворку, який буде використаний для розробки. Наприклад, для бекенду можна використовувати Node.js або Django, а для фронтенду – React або Angular.

Забезпечення безпеки:

Питання безпеки є критичним. Необхідно забезпечити захист особистих даних працівників та фінансової інформації. Це може включати шифрування даних, автентифікацію та авторизацію користувачів, захист від SQL-ін'єкцій та інших видів атак.

Розробка архітектури системи:

Потрібно спроектувати архітектуру системи, враховуючи шар представлення (інтерфейс користувача), шар бізнес-логіки та шар бази даних. Це забезпечить розподіл функціональних обов'язків та покращить масштабованість і керованість системи.

Розробка інтерфейсу користувача:

Інтерфейс користувача повинен бути інтуїтивно зрозумілим і зручним. Потрібно забезпечити простоту введення даних та зрозуміле відображення звітів і графіків. Важливо врахувати потреби кінцевих користувачів та забезпечити їм комфортну роботу з системою.

Проектування моделі бази даних:

Концептуальна модель: Описує відносини між сутностями в системі та їх атрибутами. Допомагає визначити, які дані необхідні для системи і як вони пов’язані між собою.

Логічна модель: Описує структуру таблиць та відносини між ними. Допомагає визначити, які таблиці потрібно створити і які взаємозв'язки між ними необхідні.

Фізична модель: Описує, як дані фактично зберігаються на диску або інших засобах зберігання. Визначає типи даних, індекси, ключі та методи зберігання для оптимальної продуктивності.

Тестування та налагодження:

Після завершення розробки програмного забезпечення та баз даних необхідно провести тестування та налагодження системи. Це включає функціональне тестування, тестування безпеки, тестування продуктивності та усунення знайдених помилок. Тестування допоможе забезпечити, що система працює коректно перед її впровадженням.

Ретельне опрацювання цих питань допоможе створити ефективну та надійну автоматизовану систему обліку працівників і заробітної плати, яка буде відповідати вимогам користувачів та забезпечить високий рівень продуктивності та безпеки.

2.2 Тип програмного забезпечення для інформаційної системи

Автоматизована інформаційна система для обліку та управління працівниками і заробітною платою передбачає можливість використання системи звичайним користувачем. Отже, платформа, на якій використовуватиметься програмний засіб, повинна бути доступною. Було вирішено розробити програмне забезпечення у вигляді веб-додатку, який може бути запущений у будь-якому середовищі, що підтримує веб-браузер.

Веб-додаток передбачає, перш за все, мобільну версію системи. Оскільки багато операцій, що здійснюються користувачем, можуть бути зроблені у час, коли доступ до персонального комп'ютера обмежений, пріоритетом для веб-додатку є мобільна версія. Це дозволить користувачам здійснювати облік працівників та заробітної плати за допомогою смартфонів та планшетів. Система також передбачає сторінку для персонального комп'ютера у випадку, коли користувачеві потрібно працювати з даними, використовуючи комп'ютер.

Для забезпечення доступності до даних у будь-якому місці, необхідний варіант клієнт-серверної архітектури, коли сервер є віддаленим. Це дозволить користувачам отримувати доступ до системи незалежно від їхнього місця перебування. Для досягнення найвищого рівня доступності і зручності використання, сервер необхідно розмістити на віртуальному сервері, використовуючи хмарні технології. Хмарні технології забезпечують високу доступність, масштабованість та безпеку даних.

Клієнтом виступає веб-браузер, який може бути запущений на будь-якій підтримуваній системі, будь то смартфон, планшет чи персональний комп'ютер. Такий підхід дозволяє забезпечити максимальну гнучкість і доступність для кінцевих користувачів, що є критично важливим для системи обліку працівників та заробітної плати.

2.3 Вибір технології та мови програмування для розробки автоматизованої інформаційної системи

Для проектування інформаційного забезпечення автоматизованої системи обліку працівників і заробітної плати необхідно обрати мову програмування та технології, які найкраще підходять для вирішення поставлених задач. Вибір мови програмування залежить від різних факторів, таких як складність проекту, доступність ресурсів, вимоги до продуктивності, масштабованості та безпеки.

Цей розділ проектування інформаційного забезпечення для автоматизованої системи обліку працівників і заробітної плати описує ресурсну базу наступних завдань:

Вибір мови програмування для розробки додатків і систем.

Визначення технологій та фреймворку для розробки інформаційної системи.

Розробка структури бази даних та вибір системи управління базами даних (СУБД).

Вибір системи зберігання даних та забезпечення їх безпеки.

Перед вибором мови програмування для реалізації автоматизованої системи обліку працівників і заробітної плати потрібно розглянути доступні варіанти:

Python – це інтерпретована мова програмування високого рівня, яка зазвичай використовується в розробці веб-додатків та наукових обчислень. Python також може бути використаний для створення програмного забезпечення для автоматизованої системи обліку працівників і заробітної плати. Python має читабельний та простий синтаксис, що робить його добрим вибором для розробки прототипів та швидкого прототипування. Для розробки веб-додатків на Python можна використовувати фреймворки Django або Flask.

JavaScript – це мова програмування, яка часто використовується в розробці веб-додатків, але може бути використана й для розробки додатків для автоматизованої системи обліку працівників і заробітної плати. JavaScript дозволяє створювати динамічні та інтерактивні веб-сторінки та додатки. JavaScript має велику кількість бібліотек та фреймворків, що полегшують розробку програмного забезпечення, таких як React, Angular, або Vue.js для фронтенду, і Node.js для бекенду.

Java – це об'єктно-орієнтована мова програмування, що має широке застосування в розробці додатків. Java є платформонезалежною, тому програми, написані на Java, можуть працювати на будь-якій операційній системі. Java підтримує багато фреймворків, бібліотек та інших інструментів, що робить її відмінним вибором для розробки автоматизованої системи обліку працівників і заробітної плати. Для веб-додатків на Java можна використовувати фреймворки Spring або Java EE.

Вибір технології та мови програмування залежить від конкретних вимог проекту, включаючи вимоги до продуктивності, масштабованості та безпеки. Потрібно також враховувати доступні ресурси та експертність команди розробників, щоб обрані технології відповідали можливостям команди та дозволяли ефективно реалізувати поставлені завдання.

Таблиця 3.2 – порівняння мов програмування для використання в практичній реалізації системи

Мова програмування	Переваги	Недоліки
Python	Простий синтаксис, велика кількість бібліотек, хороша продуктивність	Менш ефективний для високопродуктивних додатків

PHP	Велика кількість фреймворків, хороша масштабованість, висока продуктивність	Вимагає великої кількості ресурсів для запуску, менш простий синтаксис.
JavaScript	Легка взаємодія з компонентами веб–додатків, широкі можливості для розробки інтерфейсів	Менш ефективний для розробки масштабних додатків

Отже, вибір мови програмування для реалізації інформаційного забезпечення автоматизованої системи обліку працівників і заробітної плати залежить від різних факторів. Оскільки програмний засіб буде реалізовано у вигляді веб-додатку і наявність клієнт-серверної архітектури необхідна, потрібно визначитись із технологіями та мовою програмування, які будуть використані для реалізації як клієнтської, так і серверної частини системи. Крім того, важливим фактором є наявність бібліотек та фреймворків для розробки додатків, які можуть значно полегшити процес розробки.

Під час вибору мови програмування для реалізації автоматизованої інформаційної системи для обліку працівників і заробітної плати, враховано вимоги до продуктивності, функціональність та зручності розробки, щоб забезпечити ефективну та швидку розробку інформаційного забезпечення та баз даних.

Отже, для реалізації клієнтської частини було обрано мову програмування JavaScript для реалізації динамічних змін на веб-сторінках, HTML для реалізації структури та маркування контенту на веб-сторінках, CSS для стилізації веб-сторінок, розроблених за допомогою HTML. Для реалізації серверної частини було обрано мову програмування PHP із використанням фреймворку Laravel. Вибір фреймворку Laravel обґрунтований тим, що фреймворк допомагає

розробнику сфокусуватись на розробці бізнес-частини програмного засобу, не витрачаючи час на формування певної базової архітектури програми на технічному рівні. Laravel надає велику кількість готових рішень для розробки веб-додатків “із коробки”.

2.4 **Забезпечення безпеки**

Автоматизована інформаційна система обліку працівників і заробітної плати буде реалізована із використанням клієнт-серверної архітектури, що передбачає налаштування віддаленого ресурсу з публічним доступом. Тому необхідна реалізація розподіленого доступу із використанням облікових записів. Кожен користувач повинен зареєструвати свій обліковий запис, після чого всі збережені операції будуть прив’язані до зареєстрованого облікового запису. Авторизація у систему здійснюється за допомогою електронної пошти та паролю. Пароль не повинен зберігатися в базі даних у звичайному вигляді. Перед зберіганням паролю, він повинен бути зашифрованим.

Для шифрування паролів використовується BCrypt, що входить до складу фреймворку Laravel. BCrypt - це алгоритм хешування паролів, призначений для збереження паролів у безпечному форматі. Головна його перевага полягає в тому, що він ускладнює атаки на паролі шляхом повільного обчислення хешу. Це робить BCrypt надійним вибором для збереження паролів у системах, які вимагають високого рівня безпеки. У документації дипломного проекту можна вказати, як саме використовувати BCrypt для збереження та перевірки паролів користувачів у вашій програмі або веб-додатку.

Отже, для забезпечення безпеки даних користувачів і реалізації розподіленого доступу, реалізовується система облікових записів, коли кожен користувач має доступ до свого облікового запису, і всі операції відбуваються у контексті окремого користувача. Авторизація у обліковий запис здійснюється за допомогою електронної пошти та паролю. Пароль зберігається у зашифрованому

вигляді, що означає, що навіть у випадку атаки на базу даних, дані користувачів залишаться у безпеці.

Оскільки система побудована на клієнт-серверній архітектурі і є веб-додатком, необхідне забезпечення безпеки на рівні комунікації клієнта із сервером. Комунікація клієнта із сервером відбувається із використанням протоколу HTTPS.

HTTP (Hypertext Transfer Protocol) — це стандартний протокол для передачі гіпертекстових документів інформації на веб-сайтах. Він використовується для передачі різних видів даних, таких як текст, зображення, відео, аудіо та інші ресурси між веб-серверами і веб-клієнтами (наприклад, веб-браузерами).

HTTPS (Hypertext Transfer Protocol Secure) — це захищена версія протоколу HTTP. Вона використовує шифрування SSL (Secure Sockets Layer) або його спадкоємця TLS (Transport Layer Security), щоб захистити конфіденційні дані, які передаються між веб-сервером і веб-клієнтом. HTTPS забезпечує конфіденційність, цілісність та автентичність даних, що передаються через Інтернет, і робить з’єднання більш безпечним для користувачів. Такий протокол особливо важливий при передачі чутливої інформації, такої як паролі, особисті дані, банківська інформація тощо.

2.5 **Архітектура інформаційної системи**

Автоматизована інформаційна система обліку працівників і заробітної плати є веб-додатком, побудованим на клієнт-серверній архітектурі. Для реалізації клієнтської і серверної частини необхідний певний шаблон, а саме MVC (Model - View - Controller).

MVC, або Model-View-Controller, - це шаблон проектування для розробки програмного забезпечення, особливо популярний у веб-розробці. Він розділяє компоненти програми на три основні частини:

Модель (Model):

Модель представляє дані програми та бізнес-логіку, яка опрацьовує ці дані. Модель не залежить від інших частин шаблону і може взаємодіяти як з контролером, так і з представленням. В контексті нашої системи, модель відповідає за обробку даних про працівників, їх заробітну плату та інші фінансові операції.

Вид (View):

Вид відповідає за відображення даних користувачу та інтерфейс користувача. Він отримує дані з моделі та відображає їх у вигляді, зрозумілому користувачу. Вид також може надсилати користувачеві вхідні дані до контролера для обробки. У нашій системі вид відповідає за відображення інформації про працівників та заробітну плату у вигляді таблиць, графіків і звітів.

Контролер (Controller):

Контролер взаємодіє з користувачем та відповідає за обробку вхідних даних. Він отримує вхідні дані від користувача через представлення, виконує необхідну логіку та змінює стан моделі. Крім того, контролер також відправляє дані у відображення для подальшого відображення користувачеві. У нашій системі контролер обробляє запити користувачів, такі як додавання нового працівника, оновлення інформації про заробітну плату та інші операції.

Основна перевага MVC полягає у тому, що він дозволяє розділити логіку програми на окремі компоненти, що спрощує розробку, тестування та підтримку коду. Крім того, цей підхід дозволяє змінювати один компонент (наприклад, вигляд) без впливу на інші компоненти (модель або контролер), що полегшує розширення та зміни програми.

Отже, система є веб-додатком, побудованим за шаблоном MVC, і цей же додаток грає роль сервера. Клієнтом виступає веб-браузер. Архітектура системи у загальному вигляді має наступний вигляд:

Клієнтська частина (Front-end):

Реалізована за допомогою JavaScript, HTML та CSS.

Відповідає за взаємодію з користувачем, відображення даних та отримання введених даних від користувача.

Взаємодіє з серверною частиною через API-запити.

Серверна частина (Back-end):

Реалізована за допомогою PHP із використанням фреймворку Laravel.

Відповідає за обробку запитів від клієнтської частини, виконання бізнес-логіки та взаємодію з базою даних.

Використовує MVC-шаблон для організації коду та розподілу обов'язків.

База даних:

Використовується для зберігання даних про працівників, їх заробітну плату та інші фінансові операції.

Реалізована за допомогою реляційної бази даних, наприклад MySQL або PostgreSQL.

Інтегрована із серверною частиною для зберігання та отримання даних.

Комунікаційний рівень:

Використання протоколу HTTPS для забезпечення безпеки передачі даних між клієнтською та серверною частинами.

Аутентифікація та авторизація користувачів за допомогою токенів або сесій

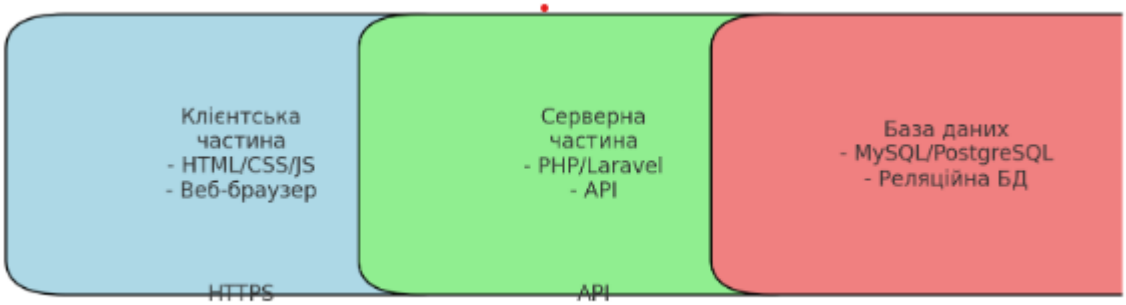

Рисунок 2.1 - Загальна архітектура інформаційної системи

На рисунку 2.1 відображено загальну архітектуру інформаційної системи. Схема починається з клієнта (веб-браузеру), який звертається до сервера за допомогою HTTPS протоколу. Сервером є додаток, побудований за допомогою фреймворку Laravel. Laravel спрощує процес розгортання та конфігурування застосунків, надаючи вбудовані засоби для швидкої розробки. Він використовує конвенції над конфігурацією, що дозволяє розробникам швидше створювати додатки, орієнтовані на бізнес.

Основні переваги Laravel включають:

Швидке створення застосунків: Laravel дозволяє швидко створювати додатки з мінімальною конфігурацією.

Вбудований контейнер: Laravel постачається з вбудованими контейнерами для управління залежностями, що спрощує розгортання застосунків.

Автоконфігурація: Laravel надає автоматичну конфігурацію на основі залежностей та звичайних конфігураційних стандартів.

Компоненти: Laravel має багато вбудованих компонентів для різних задач, таких як кешування, безпека, доступ до даних тощо.

Підтримка вбудованих баз даних: Laravel має підтримку для вбудованих баз даних, таких як SQLite, що дозволяє швидко розпочати розробку без необхідності налаштування зовнішнього сервера баз даних.

Узагальнюючи, Laravel - це потужний інструмент для розробки PHP-додатків, який спрощує процес розгортання та розвитку застосунків, дозволяючи розробникам швидше створювати ефективні програми.

Рухаючись по архітектурі системи далі, можна виділити три основних шари веб-додатку: шар вигляду, шар контролера та шар моделі.

Шар вигляду (View):

Шар вигляду представляє собою набір HTML-сторінок із використанням CSS для стилізації та JavaScript для реалізації динамічних змін на веб-сторінках. Фактично, шар вигляду - це графічний інтерфейс користувача.

Шар контролера (Controller):

Шар контролера відповідає за обробку запитів від користувача, отриманих через інтерфейс. Контролер отримує введені дані від користувача, виконує необхідну бізнес-логіку, звертається до моделі для отримання або зміни даних, і передає оновлені дані назад до вигляду для відображення. Контролер також відповідає за валідацію даних, що надходять від користувача, і перетворення бізнес-даних у формат, придатний для відображення у вигляді.

Шар моделі (Model):

Модель представляє бізнес-логіку та відповідає за доступ до даних. Вона взаємодіє з базою даних, виконуючи запити на отримання, оновлення, видалення та додавання даних. Модель також забезпечує логіку, необхідну для маніпулювання цими даними.

Детальна архітектура шару представлення

Шар представлення є проміжним шаром між інтерфейсом користувача та бізнес-логікою. Він включає наступні компоненти:

Контролери: Відповідають за обробку HTTP-запитів, виконують бізнес-логіку та взаємодіють з моделлю. Наприклад, контролер може обробляти запит на створення нового працівника, валідувати введені дані, зберігати їх у базі даних за допомогою моделі та повертати відповідь клієнту.

Шаблони: Використовуються для генерування HTML-коду, що відображається у браузері користувача. Шаблони можуть містити динамічний контент, отриманий від моделі.

Міжпрограмні засоби (middleware): Використовуються для обробки запитів та відповідей на проміжних етапах. Наприклад, вони можуть здійснювати перевірку автентифікації користувачів або логування запитів.

Архітектура клієнтської частини також включає різні компоненти, які забезпечують взаємодію користувача з системою:

HTML: Використовується для створення структури веб-сторінок.

CSS: Використовується для стилізації веб-сторінок, забезпечуючи привабливий і зручний інтерфейс.

JavaScript: Використовується для реалізації динамічних змін на веб-сторінках та взаємодії з серверною частиною через AJAX-запити.

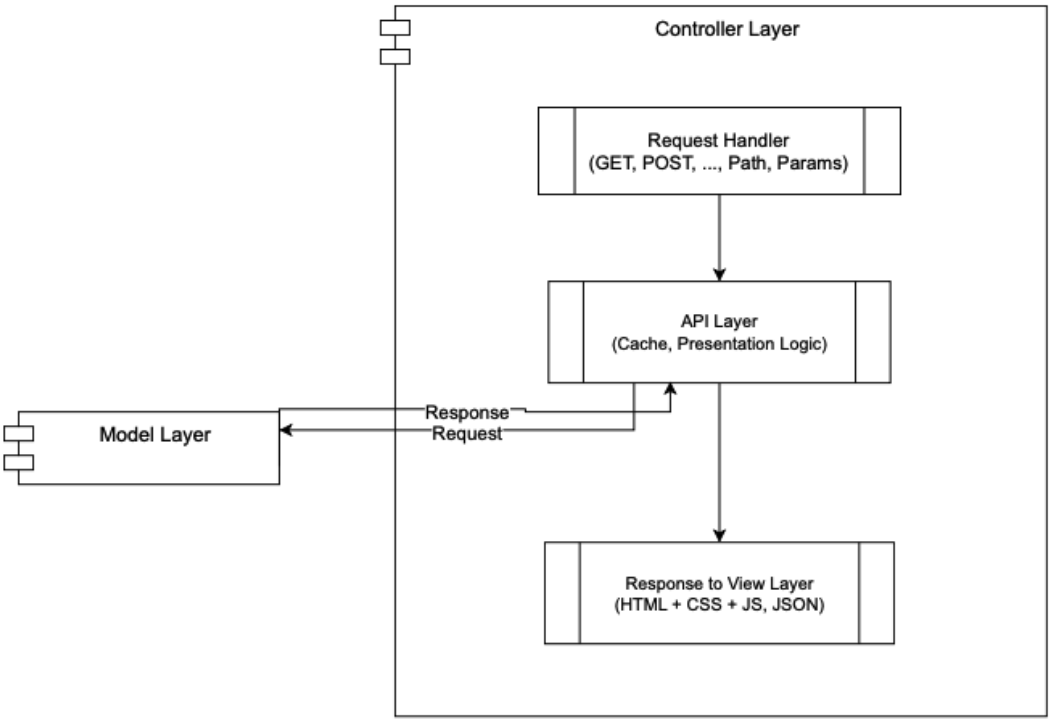

Рисунок 2.2 - Архітектура шару представлення

Шар моделі - це шар бізнес-логіки. Даний рівень побудований на сервісно орієнтованій архітектурі.

Сервісно орієнтована архітектура (SOA) - це підхід до розробки програмного забезпечення, в якому функціональність програми організована у вигляді набору незалежних компонентів, які виконують обмежені функції та мають чітко визначений інтерфейс для взаємодії з іншими компонентами. Кожен компонент (або "сервіс") може бути розглянутий як окрема функціональна одиниця, яка може бути використана іншими компонентами програми через мережу або інші засоби комунікації.

Основна ідея SOA полягає в тому, щоб розбити програму на менші, самодостатні сервіси, які можна легко розвивати, масштабувати та повторно використовувати. Кожен сервіс може бути реалізований, використовуючи будь-яку технологію чи мову програмування, і мати свою власну базу даних та бізнес-логіку. В нашому випадку, сервісно орієнтована архітектура реалізована на логічному рівні.

Переваги SOA включають підвищену гнучкість, орієнтованість на бізнес-потреби, зменшення залежності між компонентами, можливість повторного використання, а також легшу інтеграцію з існуючими системами. Однак для успішної реалізації SOA необхідно правильно спроектувати сервіси та їх інтерфейси, а також забезпечити ефективну систему управління конфігураціями та моніторингу.

Саме на цьому рівні реалізований шар бази даних та безпосередньо зберігання даних та реалізація підключення до системи управління базами даних. Архітектура шару бізнес-логіки:

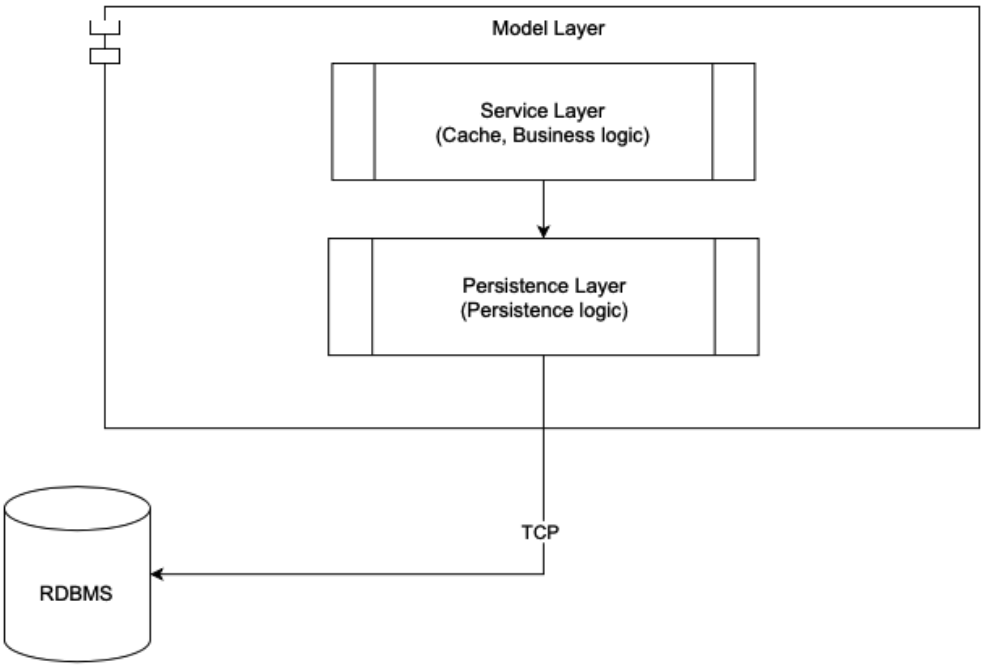

Рисунок 2.3 - Архітектура шару моделі

2.6 **Проєктування та розробка графічного інтерфейсу користувача**

2.6.1 **Вимоги до інтерфейсу**

Для розробки інформаційного забезпечення та баз даних системи обліку працівників і заробітної плати необхідний інтерфейс, який дозволяв би легко та зручно взаємодіяти з даними та переглядати звіти у зручному графічному вигляді. Основні вимоги до інтерфейсу створення інформаційного забезпечення мають бути такі:

Легкість використання:

Інтерфейс повинен бути простим та зрозумілим для будь-якого користувача, навіть без спеціальних технічних знань. Легкість використання забезпечує мінімізацію часу на навчання та зменшення кількості помилок при введенні даних.

Функціональність:

Інтерфейс повинен забезпечувати можливість додавання, видалення та редагування даних про працівників та фінансові операції. Користувач повинен мати змогу переглядати зміни у вигляді графіків, що надає можливість аналізувати дані більш ефективно.

Зручність відображення даних:

Інтерфейс повинен мати зручну та логічну структуру відображення даних, що дозволяє користувачеві легко орієнтуватися та знайти потрібну інформацію. Важливо, щоб дані були представлені в зручному для сприйняття вигляді.

Адаптивність:

Інтерфейс повинен бути адаптивним і мати правильний вигляд на будь-якому пристрої, що підтримує роботу веб-браузера. Це забезпечить доступність системи для користувачів на різних пристроях, включаючи смартфони, планшети та персональні комп'ютери.

Підтримка стандартів:

Інтерфейс повинен дотримуватися стандартів програмування, що дозволить його легко інтегрувати з іншими системами та розробляти додаткові функції. Дотримання стандартів також забезпечує стабільність та надійність системи.

Загалом, інтерфейс створення інформаційного забезпечення має бути зрозумілим, зручним та наділеним додатковими функціями. Окрім цього, важливо, щоб інтерфейс був адаптивним. Веб-сторінка, відкрита на будь-якому

девайсі, що підтримує веб-браузер, повинна виглядати правильно та бути повною в функціональному об'ємі.

На даному етапі було вирішено розробити прототип дизайну інтерфейсу користувача для дизайну структури додатку. Для кожного компоненту дизайн буде реалізовано більш детально вже безпосередньо під час його фактичної реалізації.

Проєктування інтерфейсу

Для розробки графічного інтерфейсу користувача системи обліку працівників і заробітної плати, враховуючи вищезазначені вимоги, буде створено прототип, який включатиме наступні основні елементи:

Головна сторінка:

Вітальне повідомлення та коротка інформація про систему.

Навігаційне меню для швидкого доступу до основних розділів системи (наприклад, додавання нових працівників, перегляд звітів, налаштування облікового запису).

Форма для додавання/редагування працівників:

Поля для введення даних про працівників, такі як ім'я, прізвище, контактна інформація, позиція, заробітна плата тощо.

Кнопки для збереження змін та скасування дій.

Таблиця з працівниками:

Відображення списку працівників з можливістю сортування та фільтрації даних.

Кнопки для редагування та видалення записів.

Графіки та звіти:

Інтерактивні графіки для візуалізації даних про заробітну плату та інші фінансові показники.

Звіти про зміни у складі працівників та фінансових операціях.

Налаштування облікового запису:

Можливість зміни особистих даних, паролю та налаштувань системи.

Прототип інтерфейсу користувача

На даному етапі розробки був створений прототип інтерфейсу користувача для демонстрації основних елементів та їх розташування. Прототип дозволяє отримати загальне уявлення про вигляд системи та взаємодію користувача з нею. Подальша деталізація і реалізація інтерфейсу будуть виконані під час фактичної розробки..

2.6.2 **Загальні можливості структури інтерфейсу**

Для забезпечення зручного та інтуїтивно зрозумілого інтерфейсу розробки інформаційного забезпечення автоматизованої інформаційної системи обліку та управління фінансами комунальних робітників, його структура може включати наступні компоненти:

1. Панель навігації: це важливий елемент інтерфейсу, що дозволяє користувачам легко переключатись між різними розділами програми та виконувати необхідні дії. Панель навігації може містити посилання на такі розділи, як "Співробітники", "Регіони", "Створення івенту", "Підтвердження зарплатні". Панель навігації розміщена зверху і є зафіксованою. Прототип дизайну панелі навігації:

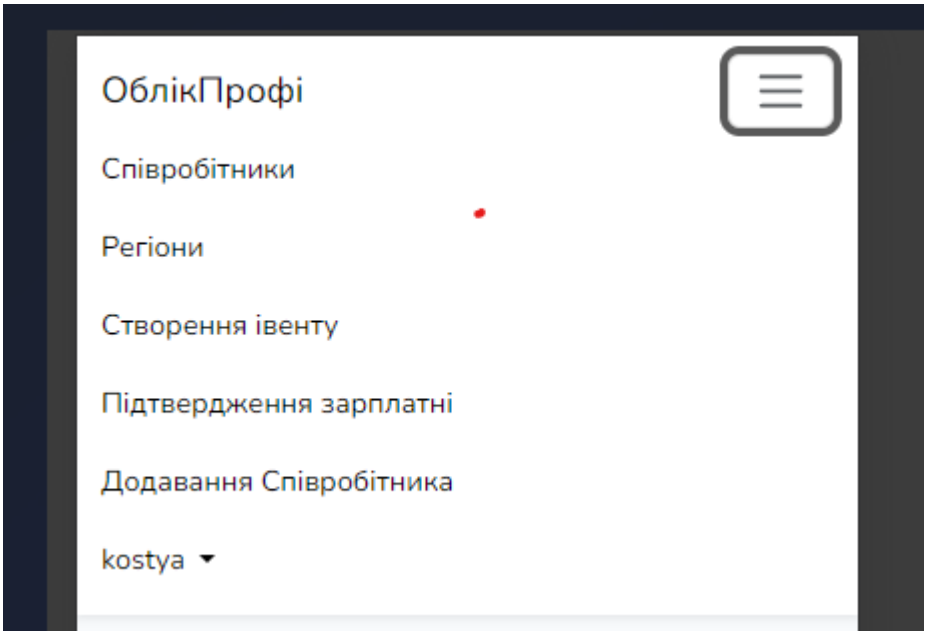

Рисунок 2.4 - Прототип дизайну для компоненту "Панель навігації".

2. Розділ "Регіони": цей розділ містить загальну інформацію про створені райони до яких можна прив’язувати співробітників. Основні функції які можна виконувати на цьому етапі – це видалення, редагування району, пошук по назві району і створення нового

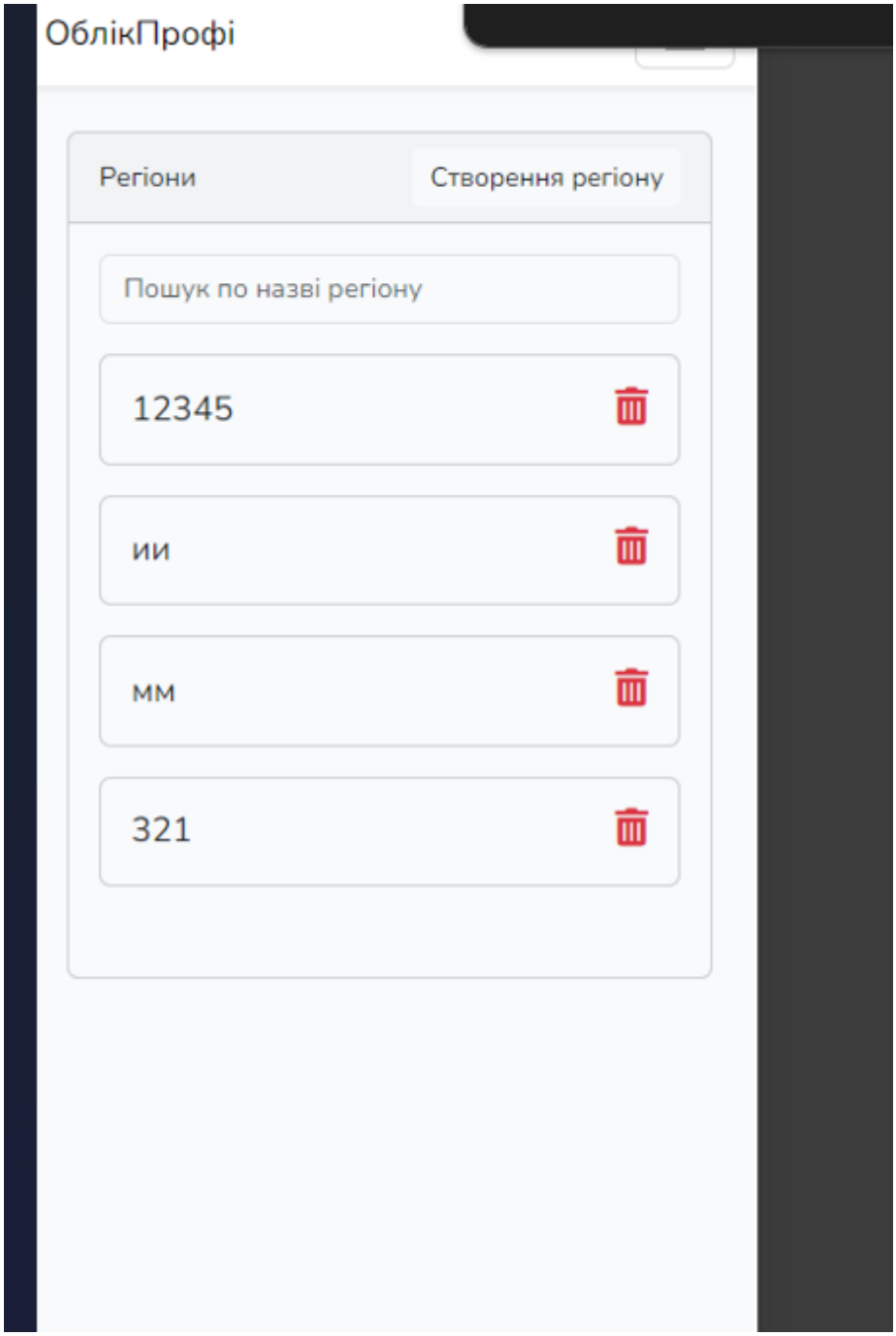

Рисунок 2.5 - Прототип дизайну для компоненту "Район"

3. Розділ "Співробітники" містить форму для створення нового співробітника та редагування інших, сторінка ця доступна тільки для двух ролей, а саме адміністратор та адмін. Адміністратор може переглядати тільки адміністраторів і користувачів свого району а адмін може переглядати всіх користувачів. Є пошук по унікальному полю – пошта користувача

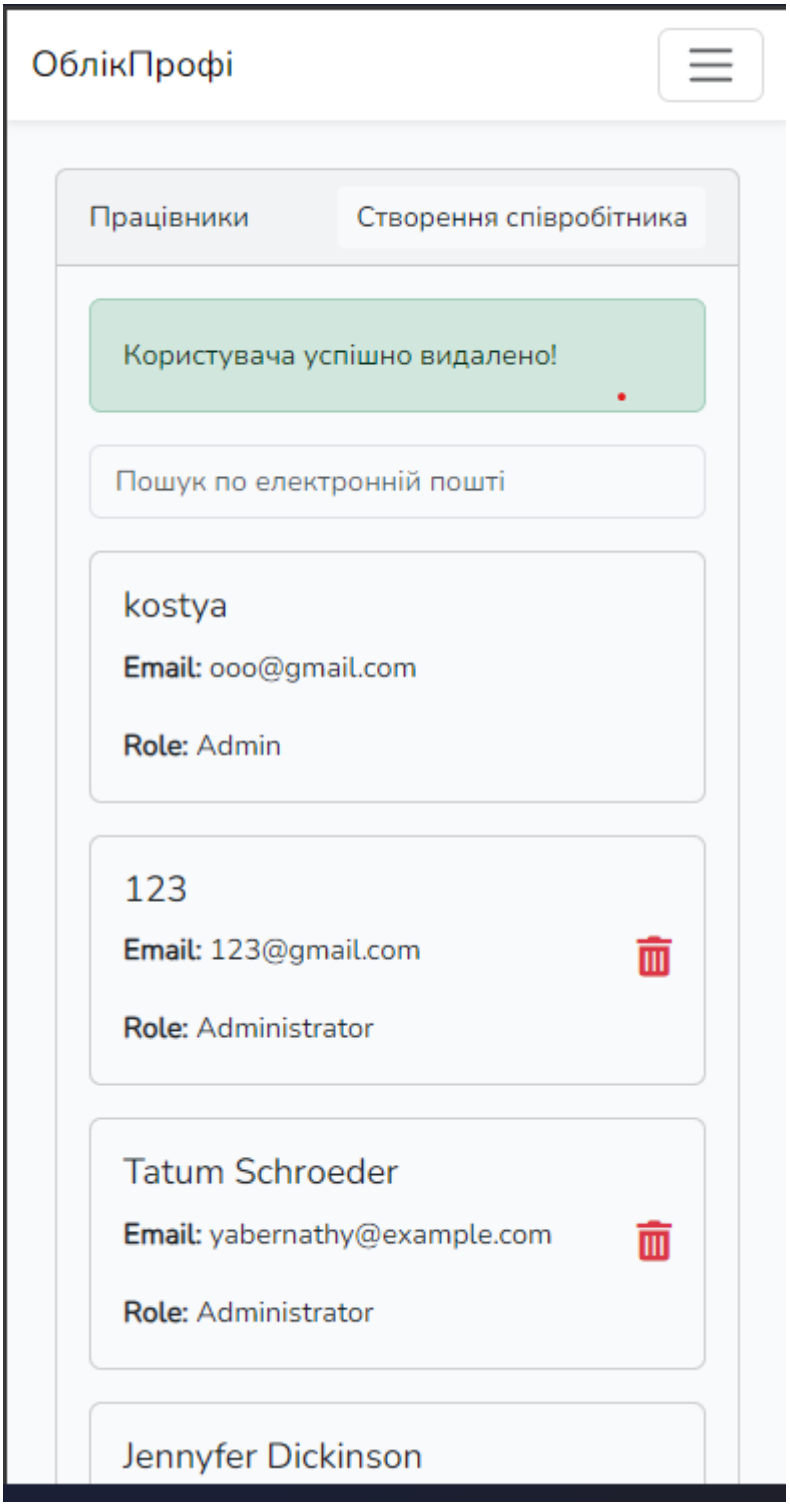

Рисунок 2.6 - Прототип дизайну для компоненту Співробітники

4. Розділ "Створення івенту": цей розділ містить форму для створення івенту для робітників, які будуть бачити його зі свого аккаунту і тільки для свого району, тоюто івент створюється з прив’язкою до району або районів, як його створить адмін бо ця функція є тільки в нього

5. Розділ "Додавання співробітника": цей розділ містить форму для додавання нового користувача у систему. По ролям, адміністратор не може видати роль вищу за нього

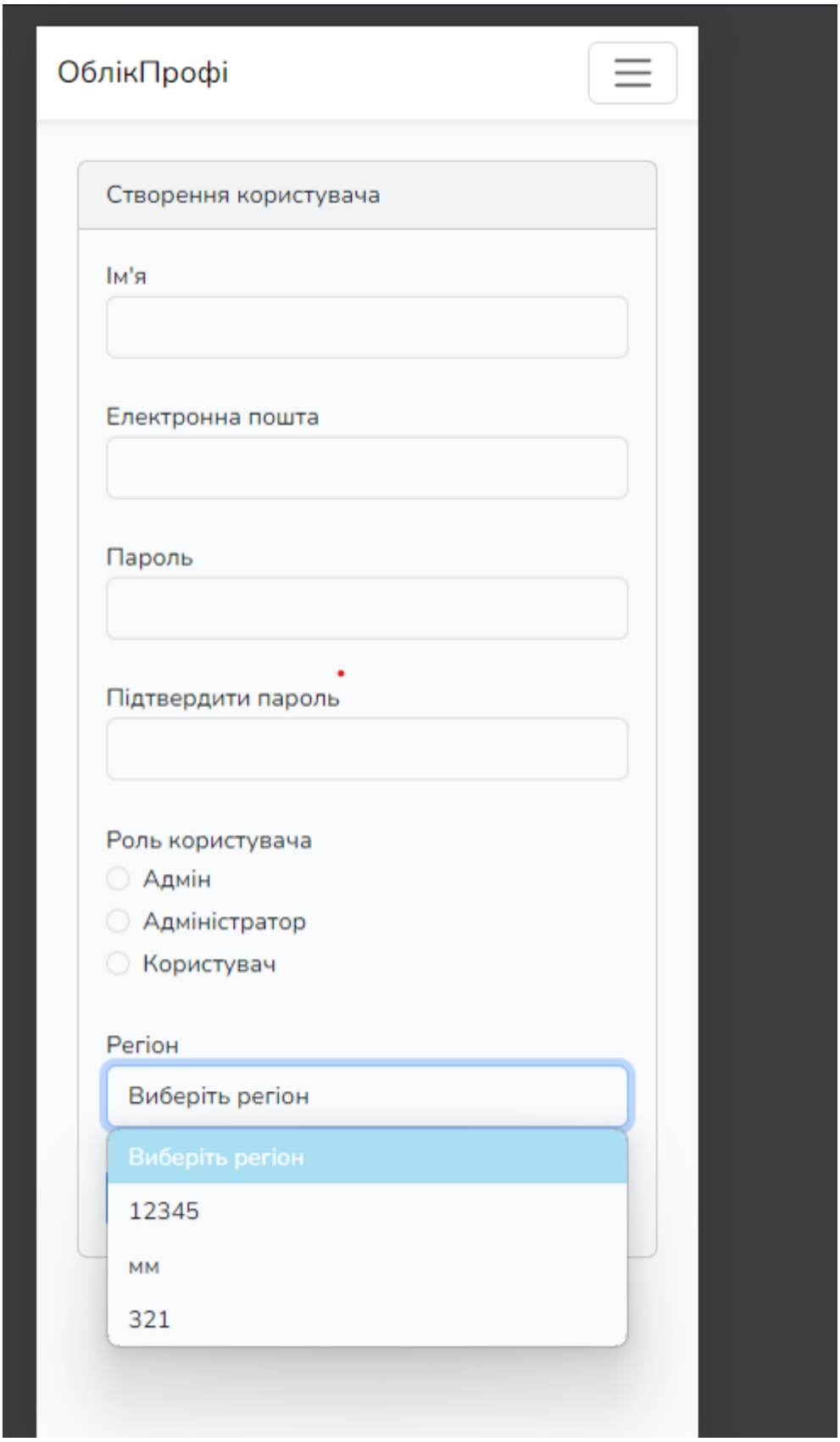

Рисунок 2.9 - Прототип дизайну розділу "Створення користувача"

2.7 Проектування моделі бази даних

База даних автоматизованої інформаційної системи повинна містити три рівні представлення: концептуальний, фізичний та логічний.

A. Концептуальне представлення бази даних – це високорівневе описання структури та залежностей між даними в предметній області обліку та управління особистими фінансами. Наприклад, можна визначити таблиці, які містять інформацію про операції фінансових надходжень, видатків та категорій. Кожна фінансова операція має свою категорію, у одної категорії може бути декілька операцій. Концептуальне представлення допомагає зрозуміти взаємозв'язки між різними об'єктами в системі та визначити ключові аспекти бази даних.
B. Фізичне представлення бази даних – це опис технічних аспектів зберігання даних, таких як формати файлів, типи дискових пристроїв, що використовуються для зберігання інформації системи обліку та управління особистими фінансами. Фізичне представлення бази даних включає в себе детальні відомості про розташування та організацію даних на дисках, а також процеси забезпечення безпеки і зменшення ризиків втрати даних.
C. Логічне представлення бази даних – це опис способів, якими дані можуть бути доступні та опрацьовуватися за допомогою запитів. Логічне представлення включає в себе опис схем бази даних, запитів, що дозволяють виконувати різні операції з даними, та інших елементів, що дозволяють взаємодіяти з інформацією системи обілку та управління особистими фінансами. Логічне представлення бази даних дозволяє здійснювати пошук, фільтр тощо.

Наступні розділи роботи наводять детальний опис кожного рівня представлення бази даних для автоматизованої інформаційної системи обліку та управління особистими фінансами.

2.7.1 Створення концептуальної моделі

Концептуальне представлення бази даних для інформаційної системи обліку та управління працівниками і заробітною платою почалось з аналізу

предметної області. Основне питання розбору – визначення об'єктів та їх взаємозв'язків у системі. Оскільки система передбачає сервісно-орієнтований підхід, то і об'єкти існують у контексті свого сервісу. Тому можна виділити наступні основні сервіси: сервіс облікових записів, сервіс авторизації, сервіс профілю користувача, сервіс фінансів.

Сервіс облікових записів містить сутність “Обліковий запис” (Account).

Сервіс авторизації містить сутність “Пароль авторизації” (Authentication Password).

Сервіс профілю користувача містить сутність “Профіль” (Profile).

Сервіс фінансів містить наступні сутності: “Категорія операції” (Operation Category), “Фінансова операція” (Finance Operation).

Концептуальна модель опису зв'язків

Отже, сутність “Обліковий запис” є репрезентацією одного користувача. Один користувач має зв'язок із одним об'єктом сутності “Пароль авторизації” та із одним об'єктом сутності “Профіль”. Користувач має зв'язок із множиною об'єктів сутності “Категорія операції”. Користувач має зв'язок із множиною об'єктів сутності “Фінансова операція”. Об'єкт сутності “Фінансова операція” має зв'язок із одним об'єктом сутності “Категорія операції”, в той час коли об'єкт сутності “Категорія операції” має зв'язок із множиною об'єктів сутності “Фінансова операція”.

Крім того, необхідно визначити ключові атрибути кожного об'єкту. Наприклад, для сутності користувача це стали такі атрибути, як унікальний ідентифікатор користувача, електронна пошта користувача, дата реєстрації, статус користувача тощо.

Рисунок 2.11 – концептуальна модель опису зв’язків розроблюваної бази даних для системиДалі, створена модель, яка описує зв'язки між різними об'єктами в системі.

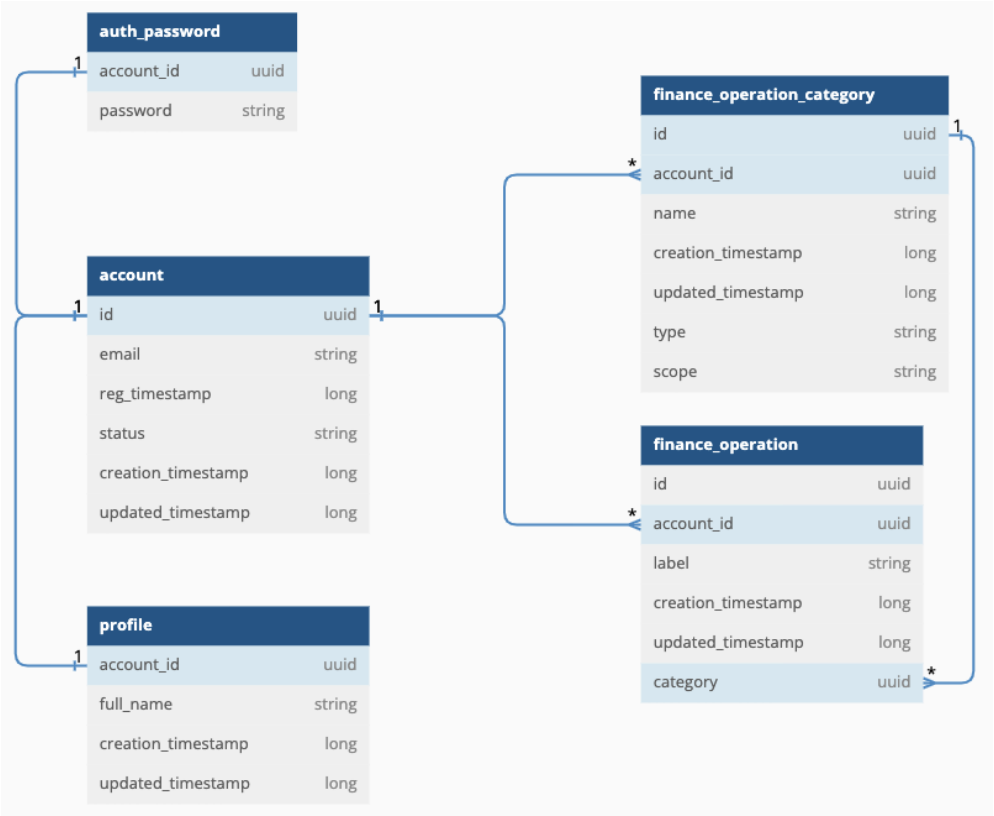

Рисунок 2.11 – концептуальна модель опису зв’язків розроблювальної бази даних для системи

Концептуальна модель атрибутів

Сутності і їх ключові атрибути:

Обліковий запис (Account):

Унікальний ідентифікатор

Електронна пошта

Дата реєстрації

Статус користувача

Пароль авторизації (Authentication Password):

Унікальний ідентифікатор

Хеш паролю

Дата останньої зміни паролю

Профіль (Profile):

Унікальний ідентифікатор

Ім'я

Прізвище

Контактна інформація

Категорія операції (Operation Category):

Унікальний ідентифікатор

Назва категорії

Опис категорії

Фінансова операція (Finance Operation):

Унікальний ідентифікатор

Сума

Дата операції

Категорія операції

Опис операції

2.7.2 **Створення логічної моделі**

Після створення концептуального представлення бази даних автоматизованої інформаційної системи обліку та управління особистими фінансами, наступним етапом стало створення логічного представлення. Логічне представлення полягає в перетворенні концептуальної моделі в модель, яка враховує особливості конкретної системи управління базою даних.

На етапі створення логічного представлення визначено сутності, атрибути та зв'язки між ними, відповідно до концептуальної моделі. При цьому враховані особливості використовуваної системи управління базою даних. Враховано, що в базі даних фінансові операції надходження і видатку є різними сутностями, а тому вони мають окремі таблиці. Таке ж саме правило застосовується і для категорій.

Також на етапі створення логічного представлення були визначені обмеження цілісності даних, такі як обов'язковість заповнення певних атрибутів, обмеження на діапазон значень атрибутів та заборона на видалення записів, що мають зв'язки з іншими записами в базі даних.

Рисунок 2.13 – описана логічна модель з урахуванням взаємозв’язків.

Таблиці та їх зв'язки

Основні сутності та їх атрибути, які визначено в логічній моделі:

Обліковий запис (Account):

id (PK)

email

registration_date

status

Пароль авторизації (Authentication Password):

id (PK)

account_id (FK)

password_hash

last_updated

Профіль (Profile):

id (PK)

account_id (FK)

first_name

last_name

contact_info

Категорія операції (Operation Category):

id (PK)

account_id (FK)

category_name

description

Фінансова операція (Finance Operation):

id (PK)

account_id (FK)

category_id (FK)

amount

operation_date

description

Обмеження цілісності даних

Для забезпечення цілісності даних були визначені наступні обмеження:

Обов'язковість заповнення певних атрибутів: Наприклад, атрибути email та password_hash повинні бути обов'язково заповнені для кожного облікового запису.

Обмеження на діапазон значень атрибутів: Наприклад, атрибут amount для фінансових операцій повинен бути більшим за нуль.

Заборона на видалення записів з зв'язками: Наприклад, обліковий запис не може бути видалений, якщо він має пов'язані фінансові операції або категорії операцій.

Завершуючи створення логічного представлення бази даних автоматизованої інформаційної системи обліку та управління особистими фінансами, були описані всі таблиці та їх зв'язки, а також визначені обмеження цілісності даних, застосовані в програмуванні самого проєкту. Це дозволило визначити структуру бази даних, яка відповідає концептуальній моделі та потребам системи управління базою даних і перейти до створення фізичного представлення.

Перехід до фізичного представлення

Після завершення створення логічної моделі, наступним кроком буде перехід до фізичного представлення бази даних. Це включає технічні аспекти зберігання даних, такі як формати файлів, типи дискових пристроїв, що використовуються для зберігання інформації системи, а також процеси забезпечення безпеки і зменшення ризиків втрати даних.

2.7.3 **Створення фізичної моделі**

Після створення логічного представлення бази даних, наступним етапом стало створення фізичного представлення. Фізичне представлення полягає в перетворенні логічної моделі в модель, яка відображає реальні характеристики фізичної реалізації бази даних та їх типів даних.

Структура та параметри фізичної реалізації

На етапі створення фізичного представлення була визначена структура та параметри фізичної реалізації бази даних. Це включало в себе вибір типу сервера баз даних, налаштування параметрів бази даних та визначення способу

зберігання даних. Було обрано реляційну базу даних MySQL для її надійності, масштабованості та широкої підтримки спільнотою розробників.

Вибір типу даних

Типи даних, що використовуються в таблицях бази даних, були обрані відповідно до вимог логічної моделі:

INT: для числових значень, таких як унікальні ідентифікатори.

VARCHAR: для текстових даних, таких як імена, електронні адреси.

DATE/TIMESTAMP: для зберігання дат і часу.

DECIMAL: для зберігання фінансових значень.

Параметри забезпечення безпеки даних

Також на етапі створення фізичного представлення були визначені параметри забезпечення безпеки даних:

Аутентифікація та авторизація:

Процеси ідентифікації користувачів та визначення їхніх прав доступу до даних реалізовані за допомогою логінів та паролів. Для додаткової безпеки можна використовувати двофакторну аутентифікацію та біометричні технології.

Шифрування даних:

Застосовується для забезпечення конфіденційності даних. Для цього використовуються алгоритми шифрування, такі як AES (Advanced Encryption Standard) для шифрування збережених даних і TLS (Transport Layer Security) для захисту даних під час передачі.

Захист від SQL-ін'єкцій:

Захист від вразливості, при якій зловмисник може використати SQL-запит для витягування або зміни даних в базі даних. Використовуються

параметризовані запити та підготовлені вислови, а також фільтрація введених користувачем даних.

Резервне копіювання:

Забезпечує можливість відновлення даних у разі їх втрати або пошкодження. Використовуються різні методи резервного копіювання, такі як повний бекап та інкрементальне копіювання. Підтримка регулярного резервного копіювання зберігає останні версії даних.

Моніторинг та аудит:

Процес відстеження доступу до даних та змін у базі даних для виявлення потенційних загроз безпеці. Використовуються системи моніторингу та аудиту, такі як Microsoft SQL Server Audit, Oracle Audit Vault тощо. Це дозволяє своєчасно виявляти та реагувати на аномальні дії або порушення політики безпеки.

Завершення створення фізичного представлення

Завершуючи створення фізичного представлення бази даних автоматизованої інформаційної системи обліку та управління особистими фінансами, було забезпечено її реалізацію згідно з встановленими параметрами та забезпечено правильну інтеграцію із системою управління базою даних. Це дозволило забезпечити ефективне та безпечне функціонування бази даних та системи в цілому.

2.7.4 **Загальний вигляд сутностей бази даних**

Після опису створення концептуального, логічного та фізичного представлення бази даних, сформовано загальний вигляд сутностей (таблиць) для початку розробки, програмування та подальшої розробки системи.

Таблиця "Обліковий запис" (account)

id (UUID, PRIMARY KEY) – унікальний ідентифікатор користувача

email (VARCHAR) – електронна пошта користувача

reg_timestamp (BIGINT) – дата реєстрації користувача

status (VARCHAR) – статус користувача

creation_timestamp (BIGINT) – дата створення об'єкту

updated_timestamp (BIGINT) – дата останнього оновлення користувача

Таблиця "Пароль авторизації" (auth_password)

account_id (UUID, PRIMARY KEY) – ідентифікатор користувача

password (VARCHAR) – зашифрований пароль користувача

Таблиця "Профіль" (profile)

account_id (UUID, PRIMARY KEY) – ідентифікатор користувача

full_name (VARCHAR) – повне ім'я користувача

creation_timestamp (BIGINT) – дата створення об'єкту

updated_timestamp (BIGINT) – дата останнього оновлення об'єкту

Таблиця "Операція надходження" (finance_add_operation)

id (UUID, PRIMARY KEY) – унікальний ідентифікатор операції

account_id (UUID) – ідентифікатор користувача

label (VARCHAR) – примітка операції

creation_timestamp (BIGINT) – дата створення об'єкту

updated_timestamp (BIGINT) – дата останнього оновлення об'єкту

category (UUID) – ідентифікатор категорії

Таблиця "Операція видатку" (finance_remove_operation)

id (UUID, PRIMARY KEY) – унікальний ідентифікатор операції

account_id (UUID) – ідентифікатор користувача

label (VARCHAR) – примітка операції

creation_timestamp (BIGINT) – дата створення об'єкту

updated_timestamp (BIGINT) – дата останнього оновлення об'єкту

category (UUID) – ідентифікатор категорії

Таблиця "Категорія операції надходження" (finance_add_operation_category)

id (UUID, PRIMARY KEY) – унікальний ідентифікатор категорії

account_id (UUID) – ідентифікатор користувача

name (VARCHAR) – назва категорії

creation_timestamp (BIGINT) – дата створення об'єкту

updated_timestamp (BIGINT) – дата останнього оновлення об'єкту

type (VARCHAR) – тип категорії

scope (VARCHAR) – область видимості категорії

Таблиця "Категорія операції видатку" (finance_remove_operation_category)

id (UUID, PRIMARY KEY) – унікальний ідентифікатор категорії

account_id (UUID) – ідентифікатор користувача

name (VARCHAR) – назва категорії

creation_timestamp (BIGINT) – дата створення об'єкту

updated_timestamp (BIGINT) – дата останнього оновлення об'єкту

type (VARCHAR) – тип категорії

scope (VARCHAR) – область видимості категорії

Розділ проєктування інформаційного забезпечення став важливим етапом у розробці автоматизованої інформаційної системи обліку та управління особистими фінансами. Розділ дозволив обрати мову програмування та технологію для безпосередньої реалізації інформаційної системи. Вибір мови програмування для реалізації системи залежав від вимог до продуктивності та функціональності, а також від наявності необхідних бібліотек та фреймворків.

Цей розділ дозволив визначити загальну архітектуру інформаційної системи, її необхідні компоненти, функціональності та взаємодії між ними. Декомпозиція компонентів системи є повною і завершеною. Для детальної

реалізації окремих компонентів рівень декомпозиції залежить від самого компоненту.

У даному розділі також було розглянуто проєктування бази даних, окремих сутностей та зв'язку між ними. Було розглянуто концептуальну, логічну та фізичну моделі даних. Виконаний аналіз щодо обґрунтованості вибору реляційної бази даних. Спроєктована база даних дозволить ефективно зберігати необхідні дані у контексті інформаційної системи.

Окрім того, у розділі проєктування було створено прототип дизайну для графічного інтерфейсу користувача. Прототип дизайну задає загальну структуру кожного розділу. Детальніший дизайн графічного інтерфейсу користувача буде виконуватись паралельно із розробкою цього інтерфейсу.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

1.1 Вибір програмного інструментарію

1.1.1 Бази даних

Дві популярні бази даних, які розглянуто для використання при розробці інформаційного забезпечення автоматизованої інформаційної системи обліку та управління особистими фінансами – PostgreSQL та MongoDB.

PostgreSQL:

PostgreSQL – це потужна та надійна об'єктно-реляційна система керування базами даних (СКБД). Вона використовує мову запитів SQL для зберігання та обробки даних. PostgreSQL надає широкі можливості для розробки різноманітних додатків, від веб-сайтів до складних корпоративних систем.

MongoDB:

MongoDB – це нереляційна база даних, яка використовує документи для зберігання даних. Вона підтримує широкий спектр операцій, включаючи збереження, оновлення, вилучення та пошук даних. MongoDB дозволяє зберігати дані у форматі JSON, що забезпечує легкий доступ до даних для розробників.

Таблиця 3.1 – порівняння баз даних для використання в практичній реалізації продукту

	PostgreSQL	MongoDB
Тип бази даних	Об’єктно-реляційна	Нереляційна
Мова запитів	SQL	MongoDB Query Language
Схема даних	Статична, може бути динамічною	Динамічна
Підтримка транзакцій	Повна підтримка транзакцій	Обмежена підтримка транзакцій
Підтримка реплікації	Так	Так

Продовження таблиці 3.1 - порівняння баз даних для використання в практичній реалізації продукту

Масштабованість	Горизонтальна та вертикальна	Горизонтальна
Доступність	Доступний на багатьох операційних системах	Доступний на багатьох операційних системах
Додаткові можливості	Повна підтримка ACID, гнучкість вибору типу індексів, розвинена система прав доступу	Вбудована підтримка геопросторових запитів, гнучкість у використанні складних документів

Для розробки інформаційного забезпечення та насамперед баз даних обрано PostgreSQL. Це пов'язано з тим, що PostgreSQL є реляційною базою даних, яка підтримує транзакції ACID, що є важливим аспектом для систем з великим обсягом даних та високою надійністю. Також PostgreSQL має багато інструментів для оптимізації запитів та широко використовується в індустрії.

Загалом, вибір бази даних залежить від конкретних потреб проекту. Враховуючи особливості задачі дипломного проекту, PostgreSQL є кращим вибором з точки зору надійності та організації даних.

1.1.2 **Програмне середовище**

Для розробки програмного забезпечення для автоматизованої інформаційної системи обліку та управління особистими фінансами, створеного на PHP з використанням фреймворку Laravel, можна використовувати різні програмні середовища, такі як NetBeans, IntelliJ IDEA та Eclipse. Оскільки NetBeans є застарілим продуктом, у цій роботі порівняємо IntelliJ IDEA та Eclipse.

IntelliJ IDEA:

IntelliJ IDEA, розроблена компанією JetBrains, відома своєю потужністю та розширеними можливостями для розробки на мовах Java, Kotlin, а також PHP за допомогою плагінів. Вона включає в себе інтелектуальні функції, такі як автоматичне доповнення коду, аналіз коду, рефакторинг та багато інших. IntelliJ IDEA також підтримує широкий спектр інструментів для розробки веб-додатків, мобільних додатків та інших типів програм.

Eclipse:

Eclipse – з іншого боку, є вільно розповсюджуваною платформою, розробленою фондом Eclipse. Вона відома своєю гнучкістю та розширюваністю завдяки широкому спектру плагінів, доступних для розширення її функціоналу. Eclipse також підтримує різні мови програмування та типи проектів, включаючи Java, C/C++, PHP, Python та інші.

Таблиця 3.2 – порівняння програмних середовищ для використання в практичній реалізації продукту

	IntelliJ IDEA	Eclipse
Вартість	Платне, є безкоштовна версія для навчання	Безкоштовне
Підтримка Git	Є	Є
Нативна підтримка Web-розробки	Є	Є, за використанням плагінів
Кількість плагінів	Значна кількість плагінів від самого розробника та спільноти	Значна кількість плагінів від різних розробників
Інструменти для тестування та налагодження	Багато інтегрованих засобів	Багато інтегрованих засобів

Оскільки для реалізації інформаційного забезпечення системи був обраний фреймворк Laravel, а середовище розробки IntelliJ IDEA має інструменти для роботи із Laravel “із коробки”, то вибір стає саме на IntelliJ IDEA. Це середовище розробки має велику спільноту і є передовим навідміну від Eclipse. IntelliJ IDEA відома своєю високою продуктивністю та широким спектром функцій, які полегшують розробку програмного забезпечення. Вона має потужні інтегровані інструменти для аналізу, автоматизації та підтримки розробки.

IntelliJ IDEA надає широку підтримку для різних мов програмування, включаючи Java, Kotlin, JavaScript, TypeScript, PHP, Python та інші, що робить її привабливою для розробників з різних сфер. Команда розробників за IntelliJ IDEA активно веде розробку, випускаючи нові версії та вдосконалення. Також JetBrains надає хорошу підтримку користувачам.

Зважаючи на ці переваги, можна зробити висновок, що IntelliJ IDEA від JetBrains є оптимальним вибором програмного середовища для розробки інформаційного забезпечення для автоматизованої інформаційної системи обліку та управління особистими фінансами.

1.2 **Файлова структура проєкту**

Файлова структура проєкту для розробки інформаційного забезпечення та баз даних системи обліку та управління працівниками і заробітною платою включає наступні компоненти:

Коренева папка проєкту:

Це основна папка проєкту, яка містить всі інші підпапки та файли. Вона може мати назву, що відповідає назві проєкту або його ідентифікатору.

api:

Відповідно до архітектури системи, ця папка містить код для шару представлення, який є найближчим до контролера у шаблоні MVC. Шар представлення перетворює об'єкти бізнес-логіки у об'єкти, які можуть бути використані для подання інформації на графічний інтерфейс користувача. Крім того, тут може бути реалізована логіка для глобального чи локального кешування.

app:

Ця папка містить код для запуску інформаційної системи та програмний код, що відноситься до стадії запуску системи, включаючи файли конфігурації системи та налаштування на рівні програмного коду.

biz:

Відповідно до архітектури системи, ця папка містить код для шару бізнес-логіки. Це останній шар у наведеній архітектурі. Тут реалізовані основні сутності системи - моделі та методи взаємодії з ними. Кожна сутність має свій сервіс, і взаємодія з сутностями здійснюється через відповідний сервіс.

diagrams:

Ця папка містить всю документацію, пов'язану з проєктом, включаючи специфікації вимог, технічні описи, діаграми архітектури, звіти про тестування та іншу документацію, необхідну для розуміння та розвитку проєкту.

ui:

Відповідно до архітектури системи, ця папка містить код для шару представлення, а саме шар контролерів. Тут реалізований код для контролерів у

шаблоні MVC, а також графічний інтерфейс користувача у вигляді HTML-сторінок з CSS-стилями та кодом на JavaScript.

pom.xml:

Цей файл є основним файлом конфігурації проєкту в середовищі розробки на основі інструменту управління проєктами Maven для Java-програм. Його основна функція – описувати конфігурацію проєкту, включаючи залежності, плагіни, налаштування збірки тощо. Це дозволяє Maven автоматизувати процеси збірки, тестування та розгортання проєкту.

Додаткові папки:

Залежно від потреб проєкту, можуть бути додаткові папки для специфічних компонентів або модулів. Наприклад, папки для інтеграції з іншими системами, документації API, додаткових статичних сторінок веб-сайту тощо.

Додаткові файли:

Службові файли, які використовуються для інтеграції з іншими інструментами, такими як Git, або файли, що стосуються конкретного середовища розробки.

Ця файлова структура дозволяє організовувати проєкт у логічні групи файлів і директорій, забезпечуючи зручність розробки, управління та збереження даних. Важливо, щоб загальна структура відповідала архітектурі інформаційної системи. Структура для детальної реалізації системи може варіюватися залежно від конкретних вимог та потреб проєкту, і її можна адаптувати відповідно до власних вимог і стандартів розробки.

4 БІЗНЕС-ПЛАН

4.1 Опис продукту

4.1.1 Вступ

Автоматизована інформаційна система для обліку та управління особистими фінансами представлена у вигляді веб-додатку. Вона забезпечує користувачам зручний інструмент для контролю особистих витрат, прибутків та змін у бюджеті. Ці інструменти допомагають планувати бюджет, виявляти зайві витрати, що в результаті покращує економічне становище користувача.

Облік особистих фінансів стає все більш актуальним у сучасному світі, де фінансова грамотність є ключовим фактором успішного управління власними ресурсами. Завдяки автоматизованій інформаційній системі користувачі зможуть краще розуміти свої фінансові потоки, приймати обґрунтовані рішення щодо витрат та заощаджень, а також планувати свої фінансові цілі на майбутнє.

Монетизація системи можлива через продаж доступу до неї або надання преміальних функцій за додаткову плату, що робить її привабливою як для кінцевих користувачів, так і для бізнесу, який надає такі послуги.

4.1.2 Основні функції та переваги

Система пропонує широкий спектр функцій, що дозволяють користувачам ефективно керувати своїми фінансами:

Ведення обліку доходів та витрат:

Можливість введення даних про доходи та витрати у різних категоріях, таких як харчування, розваги, транспорт, комунальні послуги тощо.

Автоматичне підрахування залишку коштів, що дозволяє користувачам бачити загальний фінансовий стан у режимі реального часу.

Підтримка мультивалютних операцій для користувачів, які працюють з різними валютами.

Аналіз фінансового стану:

Відображення даних у вигляді графіків та діаграм, що допомагає візуально оцінити стан фінансів та динаміку змін.

Генерація звітів за певні періоди, які можуть бути збережені або експортувані у різні формати, такі як PDF, Excel тощо.

Автоматичні рекомендації на основі аналізу витрат та доходів, що допомагають користувачам оптимізувати свої витрати.

Бюджетування:

Можливість створення та управління бюджетами для різних категорій витрат та періодів.

Сповіщення про перевищення встановленого бюджету, що допомагає уникати непередбачених витрат.

Функція планування великих витрат з урахуванням поточного фінансового стану.

Класифікація операцій:

Користувач може створювати власні категорії для доходів та витрат, що дозволяє персоналізувати систему під свої потреби.

Можливість фільтрування операцій за категоріями, датами, сумами та іншими параметрами для швидкого доступу до необхідної інформації.

Пошук по транзакціях, що дозволяє швидко знаходити потрібні записи.

Безпека та конфіденційність:

Використання сучасних методів шифрування для захисту даних користувачів.

Двофакторна автентифікація для забезпечення безпеки доступу до системи.

Регулярне оновлення системи безпеки та моніторинг можливих загроз.

4.1.3 Цільова аудиторія

Цільова аудиторія включає широкий спектр користувачів, кожен з яких має свої специфічні потреби у використанні системи:

Індивідуальні користувачі:

Люди, які бажають контролювати свої особисті фінанси, планувати бюджет та аналізувати витрати.

Студенти, які хочуть навчитися керувати своїми фінансами з раннього віку.

Сімейні користувачі:

Родини, що хочуть вести спільний облік доходів і витрат, та планувати сімейний бюджет.

Молоді пари, які планують спільне фінансове майбутнє та бажають уникнути фінансових труднощів.

Малий бізнес:

Маленькі підприємства та фрілансери, які потребують простого та зручного інструменту для управління фінансами.

Сімейні бізнеси, які хочуть мати контроль над фінансами та планувати розвиток.

Фінансові консультанти:

Професіонали, які надають послуги фінансового консалтингу та потребують інструментів для аналізу та планування фінансів клієнтів.

Консультанти, які бажають надавати клієнтам більш детальну та точну інформацію про їхні фінансові операції.

4.2 Аналіз ринку

4.2.1 Огляд ринку

Ринок програмного забезпечення для управління особистими фінансами є швидко зростаючим. Зростаюча фінансова грамотність та необхідність контролю

витрат сприяють зростанню попиту на такі інструменти. Веб-додатки для обліку фінансів стають дедалі популярнішими завдяки своїй доступності, функціональності та зручності використання.

Основними гравцями на ринку є великі компанії, які пропонують комплексні рішення для управління фінансами, а також невеликі стартапи, що спеціалізуються на окремих аспектах фінансового обліку. Конкуренція на ринку досить висока, що стимулює постійне вдосконалення продуктів та впровадження нових функцій.

4.2.2 Оцінка попиту

Попит на інструменти для управління особистими фінансами постійно зростає. Цьому сприяє збільшення фінансової грамотності серед населення, необхідність контролю витрат у складних економічних умовах, а також бажання користувачів планувати своє фінансове майбутнє.

Очікується, що ринок програм для обліку та управління фінансами продовжить зростати протягом наступних кількох років. Особливий попит прогнозується серед молоді, яка активно користується цифровими технологіями, а також серед малого бізнесу та фрілансерів, які потребують зручних та доступних інстру ментів для фінансового обліку.

4.3 Маркетингова стратегія

4.3.1 Ціноутворення

Для монетизації продукту пропонуються наступні моделі ціноутворення:

Безкоштовна версія:

Обмежений функціонал, підтримуваний рекламою.

Дозволяє користувачам ознайомитися з основними можливостями системи без фінансових витрат.

Преміум-підписка:

Повний доступ до всіх функцій без реклами.

Місячна або річна оплата.

Додаткові функції, такі як розширені аналітичні інструменти, інтеграція з банківськими рахунками, персоналізовані рекомендації.

Разовий платіж:

Одноразова покупка без подальших платежів, що надає довічний доступ до преміум-функцій.

Привабливий варіант для користувачів, які не бажають підписуватися на регулярні платежі.

4.3.2 Канали розповсюдження

Основними каналами розповсюдження будуть:

Онлайн-маркетинг:

Використання соціальних мереж (Facebook, Instagram, LinkedIn) для просування продукту та залучення нових користувачів.

SEO-оптимізація для підвищення видимості в пошукових системах.

Контекстна реклама в Google Ads, яка дозволяє залучати користувачів, зацікавлених у фінансових інструментах.

Email-маркетинг для підтримання зв'язку з існуючими користувачами, інформування про нові функції, акції та інші важливі новини.

Контент-маркетинг: створення та публікація корисного контенту на тему фінансового планування, обліку витрат та доходів, що сприятиме залученню нових користувачів та покращенню репутації продукту.

Партнерства:

Співпраця з фінансовими консультантами, банками та іншими фінансовими установами для просування продукту серед їх клієнтів.

Включення продукту до пакетів фінансових послуг, що пропонуються банками та іншими фінансовими установами.

Мобільні платформи:

Розповсюдження через App Store та Google Play, що дозволить залучити користувачів мобільних пристроїв.

Використання платформи Progressive Web Apps (PWA) для забезпечення кросплатформеного доступу до продукту без необхідності завантаження з магазинів додатків.

Реклама на спеціалізованих платформах:

Реклама на фінансових та бізнесових форумах, веб-сайтах та блогах, що мають аудиторію, зацікавлену в управлінні фінансами.

Участь у фінансових виставках та конференціях для просування продукту серед професіоналів та зацікавлених користувачів.

4.4 Фінансовий план

4.4.1 Прогноз доходів

Прогноз доходів базується на таких джерелах:

Підписка на преміум-версію:

Передбачається, що більшість доходів буде отримана від підписок на преміум-версію. Очікується, що значна частина користувачів перейде на преміум-підписку після ознайомлення з можливостями безкоштовної версії.

Рекламні доходи:

Доходи від реклами в безкоштовній версії. Ці доходи можуть бути збільшені шляхом залучення рекламодавців, зацікавлених у фінансово грамотній аудиторії.

Дохід від одноразових платежів:

Дохід від продажу довічного доступу до преміум-функцій. Цей варіант буде привабливий для користувачів, які не бажають платити за підписку щомісяця або щорічно.

Прогноз доходів включає поступове зростання протягом перших трьох років після запуску продукту, з найбільшим приростом у перші два роки завдяки активним маркетинговим зусиллям та зростаючій базі користувачів.

4.4.2 Початкові витрати

Основні початкові витрати включають:

Розробка програмного забезпечення:

Витрати на розробку веб-додатку та мобільних додатків, включаючи заробітну плату розробників, витрати на інструменти розробки та тестування.

Витрати на впровадження функціоналу безпеки, інтеграцію з банківськими системами та іншими сервісами.

Маркетинг:

Витрати на рекламні кампанії в соціальних мережах, контекстну рекламу, SEO-оптимізацію, контент-маркетинг та інші маркетингові заходи.

Витрати на створення маркетингових матеріалів, таких як відео, банери, статті тощо.

Інфраструктура:

Витрати на сервери, хостинг та підтримку, включаючи витрати на забезпечення високої доступності та безпеки даних.

Витрати на резервне копіювання та відновлення даних, а також на моніторинг та аудит системи.

Заробітна плата:

Витрати на оплату праці команди розробників, маркетологів, дизайнерів та технічної підтримки.

Витрати на навчання та професійний розвиток співробітників для підтримки високого рівня кваліфікації команди.

4.5 План розвитку

План розвитку передбачає постійне вдосконалення продукту та розширення функціоналу. Основні напрямки розвитку включають:

Впровадження нових функцій:

Додавання нових функцій на основі відгуків користувачів, таких як більш детальна аналітика, додаткові інструменти планування, інтеграція з іншими фінансовими сервісами.

Впровадження автоматичних нагадувань та сповіщень про важливі фінансові події та строки.

Розширення можливостей аналітики та автоматизації:

Розробка більш детальних та інтуїтивно зрозумілих аналітичних інструментів для кращого розуміння фінансових потоків.

Впровадження інструментів для автоматизації повторюваних завдань, таких як автоматичне категоризування витрат, планування платежів тощо.

Інтеграція з новими фінансовими інструментами та сервісами:

Інтеграція з новими банками та фінансовими установами для забезпечення більшого охоплення користувачів.

Впровадження підтримки криптовалют та інших нових фінансових інструментів.

Постійне оновлення безпеки та захисту даних:

Регулярні оновлення системи безпеки для захисту даних користувачів від нових загроз.

Впровадження додаткових заходів безпеки, таких як біометрична автентифікація, моніторинг підозрілої активності тощо.

Розширення на нові ринки:

Вихід на нові регіональні ринки, включаючи локалізацію продукту для різних мов та культурних особливостей.

Співпраця з місцевими фінансовими установами та партнерами для збільшення охоплення.

Зворотний зв'язок та підтримка користувачів:

Постійне покращення служби підтримки користувачів для швидкого вирішення проблем та запитів.

Використання зворотного зв'язку від користувачів для вдосконалення продукту та впровадження нових функцій.

Цей бізнес-план дозволяє побачити перспективи розвитку продукту та забезпечити його успішне впровадження на ринок. Виконання запланованих заходів сприятиме зростанню бази користувачів, збільшенню доходів та зміцненню позицій продукту на ринку.

Висновок

Розробка автоматизованої інформаційної системи для обліку та управління працівниками і заробітною платою є важливим кроком у напрямку покращення ефективності управління персоналом у комунальних підприємствах. Даний проєкт включає детальний аналіз потреб організацій, вибір оптимальних технологій, розробку архітектури системи, створення бази даних та планування бізнес-стратегії для успішного впровадження продукту на ринок.

Ключові аспекти роботи:

Аналіз потреб та функціональні вимоги:

Визначено основні функції, необхідні для ефективного обліку та управління працівниками та їхньою заробітною платою, такі як ведення обліку працівників, розрахунок заробітної плати, аналіз робочого часу та управління кадрами.

Вибір технологій та інструментів:

Обрано мови програмування, фреймворки та системи управління базами даних, які найкраще відповідають вимогам проєкту. Використання PHP у поєднанні з Laravel забезпечує гнучкість та надійність розробки серверної частини, а PostgreSQL надає можливість ефективного зберігання та обробки даних.

Розробка архітектури системи:

Визначено архітектурний підхід на основі шаблону MVC, що забезпечує чітке розділення логіки, представлення та взаємодії з даними. Це дозволяє підвищити зручність розробки, тестування та підтримки коду.

Проектування та розробка бази даних:

Створено концептуальну, логічну та фізичну моделі бази даних, що забезпечують структуроване та ефективне зберігання даних про працівників, їхні посади, робочий час та заробітну плату. Застосовано сучасні методи шифрування для забезпечення конфіденційності та безпеки даних.

Бізнес-план та маркетингова стратегія:

Розроблено бізнес-план, який включає опис продукту, аналіз ринку, маркетингову стратегію та фінансовий план. Визначено основні канали розповсюдження, моделі монетизації та прогноз доходів, що дозволяє забезпечити комерційний успіх продукту.

План розвитку:

Визначено основні напрямки розвитку продукту, такі як впровадження нових функцій, розширення можливостей аналітики та автоматизації, інтеграція з новими фінансовими інструментами та сервісами, постійне оновлення безпеки та розширення на нові ринки.

Висновки та перспективи

Впровадження автоматизованої інформаційної системи для обліку та управління працівниками і заробітною платою сприятиме підвищенню ефективності управління персоналом у комунальних підприємствах, зменшенню витрат на адміністрування та покращенню контролю за виплатами заробітної плати. Завдяки інтеграції з сучасними технологіями та використанню передових методів розробки, система забезпечить високу надійність, безпеку та зручність використання.

Реалізація даного проєкту відкриває нові можливості для автоматизації бізнес-процесів у сфері управління персоналом, що дозволить організаціям оптимізувати свою діяльність та зосередитися на стратегічних задачах. Це, у свою чергу, сприятиме покращенню якості надання послуг та підвищенню рівня задоволеності працівників.

В цілому, автоматизована інформаційна система обліку та управління працівниками і заробітною платою є перспективним інструментом, який відповідає сучасним вимогам ринку та має великий потенціал для подальшого розвитку та вдосконалення.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. The complementary use of IDEF and UML modelling approaches / C.-H. Kim et al. Computers in Industry. 2003. Vol. 50, no. 1. P. 35–56.
2. Effective Java. 3rd ed. Addison-Wesley Professional, 2017. 416 p.
3. Abhishek, Soumili Chandra, Soumili Chandra. Analysis and Comparison of the Spring Framework, Struts Framework, Vaadin Framework, and Play Framework Performance, Used to Create Web Applications in Java. International Journal of Advanced Research in Science, Communication and Technology. 2023. P. 277–282.
4. Thakur R. N., Pandey U. S. The Role of Model-View Controller in Object Oriented Software Development. Nepal Journal of Multidisciplinary Research. 2019. Vol. 2, no. 2. P. 1–6.
5. DEVELOPMENT OF WEB APPLICATION USING MODEL VIEW CONTROLLER (MVC) ARCHITECTURE. International Journal of Progressive Research in Engineering Management and Science. 2023.
6. Dylan D. Schmorrow, Cali M. Fidopiastis (2009). "Foundations of Augmented Cognition: Directing the Future of Adaptive Systems".
7. Steven C. McConnell (2004). "Code Complete: A Practical Handbook of Software Construction".
8. Robert C. Martin (2008). "Clean Code: A Handbook of Agile Software Craftsmanship".
9. Robert M. Kaplan, Dennis P. Saccuzzo (2018). Psychological Testing: Principles, Applications, and Issues.