Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка програмного модуля для парсингу документів з використанням фреймворку Laravel

Виконав: студент IV курсу, групи СП-43
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

_____ Трембецький Т.Н.
(підпис) (прізвище та ініціали)

Керівник _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент _____ Матійчук Л.П.
(підпис) (прізвище та ініціали)

Тернопіль 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р.
(підпис) (прізвище та ініціали)
«__» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Студенту Трембецькому Тарасу Назарійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного модуля для парсингу документів з використанням фреймворку Laravel

Керівник роботи Стоянов Юрій Миколайович, к.т.н., доцент кафедри програмної інженерії
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 18 » *квітня* 2025 року № *4/7-335*

2. Термін подання студентом завершеної роботи 23 червня 2025 р.

3. Вихідні дані до роботи Наукові публікації, електронні ресурси, підручники , посібники з тематики дослідження.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області. 1.1. Постановка задачі. 1.2. Аналіз відомих рішень в даній предметній області. 1.3. Специфікація вимог до системи. 2. Проектування системи для парсингу документів. 2.1. Архітектура модуля. 2.2. Проектування бази даних. 3. Програмна реалізація та тестування модуля для парсингу документів. 3.1. Вибір засобів реалізації програмного модуля. 3.2. Програмна реалізація модуля. 3.3. Програмна реалізація бази даних. 3.4. Тестування модуля. 3.5. Інструкція користувача системи. 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1.Тема. 2. Мета роботи. 3-7. Аналіз існуючих аналогів. 8. Архітектура модуля. 9. Технології реалізації модуля. 10. Програмна реалізація модулів – імпорт/експорт. 11. Реалізація бази даних. 12-13. Тестування модуля. 14. Інструкція користувача та висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Лазарюк В.В., к.т.н., доцент кафедри МТ		
Нормоконтроль	Стоянов Ю.М., доцент кафедри програмної інженерії		

7. Дата видачі завдання 18 квітня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області	18.04.2025-22.04.2025	*Виконано*
2	Огляд існуючих рішень	23.04.2025-27.04.2025	*Виконано*
3	Проєктування та розробка платформи	27.04.2025-30.04.2025	*Виконано*
4	Підготовка пояснювальної записки	01.05.2025-16.05.2025	*Виконано*
5	Спецчастина	17.05.2025-06.06.2025	*Виконано*
6	Підготовка презентації та доповіді	07.06.2025-08.06.2025	*Виконано*
7	Попередній захист	9.06.2025	*Виконано*
8	Нормоконтроль		
9	Рецензування		
10	Занесення диплома в електронний архів		
11	Допуск до захисту у зав. кафедри		

Студент _____ Трембецький Т.Н.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр містить:72с., 59 рис., 4 табл., 24 джер.

Ключові слова: автоматизація обробки документів, парсинг документів, архітектура системи, програмні компоненти, Laravel, PHP, MySQL, структура бази даних.

Метою кваліфікаційної роботи є розробка програмних компонентів для парсингу документів з використанням фреймворку Laravel.

Розглянуто процес проектування бази даних, реалізацію програмних компонентів із використанням PHP, Laravel та MySQL, а також проведено функціональне тестування.

Проведено аналіз предметної області, здійснено огляд сучасних підходів до обробки даних та розглянуто наявні програмні рішення. Сформульовано основні вимоги до системи, розроблено її архітектурну модель на основі трирівневої структурної концепції.

Описано інструкцію користувача щодо взаємодії із системою, що дозволяє імпортувати, фільтрувати, експортувати дані та отримувати звіти. Результати дослідження підтверджують ефективність розробленої системи в автоматизації роботи з великими масивами даних, що сприяє покращенню продуктивності та точності інформаційної обробки.

ABSTRACT

The qualification work for the bachelor's degree contains: 72 p., 59 fig., 4 tab., 25s.

Key words: automation of document processing, document parsing, system architecture, software components, laravel, php, mysql, database structure.

The purpose of the qualification work is to develop software components for document parsing using the Laravel framework.

The database design process, implementation of software components using PHP, Laravel and MySQL, and functional testing were considered.

An analysis of the subject area was conducted, a review of modern approaches to data processing was conducted, and existing software solutions were considered. The main requirements for the system were formulated, and its architectural model was developed based on a three-level structural concept.

The user manual for interacting with the system is described, allowing you to import, filter, export data, and receive reports. The results of the study confirm the effectiveness of the developed system in automating work with large data sets, which contributes to improving the productivity and accuracy of information processing.

ЗМІСТ

ВСТУП

З кожним роком дані стають все більш цінним ресурсом для будь-якої організації, будь то комерційна компанія чи державна установа. У центрі процесу аналізу даних часто знаходиться Excel – потужний інструмент для роботи з таблицями, який вже багато років є невід'ємною частиною офісного життя. Попри свою універсальність, сучасні обсяги інформації та вимоги до швидкості обробки даних змушують шукати нові підходи до використання цього інструменту.

Ручна робота з великими масивами даних в Excel не тільки трудомістка, але й піддається ризику помилок, які можуть стати причиною неправильних управлінських рішень. Крім того, виникає потреба у більш тісній інтеграції Excel з іншими IT-системами підприємства, що вимагає спеціалізованих технічних рішень.

Розробка компонентів для автоматизації процесів роботи з Excel-документами та їх інтеграція в корпоративні системи відкриває нові можливості: від пришвидшення обробки даних до підвищення точності інформації та покращення взаємодії між підрозділами. Це підвищує ефективність роботи окремого співробітника та сприяє глибшій аналітиці поточних процесів у компанії.

За допомогою спеціалізованих компонентів можна уніфікувати методи роботи з даними, що, в свою чергу, призведе до зменшення помилок та розбіжностей у даних. Це також сприяє більш ефективному навчанню нових співробітників і полегшує процес впровадження нових технологій у робочі процеси.

Метою кваліфікаційної роботи є розробка програмних компонентів для парсингу документів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести огляд літературних джерел;
- виконати проектування програмних компонентів;
- виконати реалізацію програмних модулів;
- провести тестування програмних модулів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка задачі

Було поставлено завдання розробити програмні компоненти для обробки документів формату xlsx та xls, які складаються за визначеними правилами. Ці формати стали стандартами для зберігання та обробки даних у багатьох сферах діяльності, тому автоматизація процесів роботи з ними є важливою для підвищення ефективності. Одним з найбільш популярних і поширених додатків для обробки файлів цих форматів є Microsoft Excel. Excel є потужним інструментом, який використовується в різноманітних сферах, таких як бізнес, наука, освіта, фінанси, дослідження та аналітика, що робить його основним інструментом для роботи з даними в організаціях по всьому світу.

Excel-файл — це електронна таблиця, створена за допомогою програмного забезпечення Microsoft Excel, одного з найбільш потужних інструментів для роботи з даними в світі. Завдяки своїм можливостям, Excel дозволяє виконувати складні математичні, статистичні та фінансові розрахунки, створювати графіки, діаграми та інші візуалізації, а також обробляти великий обсяг даних. Excel активно використовується не лише для зберігання даних, але й для їх аналізу, створення звітів, прогнозів і навіть автоматизації рутинних завдань через макроси. Це робить Excel незамінним інструментом для професіоналів у різних галузях.

Сьогодні дані є найціннішим активом для будь-якої організації, і здатність ефективно управляти ними може значно підвищити продуктивність і забезпечити прийняття обґрунтованих і своєчасних рішень. Усі сучасні компанії використовують великі обсяги даних для аналізу ринку, управління фінансами, прогнозування та прийняття стратегічних рішень. Проте без належної автоматизації обробка цих даних займає величезну кількість часу і ресурсів. Компоненти для обробки Excel-файлів дозволяють автоматизувати рутинні процедури, такі як збір, обробка, аналіз, оновлення та розподіл інформації, що значно зменшує час на виконання цих завдань. Крім того, автоматизація знижує

ймовірність людських помилок, що важливо для забезпечення точності та надійності даних.

Використання спеціалізованих програмних компонентів для автоматизованої обробки даних у форматах Excel дозволяє інтегрувати ці інструменти в існуючі інформаційні системи компанії. Це забезпечує безперебійне та ефективне взаємодія між різними системами і базами даних підприємства. Інтеграція даних між різними програмними засобами робить можливим швидке і точне оновлення інформації в різних системах і зменшує кількість помилок, що можуть виникнути через неузгодженість даних. Це особливо важливо для великих організацій, які працюють з величезними обсягами даних, оскільки дозволяє забезпечити актуальність і доступність інформації для кінцевих користувачів, що у свою чергу підвищує ефективність та оперативність прийняття рішень.

Особливу увагу слід приділити аспектам реалізації інструментів обробки даних, таких як фільтрація, сортування та агрегування даних, що дозволяє зручніше працювати з великими масивами інформації. Розробка таких методів для обробки великих обсягів даних дозволяє не тільки підвищити продуктивність, а й створити нові можливості для більш глибокого аналізу та візуалізації інформації. Це, в свою чергу, дозволяє компаніям проводити більш точний аналіз даних, робити прогнози, формувати звіти та приймати рішення на основі надійної і актуальної інформації.

Крім того, розробка таких компонентів також сприяє покращенню роботи з даними для кінцевих користувачів. Спеціалізовані компоненти дозволяють спростити інтерфейс роботи з даними, уніфікувати методи роботи з інформацією і зробити процес навчання нових співробітників значно простішим. Це дозволяє більш швидко та ефективно впроваджувати нові технології в робочі процеси компанії і забезпечує кращу адаптацію працівників до змін. В результаті це веде до зниження кількості помилок, підвищення ефективності роботи та загального підвищення продуктивності в організації.

1.2 Аналіз відомих рішень в даній предметній області

Були розглянуті приклади рішень, пов'язані з аналогічними задачами обробки даних. Необхідно розглянути наступні напрямки: інструменти для обробки Excel-файлів, веб-сайти з особливостями роботи фільтрів, а також сайти, які пропонують можливості для перегляду великої кількості даних. Такими прикладами є tableconvert.com, https://www.dataconverter.org/ , https://csvjson.com/csv2json, https://www.online2pdf.com/, https://www.aconvert.com/ та https://www.zamzar.com/ відповідно.

TableConvert.com, Рис. 1.1. TableConvert.com — це онлайн-сервіс, який надає можливість конвертувати табличні дані між різноманітними форматами, включаючи CSV, JSON, XML, HTML, SQL та Excel. Сервіс дозволяє завантажувати, редагувати та завантажувати дані у необхідному форматі. Він забезпечує безпеку даних, не зберігаючи їх на сервері після завершення роботи. На рисунку 1 представлено головну сторінку TableConvert.com.

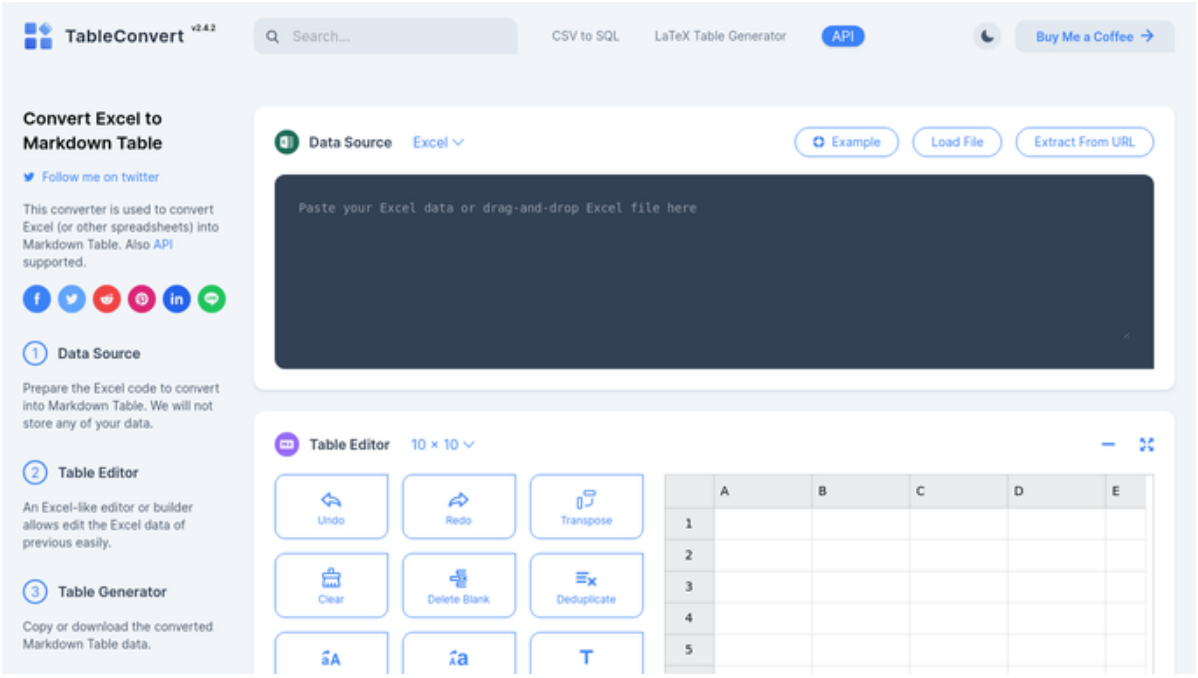

Рис. 1.1. Головна сторінка TableConvert.com

Однією з ключових переваг TableConvert є можливість редагувати дані безпосередньо на сайті до їх конвертації. При цьому для використання сервісу не потрібно встановлювати додаткове програмне забезпечення — всі операції виконуються безпосередньо в браузері.

ConvertCSV (https://www.convertcsv.com/), Рис. 1.2. Це онлайн-сервіс для конвертації CSV файлів у різні формати, такі як Excel, JSON, SQL, XML і інші. Можна редагувати та перетворювати дані перед збереженням.

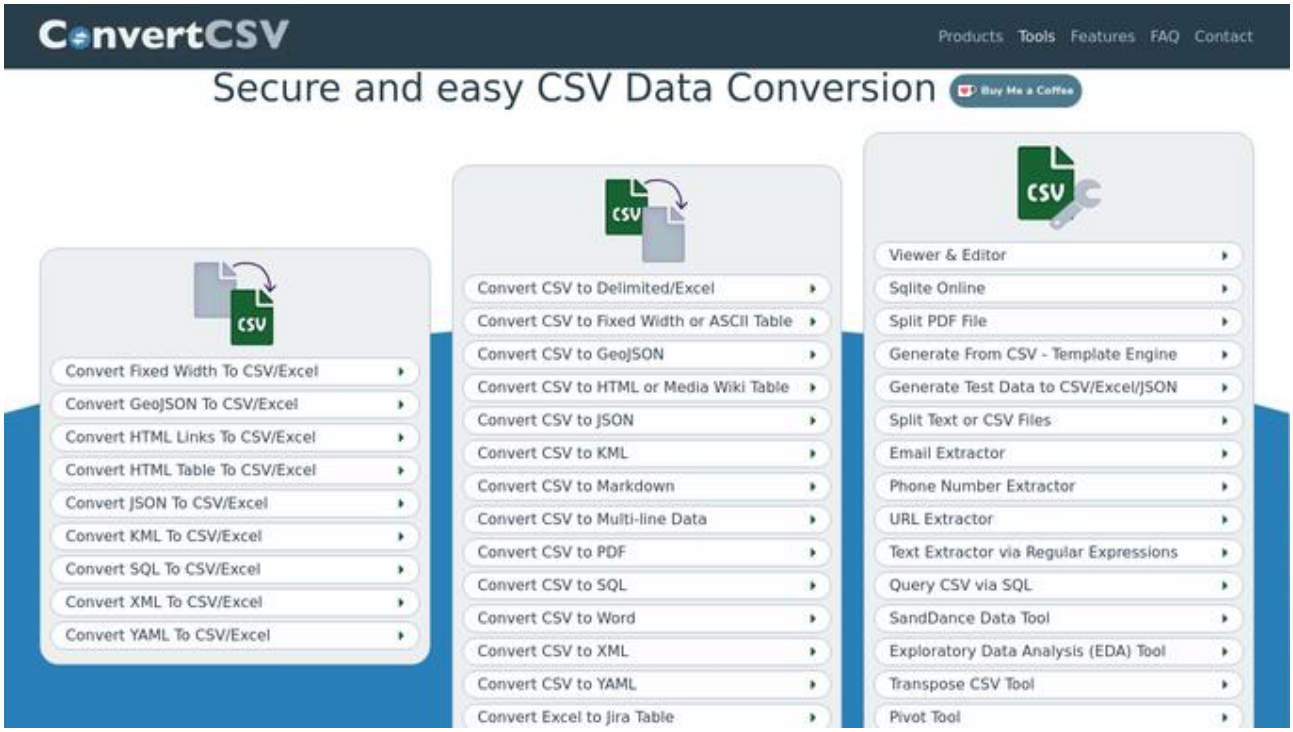

Рис. 1.2. Головна сторінка ConvertCSV

Data Converter (https://www.dataconverter.org/), Рис. 1.3. Платформа, яка дозволяє конвертувати дані між різними форматами, такими як CSV, Excel, XML, JSON, TSV тощо. Також підтримує редагування даних перед конвертацією.

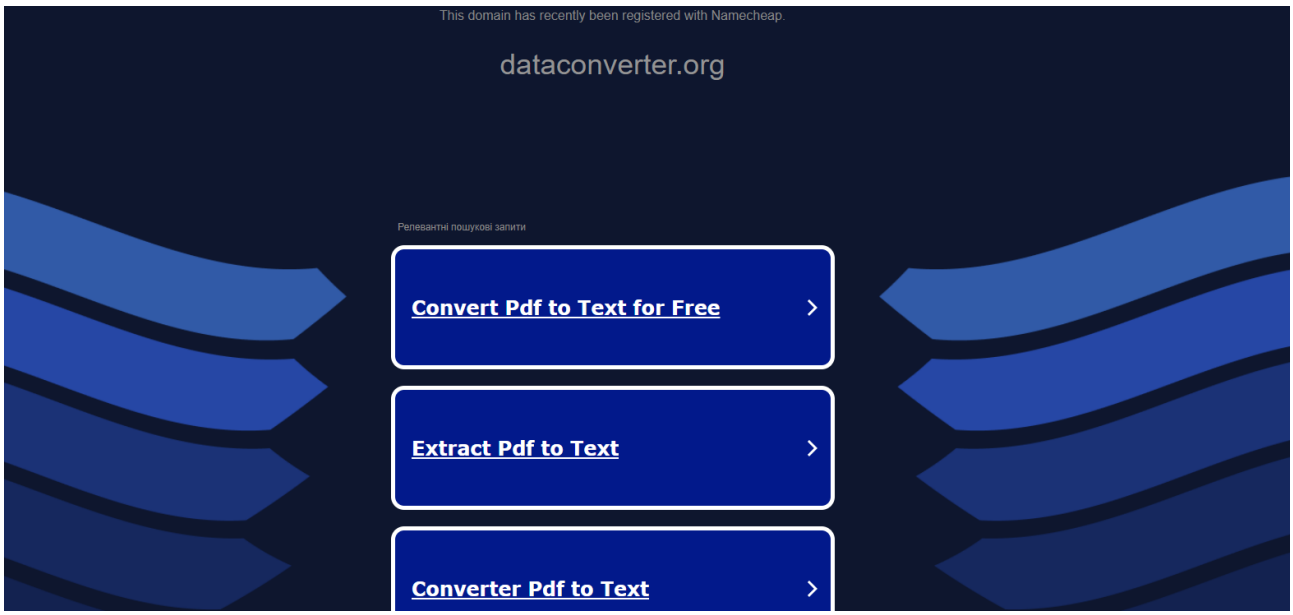

Рис. 1.3. Головна сторінка Data Converter

CSVJSON (https://csvjson.com/csv2json), Рис. 1.4. Онлайн інструмент для конвертації CSV файлів у JSON і навпаки. Це корисний ресурс для швидкої конвертації і маніпуляцій з даними.

Query	Impressions	Clicks	Average Position	CTR
	106,402 % of Total: 100.00% (106,402)	11,246 % of Total: 100.00% (11,246)	6.6 % of Total: 100.00% (6.6)	10.57% Avg for View: 10.57% (0.00%)
1. json beautifier	39,464 (37.09%)	265 (2.36%)	7.3(112.36%)	0.67%
2. (not set)	15,903 (14.95%)	1,648 (14.65%)	6.4 (95.59%)	10.36%
3. csv to json	11,805 (11.09%)	5,635 (50.11%)	1.4 (7.03%)	47.73%
4. json beautify	5,923 (5.57%)	71 (0.63%)	5.7 (84.51%)	1.20%
5. beautify json	5,204 (4.89%)	33 (0.29%)	6.5 (97.40%)	0.63%
6. convert csv to json	2,345 (2.20%)	895 (7.96%)	1.2 (2.95%)	38.17%
7. json converter	1,455 (1.37%)	22 (0.20%)	9.1(144.80%)	1.51%
8. csv to json converter	1,101 (1.03%)	281 (2.50%)	2.0 (17.54%)	25.52%
9. sql to json	939 (0.88%)	425 (3.78%)	1.1 (1.16%)	45.26%
10. json to sql	889 (0.84%)	23 (0.20%)	7.8(121.16%)	2.59%

Рис. 1.4. Головна сторінка CSVJSON

Online2PDF (https://www.online2pdf.com/), Рис. 1.5. Хоча основний фокус цього сервісу — це конвертація PDF-файлів, він також підтримує конвертацію таблиць у форматі CSV, Excel та інших.

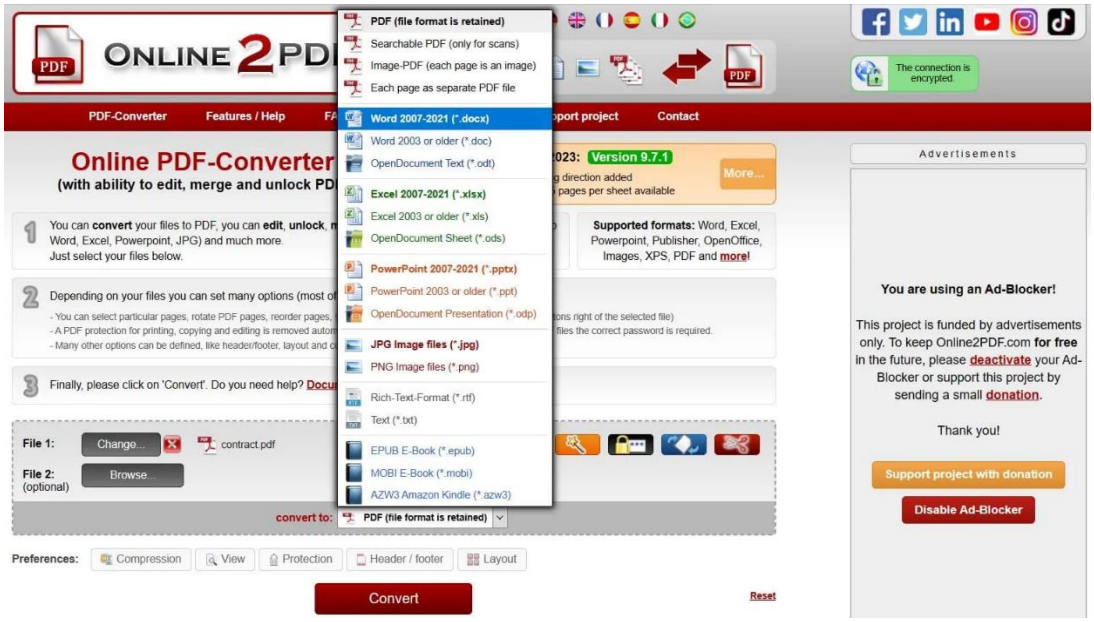

Рис. 1.5. Головна сторінка Online2PDF

AConvert (https://www.aconvert.com/), Рис. 1.6. Онлайн сервіс для конвертації різних типів файлів, включаючи таблиці. Він дозволяє конвертувати документи в різні формати і редагувати їх.

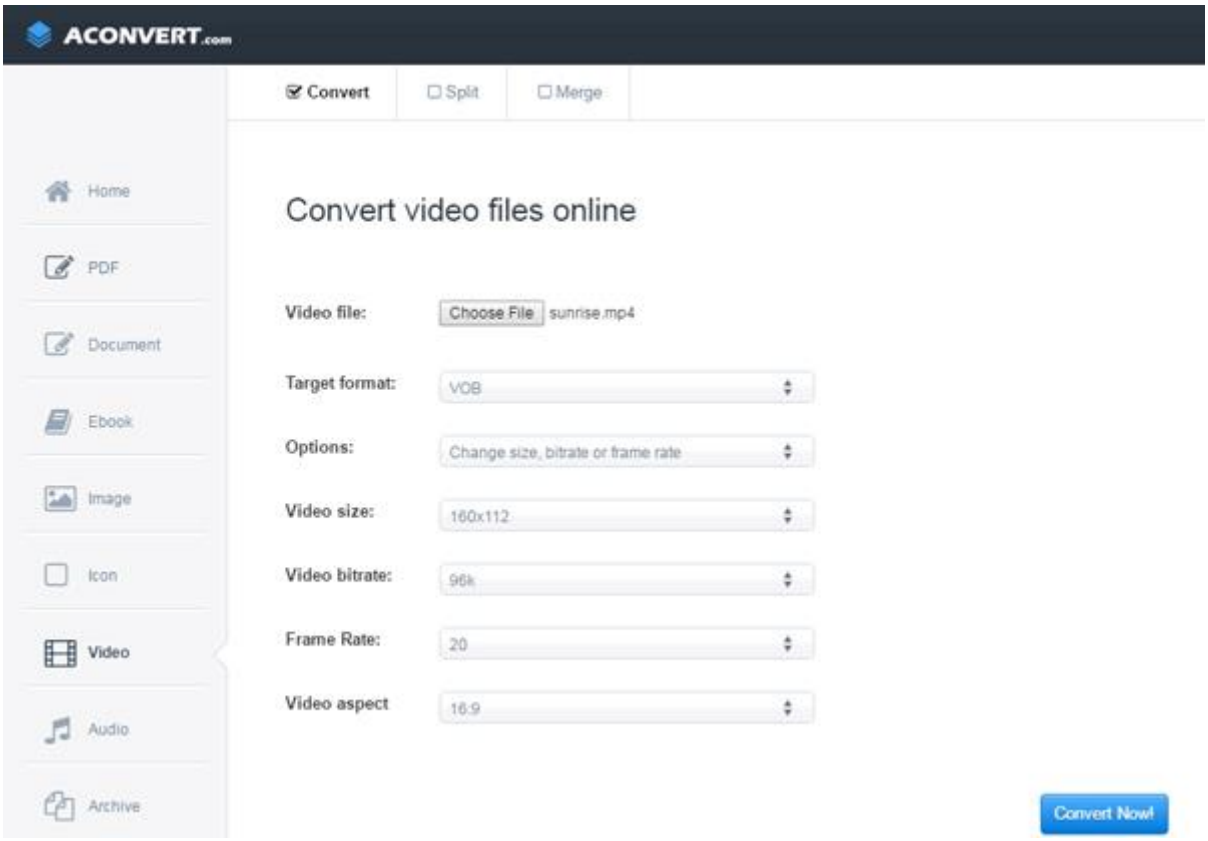

Рис. 1.6. Головна сторінка AConvert

Zamzar (https://www.zamzar.com/), Рис. 1.7. Популярний онлайн інструмент для конвертації файлів між різними форматами, зокрема підтримує Excel, CSV, JSON та інші формати таблиць.

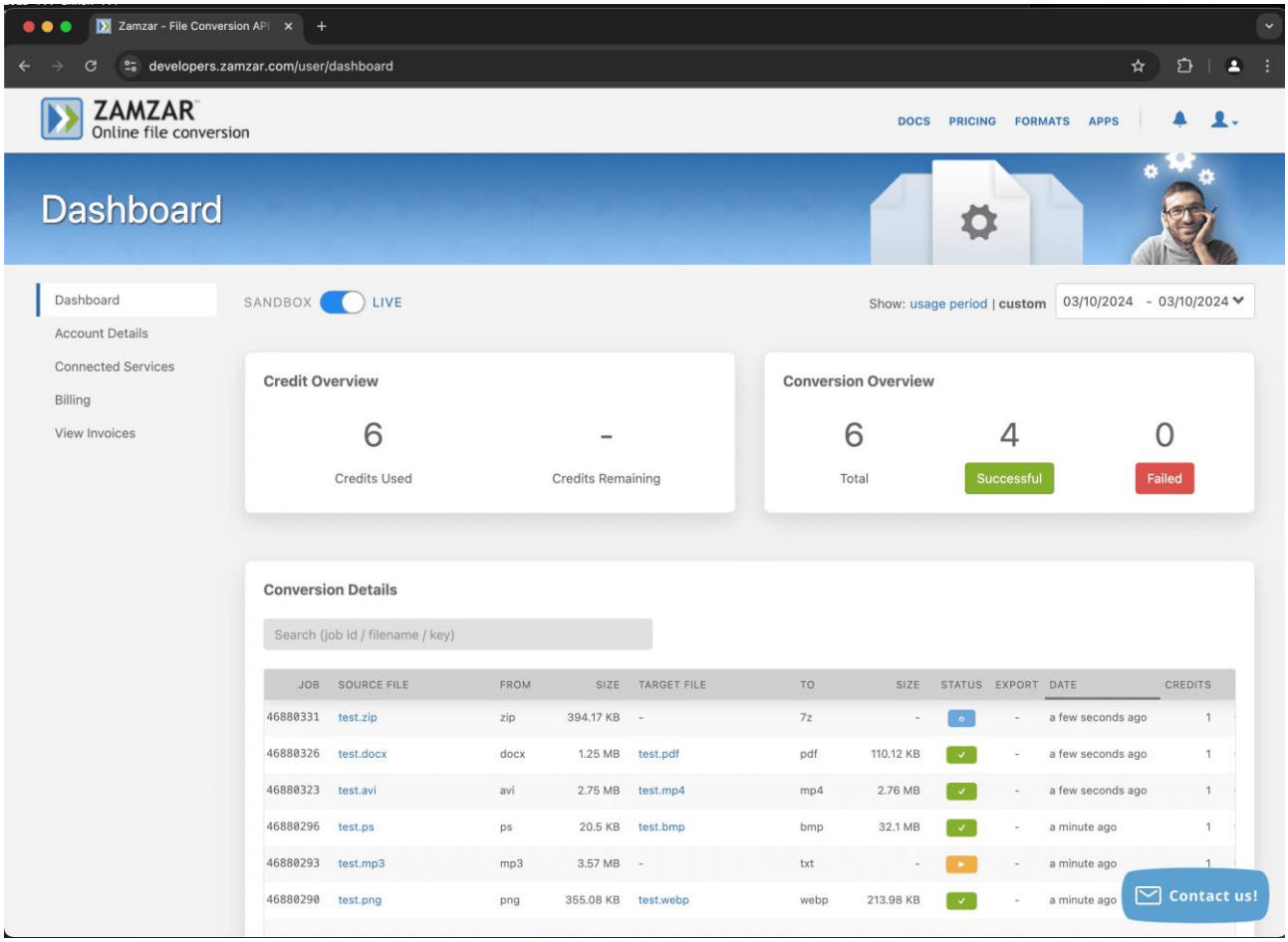

Рис. 1.7. Головна сторінка Zamzar

Ці сервіси дозволяють працювати з таблицями та конвертувати дані в різні формати), в таблиці 1.1. наведено переваги та недоліки проаналізованих сервісів.

Таблиця 1.1. Аналіз переваг та недоліки проаналізованих сервісів

Сервіс	Переваги	Недоліки
TableConvert.com	- Підтримка конвертації між багатьма форматами (CSV, XML, Excel, SQL)	- Безкоштовна версія має обмеження на кількість файлів для конвертації
	- Можливість редагувати дані перед збереженням	- Платні версії можуть бути дорогими для великих обсягів даних
	- Інтерфейс простий у використанні	
ConvertCSV	- Легкий у використанні інтерфейс для конвертації CSV у різні формати	- Підтримує лише обмежену кількість форматів порівняно з іншими сервісами
	- Швидка обробка і конвертація даних	- Можуть виникати проблеми при роботі з великими обсягами даних
Data Converter	- Підтримка широкого спектра форматів	- Може бути не таким швидким при обробці великих файлів
	- Можливість працювати з великими даними	- Інтерфейс може бути не таким інтуїтивно зрозумілим для новачків
CSVJSON	- Простий інтерфейс для конвертації між CSV і JSON	- Обмежений функціонал в порівнянні з більш багатофункціональними інструментами
	- Можливість перевіряти і редагувати JSON-формат перед збереженням	- Підтримує тільки обмежену кількість форматів
Online2PDF	- Підтримує конвертацію з/в PDF, включаючи CSV	- В основному орієнтований на конвертацію PDF-файлів, а не таблиць
	- Можливість виконувати інші операції, такі як злиття та розбиття PDF	- Для складних операцій з таблицями PDF формат може бути обмежуючим
AConvert	- Широкий вибір форматів для конвертації	- Потрібен час на завантаження файлів через інтерфейс браузера, що може бути незручним при великих обсягах
	- Можливість конвертувати документи в будь-якому форматі	- Може обмежувати розмір файлів у безкоштовній версії
Zamzar	- Проста система для конвертації файлів	- Безкоштовна версія має обмеження на кількість конвертацій в день
	- Широкий спектр підтримуваних форматів	- Потрібна реєстрація для доступу до деяких функцій

1.3. Специфікація вимог до системи

При проектуванні компонентів необхідно враховувати функціональні та нефункціональні вимоги до системи, в яку ці компоненти будуть інтегровані. Важливо також враховувати ці вимоги при роботі над реалізацією компонентів.

Функціональні вимоги представлено в таблиці 1.2. Розглянемо функціональні вимоги до системи:

1. Система повинна забезпечувати можливість завантаження файлів користувачем.
2. Система повинна забезпечувати можливість відправки файлу користувачеві для скачування.
3. Система повинна надавати можливість авторизації користувача.
4. Система повинна забезпечувати вільне переключення між компонентами обробки файлів для авторизованого користувача.

Таблиця 1.2. Аналіз функціональних вимог

№	Вимога	Опис
1	Завантаження файлів користувачем	Система повинна забезпечувати можливість завантаження файлів користувачем.
2	Відправка файлів користувачеві для скачування	Система повинна забезпечувати можливість відправки файлів користувачеві для скачування.
3	Авторизація користувача	Система повинна надавати можливість авторизації користувача.
4	Перемикання між компонентами обробки файлів для авторизованого користувача	Система повинна забезпечувати вільне переключення між компонентами обробки файлів для авторизованого користувача.

Нефункціональні вимоги представлено в таблиці 1.3. Розглянемо нефункціональні вимоги до системи:

1. Система повинна бути реалізована на мові програмування PHP версії 8.2 та фреймворку Laravel версії не менше 10.

2. Система повинна використовувати СУБД MySQL.

3. Система повинна запускатися та коректно відображатися в будь-якому з популярних браузерів: Google Chrome, Opera, Microsoft Edge, Mozilla Firefox.

Таблиця 1.3. –Аналіз нефункціональних вимог

№	Вимога	Опис
1	Мова програмування та фреймворк	Система повинна бути реалізована на мові програмування PHP версії 8.2 та фреймворку Laravel версії не менше 10.
2	Використання СУБД	Система повинна використовувати СУБД MySQL.
3	Підтримка браузерів	Система повинна запускатися та коректно відображатися в будь-якому з популярних браузерів: Google Chrome, Opera, Microsoft Edge, Mozilla Firefox.

При розробці компонентів необхідно враховувати різні сценарії їх використання. У цьому випадку користувач є єдиним актором, який взаємодіє з системою обробки документів. Цей користувач має доступ до всіх функцій системи. Діаграма варіантів використання буде відображати різні сценарії, в рамках яких користувач взаємодіє з компонентами. Діаграми варіантів використання представлена на рисунках 1.8 та 1.9.

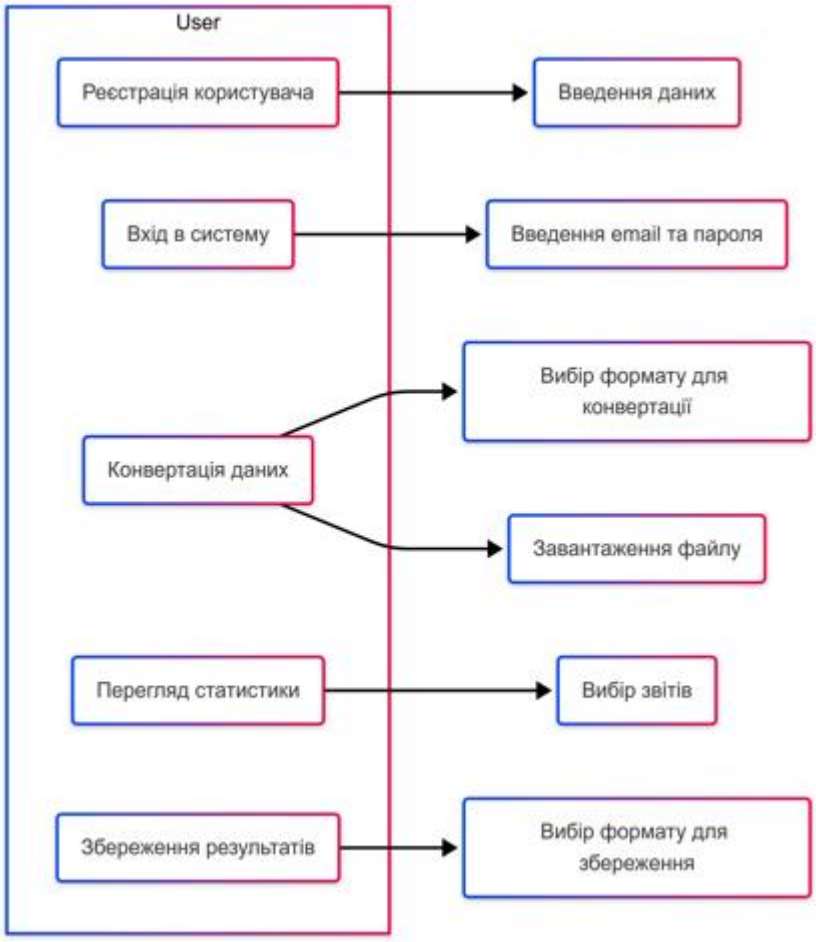

Рис. 1.8. Діаграма варіантів використання для користувача

Представлені наступні варіанти використання:

1. Робота з даними подій. Використовуючи компонент обробки файлів з даними подій, користувач може переглядати та аналізувати ці дані.

2. Робота з даними звернень. Використовуючи компонент обробки файлів з даними звернень, користувач може переглядати та аналізувати ці дані.

3. Імпорт даних. Користувач може завантажити в систему файл з новими даними, які імпортуються в таблицю бази даних.

4. Завантажити звіт. Користувач може завантажити повний звіт по всіх даних або тільки по відфільтрованих.

5. Відфільтрувати дані. Користувач може знайти необхідні дані в таблиці за допомогою фільтрів.

Рис. 1.9. Діаграма варіантів використання для адміністратора

6. Подивитись довідку по фільтрам. Користувач може подивитись довідку по фільтрах таблиці.

7. Експортувати дані. Користувач може експортувати всі або тільки відфільтровані дані з таблиці.

8. Завантаження даних про послуги. Використовуючи компонент обробки файлів з даними про послуги, користувач може завантажити дані в систему.

9. Завантаження статистики по частці звернень. Використовуючи компонент обробки файлів зі статистикою по частці звернень, користувач може завантажити дані по частці звернень в систему.

Було проведено детальний огляд предметної області та існуючих рішень для обробки та візуалізації даних у форматі Excel. Основний предмет розробки – це створення програмних компонентів для обробки Excel-файлів, що вимагає врахування особливостей цих форматів та можливостей їх інтеграції з інформаційними системами компанії.

2. ПРОЕКТУВАННЯ СИСТЕМИ ДЛЯ ПАРСИНГУ ДОКУМЕНТІВ

2.1. Архітектура модуля

Архітектура цієї системи будується на класичній трирівневій архітектурі веб-додатку, яка включає три основні шари: подання, бізнес-логіку та зберігання даних. Кожен рівень взаємодіє з іншими, забезпечуючи працездатність додатку. Трирівнева архітектура полегшує масштабування, підтримку та розробку додатків, оскільки розділяє функціональність на логічні блоки.

На рисунку 2.1 представлена архітектура всієї системи, товстим пунктиром позначені розроблювані компоненти.

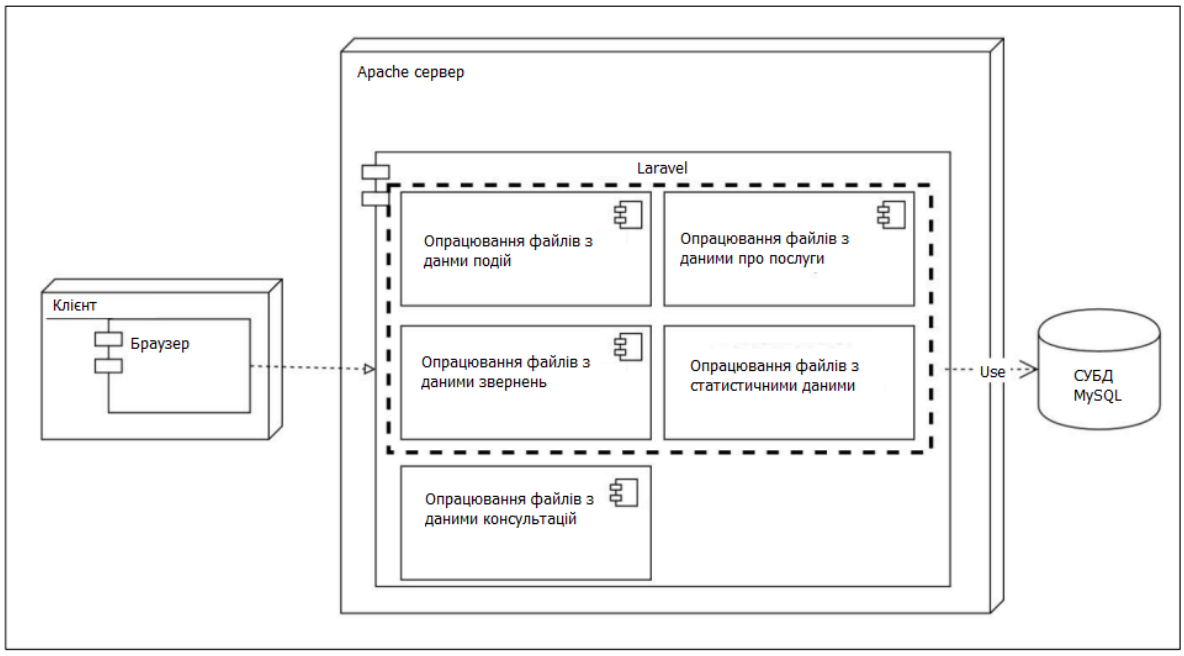

Рис. 2.1. Діаграма компонентів системи

Шар подання (клієнтська частина). Цей рівень відповідає за відображення користувацького інтерфейсу та обробку взаємодій з користувачем.

Шар бізнес-логіки (сервер Apache). На цьому рівні відбувається обробка запитів від клієнтської частини, виконання логіки додатку та взаємодія з базою

даних. Бізнес-логіка реалізується за допомогою серверних мов програмування. У цьому прикладі використовується мова PHP з фреймворком Laravel.

Шар зберігання даних (СУБД MySQL): Цей компонент відповідає за збереження всіх даних, необхідних для роботи додатку. Шар зберігання даних забезпечує надійне та ефективне управління інформацією, з якою працює додаток.

Компонент обробки файлів з даними подій та компонент обробки файлів з даними звернень мають однакові функціональні вимоги.

Функціональні вимоги. Розглянемо функціональні вимоги до цих компонентів:

Компонент повинен надавати можливість завантаження даних з Excel-файла в базу даних, відповідна діаграма послідовності представлена на рисунку 2.2.

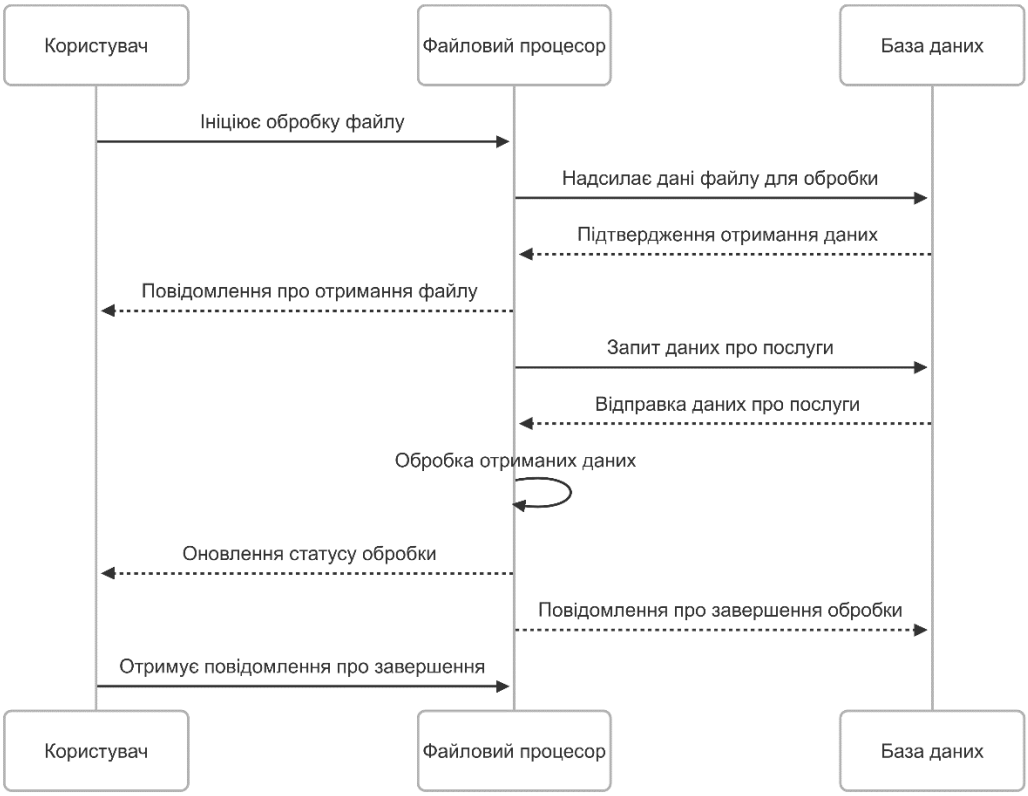

Рис. 2.2. Діаграма послідовності взаємодій компонента обробки файлів з даними про послуги з базою даних

Компонент повинен надавати візуалізацію даних з бази даних у вигляді таблиці.

Компонент повинен надавати можливість фільтрації даних у таблиці.

Компонент повинен надавати довідку по видам фільтрів таблиці.

Компонент повинен надавати можливість скачати звіт у вигляді Excel-файла, складеного за певними правилами, як тільки за відфільтрованими даними, так і за всіма. Відповідна діаграма послідовності для реалізації даного функціоналу представлена на рисунку 2.3.

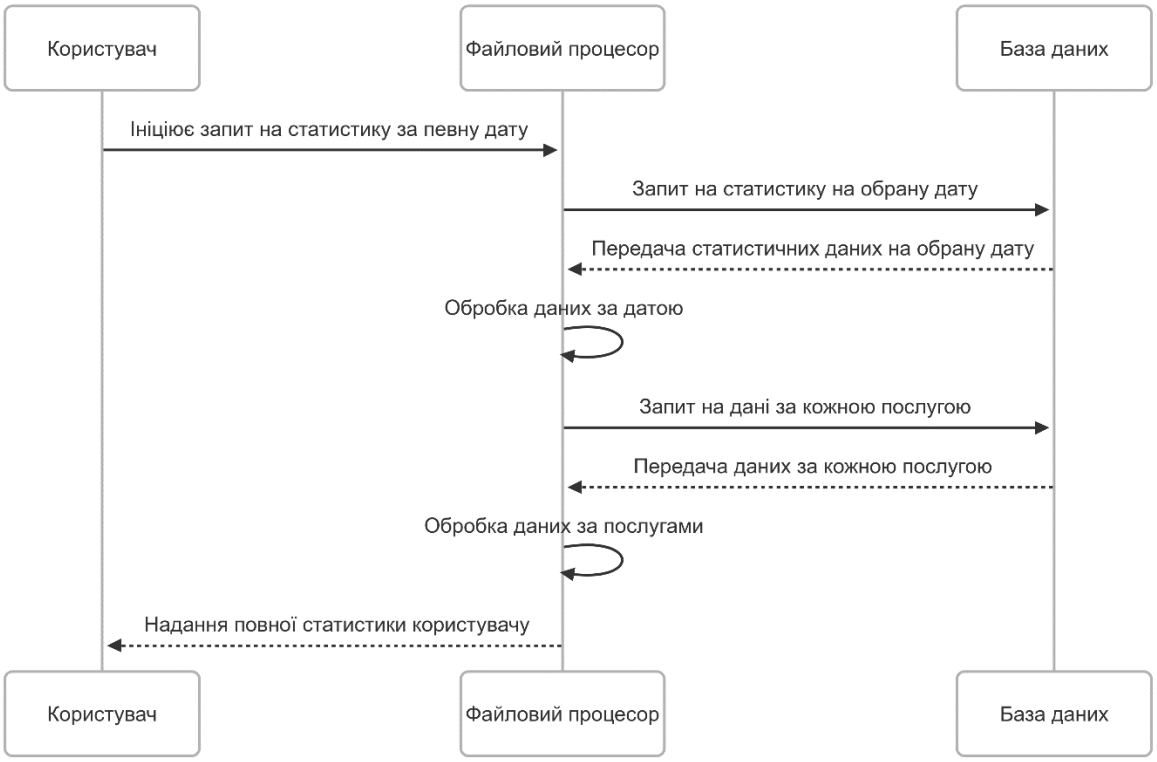

Рис. 2.3. Діаграма послідовності взаємодій компонента обробки файлів зі статистикою щодо частки звернень з базою даних

Компонент повинен надавати можливість експортувати як усі дані з бази даних, так і тільки відфільтровані.

2.2. Проектування бази даних

При проектуванні структури підсистеми зберігання інформації використано компонентний підхід. Розглянемо ці процеси більш детально та поетапно.

Компонент обробки файлів з даними про події. Цей компонент відповідає за обробку Excel-файлів з даними про планові та позапланові події, що стосуються різних сфер ЖКГ. Дані файли мають певну незмінну структуру у вигляді таблиці на 63 стовпці.

При імпорті вмісту файлу в базу даних необхідно перевіряти наявність дублюючихся значень по стовпцю «Ідентифікатор події», в базі даних це поле event_id. Дані записуються в таблицю for_omas_control. У цій таблиці запис про подію розділений на безліч полів, таких як дата і час виникнення події, суб'єкт і район, де відбулася подія, а також короткий опис події і багато її характеристик, наприклад, сфера, тип інциденту, найменування об'єкта, адреса об'єкта, дата та час усунення. Поля містять різні типи даних, такі як текст, числа, дати, що необхідно враховувати при реалізації фільтрів для таблиці.

На рисунку 2.4. представлено фрагмент структури таблиці бази даних з полями, які зберігають дані про місце події для компонента обробки файлів з даними про події.

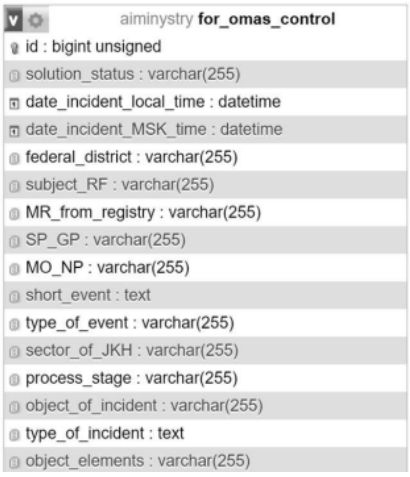

Рис. 2.4. Фрагмент структури таблиці бази даних з полями, які зберігають дані про місце події

Компонент обробки файлів з даними про звернення. Цей компонент відповідає за обробку Excel-файлів з даними про звернення або інциденти. Дані файли мають певну незмінну структуру у вигляді таблиці на 41 стовпець. Дані для цієї таблиці знаходяться в соціальних мережах у вигляді записів або коментарів користувачів, люди, що займаються моніторингом соціальних мереж, формують цю таблицю.

При імпорті вмісту файлу в базу даних необхідно перевіряти наявність дублюючихся значень по стовпцю «Номер інциденту», в базі даних це поле local_id. Дані записуються в таблицю tsur_detaling. У цій таблиці запис про подію розділений на безліч полів, таких як регіон, виконавець, загальний час першої відповіді, час з дати створення звернення до часу першої відповіді, тип інциденту, наявність медіафайлів у повідомленні, URL автора. Поля містять різні типи даних, такі як текст, числа, дати, що необхідно враховувати при реалізації фільтрів для таблиці.

На рисунку 2.5 представлено фрагмент структури таблиці бази даних з полями, що містять інформацію про виконавця, який реагує на звернення і деякі його дії, для компонента обробки файлів з даними про звернення.

Рис. 2.5 . Фрагмент структури таблиці бази даних з полями, які зберігають дані про виконавців

Компонент обробки файлів з даними про послуги. Цей компонент відповідає за обробку Excel-файлів з даними про послуги, які надає компанія. Дані файли мають певну незмінну структуру у вигляді кількох листів: «загальний», «оцінка якості по місяцях», «подані заяви по місяцях», «відмови по заявах по місяцях», «позитивні по місяцях». На кожному листі знаходиться певна таблиця.

На листі «загальний» знаходиться весь список послуг з посиланнями на ці послуги, а також загальна інформація з усіх інших листів: загальна кількість поданих заяв по роках, загальна кількість відмов по роках, загальна кількість чернеток по роках, загальна кількість помилок по роках, загальна оцінка, підсумкові значення.

На листі «оцінка якості по місяцях» знаходиться весь список послуг з посиланнями на ці послуги, а також всі оцінки, розбиті по роках і місяцях.

На листі «подані заяви по місяцях» знаходиться весь список послуг з посиланнями на ці послуги, а також загальна кількість заяв на послуги, розбита по місяцях і роках.

На листі «відмови по заявах по місяцях» знаходиться весь список послуг з посиланнями на ці послуги, а також загальна кількість відмов по заявах на послуги, розбита по місяцях і роках.

На листі «позитивні по місяцях» знаходиться весь список послуг з посиланнями на ці послуги, а також загальна кількість позитивних заявок на послуги, розбита по місяцях і роках.

Даний компонент повинен імпортувати дані з файлу в певні таблиці бази даних:

1. Таблиця service. Ця таблиця містить повний список послуг і має поля id, назва, посилання на госпослуги, в неї імпортується інформація з листа «загальний».

2. Таблиця quality_control_by_month. В цю таблицю імпортується інформація з листа «оцінка якості по місяцях», тобто записується оцінка якості за кожен місяць кожного року по кожній послузі, ця таблиця має поля id, service_id для зв'язку з таблицею service, date і grade.

3. Таблиця statements_by_month. Ця таблиця має поля id, service_id для зв'язку з таблицею service, date, data_quantity, number_of_failures, number_of_positive, в неї імпортуються дані з листів «подані заяви по місяцях», «відмови по заявах по місяцях», «позитивні по місяцях».

На рисунку 2.6 представлена структура таблиць для компонента обробки файлів з даними про послуги.

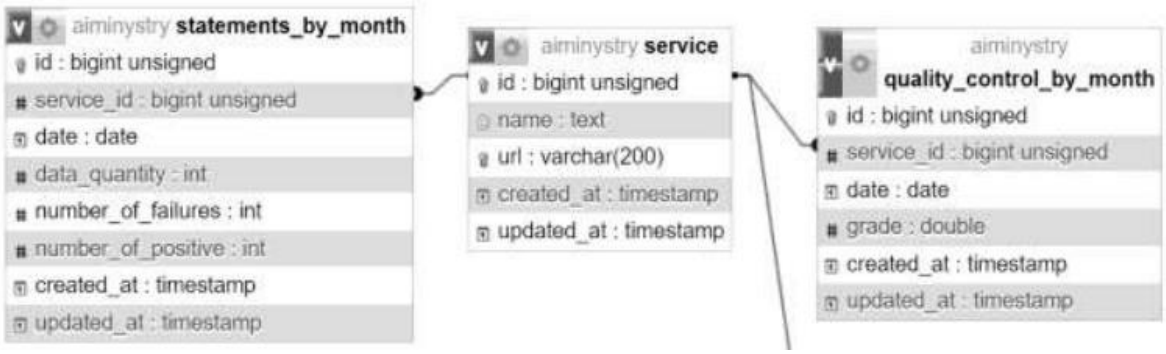

Рис. 2.6. Структура таблиць для компонента обробки файлів з даними про послуги.

Компонент обробки файлів зі статистикою по долі звернень. Цей компонент відповідає за обробку Excel-файлів зі статистичними даними по долі звернень по наданню послуг. Дані файли мають певну незмінну структуру у вигляді таблиці на 299 стовпців. Для обробки даних з файлу в цій таблиці зазначені найменування послуги, сфера надання послуги, а також кількість заявок, поданих в електронному вигляді, загальна кількість заявок і відсоток електронних заявок від загальної кількості.

Цей компонент повинен надавати можливість:

1. Імпортувати дані з файлу в певні таблиці бази даних.
2. Вибір місяця та року для завантаження файлу.

Цей компонент використовує таблицю service з компонента обробки файлів з даними про послуги, а також таблиці area та appeals_by_months. Таблиця area має повний список районів і округів, імпортованих з завантаженого файлу, з полями id та name.

У таблиці appeals_by_months містяться дані про всі звернення по кожному округу чи району за кожен рік і місяць, включаючи кількість електронних заявок, загальну кількість заявок і частку електронних заявок від загальних.

На рисунку 2.7 представлена структура таблиць для компонента обробки файлів зі статистикою по долі звернень, а також відображена зв'язок між таблицями.

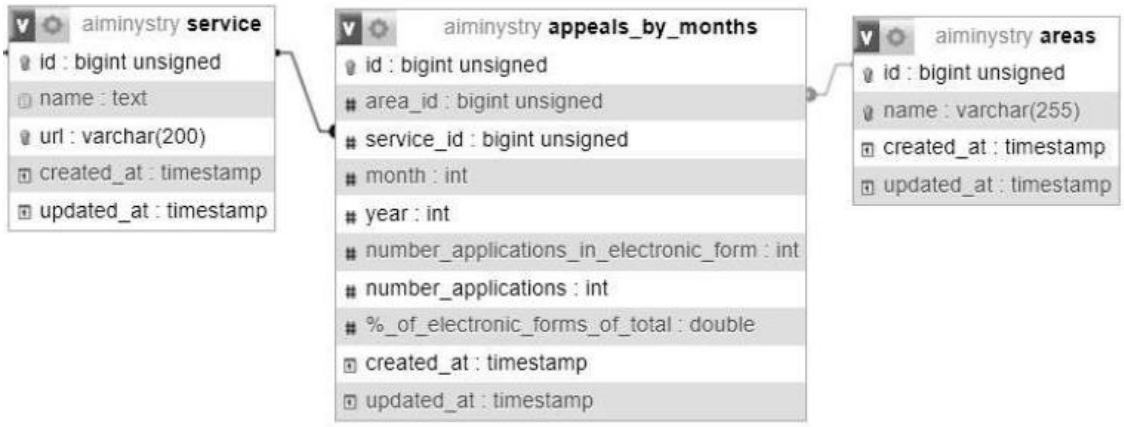

Рис. 2.7. Структура таблиць для компонента обробки файлів зі статистикою

На початковому етапі користувач завантажує файл, вказуючи бажаний місяць і рік. Потім система імпортує дані про всі регіони, що містяться в файлі, в таблицю під назвою area. Також система отримує зі файлу список послуг за сферами «майно, будівництво» та «будівництво» і з бази даних інформацію про послуги, що зберігаються в таблиці service. Далі відбувається перевірка вказаного місяця, і якщо це січень, дані безпосередньо зберігаються в базу даних для кожної послуги, району і вказаного року для січня. Якщо це не січень, система витягує дані з бази даних для відповідного місяця та року, після чого віднімає їх від імпортованих значень. Отримані результати віднімання потім записуються назад в базу даних.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ ДЛЯ ПАРСИНГУ ДОКУМЕНТІВ

3.1. Вибір засобів реалізації програмного модуля

Реалізація проводилась з використанням мови програмування PHP [1]. Це скриптова мова програмування загального призначення, яка часто використовується для розробки веб-додатків та веб-сайтів. PHP зазвичай інтегрується безпосередньо в HTML [4] і може бути вбудований у HTML-код або використовуватись у поєднанні з різними веб-фреймворками, системами управління контентом та іншими інструментами для створення динамічних веб-додатків. PHP також підтримує велику кількість баз даних.

Також був використаний Laravel [3], популярний фреймворк для створення веб-додатків на мові програмування PHP. Він надає розробникам безліч готових інструментів і бібліотек для спрощення процесу створення веб-додатків, включаючи зручну маршрутизацію, роботу з базами даних, аутентифікацію користувачів і багато іншого. Laravel також забезпечує підтримку сучасних практик розробки, таких як MVC (Model-View-Controller).

Системи управління базами даних MySQL [6] — одна з найпопулярніших систем управління реляційними базами даних (СУБД) у світі. MySQL підтримує безліч функцій, включаючи можливість зберігання великих обсягів інформації, а також має високу продуктивність і надійність.

Веб-додаток phpMyAdmin [7] — безкоштовний інструмент з відкритим вихідним кодом, призначений для адміністрування баз даних MySQL через веб-інтерфейс. Він надає зручні засоби для створення, видалення та зміни баз даних, таблиць, полів, а також для виконання SQL-запитів, імпорту та експорту даних і багатьох інших операцій.

У фреймворку Laravel реалізовано патерн MVC (Model-View-Controller) для структурування додатків.

Модель (Model). Моделі в Laravel представляють собою шар даних, який відповідає за роботу з базою даних. Кожна модель зазвичай пов'язана з певною

таблицею в базі даних і надає спосіб взаємодії з цими даними. Моделі також містять бізнес-логіку додатку, валідацію даних та інші методи, що виконують маніпуляції з даними.

Представлення (View). Представлення в Laravel представляють собою шаблони, які відповідають за відображення даних користувачу. Вони можуть бути у вигляді HTML-сторінок, JSON-відповідей або інших типів даних, які повертаються користувачу. Laravel надає потужні інструменти для роботи з представленнями, включаючи шаблонізацію, роботу з макетами і можливість передачі даних з контролера в представлення.

Контролер (Controller). Контролери в Laravel обробляють вхідні HTTP-запити і взаємодіють як з моделями, так і з представленнями. Вони приймають запити від клієнта, звертаються до відповідної моделі для отримання необхідних даних, а потім передають ці дані в представлення для відображення. Контролери також можуть обробляти вхідні дані від користувача, виконувати необхідні операції та повертати результат користувачу.

Одним з головних елементів фреймворку Laravel є маршрутизація [14]. У Laravel маршрутизація є механізмом визначення того, як додаток обробляє вхідні HTTP-запити. Маршрути забезпечують централізовану систему маршрутизації запитів у додатку, дозволяючи визначити, які дії повинні бути виконані в залежності від URL-адреси, з якої надходить запит. Механізм маршрутизації в Laravel забезпечує структуроване управління потоком HTTP-запитів і підвищує ефективність обробки запитів веб-додатком. Код файлу маршрутизації представлений у лістингу додатка А.

Макети в Laravel використовуються для створення загальної структури HTML-сторінки, що полегшує управління та обслуговування коду. Макет дозволяє визначити основний шаблон сторінки, включаючи елементи, які повторюються на всіх сторінках, також у основному макеті підключаються JavaScript-бібліотеки, шрифти, CSS-стилі. Фрагмент коду основного макета представлений у додатка А.

3.2. Програмна реалізація модуля

Компонент обробки файлів з даними про події. При реалізації компонента в першу чергу був створений клас моделі Event і міграція для створення таблиці for_omas_control в базі даних. Фрагмент коду файлу міграції представлений в додатку А. Також був створений клас контролера EventController. Цей клас має кілька ключових методів. Метод import(Request $request) імпортує дані з файлу Excel та оновлює базу даних. Код цього методу представлений на рисунку 3.1.

```php
public function import(Request $request)
{
    $request->validate([
        'file' => 'required|file|mimes:xlsx,xls'
    ], [
        'file.required' => 'Виберіть файл для завантаження',
        'file.mimes' => 'Вибраний файл повинен бути формату Excel (XLSX, XLS)'
    ]);

    $file = $request->file("file");
    if($file->isValid())
    {
        Excel::import(new EventImport, $file);
        return redirect()->back()->with('success', 'Все добре');
    } else {
        return redirect()->back()->withErrors(['file' => 'Вибраний файл некоректний']);
    }
}
```

Рис. 3.1. Код методу import() класу EventController

Метод export() дозволяє скачати звіт, складений за певними правилами, використовуючи всі дані з таблиці бази даних. Метод exportFilteredData(Request $request) дозволяє скачати звіт, складений за певними правилами, з відфільтрованих даних, на основі переданих фільтрів. Код цього методу представлений на рисунку 3.2.

```php
public function exportFilteredData(Request $request)
{
    $filters = $request->input('filters', []);
    $query = Event::select('MR_from_registry',
        'MO_NP',
        'address_of_object',
        'sector_of_JKH',
        'type_of_event',
        'short_event',
        'name_of_expl_org',
        'date_incident_local_time',
        'date_incident_elimination_local_time',
        'solution_status');
    $filters = json_decode($filters, true);
    $filters = is_array($filters) ? $filters : [];
    foreach ($filters as $filter) {
        $field = $filter['field'];
        $type = $filter['type'];
        $value = $filter['value'];
        switch ($type) {
            case 'like':
                $query->where($field, 'like', '%' . $value . '%');
                break;
            case '=':
            case '<':
            case '<=':
            case '>':
            case '>=':
            case '!=':
                $query->where($field, $type, $value);
                break;
            default:
```

Рис. 3.2. Код методу exportFilteredData() класу EventController

Метод viewDataTable() відображає таблицю подій на веб-сторінці, завантажуючи дані з бази даних. Код цього методу представлений на рисунку 3.3.

```php
public function viewDataTable()
{
    $events = Event::all();
    $json = file_get_contents(public_path('data/events.json'));
    $columns = json_decode($json, true);
    return view("/events/events", ['events' => $events, 'columns' => $columns]);
}
```

Рис. 3.3. Код методу viewDataTable() класу EventController

Метод import() використовує клас EventImport, який імпортує інформацію з файлу в таблицю бази даних. Фрагмент класу імпорту EventImport представлений на рисунку 3.4.

```php
class EventImport implements ToModel, WithValidation, WithHeadingRow {
    public function model(array $row){
        $row = $this->transformValues($row);
        return new Event([
            'solution_status' => $row['Статус рішення інцидента/аварії'],
            'date_incident_local_time' => $row['Дата і місцевий час виникнення події'],
            // Інші поля
        ]);
    }

    public function rules(): array {
        return ['event_id'=> 'unique:for_omas_control'];
    }

    public function transformValues(array $row){
        if (!empty($row['Дата і місцевий час виникнення події'])){
            $row['Дата і місцевий час виникнення події'] = DateTime::createFromFormat('d.m.Y H:i
        }
        return $row;
    }
}
```

Рис. 3.4. Код класу EventImport

Методи export() та exportFilteredData(Request $request) використовують клас EventExport для формування та відправки звіту користувачеві. Звіт формуються з правилами, описаними при проектуванні компонента. Фрагмент коду класу експорту EventExport представлений в додатках.

Частина функцій компонента, таких як фільтрація даних в таблиці, відображення довідки по фільтрації, генерація таблиці, реалізована з допомогою JavaScript [5] коду та прописана в файлі представлення. Були використані такі бібліотеки, як Tabulator.js [12] для відображення таблиці та реалізації фільтрів у заголовках таблиці, DataRangePicker.js [13] для коректної запису дати в фільтри, які знаходяться в заголовках таблиці.

Компонент обробки файлів з даними про звернення. Цей компонент схожий на попередній, міграція, модель, представлення, контролер, файл з класом імпорту мають схожу структуру. Відмінності є при формуванні звіту та його відправці користувачеві. Логіка обробки даних та формування звіту представлена в AppealExport. Основна різниця — у структурі завантаженого файлу та в структурі сформованого звіту.

Цей клас реалізує кілька інтерфейсів, надаючи спеціальні функції для форматування та структуризації даних для експорту. Клас містить методи для формування заголовків, визначення стилів, ширини стовпців та маппінгу даних для відображення в таблиці.

Компонент обробки файлів з даними про послуги. Оскільки цьому компоненту необхідно 3 таблиці в базі даних, то були створені відповідні моделі та міграції для створення таблиць у базі даних. Модель Service прив'язана до таблиці service, модель QualityControlByMonth до таблиці quality_control_by_month, а модель StatementsByMonth до таблиці statements_by_month. Окрім зв'язку моделі з таблицею, необхідно прописати відносини моделі Service з іншими моделями, щоб через певну послугу можна було звертатися до її оцінки якості по місяцях та до кількості поданих, відмовлених та позитивних звернень. Код класу моделі Service представлений на рисунку 3.5.

```php
class Service extends Model
{
    use HasFactory;
    protected $table = 'service';
    protected $guarded = ['id'];

    public function statement(): HasMany
    {
        return $this->hasMany(StatementsByMonth::class);
    }

    public function quality(): HasMany
    {
        return $this->hasMany(QualityControlByMonth::class);
    }

    public function appealByMonths(): HasMany
    {
        return $this->hasMany(AppealByMonths::class);
    }
}
```

Рис. 3.5. Код класу моделі Service

У цьому коді видно, що модель Service має відносини один до багатьох з моделями StatementsByMonth та QualityControlByMonth. Оскільки значення для певної послуги в таблицях quality_control_by_month та statements_by_month

зберігаються по місяцях, то для однієї послуги може бути більше однієї зв'язаної записи в цих таблицях. Відповідно в моделях, що зв'язуються з іншими таблицями, прописана зворотна залежність через конструкцію BelongsTo.

Також був створений контролер ServiceController та прописаний метод import(Request $request), в якому прописана логіка імпорту даних із завантаженого файлу. Код методу import() класу ServiceController представлений на рисунку 3.6.

```php
public function import(Request $request)
{
    $request->validate([
        'file' => 'required|file|mimes:xlsx,xls'
    ], [
        'file.required' => 'Виберіть файл для завантаження',
        'file.mimes' => 'Вибраний файл повинен бути формату Excel (XLSX, XLS)'
    ]);

    $file = $request->file("file");
    if($file->isValid())
    {
        Excel::import(new ServiceImport, $file);
        Excel::import(new QualityControlByMonthImport, $file);
        Excel::import(new StatementsByMonthImport, $file);
        return redirect()->back()->with('success', 'Все добре');
    } else {
        return redirect()->back()->withErrors(['file' => 'Вибраний файл некоректний']);
    }
}
```

Рис. 3.6. Код методу import() класу ServiceController

Цей метод використовує 3 класи імпорту: ServiceImport, QualityControlByMonthImport і StatementsByMonthImport.

Вони використовуються в певній послідовності, відповідно до діаграми послідовності. Першим використовується клас ServiceImport, який імпортує дані про послуги з завантаженого Excel-файла з листа «загальний» в таблицю service в базі даних, записуючи ім'я послуги та посилання на неї на сайт державних послуг, унікальність перевіряється за посиланням. Код класу ServiceImport представлений на рисунку 3.7.

```php
class ServiceImport implements ToModel, WithHeadingRow, WithMultipleSheets, WithValidation
{
    use Importable;
    private $rowCount = 0;
    public static $fieldsIndexes = [
        'LEVEL' => 'Уровень',
        'URL' => 'Форма'
    ];

    public function model(array $row)
    {
        $existingRecord = Service::where('url', $row[self::$fieldsIndexes['URL']])->first();
        if(!empty($row[self::$fieldsIndexes['URL']]) && !$existingRecord) {
            return new Service([
                'name' => $row[self::$fieldsIndexes['LEVEL']],
                'url' => $row[self::$fieldsIndexes['URL']]
            ]);
        }
    }

    public function rules(): array {
        return ['url' => 'unique:service'];
    }

    public function sheets(): array {
        return ['общий' => $this];
    }
```

Рис. 3.7. Код класу ServiceImport

Другим використовується клас імпорту QualityControlByMonthImport, який з завантаженого Excel-файла з листа "оцінка якості по місяцях" зчитує послугу, перевіряє її за посиланням в таблиці service, і якщо така послуга є, записує оцінку якості по кожному місяцю в таблицю quality_control_by_month.

```php
class QualityControlByMonthImport implements ToCollection, WithMultipleSheets
{
    public function collection(Collection $rows)
    {
        $startReadingRow = 8;
        $startReadingItem = 6;
        $headingRowNumber = 7;
        $serviceIdQualifier = 5;

        for ($row = $startReadingRow; $row <= $rows->count()-2; $row++) {
            for ($item = $startReadingItem; $item <= $rows[$row]->count()-1; $item++) {
                $service = Service::where('url', $rows[$row][$serviceIdQualifier])->first();
                if ($rows[$headingRowNumber][$item] != null) {
                    $date = DateTime::createFromFormat('Y-m-d', $rows[$headingRowNumber][$item]);
                    QualityControlByMonth::updateOrCreate(
                        ['service_id' => $service->id, 'date' => $date->format('Y-m-d')],
                        ['grade' => $rows[$row][$item]]
                    );
                }
            }
        }
    }

    public function sheets(): array {
        return ['оцінка якості по місяцях' => $this];
    }
}
```

Рис. 3.8. Код класу QualityControlByMonthImport

Третім використовується імпорт класу StatementsByMonthImport. Код імпорту класу StatementsByMonthImport представлений на рисунку 3.9.

```php
class StatementsByMonthImport implements WithMultipleSheets
{
    public function sheets(): array
    {
        return [
            'подані заяви по місяцях' => new StatementsSubmittedByMonthImport(),
            'відмови по заявах по місяцях' => new StatementsNegativeByMonthImport(),
            'позитивні по місяцях' => new StatementsPositiveByMonthImport(),
        ];
    }
}
```

Рис. 3.9. Код імпорту класу StatementsByMonthImport

Цей клас використовується для імпорту даних з кількох листів завантаженої Excel-таблиці. У методі sheets() визначені відповідності між назвами листів у

таблиці та класами, які відповідають за імпорт даних з цих листів. Оскільки в таблицю statements_by_month необхідно записати інформацію з кількох листів завантаженої Excel-таблиці, на кожен з цих листів призначений клас імпорту.

```php
class StatementsSubmittedByMonthImport implements ToCollection
{
    public function collection(Collection $rows)
    {
        $startReadingRow = 8;
        $startReadingItem = 6;
        $headingRowNumber = 7;
        $serviceIdQualifier = 5;

        for ($row = $startReadingRow; $row <= $rows->count()-2; $row++) {
            for ($item = $startReadingItem; $item <= $rows[$row]->count()-1; $item++) {
                $service = Service::where('url', $rows[$row][$serviceIdQualifier])->first();
                if ($rows[$headingRowNumber][$item] != null) {
                    $date = DateTime::createFromFormat('Y-m-d', $rows[$headingRowNumber][$item]);
                    StatementsByMonth::updateOrCreate(
                        ['service_id' => $service->id, 'date' => $date->format('Y-m-d')],
                        ['data_quantity' => $rows[$row][$item]]
                    );
                }
            }
        }
    }
}
```

Рис. 3.10. Код класу StatementsSubmittedByMonthImport

Цей клас зчитує з завантаженого Excel-файла з листа "оцінка якості по місяцях" послугу, перевіряє за посиланням, чи є така послуга в таблиці service, і якщо така є, записує кількість поданих заявок по місяцях в таблицю statements_by_month.

Компонент обробки файлів зі статистикою по долі звернень. Цьому компоненту необхідно 3 таблиці в базі даних. Модель Service та таблиця service були описані у реалізації компонента обробки файлів з даними про послуги. Також були створені ще 2 моделі та міграції для створення таблиць у базі даних. Модель Area прив'язана до таблиці areas, а модель AppealByMonths до таблиці appeals_by_months. Модель Area, так само як і модель Service, зв'язана з моделлю AppealByMonths відносинами один до багатьох, оскільки для однієї конкретної послуги в одному конкретному районі може бути різна кількість звернень по різних

місяцях та роках. Код класу моделі AppealByMonths представлений на рисунку 3.11.

```php
class AppealByMonths extends Model
{
    use HasFactory;
    protected $table = 'appeals_by_months';
    protected $guarded = ['id'];

    public function service(): BelongsTo
    {
        return $this->belongsTo(Service::class);
    }

    public function area(): BelongsTo
    {
        return $this->belongsTo(Area::class);
    }
}
```

Рис. 3.11. Код класу моделі AppealByMonths

У конкретної послуги може бути кілька звернень за різні місяці, роки, у різних районах. Також у певному районі може бути багато звернень по різних послугах, місяцях і роках. Це зв'язок побудовано так, щоб при зверненні до певної послуги чи району можна було отримати статистику за певний місяць і рік.

Був створений клас контролера AppealByMonthsController та реалізований метод import(Request $request), у якому прописана логіка імпорту даних із завантаженого файлу. Код методу import() класу AppealByMonthsController представлений на рисунку 3.12.

У цьому методі додано валідацію даних, які надходять з клієнтської частини. Окрім валідації розширення та наявності файлу, яка є в усіх компонентах, також додана валідація на коректність місяця та року. Метод import() використовує два класи імпорту: AreaImport та AppealByMonthsImport. Вони використовуються в певній послідовності, згідно з діаграмою послідовності. Вважається, що при використанні цього компонента база даних вже має список послуг у таблиці service.

```php
public function import(Request $request)
{
    $request->validate([
        'file' => 'required|file|mimes:xlsx,xls',
        'month' => 'required|integer|between:1,12',
        'year' => 'required|integer|between:2000,3000'
    ], [
        'file.required' => 'Виберіть файл для завантаження',
        'file.mimes' => 'Вибраний файл повинен бути формату Excel (XLSX, XLS)',
        'month.required' => 'Виберіть місяць',
        'month.integer' => 'Значення місяця повинно бути числом',
        'month.between' => 'Значення місяця повинно бути від 1 до 12',
        'year.integer' => 'Значення року повинно бути числом',
        'year.between' => 'Значення року повинно бути від 2000 до 3000'
    ]);

    $file = $request->file("file");
    $month = intval($request->input('month'));
    $year = intval($request->input('year'));

    if ($file->isValid()) {
        Excel::import(new AreaImport, $file);
        Excel::import(new AppealByMonthsImport($month, $year), $file);
        return redirect()->back()->with('success', 'Все добре');
    } else {
        return redirect()->back()->withErrors(['file' => 'Вибраний файл некоректний']);
    }
}
```

Рис. 3.12. Код методу import() класу AppealByMonthsController

Першим використовується імпорт класу AreaImport, який проходить по осередках завантаженого Excel-файла до осередку зі значенням «Ітогове значення», зчитує всі райони та додає їх до таблиці areas в базі даних. Код класу імпорту AreaImport представлений на рисунку 3.13.

```php
class AreaImport implements ToCollection
{
    public function collection(Collection $rows)
    {
        $areasRow = 0;
        $areaStepInRow = 5;
        $startReadingItem = 5;
```

Рис. 3.13. Код імпорту класу AreaImport

На рисунку 3.14. представлено фрагмент діаграми класів для даного програмного модуля.

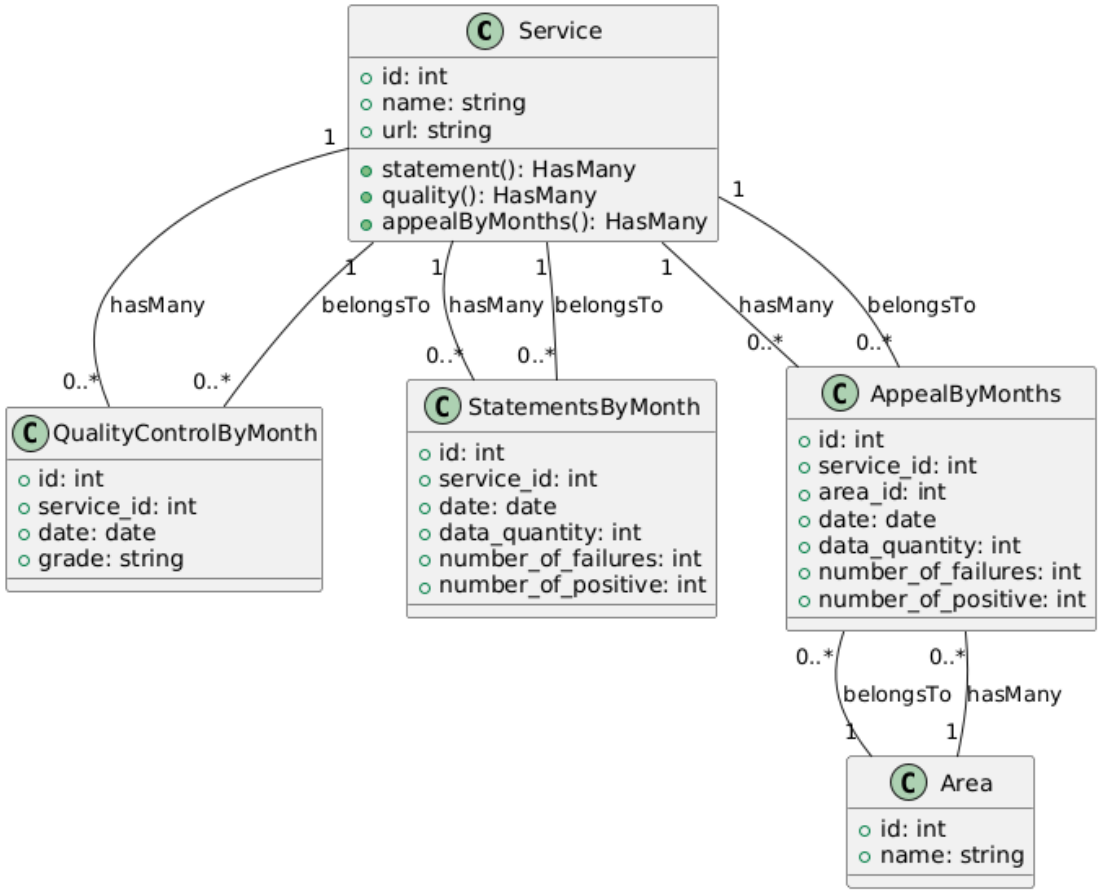

Рис. 3.14. Діаграма класів

Другим використовується імпорт класу AppealByMonthsImport, який заповнює таблицю appeals_by_months даними з завантаженого Excel-файла. В екземпляр цього класу передаються місяць і рік, які ввів користувач при заповненні форми. Далі скрипт починає шукати послуги зі сфер «майно, будівництво» і «будівництво», які є в таблиці service, після чого в циклі з таблиці areas скрипт отримує район. Потім відбувається перевірка на зазначений місяць, і якщо це січень, дані безпосередньо зберігаються в таблицю appeals_by_months, тобто для цієї послуги, всіх районів, січня та зазначеного року створюються записи в таблиці. Якщо місяць не січень, система з допомогою SQL-запиту витягує дані з таблиці за попередні місяці для відповідного року, після чого кожного разу віднімає їх від імпортованих значень. Таким чином скрипт виконується за допомогою подвійного циклу: основний перебирає послуги, а вкладений перебирає райони і розраховує значення звернень, якщо обраний місяць не січень, а потім записує всі дані в

таблицю appeals_by_months, перевіряючи унікальність запису за полями: послуга, район, місяць, рік. Отримані результати віднімання потім записуються в базу даних.

Скрипт забезпечує динамічне оновлення даних залежно від введених користувачем параметрів, коректно враховуючи попередні періоди та забезпечуючи унікальність записів у таблиці appeals_by_months.

3.3. Програмна реалізація бази даних

Програмна реалізація бази даних для цього проекту потребує створення таблиць та визначення зв'язків між ними в системі керування базами даних (СУБД). Оскільки в проекті використовується MySQL, надам приклад SQL-скриптів для створення таблиць, міграцій, та реалізації зв'язків між ними.

Створення таблиць. Таблиця service, Рис. 3.15.

```
CREATE TABLE service (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    url VARCHAR(255) UNIQUE NOT NULL
);
```

Рис. 3.15. SQL таблиці service

Таблиця area, Рис. 3.16.

```
CREATE TABLE area (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL
);
```

Рис. 3.16. SQL таблиці area

Таблиця appeals_by_months, Рис. 3.17.

```sql
CREATE TABLE appeals_by_months (
    id INT AUTO_INCREMENT PRIMARY KEY,
    service_id INT NOT NULL,
    area_id INT NOT NULL,
    date DATE NOT NULL,
    data_quantity INT,
    number_of_failures INT,
    number_of_positive INT,
    FOREIGN KEY (service_id) REFERENCES service(id),
    FOREIGN KEY (area_id) REFERENCES area(id),
    UNIQUE(service_id, area_id, date)
);
```

Рис. 3.17. SQL таблиці appeals_by_months

Таблиця quality_control_by_month, Рис. 3.18.

```sql
CREATE TABLE quality_control_by_month (
    id INT AUTO_INCREMENT PRIMARY KEY,
    service_id INT NOT NULL,
    date DATE NOT NULL,
    grade VARCHAR(50),
    FOREIGN KEY (service_id) REFERENCES service(id),
    UNIQUE(service_id, date)
);
```

Рис. 3.18. SQL таблиці quality_control_by_month

Таблиця statements_by_month, Рис. 3.19.

```sql
CREATE TABLE statements_by_month (
    id INT AUTO_INCREMENT PRIMARY KEY,
    service_id INT NOT NULL,
    date DATE NOT NULL,
    data_quantity INT,
    number_of_failures INT,
    number_of_positive INT,
    FOREIGN KEY (service_id) REFERENCES service(id),
    UNIQUE(service_id, date)
);
```

Рис. 3.19. SQL таблиці statements_by_month

Міграції в Laravel. У Laravel для створення таблиць використовуються міграції. Ось приклади міграцій для таблиць, згаданих вище:

Міграція для таблиці service, Рис. 3.20.

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateServiceTable extends Migration
{
    public function up()
    {
        Schema::create('service', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('url')->unique();
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('service');
    }
}
```

Рис. 3.20. Міграція для таблиці service

Міграція для таблиці area, Рис. 3.21.

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateAreaTable extends Migration
{
    public function up()
    {
        Schema::create('area', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('area');
    }
}
```

Рис. 3.21. Міграція для таблиці area

Міграція для таблиці appeals_by_months, Рис. 3.22.

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateAppealsByMonthsTable extends Migration
{
    public function up()
    {
        Schema::create('appeals_by_months', function (Blueprint $table) {
            $table->id();
            $table->foreignId('service_id')->constrained('service');
            $table->foreignId('area_id')->constrained('area');
            $table->date('date');
            $table->integer('data_quantity')->nullable();
            $table->integer('number_of_failures')->nullable();
            $table->integer('number_of_positive')->nullable();
            $table->unique(['service_id', 'area_id', 'date']);
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('appeals_by_months');
    }
}
```

Рис. 3.22. Міграція для таблиці appeals_by_months

Міграція для таблиці quality_control_by_month, Рис. 3.23.

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateQualityControlByMonthTable extends Migration
{
    public function up()
    {
        Schema::create('quality_control_by_month', function (Blueprint $table) {
            $table->id();
            $table->foreignId('service_id')->constrained('service');
            $table->date('date');
            $table->string('grade')->nullable();
            $table->unique(['service_id', 'date']);
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('quality_control_by_month');
    }
}
```

Рис. 3.23. Міграція для таблиці quality_control_by_month

Міграція для таблиці statements_by_month, Рис. 3.24.

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateStatementsByMonthTable extends Migration
{
    public function up()
    {
        Schema::create('statements_by_month', function (Blueprint $table) {
            $table->id();
            $table->foreignId('service_id')->constrained('service');
            $table->date('date');
            $table->integer('data_quantity')->nullable();
            $table->integer('number_of_failures')->nullable();
            $table->integer('number_of_positive')->nullable();
            $table->unique(['service_id', 'date']);
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('statements_by_month');
    }
}
```

Рис. 3.24. Міграція для таблиці statements_by_month

Ці SQL-скрипти та міграції для Laravel створюють базу даних з необхідними таблицями та зв'язками між ними, відповідно до вимог проекту. Вони охоплюють таблиці для зберігання даних про послуги, райони, статистику звернень, оцінку якості та заяви.

3.4. Тестування модуля

Для тестування компонентів використовувалося функціональне тестування. Функціональне тестування — це процес перевірки програмного забезпечення для оцінки виконання функціональних вимог, тобто здатності програмного забезпечення в певних умовах вирішувати задачі, необхідні користувачам [17].

Функціональні вимоги визначають, що саме робить програмне забезпечення, які завдання воно вирішує.

Компонент обробки файлів з даними про події
Набір тестів на функціональність компонента обробки файлів з даними про події представлений у таблиці 3.1.

Таблиця 3.1. Тестування компонента обробки файлів з даними про події

№	Аспект роботи	Дія	Результат	Тест пройдено?
1	Завантаження xlsx файлу для компонента	1. Натиснути кнопку «Вибрати файл».	Дані з файлу завантажаться в базу даних і додадуться в таблицю	Так
		2. Вибрати xlsx файл для компонента.		
		3. Натиснути кнопку «Завантажити».		
2	Завантаження xls файлу для компонента	1. Натиснути кнопку «Вибрати файл».	Дані з файлу завантажаться в базу даних і додадуться в таблицю	Так
		2. Вибрати xls файл для компонента.		
		3. Натиснути кнопку «Завантажити».		
3	Завантаження «нічого»	Натиснути кнопку «Завантажити».	Отримано повідомлення про необхідність завантажити файл	Так
4	Завантаження не xlsx файлу	1. Натиснути кнопку «Вибрати файл».	Отримано повідомлення: «Вибраний файл повинен бути формату Excel (XLSX, XLS)»	Так
		2. Вибрати не xlsx файл для компонента.		
		3. Натиснути кнопку «Завантажити».		
5	Завантажити звіт по всіх даних з таблиці	1. Натиснути кнопку «Завантажити повний звіт».	Файл завантажено, дані експортуються коректно	Так
		2. Завантажити xlsx файл		
6	Перегляд довідки по методам фільтрації	Натиснути кнопку «Довідка по методам фільтрації».	Відкриється модальне вікно з текстом по фільтрам	Так

Продовження таблиці 3.1

7	Прибрати стовпець «Область»	1. Відкрити селектор «Стовпці».	Стовпець «Область» зникне, стане неактивним	Так
		2. Прибрати галочку на позиції «Область».		
8	Додати фільтр даних	1. Натиснути кнопку «Додати фільтр».	В таблиці залишаться лише записи	Так
		2. Вибрати колонку		
		3. Вибрати метод фільтрації «Точне значення».		
		4. Прописати значення для фільтрації.		
9	Завантажити звіт тільки по записам, де в полі	1. Натиснути кнопку «Додати фільтр».	Файл завантажено, дані експортуються коректно	Так
		2. Вибрати колонку		
		3. Вибрати метод фільтрації «Точне значення».		
		4. Прописати значення для фільтрації		
		5. Натиснути кнопку «Завантажити звіт».		
		6. Завантажити xlsx файл		
10	Експортувати всі дані	1. Натиснути кнопку «Експорт»	Файл завантажено, дані експортуються коректно	Так
		2. Завантажити xlsx файл		
11	Експортувати тільки відфільтровані дані	1. Натиснути кнопку «Додати фільтр».	Файл завантажено, дані експортуються коректно	Так
		2. Вибрати колонку		
		3. Вибрати метод фільтрації «Точне значення».		
		4. Прописати значення для фільтрації.		
		5. Натиснути кнопку «Експорт».		
		6. Завантажити xlsx файл		
12	Використання фільтру в заголовку	В фільтрі стовпця	В таблиці залишаться лише записи	Так
13	Додати два фільтри даних	1. Натиснути кнопку «Додати фільтр».	В таблиці залишаться лише записи	Так
		2. Вибрати колонку		
		3. Вибрати метод фільтрації «Точне значення».		
		4. Прописати значення для фільтрації.		
		5. Натиснути кнопку «Додати фільтр».		

Продовження таблиці 3.1

		6. Вибрати колонку		
		7. Вибрати метод фільтрації «Точне значення».		
		8. Прописати значення для фільтрації		
14	Очистити фільтри	1. Натиснути кнопку «Додати фільтр».	Отримати таблицю без фільтрів, зі всіма даними	Так
		2. Вибрати колонку		
		3. Вибрати метод фільтрації «Точне значення».		
		4. Прописати значення для фільтрації.		
		5. Побачити, що в таблиці залишились лише записи, де в полі прописано		
		6. Натиснути кнопку «Очистити».		

Було проведено функціональне тестування всіх компонентів. Результати тестування підтверджують виконання всіх функціональних вимог до компонентів, а також коректну працездатність системи в рамках поставленої задачі.

3.5. Інструкція користувача системи

Розглянемо детальніше основні моменти, які виникають в процесі роботи з програмним модулем парсингу та опрацювання документів.

1. Робота з даними подій, представлено на рисунку 3.25. Функціонал. Перегляд даних подій, які містять інформацію про конкретні події, їх час, учасників, та інші важливі параметри. Можливість сортування, групування та фільтрації цих даних. Графічне представлення даних (діаграми, графіки). Обробка файлів з даними подій (експорт/імпорт).

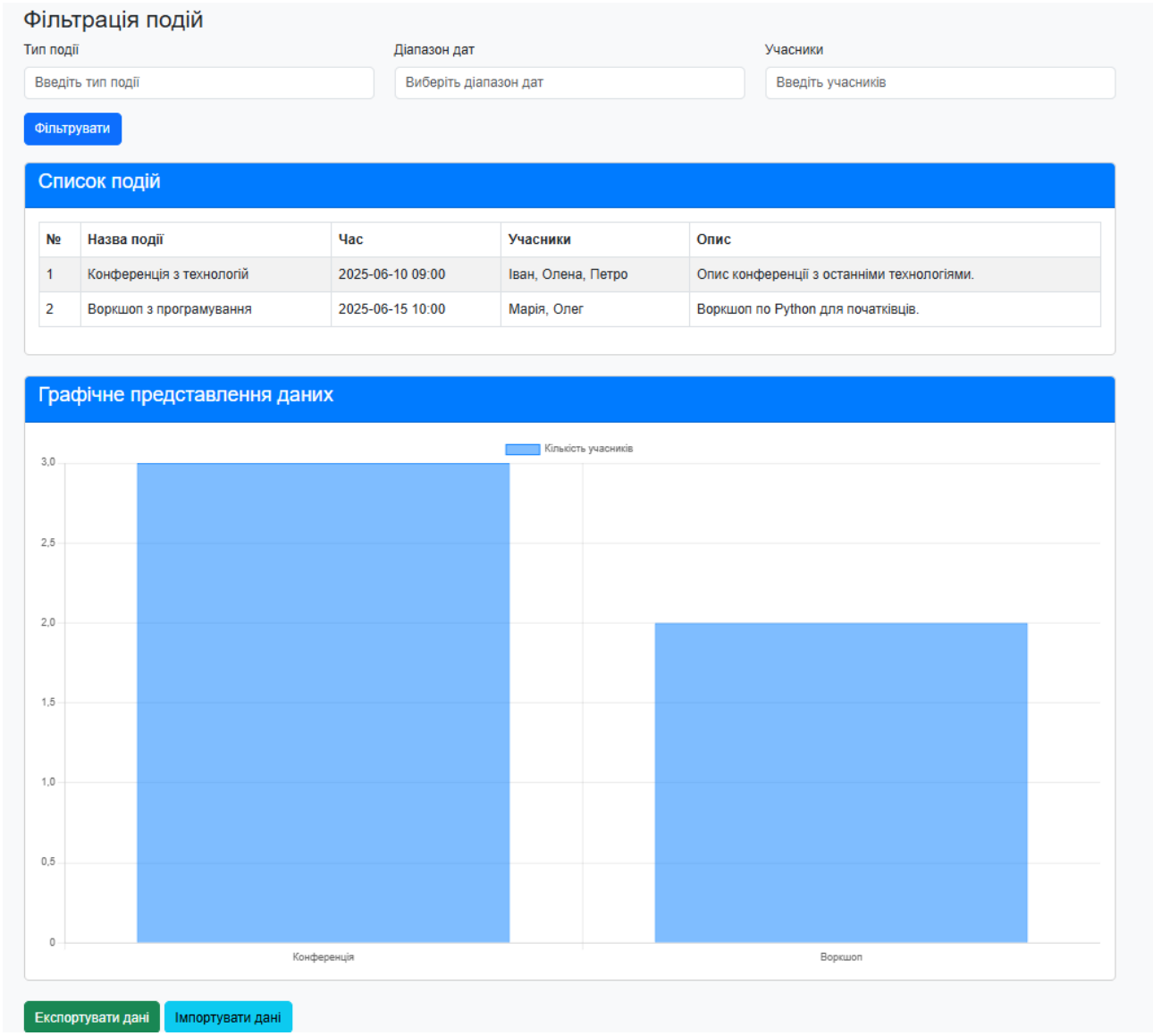

Рис. 3.25. Сторінка фільтрації подій

2. Робота з даними звернень, Рис. 3.36. Перегляд історії звернень, аналіз типів звернень, їх статусів та часового розподілу. Можливість пошуку за різними критеріями (наприклад, по категорії звернення, часу тощо). Визначення частки відкритих/закритих звернень, аналітика. Обробка файлів з даними звернень (фільтрація, сортування).

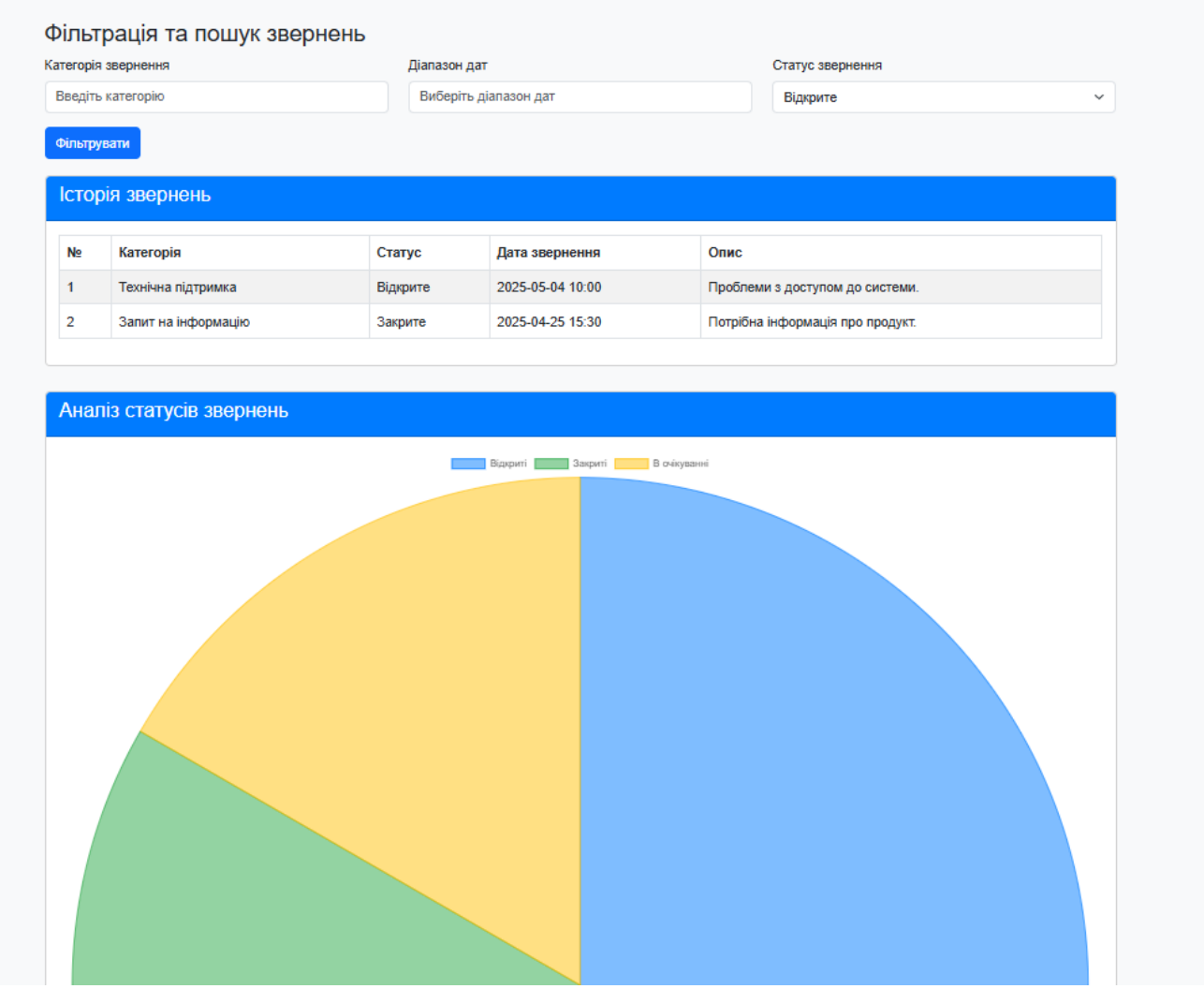

Рис. 3.36. Сторінка з даними звернень

3. Імпорт даних, Рис. 3.37. Користувач може завантажити Excel файл, що містить нові дані, які будуть автоматично імпортовані в таблицю бази даних. Валідація даних перед імпортом (формати, обов'язкові поля тощо).

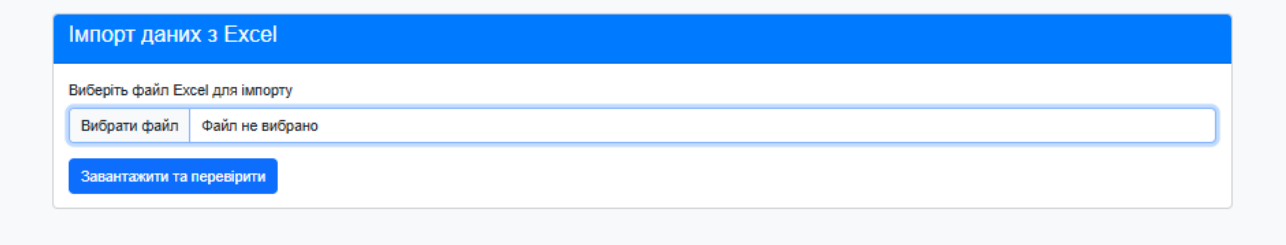

Рис. 3.37. Сторінка імпорту даних

4. Завантажити звіт, Рис. 3.38. Користувач може завантажити повний звіт або тільки звіт по відфільтрованих даних. Можливість вибору формату звіту (Excel, PDF, CSV). Генерація звітів на основі вибраних фільтрів.

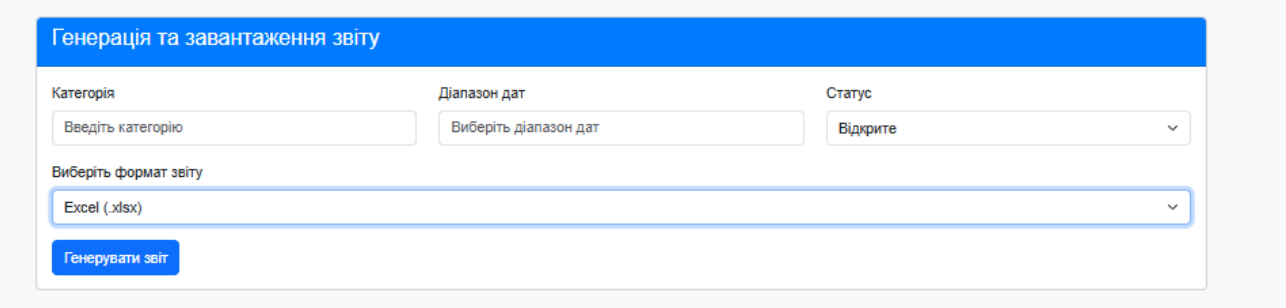

Рис. 3.38. Сторінка завантаження звітів

5. Відфільтрувати дані, Рис. 3.39 Можливість фільтрації даних в таблиці за різними критеріями, такими як дата, категорія, статус і т.д. Збереження налаштувань фільтру для повторного використання.

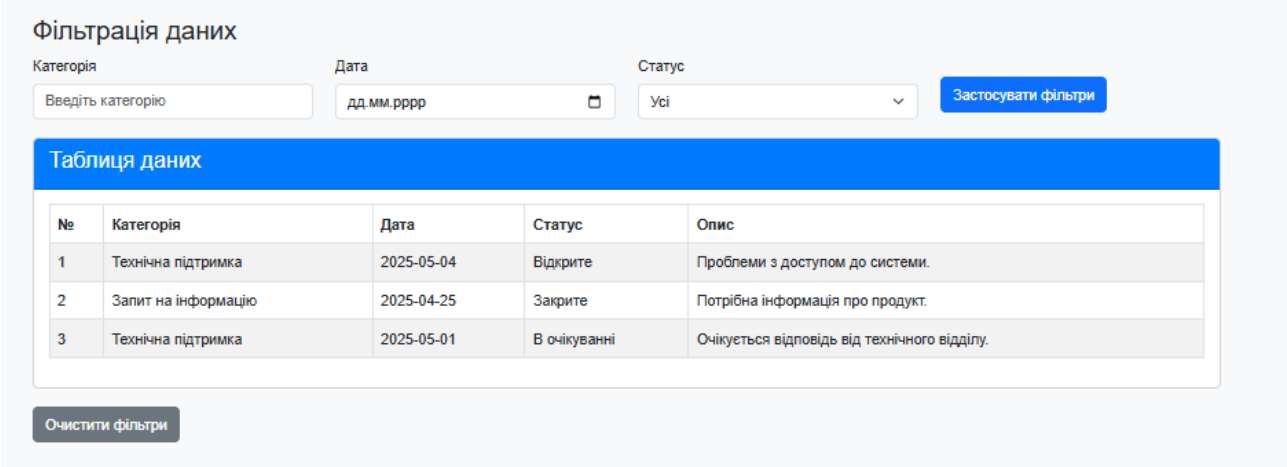

Рис. 3.39. Сторінка фільтрування даних

6. Подивитись довідку по фільтрам, Рис. 3.40. Користувач може отримати довідку по можливих фільтрах і методах їх використання. Інтерфейс із підказками або інтерактивною довідкою.

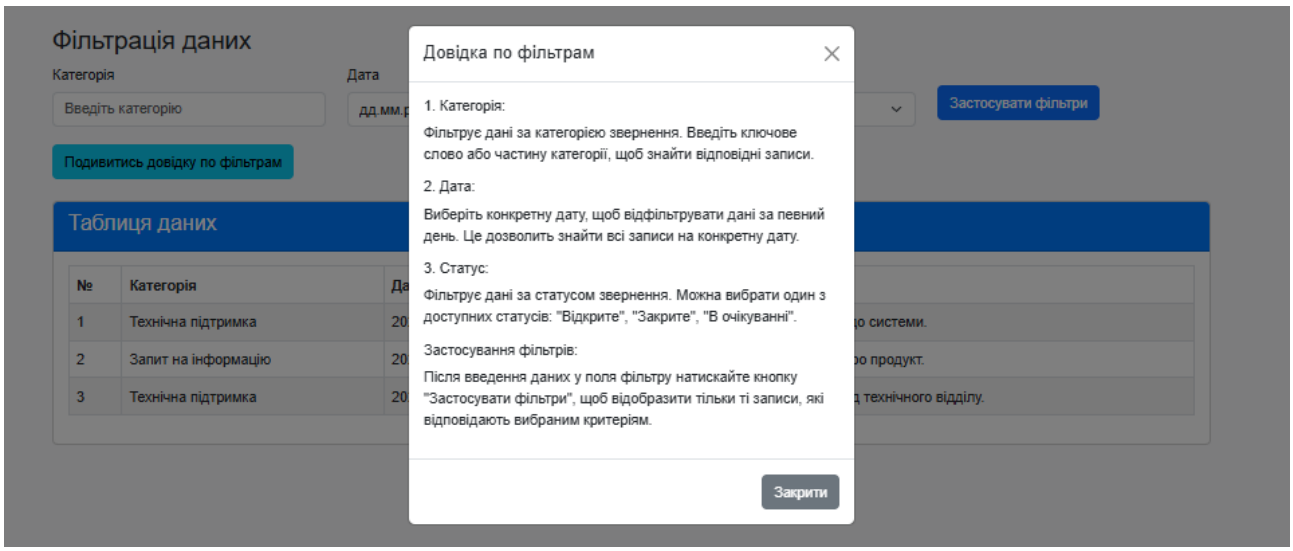

Рис. 3.40. Сторінка перегляду довідки

7. Експортувати дані, Рис. 3.41. Користувач може експортувати дані в різних форматах (Excel, CSV, PDF), з можливістю вибору тільки відфільтрованих даних.

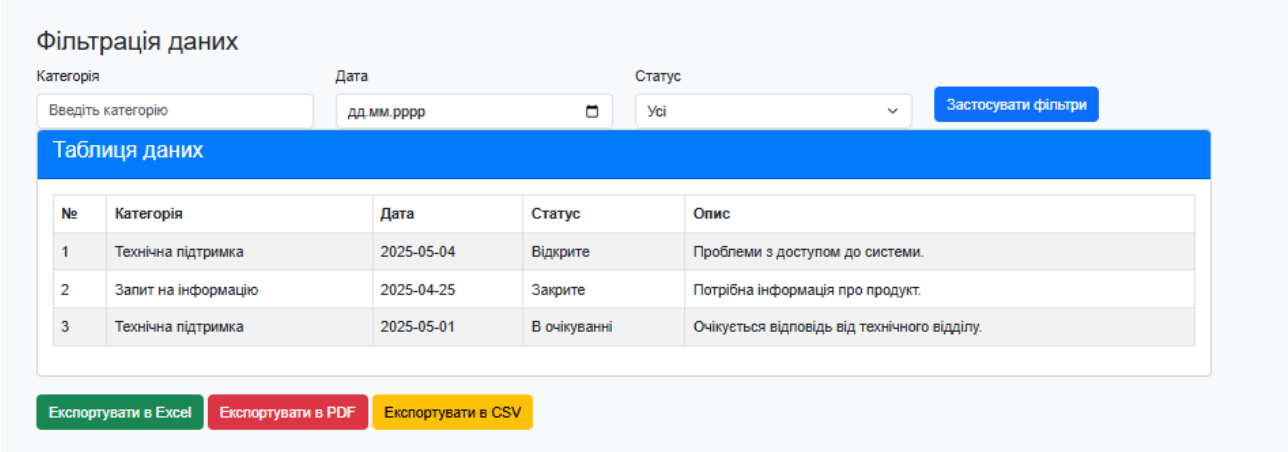

Рис. 3.41. Сторінка експорту даних

8. Завантаження даних про послуги, Рис. 3.42. Користувач може завантажити Excel файл із даними про послуги, що надаються, і система автоматично обробить та завантажить ці дані. Інтеграція з іншими модулями для аналізу ефективності наданих послуг.

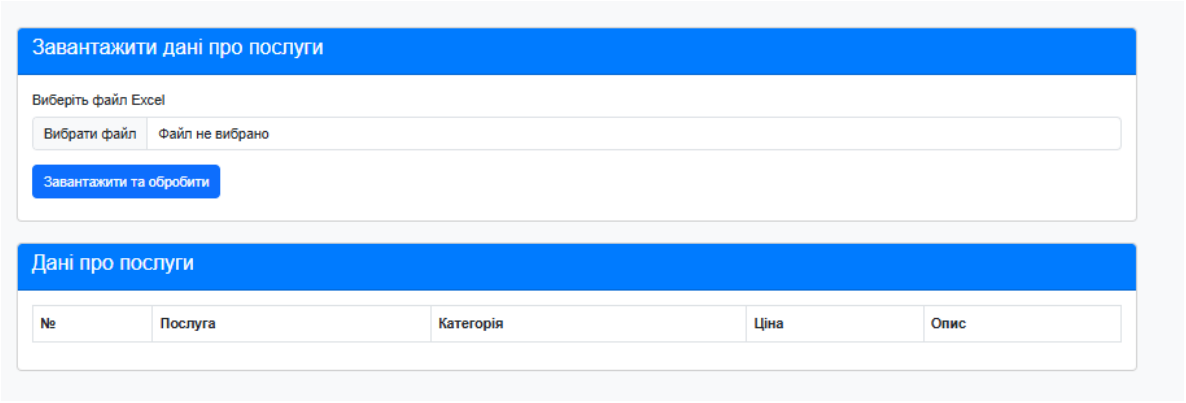

Рис. 3.42. Сторінка завантаження даних про послуги

9. Завантаження статистики по частоті звернень, Рис. 3.43.

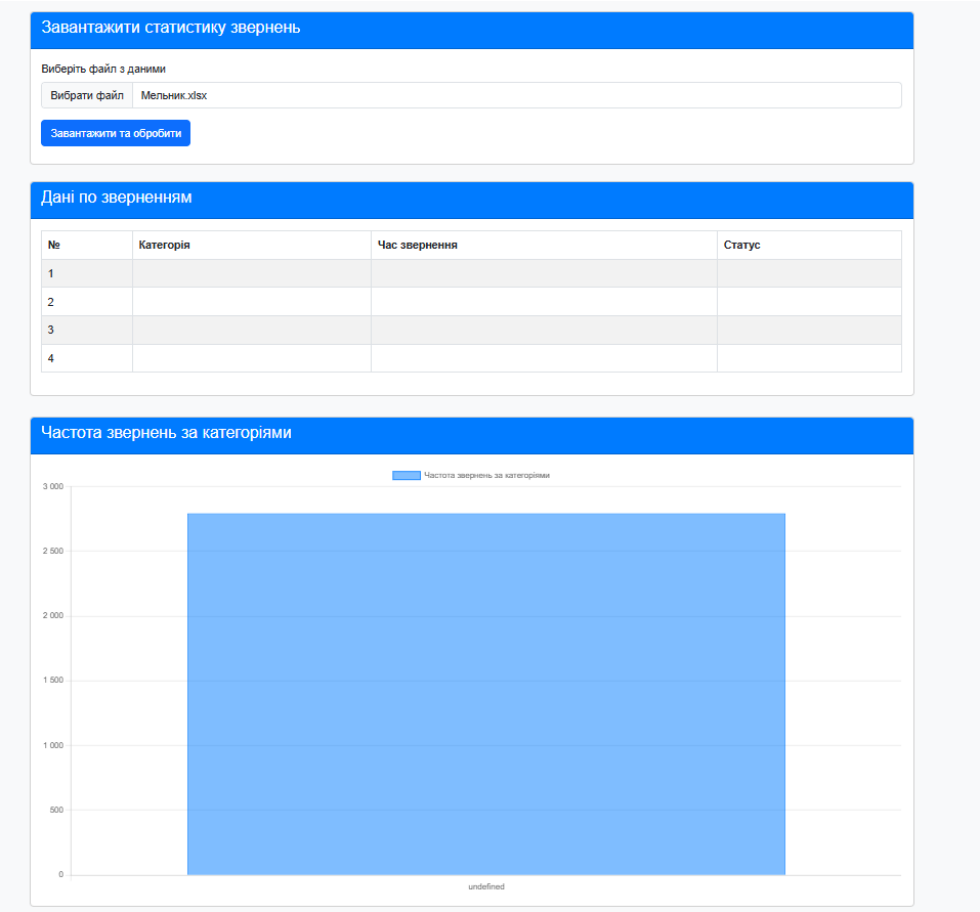

Рис. 3.43. Сторінка завантаження статистики

Користувач може завантажити статистичні дані про частку звернень, їх розподіл за різними категоріями, а також часова статистика. Аналіз статистичних даних і побудова відповідних графіків та діаграм.

4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Порядок надання домедичної допомоги постраждалим при раптовій зупинці серця.

Домедична допомога у разі зупинки серця є надзвичайно важливим заходом, адже за 5-6 хвилин без кровопостачання кора головного мозку починає зазнавати на собі дію процесів, що мають незворотній характер.

Навички першої допомоги – критична необхідність для кожної людини, особливо в умовах війни. Сьогодні це без перебільшень може врятувати життя.

Послідовність дій при наданні домедичної допомоги дорослим при раптовій зупинці кровообігу за відсутності автоматичного зовнішнього дефібрилятора:

1) перед наданням допомоги необхідно переконатися у відсутності небезпеки та за її відсутності перейти до наступного кроку;

2) визначити наявність свідомості – обережно потрясти дорослого за плече та голосно звернутися до нього, наприклад, «З Вами все гаразд? Вам потрібна допомога?»

3) якщо дорослий реагує:

а) залишити його у попередньому положенні, якщо йому нічого не загрожує;

б) з’ясувати характер події, що сталася;

в) здійснити виклик екстреної медичної допомоги та дотримуватись вказівок диспетчера прийому виклику;

г) за необхідності надати дорослому зручного положення;

ґ) забезпечити нагляд за дорослим до приїзду бригади екстреної (швидкої) медичної допомоги;

4) якщо дорослий не реагує:

а) звернутися до осіб, які поряд, за допомогою. Якщо випадкових свідків декілька, слід звертатися до конкретної особи;

б) якщо дорослий лежить на животі, повернути його на спину та відновити прохідність дихальних шляхів.

Якщо стан дорослого пов’язаний з отриманням травми, наприклад падіння з висоти, вважати, що у нього є травма в шийному відділі хребта. В такому випадку слід максимально обмежити рухи в шийному відділі хребта;

в) відновити прохідність дихальних шляхів, визначити наявність дихання за допомогою прийому: «чути, бачити, відчувати». Наявність дихання визначати до 10 секунд. Якщо виникли сумніви чи є дихання, або воно ненормальне, вважати, що дихання відсутнє;

5) якщо дорослий дихає нормально, при відсутності свідомості слід

перевести його у стабільне бокове положення, здійснити виклик екстреної медичної допомоги та перевіряти кожні 3-5 хвилин дихання до моменту приїзду бригади екстреної (швидкої) медичної допомоги. Якщо є настороженість щодо травми потрібно уникати повороту в стабільне бокове положення, а забезпечити прохідність дихальних шляхів методом висування нижньої щелепи до приїзду бригади екстреної (швидкої) медичної допомоги;

6) якщо дихання відсутнє:

а) здійснити виклик екстреної медичної допомоги та дотримуватись вказівок диспетчера прийому виклику. Якщо є інші випадкові свідки, слід сказати їм здійснити виклик екстреної медичної допомоги та негайно перейти до наступного кроку. При наявності гучного зв’язку на телефоні слід здійснювати виклик екстреної медичної допомоги та одночасно проводити серцево-легеневу реанімацію;

б) розпочати проведення серцево-легеневої реанімації: виконати 30 натискань на середину грудної клітки глибиною не менше 5 см (не більше 6 см), з частотою 100 натискань (не більше 120) за хвилину; виконати 2 вдихи з використанням маски-клапану, дихальної маски тощо. При відсутності захисних засобів можна не виконувати штучне дихання, а проводити тільки натискання на грудну клітку. Виконання двох штучних вдихів повинно тривати не більше 5 секунд; після двох вдихів продовжити натискання на грудну клітку відповідно до наведених рекомендацій у цьому підпункті; не слід переривати натиснення на грудну клітку дорослому більше ніж на 10 секунд;

7) змінювати особу, що проводить натискання нагрудну клітку, кожні 2 хвилини.

8) припинити проведення серцево-легеневої реанімації до прибуття бригадиї екстреної (швидкої) медичної допомоги за наступних умов:

- при появі у дорослого явних ознак життя; відновлення самостійного нормального дихання, координованої рухової активності, відкривання очей;
- виникненні загрози життю рятівнику та/або дорослому;
- неможливості проведення серцево-легеневої реанімації внаслідок значного фізичного виснаження.

Послідовність дій при наданні домедичної допомоги дорослим при раптовій зупинці кровообігу та наявності автоматичного зовнішнього дефібрилятора:

1) перед наданням допомоги переконатися у відсутності небезпеки та за її відсутності перейти до наступного кроку;

2) визначити наявність свідомості – обережно потрясти дорослого за плече та голосно звернутися до нього, наприклад, «З Вами все гаразд? Вам потрібна допомога?»;

3) якщо дорослий не реагує:

а) звернутися до осіб, які поряд, за допомогою. Якщо випадкових свідків декілька, слід звертатися до конкретної особи;

б) якщо дорослий лежить на животі, повернути його на спину та відновити прохідність дихальних шляхів. Якщо стан дорослого пов’язаний з отриманням травми, наприклад падіння з висоти, вважати, що у нього є травма в шийному відділі хребта. В такому випадку слід максимально обмежити рухи в шийному відділі хребта;

в) відновити прохідність дихальних шляхів, визначити наявність дихання за допомогою прийому: «чути, бачити, відчувати». Наявність дихання визначати до 10 секунд. Якщо виникли сумніви, чи є дихання, або воно ненормальне, вважати, що дихання відсутнє;

г) здійснити виклик екстреної медичної допомоги та дотримуватись вказівок диспетчера прийому виклику. Якщо є інші випадкові свідки, слід сказати їм

здійснити виклик екстреної медичної допомоги та негайно перейти до наступного кроку. При наявності гучного зв'язку на телефоні слід здійснювати виклик екстреної медичної допомоги та одночасно проводити серцево-легеневу реанімацію. Якщо рятувальник один, і не відомо, що автоматичний зовнішній дефібрилятор знаходиться у безпосередній близькості від місця події, не слід залишати дорослого та шукати автоматичний зовнішній дефібрилятор – в такому випадку слід проводити серцево-легеневу реанімацію як передбачено у пункті 3 цього Порядку;

г) розпочати проведення серцево-легеневої реанімації: виконати 30 натискань на середину грудної клітки глибиною не менше 5 см (не більше 6 см), з частотою 100 натискань (не більше 120) за хвилину; виконати 2 вдихи рекомендовано з використанням маски-клапану, дихальної маски тощо. При відсутності захисних засобів можна не виконувати штучне дихання, а проводити тільки натискання на грудну клітку. Виконання двох штучних вдихів повинно тривати не більше 5 секунд; після двох вдихів продовжити натискання на грудну клітку відповідно до наведених рекомендацій у цьому підпункті; не слід переривати натискання на грудну клітку дорослому більше ніж на 10 секунд;

4) змінювати особу, що проводить натискання на грудну клітку, кожні 2 хвилини. У випадку якщо особа, яка проводить натискання на грудну клітку, відчуває виснаження, заміну слід виконати раніше, ніж через 2 хвилини;

5) як тільки автоматичний зовнішній дефібрилятор наявний на місці події, слід негайно: увімкнути пристрій та чітко дотримуватись голосових вказівок; якщо осіб які надають допомогу декілька, вмикання автоматичного зовнішнього дефібрилятора та приклеювання електродів до грудної клітки дорослого слід одночасно з проведенням компресій на грудну клітку; у випадку необхідності проведення дефібриляції, прослідкувати, щоб ніхто не торкався дорослого; після проведення дефібриляції слід негайно розпочати натискання на грудну клітку;

6) припинити проведення серцево-легеневої реанімації до прибуття бригади екстреної (швидкої) медичної допомоги за наступних умов: при появі у дорослого

явних ознак життя; відновлення самостійного дихання, координованої рухової активності; виникненні загрози життю рятівнику та/або дорослому.

У випадку появи явних ознак життя до приїзду бригади екстреної (швидкої) медичної допомоги, електроди від автоматичного зовнішнього дефібрилятора слід залишити на грудній клітці дорослого.

4.2 Планування робіт щодо охорони праці

У плануванні робіт з охорони праці як вихідні дані використовують:

- план економічного і соціального розвитку підприємства;
- план організаційно-технічних, технологічних та інших заходів, направлених на підвищення ефективності, пошук резервів підприємства;
- матеріали паспортизації цехів, дільниць, робочих місць на відповідність їх вимогам охорони праці; а також атестації робочих місць за умовами праці;
- комплексні заходи щодо досягнення встановлених нормативів безпеки, гігієни праці і виробничої санітарії;
- матеріали розслідувань нещасних випадків, професійних захворювань та аварій;
- приписи органів державного нагляду, служб охорони праці;
- рекомендації комісії з питань охорони праці підприємства; пропозиції уповноважених трудового колективу з питань охорони праці;
- результати адміністративно-громадського контролю стану охорони праці;
- постанови профспілкового комітету, рішення зборів (конференцій) трудового колективу з питань охорони праці;
- відповідні накази адміністрації, документи вищестоящих господарських і профспілкових органів.

Важливе значення у системі планування робіт щодо забезпечення безпеки праці на підприємствах має розробка розділу «Охорона праці» колективного договору (угоди, трудового договору), що укладається між власником і трудовим колективом, за рішенням якого його інтереси може представляти також і профспілка.

Одним з найважливіших розділів колективного договору є інженерно-технічні заходи задля досягнення нормативів безпеки, гігієни праці і виробничого середовища, підвищення наявного рівня охорони праці. До цільових заходів належать, наприклад, розробка, виготовлення і установка нових, ефективніших інженерно-технічних засобів охорони праці, огорож, засобів сигналізації і контролю, запобіжних пристроїв тощо.

Заходи з охорони праці мають бути забезпечені проектно-кошторисно конструкторською та іншою технічною документацією, фінансуванням і матеріальними ресурсами. Згідно зі статтею 19 закону України «Про охорону праці», фінансування охорони праці на підприємстві здійснюється власником.

На підприємствах, в галузях і на державному рівні створюються фонди охорони праці. Такі ж фонди створюються органами місцевого і регіонального самоврядування для потреб регіону. В державний, галузеві та регіональні фонди охорони праці направляються, разом із коштами державного і місцевого бюджетів, відрахування підприємств (в розмірі не менше 0,5% від об’єму реалізації продукції, виконаних робіт, наданих послуг, а для бюджетних підприємств – не менше 0,2% від фонду оплати праці) та іншими надходженнями, кошти, отримані від застосування органами державного нагляду штрафних санкцій до підприємств, а також кошти від штрафування фізичних осіб, винних у порушеннях вимог охорони паці.

На підприємствах фінансування заходів з охорони праці може здійснюватись, зазвичай за рахунок:

- виробничих витрат (матеріальні витрати на удосконалення технології та організації виробництва, підтримку основних виробничих фондів у робочому стані, утримання засобів колективного захисту та ін.);

- амортизації основних фондів;
- капітальних вкладень;
- банківського кредиту;
- кошторису затрат бюджетних організацій і установ; · фонду охорони праці.

Засоби фонду охорони праці підприємства повинні використовуватися тільки на виконання комплексних заходів, спрямованих на досягнення встановлених нормативів з охорони праці, подальше підвищення рівня охорони праці в умовах діючого виробництва.

До цих заходів, які є основою для розробки колективного договору, належать:

- атестація робочих місць на відповідність їх вимогам нормативних актів про охорону праці;
- навчання працівників з питань охорони праці;
- забезпечення працівників спецодягом, спецвзуттям та іншими засобами індивідуального захисту;
- проведення обов'язкових медичних оглядів працівників; · проведення експертизи технічного стану будівель і споруд;
- приведення рівня шуму, вібрації, ультразвуку, іонізуючих та інших шкідливих випромінювань на робочих місцях у відповідність вимогам чинних нормативних актів;
- розробка, виготовлення і монтаж нових або реконструкція існуючих вентиляційних систем, аспіраційних, порохо- і газоуловлювальних пристроїв для приведення у відповідність вимогам чинних нормативних актів про охорону праці та їх паспортизація тощо.

Першочерговість виконання заходів встановлюється на основі атестації робочих місць на відповідність їх вимогам нормативних актів з охорони праці, аналізу причин нещасних випадків на виробництві і професійних захворювань з урахуванням їх економічної і соціальної значимості.

Засоби фонду охорони праці підприємства не повинні витрачатися на ремонтні та інші роботи, пов'язані з підтриманням в належному технічному стані

основних фондів (включаючи інженерно-технічні засоби безпеки, засоби колективного захисту працівників), на надання пільг і компенсацій працівникам, на природоохоронні заходи.

Засоби галузевих і регіональних фондів охорони праці підприємствам направляються згідно з затвердженими кошторисами витрат на фінансування заходів, передбачених відповідно галузевими і регіональними програмами покращення стану безпеки, гігієни праці та виробничої санітарії.

До заходів, які здійснюються за рахунок коштів галузевого фонду охорони праці, належать, наприклад:

- створення і впровадження автоматизованих систем попередження аварій, пожеж, нещасних випадків на виробництві, а також систем управління охороною праці;
- розробка і виконання програм метрологічного, кадрового і організаційного забезпечення атестації робочих місць;
- нормативно-правове забезпечення охорони праці;
- проведення і впровадження у виробництво науково-дослідних, проектноконструкторських робіт з безпеки і гігієни праці.

За рахунок регіонального фонду охорони праці виконуються, наприклад, такі заходи :

- приведення у відповідність з вимогами нормативних актів приміщень для зберігання отруйних, пожежо- і вибухонебезпечних речовин;
- утилізація небезпечних і шкідливих відходів виробництва;
- забезпечення безпечної експлуатації об’єктів з постійним перебуванням великої кількості людей (магазинів, навчальних закладів та ін.).

Керівництво державним фондом охорони праці здійснює Державний комітету країни з нагляду за охороною праці (Держгірпромнагляд). Кошти цього фонду відповідно до кошторису, затвердженого Держгірпромнаглядом за узгодженням з Мінфіном, направляються на фінансування заходів комплексного вирішення завдань з охорони праці, передбачених національною програмою покращення стану безпеки, гігієни праці і виробничого середовища, проведення науководослідних і

проектно-конструкторських робіт в межах цієї програми, сприяння становленню і розвитку спеціалізованих підприємств і виробництв, науковотехнічних центрів і експертних груп та здійснення цих заходів.

Таким чином, на Україні створена і функціонує єдина система планування і фінансування заходів з охорони праці: на підприємствах і в галузях, на регіональному і державному рівнях.

ВИСНОВКИ

В рамках цієї випускної кваліфікаційної роботи були розроблені програмні компоненти системи обробки документів, які будуть використовуватись замовником для покращення внутрішніх робочих процесів. На етапі дослідження були вивчені сучасні підходи та технології в області обробки документів. Було розглянуто існуючі рішення та їх недоліки, що дозволило визначити напрямки для покращення. На основі проведеного аналізу були спроектовані архітектура та специфікації програмних компонентів.

На етапі розробки були застосовані необхідні методи та інструменти програмування, також була проведена серія тестів, включаючи функціональне тестування.

При виконанні поставленої задачі були досягнуті наступні основні результати:

- проведено огляд літератури;
- виконано проектування програмних компонентів;
- виконано реалізацію програмних компонентів;
- проведено тестування програмних компонентів.

Ці компоненти були повністю протестовані та впроваджені в дослідну експлуатацію. В рамках подальших досліджень планується підтримка компонентів, їх оптимізація, покращення системи, а також написання нових компонентів системи обробки документів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. М.Р. Петрик, Д.М. Михалик, О.Ю. Петрик, Г.Б. Цуприк. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – “Інженерія програмного забезпечення” для усіх форм навчання [Текст] – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя – 2020 – 27 с.

2. Pop, Paul. PHP & MySQL: Novice to Ninja. SitePoint, 2012.

3. Mackenzie, Christopher, and Kyle Fiedler. PHP and MySQL Web Development. Addison-Wesley, 2016.

4. Jones, James. Modern PHP: New Features and Good Practices. O'Reilly Media, 2015.

5. Welling, Luke, and Laura Thomson. PHP and MySQL Web Development. 5th ed. Addison-Wesley, 2016.

6. Graham, Tom. Laravel Up & Running: A Framework for Building Modern PHP Apps. O'Reilly Media, 2017.

7. Miller, Daniel. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. 4th ed. O'Reilly Media, 2017.

8. Freeman, Andrew. Laravel: Code Bright. Tuts+, 2014.

9. Tuts+, Laravel Team. Laravel Documentation. Laravel LLC. Accessed April 20, 2025. https://laravel.com/docs.

10. Schmitt, Roland. PHP Objects, Patterns, and Practice. Wiley, 2010.

11. Bryk, O., Mudryk, I., Holubovskyi, M., & Stoianov, Y. (2024). Machine learning models and methods aspects of processing unstructured data. In Proceedings of the 1st International Workshop on Bioinformatics and Applied Information Technologies (BAIT 2024) Zboriv, Ukraine (pp. 64-74).

12. Zentner, William. Mastering Laravel. Packt Publishing, 2016.

13. Bach, Klement. PHP & MySQL for Dynamic Web Sites: Visual QuickPro Guide. Peachpit Press, 2010.

14. LaraShout, Timothy. The Complete Laravel Tutorial Series. Laravel News, 2024. https://laravel-news.com.

15. Smith, Michael. Advanced PHP Programming. Sams Publishing, 2004.

16. Schenk, Julian. The Art of Laravel. LaravelPress, 2018.

17. Swanson, Eric. PHP: A Beginner’s Guide. McGraw-Hill, 2014.

18. Kelsey, Aaron. Pro Laravel. Apress, 2020.

19. Billingsley, Graham. Practical Laravel Projects. 2nd ed. O'Reilly Media, 2022.

20. Кодекс цивільного захисту України [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/5403-17#Text

21. Про затвердження Положення про підсистему реагування на надзвичайні ситуації, проведення аварійно-рятувальних та інших невідкладних робіт єдиної державної системи цивільного захисту : наказ Міністерства внутрішніх справ України від 04.05.2016 №356.

22. Про затвердження Положення про штаб з ліквідації наслідків надзвичайної ситуації та видів оперативно-технічної і звітної документації штабу з ліквідації наслідків надзвичайної ситуації : наказ Міністерства внутрішніх справ України від 26.12.2014 №1406.

23. Про організацію роботи штабу з ліквідації наслідків надзвичайної ситуації та забезпечення його готовності : наказ Державної служби України з надзвичайних ситуацій від 16.03.2015 №149.

24. Рекомендації для роботодавців щодо організації виконання робіт підвищеної небезпеки під час воєнних (бойових) дій [Електронний ресурс] – Режим доступу: https://dsp.gov.ua/faq/rekomendatsii-dlia-robotodavtsiv-shchodo-orhanizatsiivykonannia-robit-pidvyshchenoi-nebezpeky-pid-chas-voiennykh-boiovykh-dii/

ДОДАТКИ

Додаток А

Лістинг програмного модуля

1. Маршрутизація (web.php)

```
use App\Http\Controllers\FileController;

Route::get('/events', [FileController::class, 'viewDataTable']);
Route::post('/events/import', [FileController::class, 'importEvent']);
Route::post('/events/export', [FileController::class, 'exportEvent']);

Route::get('/appeals', [FileController::class, 'viewAppeals']);
Route::post('/appeals/import', [FileController::class, 'importAppeal']);
Route::post('/appeals/export', [FileController::class, 'exportAppeal']);

Route::get('/services', [FileController::class, 'viewServices']);
Route::post('/services/import', [FileController::class, 'importService']);
Route::post('/services/export', [FileController::class, 'exportService']);
```

2. Контролер (FileController.php)

```
namespace App\Http\Controllers;

use App\Models\Event;
use App\Models\Appeal;
use App\Models\Service;
use App\Imports\EventImport;
use App\Imports\AppealImport;
use App\Imports\ServiceImport;
use Maatwebsite\Excel\Facades\Excel;
use Illuminate\Http\Request;

class FileController extends Controller
{
    // Метод для перегляду таблиці подій
    public function viewDataTable()
    {
        $events = Event::all();
        return view('events.index', ['events' => $events]);
    }

    // Метод для імпорту даних про події
    public function importEvent(Request $request)
    {
        $request->validate([
            'file' => 'required|file|mimes:xlsx,xls'
        ], [
            'file.required' => 'Виберіть файл для завантаження',
            'file.mimes' => 'Файл має бути у форматі Excel (XLSX, XLS)'
        ]);

        $file = $request->file("file");

        if ($file->isValid()) {
            Excel::import(new EventImport, $file);
            return redirect()->back()->with('success', 'Файл успішно
завантажено');
        } else {
            return redirect()->back()->withErrors(['file' => 'Файл некоректний']);
```

```
        }
    }

    // Метод для експорту даних про події
    public function exportEvent(Request $request)
    {
        $events = Event::all();
        return Excel::download(new EventExport($events), 'events.xlsx');
    }

    // Метод для перегляду звернень
    public function viewAppeals()
    {
        $appeals = Appeal::all();
        return view('appeals.index', ['appeals' => $appeals]);
    }

    // Метод для імпорту даних про звернення
    public function importAppeal(Request $request)
    {
        $request->validate([
            'file' => 'required|file|mimes:xlsx,xls'
        ], [
            'file.required' => 'Виберіть файл для завантаження',
            'file.mimes' => 'Файл має бути у форматі Excel (XLSX, XLS)'
        ]);

        $file = $request->file("file");

        if ($file->isValid()) {
            Excel::import(new AppealImport, $file);
            return redirect()->back()->with('success', 'Файл успішно
завантажено');
        } else {
            return redirect()->back()->withErrors(['file' => 'Файл некоректний']);
        }
    }

    // Метод для експорту даних про звернення
    public function exportAppeal(Request $request)
    {
        $appeals = Appeal::all();
        return Excel::download(new AppealExport($appeals), 'appeals.xlsx');
    }

    // Метод для перегляду послуг
    public function viewServices()
    {
        $services = Service::all();
        return view('services.index', ['services' => $services]);
    }

    // Метод для імпорту даних про послуги
    public function importService(Request $request)
    {
        $request->validate([
            'file' => 'required|file|mimes:xlsx,xls'
        ], [
            'file.required' => 'Виберіть файл для завантаження',
            'file.mimes' => 'Файл має бути у форматі Excel (XLSX, XLS)'
        ]);

        $file = $request->file("file");
```

```
        if ($file->isValid()) {
            Excel::import(new ServiceImport, $file);
            return redirect()->back()->with('success', 'Файл успішно
завантажено');
        } else {
            return redirect()->back()->withErrors(['file' => 'Файл некоректний']);
        }
    }

    // Метод для експорту даних про послугу
    public function exportService(Request $request)
    {
        $services = Service::all();
        return Excel::download(new ServiceExport($services), 'services.xlsx');
    }
}
```

3. Імпорт класів (EventImport, AppealImport, ServiceImport)

EventImport

```
namespace App\Imports;

use App\Models\Event;
use Maatwebsite\Excel\Concerns\ToModel;
use Maatwebsite\Excel\Concerns\WithHeadingRow;

class EventImport implements ToModel, WithHeadingRow
{
    public function model(array $row)
    {
        return new Event([
            'event_id' => $row['event_id'],
            'name' => $row['name'],
            'date' => $row['date'],
            'type' => $row['type'],
            // додаткові поля
        ]);
    }
}
```

AppealImport

```
namespace App\Imports;

use App\Models\Appeal;
use Maatwebsite\Excel\Concerns\ToModel;
use Maatwebsite\Excel\Concerns\WithHeadingRow;

class AppealImport implements ToModel, WithHeadingRow
{
    public function model(array $row)
    {
        return new Appeal([
            'appeal_id' => $row['appeal_id'],
            'subject' => $row['subject'],
            'date' => $row['date'],
            'status' => $row['status'],
            // додаткові поля
        ]);
    }
```

```
}
```

ServiceImport

```
namespace App\Imports;

use App\Models\Service;
use Maatwebsite\Excel\Concerns\ToModel;
use Maatwebsite\Excel\Concerns\WithHeadingRow;

class ServiceImport implements ToModel, WithHeadingRow
{
    public function model(array $row)
    {
        return new Service([
            'service_id' => $row['service_id'],
            'name' => $row['name'],
            'url' => $row['url'],
            // додаткові поля
        ]);
    }
}
```

4. Експорт класів (EventExport, AppealExport, ServiceExport)

EventExport

```
namespace App\Exports;

use App\Models\Event;
use Maatwebsite\Excel\Concerns\FromCollection;

class EventExport implements FromCollection
{
    protected $events;

    public function __construct($events)
    {
        $this->events = $events;
    }

    public function collection()
    {
        return $this->events;
    }
}
```

AppealExport

```
namespace App\Exports;

use App\Models\Appeal;
use Maatwebsite\Excel\Concerns\FromCollection;

class AppealExport implements FromCollection
{
    protected $appeals;

    public function __construct($appeals)
    {
```

```
        $this->appeals = $appeals;
    }

    public function collection()
    {
        return $this->appeals;
    }
}
```

ServiceExport

```
namespace App\Exports;

use App\Models\Service;
use Maatwebsite\Excel\Concerns\FromCollection;

class ServiceExport implements FromCollection
{
    protected $services;

    public function __construct($services)
    {
        $this->services = $services;
    }

    public function collection()
    {
        return $this->services;
    }
}
```

Додаток Б

DDL Бази даних

```
CREATE TABLE events (
    id INT AUTO_INCREMENT PRIMARY KEY,
    event_id VARCHAR(255) UNIQUE NOT NULL,
    name VARCHAR(255) NOT NULL,
    date DATETIME NOT NULL,
    type VARCHAR(255) NOT NULL,
    federal_district VARCHAR(255),
    subject_rf VARCHAR(255),
    mo_np VARCHAR(255),
    short_event TEXT,
    sector_of_jkh VARCHAR(255),
    process_stage VARCHAR(255),
    object_of_incident VARCHAR(255),
    type_of_incident VARCHAR(255),
    object_elements TEXT,
    date_post_update DATETIME,
    user_login VARCHAR(255),
    user_name VARCHAR(255),
    organization_name VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);

CREATE TABLE appeals (
    id INT AUTO_INCREMENT PRIMARY KEY,
    appeal_id VARCHAR(255) UNIQUE NOT NULL,
    subject VARCHAR(255) NOT NULL,
    date DATETIME NOT NULL,
    status VARCHAR(255) NOT NULL,
    region VARCHAR(255),
    group_themes VARCHAR(255),
    resolution VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);

CREATE TABLE services (
    id INT AUTO_INCREMENT PRIMARY KEY,
    service_id VARCHAR(255) UNIQUE NOT NULL,
    name VARCHAR(255) NOT NULL,
    url VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);

CREATE TABLE quality_control_by_month (
    id INT AUTO_INCREMENT PRIMARY KEY,
```

```
        service_id INT NOT NULL,
        date DATE NOT NULL,
        grade DECIMAL(5, 2),
        FOREIGN KEY (service_id) REFERENCES services(id) ON DELETE CASCADE,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
    );

    CREATE TABLE statements_by_month (
        id INT AUTO_INCREMENT PRIMARY KEY,
        service_id INT NOT NULL,
        date DATE NOT NULL,
        data_quantity INT,
        number_of_failures INT,
        number_of_positive INT,
        FOREIGN KEY (service_id) REFERENCES services(id) ON DELETE CASCADE,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
    );

    CREATE TABLE areas (
        id INT AUTO_INCREMENT PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
    );

    CREATE TABLE appeals_by_months (
        id INT AUTO_INCREMENT PRIMARY KEY,
        service_id INT NOT NULL,
        area_id INT NOT NULL,
        month INT NOT NULL,
        year INT NOT NULL,
        number_applications_in_electronic_form INT,
        number_applications INT,
        percent_of_electronic_forms_of_total DECIMAL(5, 2),
        FOREIGN KEY (service_id) REFERENCES services(id) ON DELETE CASCADE,
        FOREIGN KEY (area_id) REFERENCES areas(id) ON DELETE CASCADE,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
    );

    CREATE TABLE filters (
        id INT AUTO_INCREMENT PRIMARY KEY,
        user_id INT NOT NULL,
        filter_name VARCHAR(255),
        filter_conditions TEXT,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
    );
```