МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Аналітична підсистема веб-орієнтованого середовища з вивчення дисципліни «Управління ІТ проектами»»

за спеціальністю122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.мз-81с Яковенко Андрій Володимирович

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ « »грудня 2019 р.

Науковий керівник _____ к.т.н., доц.,Гайдабрус Б. В.
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2019

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«__» _____ 2019р.

ЗАВДАННЯ

На кваліфікаційну роботу магістра студентові

Яковенко Андрій Володимирович
(прізвище,ім'я, по батькові)

1 Тема проекту *Аналітична підсистема веб-орієнтованого середовища з вивчення дисципліни «Управління ІТ проектами»*

затверджена наказом по університету від «19» листопада 2019 р. №2296-ІІІ

2Термін здачі студентом закінченого проекту «_10__» ___грудня___ 2019р.

3 Вхідні дані до проекту Веб-орієнтоване середовище з вивчення дисципліни «Управління ІТ проектами».

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) Дослідження актуальності проблеми. Аналіз аналогів та визначення наявних проблем. Аналіз предметної області, Постановка задачі та методи дослідження, Моделювання аналітичної підсистеми, Реалізація аналітичної підсистеми, Висновки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Дослідження актуальності проблеми. Аналіз аналогів та визначення наявних проблем. Аналіз предметної області, Постановка задачі та методи дослідження, Моделювання аналітичної підсистеми, Реалізація аналітичної підсистеми, Висновки. Презентація.

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускного проекту	Термін виконання етапів проекту	Примітка
1	Аналіз предметної області	07.09 – 10.09	
2	Вибір інструментів реалізації	11.09 – 15.09	
3	Дизайн аналітичної підсистеми	16.09 – 25.09	
3	Модернізація бази даних	27.09 – 05.10	
4	Модернізація бекенду	07.10 – 17.10	
5	Розробка загального модулю бібліотеки для фронтенду	19.10 – 30.10	
6	Інтеграція в панель користувача	03.10 – 14.10	
7	Інтеграція в панель адміністратору	15.10 – 22.10	
8	Деплой	25.10 – 10.11	

Магістрант _____ Яковенко А. В.

Керівник роботи _____ к.т.н., доц. Гайдабрус Б.В.

РЕФЕРАТ

До дипломного проекту

Яковенко Андрія Володимировича на тему «Аналітична підсистема веб-орієнтованого середовища з вивчення дисципліни «Управління ІТ проектами»»

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 20 найменувань, додатків. Загальний обсяг роботи − 78 сторінок, у тому числі 52 сторінок основного тексту, 2 сторінки списку використаних джерел, 14 сторінок додатків.

Кваліфікаційну роботу магістра присвячено розробці аналітичної підсистеми веб-орієнтованого середовища з вивчення дисципліни «Управління ІТ проектами». В роботі проведено аналіз предметної області, аналіз аналогічних рішень, проектування аналітичної підсистеми та модернізацію існуючої системи. У роботі виконано розробку загального модулю аналітичної підсистеми та модернізацію бекенду. Результатом проведеної роботи є веб-орієнтоване середовище з вивчення дисципліни «Управління ІТ проектами». Практичне значення роботи полягає у вивченню дисципліни «Управління ІТ проектами» студентами та викладачами для розуміння того як правильно організовувати менеджмент в проектах.

Ключові слова: діаграма WBS, PERT, Gantt Chart, IT проекти, управління IT проектами, менеджмент.

ЗМІСТ

ВСТУП

Управління проектами - це низка завдань, які виконуються для створення визначеного продукту, послуги або результату, як правило, у визначені строки. Управління проектами включає співпрацю з роботою та управління завданнями. Зазвичай проект має менеджера проекту та команду проекту. Керівник проекту може керувати декількома проектами, якщо він має достатню для цього можливостей.

Мета управління проектами - створити та підтримувати середовище, де команда проекту може працювати разом для досягнення найкращого набору результатів в рамках обмежень проекту.

Управління проектами використовується в найрізноманітніших галузях починаючи з IT сфери, будівництво, військова діяльність, бізнес та інше.

ITіндустрія має унікальну готовність відточувати методології управління проектами. Оскільки IT проекти є ідеальними кандидатами для впровадження стратегій управління, наприлад, канбан та налагодження способу їх роботи. Уявіть що управління IT проектами допоможе збільшити продуктивність на 155% за дев'ять коротких місяців шляхом переосмислення черги, планування, організації робочого процесу.

Завдання керівника проекту полягають в наступному:

— уточнення цілей проекту та призначення завдань та обов'язків, планування та відстеження термінів проекту та детальних етапів;
— перевірка ходу проекту та його дотримання часової шкали, бюджету та вимог;
— управління командою передбачає подолання конфліктів на нижчому рівні, спілкування з членами команди та прив’язання до ключових зацікавлених сторін;

— керівник проекту повинен володіти не тільки технічними навичками, а й низкою м'яких навичок, включаючи управління командою, соціальну компетентність, самоуправління та управління стресом.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ УПРАВЛІННЯ ІТ ПРОЕКТАМИ

1.1 Дослідження актуальностіпроблеми

Управління проектами передбачає планування та організацію ресурсів компанії для переміщення конкретного завдання, події чи обов'язку до виконання. Це може включати разовий проект або поточну діяльність, а керовані ресурси включають персонал, фінанси, технології та інтелектуальну власність.

Управління проектами часто асоціюється із сферами інжинірингу та будівництва, а останнім часом - охорони здоров'я та інформаційних технологій (IT), які, як правило, мають складний набір компонентів, які повинні бути завершені та зібрані у встановленому порядку для створення функціонуючого продукту.

Взагалі процес управління проектом включає такі етапи: планування, ініціація, виконання, моніторинг та закриття.Наприклад, в архітектурі проекту план починається з ідеї, просувається до малюнків і переходить до складання креслення, з тисячами дрібниць, що збираються між кожним кроком. Архітектор - це лише одна людина, яка забезпечує одну частину головоломки. Керівник проекту складає все це разом.

Кожен проект зазвичай має бюджет і часові рамки. Управління проектами підтримує все плавно, вчасно та в бюджеті. Це означає, що коли запланований термін закінчується, керівник проекту може утримувати всіх членів команди, що працюють над проектом, до кінця.

*Приклад управління проектами.*Керівнику проекту покладено завдання провести команду з розробки програмних продуктів. Розробка проекту починається з визначення обсягу проекту. Потім призначають завдання проектній групі, до якої можуть входити розробники, інженери, технічні письменники та фахівці із забезпечення якості. Керівник проекту створює графік та встановлює терміни.

Часто менеджер проекту використовує візуальні подання робочого процесу, такі як діаграми Ганта або графіки PERT, щоб визначити, які завдання слід виконати, якими відділами. Потім встановлюють бюджет, який включає достатні

кошти для утримання проекту в межах бюджету навіть за умови несподіваних непередбачених ситуацій. Керівник проекту також гарантує, що команда має ресурси, необхідні для створення, тестування та розгортання програмного продукту.

Коли велика IT-компанія, наприклад CiscoSystemsInc., набуває менших компаній, ключовою частиною роботи керівника проекту є інтеграція членів групи проектів з різних груп та прищеплення почуття групової мети щодо досягнення кінцевої мети. Керівники проектів можуть мати певні технічні ноу-хау, але також мають важливе завдання - сприймати корпоративне бачення та забезпечувати відчутні результати вчасно та в межах бюджету.

*Елементи управління IT проектами.*Управління операціями та управління проектами передбачають ефективний розподіл ресурсів для досягнення результатів найефективнішим та найефективнішим можливим способом та на найвищому рівні можливої якості, враховуючи ресурси, відведені на досягнення.

Управління операціями також орієнтоване на досягнення результатів.Однак управління операціями відрізняється від управління проектами своїм постійним характером; управління операціями зосереджується на постійній постачанні одних і тих самих продуктів або послуг, і це робить це, використовуючи повторювані процеси та одні і ті ж проектні команди.

Управління проектами та управління продуктами.Організації в 21 столітті все більше сприймали управління продуктами як додаткову дисципліну.Хоча обидві дисципліни зосереджені на управлінні справами бізнесу, їх роль є різною. Як уже було встановлено, керівники проектів керують проектами - тимчасовими починаннями, що зумовили початкові та кінцеві точки. З іншого боку, менеджери продуктів відповідають за весь продукт і володіють його успіхом, а також за його підтримку протягом усього життєвого циклу.

Обов'язки керівника проекту.Керівники бізнесу визнають управління проектами специфічною функцією в організації та наймають осіб, спеціально навчених у цій дисципліні - тобто керівників проектів - для управління потребами організації в проекті.

Управління завданнями.Керівники проектів можуть використовувати різні

методи та підходи для виконання проектів, зазвичай вибираючи найкращий підхід, виходячи з характеру проекту, організаційних потреб та культури, навичок тих, хто працює над проектами та інших факторів.

Управління проектом включає кілька етапів. Хоча термінологія для цих кроків різна, до них часто включають:

— визначення цілей проекту;

— визначення кроків, необхідних для досягнення цих цілей;

— визначення ресурсів, необхідних для виконання цих кроків;

— визначення бюджету та часу, необхідних для кожного з етапів, а також проекту в цілому;

— контроль за фактичним виконанням та виконанням робіт;

— забезпечення готового результату.

У рамках потужного плану управління проектами керівники проектів здійснюють контроль для оцінки ефективності та прогресу відповідно до встановленого графіку, бюджету та цілей, викладених у плані управління проектом. Це часто називають обсягом проекту.

Оскільки проекти часто вимагають проектної групи або колективів працівників, які зазвичай не працюють разом, ефективне управління проектами вимагає міцних комунікативних та переговорних навичок. Керівникам проектів також необхідно тісно співпрацювати з кількома зацікавленими сторонами, які мають інтереси в будь-якому проекті, інша сфера, де важливі міцні комунікативні та переговорні навички.

*Види управління проектами.*Багато типів управління проектами було розроблено для задоволення конкретних потреб певних галузей чи типів проектів.

*Управління проектами «waterfall».*Це схоже на традиційне управління проектами, але включає застереження, що кожне завдання необхідно виконати до початку наступного. Кроки є лінійними і прогрес протікає в одну сторону. Через це в управлінні проектами важлива увага до послідовностей завдань та термінів. Часто розмір команди, яка працює над проектом, буде зростати, коли менші завдання виконуються і починаються більші завдання.

Складний менеджмент проектів.Індустрія комп'ютерного програмного забезпечення однією з перших застосувала цю методологію. Маючи основу, що бере початок у 12 основних принципах AgileManifesto, кероване управління проектами - це ітеративний процес, орієнтований на постійний моніторинг та вдосконалення результатів. По суті, високоякісні результати - результат надання цінності клієнтам, взаємодії в команді та адаптації до поточних ділових обставин.

Швидке управління проектами не дотримується послідовного поетапного підходу. Натомість етапи проекту завершуються паралельно один одному різними членами команди в організації. Цей підхід може знайти та виправити помилки, не потребуючи перезавантаження всієї процедури.

Ощадливе управління проектами.Ця методологія полягає втому, щоб уникнути відходів - і марна трата часу, і ресурсів. Принципи цієї методології були викладені з японської виробничої практики. Основна ідея, що стоїть за ними, - створити більшу цінність для клієнтів з меншими ресурсами.

Існує набагато більше методологій та видів управління проектами, ніж перераховано тут, але це деякі найпоширеніші. Тип, що використовується, залежить від уподобань керівника проекту чи компанії, проектом якої керується.

1.2 Аналіз аналогів та визначення наявних проблем

Дослідження, яке було виконано протягом виконання роботи показало, що прямих безпосередніх аналогів представленої тематики немає, оскільки основна частина диплому є унікальною та прив'язаною сугубо до дисципліни «Управління ІТ проектами» у Сумському державному університеті. Є дуже багато різних сервісів, які дозволяють практикуватися та отримувати нові знання з дисципліни «Управління ІТ проектами» але ці всі сервіси потрібно буде використовувати окремо.

Розглянемо шість програмних засобів для управління проектами.

Clarizen [1] - 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.

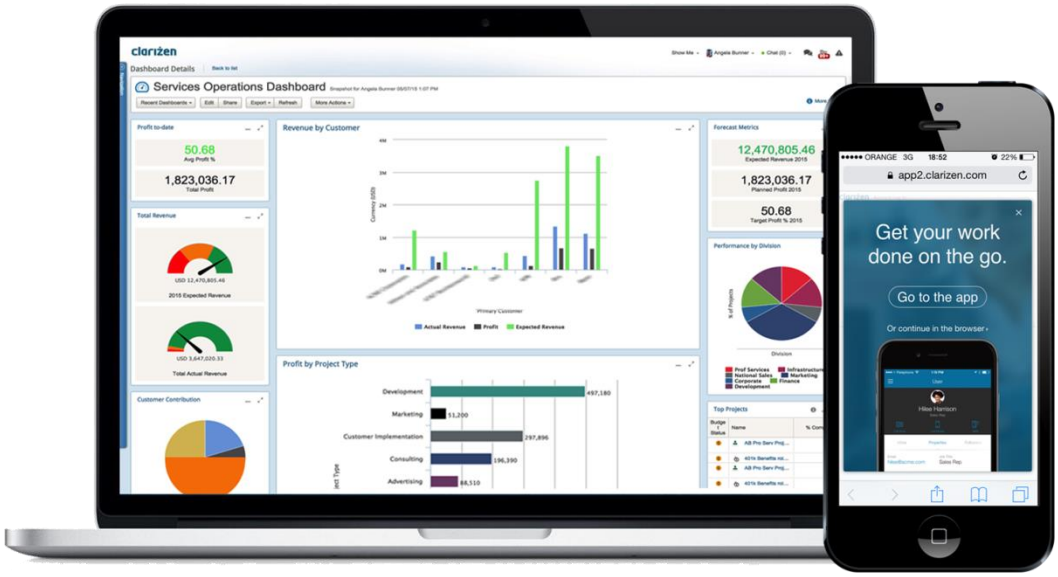

Рисунок 1.1 - Clarizen

monday.com [2] - Інтуїтивно зрозумілий інструмент планування з дошками kanban, трекер проектів, функції співпраці та автоматизація робочих процесів (рис. 1.2).

monday.com [2] - це чудове програмне забезпечення для управління проектами через те, що інструмент усунув безліч деталей типових інструментів управління та орієнтованих на прості, візуально інтуїтивні макети, які допомагають уточнити послідовність роботи.

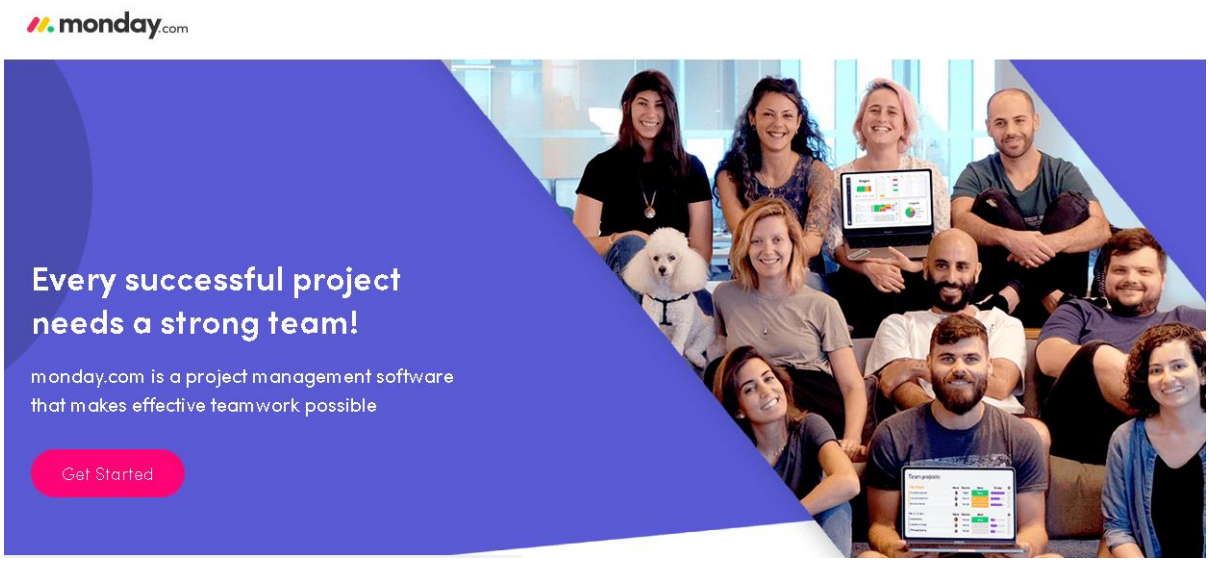

Рисунок 1.2 – monday.com

Celoxis [3] - Популярний, широко використовуваний веб-інструмент із всеосяжними функціями управління проектами та портфелем (рис. 1.3).

Celoxis [3] - це всеосяжна і веб-платформа «все в одному» для управління портфелем проектів та спільної роботи. Celoxis [3] - це одне з найбільш прийнятих у світі інструментів PM, це програмне забезпечення вибору для таких брендів, як HBO, Rolex, Virgin Care, KPMG, Singapore Post, Del Monte, LG та Deloitte для впорядкування своїх проектів, процесів та людей.

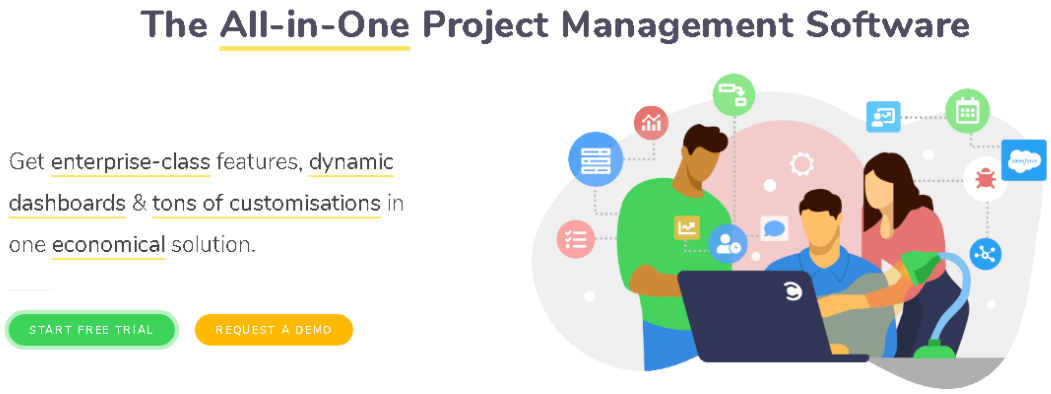

Рисунок 1.3 - Celoxis

10,000ft [4] - надійний, доступний за ціною програмне забезпечення для

управління проектами, яке легко вивчити - і масштабувати на рівні підприємства (рис. 1.4).

Програмне забезпечення для управління проектами та ресурсами 10,000ft [4] дозволяє організаціям легко бачити та діяти на найважливіших даних для проектів та 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 1.4 - 10,000ft

Ravetree [5] - нагородами рішення для управління проектами з великою кількістю вбудованих функцій спритного управління проектами (рис. 1.5).

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.

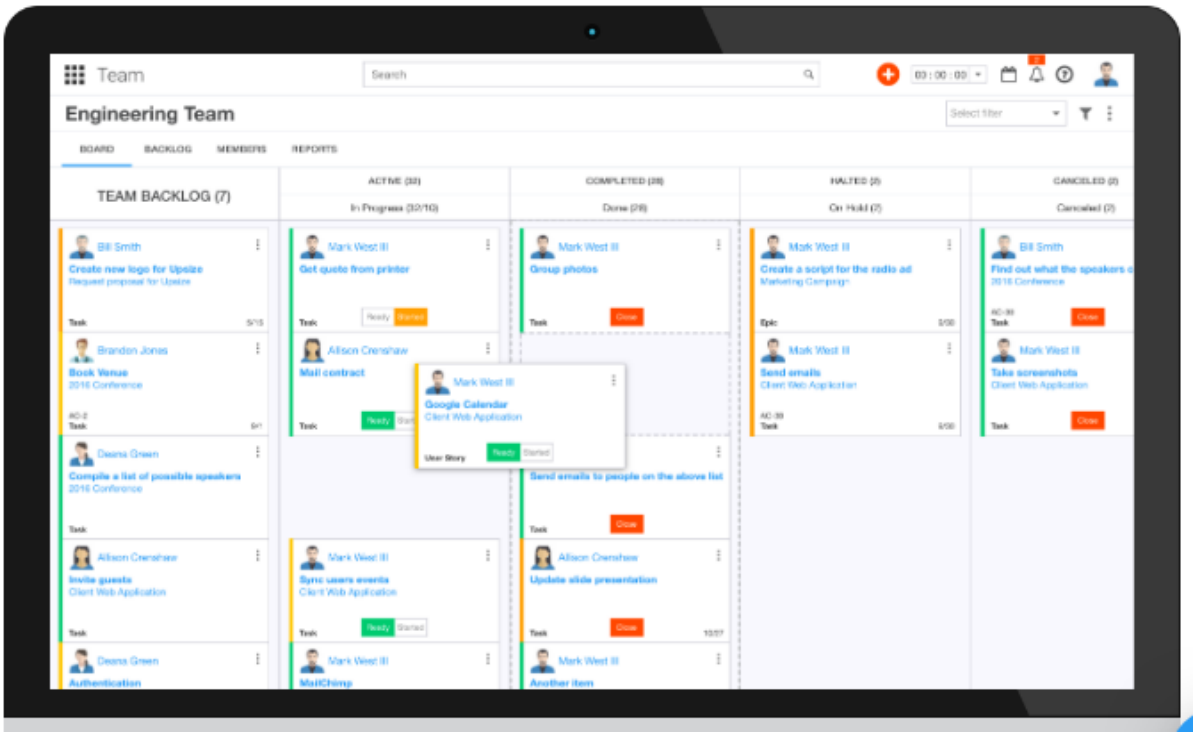

Рисунок 1.5 - Ravetree

Wrike [6] - програмне забезпечення для співпраці та управління проектами на основі хмар, яке легко масштабувати (рис 1.6).

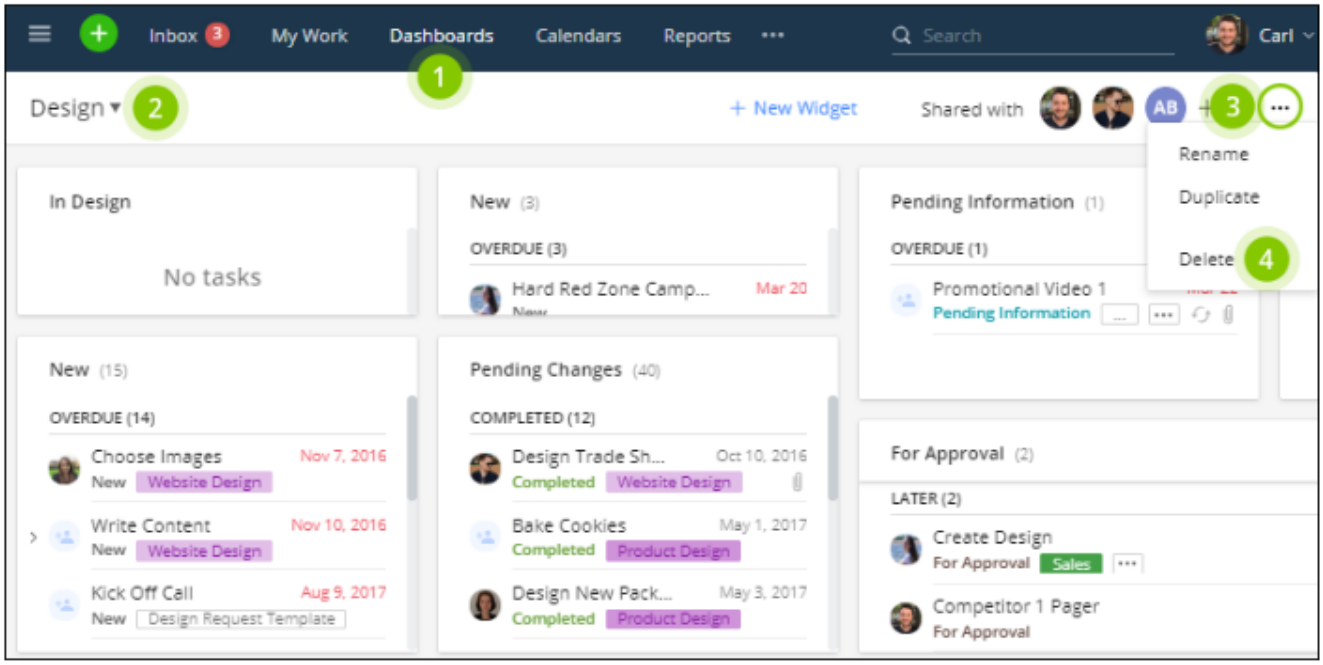

Рисунок 1.6 – Wrike

Wrike [6] - це інструмент спільної роботи в режимі реального часу для підприємств. Функціонування інструментарію «Wrike» в рамках планування проектів, співпраці, звітності, автоматизації робочих процесів та інтеграції сторонніх організацій. Wrike [6] - це хороший інструмент, якщо ви хочете створити ефективність управління робочим потоком.

Оскільки тема дипломного проекту це аналітична підсистема веб-орієнтованого середовища з вивчення дисципліни «Управління IT проектами», то для аналізу діаграм потрібні дані. Звичайно аналіз діаграми людина робити вручну і для цього потрібно дуже багато часу оскільки діаграми бувають дуже великі.

Для аналізу діаграми WBS потрібні якісь встановлені стандартом метрики, щоб можно було орієнтуватися і розуміти те, що на діаграмі зображено за допомогою програмного коду. Для цього при створенні діаграми, на кожному блоці WBSбуде можливість встановити «тег» (метрику) для того щоб можно було «читати» діаграму в коді і проаналізувати діаграму.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Дипломний проект буде у вигляді реалізації додаткових функціональностей існуючого веб-орієнтованого середовища з додаванням аспектів аналітики. Для цього потрібно буде допрацювати серверну частину, яка відповідає за статистику, клієнтська частина яка буде показувати статистику по проходженню тесту та база даних для збереження статистичних даних. Метою створення аналітичної підсистеми для веб-орієнтованого середовища з дисципліни IT Project Management тому що аналогів в відкритому доступі немає але є можливість, що компанії створювали такі продукти але без публікації.

Задачі на дипломний проект :

— вибір технологій для деплою додатку;

— проектування архітектури аналітичної частини проекту;

— проектування бази даних для збереження статистичних даних;

— реалізація частини фронтенду для статистики по проходженню тестів;

— реалізація частини фронтенду для статистики по перевірці WBSдіаграми;

— реалізація бекенду для статистики;

— реалізація CQRS-архітектури системи;

— реалізація клієнтської частини для підказок побудови діаграм;

— оновлення всієї системи на новіші версії фреймворків.

2.2 Методи дослідження

Даний програмний продукт розроблений як частина веб-орієнтованого середовище, а точніше вбудовано в самий додаток SPA (SinglePageApplication).

Для зручного розроблення використовується система управління версіями Github [18].

Github [18] — платформа для розробки та хостингу вашої кодової бази.

Аналітична підсистема розроблюється за допомогою Angular 8 [7] + TypeScript [9] – фронтенд, бекенд – PHPFrameworkLaravel 5.6 [11].

Angular [7] з «коробки» використовує практики по організації файлів та каталогів, систему ін’єкцій залежностей, систему для збірки Webpack та використовує мову програмування TypeScript [9].

Для управлінняданими використовується RxJS [13]. Реактивне програмування за допомогою RxJSвикористовується для того, щоб мати змогу в реальному часі оновлювати дані в сховищі фронтенду. RxJSвикористовується для виконання HTTPзапитів на бекенд та обробки даних після виконання запиту.

Так як аналітична підсистема буде використовуватися тільки для клієнта, то потрібно винести код для аналізу діаграми в загальну бібліотеку оскільки це буде етап для масштабування проекту і перевикористання функціоналу в інших системах або додатках на Angular [7].

Laravel [11] - безкоштовнийPHP фреймворк звідкритимкодом в якому реалізована архітектура MVC, надає такі можливості як роутинг, валідація вхідних даних, робота з базою даних, створення консольних скриптів, робота з чергами та багато іншого.

Docker [14] - це програмна платформа, яка дозволяє швидко створювати, тестувати та розгортати програми.Docker пакує програмне забезпечення у стандартизовані одиниці, які називаються контейнерами, у яких є все необхідне для роботи програмного забезпечення, включаючи бібліотеки, системні інструменти, код та час виконання. Використовуючи Docker, ви можете швидко розгортати та масштабувати програми у будь-якому середовищі та знати, що ваш код буде працювати. Контейнер - це стандартний блок програмного забезпечення, який пакує коди та всі його залежності, тому програма швидко та надійно працює з одного обчислювального середовища в інше. тобто: Якщо ви уявляєте контейнер всередині корабля, ви можете уявити це правильно? у цього контейнера є стільки речей, що

дорівнює контейнеру в Docker, який є обгорткою інших пакетів і додатків, як, наприклад, показано на цій фотографії. Контейнер також називається екземпляром зображень докера.

В першу чергу потрібно змоделювати архітектуру аналітичної підсистеми веб-орієнтованого середовища так, щоб в майбутньому була можливість розширювати фунціонал створеної системибез необхідності змінювати функціонал існуючої системи.

Для зберігання даних статистичних даних було вибрано базу даних MySQL8 [12], яка підтримує роботу з JSON та надає розробнику можливості для роботи з цим форматом, а саме фільтрацію, вибірку та інше. База даних для аналітичної підсистеми буде окремою оскільки це статистичні дані, то до них запитів буде набагато більше ніж запису.Для цього було вибрано тип архітектури бекендуCQRS [19].

CQRS [19] – розділення інтерфейсів для запису і читання для підвищення продуктивності, масштабованості і безпеки. Також це спрощує процеси пов’язані з розвитком системи, завдяки додатковій гнучкості, і дозволить уникнути конфліктів злиття на рівні домену, що викливаються командами поновлення. CQRS розшифровується як розділення відповідальності за запити команд. Це шаблон, який я вперше почув, описаний Грегом Янг. В основі лежить поняття, що ви можете використовувати іншу модель для оновлення інформації, ніж модель, яку ви використовуєте для читання інформації. У деяких ситуаціях цей поділ може бути цінним, але слід враховувати, що для більшості систем CQRS додає ризикованої складності. Основний підхід, який люди використовують для взаємодії з інформаційною системою - це трактувати її як сховище даних CRUD. Під цим я маю на увазі, що у нас є ментальна модель деякої структури записів, де є можливість створювати нові записи, читати записи, оновлювати існуючі записи та видаляти записи (рис. 2.1).

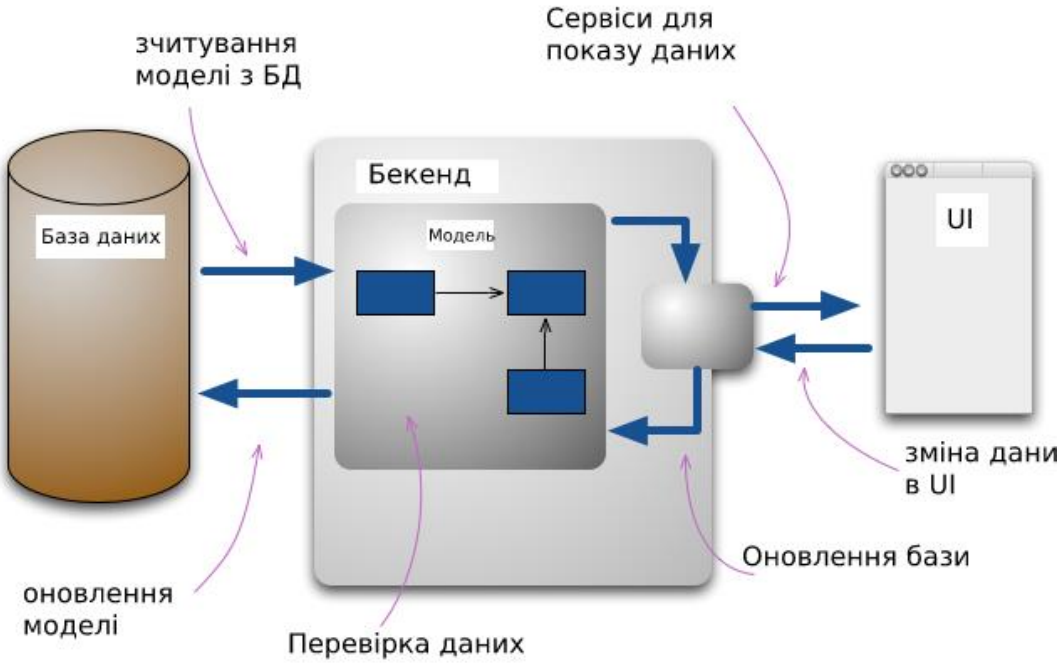

Рисунок 2.1 – класична архітектура

У найпростішому випадку наша взаємодія стосується збереження та отримання цих записів. Оскільки потреби розробника стають все більш досконалими, потрібно наполегливо відходити від цієї моделі. За допомогою цього архітектурного шаблону проектування можно подивитися на інформацію по-іншому до сховища записів, можливо, зібравши кілька записів в один або сформувавши віртуальні записи, об’єднавши інформацію для різних місць. На стороні оновлення є можливість знайти правила перевірки, які дозволяють зберігати лише певні комбінації даних, або навіть можуть зробити висновок про збереження даних, що відрізняються від тих, які надаються користувачем.

Коли користувачі взаємодіють з інформацією, то використовують різні презентації цієї інформації, кожна з яких є різним представленням. Зазвичай розробники будують власну концептуальну модель, яку використовують для маніпулювання основними елементами моделі. Якщо ви використовуєте Доменну модель, то це, як правило, концептуальне подання домену. Зазвичай ви також робите стійке сховище максимально наближеним до концептуальної моделі.

Ця структура декількох шарів представлення може бути досить складною, але коли люди це роблять, то все одно розв'язують її до єдиного концептуального подання, яке виступає як концептуальна точка інтеграції між усіма презентаціями.

Зміна, яку вводить CQRS [19], полягає в тому, щоб розділити цю концептуальну модель на окремі моделі для оновлення та відображення, які вона посилається як Command і Query відповідно відповідно до словника CommandQuerySeparation (рис. 2.2). Обґрунтування полягає в тому, що для багатьох проблем, особливо у складніших областях, наявність однакової концептуальної моделі для команд та запитів призводить до більш складної моделі, яка не справляється з користю.

Під окремими моделями найчастіше розуміємо різні об'єктні моделі, ймовірно, що працюють в різних логічних процесах, можливо, на окремому апаратному забезпеченні. Програмне забезпечення «бачить», що користувач переглядає веб-сторінку, відображену за допомогою моделі запиту. Якщо користувач ініціює зміну, яка перенаправляється в окрему командну модель для обробки, отримана зміна передається моделі запиту для надання оновленого стану.

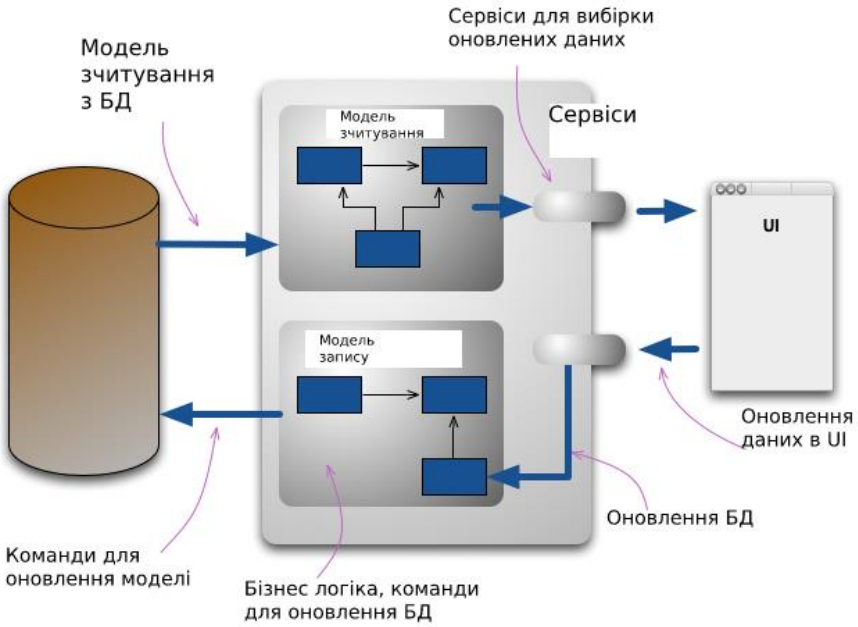

Рисунок 2.2 – CQRS архітектура

Тут є місце для значних варіацій. Моделі в пам'яті можуть мати спільну базу даних, і в цьому випадку база даних виконує функції зв'язку між двома моделями. Однак є можливість використовувати окремі бази даних, ефективно перетворюючи базу даних запиту в базу даних звітів у реальному часі. У цьому випадку повинен бути механізм зв'язку між двома моделями або їх базами даних.

Дві моделі можуть бути не окремими об'єктними моделями, можливо, що одні і ті ж об'єкти мають різні інтерфейси для своєї команди та сторони запиту, а не представлення у реляційних базах даних.

3 ПРОЕКТУВАННЯ АНАТІТИЧНОЇ ПІДСИСТЕМИ ВЕБ-ОРІЄНТОВАНОГО СЕРЕДОВИЩА З ВИВЧЕННЯ ДИСЦИПЛІНИ «УПРАВЛІННЯ ІТ ПРОЕКТАМИ»

3.1 Вимоги до аналітичної підсистеми веб-орієнтованого середовища

Аналітична підсистема веб-орієнтованого середовища з вивчення дисципліни «Управління ІТ проектами» повинна бути доступною, масштабованою та зручною як в плані користувацької частини так і в плані подальшої розробки. В аналітичній підсистемі повинна бути закладена гнучка архітектура, яка дозволила масштабувати операції на запис та зчитування. Інтерфейс аналітичної підсистеми повинен містити в собі елементи управління, які дозволяють фільтрувати статистику користувачів за різними параметрами.

Система повинна відповідати наступним вимогам:

— висока швидкодія;

— мати захищене з’єднання;

— повинен бути перегляд результату діаграми;

— перегляд результату тесту;

— перегляд кількості правильних відповідей діаграми;

— розмежування прав;

— фільтрація статистики за різними критеріями;

— видалення статистики;

— пошук користувачів по імені;

— пошук статистики по темі;

— статистика повинна швидко завантажуватися.

3.2 Структура 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 кодуGithub [18] 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 адміністратору. Збірка модуля статистики завжди залежить від загальної бібліотеки оскільки без неї робота неможлива.

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

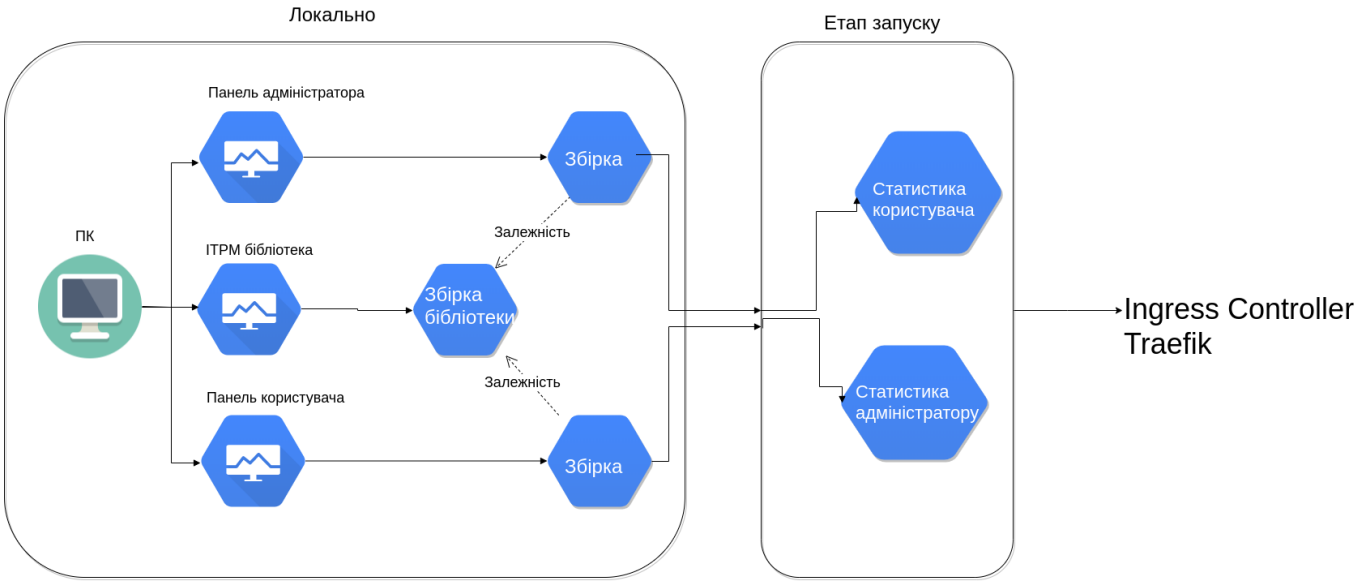

Рисунок 3.1 – Архітектура фронтенду системи

Архітектура бекенду (рис. 3.2) спроектованана 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].

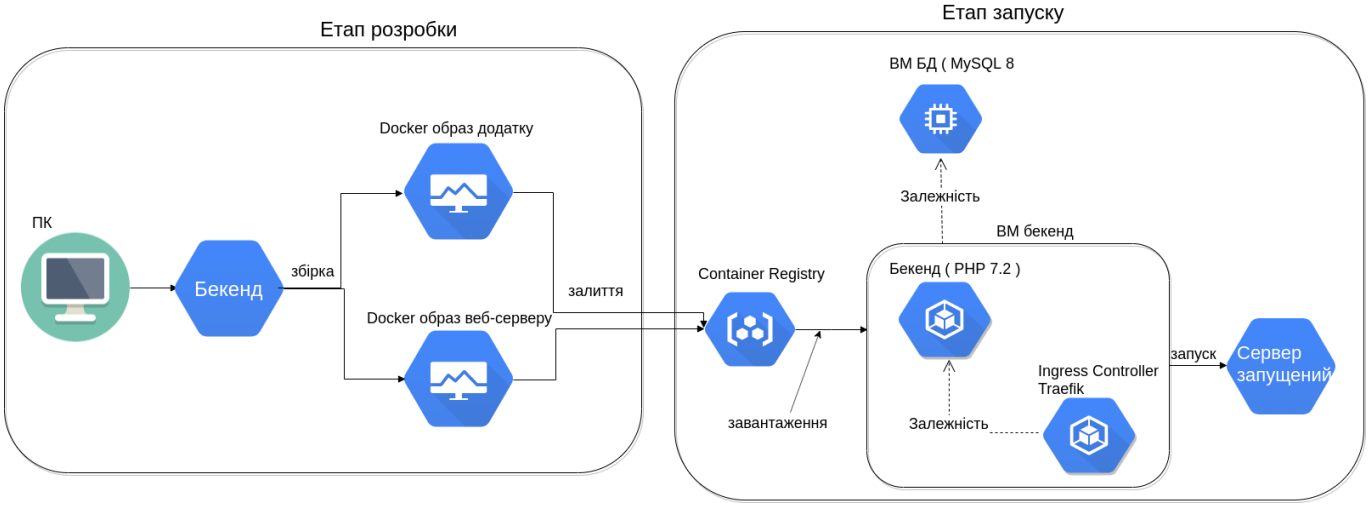

Рисунок 3.2 — Архітектура бекенду

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

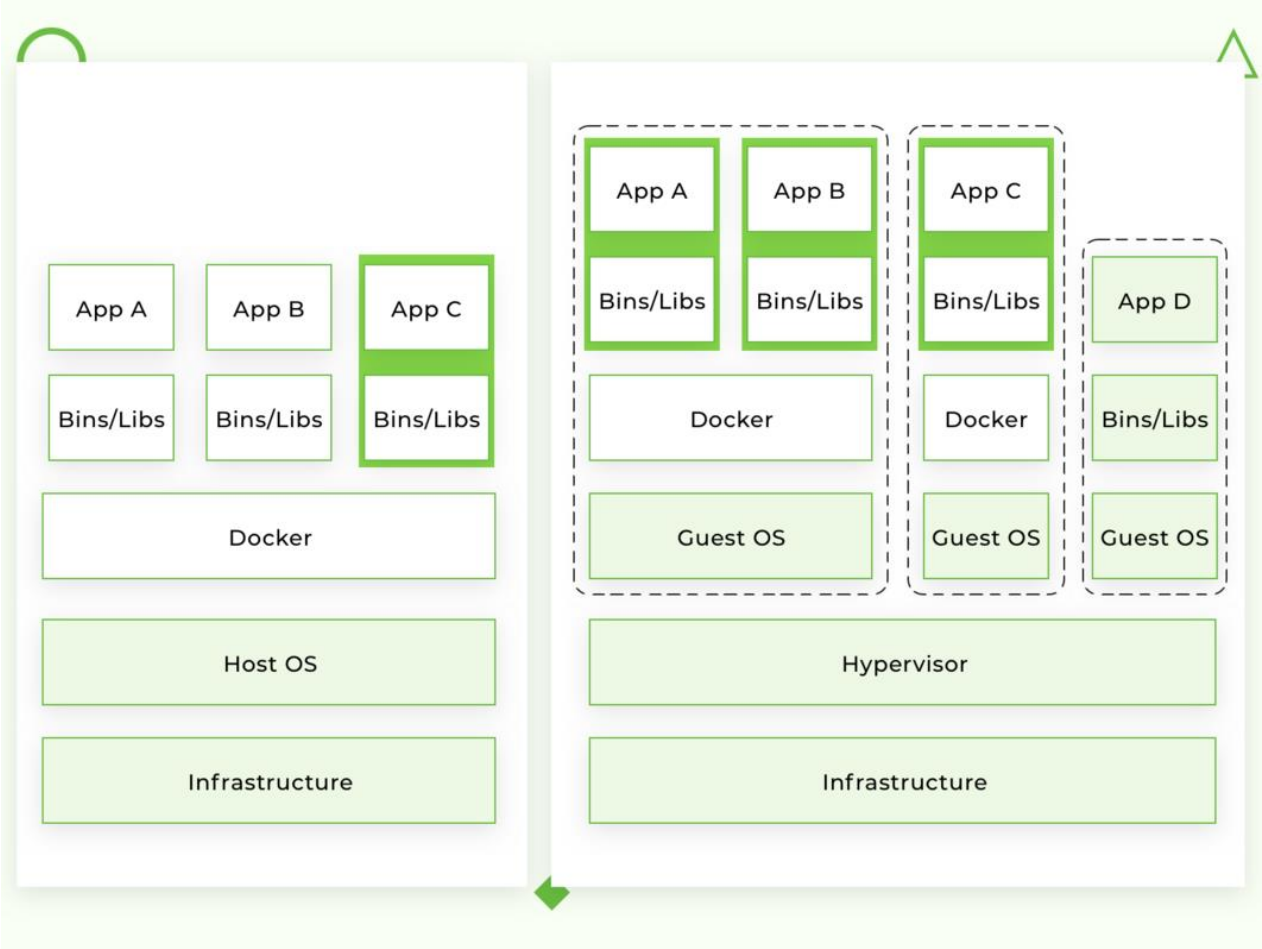

Рисунок 3.3 – Архітектура Docker

3.3 Моделювання інформаційної системи

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

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

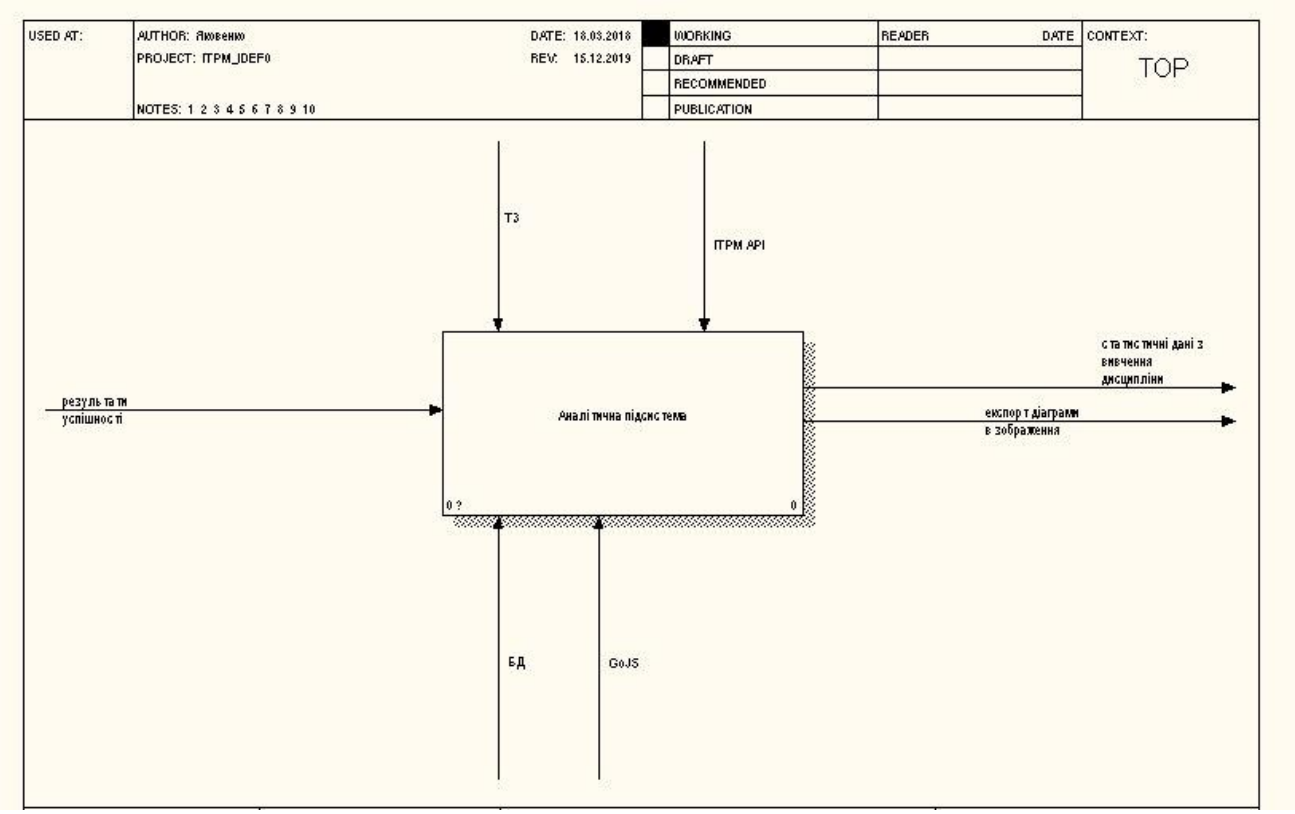

Рисунок 3.4 – Контекстна діаграма аналітичної підсистеми у нотаціїIDEF0

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.

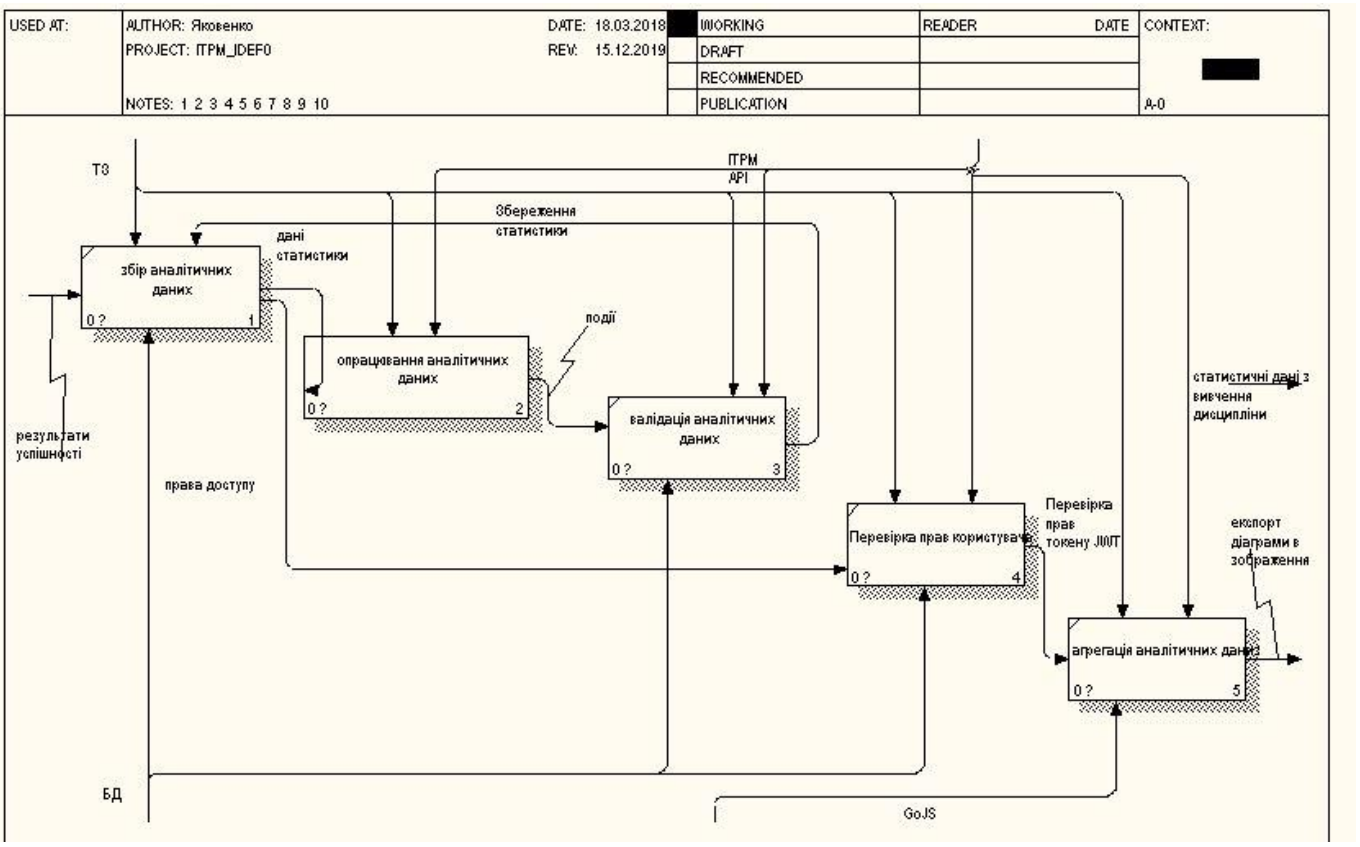

Рисунок 3.5 – Перший рівень декомпозиціїпроцесів аналітичної підсистеми

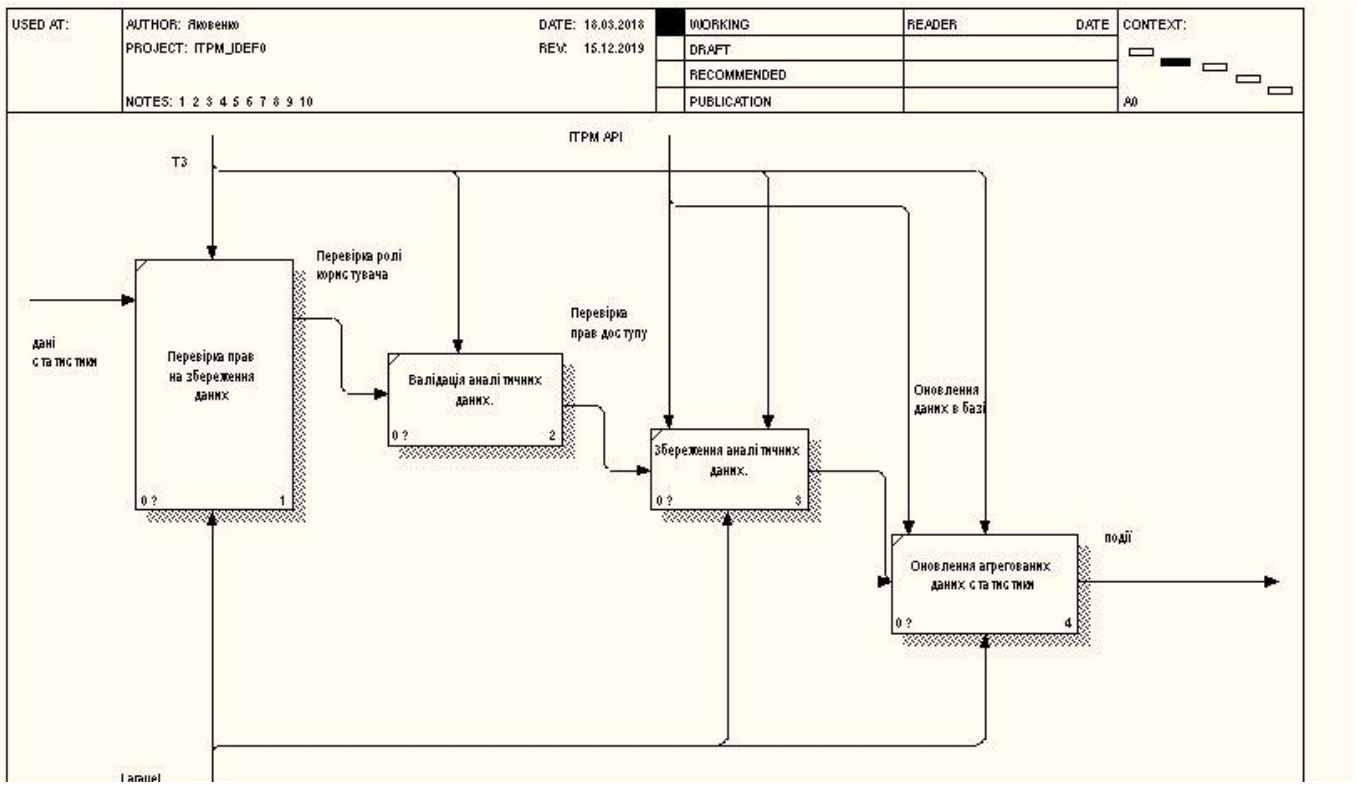

Рисунок 3.6 – Декомпозиція процесу збору аналітичних даних

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

Варіанти 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 аналітики користувача по тесту;
- 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).

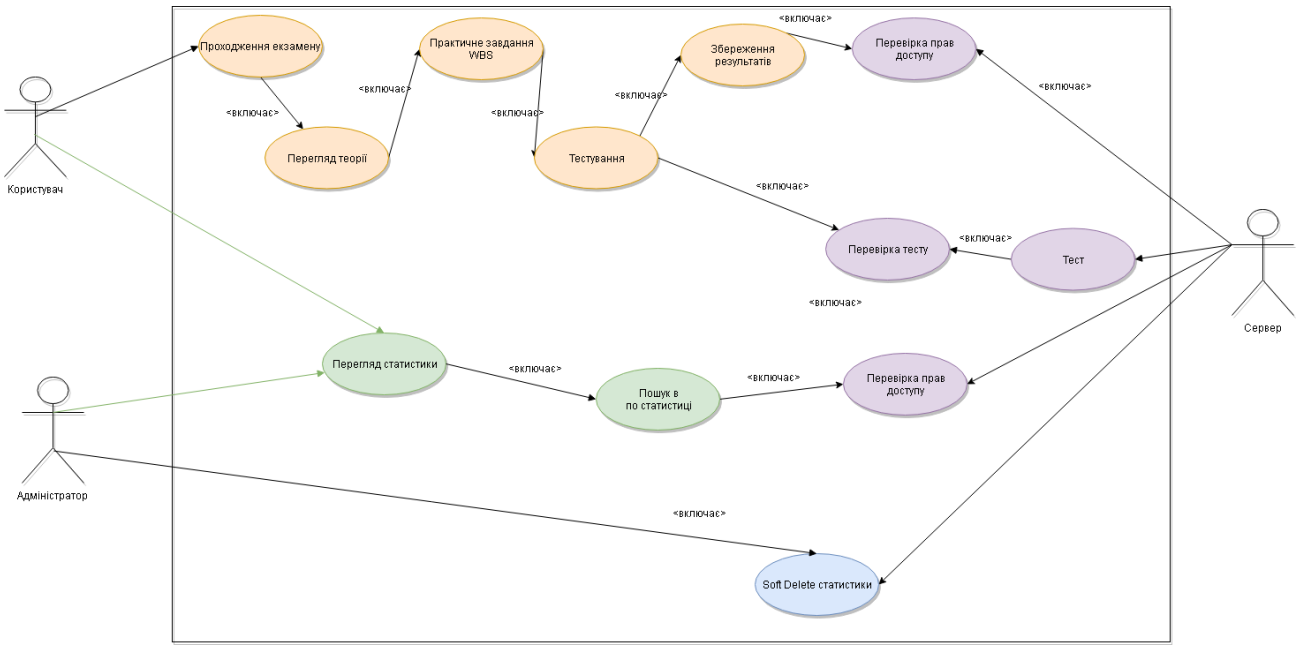

Рисунок 3.7 – Діаграма варіантів використання

3.4 Проектування бази даних

Щоб зберігати дані для аналітики потрібно спроектувати дві бази даних. Перша база буде використовуватися для зберігання статистичних даних, тобто запису, а інша база потрібна для зчитування даних та віддачі користувачу.

В результаті моделювання бази даних для операції запису потрібні наступні таблиці:

— статистичні агрегати.

Для моделювання бази даних операцій зчитування статистичних даних потрібні наступні таблиці:

— результати тесту;

— результати діаграми;

— теорія;

— користувачі;

— таблиця міграцій.

При збереженні результату статистики записуємо дані в базу даних для

«сирих» даних, які потім будуть агреговані для зчитування даних.

Таблиця для статистичних агрегатів містить в собі поля для ідентифікації користувача, тесту, діаграми та зберігає результати тесту і діаграми в jsonbформаті. Таблиці результати теста, діаграми, теорія та користувачі будуть агреговані та зібрані на рівні додатку і записані в окрему базу для зчитування.

Архітектурний шаблон проектування який винує агрегацію даних та операція запису та зчитування називається EventSourcing [21].

EventSourcing[21] – це збереження поточного стану даних в доменній області і використовується як інкрементне сховище для запису повних серій дій з цими даними. Сховище діє як система запису, і його можна використовувати для матеріалізації об'єктів домену. Це дозволяє спростити завдання в складних доменах, усуваючи необхідність синхронізації моделі даних і домену для бізнесу при одночасному підвищенні продуктивності, масштабованості і швидкості реагування. Крім того, забезпечується сумісність транзакційних даних і збереження всіх журналів аудиту та історії, за допомогою яких можна використовувати компенсуючі дії.

Більшість додатків працюють з даними, і типовий підхід, який використовується в додатку, - підтримка поточного стану даних за рахунок оновлення по мірі роботи з ними. Наприклад, у традиційній моделі створення, читання, оновлення та видалення (CRUD) типова обробка даних являє собою зчитування даних зі сховища, внесення в них певних змін і оновлення їх поточного стану за допомогою нових значень (часто за допомогою транзакцій блокування даних).

Шаблон джерела подій визначає підхід до обробки операцій з даними на основі послідовності подій, кожне з яких записується в інкрементіруемое сховище. Код додатка відправляє ряд подій, які примусово описують кожну дію з даними в сховище подій, де події зберігаються. Кожна подія означає деякий набір змін в даних, наприклад, «AddedItemToOrder».

Події зберігаються в сховищі подій, яке виступає в якості системи запису (що заслуговує довіру джерело даних) поточного стану даних. Сховище подій зазвичай

публікує ці події для оповіщення споживачів і при необхідності надання їм можливості обробляти ці події. Споживачі можуть, наприклад, запустити завдання, які застосовують операції в події до інших систем, або виконати будь-яке інше пов'язане дію, необхідне для завершення операції. Зверніть увагу, що код програми, що створює події, ніяк не пов'язаний з системами, підписаними на ці події.

Стандартне використання подій, опублікованих сховищем подій, включає підтримку матеріалізованих уявлень сутностей в міру їх зміни в додатку за допомогою дій, а також інтеграцію з зовнішніми системами. Наприклад, система може підтримувати матеріалізоване уявлення всіх замовлень клієнта, використовуваних для заповнення частин призначеного для користувача інтерфейсу. У міру додавання нових замовлень, додавання або видалення позицій в замовленні або додати деталь про доставку в додатку події, що описують ці зміни, можна обробляти, а також використовувати для поновлення матеріалізованого уявлення.

Крім того, додатки можуть в будь-який момент вважати історію подій і використовувати її для матеріалізації поточного стану сутності за рахунок відтворення і використання всіх подій, які відносяться до цієї сутності. Це можна зробити за запитом для матеріалізації об'єкта домену при обробці запиту або в запланованої задачі для збереження стану сутності у вигляді матеріалізованого уявлення для підтримки рівня уявлення.

На схемі нижче представлений огляд шаблону, включаючи деякі варіанти використання потоку подій, наприклад створення матеріалізованого уявлення, інтеграцію подій із зовнішніми програмами і системами, а також відтворення подій для створення проекцій поточного стану певних сутностей.

Архітектура Event Sourcing на рисунку 3.8.

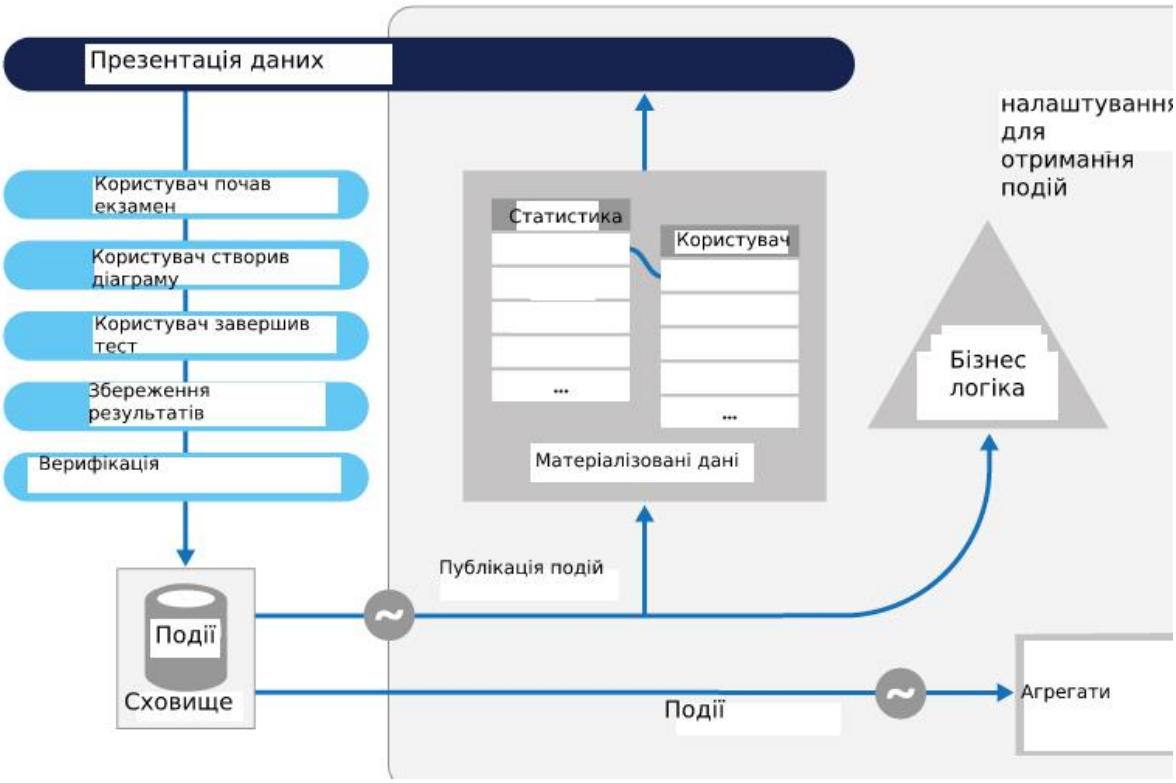

Рисунок 3.8 – Архітектура EventSourcing

Архітектура RESTful взаємодії бекенду та користувача на рисунку 3.9. Взаємодія між клієнтом та сервером виконується за допомогою HTTPзапитів. Клієнт запитує певний ресурс, а сервер відповідає та видає результат по певному ресурсу. Клієнт може оновити, створити, прочитати чи видалити ресурс якщо в нього є права. Бекенд має розмежування маршрутів по префіксу. За префіксом маршруту приховується система розмежування прав.

Бекенд має відповідати клієнту в певному зазначеному стандарті JSON. Бекенд повинен дотримуватися архітектурних обмежень, щоб клієнт мав змогу дотримуватися контракту між бекендом. Клієнт має відправляти запити на бекенд в зазначену форматі, а бекенд має в цьому форматі відправляти дані назад клієнту.

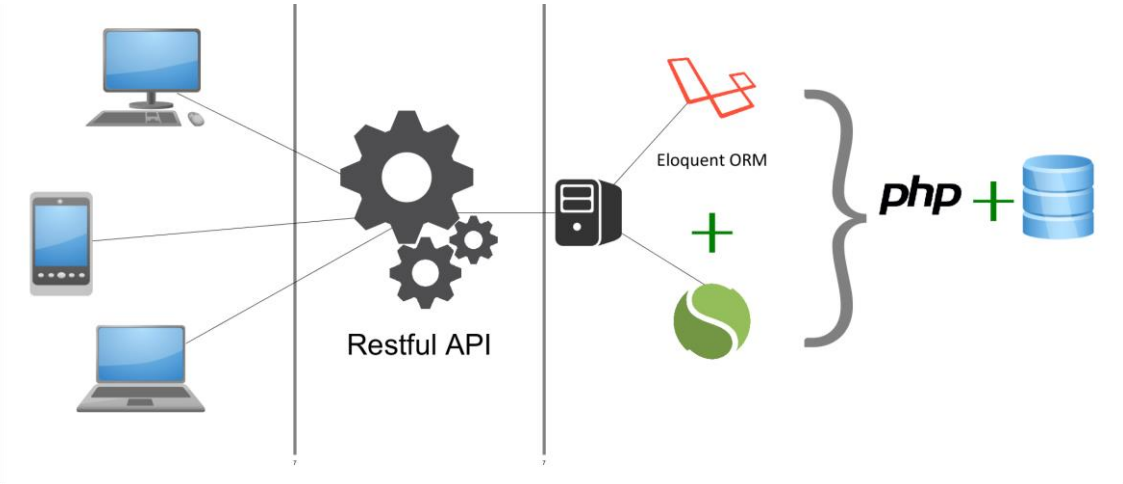

Рисунок 3.9 – архітектура RESTfulбекенду

Проектування структури проекту. Проект має буди розділеним на 4 частини: фронтенд частина для користувача, фронтенд частина для адміністратору, бекенд частина та загальна бібліотека.

Архітектура внутрішньої взаємодії всередині бекенду (рис 3.10).

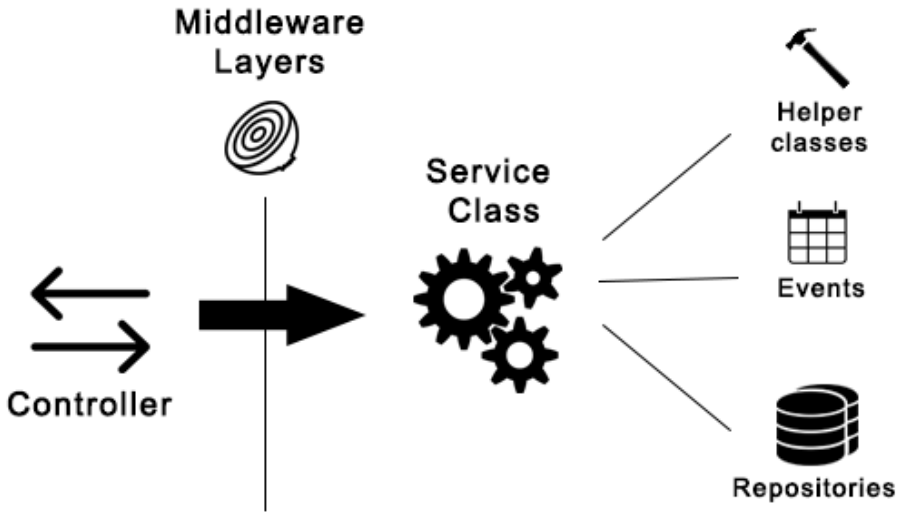

Рисунок 3.10 – Архітектура взаємодії фреймворку

Клієнт відправляє запит по зазначеному маршруту, маршрут викликає Middeware – проміжне програмне забезпечення, яке перевірє доступ користувача до цього маршруту. Наступним кроком викликається контролер, який в свою чергу виклиє сервіси для обробки запиту. Сервіси використовують репозиторії для роботи

з базою даних. Для створення події при записі статистичних даних використовується шина даних, яка публікує події підписникам, а підписними в свою чергу оброблють подію.

Програмний код який виконує проміжні результати і який допомагає обробити запит користувача але не містить в собі бізнес логіки по зміні даних називається помічник «Helper».

Архітектура загальної бази даних (рис. 3.11), яка містить в собі таблиці для статистики, які були агреговані з таблиці агрегатів «сирих» даних результатів екзамену.

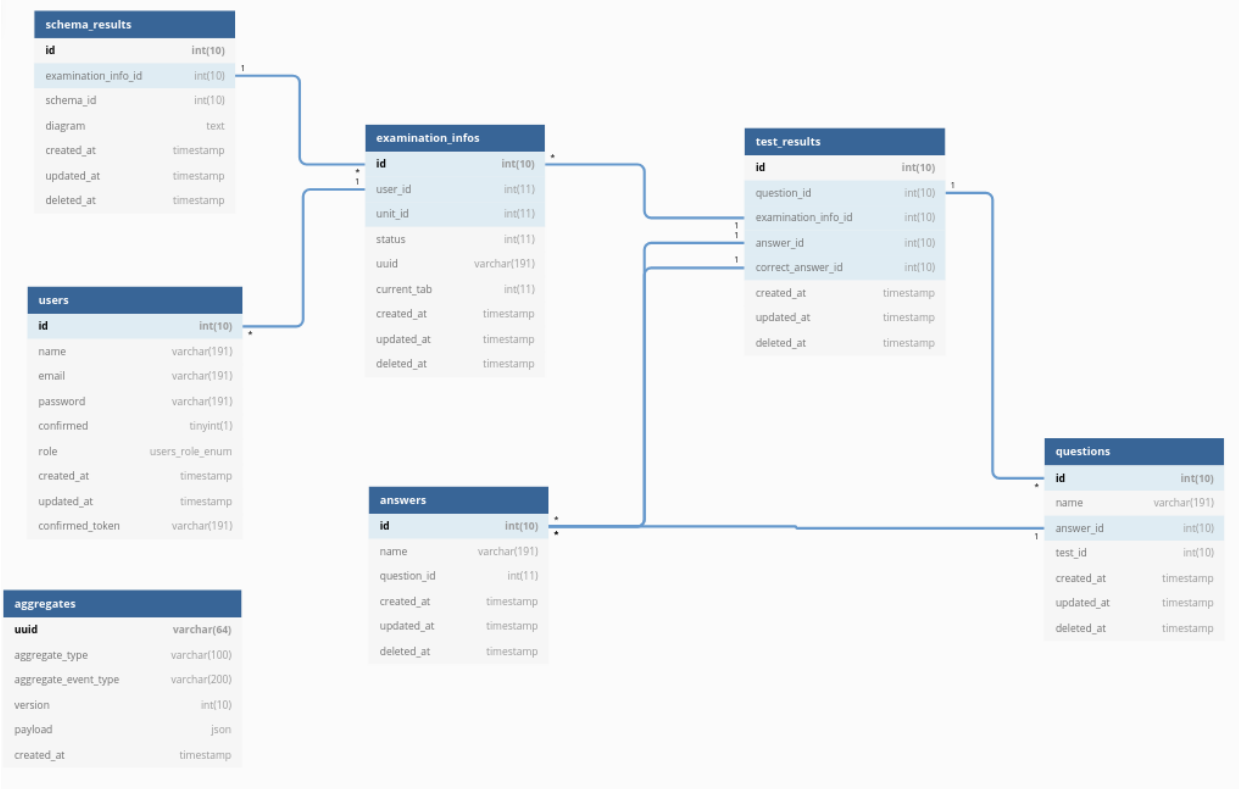

Рисунок 3.11 – Архітектура бази даних

Архітектура бази даних містить в собі таблицю для міграцій, також таблицю користувачів, тем, курсів, лекцій, діаграм, тестів, запитань та відповідей для тесту, теорії, екзаменів і файлів.

4 РЕАЛІЗАЦІЯ АНАЛІТИЧНОЇ ПІДСИСТЕМИ ВЕБ-ОРІЄНТОВАНОГО СЕРЕДОВИЩАЗ ВИВЧЕННЯ ДИСЦИПЛІНИ «УПРАВЛІННЯ ІТ ПРОЕКТАМИ»

4.1 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).

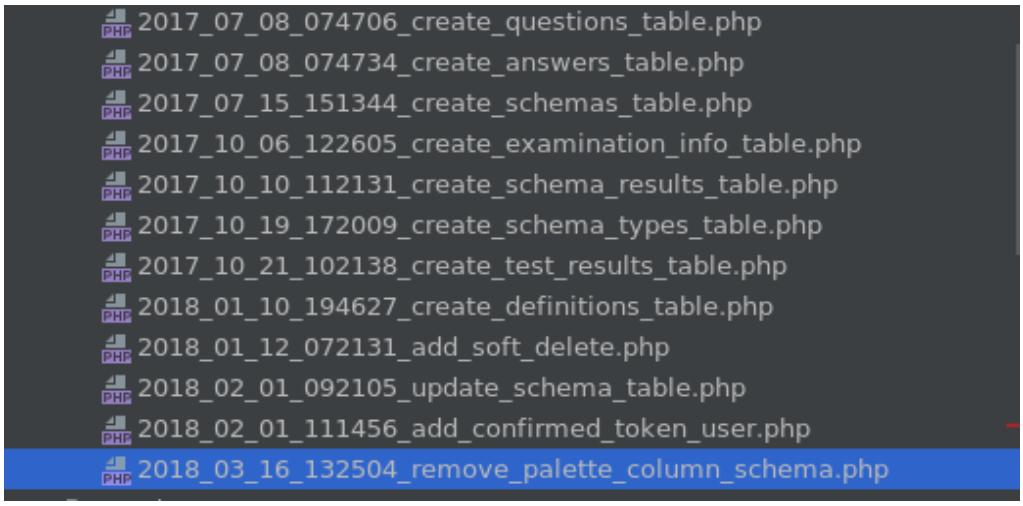

Рисунок 4.1 — Міграції для бази даних

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

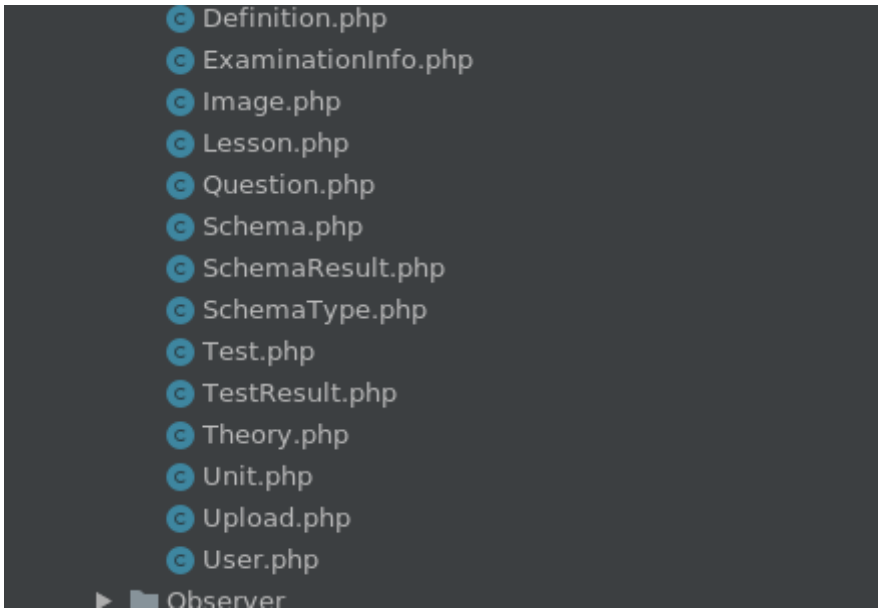

Рисунок 4.2 — Моделі даних

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

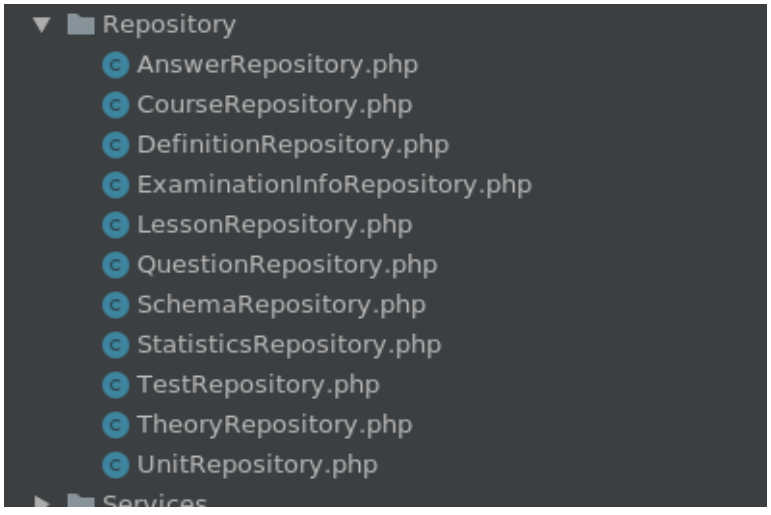

Рисунок 4.3 — Репозиторії моделей

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

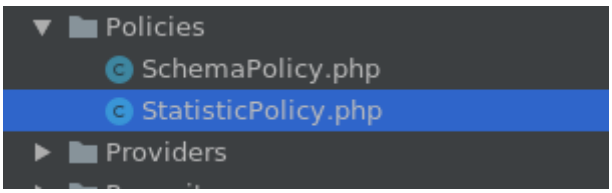

Рисунок 4.4 — Політика розмежування статистики

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).

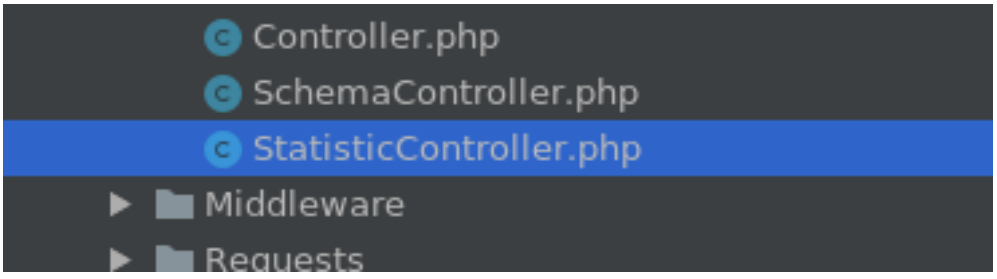

Рисунок 4.5 — Контролер для обробки статистики

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 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 (рис 4.6).

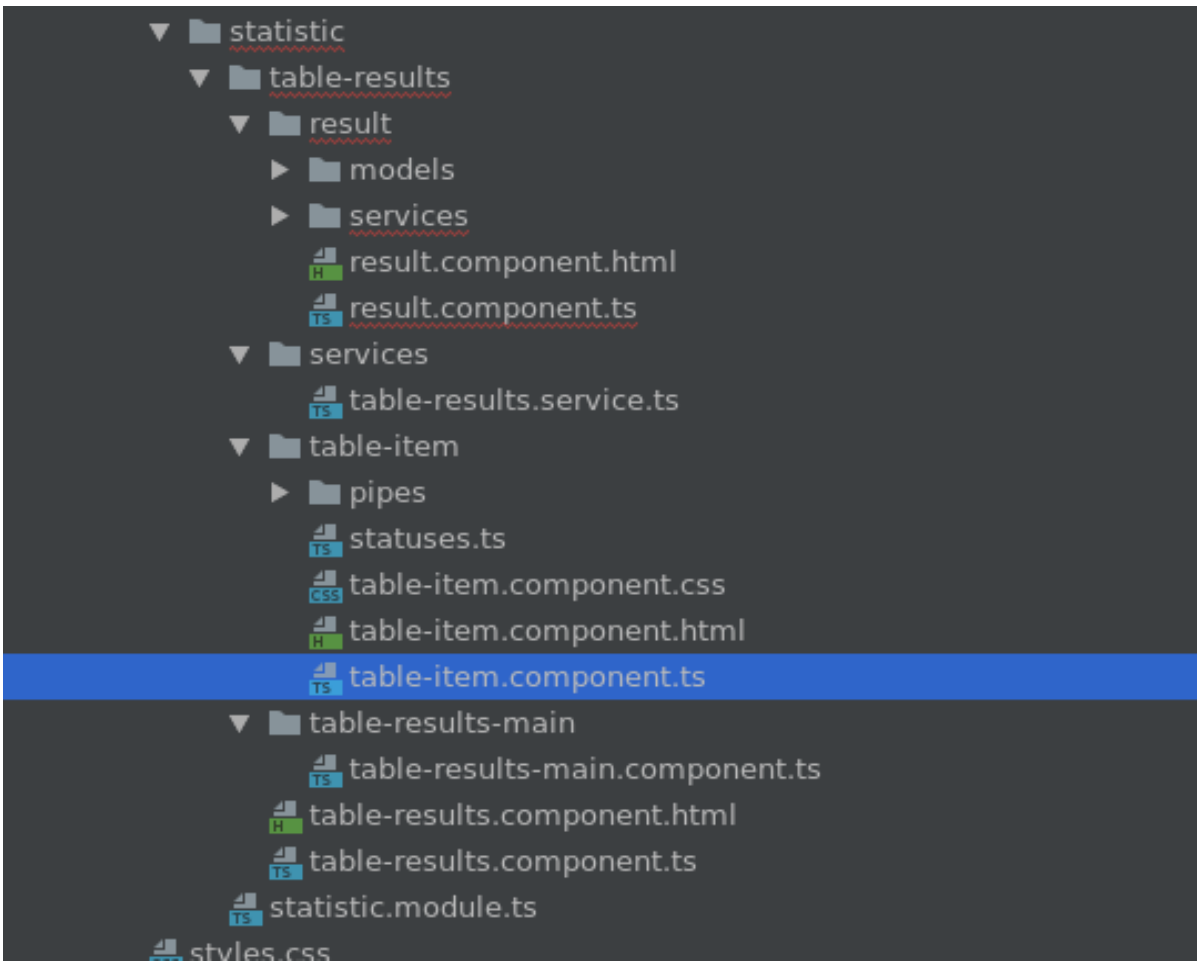

Рисунок 4.6 — Модуль статистики

Підключення модулю статистики на панелі користувача (рис 4.7).

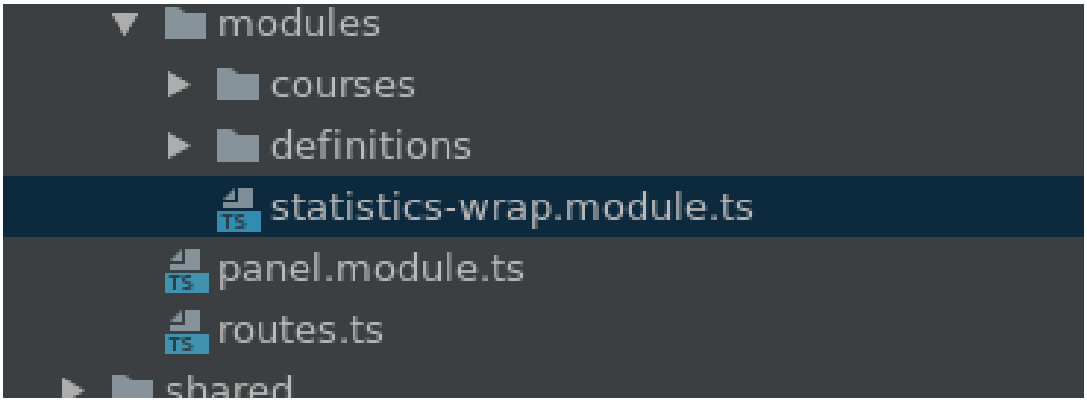

Рисунок 4.7 — Підключення модулю статистики

У панелі користувача статистика відображається наступним чином (рис 4.8).

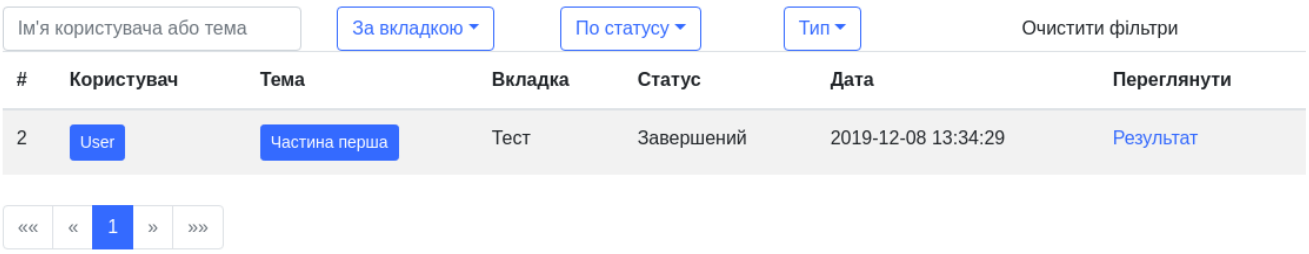

Рисунок 4.8 — Лістинг статистики

Фільтрація статистики відбувається за такими критеріями як владка, статус та тип статистики. Фільтрація по критерію вкладки (рис. 4.9).

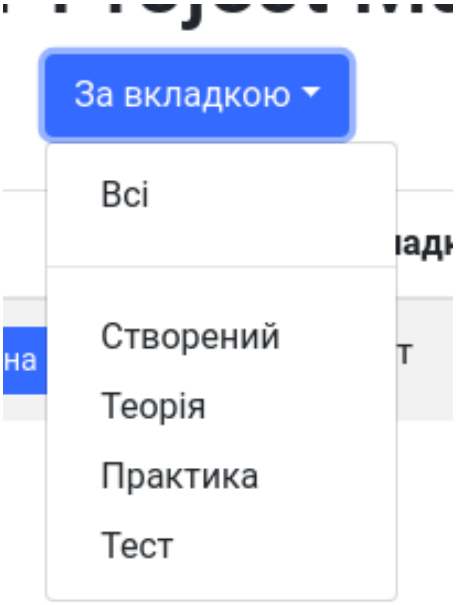

Рисунок 4.9 – Фільтрація за вкладкою

Фільтрація по критерію статусу проходження екзамену (рис. 4.10).

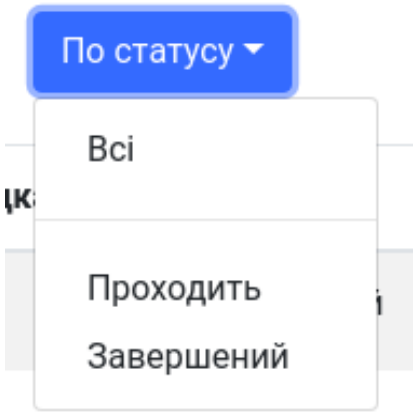

Рисунок 4.10 – Фільтрація за статусом

Фільтрація статистики по типу видаленої статистики чи показу всієї статистики (рис. 4.11).

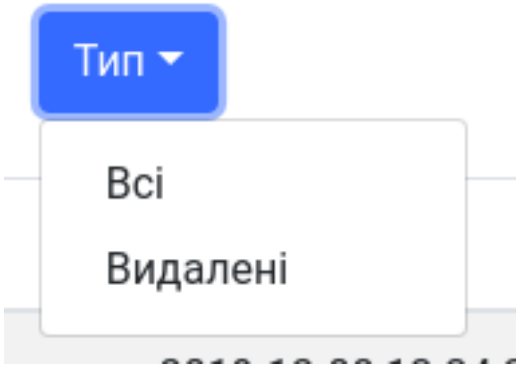

Рисунок 4.11 – Фільтрація по типу

Сторінка відображення статистики по вибраній темі з практичної частини (рис. 4.11). Нарисунку можна побачити кількість правильних відповідей та на самій діаграмі текст «Правильно» означає що блок перетягнуто правильно.

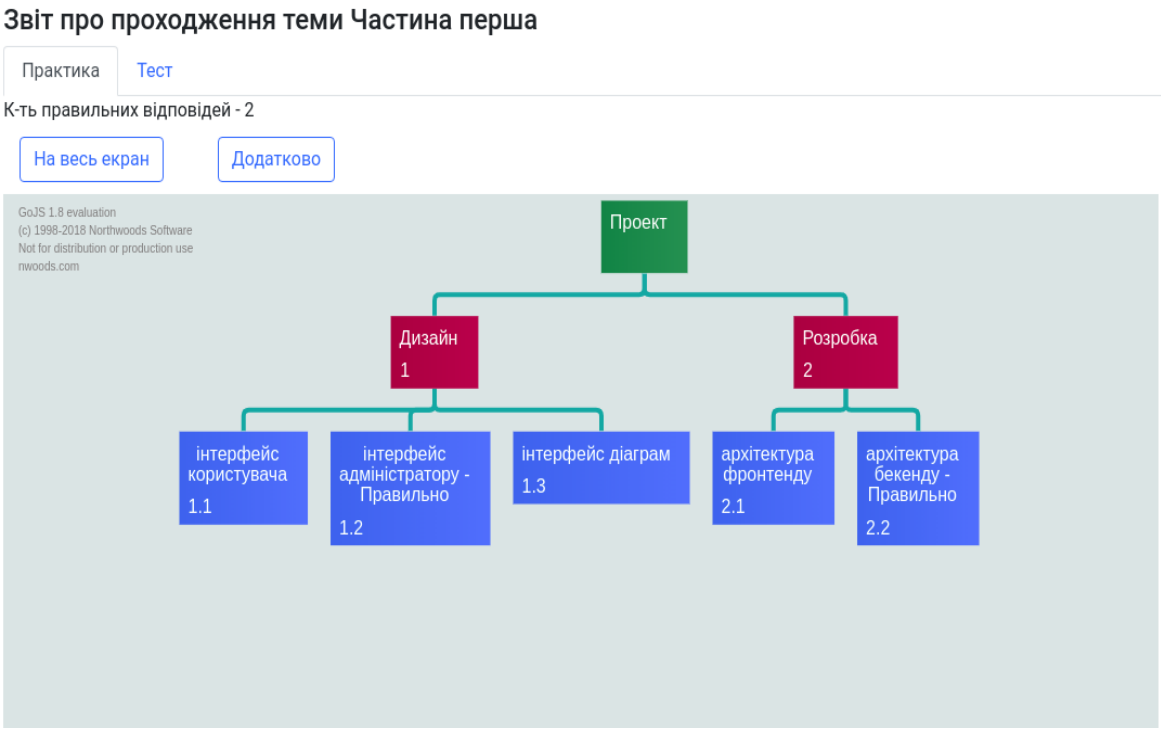

Рисунок 4.11 — Сторінка статистики

Сторінка статистики по конкретній темі за результатами тестів має наступний вигляд (рис. 4.12).

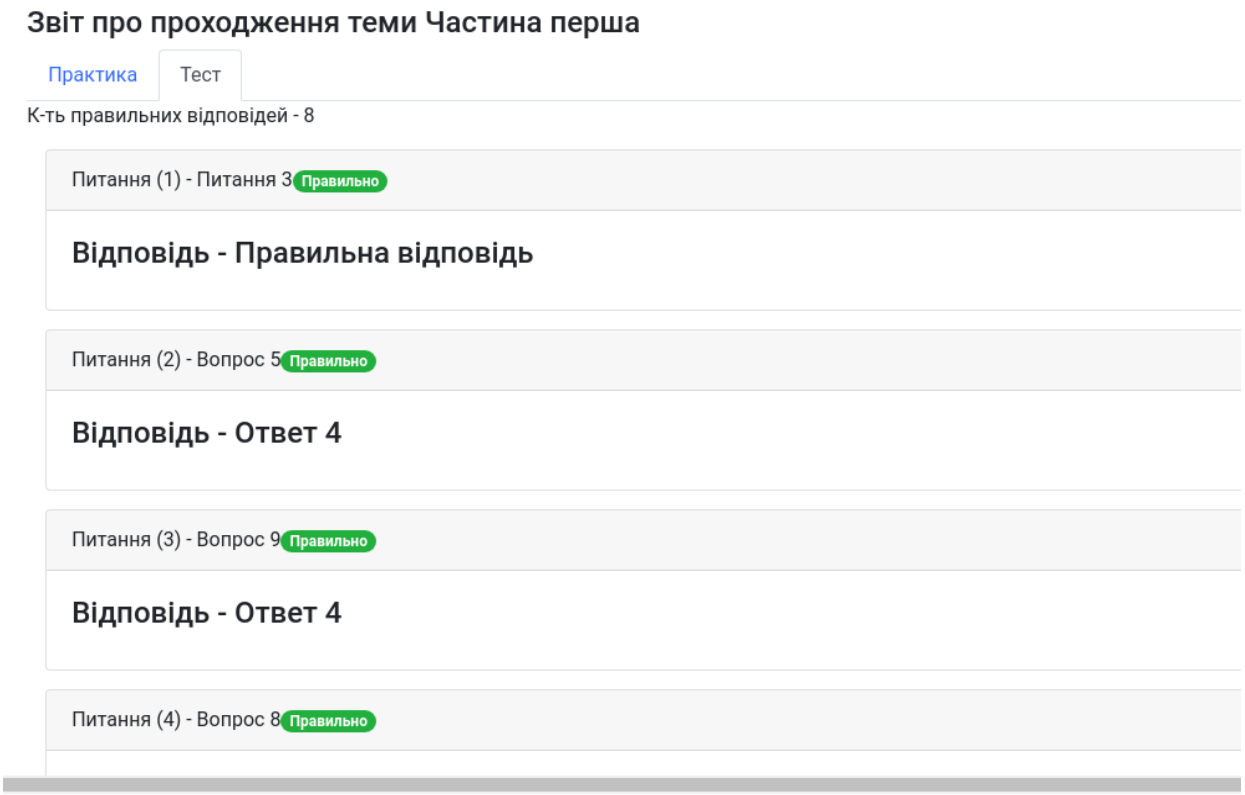

Рисунок 4.12 — Сторінка результатів тестування

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/.env
```

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, підключення необхідних директорій для роботи додатку:

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, пере направлення 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"

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

Длязапускудодаткупотрібновиконатикомандувтерміналіразомзпроектом: docker-compose up. Результат команди успішного запуску (рис. 4.13).

```
jashka@jashka ≡ ~/job/projects/php/managment ≡ docker-compose up
Starting managment_admin_panel_1 ... done
Starting managment_database_1    ... done
Starting managment_user_panel_1  ... done
Starting managment_traefik_1     ... done
Starting managment_cache_1       ... done
managment_app_1 is up-to-date
Attaching to managment_admin_panel_1, managment_user_panel_1, managment_cache_1, managment_traefik_1, managment_database_
admin_panel_1  |
admin_panel_1  | > ap@0.0.0 start /var/www/app
admin_panel_1  | > ng serve --aot --host 0.0.0.0 --disableHostCheck --public-host http://zzz-yashka.asuscomm.com/admin-pa
admin_panel_1  |
user_panel_1   |
user_panel_1   | > app@0.0.0 start /var/www/app
user_panel_1   | > ng serve --aot --host 0.0.0.0 --disableHostCheck --public-host http://zzz-yashka.asuscomm.com/user-pan
user_panel_1   |
cache_1        | 1:C 08 Dec 2019 16:06:39.234 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
cache_1        | 1:C 08 Dec 2019 16:06:39.234 # Redis version=5.0.5, bits=64, commit=00000000, modified=0, pid=1, just st
cache_1        | 1:C 08 Dec 2019 16:06:39.234 # Warning: no config file specified, using the default config. In order to
cache_1        | 1:M 08 Dec 2019 16:06:39.236 * Running mode=standalone, port=6379.
cache_1        | 1:M 08 Dec 2019 16:06:39.237 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/
cache_1        | 1:M 08 Dec 2019 16:06:39.237 # Server initialized
cache_1        | 1:M 08 Dec 2019 16:06:39.237 # WARNING overcommit_memory is set to 0! Background save may fail under low
1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
```

Рисунок 4.13 — Запуск додатку

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 (рис. 4.14).

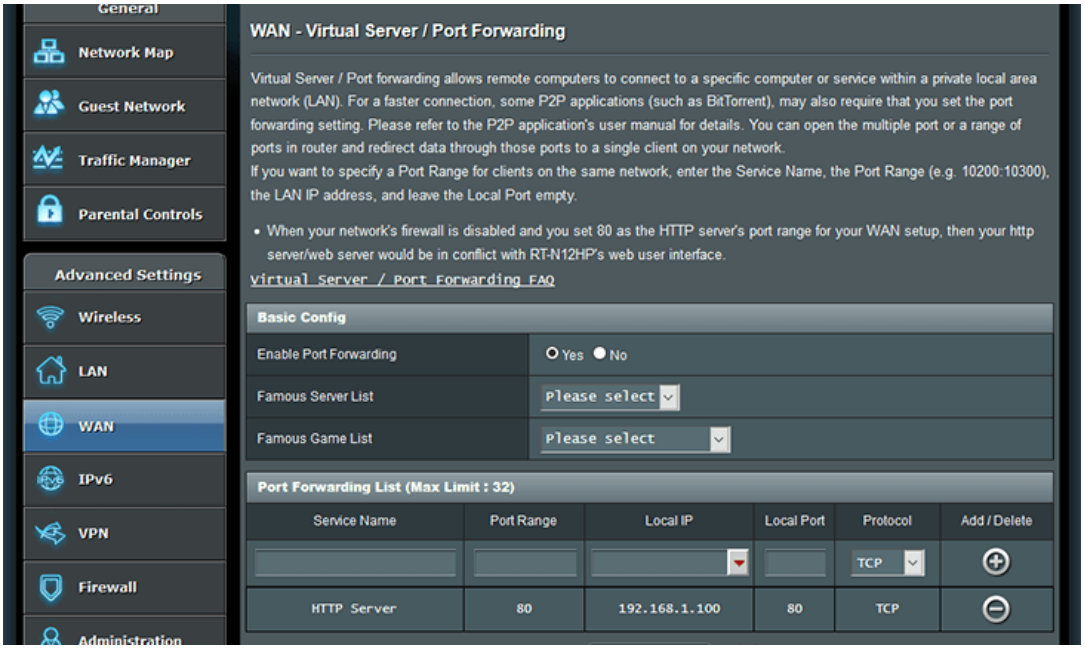

Рисунок 4.14 – Налаштування маршрутизатору

За допомогою перенаправленню 80 портує можливість побачити, що додаток

доступний для мережі Інтернет по такій адресіhttp://zzz-yashka.asuscomm.com (рис. 4.15).

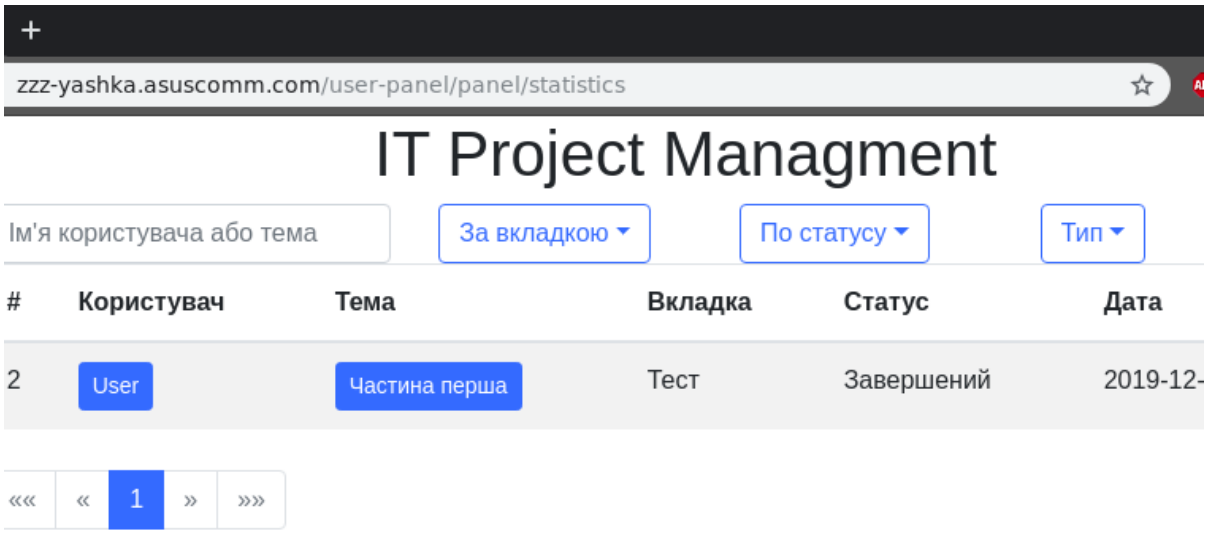

Рисунок 4.15 – URLдодатку

ВИСНОВКИ

Продуктом проекту єаналітична підсистема веб-орієнтованого середовища з вивчення дисципліни «Управління ІТ проектами». В результаті проведеної роботи було зроблено: лістинг статистики з фільтрацією по користувачу, по імені користувача або темі, по статусу проходження, реалізовано «soft-delete» видалення статистики, що дозволяє користувачу проходити тест повторно але адміністратор має можливість переглянути всю статистику по користувачу. Для адміністратору реалізовано список активних користувачів, які проходять тест по конкретній темі, реалізоване розмежування прав перегляду статистики: користувач може переглядати тільки свою статистику, а адміністратор статистику всіх користувачів.

За весь час розробки аналітичної підсистеми веб-орієнтованого середовища з вивчення дисципліни «Управління ІТ проектами» було проведено такі роботи як оновлення версії фреймворків, оновлення кодової бази фронтенду та бекенду стосовно нових версій фреймворків, прийнято та реалізовано рішення self-hosted розміщення додатку та його постійного деплою і використання архітектури CQRS для реалізації аналітичної підсистеми.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Clarizen - повнофункціональне програмне забезпечення для управління проектами [Електронний ресурс] - Режим доступу до ресурсу : https://www.clarizen.com/.
2. monday.com - інструмент планування з дошками kanban, трекер проектів [Електронний ресурс] – Режим доступу до ресурсу: https:// monday.com .
3. Celoxis - веб-інструмент із всеосяжними функціями управління проектами та портфелем [Електронний ресурс] - Режим доступу до ресурсу: https://celoxis.com.
4. 10,000ft - програмне забезпечення для управління проектами [Електронний ресурс] - Режим доступу до ресурсу: https://www.10000ft.com.
5. Ravetree - нагородами рішення для управління проектами з великою кількістю вбудованих функцій спритного управління проектами [Електронний ресурс] - Режим доступу до ресурсу: https://www.ravetree.com.
6. Wrike - програмне забезпечення для співпраці та управління проектами на основі хмар, яке легко масштабувати [Електронний ресурс] - Режим доступу до ресурсу: https://www.wrike.com.
7. Angular – фреймворк для фронтенду розроблений компанією Google [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/.
8. Javascript - мова програмування бекенду, фронтенду [Електронний ресурс] – Режим доступу до ресурсу: learn.javascript.ru/intro/.
9. TypeScript — мова програмування для розширення можливостей JavaScript[Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/.
10. PHP — мова програмування, яка дозволяє за допомогою скриптів створювати бекенд [Електронний ресурс] – Режим доступу до ресурсу: http://php.net/.

11. Laravel — PHP фреймворк, який використовується для написання бекенду [Електронний ресурс] — Режим доступу до ресурсу: https://laravel.com.
12. MySQL —система управління базами даних [Електронний ресурс] — Режим доступу до ресурсу: https://www.mysql.com/.
13. RxJS — бібліотека для реактивного програмування [Електронний ресурс] - Режим доступу до ресурсу: http://reactivex.io/rxjs/manual/index.html.
14. Docker — система віртуалізації на основі механізмів Linux [Електронний ресурс] - Режим доступу до ресурсу: https://www.docker.com/.
15. GoJS — бібліотека для створення інтерактивних діаграм [Електронний ресурс] - Режим доступу до ресурсу: https://gojs.net/latest/index.html.
16. DHTMLX — бібліотека для створення діаграмГанта [Електронний ресурс] - Режим доступу до ресурсу: https://dhtmlx.com/.
17. Google Cloud — «хмарна» платформадля підняття інфраструктури [Електронний ресурс] - Режим доступу до ресурсу: https://cloud.google.com/.
18. Github — сервіс для надання хостингу коду [Електронний ресурс] - Режим доступу до ресурсу: ttps://github.com/.
19. CQRS – архітектурний шаблон проектування [Електронний ресурс] — Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/azure/architecture/patterns/cqrs.
20. Роберт Седжвик. Алгоритми на C++. Boston/San Francisco/New York/Toronto/Montreal/London/Munich/Paris/Madrid/Capetown/Tokyo/Singapore/Mexico City — Addison-Wesley, 2014 —634-645.
21. EventSourcing – архітектурний шаблон проектування [Електронний ресурс] —Режим доступу до ресурсу: https://martinfowler.com/eaaDev/EventSourcing.html.

ДОДАТОК А

ПЛАНУВАННЯ РОБІТ

Деталізація мети проекту методом SMART.
Результати деталізації методом SMART розміщені у табл. А.1.

Таблиця А.1 − Деталізація мети методом SMART

Specific (конкретна)	Створити аналітичну підсистему веб-орієнтованого середовища з вивчення дисципліни «Управління ІТ проектами».
Measurable (вимірювана)	Проходи практичні та теоретичні екзамени з різних тем дисципліни «Управління ІТ проектами».
Achievable (досяжна)	Вивчати дисципліну «Управління ІТ проектами» за рахунок практичної роботи з діаграмами та перевірки своїх знань.
Relevant (реалістична)	Прискорити процес вивчення дисципліни з «Уплавління ІТ проектами» та набути навичок з розробки діаграм.
Time-framed (обмежена у часі)	Реалізувати необхідний функціонал не пізніше обумовленого із замовником терміну.

Мета проекту: Створення аналітичної підсистеми веб-орієнтованого середовища з вивчення дисципліни «Управління ІТ проектами» для студентів.

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 IT 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 1.

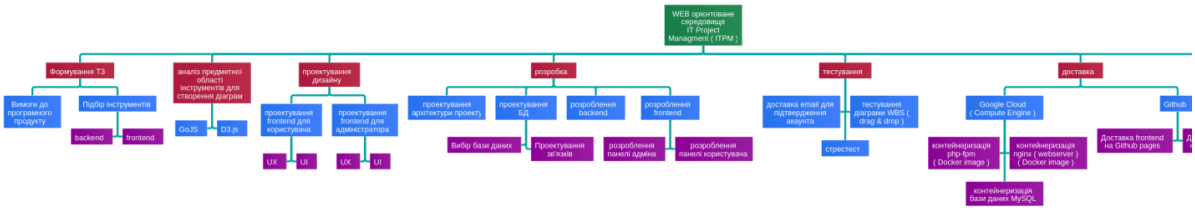

Рисунок 1 – WBS структура проекту

Організаційна структура проекту (OBS)

OBS – це структура розбиття організації визначає організаційні відносини і використовується як основа для розподілу робочих обов'язків. Перетин OBS і WBS визначає точки управлінської підзвітності за роботу під назвою Контрольні (або Витрати). Будуємо діаграму OBS (рис. 2).

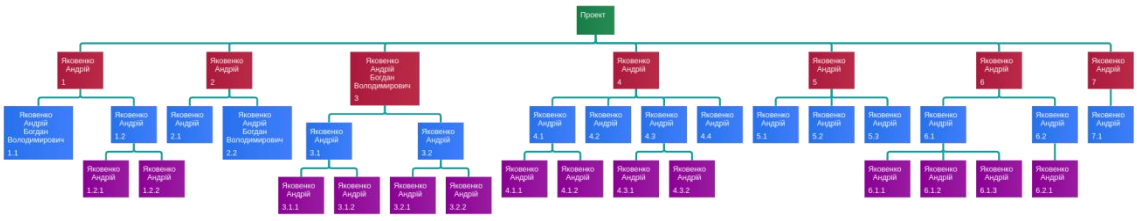

Рисунок 2 – діаграма OBS

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.

Календарний графік зображено на рисунку 3.

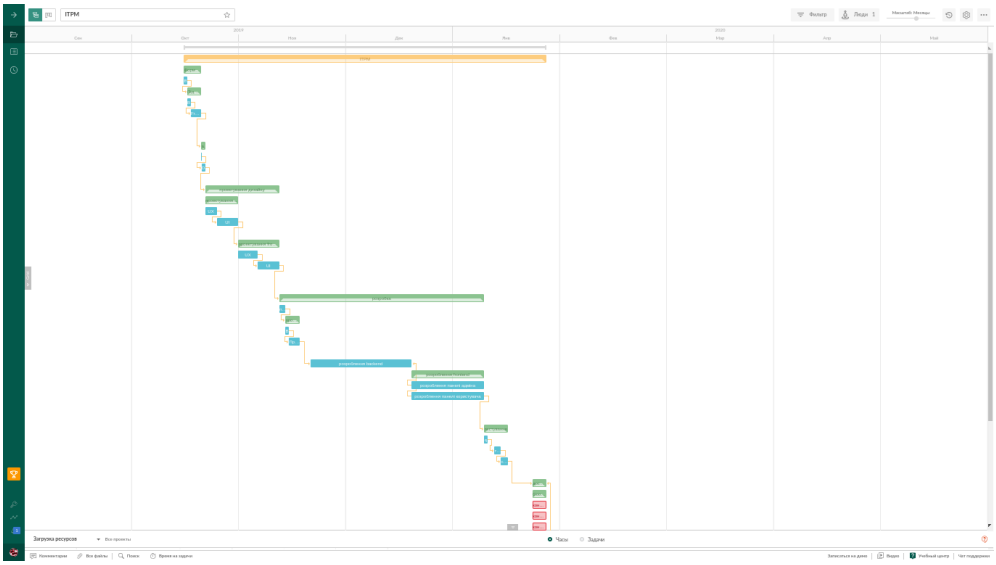

Рисунок 4 – діаграма Ганта

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 закінчився його 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 А.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 А.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 йморівності-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 А.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.

ДОДАТОК Б

КОД БЕКЕНДУ

Файл StatisticsRepository.php призначений для пошуку статистичних даних за різними 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
{

    /**
     * @return \Illuminate\Database\Eloquent\Builder
     */
public function getQuery()
    {
return ExaminationInfo::query();
    }

    /**
     * @param array $params
     * @param $userData
     * @internal param $user
     * @return \Illuminate\Contracts\Pagination\LengthAwarePaginator
     */
public function search(array $params, $userData)
```

```
    {
        $query = $this->getQuery();
        $status = $this->getStatus($params);
        $tab = $this->getTab($params);
        $user = $this->getUser($params);
        $role = $this->getRole($userData);
        $unit = $this->getUnit($params);
        $search = $this->getSearch($params);

if ($role === 'user') {
            $query = $this->filterByUser($query, $userData->id);
        }

if ($role === 'admin') {
            $query = $this->filterByUser($query, $user);
            $query = $query->withTrashed();
        }

        $query->with([
            'user',
            'unit'
        ]);

        $query = $this->filterByStatus($query, $status);
        $query = $this->filterByTab($query, $tab);
        $query = $this->filterByUnit($query, $unit);
        $query = $this->filterBySearch($query, $search);

        $query = $query->latest();

        $data = $query->paginate();

return $data;

    }

    /**
     * @param $id
     * @param $userData
     * @return mixed
     */
public function loadResult($id, $userData)
    {
        $query = $this->getQuery();
        $role = $this->getRole($userData);

if ($role === 'user') {
            $query = $this->filterByUser($query, $userData->id);
        }

        $result = $query
            ->where('id', '=', $id)
            ->with([
```

```
            'user',
            'unit.theory',
            'schema',
            'test.correct_answer',
            'test.answer',
            'test.question',
        ])
        ->first();

    $result->schema->append('type');

return $result;

    }

public function softDelete(int $id)
    {
        \DB::transaction(function () use ($id) {
            $examination = $this
                ->getQuery()
                ->with(['schema', 'test'])
                ->find($id);

            $schemaResult = $examination->schema;
            $schemaResult->delete();

            $testResults = $examination->test;
foreach ($testResults as $result) {
                $result->delete();
            }

            $examination->delete();
        });
    }

public function restore(int $id)
    {

    }

    /**
     * @param Builder $query
     * @param $unit
     * @return Builder
     */
protected function filterByUnit(Builder $query, $unit)
    {
if ($unit) {
            $query->where('unit_id', '=', $unit);
        }

return $query;
    }
```

```
    /**
     * @param Builder $query
     * @param $userId
     * @return Builder
     */
protected function filterByUser(Builder $query, $userId)
    {
if ($userId) {
            $query->where('user_id', '=', $userId);
        }

return $query;
    }

    /**
     * @param Builder $query
     * @param $tab
     * @return Builder
     */
protected function filterByTab(Builder $query, $tab)
    {
if ($tab) {
            $query->tab(intval($tab));
        }

return $query;
    }

    /**
     * @param Builder $query
     * @param $status
     * @return Builder
     */
protected function filterByStatus(Builder $query, $status)
    {
        // $query->where('status', '>', ExaminationInfo::STARTED);
if ($status) {
            $query->where('status', '=', $status);
        }

return $query;
    }

    /**
     * @param Builder $query
     * @param $search
     * @return Builder
     */
protected function filterBySearch(Builder $query, $search)
    {
if ($search) {
            $query->whereHas('unit', function (Builder $query) use ($search) {
```

```
                $query->where('name', 'like', '%' . $search . '%');
            })
                ->orWhereHas('user', function (Builder $query) use ($search) {
                    $query->where('name', 'like', '%' . $search . '%');
                });
        }

return $query;
    }

    /**
     * @param array $params
     * @return mixed
     */
protected function getSearch(array $params)
    {
return array_get($params, 'search');
    }

    /**
     * @param $params
     * @return mixed
     */
protected function getUnit($params)
    {
return array_get($params, 'unit_id');
    }

    /**
     * @param $params
     * @return mixed
     */
protected function getStatus($params)
    {
return array_get($params, 'status');
    }

    /**
     * @param $params
     * @return mixed
     */
protected function getTab($params)
    {
return array_get($params, 'current_tab');
    }

    /**
     * @param $params
     * @return mixed
     */
protected function getUser($params)
    {
return array_get($params, 'user_id');
```

```
    }

    /**
     * @param $user
     * @return array
     */
protected function getRole($user)
    {
return $user->role;
    }
}
```

Файл StatisticsController.php призначений для обробки запитів на отримання статистичних даних

```
<?php

namespace App\Http\Controllers;

use App\Model\ExaminationInfo;
use App\Policies\StatisticPolicy;
use Illuminate\Cache\Repository as CacheRepository;
use Illuminate\Http\Request;
use Repository\ExaminationInfoRepository;
use Repository\StatisticsRepository;

class StatisticController extends Controller
{
    /**
     * @var ExaminationInfoRepository
     */
private $statisticsRepository;

    /**
     * @var CacheRepository
     */
private $cacheRepository;

    /**
     * StatisticController constructor.
     * @param StatisticsRepository $statisticsRepository
     * @param CacheRepository $cacheRepository
     * @internal param CacheManager $cacheManager
     * @internal param ExaminationInfo $examinationInfo
     */
public function __construct(StatisticsRepository $statisticsRepository, CacheRepository
$cacheRepository)
    {
        $this->statisticsRepository = $statisticsRepository;
        $this->cacheRepository = $cacheRepository;
```

```
    }

    /**
     * @param Request $request
     * @return \Illuminate\Http\JsonResponse
     */
public function filter(Request $request)
    {
        $data = $this->statisticsRepository->search($request->all(), auth()->user());

return response()->json(compact('data'));
    }

    /**
     * @param $id
     * @param ExaminationInfo $examinationInfo
     * @return \Illuminate\Http\JsonResponse
     * @internal param ExaminationInfo $statistic
     */
public function show($id, ExaminationInfo $examinationInfo)
    {
        $user = auth()->user();
        $statistic = $examinationInfo->withTrashed()->find($id);

if (!$user->can(StatisticPolicy::CAN_VIEW, $statistic)) {
return response()->json([
                'message' => trans('api.access_denied')
            ], 403);
        }

        $key = 'result_' . $id;

if (!$this->cacheRepository->has($key)) {
            $this->cacheRepository->forever($key, $this->statisticsRepository->loadResult($id, auth()-
>user()));
        }

        $data = $this->cacheRepository->get($key);

return response()->json(compact('data'));
    }

public function softDelete(int $id, ExaminationInfo $examinationInfo)
    {
        $user = auth()->user();
        $statistic = $examinationInfo->withTrashed()->find($id);

if (!$user->can(StatisticPolicy::CAN_SOFT_DELETE, $statistic)) {
return response()->json([
                'message' => trans('api.access_denied')
            ], 403);
        }
```

```
        $this->statisticsRepository->softDelete($id);

return response()->json([
            'message' => trans('api.soft_deleted')
        ]);
    }
}
```

ДОДАТОК В

Файл table-results.component.ts призначений для лістингу статистики та фільтрації.

```
import { Component, OnInit } from "@angular/core";

import { Subject } from "rxjs";

import { ALL, SOFT_DELETE, TableResultsService } from "./services/table-results.service";
import { IStatus, statuses, tabs } from "./table-item/statuses";

import { Result } from './result/models/result';
import { AuthService } from '../../auth/services/auth.service';
import { PaginationService } from '../../pagination/pagination/service/pagination.service';
import { Unit } from '../../itpm/models/unit';
import { User } from '../../auth/model/user';
import { IPagination } from '../../pagination/pagination/model/pagination';
import { debounceTime, distinctUntilChanged } from 'rxjs/operators';

@Component({
selector: 'app-table-results',
templateUrl: './table-results.component.html',
})
export class TableResultsComponent implements OnInit {

public tabs: Array<IStatus>;
public statuses: any;
public search: string;
public searchSubject: Subject<string> = new Subject<string>();
public isAdmin: boolean = false;

constructor(private TableResultsService: TableResultsService,
private AuthService: AuthService,
public statistics: PaginationService<Result>) {
   this.isAdmin = this.AuthService.getUser().role === 'admin';
   this.tabs = Object.keys(tabs).map((key) => {
return {id: +key, text: tabs[key]};
   });
   this.statuses = Object.keys(statuses).map((key) => {
return {id: +key, text: statuses[key]};
   });
  }

public ngOnInit() {
this.TableResultsService.getStatistics().subscribe(pagination => {
this.statistics.update(pagination);
```

```
      this.searchSubject
        .pipe(
debounceTime(500),
distinctUntilChanged(),
        )
        .subscribe(model => {
this.TableResultsService.bySearch(model).getStatistics().subscribe(this.update.bind(this));
        });
    });
  }

public onSearchChange(search: string) {
this.searchSubject.next(search);
  }

public filterByTab(tab: number) {
this.TableResultsService.byTab(tab).getStatistics().subscribe(this.update.bind(this));
  }

public filterByStatus(status: number) {
this.TableResultsService.byStatus(status).getStatistics().subscribe(this.update.bind(this));
  }

public filterByUnit(unit: Unit) {
this.TableResultsService.byUnit(unit.id).getStatistics().subscribe(this.update.bind(this));
  }

public filterByUser(user: User) {
this.TableResultsService.byUser(user.id).getStatistics().subscribe(this.update.bind(this));
  }

public all() {
this.TableResultsService.byType(ALL).getStatistics().subscribe(this.update.bind(this));
  }

public bySoftDelete() {
this.TableResultsService.byType(SOFT_DELETE).getStatistics().subscribe(this.update.bind(this));
  }

public softDelete(id: number) {
this.TableResultsService.softDelete(id).subscribe((result: Result) => {
const stat = this.statistics.getFromCollection(id);
      stat.deleted_at = new Date().toISOString();
this.statistics.replaceInCollection(id, stat);
    });
  }

public clearFilters() {
    this.search = '';
this.searchSubject.next(this.search);
this.TableResultsService.clearParams().getStatistics().subscribe(this.update.bind(this));
  }
```

```
protected update(pagination: IPagination<Result>) {
this.statistics.update(pagination);
  }

}
```

Файл table-results.component.html призначений для макету сторінки

```
<div class="row">
<div class="col-md-12">
<div class="row">
<div class="col-md-3">
<input [(ngModel)]="search" (ngModelChange)="onSearchChange($event)" type="text" class="form-
control"
placeholder="Ім'я користувача або тема">
</div>
<div class="col-md-2">
<div ngbDropdown class="d-inline-block">
<button class="btn btn-outline-primary" id="dropdownTab"
ngbDropdownToggle>Завкладкою</button>
<div ngbDropdownMenu aria-labelledby="dropdownTab">
<button (click)="filterByTab(null)" class="dropdown-item">Всі</button>
<hr>
<button *ngFor="let tab of tabs" (click)="filterByTab(tab.id)" class="dropdown-item">
                    {{tab.text}}
</button>
</div>
</div>
</div>
<div class="col-md-2">
<div ngbDropdown class="d-inline-block">
<button class="btn btn-outline-primary" id="dropdownStatus"
ngbDropdownToggle>Постатусу</button>
<div ngbDropdownMenu aria-labelledby="dropdownStatus">
<button (click)="filterByStatus(null)" class="dropdown-item">Всі</button>
<hr>
<button *ngFor="let status of statuses" (click)="filterByStatus(status.id)"
class="dropdown-item">{{status.text}}
</button>
</div>
</div>
</div>
<div class="col-md-2">
<div ngbDropdown class="d-inline-block">
<button class="btn btn-outline-primary" id="dropdownStatus" ngbDropdownToggle>Тип</button>
<div ngbDropdownMenu aria-labelledby="dropdownStatus">
<button (click)="all()" class="dropdown-item">Всі</button>
<button (click)="bySoftDelete()" class="dropdown-item">Видалені</button>
</div>
</div>
```

```
</div>
<div class="col-md-2">
<button type="button" class="btn btn-default" (click)="clearFilters()">Очиститифільтри</button>
</div>
</div>
</div>
<div class="col-md-12">
<div *ngIf="statistics.data" class="table-responsive">
<table class="table table-striped">
<thead>
<tr>
<th>#</th>
<th>Користувач</th>
<th>Тема</th>
<th>Вкладка</th>
<th>Статус</th>
<th>Дата</th>
<th>Переглянути</th>
<th *ngIf="isAdmin">#</th>
</tr>
</thead>
<tbody>
<tr *ngFor="let item of statistics.data"
table-item
                [item]="item"
                (onClickUnit)="filterByUnit($event)"
                (onClickUser)="filterByUser($event)"
                (onSoftDelete)="softDelete($event)"
>
</tr>
</tbody>
</table>
</div>
</div>

<div class="col-md-12">
<pagination [paginator]="statistics"></pagination>
</div>

</div>
```

Файл result.component.ts призначений для показу результату статистики

```
import { Component, OnInit } from "@angular/core";
import { ActivatedRoute } from "@angular/router";

import { ResultService } from "./services/result.service";
import { Result } from "./models/result";
```

```
import { DiagramInstanceConfig, instances } from '../../../diagram/instances';
import { Permissions } from '../../../diagram/security/permission/permissions';
import { rc4 } from '../../../diagram/security/rc4';

@Component({
selector: 'app-result',
templateUrl: `./result.component.html`,
})
export class ResultComponent implements OnInit {

public result: Result;
public countCorrectTestAnswers: number = 0;
public countCorrectSchemaAnswers: number = 0;
public countAllSchemaAnswers: number = 0;
public countAllTestAnswers: number = 0;

public config: any = {};
public types: DiagramInstanceConfig[] = [];

constructor(private ActivatedRoute: ActivatedRoute,
private ResultService: ResultService) {
   this.types = instances;
   this.config.role = 'user';
   this.config.permission = Permissions.ONLY_READ;
   this.config.controls = false;
   this.config.showBtnSave = false;
  }

public ngOnInit() {
const id: number = +this.ActivatedRoute.snapshot.params['id'];
this.ResultService.getResult(id).subscribe(data => {
     this.result = data;
     this.countCorrectSchemaAnswers = this.result.schema.diagram.nodeDataArray
        .filter(node => node.hasOwnProperty('dropped_data'))
        .filter(node => node.result.result === true).length;
     this.countCorrectTestAnswers = this.result.test.filter(answer => answer.is_right === true).length;
     this.countAllSchemaAnswers = this.result.schema.diagram.nodeDataArray
        .filter(node => node.hasOwnProperty('dropped_data'))
        .length;
     this.countAllTestAnswers = this.result.test.length;
});
  }
}
```

Файл result.component.html призначений для макету статистики

```
<div class="row">
<div *ngIf="result; else loader" class="col-md-12">

<h4>Звітпропроходженнятеми {{result.unit.name}}</h4>
```

```
<ngb-tabset [destroyOnHide]="false">
<ngb-tab title="Практика">
<ng-template ngbTabContent>
<div>К-тьправильнихвідповідей - {{ countCorrectSchemaAnswers }} з {{ countAllSchemaAnswers
}}</div>
<schema [instances]="types"
                    [schema]="result.schema"
                    [config]="config"></schema>
</ng-template>
</ngb-tab>
<ngb-tab title="Тест">
<ng-template ngbTabContent>
<div>К-тьправильнихвідповідей - {{ countCorrectTestAnswers }} з {{ countAllTestAnswers }}</div>
<div *ngFor="let qa of result.test; let questionIndex = index"
style="margin-top: 15px"
class="col-md-12">
<div class="card">
<div class="card-header">
<span>Питання ({{questionIndex + 1}}) - {{qa.question.name}}</span>
<span
class="badge badge-pill"
                                [ngClass]="{
              'badge-danger': !qa.is_right,
              'badge-success': qa.is_right
             }">  {{qa.is_right ? ' Правильно ' : ' Неправильно '}} </span>
</div>
<div class="card-body">
<h4 class="card-title">Відповідь - <span>{{qa.answer.name}}</span></h4>
<p *ngIf="!qa.is_right" class="card-text">Правильнавідповідь -
                            {{qa.correct_answer.name}}</p>
</div>
</div>

</div>

</ng-template>
</ngb-tab>
</ngb-tabset>
</div>

<ng-template #loader>Loading...</ng-template>

</div>
```

result.service.ts

```
import { Injectable } from "@angular/core";
```

```
import { Observable } from "rxjs";
import { map } from 'rxjs/operators';

import { QuestionResult, Result } from "../models/result";
import { AppHttpClient } from '../../../../itpm/services/app-http-client';
import { rc4 } from '../../../../diagram/security/rc4';

@Injectable({
providedIn: 'root',
})
export class ResultService {

private api: string;

constructor(private HttpClient: AppHttpClient) {
    this.api = `statistics`;
  }

public getResult(id: number): Observable<Result> {
return this
      .HttpClient
      .withoutPrefixUrl()
      .get<Result>(`${this.api}/${id}`)
      .pipe(
map((result: Result) => {
          result.test = this.fillRightAnswers(result.test);
          result.schema.diagram = this.decodeSchemaData(result.user_id, result.schema.diagram);
return result;
        })
      );
  }

private fillRightAnswers(questions: Array<QuestionResult>): Array<QuestionResult> {
return questions.map((question: QuestionResult) => {
      question.is_right = question.correct_answer.id === question.answer.id;
return question;
    });
  }

private decodeSchemaData(userId: number, schema: string): any {
const res = rc4(userId.toString(), schema);

returnJSON.parse(res);
  }
}
```

Файл result.ts призначений для опису моделей на фронтенді

```
import { Answer } from '../../../../itpm/models/test/answer';
import { Question } from '../../../../itpm/models/test/question';
import { Unit } from '../../../../itpm/models/unit';
import { User } from '../../../../auth/model/user';

export interface ISchema {
  created_at: string;
id: number;
type: string;
diagram: string | any;
  examination_info_id: number;
  schema_id: number;
  updated_at: string;
}

export interface QuestionResult {
answer: Answer;
  answer_id: number;
  correct_answer: Answer;
  correct_answer_id: number;
  examination_info_id: number;
id: number;
question: Question;
  question_id: number;
  is_right?: boolean;
}

export interface Result {
  current_tab: number;
id: number;
schema: ISchema;
status: number;
test: Array<QuestionResult>;
unit: Unit;
  unit_id: number;
user: User;
  user_id: number;
uuid: string;
  created_at: string;
  updated_at: string;
  deleted_at?: string;
}
```