МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему: **«Реалізація фронт та бекенду веб-орієнтованого середовища в інформаційній системі «Управління ІТ проектами»**

Виконав ст. гр. Іт-62
Спеціальності 126 – „Інформаційні системи та технології ”_____
(шифр і назва)
Долінський Андрій Григорович____
(Прізвище та ініціали)

Керівник: к.ф-м.н., в.о., доц. Чухрай Л.В.
(Прізвище та ініціали)

Рецензент.
(Прізвище та ініціали)

ДУБЛЯНИ-2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

другий (магістерський) рівень вищої освіти
за спеціальністю – 126 – "Інформаційні системи та технології"

"ЗАТВЕРДЖУЮ"
Завідувач кафедри_____
д.т.н., проф. А.М. Тригуба
"____"_____2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Долінський Андрій Григорович

1. Тема роботи: «Реалізація фронт та бекенду веб-орієнтованого середовища в інформаційній системі « Управління ІТ проектами»

Керівник роботи Чухрай Любомир Володимирович, к.ф.-м.н., в.о., доцента
Затверджені наказом по університету 12 вересня 2024 року № 616/к-с.
2. Строк подання студентом роботи 06.12.2024 р.

3. Початкові дані до роботи: Створення веб-додатку для управління ІТ-проєктами з акцентом на інтеграцію фронтенд- і бекенд-частини, Аналіз існуючих систем управління проєктами, Структура веб-сайту центру туристичної інформації.

4. Зміст розрахунково-пояснювальної записки:
1. Аналіз методології управління ІТ-проєктами
2. Встановлення завдань і методологія дослідження
3. Розробка аналітичної підсистеми для веб-орієнтованого середовища
4. Реалізація аналітичної підсистеми в контексті веб-середовища
5. Охорона праці та безпека в надзвичайних ситуаціях
6. Висновки та пропозиції
7. Список використаних джерел.

5. Перелік презентаційного матеріалу (з зазначенням обов'язкових елементів):_____
_____
_____
_____
_____
_____
_____
_____
_____

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 4	*Чухрай Л.В., в.о. доцента кафедри інформаційних технологій*		
5	*Городецький І.М., доцент кафедри фізики, інженерної механіки та безпеки виробництва*		

7. Дата видачі завдання 28.04.2023.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання роботи	Приміт-ка
1	*Написання першого розділу та означення головних завдань роботи*	12.09 - 01.10.24	
2	*Виконання другого розділу та формування головних показників для розрахунків*	12.09 - 01.10.24	
3.	*Виконання третього розділу та формування початкових даних*	01.10 - 01.11.24	
5.	*Написання розділу: «Охорона праці»*	01.10 - 01.11.24	
7.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів графічної частини*	01.11 - 01.12.24	
8.	*Завершення роботи в цілому*	01.11 - 01.12.24	

Студент _____ Долінський А.Г.
(підпис)

Керівник роботи _____ Чухрай Л.В.
(підпис)

УДК 004.41:004.45:004.6:005.8

«Реалізація фронт та бекенду веб-орієнтованого середовища в інформаційній системі «Управління ІТ проектами». Долінський А.Г. Кваліфікаційна робота. Кафедра ІТ. – Дубляни, Львівський НУП, 2024.

66 с. текст. част., 38 рис., 9 табл., 39 літ. джерел.

У кваліфікаційній роботі була реалізована веб-орієнтована інформаційна система «Управління ІТ-проектами», що включає в себе як розробку фронтенд, так і бекенд частин. Для цього був проведений детальний аналіз вимог до системи та спроектована її архітектура, що дозволяє забезпечити зручність і ефективність управління проектами на всіх етапах — від планування до моніторингу виконання завдань.

Фронтенд частина системи була реалізована з використанням сучасних веб-технологій, що забезпечують зручний інтерфейс для користувачів. Інтерфейс дозволяє керувати проектами, відстежувати прогрес виконання завдань, призначати обов'язки учасникам команд, переглядати статистику та звіти.

В бекенд частині системи реалізована логіка обробки запитів, взаємодія з базою даних і забезпечення функціонування основних процесів: створення, редагування та видалення проектів і завдань, призначення відповідальних осіб, а також моніторинг стану проектів у реальному часі. Для розробки бекенду були використані потужні фреймворки та технології, що дозволяють забезпечити високу продуктивність і масштабованість системи.

В результаті реалізації системи були досягнуті основні цілі — створення зручного інструменту для управління ІТ-проектами, який дозволяє забезпечити високий рівень ефективності і прозорості на всіх етапах реалізації проектів, а також забезпечити надійність і безпеку обробки даних.

ЗМІСТ

ВСТУП

У сучасних умовах динамічного розвитку інформаційних технологій автоматизація управління IT-проектами є однією з ключових задач для забезпечення конкурентоспроможності компаній. Ефективне управління проектами вимагає інтегрованих рішень, які дозволяють планувати, контролювати та аналізувати виконання завдань у режимі реального часу. Веб-орієнтовані системи є найбільш актуальними рішеннями завдяки їхній доступності, масштабованості та можливостям інтеграції з іншими інструментами.

Метою роботи є розробка фронтенду та бекенду веб-орієнтованого середовища в інформаційній системі «Управління IT-проектами», яке забезпечуватиме ефективну автоматизацію процесів планування, моніторингу і звітності в рамках IT-проектів.

Для досягнення мети були поставлені наступні **завдання**:

1. Провести аналіз сучасних технологій та підходів до розробки веб-орієнтованих інформаційних систем.
2. Розробити архітектуру системи, включаючи структуру бази даних і взаємодію фронтенд та бекенд частин.
3. Реалізувати бекенд із використанням сучасних серверних технологій для забезпечення обробки даних та API.
4. Розробити фронтенд із зручним та інтуїтивно зрозумілим інтерфейсом користувача.
5. Провести тестування системи для перевірки її працездатності, функціональності та безпеки.
6. Проаналізувати результати розробки та сформулювати рекомендації щодо подальшого розвитку системи.

Об'єктом є процеси управління IT-проектами, що включають планування, організацію, виконання та контроль за проектами в межах інформаційної системи. Зокрема, це включає розробку та реалізацію веб-орієнтованих інформаційних систем для управління проектами, які використовують як фронтенд, так і бекенд

технології для забезпечення функціональності, безпеки та зручності користування.

Предметом є методи та підходи до розробки і реалізації компонентів веб-орієнтованих інформаційних систем для управління IT-проектами. Це включає проєктування та впровадження архітектури системи, розробку інтерфейсу користувача (фронтенд) і серверної частини (бекенд), а також проектування та оптимізацію бази даних для зберігання інформації про проекти, завдання, користувачів і їхні взаємодії.

РОЗДІЛ 1.
АНАЛІЗ МЕТОДОЛОГІЇ УПРАВЛІННЯ ІТ ПРОЄКТАМИ

1.1 Аналіз наслідків та впливу проблеми

Розробка та впровадження веб-орієнтованих середовищ для інформаційних систем управління ІТ-проектами є складним завданням, яке впливає на ефективність проектного менеджменту та досягнення організаційних цілей. Недосконаліть існуючих систем, їх обмежена функціональність або складність інтеграції можуть створювати значні проблеми для команд, що працюють над ІТ-проектами. Основні аспекти впливу цієї проблеми можна охарактеризувати наступним чином:

Відсутність зручного та функціонального веб-інтерфейсу для моніторингу проектів призводить до труднощів у доступі до актуальної інформації про стан виконання завдань.

Відсутність прозорості у виконанні задач спричиняє неузгодженість дій між членами команди та зниження швидкості прийняття рішень.

Неякісна інтеграція фронт- та бекенд-частин інформаційної системи ускладнює комунікацію між учасниками проекту.

Недостатня інтерактивність або складність у використанні інструментів системи може спричиняти втрату часу на пошук потрібної інформації або невірне розуміння задач.

Відсутність автоматизованих інструментів у бекенді для обробки даних, створення звітів або оцінки прогресу проекту.

Зростає обсяг рутинної роботи, яку мають виконувати члени команди вручну, що знижує їхню продуктивність.

Використання застарілих або неефективних технологій у фронт- чи бекенді.

Підвищується ймовірність виникнення помилок під час розробки або використання системи, що негативно впливає на кінцеву якість проектів.

Відсутність модулів аналітики та системи моніторингу у веб-орієнтованій системі.

Учасники проекту не можуть отримати необхідні дані для аналізу ефективності роботи або виявлення критичних точок у проектах.

Проблеми, пов'язані з реалізацією фронт- та бекенд-частин веб-орієнтованих систем для управління IT-проектами, мають значний вплив на продуктивність команд, якість виконання проектів і досягнення цілей. Їх вирішення через модернізацію підходів до розробки таких систем є ключовим завданням, яке дозволить зменшити негативні наслідки та підвищити ефективність процесів управління проектами.

1.2. Огляд подібних рішень і виявлення існуючих проблем

Системи управління проектами чудово працюють, якщо втрачається контроль над проектами у компанії або хочеться мати всю інформацію про проекти в одному місці. Не всі рішення будуть ідеально відповідати потребам, тому існує список найкращих програм для управління проектами.

FlexiProject **–** це комплексне та вдосконалене програмне забезпечення для управління проектами та портфелями. Він надає проектним командам інструменти для планування та ефективного виконання проектів будь-якого масштабу. “Серцем” програми є просунутий, але інтуїтивно зрозумілий графік з діаграмою Ганта та дошками Канбан. Для кожного проекту, окрім календарного плану, можна розробити комплексний фінансовий бюджет, реєстр ризиків, набір продуктів, ресурсів та план проекту, яким можна керувати. Функція автоматичних оглядів проектів – чудовий інструмент для автоматизації процесу звітування за проектами перед керівництвом компанії. FlexiProject є найкращим у стратегічному управлінні проектами, пропонуючи можливість пов’язувати проекти зі стратегічними цілями компанії та керувати ресурсами і портфелями проектів.

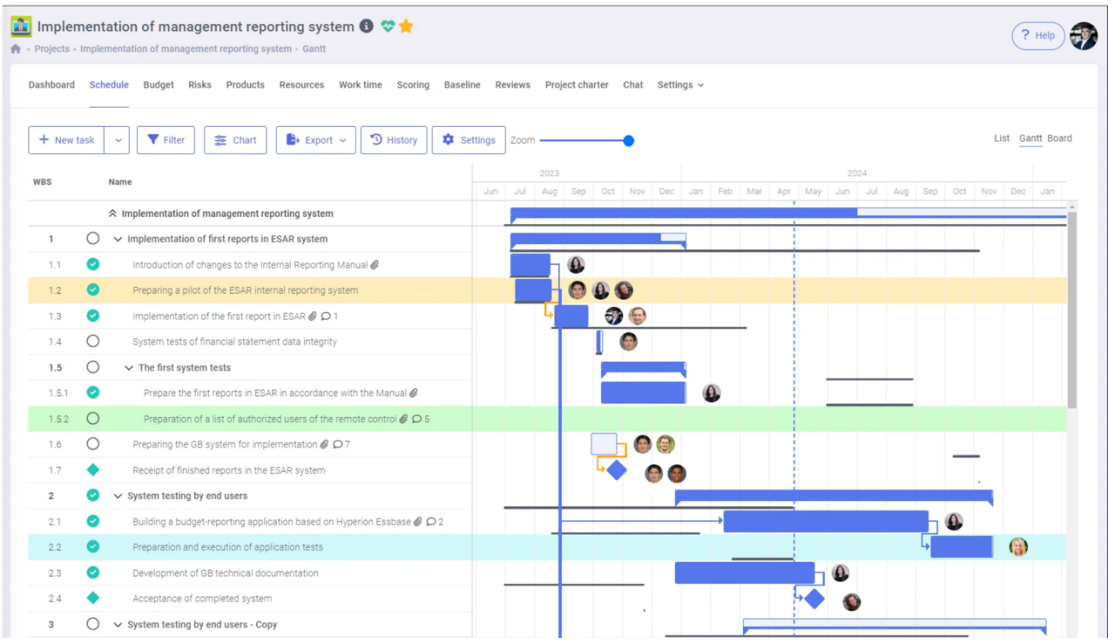

Рисунок 1.1. Інтерфейс додатку FlexiProject для управління проектами та портфелями

Wrike **–** це платформа для управління проектами з багатьма функціями. Він дозволяє створювати та налаштовувати командні робочі простори та необмежену кількість папок, проектів, завдань та підзадач. Динамічні форми запитів автоматизують процес введення роботи. Платформа дозволяє структурувати роботу відповідно до потреб команди та розподіляти завдання на основі продуктивності та можливостей. Він підтримує міжкомандну співпрацю, управління проектами, просування стратегічних ініціатив та досягнення цілей. Wrike надає миттєву інформацію та інформаційні панелі в режимі реального часу, щоб полегшити прийняття рішень на основі даних. Він інтегрується з різними інструментами та платформами, такими як Email, Zoom, Slack та MS Teams, що дозволяє автоматизувати робочий процес. Повнофункціональні мобільні додатки дозволяють отримати доступ до платформи Wrike з будь-якого місця, онлайн і офлайн.

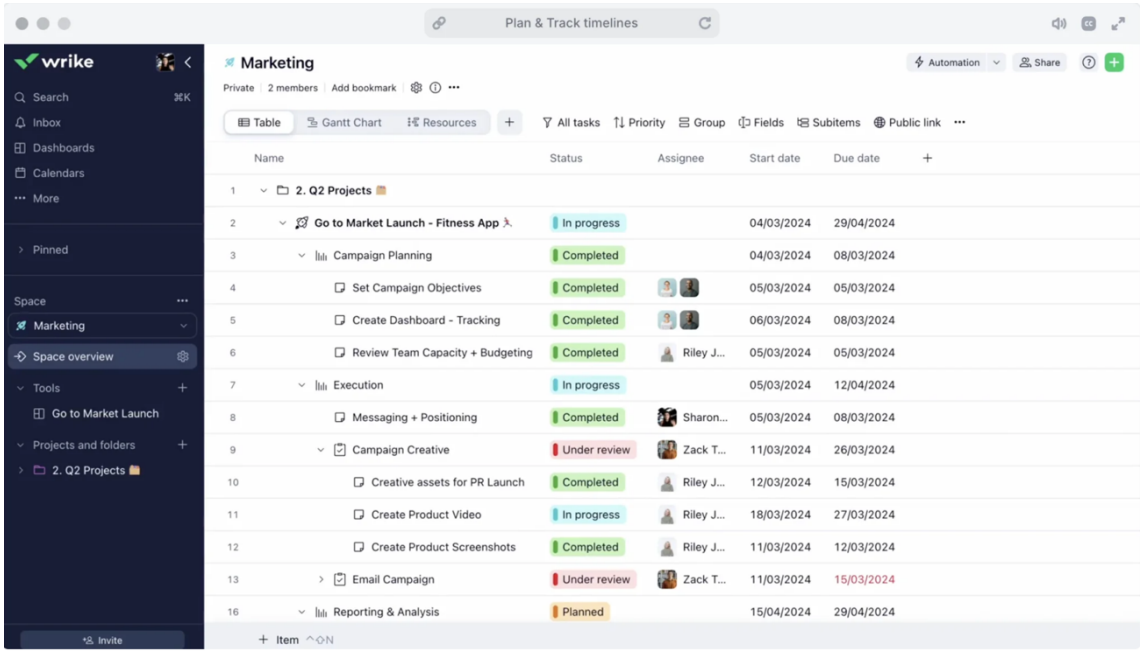

Рисунок 1.2. Інтерфейс додатку Wrike.

Redmine- інструмент дозволяє запускати кілька проектів одночасно і створювати завдання. Він чудово підходить для відстеження прогресу, а також для додавання та ведення документації по проекту. Додатковою перевагою є діаграма Ганта, яка доступна в системі. Redmine – це програма з відкритим вихідним кодом, яка дозволяє практично будь-яку кастомізацію та вільне використання. Це популярний інструмент у командах розробників, який дозволяє, з одного боку, керувати роботою її членів, а з іншого – дає змогу іншим працівникам повідомляти про проблеми або нові функції.

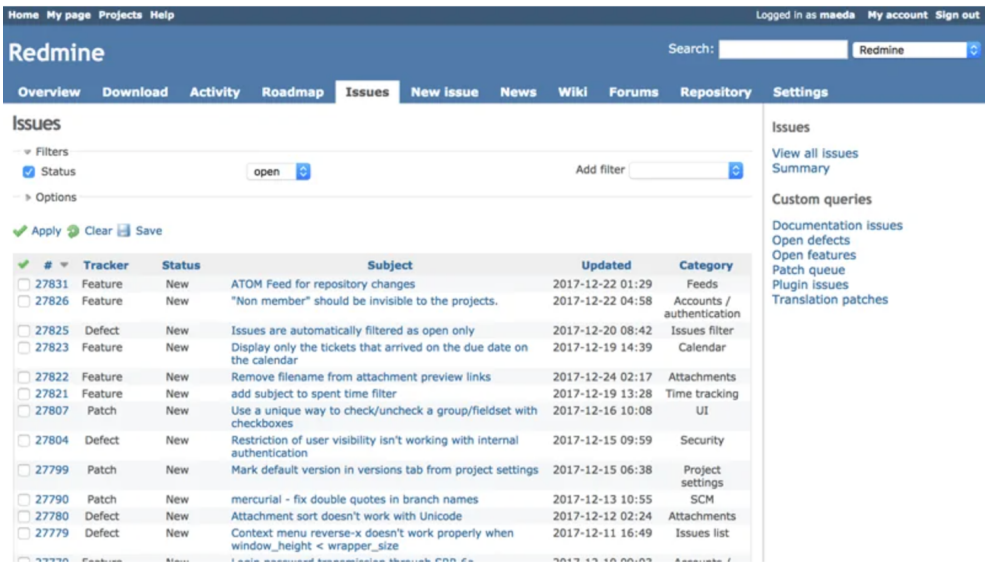

Рисунок 1.2. Інтерфейс додатку Redmine

У сфері управління проектами Redmine є дуже гнучким інструментом, який дозволяє користувачам відстежувати хід проекту та обговорювати різні його етапи.

Microsoft Project дозволяє створювати проєкти з будь-яким рівнем вкладеності завдань. Система також дозволяє створювати діаграму Ганта і додавати віхи. Інструмент дозволить вам ефективно реалізовувати різні проекти, від невеликих проектів до великих ініціатив. Безсумнівно, це одне з найстаріших і найдовше розроблених програм для управління проектами. Незважаючи на численні зміни та конкуренцію, цей додаток досі супроводжує багато команд у реалізації проектів. Суттєвою перевагою цієї системи є інтеграція додатків Microsoft 365 та функціоналу One Cloud. Значною перевагою залишається інтерфейс, який знайомий більшості користувачів MS Excel, але дозволяє набагато більше, ніж відома (і не завжди улюблена) програма електронних таблиць. Ще однією перевагою MS Project є його дуже велика база знань і висока зворотна сумісність. Microsoft заявляє про зворотну сумісність аж до Project 2007, що може бути необхідним для деяких проектних офісів.

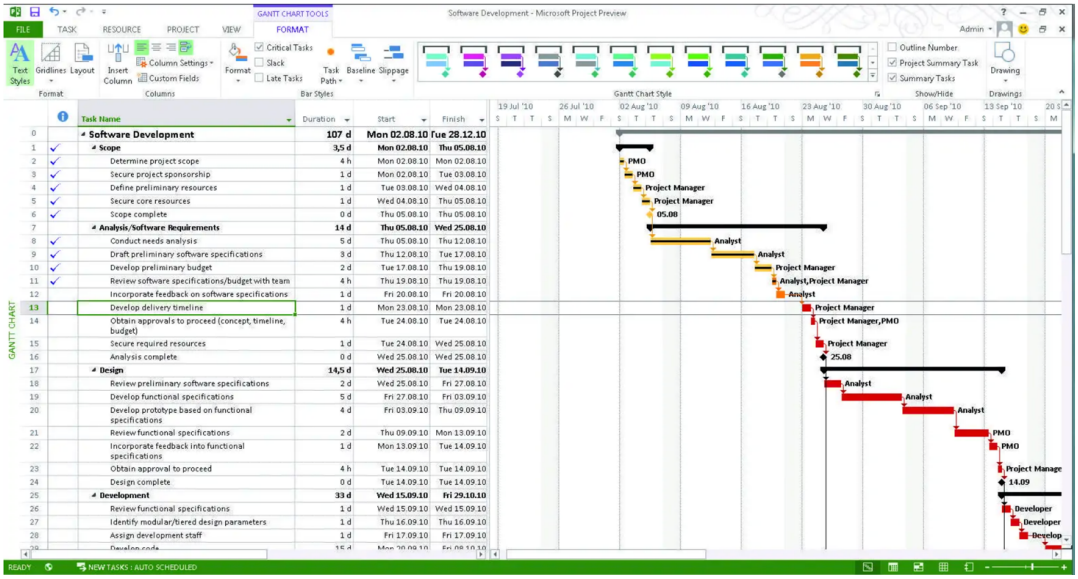

Рисунок 1.3. Інтерфейс додатку Microsoft Project

Існує багато програм для управління організаційними проектами. Вибір

правильного інструменту залежить від потреб компанії, кількості та складності проектів. Перед тим, як зробити вибір, варто протестувати вказане програмне забезпечення, щоб зрозуміти, чи відповідає обраний додаток усім очікуванням ОУП, менеджера проекту або керівництва компанії.

Clarizen - повнофункціональне програмне забезпечення для управління проектами з додатковими функціями, що містить додаткові інструменти для управління портфелем, ресурсами та робочим потоком. Clarizen [1] - це програмне рішення для автоматизованих професійних служб корпоративного рівня, розроблене для прискорення вашого бізнесу - інтеграція роботи, контенту та процесів, що забезпечують ефективнішу роботу (рис. 1.1). Справжня увага Clarizen [1] - це робити проекти швидше, заощаджуючи робочі процеси. Clarizen [1] - чудовий інструмент управління проектами, якщо у вас є багато повторюваних проектів, які потребують повторюваних процесів, оскільки автоматизація робочого процесу досить гнучка і потужна.

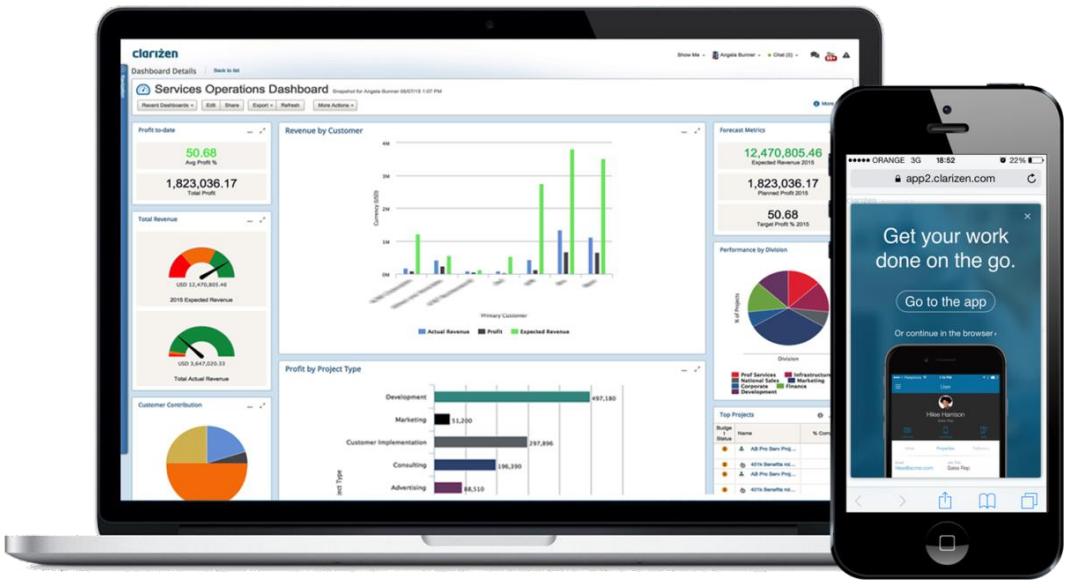

Рисунок 1.5 - Clarizen

Програмне забезпечення для управління проектами та ресурсами 10,000ft [4] дозволяє організаціям легко бачити та діяти на найважливіших даних для проектівта людей. Зосередившись на широкій картині, 10,000ft [4] надає бізнесу точні дані в потрібний час для прийняття впевнених рішень щодо своїх проектів, команд та портфелів.

Рисунок 1.6 - 10,000ft

Для управління проектами з великою кількістю вбудованих функцій спритного управління проектами існує програма **Ravetree** (рисунок. 1.6).

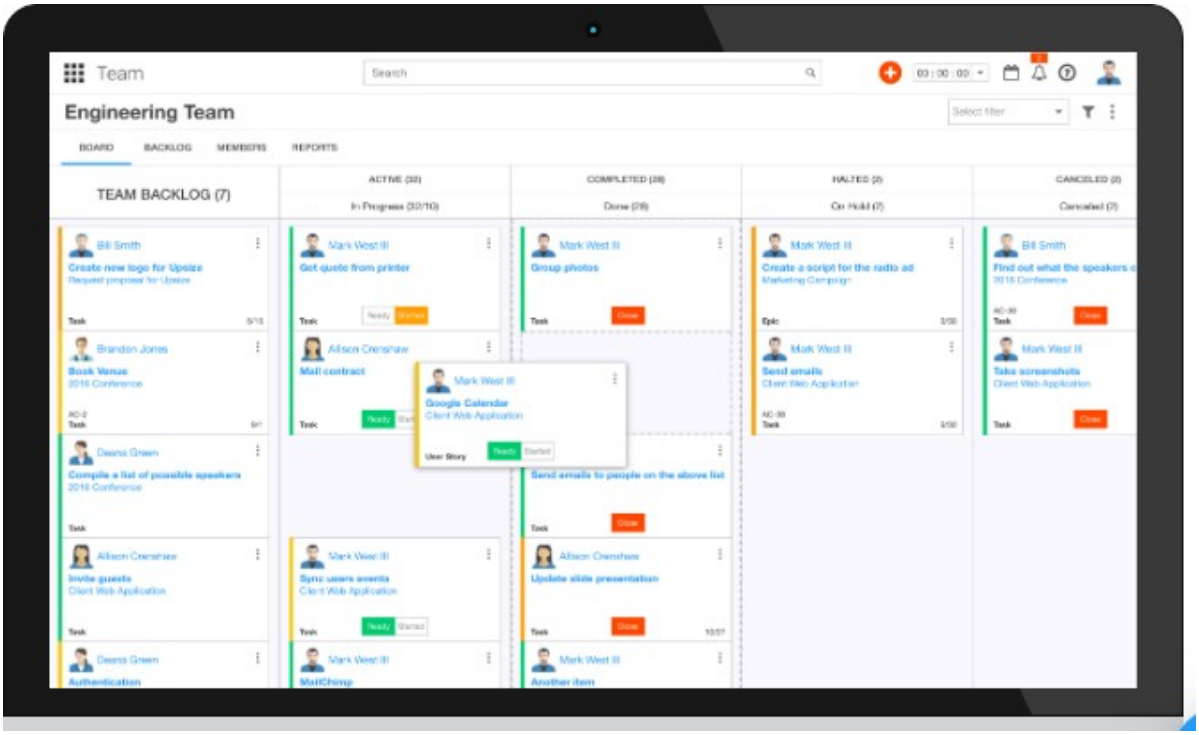

Рисунок 1.6 – Ravetree

Ravetree [5] - це програмна платформа для управління роботою, яка дає можливість командам швидше доставляти роботу, бути більш обізнаними та витрачати менше часу на пошук інформації. Проектні організації у всьому світі

використовують Ravetree для управління своїми проектами, ресурсами та інформацією про клієнтів - все в одному місці

1.3. Планування ризиків проекту

Планування ризиків є невід'ємною частиною управління проектом, що забезпечує своєчасну ідентифікацію, оцінку та мінімізацію впливу ризиків на виконання проекту. Цей процес дозволяє створити чіткий план дій для реагування на можливі загрози, зменшуючи ймовірність негативних наслідків і забезпечуючи досягнення цілей проекту.

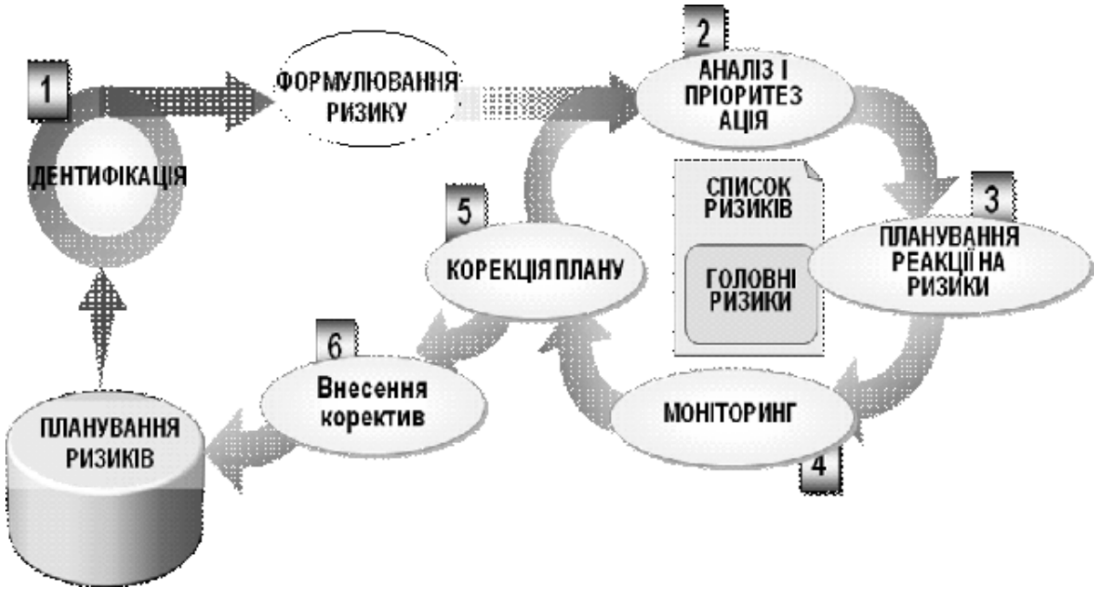

Рисунок 1.5. Управління ризиками на протязі життєвого циклу проекту

Цей рисунок ілюструє процес планування ризиків у проекті, представлений у вигляді циклічного підходу з ключовими етапами.

1. Перший етап включає виявлення потенційних ризиків. Це процес, спрямований на формування переліку всіх можливих загроз для проекту, які можуть вплинути на його виконання.

2. На етапі формулювання ризиків кожен ризик чітко описується, включаючи його причини, можливі наслідки та області впливу.

3. Ризики оцінюються за рівнем імовірності їх виникнення та потенційного впливу на проект. Цей етап допомагає визначити головні ризики, які потребують першочергової уваги.

4. Для кожного значущого ризику розробляються відповідні стратегії управління. Це можуть бути заходи для уникнення ризику, зменшення його впливу, прийняття або передання іншій стороні.

5. Постійне спостереження за ризиками в процесі виконання проекту. Це дозволяє вчасно виявляти зміни в їх статусі та реагувати на нові загрози.

6. На основі моніторингу ризиків і змін у проекті вносяться корективи до плану управління ризиками, що забезпечує його актуальність і ефективність.

Циклічний характер процесу показує, що управління ризиками є безперервною діяльністю. Вона включає постійне повернення до етапу ідентифікації та аналізу ризиків для адаптації до змінних умов проекту. Це забезпечує проактивний підхід до управління загрозами, що дозволяє мінімізувати їхній вплив на успіх проекту.

Управління проектами є комплексним процесом, який вимагає чіткої структуризації етапів, контролю виконання завдань та своєчасного внесення коректив. Ефективна реалізація проекту залежить від здатності команди адаптуватися до змін і забезпечувати відповідність результатів початковим вимогам.

Рисунок. 1.6. Блок-схема процесу управління ризиками за фазами життєвого циклу

На даному рисунку показано послідовність виконання етапів проекту із циклічним підходом до перевірки й виправлення помилок. Він структурований як блок-схема, яка відображає логіку роботи над проектом.

1. Початковий етап, де проводяться організаційні заходи, такі як збір вимог, планування ресурсів і визначення ключових завдань проекту.

2. На кожному етапі проводиться перевірка на відповідність очікуванням та виявлення помилок чи невідповідностей. У разі потреби повертаються до попереднього етапу для внесення змін.

3. Цей етап включає уточнення деталей, налаштування технічних і організаційних процесів для ефективного виконання.

4. Основний етап, де відбувається реалізація запланованих дій і завдань. Використовуються ресурси, і команда працює над досягненням цілей проекту.

5. На завершальних етапах проекту проводиться остаточна перевірка, коригування та забезпечення відповідності кінцевого результату початковим вимогам.

6. Фінальний етап, де проект формально завершується, здійснюється підготовка підсумкової документації, звітів і передача результатів стейкхолдерам.

Ця схема ілюструє ітераційний підхід, де перевірка та виправлення є постійним процесом на кожному етапі. Такий підхід забезпечує гнучкість і адаптивність проекту до змінних умов і вимог, що підвищує його успішність і якість кінцевого результату.

РОЗДІЛ 2.
ВСТАНОВЛЕННЯ ЗАВДАНЬ І МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ

2.1. Планування змісту структури робіт ІТ-проекту

WBS відображає взаємозв'язок кожного завдання з іншими завданнями, цілим і кінцевим. Він показує розподіл відповідальності, а також визначає необхідні ресурси та доступний час на кожному етапі моніторингу та управління проектами.

Мета WBS - зробити великий проект більш керованим. Розбиття на дрібні частини означає, що робота може виконуватися одночасно різними членами команди, що призведе до кращої продуктивності команди та спрощення управління проектами в цілому.

Продуктом у даному проекті є аналітична підсистема веб-орінтованого середовища з вивчення дисципліни «Управління ІТ проектами». Основні дії проекту: формування ТЗ, аналіз предметної області інструментів для створерення аналітичної підсистеми, розробка анатіличної підсистеми, тестування та доставка. Після цього кожну з вищевказаних дій потрібно декомпозувати до елементарних робіт. Наймасштабнішою гілкою у проекті є розробка. Даний пункт розбиваємо на проектування архітектури проекту за допомогою CQRS [19], проектування БД, розроблення Backend та розроблення Frontend.

Вигляд WBS структури представлено на рисунку 2.1.

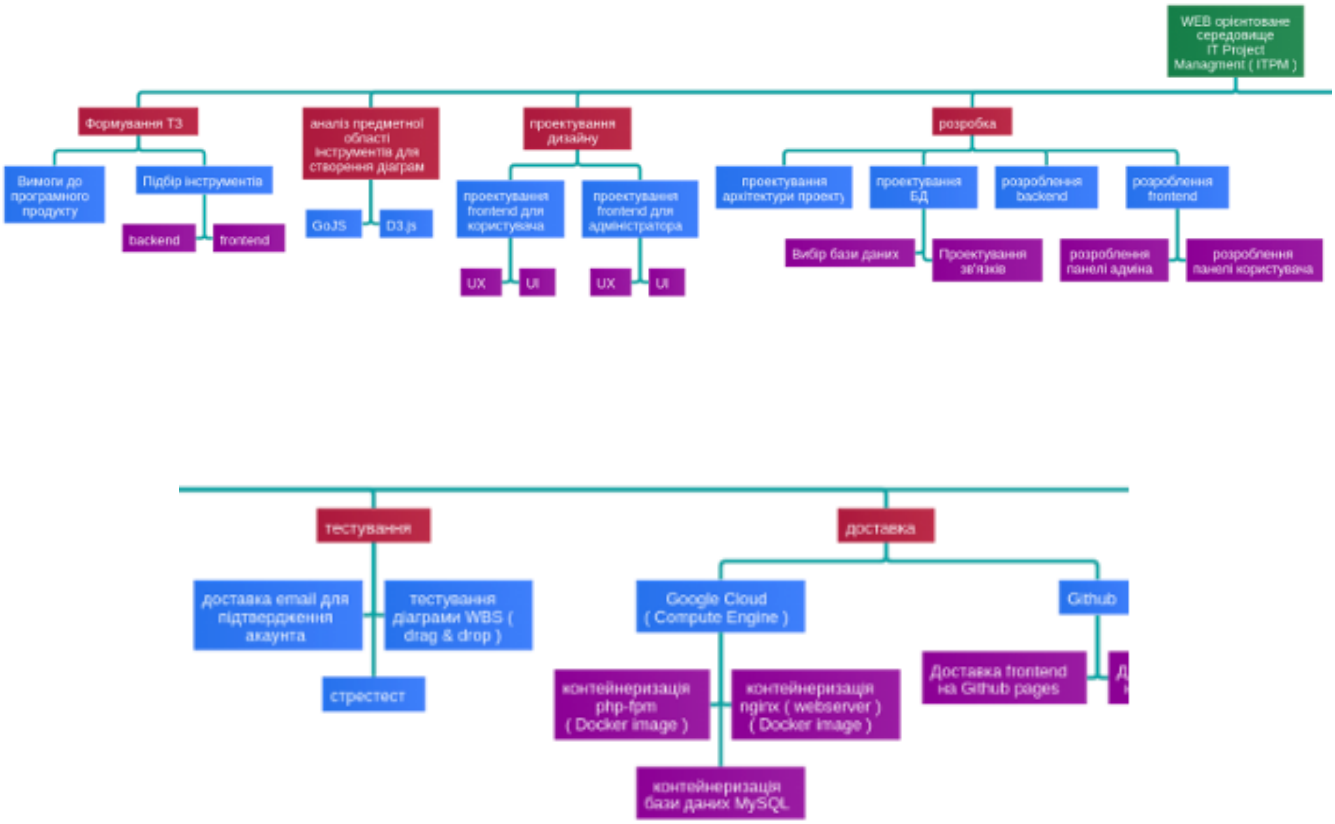

Рисунок 2.1 – WBS структура проекту

Для кожного етапу проекту необхідно чітко визначити елементарні роботи, що дозволить ефективно реалізувати кожну частину проекту. Використання WBS дозволяє зберігати контроль над виконанням завдань на кожному етапі і забезпечує своєчасне завершення проекту.

Застосування WBS сприяє кращому управлінню великими проектами, деталізує процеси та покращує організацію роботи, що є важливим для досягнення успішного результату в проекті.

Застосування WBS у проекті надає також інші переваги, такі як:

Покращення комунікації в команді. Кожен учасник чітко розуміє свої завдання і місце у загальному контексті проекту.

Більш точне планування часу. Завдяки деталізації завдань можна точніше оцінити час на виконання кожної частини проекту.

Зменшення ймовірності помилок. Розбиття складних завдань на прості дозволяє знизити ймовірність неусвідомлених помилок або пропусків на етапі виконання.

Таким чином, використання WBS в управлінні проектами є критично важливим для забезпечення успіху проекту, особливо в контексті складних і великих IT-проектів, таких як розробка аналітичної підсистеми. Завдяки цьому інструменту можна не лише спростити організацію роботи, але й значно підвищити її ефективність, забезпечуючи систематичний підхід до управління проектом.

2.2 Структура та завдання frontend розробки

Фронтенд це клієнтська сторона додатка чи сайта. Це – саме те, що ви бачите у себе на екрані під час взаємодії з певним цифровим рішенням, тобто, всі кнопки, плеєри, поля, посилання, меню і тому подібне.

З цього витікає, що фронтенд технологія спрямована на створення інтерфейсу, який ви бачите, і функціонала, який розгортається на стороні клієнта. У цьому полягає головна відмінність фронтенду від дизайну. Адже важливо не лише відтворити зовнішній вигляд кожної кнопки, рядка, картинки та іконки, але й зробити так, щоб усі вони виконували певні функції: при натисканні на іконку меню – відкривалося меню, при натисканні на посилання – виконувався перехід за цим посиланням, при наведенні на картинку – відображався альтернативний текст і так далі.

Розробка front end – це процес створення компонентів, що взаємодіють з користувачами, коли вони відкривають ваше цифрове рішення. Нижче ми пропонуємо розглянути типові складові фронтенду, притаманні майже кожному такому рішенню. Але, в залежності від специфіки проєкту, їх список може буди дещо довшим.

Інтерфейс — це низка засобів та правил, за якими компоненти цифрового рішення обмінюються інформацією. Обмін може відбуватися зі стороннім програмним забезпеченням, периферійними пристроями, прошивкою пристрою, а також самим користувачем.

Кнопки – це елемент інтерфейсу, який використовується для виконання цільової дії користувачем. Це може бути підтвердження введення даних, завершення сеансу, перехід на іншу сторінку, додавання товару в кошик, оформлення покупки та багато іншого.

Вікна та поля для введення даних

Вікна та поля для введення даних призначені для введення певної інформації користувачем за допомогою пристрою-носія програмного забезпечення або периферійного обладнання. Надалі, ця інформація використовується цим програмним забезпеченням для виконання цільових дій – будь то реєстрація користувача профілю, внесення даних платіжної картки, додавання відгуку про товар або будь-що інше, що має бути надане самим користувачем.

Взаємодія з користувачем (UX)

UX (user experience) включає навігацію програмного рішення, його функціональний набір, а також все те, що відноситься до верстки – текст, зображення, розташування компонентів. Разом з цим, UX має ключову відмінність від UI, яка полягає в тому, що перший спрямований на забезпечення зручності та інтуітивності рішення, тоді як другий – виключно на створення привабливого візуалу. Зазвичай, ці компоненти нерозривні, і практично будь-який проект вимагає роботи що над одним, що над іншим.

Анімовані елементи

Анімовані елементи також є частим компонентом клієнтської сторони програмних рішень і використовуються як для прикрашання інтерфейсів та надання їм динаміки, так і з практичною метою, наприклад, для підштовхування до цільових дій, привернення уваги користувачів та ін.

Етапи фронтенд розробки

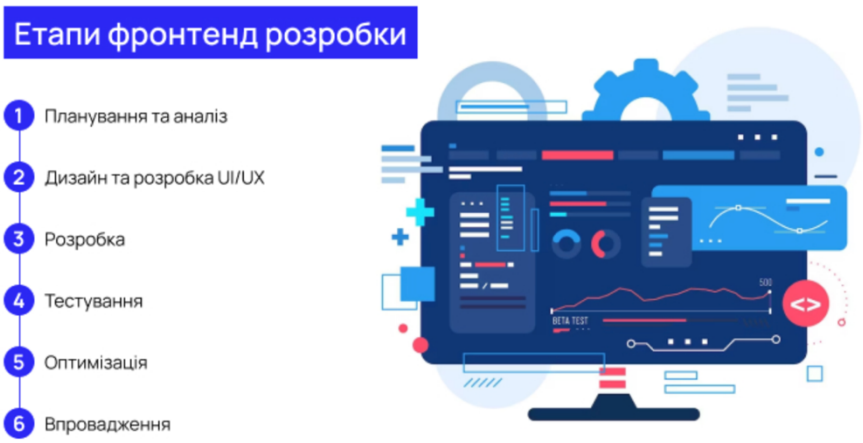

Рисунок 2.1. Етапи фронтенд-розробки

На етапі планування та аналізу вибирається технологічний стек, визначається майбутня архітектура проекту, та розробляється документація, якої потрібно буде дотримуватися всім членам команди. Після опису архітектури проекту ми приступаємо до її реалізації. На цьому етапі ми створюємо і дизайн: спочатку – скетчі, а потім, з поступовим узгодженням деталей із замовником, – hi-fi прототипи та фінальний інтерфейс.

Після того, як дизайн готовий, роботу розпочинають команди фронтенд і бекенд розробників. Найчастіше вони працюють синхронно, реалізуючи клієнтську та серверну частину рішення відповідно.

Після завершення процедури написання програмного коду, наші QA фахівці приступають до тестування проекту, покриваючи весь код тест-кейсами. При виявленні в ньому багів, проект повертається на опрацювання відповідним фахівцям для їх усунення, після чого проводиться повторне тестування. Ця процедура повторюється доти, доки проект не буде повністю відповідати своїй документації.

Для фронтенд розробки існує багато технологій. Розглянемо їх більш детально нижче.

HTML (Hypertext Markup Language) – це мова програмування, яка використовується для структурування та відображення веб-сторінки та її контенту. За допомогою HTML виконується розмітка елементів на сторінці, що

надає розробникам можливість створити всі елементи, а також визначити їх розташування та розміщення відносно один одного.

Cascading Style Sheets (каскадні таблиці стилів) – це формальна мова декорування та опису зовнішнього вигляду веб-рішень. Зокрема, за допомогою CSS відбувається стилізація зовнішнього вигляду сторінки та її елементів. Таким чином, розробники оформлюють всі компоненти сторінок, а також задають їх колір, розмір і інші особливості, тобто, все те, що перетворює схематичне зображення функціональних елементів в повноцінне рішення, відповідне до дизайн-макету.

JavaScript (JS) – це динамічна мова програмування, яка застосовується до рішень на базі HTML, забезпечуючи їх інтерактивність. Простими словами, за допомогою JS відбувається пожвавлення сторінки та створення реакції на дії користувачів. Саме завдяки JS, ви бачите повідомлення, які з'являються при наведенні на той чи інший елемент на сторінці; також ви можете натискати на кнопки, перетягувати елементи, вводити текст, надсилати запити та виконувати багато інших дій.

Рисунок 2.2. – Програмні платформи фронтенд розробки

Звичайно, вже давно існують фреймворки, які полегшують роботу розробників, зменшуючи необхідність вручну писати програмний код з нуля. Таким чином, процес розробки відбувається швидше, а функціональні можливості становляться ширшими та цікавішими. Але в основі цих фреймворків завжди лежать саме вищезазначені технології.

2.3. Вимоги та структура bekend розробки

Для створення ефективної аналітичної підсистеми веб-орієнтованого середовища використовується сучасний стек технологій і підходів до розробки, який забезпечує гнучкість, масштабованість та продуктивність системи. Основний акцент робиться на розробці клієнтської частини (фронтенду) за допомогою Angular 8 з використанням мови програмування TypeScript та бібліотеки RxJS, що забезпечує реактивну роботу з даними в реальному часі. Серверна частина (бекенд) створена на основі PHP-фреймворку Laravel 5.6, який реалізує архітектуру MVC і забезпечує ефективну обробку запитів, валідацію даних і взаємодію з базою даних MySQL 8.

Для забезпечення розширюваності функціоналу застосовується архітектурний підхід CQRS, який розділяє операції запису і читання даних. Це підвищує продуктивність і спрощує підтримку системи в майбутньому. Крім того, використання Docker для контейнеризації дозволяє швидко створювати та розгортати програму в будь-якому середовищі, що гарантує її стабільну роботу.

На рисунку нижче представлено загальну архітектуру підсистеми, яка ілюструє взаємодію між базою даних, серверною частиною та клієнтським інтерфейсом.

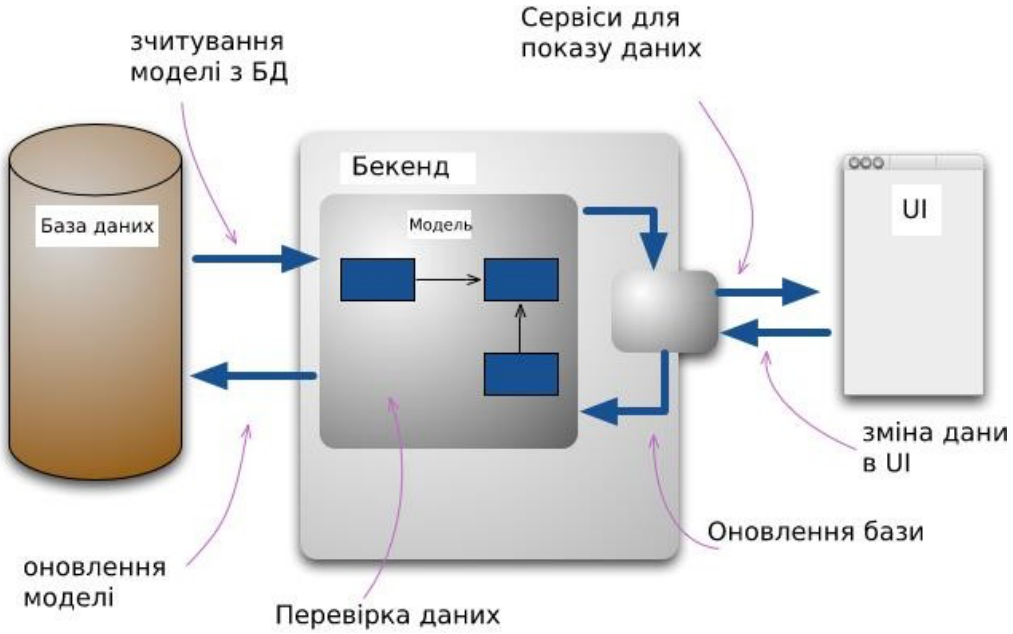

Рисунок 2.2 – Класична архітектура

На рисунку 2.2. зображено взаємодію між основними компонентами веб-додатку: базою даних, бекендом та користувацьким інтерфейсом (UI).

1. База даних: Зберігає дані у структурованій формі. Бекенд звертається до бази для зчитування або оновлення даних.

2. Бекенд: Основний логічний центр додатку. Він отримує дані з бази, перевіряє їх (валидація) та передає до користувацького інтерфейсу через спеціальні сервіси (API). Також обробляє дані, отримані від користувача через UI, і оновлює базу.

3. Користувацький інтерфейс (UI): Відображає дані користувачеві та забезпечує взаємодію з додатком. Зміни, внесені через UI, передаються до бекенду, який оновлює базу даних.

Малюнок ілюструє циклічну взаємодію між цими компонентами, що забезпечує динамічну роботу веб-додатку. Бекенд виступає як "посередник", забезпечуючи обмін даними між базою і фронтендом, при цьому гарантуючи вірність та безпеку даних.

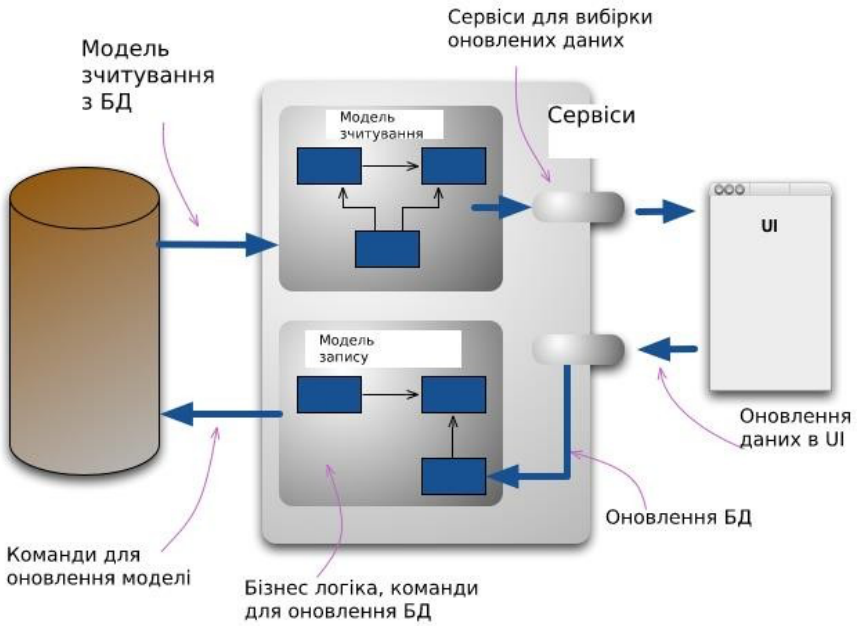

Рисунок 2.3 – CQRS архітектура

CQRS розділяє операції читання і запису даних для підвищення продуктивності, гнучкості і масштабованості системи. База даних виступає джерелом зберігання даних.

Бекенд розділений на дві частини:

Модель зчитування: відображає дані з бази, оптимізовані для швидкого отримання інформації. Дані передаються до фронтенду через сервіси, що забезпечують відображення оновлених даних.

Модель запису: відповідає за виконання команд для оновлення даних у базі. Ця модель обробляє бізнес-логіку, пов’язану із записом чи зміною даних.

Бекенд відповідає за передачу команд для оновлення моделі в базі даних і за синхронізацію даних між моделлю запису та базою.

Сервіси забезпечують взаємодію між бекендом і користувацьким інтерфейсом (UI).

Використовуються як для отримання оновлених даних (зчитування), так і для передачі змін до бази даних (запису).

Користувацький інтерфейс (UI) – відображає дані для користувача та дозволяє вносити зміни.

Усі оновлення даних у UI передаються через сервіси до моделі запису, яка оновлює базу даних. Дані зчитуються з бази даних через модель зчитування, яка передає їх у UI для відображення.

Зміни, внесені користувачем у UI, передаються через сервіси до моделі запису, де виконуються бізнес-логіка та команди для оновлення бази даних.

Оновлені дані в базі синхронізуються з моделлю зчитування для актуалізації відображення у фронтенді.

Цей підхід дозволяє системі бути більш продуктивною, забезпечуючи окремі процеси для читання і запису, а також полегшує масштабування і розвиток системи.

РОЗДІЛ 3.
РОЗРОБКА ВЕБ-ОРІЄНТОВАНОГО СЕРЕДОВИЩА ДЛЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Схема побудови інформаційної системи

Щоб зрозуміти як робить система потрібно змоделювати її архітектуру. Архітектура фронтенду (рис. 3.1) спроектовано наступним шляхом для зменшення внесення змін в панель користувача та адміністратору. Загальна бібліотека буде слугувати основною службою для постачання основного функціоналу для використання модулю статистики.

За допомогою версію контролю коду Github локально зберігається чотири репозиторію для панелі користувача, адміністратору, бекенду та загальна бібліотека для фронтенду також. Спочатку будуть створюватися сторінка статистики для користувача, а потім для адміністратора. Збірка модуля статистики завжди залежить від загальної бібліотеки оскільки без неї робота неможлива.

Деплой фронтенду буде відбуватися локально. Для кожної буде виділено окремий шлях у домені.

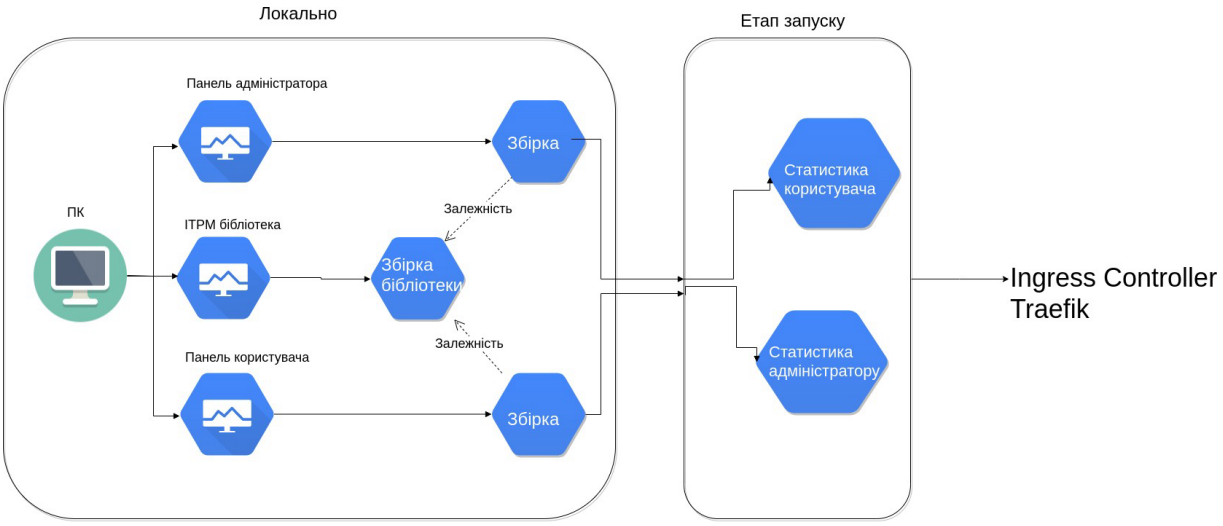

Рисунок 3.1 – Архітектура фронтенду системи

Архітектура бекенду (рисунок. 3.2) спроєктованані великі навантаження з боку операцій зчитування та операцій запису статистики в базу даних. При великій кількості підключень до бекенду, кількість його «інстансів» буде збільшено при досяганні ліміту використання пам’яті на 80 відсотків та утилізації процесору на 50 відсотків.

Масштабування бекенду, а саме баз даних для статистичних даних буде відбуватися за допомогою Docker[14].

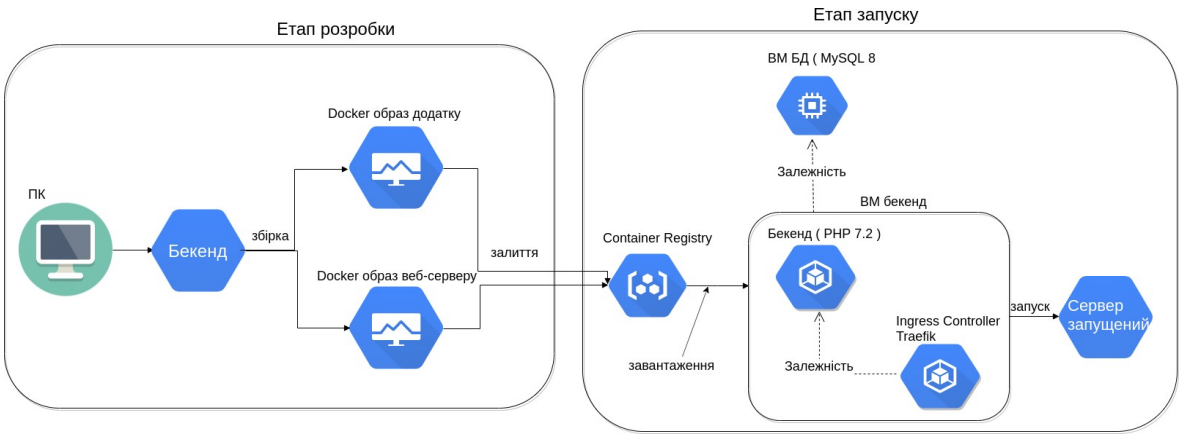

Рисунок 3.2 — Архітектура бекенду

Docker [14] - це проект з відкритим вихідним кодом для автоматизації розгортання додатків у вигляді переносних автономних контейнерів, які виконуються в хмарі або локальному середовищі (рисунок. 3.3).

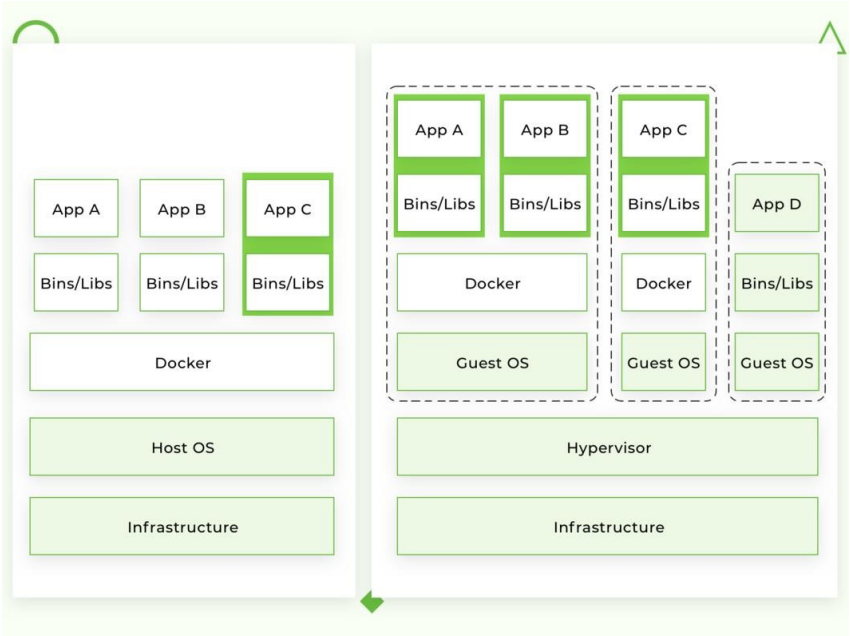

Рисунок 3.3 – Архітектура Docker

Docker має низку переваг перед іншими способами розгортання застосунків (наприклад, віртуальними машинами). Контейнери набагато легші за віртуальні машини, тому вони використовують менше ресурсів і швидше запускаються. Контейнери також легше переносити.

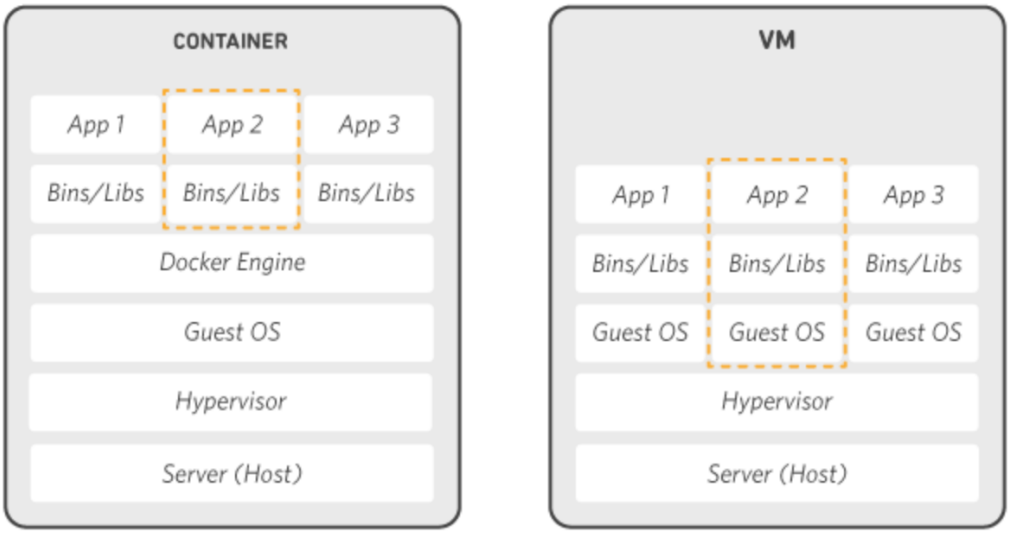

Рисунок 3.4. – Набір структурних елементів Docker

Одночасно з цим, Docker – це компанія, яка розробляє і просуває цю технологію у співпраці з постачальниками хмарних служб, а також рішень Linux і Windows, включаючи корпорацію Майкрософт.

3.2. Створення моделі інформаційної системи

Для розуміння процесів моделювання аналітичної підсистеми потрібно створити діаграму варіантів використання (Use Case) в якій буде описана модель взаємодії серверу, користувача та адміністратору. Потім потрібно створити контекстну діаграму IDEF0.

На діаграмі IDEF0 (рисунок. 3.4) зображено процес створення статистичних даних, а внизу механізми, які впливають на процес.

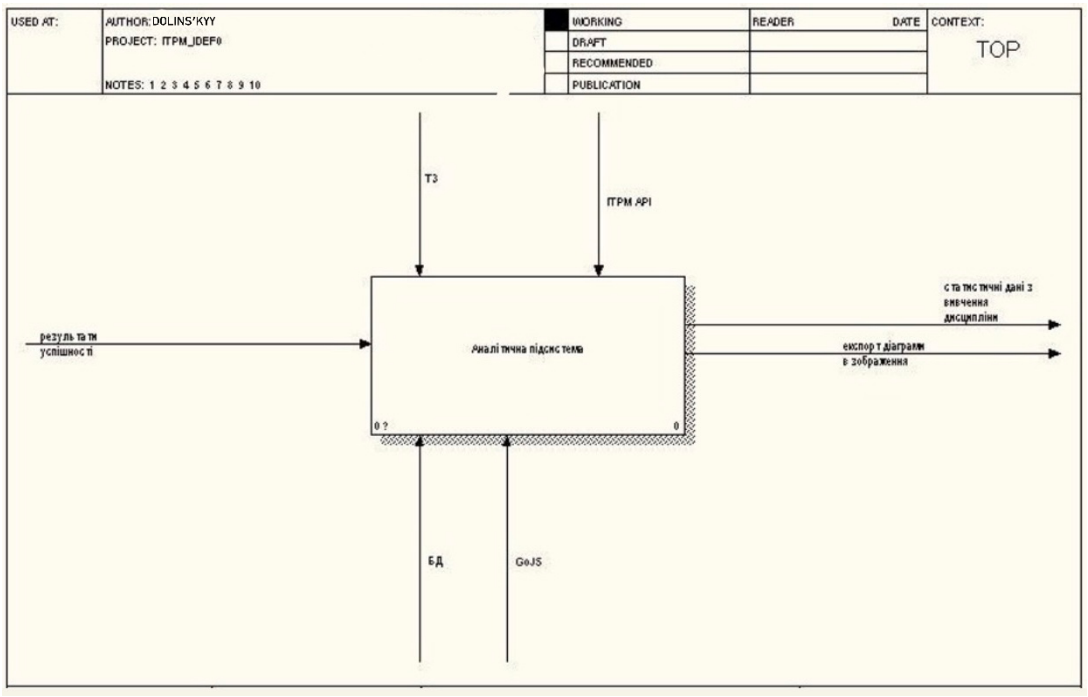

Рисунок 3.4 – Контекстна діаграма аналітичної підсистеми у нотації IDEF0

Наступним кроком декомпозиція контекстної діаграми (рис. 3.5 та рис. 3.6). Для статисти є створення діаграм у тесті. При створенні діаграми створюються унікальні UUID ідентифікатори за допомогою яких відрізняються користувачі в базі даних. За процеси, які відносяться до відображення діаграми в статистиці відповідає механізм «GoJS» [15]. Для процесу «Перегляд статистики» відповідає механізм ***«Сервер».***

Результатом процесів перегляду статистики є діаграма або зображення, яке можна завантажити як результат статистики. Статистичні дані будуть записані в базу, а при записі в базу створюються події за допомогою яких буде оновлюватися база даних яку будуть використовувати для зчитування статистичних даних.

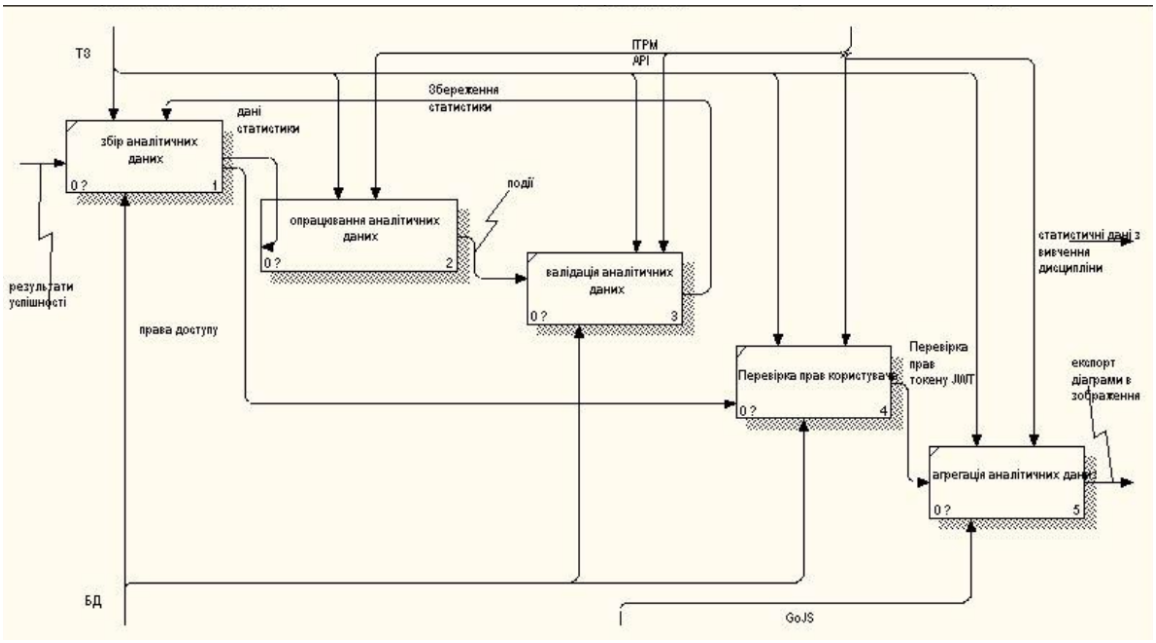

Рисунок 3.5 – Перший рівень декомпозиції процесів аналітичної підсистеми

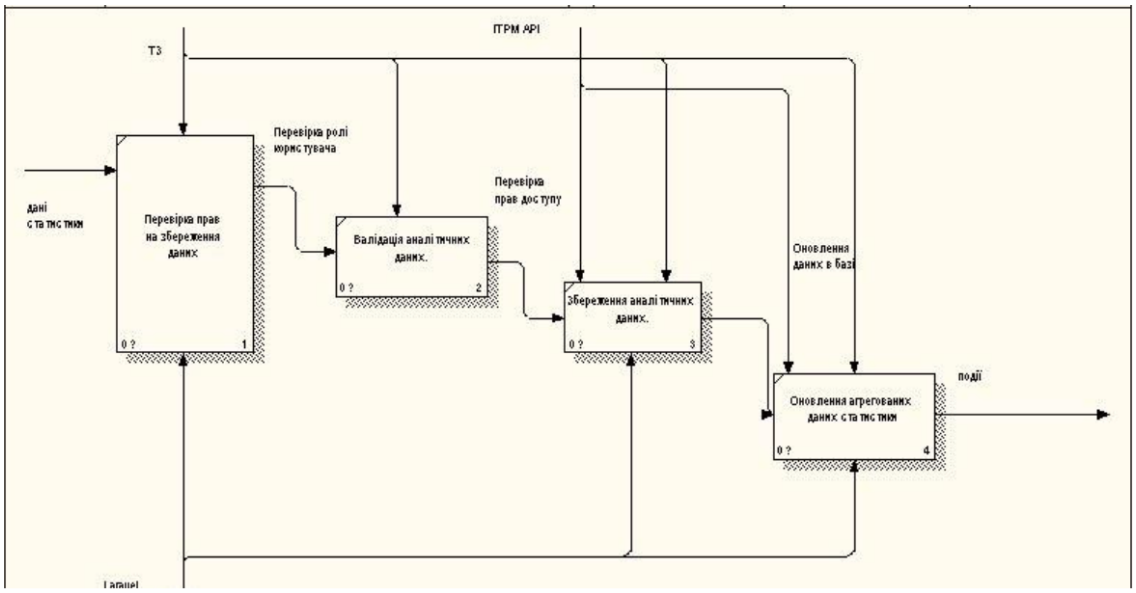

Рисунок 3.6 – Декомпозиція процесу збору аналітичних даних

Тепер створюємо діаграму варіантів використання. Виділимо 3 фактори: сервер, статистика та користувач.

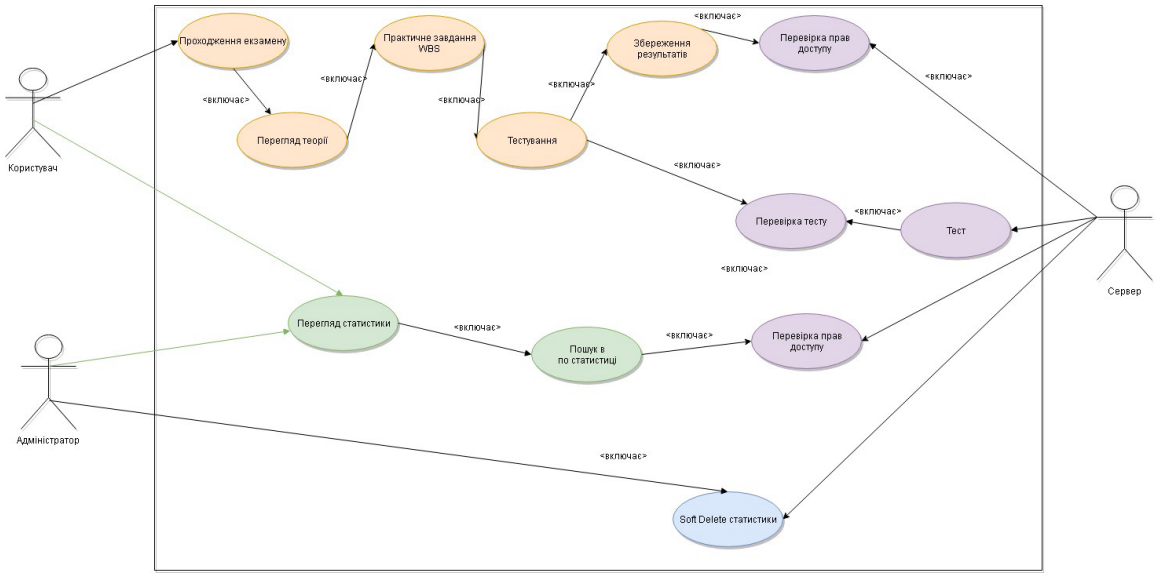

Рисунок 3.7 – Діаграма варіантів використання

Виділимо загальні варіанти використання між акторами користувач та статистика:

Варіанти використання для актора користувач:

- розмежування прав між користувачами;
- розмежування прав між адміністратором та користувачем;
- перегляд аналітики по статусу виконання екзамену;
- перегляд аналітики користувача по діаграмі;
- перегляд аналітики користувача по тесту;
- перегляд аналітики статистики користувача по темі.

Використання для актора адміністратор:

- розмежування прав між користувачами;
- розмежування прав між адміністратором та користувачем;
- перегляд аналітики по статусу виконання екзамену;
- перегляд аналітики користувача по діаграмі;
- перегляд конкретної аналітики користувача по темі.
- перегляд аналітики по конкретному користувачу;
- soft delete аналітики.

Варіанти використання для актора сервер:

- видача лістингу аналітики;

- посторінкова навігація по аналітики;
- фільтрація аналітики по статусу виконання;
- фільтрація аналітики по користувачу;
- soft delete аналітики;
- перевірка прав доступу.

У результаті була створена діаграма варіантів використання (рис. 3.7).

3.3. Проектування і реалізація бази даних

Для забезпечення ефективної роботи аналітичної системи використовується підхід ***Event Sourcing,*** який дозволяє зберігати всі дії користувачів у вигляді подій. Цей підхід забезпечує точне відображення поточного стану даних і дозволяє легко відтворювати історію змін. У рамках цієї архітектури використовуються дві бази даних: одна для зберігання "сирих" подій (операцій запису), а інша — для читання агрегованих даних.

Сховище подій записує всі дії користувачів, такі як початок тесту, створення діаграми чи завершення завдання. Ці події обробляються, агрегуються й публікуються у вигляді структурованих даних, що використовуються для побудови звітів і відображення інформації в інтерфейсі. Такий підхід дозволяє швидко обробляти запити до системи, оскільки база для читання містить лише необхідні для користувача дані.

Event Sourcing забезпечує масштабованість і зручність інтеграції з іншими системами. Усі події можуть бути використані для створення матеріалізованих даних, а також для аналізу історії дій користувачів. На рисунку нижче представлено схему архітектури, яка демонструє взаємодію між сховищем подій, бізнес-логікою, матеріалізованими даними та інтерфейсом користувача.

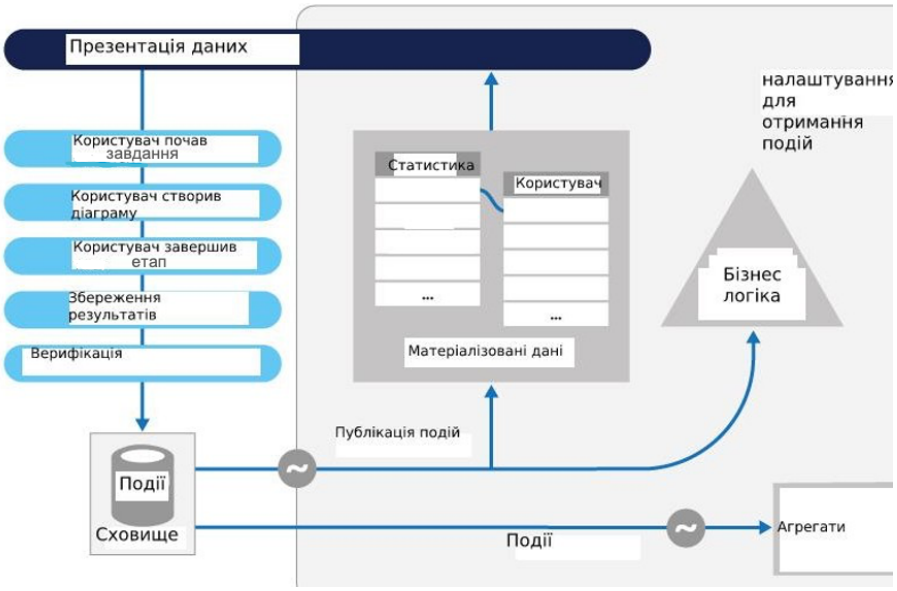

Рисунок 3.8 – Архітектура EventSourcing

На рисунку 3.8 показано архітектуру Event Sourcing у контексті роботи системи:

1. Події- дії користувача, такі як "почав тест" або "створив діаграму", записуються в сховище подій.

2. Сховище подій- це база даних, де зберігаються всі події системи. Події записуються послідовно, створюючи журнал дій.

3. Агрегація-дані з сховища подій обробляються (агрегуються) для створення статистики, матеріалізованих даних (наприклад, таблиць результатів тестів, діаграм тощо).

4. Матеріалізовані дані - це оброблені дані, які використовуються для зручного представлення користувачу або інтеграції з бізнес-логікою.

5. Бізнес-логіка - налаштовує обробку подій та формує інтеграцію з іншими компонентами системи.

6. Презентація даних - оброблені дані відображаються користувачу у вигляді таблиць, діаграм або звітів.

Ця архітектура забезпечує точність даних, масштабованість системи та можливість роботи з великою кількістю користувачів і подій. Вона також дозволяє створювати точні звіти й аналізувати дії користувачів на основі їхньої історії взаємодій із системою.

Архітектура RESTful (рисунок 3.9) забезпечує ефективну взаємодію між бекендом і клієнтом через HTTP-запити. У цій архітектурі клієнт відправляє запити для отримання, створення, оновлення або видалення ресурсів, а сервер обробляє ці запити та повертає відповідь у форматі JSON. Такий підхід гарантує, що клієнт і сервер дотримуються узгодженого стандарту обміну даними, забезпечуючи зрозумілість і зручність інтеграції.

Маршрути на сервері мають чітке розмежування за префіксами, що дозволяє ефективно керувати доступом до ресурсів. За цими префіксами ховається система розмежування прав доступу, яка перевіряє, чи має клієнт відповідні права для виконання запиту. Це забезпечує безпеку та дозволяє уникати несанкціонованого доступу до даних.

Для коректної роботи взаємодії клієнт повинен відправляти запити у відповідному форматі, який підтримує сервер, і сервер, у свою чергу, має відповідати клієнту даними в такому ж форматі. Це створює контракт між клієнтом і бекендом, якого обидві сторони дотримуються для забезпечення стабільної та надійної роботи системи.

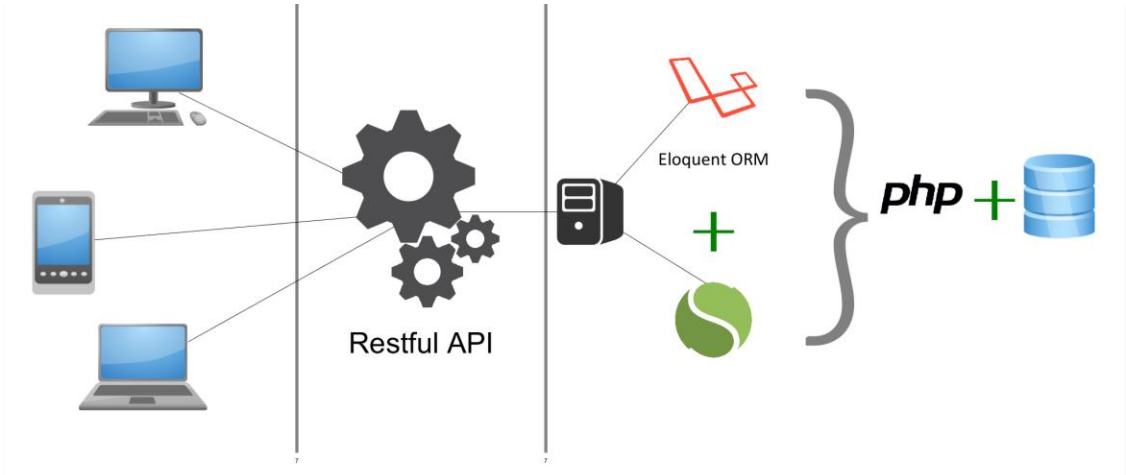

Рисунок 3.9 – Архітектура RESTful бекенду

1. Клієнт відправляє HTTP-запити до сервера для виконання операцій над ресурсами (отримання, створення, оновлення чи видалення).

2. Бекенд:

- Обробляє запити клієнта.

- Перевіряє права доступу на основі маршрутів із чітко визначеними префіксами.
- Відповідає клієнту у форматі JSON, забезпечуючи стандартизацію обміну даними.

3. Маршрути розділені за префіксами, що дозволяє організувати логіку доступу до різних частин системи, наприклад, для авторизованих або неавторизованих користувачів.

4. Формат JSON використовується для передачі даних між клієнтом і сервером, що забезпечує простоту, компактність і зрозумілість обміну інформацією.

Ця архітектура дозволяє легко масштабувати систему, інтегрувати її з іншими сервісами та забезпечувати зручну і безпечну взаємодію між клієнтським інтерфейсом і серверною частиною.

Проектування структури проекту передбачає його поділ на чотири основні частини для забезпечення чіткого розподілу функціоналу:

1. *Фронтенд для користувача*: Інтерфейс, з яким взаємодіє кінцевий користувач.

2. *Фронтенд для адміністратора*: Інтерфейс для управління системою, доступний адміністраторам.

3. *Бекенд:* Серверна частина, яка обробляє запити, виконує бізнес-логіку та взаємодіє з базою даних.

4. *Загальна бібліотека:* Набір загальних функцій і класів, які можуть використовувати всі частини проекту.

Архітектура бекенду забезпечує ефективну внутрішню взаємодію компонентів, як показано на рисунку 3.10

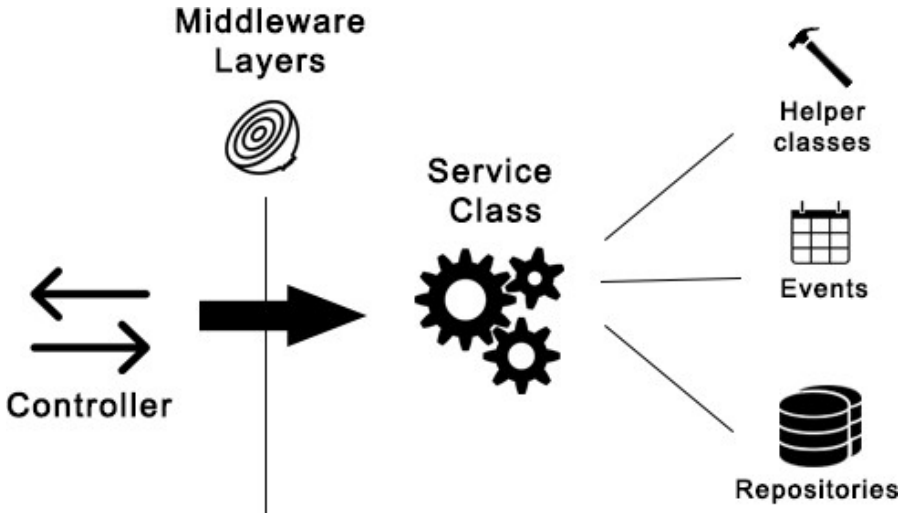

Рисунок 3.10 – Архітектура взаємодії фреймворку

Клієнт відправляє запит за певним маршрутом, де його спершу обробляє ***Middleware*** — проміжне програмне забезпечення. Middleware перевіряє, чи має користувач доступ до цього маршруту. Якщо перевірка успішна, запит передається до контролера, який викликає сервіси для подальшої обробки.

Сервіси взаємодіють із базою даних через репозиторії, що відповідають за читання і запис даних. У разі запису статистичних даних створюється подія, яка публікується через шину даних. Підписники цієї події отримують її та обробляють відповідно до своєї логіки.

Для виконання допоміжних функцій, які не пов’язані з бізнес-логікою (наприклад, форматування чи валідація даних), використовуються Helper-класи. Вони спрощують роботу з даними, але не змінюють їхньої суті.

На малюнку 3.11 зображена архітектура бази даних, яка містить таблиці для зберігання агрегованих даних, отриманих із «сирих» результатів екзамену.

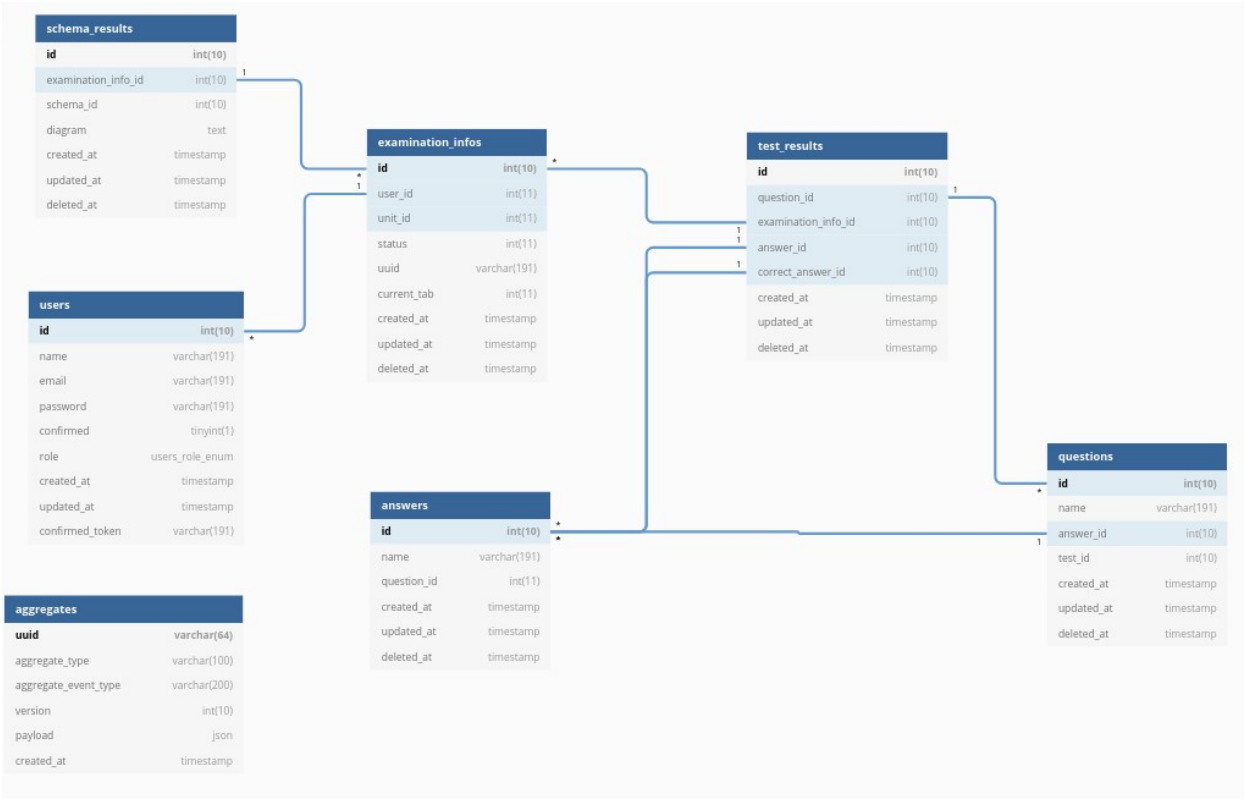

Рисунок 3.11 – Архітектура бази даних

На рисунку представлена структура бази даних із взаємозв’язками між таблицями:

- Таблиця users берігає інформацію про користувачів: їхні імена, електронну пошту, пароль, роль тощо.Відповідає за ідентифікацію користувачів у системі.
- Таблиця examination_infos зберігає інформацію про проведені екзамени для кожного користувача, включаючи статус і деталі про екзамен. Пов’язана з таблицею users через поле user_id.
- Таблиця schema_results зберігає результати екзаменів і діаграм. Зв’язана з таблицею examination_infos через поле examination_info_id.
- Таблиця test_results містить результати тестів, включаючи відповіді користувача, правильні відповіді та запитання. Пов’язана з таблицею examination_infos.

- Таблиця questions зберігає запитання для тестів. Пов’язана з таблицею test_results.
- Таблиця answers зберігає варіанти відповідей для кожного запитання.Пов’язана з таблицею questions.
- Таблиця aggregates зберігає агреговані дані (наприклад, статистику результатів), які отримуються з «сирих» даних.

Ця база даних побудована на основі чітких взаємозв’язків між таблицями, що забезпечує фективне зберігання даних, легкість доступу до агрегованої статистики та гнучкість у додаванні нових функцій і аналізу даних.

Такий підхід дозволяє системі зберігати як деталізовані, так і агреговані результати, що спрощує роботу із статистикою та забезпечує продуктивність і масштабованість системи.

РОЗДІЛ 4.
РЕАЛІЗАЦІЯ ВЕБ-СЕРЕДОВИЩА ІНФОРМАЦІЙНОЇ СИСТЕМИ «УПРАВЛІННЯ ІТ ПРОЕКТАМИ»

4.1. Налаштування та реалізація бекенду

Для того щоб створити аналітичну підсистему спочатку потрібно реалізувати частину на бекенді. Для цього потрібно створити файл routes.php з маршрутами куди клієнт буде звертатися за статистикою. Код бекенду наведений у додатку А.

Routes.php буде містити в собі наступний код:

```
Route::group([
'prefix' =>'statistics',
], function () {
Route::get('/', 'StatisticController@filter');
Route::get('{id}', 'StatisticController@show');
Route::post('soft-delete/{id}', 'StatisticController@softDelete');
});
```

Наступним кроком потрібно буде міграції для збереження статистичних даних та відображення (рис. 4.1).

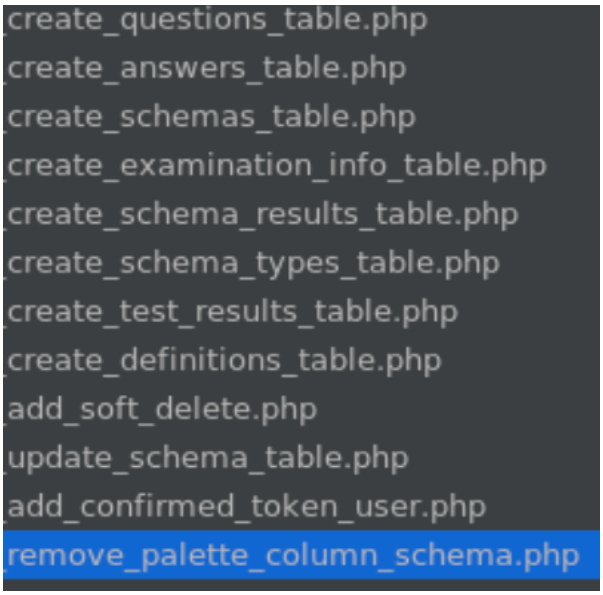

Рисунок 4.1 — Міграції для бази даних

Потім створюємо моделі даних SchemaResult.php, TestResult.php, ExaminationInfo.php, Aggregates, SchemaType для таблиць (рис. 4.2).

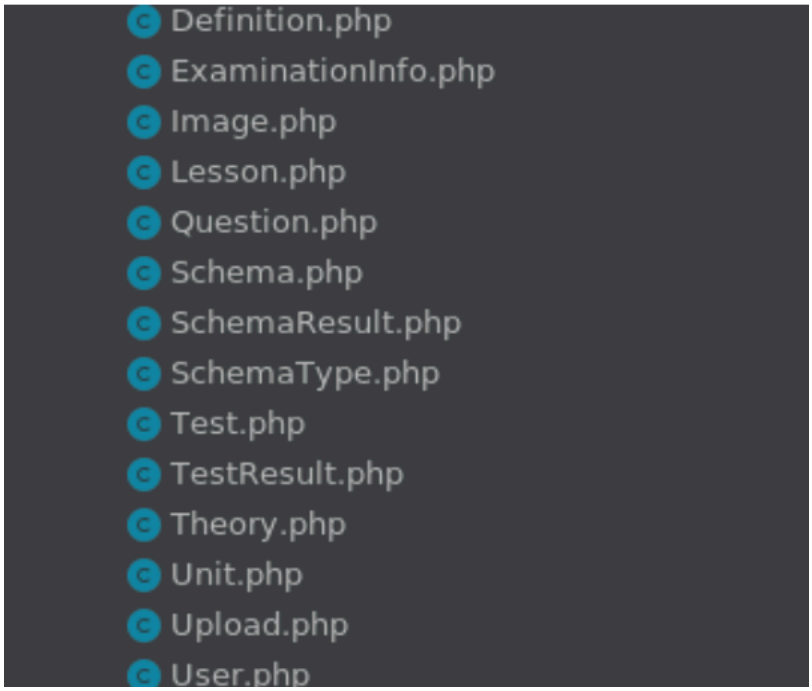

Рисунок 4.2 — Моделі даних

Створюємо репозиторії для моделей даних, щоб отримувати та оброблювати дані для зчитування та запису таблиці (рис. 4.3).

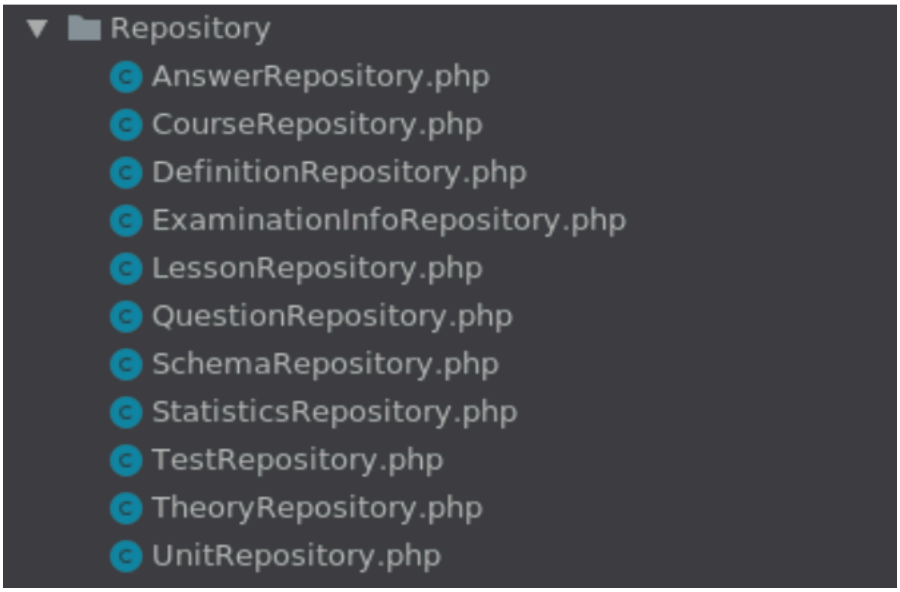

Рисунок 4.3 — Репозиторії моделей

Наступним кроком створюємо політики доступу для розмежування прав користувачів та адміністратору (рисунок 4.4).

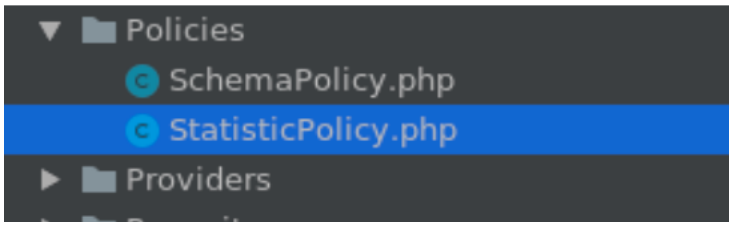

Рисунок 4.4 — Політика розмежування статистики

Код розмежування статистичних даних.

```
class StatisticPolicy
{
use HandlesAuthorization;
const CAN_SOFT_DELETE = 'softDelete';
const CAN_VIEW = 'view';
public function softDelete(User $user, ExaminationInfo $statistic)
{
return $user->role === User::ROLE_ADMIN;
}
public function view(User $user, ExaminationInfo $statistic)
{
return ($user->id === $statistic->user_id && !$statistic->is_soft_deleted)
|| $this->softDelete($user, $statistic);
}
}
```

Створюємо контролер StatisticController.php для обробки запиту на отримання статистичних даних (рис. 4.5).

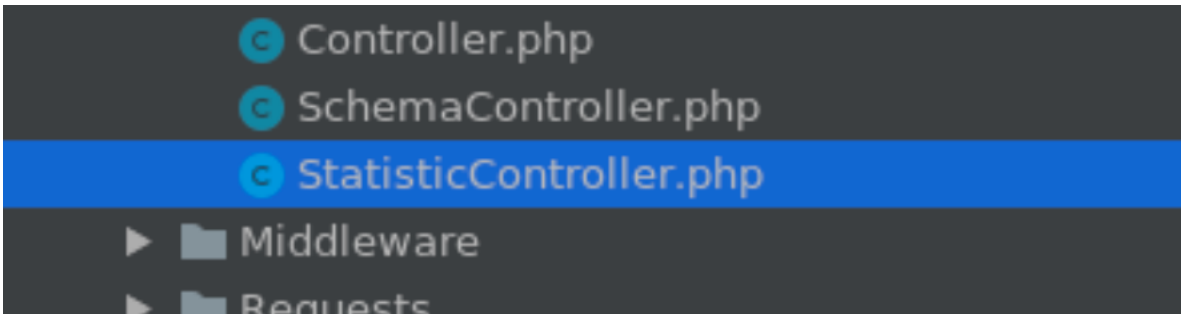

Рисунок 4.5 — Контролер для обробки статистики

Контролер StatisticController.php буде містити в собі метод для фільтрації статистики за різними критеріями, метод для отримання окремих статистичних даних та метод для м’якого видалення даних. Метод пошуку статистики за різними критеріями контролеру

StatisticController.php викликає репозиторій StatisticsRepository.php та надає можливість фільтрувати статистику за такими параметрами як: статус проходження екзамену, фільтрування по результату виконання екзамену, фільтрація по імені теми та фільтрування по користувачу. Також репозиторій для статистики надає можливість посторінкової навігації.

4.2. Налаштування та реалізація фронтенду

Для створення модуля статистики який буде використовуватися у панелі користувача та адміністратора потрібно створити модуль у загальній бібліотеці. Для цього потрібно створити компоненти для відображення лістингу статистики по екзаменам, сторінку для відображення діаграми, сторінку для відображення результатів тестів. На сторінці лістингу статистики потрібно створити фільтрацію по користучам, по імені теми, по статусу та експорт діаграми. Код фронтенду наведений у додатку Б. Створюємо необхідну структуру файлів модулю статистики (рисунок 4.6).

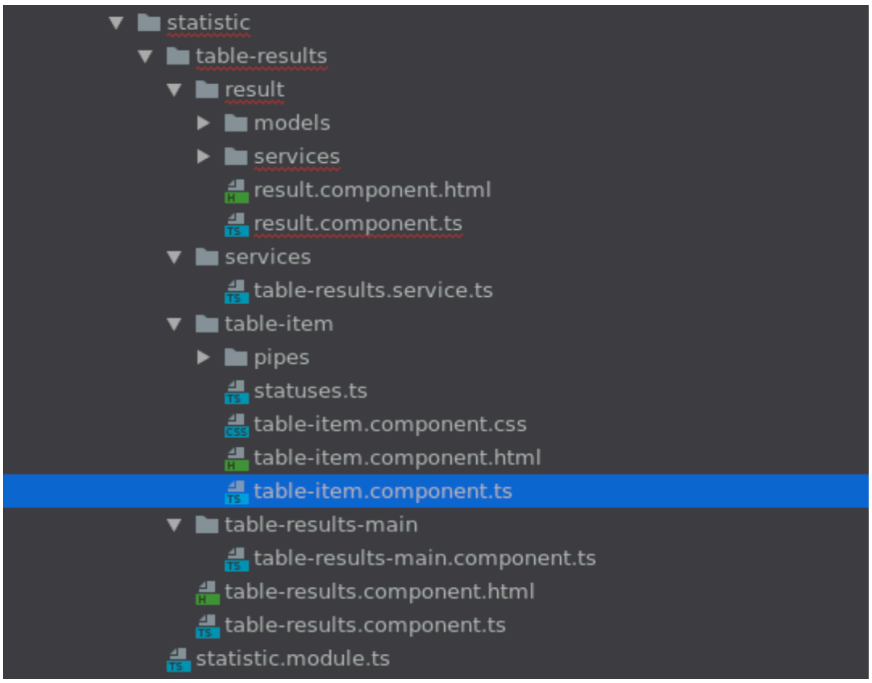

Рисунок 4.6 — Модуль статистики

Підключення модулю статистики на панелі користувача (рисунок 4.7).

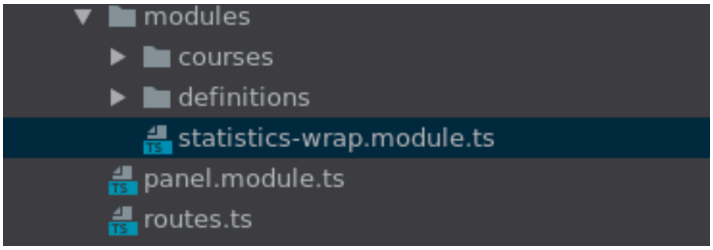

Рисунок 4.7 — Підключення модулю статистики

Фільтрація статистики відбувається за такими критеріями як владка, статус татип статистики. Фільтрація по критерію вкладки (рисунок. 4.8).

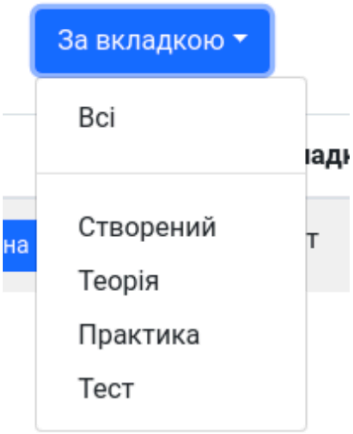

Рисунок 4.9 – Фільтрація за вкладкою

Фільтрація статистики по типу видаленої статистики чи показу всієї статистики (рисунок. 4.10).

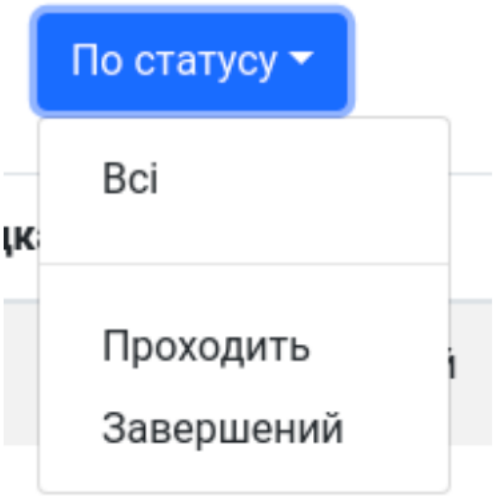

Рисунок 4.10 – Фільтрація за статусом

Сторінка відображення статистики по вибраній темі з практичної частини (рисунок. 4.10). На рисунку можна побачити кількість правильних відповідей та на самій діаграмі текст «Правильно» означає що блок перетягнуто правильно.

4.3 Деплой додатку

Для деплою додатку на операційній системі Fedora використовується Docker за допомогою якого буде відбуватися запуск додатку. Спочатку описуємо файл для конфігурації всіх контейнерів. Опис проксі серверу, який буде давати доступ до сервісів із мережі Інтернет:

```
version: '2'
services:
traefik:# IngressController
image: traefik:1.7
restart: always
ports:
- "80:80"
- "8080:8080"
volumes:
- ./traefik.toml:/etc/traefik/traefik.toml
- /var/run/docker.sock:/var/run/docker.sock
networks:
- itpm
```

Опис сервісу для запуску бекенду, який залежить від запуску сервісу для кешування та бази даних:

```
app:# Бекенд
build:
context: ./backend
dockerfile: docker/app.docker
working_dir: /var/www
volumes:
./backend:/var/www
.dev.env:/var/www/.env49
command: php artisan serve --port=8080 --host=0.0.0.0
ports:
- 9001:8080
links:
- database
- cache
-labels:
```

-"traefik.frontend.rule=Host:zzz-yashka.asuscomm.com"

"traefik.frontend.rule=PathPrefixStrip:/backend"

"traefik.enable=true"

networks:

- itpm

Опис налаштувань панелі користувача з налаштуванням відкритих портів, підключення необхідних директорій для роботи додатку:

user_panel:# Панель користувача

image: node:12

user: node

working_dir: /var/www/app

command: npm run start

volumes:

- ./user-panel:/var/www/app

- ./itpm:/var/www/itpm

ports:

- 3001:4200

labels:

- "traefik.frontend.rule=Host:zzz-yashka.asuscomm.com"

- "traefik.frontend.rule=PathPrefixStrip:/user-panel"

- "traefik.enable=true"

networks:

- itpm

Опис налаштувань панелі адміністратору з налаштуванням відкритих портів:

```
admin_panel:# Панель адміністратору
image: node:12
user: node
working_dir: /var/www/app
command: npm run start
volumes:
- ./admin-panel:/var/www/app
- ./itpm:/var/www/itpm
ports:
- 3002:4200
labels:
- "traefik.frontend.rule=Host:zzz-yashka.asuscomm.com"
- "traefik.frontend.rule=PathPrefixStrip:/admin-panel"
- "traefik.enable=true"
networks:
- itpm
```

Опис налаштувань бази даних з конфігурацією підключення, пере направлення портів з контейнеру та встановлення внутрішньої мережі:

```
database: # База даних для зчитування
image: mysql:8
volumes:
- dbdata:/var/lib/mysql
ports:
- 3306:3306
environment:
- "MYSQL_DATABASE=test"
- "MYSQL_ROOT_PASSWORD=root"
- "MYSQL_USER=root"
```

```
- "MYSQL_PASSWORD=root"
labels:
- "traefik.enable=false"
networks:
- itpm
database_es:# База даних для запису
image: mysql:5.7
volumes:
- dbdata:/var/lib/mysql
ports:
- 3306:3306
environment:
- "MYSQL_DATABASE=test"
- "MYSQL_ROOT_PASSWORD=root"
- "MYSQL_USER=root"
- "MYSQL_PASSWORD=root"
labels:
- "traefik.enable=false"
networks:
- itpm
```

Опис налаштувань сервісу для кешування з конфігурацією підключення, перенаправлення портів з контейнеру та встановлення внутрішньої мережі:

```
cache:# Сервіс для кешування статистики
image: redis
ports:
- 6378:6379
environment:
- REDIS_PASSWORD=kadj7VkdihjjiJI11
labels:
- "traefik.enable=false"52
```

networks:

- itpm

Конфігурація сховищ даних для бази даних використовується для постійного збереження бази даних при перезапуску сервісу:

volumes: # Віртуальні сховища даних

dbdata: # для баз даних

Конфігурація налаштувань внутрішньої мережі, яка використовується в усіх сервісах для забезпечення ізоляції безпеки роботи мережі:

networks: # Віртуальні мережі для додатку

itpm:

Наступним кроком описуємо файл для TraefikIngressController, файл повинен називатися traefik.toml та має містити в соби опис налаштувань веб-серверу:

```
logLevel = "DEBUG"
[file]
watch = true
[docker]
endpoint = "unix:///var/run/docker.sock"
domain = "docker.localhost"
watch = true
exposedByDefault = false
[api]
dashboard = true
```

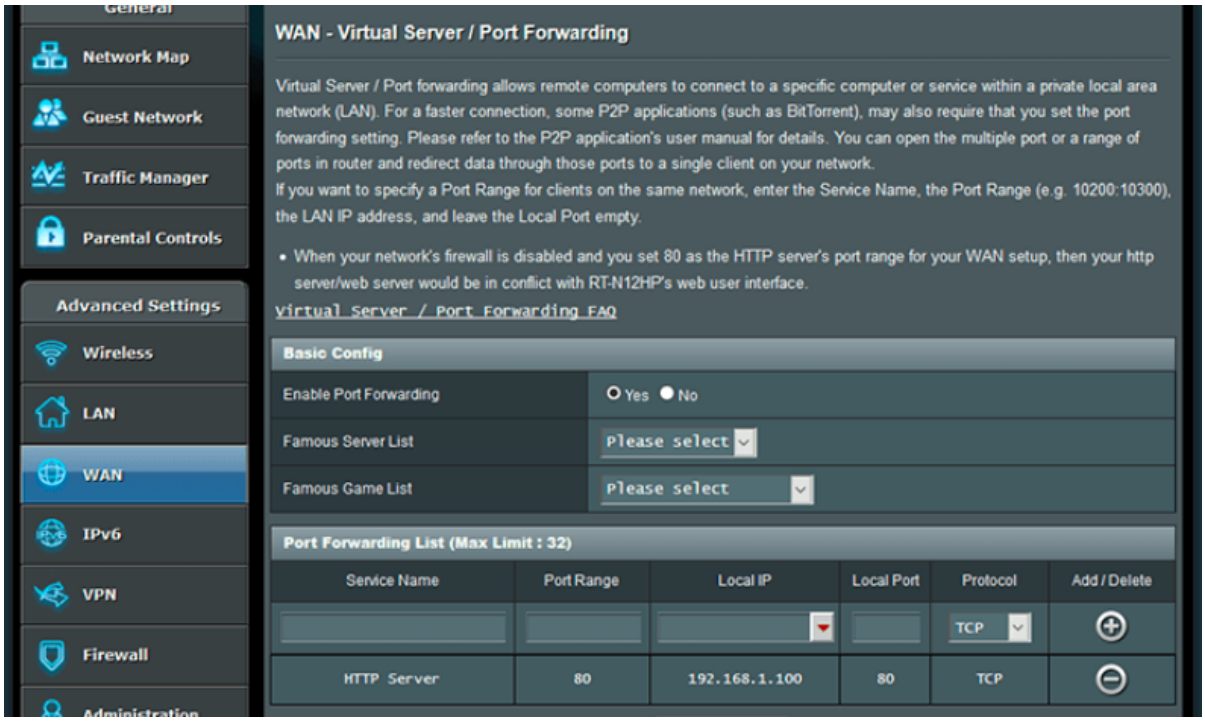

Рисунок 4.14 – Налаштування маршрутизатору

Налаштування маршрутизатора Asus RT-N12 D1 для перенаправлення 80 порту від локальної мережі на зовнішню для того, щоб мати доступ до додатку з мережі Інтернет (рисунок 4.14).

Адміністрування веб-сервера – це складний процес, що вимагає глибокого розуміння інфраструктури, програмного забезпечення і безпеки. Один із ключових аспектів цього процесу – налаштування серверного середовища.

1. Вибір операційної системи:

Першим кроком у налаштуванні сервера є вибір операційної системи (ОС). Популярні опції включають в себе Linux (наприклад, Ubuntu, CentOS) і Windows Server. Вибір ОС залежить від конкретних потреб і вмінь адміністратора.

2. Встановлення веб-сервера:

Другим кроком є встановлення веб-сервера, такого як Apache, Nginx або Microsoft IIS. Вибір веб-сервера залежить від потреб вашого проекту та платформи ОС.

3. Налаштування віртуальних хостів:

Віртуальні хости дозволяють одному серверу господарювати кількома веб-сайтами. Налаштування віртуальних хостів допомагає розподілити трафік і

забезпечити ізоляцію між сайтами.

4. Конфігурація бази даних:

Багато веб-сайтів використовують бази даних для зберігання інформації. Налаштування бази даних (наприклад, MySQL, PostgreSQL, або Microsoft SQL Server) і з'єднання з веб-сервером – це важливий аспект адміністрування.

5. Захист:

Забезпечення безпеки серверного середовища – це критично важливий аспект. Це включає в себе встановлення брандмауера, налаштування прав доступу, шифрування комунікацій (SSL/TLS), і регулярне оновлення програмного забезпечення.

6. Моніторинг і журналювання:

Важливо мати систему моніторингу, яка дозволяє відстежувати роботу сервера і виявляти проблеми. Журналювання (логування) подій допомагає аналізувати помилки і інциденти.

7. Оптимізація продуктивності:

Настроюючи серверне середовище, слід дбати про оптимізацію продуктивності. Це може включати в себе кешування, стискку, тонку настройку ресурсів і т. д.

8. Резервне копіювання і відновлення:

Забезпечення регулярного резервного копіювання даних і можливості відновлення допомагає захистити вашу інформацію від втрати.

РОЗДІЛ 5.
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1. Розробка логіко-імітаційної моделі виникнення травм і аварій

Розробка логіко-імітаційної моделі виникнення травм і аварій є складним і важливим етапом у розробці систем безпеки на виробництвах, в транспорті та інших галузях. Така модель дозволяє зрозуміти механізми виникнення аварійних ситуацій, спрогнозувати ймовірні наслідки, а також розробити ефективні заходи для запобігання інцидентам.

Методикою оцінки рівня небезпеки робочих місць, машин, виробничих процесів та окремих виробництв передбачено пошук об’єктивного критерію рівня небезпеки для конкретного об’єкта [1]. Таким показником вибрана ймовірність виникнення аварії, травми залежно від явища, що досліджується.

Для побудови логіко-імітаційної моделі процесу, формування і виникнення аварії та травми в процесі створення мікрокліматичних умов у приміщенні оцінюють відповідні небезпечні події. Кожній іх них присвоїмо ймовірність виникнення:

Шифр	Назва події	Ймовірність
P_1	Відсутність захисного заземлення	0,02
P_2	Пошкодження захисного заземлення	0,04
P_3	Спрацювання складових захисту	0,1
P_4	Неправильна експлуатація захисту	0,02
P_5	Відсутність профілактичних заходів	0,2
P_6	Відсутність захисного щита	0,12
P_7	Недотримання правил вибору взуття	0,15
P_8	Незнання правил техніки безпеки	0,1
P_9	Відсутність засобів індивідуального захисту	0,2
P_{10}	Легковажність	0,08

На основі наведених подій будуємо матрицю логічних взаємозв’язків між окремими пунктами, графічна інтерпретація якої зображено на рис. 5.1.

Розрахуємо ймовірності виникнення подій, що формують логіко-імітаційну

модель процесів створення мікрокліматичних умов. Розглянемо травмонебезпечну ситуацію, що виникає за умови роботи працівників із електронебезпекою.

Підставивши дані ймовірностей базових подій у формулу, отримаємо ймовірність події 13: $P_{13} = 0{,}2 + 0{,}4 - 0{,}2 \cdot 0{,}4 = 0{,}0592$.

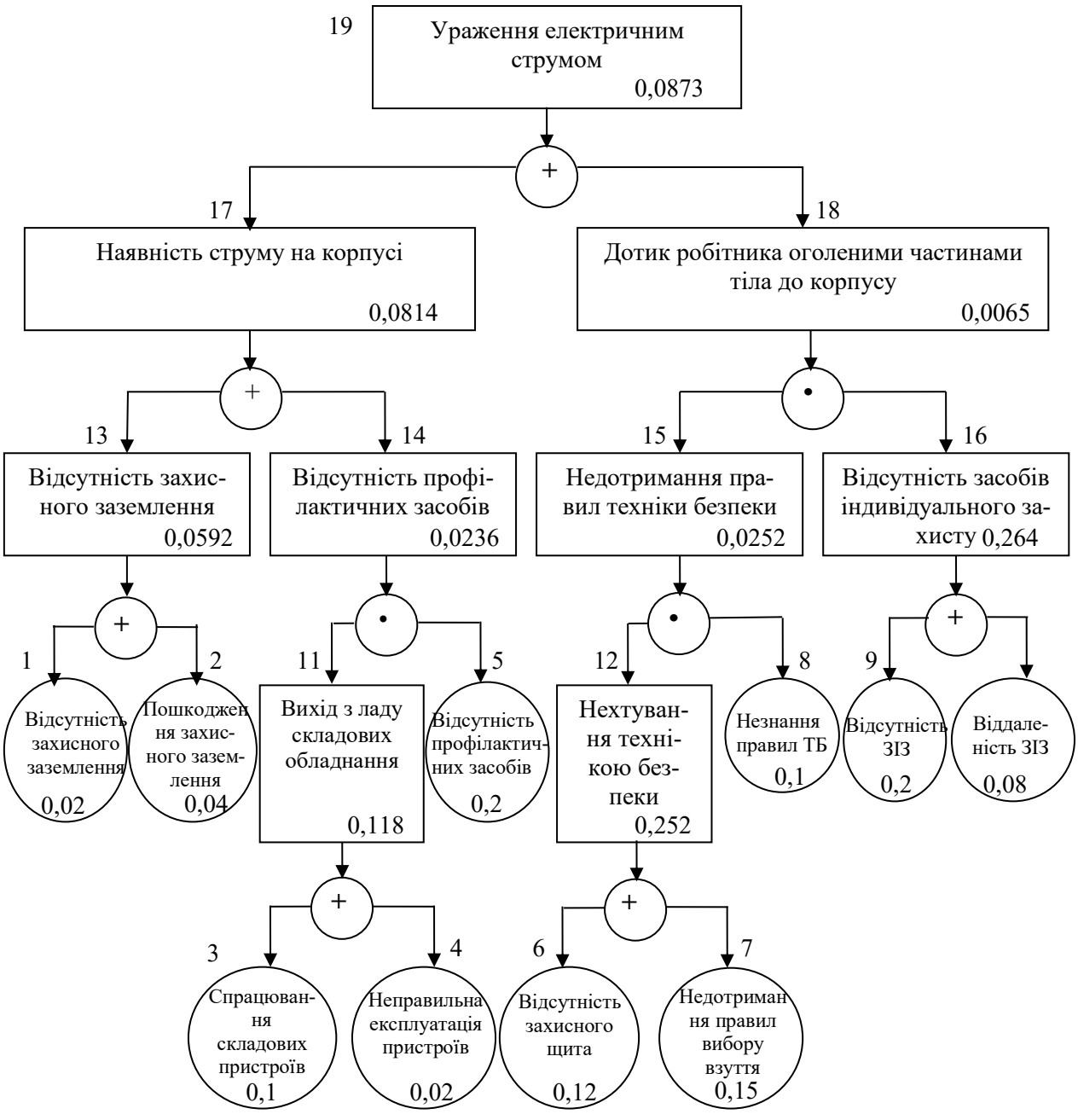

Рис. 5.1. Матриця логічних взаємозв’язків між окремими подіями травмонебезпечної ситуації

Аналогічно визначаємо ймовірність інших подій:

$$P_{11} = P_4 + P_5 - P_4P_5 = 0,3 + 0,4 - 0,3 \cdot 0,4 = 0,118.$$

$$P_{12} = P_6 + P_7 - P_6P_7 = 0,3 + 0,5 - 0,3 \cdot 0,5 = 0,252.$$

$$P_{16} = P_9 + P_{10} - P_9P_{10} = 0,2 + 0,15 - 0,2 \cdot 0,15 = 0,264.$$

$$P_{14} = P_{11} \cdot P_5 = 0,118 \cdot 0,2 = 0,0236.$$

$$P_{15} = P_{12} \cdot P_8 = 0,252 \cdot 0,1 = 0,0252.$$

$$P_{17} = P_{13} + P_{14} - P_{13} \cdot P_{14} = 0,592 + 0,0236 - 0,0592 \cdot 0,0236 = 0,0814.$$

$$P_{18} = P_{15} \cdot P_{16} = 0,264 \cdot 0,0252 = 0,0065.$$

$$P_{19} = P_{17} + P_{18} - P_{17} \cdot P_{18} = 0,0065 + 0,0814 - 0,0065 \cdot 0,0814 = 0,0873.$$

Таким чином, ймовірність перекидання машини та наслідкового виникнення травми працівника є досить мала і становить – $P_{19} \approx 0,0873$.

5.2. Планування заходів із покращення умов праці

Покращення умов праці є важливим аспектом не лише для забезпечення здоров'я та безпеки працівників, а й для підвищення ефективності роботи, мотивації персоналу та загальної продуктивності організації. Всі заходи, спрямовані на покращення умов праці, мають бути ретельно сплановані і включати комплексний підхід, який враховує фізичні, психологічні, соціальні та організаційні фактори.

До заходів щодо покращення умов праці належать всі види діяльності, спрямовані на попередження, нейтралізацію або зменшення негативної дії шкідливих і небезпечних виробничих факторів на працівників.

Рівень умов праці оцінюють порівнянням за фактичними і нормативними значеннями узагальнених (групових) показників.

Заходи щодо поліпшення умов праці здійснюють з метою створення безпечних умов праці шляхом:

- доведення до нормативного рівня показників виробничого середовища за елементами умов праці;

- захисту працівників від дії небезпечних і шкідливих виробничих факторів.

До показників ефективності заходів щодо поліпшення умов праці належать:

а) зміни стану умов праці:

- зміна кількості засобів виробництва, приведених у відповідність до вимог стандартів безпеки праці;
- покращання санітарно-гігієнічних показників;
- покращання психофізичних показників, зменшення фізичних і нервово-психічних навантажень, в т.ч. монотонних умов праці;
- покращання естетичних показників, раціональне компонування робочих місць і впорядкування робочих приміщень;

б) соціальні результати заходів:

- збільшення кількості робочих місць, що відповідають нормативним вимогам;
- зниження рівня виробничого травматизму;
- зменшення кількості випадків професійних захворювань;
- зменшення плинності кадрів через незадовільні умови праці;
- престиж та задоволення працею.

Отже, на покращення охорони праці потрібно виділити кошти на відновлення вентиляційних систем у ремонтних майстернях, естетично оформити приміщення офісу, відновити кабінет з охорони праці, поновити протипожежний інвентар.

5.3. Безпека в надзвичайних ситуаціях

Безпека в надзвичайних ситуаціях (НС) є важливим аспектом для забезпечення захисту життя, здоров'я людей, збереження матеріальних цінностей та навколишнього середовища. Надзвичайні ситуації можуть бути різноманітними за природою та масштабом, від природних катастроф (землетрусів, повеней,

ураганів) до техногенних аварій (вибухи, пожежі, витоки небезпечних речовин). Усі ці ситуації вимагають чіткої організації заходів, спрямованих на запобігання, мінімізацію наслідків і швидке відновлення нормальних умов життєдіяльності.

Актуальність проблеми природно-техногенної безпеки для населення і території, зумовлена зростанням втрат людей, що спричиняється небезпечними природними явищами, промисловими аваріями та катастрофами. Ризик надзвичайних ситуацій природного та техногенного характеру невпинно зростає, тому питання захисту цивільного населення від надзвичайних ситуацій на сьогодні є дуже важливе.

У системі цивільної оборони окремого господарства необхідно забезпечити захист населення таким чином:

Укриття в захисних спорудах, якому підлягає усе населення відповідно до приналежності, досягається створенням фонду захисних споруд.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об’єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення його у позаміській зоні.

Медичний захист проводиться для зменшення ступеня ураження людей, своєчасного надання допомоги постраждалим та їх лікування, забезпечення епідеміологічного благополуччя в районах надзвичайних ситуацій.

Радіаційний і хімічний захист включає заходи щодо виявлення і оцінки радіаційної та хімічної обстановки, організацію і здійснення дозиметричного та хімічного контролю, розроблення типових режимів радіаційного захисту, забезпечення засобами індивідуального захисту, організацію і проведення спеціальної обробки.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об’єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення у позаміській зоні.

Безпека в надзвичайних ситуаціях є надзвичайно важливим аспектом для забезпечення захисту населення, збереження життєвих ресурсів і навколишнього середовища. Важливими складовими забезпечення безпеки є профілактика,

ефективне реагування та відновлення після події. Для цього необхідно застосовувати комплексний підхід, враховуючи специфіку кожної конкретної надзвичайної ситуації, що забезпечить мінімізацію її наслідків. Залучення органів влади, підприємств, громадських організацій та населення до системи безпеки є ключем до збереження життів і зниження матеріальних втрат в умовах НС.

ВИСНОВКИ

1. Вивчення наслідків та впливу різних проблем в управлінні проектами дозволяє своєчасно виявляти та мінімізувати потенційні загрози, що можуть негативно вплинути на виконання завдань. Огляд подібних рішень дозволяє виявити існуючі проблеми та знайти шляхи їх вирішення шляхом інтеграції нових технологій і підходів.

2. Дослідження методів планування ризиків підтверджує, що ефективне управління ризиками здатне значно покращити результати проекту, зменшуючи ймовірність виникнення серйозних проблем у процесі виконання робіт. У результаті, належне планування ризиків сприяє досягненню більшої стабільності та передбачуваності у роботі з IT-проектами.

3. Розподіл проекту на окремі етапи: планування структури робіт, розробка фронтенду та бекенду, проектування бази даних дозволяє поступово та злагоджено реалізувати всі компоненти системи. Це підвищує ефективність роботи команди та забезпечує належний контроль на кожному етапі.

4. Створення моделі системи дозволяє точно визначити архітектуру, інтерфейси та взаємодію між різними компонентами. Проектування і реалізація бази даних є ключовими етапами для забезпечення зберігання та обробки інформації з високою продуктивністю, що необхідно для підтримки великого обсягу даних у системі.

5. Реалізація аналітичної підсистеми в контексті веб-середовища виявила важливість інтеграції бекенду та фронтенду. Налаштування та реалізація бекенду дозволяє забезпечити взаємодію з базою даних, обробку запитів користувачів та ефективну передачу даних до фронтенду.

6. Деплой додатку став завершальним етапом, який забезпечив можливість тестування та введення інформаційної системи в експлуатацію. Після деплою додаток стає доступним для користувачів, що дозволяє тестувати його функціональність в реальних умовах і виявляти можливі недоліки для подальшої оптимізації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Angular – фреймворк для фронтенду розроблений компанією Google. URL: https://angular.io/.

2. CQRS – архітектурний шаблон проектування. URL: https://docs.microsoft.com/ua- ua/azure/architecture/patterns/cqrs.

3. Clarizen – повнофункціональне програмне забезпечення для управління проектами. URL: https://www.clarizen.com/.

4. Celoxis – веб-інструмент із всеосяжними функціями управління проектами та портфелем. URL: https://celoxis.com.

5. DHTMLX – бібліотека для створення діаграм Ганта. URL: https://dhtmlx.com/.

6. Docker – система віртуалізації на основі механізмів Linux. URL: https://www.docker.com/.

7. EventSourcing – архітектурний шаблон. URL: https://martinfowler.com/eaaDev/EventSourcing.html.

8. GoJS – бібліотека для створення інтерактивних діаграм. URL: https://gojs.net/latest/index.html.

9. Google Cloud «хмарна» платформадля підняття інфраструктури. URL: https://cloud.google.com/.

10. Github – сервіс для надання хостингу коду. URL: ttps://github.com/.

11. Javascript – мова програмування бекенду, фронтенду. URL: learn.javascript.ua/intro/.

12. Laravel – PHP фреймворк, який використовується для написання бекенду. URL: https://laravel.com.

13. MySQL – система управління базами даних. URL: https://www.mysql.com/.

14. monday.com – інструмент планування з дошками kanban, трекер проектів. URL: https:// monday.com

15. PHP – мова програмування, яка дозволяє за допомогою скриптів створювати бекенд. URL: http://php.net/.

16. Ravetree – нагородами рішення для управління проектами з великою кількістю вбудованих функцій спритного управління проектами. URL: https://www.ravetree.com.

17. TypeScript – мова програмування для розширення можливостей JavaScript. URL: https://www.typescriptlang.org/.

18. Wrike – програмне забезпечення для співпраці та управління проектами на основі хмар, яке легко масштабувати. URL: https://www.wrike.com.

19. 10,000ft – програмне забезпечення для управління проектами. URL: https://www.10000ft.com.

20. Роберт Седжвик. Алгоритми на C++. Boston/San Francisco/New York/Toronto/Montreal/London/Munich/Paris/Madrid/Capetown/Tokyo/Singapo re/Mexico City. Addison-Wesley, 2014. 634-645.

ДОДАТКИ

ДОДАТОК А

Фрагмент коду бекенд

Файл StatisticsRepository.php призначений для пошуку статистичних даних зарізними критеріями:

```
<?php
/**
*       Created by PhpStorm.
*       User: jashka
* Date: 08.11.19
* Time: 22:35
*/
namespace Repository;
use
App\Model\ExaminationInfo
;use App\Model\Schema;
use
App\Model\SchemaResult
;use
App\Model\TestResult;
use
Illuminate\Database\Eloquent\Builder
;use
Illuminate\Database\Eloquent\Model;
use Services\UUID;
/**
*       Class ExaminationRepository
*       @package Repository
*/
class StatisticsRepository
{
```

ДОДАТОК Б

Побудова календарного графіку виконання IT-проекту

Графік Ганта потрібен для створення діаграми, яка демонструю календарний графік робіт. Кожна робота в проекті оцінюється для визначення того скільки часу потрібно виділити для того, щоб реалізувати цю роботу. За допомогою діаграми менеджер проекту має змогу бачити дати закінчення та початку робіт, також є можливість створювати задачі, які є блокуючи, а які є паралельними.

Початок роботи проекту було визначено в день видачі завдання на дипломну роботу. Точка відліку видачі теми дипломної роботи дозволили Починаючи з цієї точки було визначено тривалість виконання усіх робіт відповідно діаграмі WBS, дати їх початку та завершення. Також за діаграмою Ганта можна побачити дату завершення роботи над проектом.

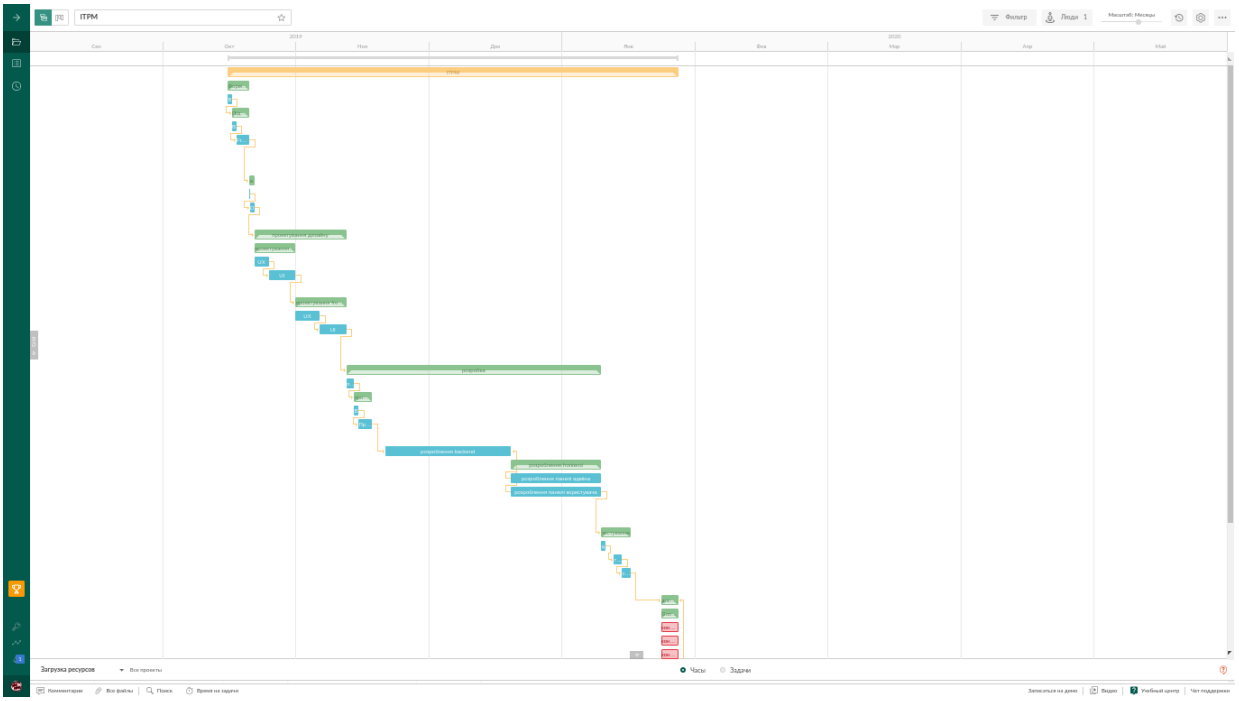

Рисунок Б.1 – діаграма Ганта

Планування ризиків проекту.

Для планування ризиків проекту потрібно звернути увагу на те, які ризики можуть виникнути під час розробки проекту та під час експлуатації. Планування

ризиків проекту потрібно проводити, щоб надати потенційним розробникам певної компанії інформацію про те, які ризики можуть виникнути. Це необхідно для того, щоб команда клієнта мала змогу вчасно реагувати на ризики та знати до яких ситуацій необхідно детальніше підготуватися, щоб не зазнати фінансових збитків компанії.

Були знайдені такі ризики у даному проекті:

— R1 – Оновлення кодової бази для нової версії Angular 8;

Оновлення кодої бази необхідне для адаптації коду до нової версії фреймворку.

— R2 – Неправильно налаштований Docker образ;

Неправильно зібраний Docker [14] образ може статися із-за людської неуваги. Наприклад, збірка Docker образу без правильної конфігурації та змінного середовища може призвести до нездатності додатку запускатися або може використовувати інші дані для підключення до бази даних та призвести до втрати даних якщо виконуємо команду очищення бази з конфігурацією до продакшин бази.

— R3 – Self-Hostedрішення для деплою додатку;

Ризик Self-Hostedрішення для деплою додатку виникає коли немає коштів для розгортання додатку в хмарних провайдерах типа AmazonWebServices, GoogleCloudPlatform, MicrosoftAzure, VMWare, IBMCloud, RedHatта інше.

— R4 – Ваше підключення не захищено;

Цей ризик виникає коли SSL сертифікат був оновлених запізно до того як закінчивсяйого строк здатності працювати.

— R5 – Неправильні стандартні стилі для діаграми Gantt;

— R6 – Оновлення загальної бібліотеки;

— R7 –Оновлення системи на новіші версії фреймворків;

Оновлення фреймворків необхідне для забезпечення стабільної роботи додатку за рахунок виходу нових оновлень, виявлених та вирішених питать з безпеки додатку, багів та інше.

За допомогою експертної оцінки було визначено основні ризики проекту, які можуть спричинити блокування розробки проекту та виникнення проблем під час експлуатації проекту.

Таблиця Б.2 – Ймовірність виникнення ризиків

Ймовірність виникнення	Ймовірність	R1	R2	R3	R4	R5	R6	R7
>30%	ймовірні	+						+
>10%	можливі				+			
<5%	мало ймовірні							

Наступним кроком була побудована таблиця втрат при виникненні ризиків на проекті.

Таблиця Б.3 – Втрати при виникненні ризиків

Час на вирішення	R1	R2	R3	R4	R5	R6	R7
> 1 дня							+
Пару годин			+	+		+	
Кілька хвилин							

За допомогою визначених втрат при виникненні ризиків та ймовірності виникнення ризиків було зпрознозовано і створено матрицю ймірівності-втрати. На даній матриці кольоровий окрас таблиці означає наступне: чим чорніший колір напроти ризику – то це критичний ризик для проекту та важкі затрати та усунення проблеми. Якщо ризик позначено більш світлішим кольором – то важливість ризику зменшується та час на усунення проблеми.

Таблиця Б.4 – Матриця ймовірність-втрати

Ймовірні			R1, R7
Можливі	R4	R4	
Мало ймовірні		R3	
	Мін.	Сер.	Макс.

В результаті проведеної роботи для виявлення ризиків проекту було критичні ризики – оновлення кодової бази проекту (R1) та оновлення версій фреймворків (R7). З цими ризиками може справитися лиші тільки програміст оскільки це потребує знання кодової бази та інструментів, які в ній використовуються.