МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ВІЗУАЛІЗАЦІЇ ДАНИХ ЗАСОБАМИ LARAVEL»**

Виконала: студентка 2 курсу, групи 8.1262
спеціальності 126 інформаційні системи та технології
(шифр і назва спеціальності)
освітньої програми інформаційні системи та штучний інтелект
(назва освітньої програми)
А.О. Похваленко
(ініціали та прізвище)

Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 126 інформаційні системи та технології
(шифр і назва)

Освітня програма інформаційні системи та штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук, д.т.н., доцент

_____ Шило Г.М.
(підпис)

" ____ " _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Похваленко Анжеліці Олександрівні
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка веб-застосунку візуалізації даних засобами Laravel

керівник роботи Матвіїшина Надія Вікторівна, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 642-с

2. Строк подання студентом роботи

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі, аналіз предметної області.
2. Прєктування.
3. Реалізація та тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	17.05.2023	
2.	Збір вихідних даних.	07.06.2023	
3.	Обробка методичних та теоретичних	28.06.2023	
	джерел.		
4.	Розробка першого та другого розділу.	30.08.2023	
5.	Розробка третього розділу.	08.11.2023	
6.	Оформлення та нормоконтроль	20.11.2023	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	13.12.2023	

Студент _____ А.О. Похваленко
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н.В. Матвіїшина
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка веб-застосунку візуалізації даних засобами Laravel»: 52 с., 21 рис., 1 табл., 17 джерел, 3 додатка.

API, CHARTS, FRAMEWORK, LARAVEL, MVC, UML.

Об’єкт дослідження – веб-застосунок, інструменти для взаємодії Laravel та charts, засоби взаємодії веб-застосунку з користувачем.

Мета роботи – розробити веб-застосунок візуалізації даних.

Методи дослідження – моделювання, проєктування, програмний, аналітичний.

В сучасному світі, де дані стали цінним ресурсом, візуалізація даних стала невід’ємною частиною багатьох галузей, включаючи бізнес, науку, освіту, та суспільство загалом. Розробка веб-застосунків для візуалізації даних стала особливо актуальною завдяки зростанню обсягу даних, зручності доступу до Інтернету та швидкому розвитку технологій веб-розробки.

Також актуальність розробки веб-застосунків для візуалізації даних підтверджується їх успішним використанням у різних галузях. У бізнесі вони допомагають аналізувати фінансові показники, моніторити ринок та приймати стратегічні рішення. В науці візуалізація даних допомагає науковцям виявляти нові залежності та регулярності в дослідженнях. У сфері освіти вона може бути використана для навчання та відображення навчальних матеріалів.

Отже, в результаті нашої роботи було створено зручний та ефективний веб-застосунок для візуалізації даних з використанням Laravel. Цей застосунок використовує підготовлені дані, які отримані з ресурсів Binance та Minfin за допомогою REST API цих сервісів.

SUMMARY

Master’s qualifying paper «Development of a Data Visualization Web Application Using Laravel»: 52 pages, 21 figures, 1 table, 17 references, 3 supplements.

API, CHARTS, FRAMEWORK, LARAVEL, MVC, UML.

The object of the study is web application, tools for interaction between Laravel and charts, means of interaction between the web application and the user.

The aim of the study is to develop a web application for data visualization.

The methods of research are modeling, design, programming, analytical.

In the modern world, where data has become a valuable resource, data visualization has become an integral part of many fields, including business, science, education, and society as a whole. The development of web applications for data visualization has become particularly relevant due to the growing volume of data, easy access to the Internet, and the rapid development of web development technologies.

The relevance of developing web applications for data visualization is also confirmed by their successful use in various industries. In business, they help analyze financial indicators, monitor the market, and make strategic decisions. In science, data visualization helps researchers identify new dependencies and regularities in their studies. In education, it can be used for teaching and displaying educational materials.

As a result of our work, we have created a convenient and efficient web application for data visualization using Laravel. This application uses prepared data obtained from Binance and Minfin resources using the REST API of these services.

ЗМІСТ

ВСТУП

В сучасному світі обробка та аналіз даних стала необхідністю в різних галузях, починаючи від бізнесу та закінчуючи наукою та громадськими ініціативами. Розробка веб-застосунків для візуалізації даних стає актуальною через зростання обсягів і складності інформації, що потребує аналізу, а також через зростання доступності великих наборів даних та потужності веб-технологій.

Завдяки веб-застосункам для візуалізації даних користувачі можуть отримувати доступ до інформації в будь-який час і з будь-якого пристрою, підключеного до Інтернету. Це особливо важливо для бізнесу, науковців та аналітиків, які мають можливість моніторити дані та приймати рішення в режимі реального часу.

Веб-застосунки для візуалізації даних дозволяють поділитися знаннями та інформацією зі світом. Інтерактивні графіки та візуалізації можуть бути опубліковані в мережі Інтернет та використані для освіти, розповсюдження наукової інформації або просування громадських ініціатив.

Виходячи з цього, було вирішено створити веб-застосунок, котрий би мав простий інтерфейс та був доступний всім бажаючим.

Актуальність дослідження: актуальність теми зумовлена необхідністю візуалізації даних у сферах бізнесу, науки, політики та багатьох інших галузях, так як зростають обсяги та складність інформації, яка потребує обробки та візуалізації.

З огляду на це, можна виділити наступні цілі і задачі нашого дослідження:

Мета: розробити веб-застосунок візуалізації даних.

Задачі:

– сформувати вимоги до системи;
– спроктувати та побудувати архітектуру системи;
– реалізувати веб-застосунок візуалізації даних;

– протестувати роботу системи.

Об’єкт дослідження: процес візуалізації даних користувачем, інструменти для реалізації веб-застосунку візуалізації даних, функціонал веб-застосунку, необхідний користувачам.

Предмет дослідження: створення веб-застосунку, що дозволяє візуалізувати дані.

Методи дослідження: моделювання, проєктування, програмний, аналітичний.

Перший розділ присвячено збору та аналізуванню вимог до системи, огляду інструментів розробки та опису веб-застосунку.

У другому розділі розглянуто етапи проєктування веб-застосунку, наведено детальний опис прецедентів.

Третій розділ присвячено реалізації та тестуванню роботи вебзастосунку, наведено детальне керівництво користувача, яке описує процес роботи з веб-застосунком візуалізації даних.

1 ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Терміни та визначення

Наведемо *загальні* терміни.

Система – веб-сайт створений на основі Laravel.

Laravel – PHP-фреймворк, призначений для розробки веб-додатків відповідно до шаблона model–view–controller.

MySQL – реляційна система керування базами даних.

ДВІ – Діаграма Варіантів Використання чи Use Case Diagram.

ДД – Діаграма Діяльності.

ДП – Діаграма Послідовності.

Користувач – людина, котра перейшла на сторінку веб-застосунку.

Перерахуємо *технічні* терміни.

API – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

БД – база даних, місце збереження інформації ІС.

.env – файл, який дозволяє зберігати та налаштовувати змінні середовища.

MVC – архітектурний шаблон, який використовується під час проєктування та розробки програмного забезпечення.

1.2 Функціональні вимоги

Призначення і цілі створення системи. Функціональне призначення системи – реалізувати можливість візуалізації даних.

Експлуатаційне призначення системи: система може експлуатуватися користувачем системи.

Мета створення системи – розробка системи візуалізації даних.

Загальні функціональні можливості системи. Система має надавати користувачам такі можливості:

– обирати/змінювати спосіб завантаження вхідних даних;
– обирати/змінювати тип діаграми для відображення;
– фільтрувати вхідні дані;
– експортувати результат до pdf.

1.3 Нефункціональні вимоги

Інтерфейс користувача. Система повинна відображати коректно інтерфейс користувача на будь-якому пристрої.

Система повинна мати коректні назви елементів, зрозумілі користувачу.

Підтримка браузерів. Система повинна працювати для наступних браузерів останніх версій: Mozilla Firefox, Google Chrome, Safari, Microsoft Edge, Opera.

Вимоги до продуктивності:

– система повинна відображати будь-яку сторінку не довше, ніж за 1 секунду;
– система повинна відправляти повідомлення не довше, ніж 2 секунди;
– система повинна генерувати відображення не довше, ніж за 2 секунди.

Вимоги до безпеки:

– система не повинна дозволяти вводити у поля дані, які можуть використовуватися, як експлойти;
– система не повинна відображати API-ключі та іншу конфіденційну інформацію, пов’язану з підключенням до зовнішніх сервісів;
– система не повинна запрашувати в користувача особисті або банківські дані;
– система не повинна містити жодних посилань та сторонні ресурси.

1.4 Опис предметної області

Предметною областю є розробка системи для візуалізації чисельних або гібридних даних. Дана система повинна надавати користувачам можливість обирати дані та діаграми для їх візуалізації. Процес взаємодії з системою проходить наступним чином. Користувач, повинен ввести адресу сайту в браузері та перейти на нього.

Після входу, користувач має можливість взаємодіяти з такими розділами як: джерело вхідних даних, тип діаграми, фільтрація, експорт результату до pdf. Процес побудови візуалізації даних проходить наступним чином. Користувач вибирає джерело вхідних даних з запропанованих або завантажує самостійно, керуючись шаблоном. Система відображає дані, використовуючи діаграму за замовчуванням. Після генерації діаграми, користувач має можливість змінити її тип та/або фільтрувати дані.

Також користувач має можливість експортувати результат відображення даних до pdf.

1.5 Опис системи

Веб-застосунки для візуалізації даних допомагають зробити інформацію більш зрозумілою та легше аналізованою. Графіки, діаграми, картографічні візуалізації та інші інтерактивні елементи дозволяють візуально подати дані та виявити закономірності, які можуть залишитися непоміченими в формі текста. Це робить аналіз даних більш ефективним та продуктивним.

Розробка веб-застосунків для візуалізації даних допомагає приймати більш обґрунтовані рішення в різних галузях. Наприклад, у бізнесі це допомагає виробникам і менеджерам зрозуміти динаміку ринку та поведінку споживачів, науковцям – візуалізувати наукові дослідження та дійти нових висновків, а громадським ініціативам – презентувати дані та показати громадськості важливі

проблеми.

У висновку, розробка веб-застосунків для візуалізації даних є актуальною завданням у сучасному світі завдяки зростанню обсягів і складності інформації, що потребує аналізу. Вони допомагають забезпечити зручний доступ до даних, підвищують ефективність аналізу, сприяють прийняттю обґрунтованих рішень та сприяють поширенню знань і інформації.

Зважаючи на актуальність таких систем, була розроблена нова система з зручним та зрозумілим для користувача інтерфейсом.

Можливості системи:

– вибір джерела вхідних даних;
– вибір діаграми для візуалізації вхідних даних;
– експорт до pdf.

1.6 Огляд аналогів

Існує багато інструментів для візуалізації даних, нижче наведено перелік та порівняння схожих за функціоналом інструментів.

Tableau Public [16]:

– переваги: краща в інтерактивності та можливості візуалізацій;
– недоліки: обмежена безкоштовна версія та менша гнучкість в порівнянні з платними планами.

Google Data Studio [9]:

– переваги: широка підтримка джерел даних Google та інших сервісів Google;
– недоліки: є обмеження у безкоштовній версії, але вони розширюються у платних планах.

Power BI [14]:

– переваги: висока функціональність та можливості для аналізу даних;
– недоліки: може бути складніше для новачків, але надає великі

можливості для досвідчених користувачів.

Infogram [10]:

– переваги: простий у використанні, добре підходить для інфографік;
– недоліки: обмежені можливості безкоштовної версії.

Chartio [3]:

– переваги: широкий функціонал для аналізу даних та візуалізації;
– недоліки: зокрема для більших обсягів даних може бути дорогим.

Plotly Chart Studio [13]:

– переваги: велика гнучкість в створенні різноманітних графіків та діаграм;
– недоліки: деякі можливості доступні лише в платних планах.

Visme [17]:

– переваги: зручний інтерфейс та готові шаблони для створення інфографік;
– недоліки: обмежена можливість в безкоштовній версії.

Система (Laravel):

– переваги: повний контроль над реалізацією та можливість інтеграції з іншими частинами додатку;
– недоліки: може вимагати програмування та розробки власних рішень для візуалізації.

Вибір між цими інструментами залежатиме від досвіду, конкретних вимог та зручності використання (див. табл. 1.1).

Таблиця 1.1 – Порівняння інструментів візуалізації

Інструмент	**Безкоштовний варіант**	**Підтримка джерел даних**	**Зручність використання**	**Можливості візуалізації**
Tableau Public	Частково	Підтримує власний формат даних, Excel,	Інтерфейс може здаватися складним для	Різноманітні графіки, карти, дашборди,

Продовження табл. 1.1

Інструмент	**Безкоштовний варіант**	**Підтримка джерел даних**	**Зручність використання**	**Можливості візуалізації**
		Google Sheets, CSV, JSON та інші.	новачків, але потужним.	можливість публікації онлайн.
Google Data Studio	Частково	Підтримує Google Sheets, BigQuery, Google Analytics, YouTube, AdWords, CSV та інші джерела даних.	Легкий інтерфейс, можливість створювати динамічні звіти.	Звіти, панелі, інтерактивні графіки, можливість спільної роботи.
Power BI	Частково	Підтримує Excel, SQL-бази, Azure, Google Analytics, Salesforce, JSON, XML та інші джерела даних.	Потужний інтерфейс для аналізу даних та створення дашбордів.	Дашборди, графіки, карти, аналіз даних, можливість інтеграції.
Infogram	Частково	Підтримує Excel, CSV, JSON, Google Sheets, REST	Легкий у використанні, можливість використання	Інфографіки, діаграми, графіки, карти,

Продовження табл. 1.1

Інструмент	**Безкоштовний варіант**	**Підтримка джерел даних**	**Зручність використання**	**Можливості візуалізації**
		API та інші.	готових шаблонів.	можливість анімації.
Chartio	Ні	Підтримує MySQL, PostgreSQL, Amazon Redshift, Google BigQuery, CSV, Excel та інші джерела даних.	Зручний інтерфейс для аналізу та візуалізації даних.	Графіки, дашборди, аналіз даних, можливість перегляду SQL-запитів.
Система (Laravel)	Так	Підтримує REST API. Легка інтеграція з іншими джерелами.	Легкий інтерфейс. Є можливість розширення.	Різні типи графіків. Є можливість розширення.

Кожен з цих інструментів може бути оптимальним в певних сценаріях використання.

2 ПРОЄКТУВАННЯ

2.1 Використання UML під час розробки системи

UML (Unified Modeling Language) – це стандартизована мова моделювання, яка використовується для розробки, документування та аналізу системних інформаційних систем. Вона надає спосіб узгодження та візуалізації різних аспектів системи, допомагаючи команді розробників та стейкхолдерам зрозуміти систему, її архітектуру і функціональність.

Ось деякі основні аспекти використання UML під час розробки системи.

Візуалізація системи. UML дозволяє створювати графічні представлення різних частин системи, такі як класи, об’єкти, компоненти, сценарії використання, діаграми послідовності тощо. Це полегшує спілкування між розробниками і стейкхолдерами і сприяє кращому розумінню системи.

Аналіз та проєктування. UML допомагає розробникам аналізувати потреби системи, створювати моделі її архітектури, класів, взаємодій та інших аспектів. Ви можете використовувати діаграми, такі як діаграми класів, діаграми послідовності, діаграми компонентів тощо для цього.

Документування. UML є ефективним інструментом для документування системи. Ви можете створити повний набір діаграм та текстового опису, що допоможе команді розробників та іншим зацікавленим стейкхолдерам зрозуміти систему.

Генерація коду. Деякі інтегровані середовища розробки (IDE) підтримують автоматичну генерацію коду на основі UML-моделей. Це може значно спростити роботу розробників та зменшити кількість помилок.

Управління змінами. UML дозволяє вносити зміни в систему, модифікувати архітектуру та аналізувати їх вплив на весь проєкт. Ви можете використовувати діаграми версій та діаграми станів, щоб краще контролювати розвиток системи.

Спрощення комунікації. Використання стандартних термінів та понять UML полегшує комунікацію між різними членами команди розробників та стейкхолдерами.

Загалом, UML є потужним інструментом для розробки систем, який допомагає покращити якість, розуміння та управління проєктом. Успішне використання UML вимагає ретельного аналізу вимог, глибокого розуміння методології моделювання та ефективного використання інструментів UML.

2.2 Діаграма варіантів використання

Діаграма варіантів використання (Use Case Diagram) – це один із видів діаграм в рамках мови моделювання UML, яка використовується для візуалізації функціональних вимог до системи та її відносин зі стейкхолдерами (користувачами) системи. Діаграма варіантів використання допомагає зрозуміти, як система взаємодіє з користувачами і іншими зовнішніми агентами.

Ось деякі ключові концепції та елементи діаграми варіантів використання.

Актори (Actors): актори представляють стейкхолдерів або зовнішні агенти, які взаємодіють з системою. Актори можуть бути людьми, іншими системами, зовнішніми програмами тощо.

Варіанти використання (Use Cases): варіанти використання представляють конкретні функціональні можливості або функції, які система надає акторам. Вони описують, як система реагує на взаємодію з акторами.

Відносини між акторами та варіантами використання: відносини показують, як актори взаємодіють з варіантами використання. Найпоширеніші відносини включають.

– асоціація (Association): вказує, який актор користується певним варіантом використання;
– наслідування (Generalization): показує узагальнення варіанту використання. Наслідування використовується, коли один варіант

використання включає функціональність іншого (підклас);

– включення (Include): показує, що один варіант використання може включати інший варіант використання як підпрограму.

Діаграма варіантів використання допомагає командам розробників та стейкхолдерам зрозуміти, як система має взаємодіяти з користувачами та іншими системами, які функціональні можливості мають бути включені в систему і які необхідності повинні бути враховані в процесі розробки.

На рисунку 2.1 представлена діаграма варіантів використання.

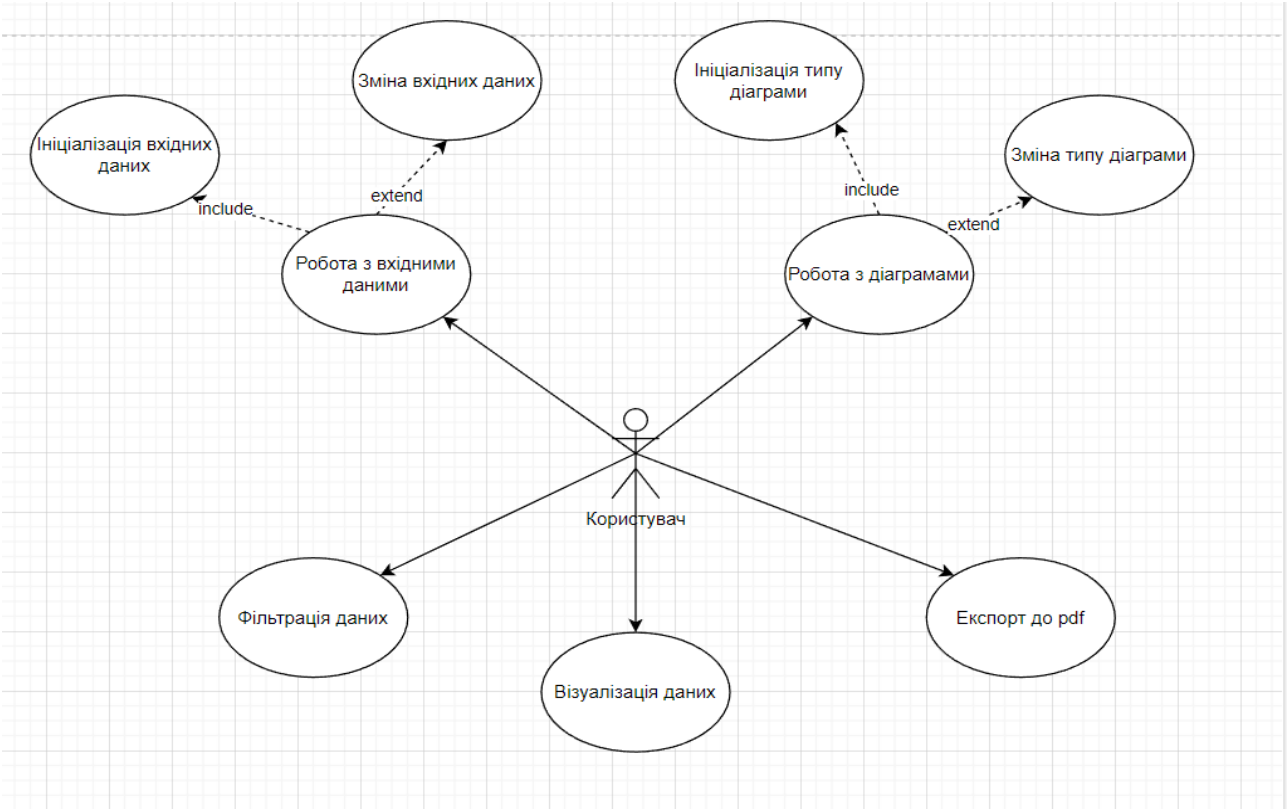

Рисунок 2.1 – Діаграма варіантів використання

На діаграмі представлено актора «Користувач», який відображає функціональні можливості користувачів системи.

Виділено 1 основний варіант використання – «Візуалізація даних». Після входу користувача на сайт системи візуалізації даних, автоматично виконуються дії «Ініціалізація вхідних даних» та «Ініціалізація типу діаграми». Ці кроки встановлюють налаштування за замовчуванням та дозволяють одразу отримати

відображення даних. Якщо користувач потребує інший набір даних або тип діаграми, він може їх змінити, вибравши їх із відповідних списків. Після того, як користувач обрав дані та тип діаграми, він натискає кнопку «Візуалізація даних», і система надсилає запит до відповідного сервісу для візуалізації отриманих даних.

Варіанти використання визначають функціональні можливості. Кожен з них представляє певний спосіб використання. Таким чином, кожен варіант використання відповідає послідовності дій для того, щоб користувач міг отримати певний результат.

2.2.1 Опис варіантів використання

Прецедент «Робота з вхідними даними»

Призначення: даний варіант використання надає можливість користувачу взаємодіяти з вхідними даними.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач потребує вибрати вхідні дані для створення візуалізації. Система пропонує список доступних джерел даних. Після вибору джерела, користувач може обирати типи діаграм і фільтри для відображення даних.

Альтернативний потік: користувач не обрав джерело вхідних даних – система повідомляє його про це, і користувач може спробувати обрати дані ще раз.

Передумова: перед початком виконання даного варіанта використання користувач повинен знаходитися на веб-сторінці системи.

Прецедент «Робота з діаграмами»

Призначення: даний варіант використання надає можливість користувачу взаємодіяти з діаграмами.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач потребує вибрати тип діаграми для візуалізації. Система

пропонує список доступних типів діаграм. Після вибору типу діаграми, користувач може натиснути кнопку «Візуалізація даних».

Альтернативний потік: користувач не обрав тип діаграми – система повідомляє його про це, і користувач може спробувати вибрати тип діаграми ще раз.

Передумова: перед початком виконання даного варіанта використання користувач повинен знаходитися на веб-сторінці системи та обрати джерело вхідних даних.

Прецедент «Ініціалізація вхідних даних»

Призначення: даний варіант використання надає можливість користувачу відразу приступити до візуалізації даних, використовуючи джерело даних за замовчуванням.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач переходить на веб-сторінку системи. Система автоматично вибирає набір вхідних даних за замовчуванням і відображає їх у відповідному полі.

Передумова: перед початком виконання даного варіанта використання користувач повинен перейти на веб-сторінку системи.

Прецедент «Ініціалізація типу діаграми»

Призначення: даний варіант використання надає можливість користувачу відразу приступити до візуалізації даних, використовуючи тип діаграми за замовчуванням.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач переходить на веб-сторінку системи. Система автоматично вибирає тип діаграми за замовчуванням і відображає їх у відповідному полі.

Передумова: перед початком виконання даного варіанта використання користувач повинен перейти на веб-сторінку системи.

Прецедент «Зміна вхідних даних»

Призначення: даний варіант використання надає можливість користувачу

змінити джерело вхідних даних.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач натискає на поле вибору джерела вхідних даних на веб-сторінці системи. Система показує список доступних джерел даних, і користувач може обрати нове джерело. Після вибору джерела, система зберігає зміни.

Альтернативний потік подій: якщо користувач не обрав нове джерело даних, система не вносить змін.

Передумова: перед початком виконання даного варіанта використання користувач повинен перейти на веб-сторінку системи.

Прецедент «Зміна типу діаграми»

Призначення: даний варіант використання надає можливість користувачу змінити тип діаграми.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач натискає на поле вибору типу діаграми на веб-сторінці системи. Система показує список доступних типів діаграм, і користувач може обрати новий тип. Після вибору діаграми, система зберігає зміни.

Альтернативний потік подій: якщо користувач не обрав новий тип діаграми, система не вносить змін.

Передумова: перед початком виконання даного варіанта використання користувач повинен перейти на веб-сторінку системи.

Прецедент «Візуалізація даних»

Призначення: даний варіант використання надає можливість візуалізувати дані на основі обраних джерела вхідних даних та типу діаграми.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач натискає на кнопку «Візуалізація даних» на веб-сторінці системи. Система відправляє запит до відповідного сервісу та візуалізує отримані дані.

Передумова: перед початком виконання даного варіанта використання користувач повинен обрати джерело вхідних даних та тип діаграми.

Виняткова ситуація 1: не обрано джерело вхідних даних – система відобразить відповідне повідомлення, і користувач може обрати джерело ще раз.

Виняткова ситуація 2: не обрано тип діаграми – система відобразить відповідне повідомлення, і користувач може обрати тип діаграми ще раз.

Виняткова ситуація 3: сервіс джерела вхідних даних не відповідає – система відобразить відповідне повідомлення, і користувач може змінити джерело вхідних даних.

Прецедент «Фільтрація даних»

Призначення: даний варіант використання надає можливість користувачу фільтрувати вхідні дані.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач обирає фільтр та вводить значення для фільтрації. Система фільтрує вхідні дані та змінює відображення.

Передумова: перед початком виконання даного варіанта використання користувач повинен натиснути на кнопку «Візуалізація даних».

Виняткова ситуація 1: введено невалідні дані – система відобразить відповідне повідомлення, і користувач може змінити дані.

Виняткова ситуація 2: не активовано прецедент «Візуалізація даних» – система блокує можливість взаємодіяти з фільтрами.

Прецедент «Експорт до pdf»

Призначення: даний варіант використання надає можливість користувачу експортувати результат візуалізації даних у формат pdf.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач натискає на кнопку «Експортувати до PDF» на веб-сторінці системи. Система завантажує поточний візуальний звіт на пристрій користувача.

Передумова: перед початком виконання даного варіанта використання користувач повинен натиснути на кнопку «Відобразити дані».

Виняткова ситуація 1: не активовано прецедент «Візуалізація даних» – система блокує кнопку «Експортувати до PDF».

2.3 Діаграма діяльності

Діаграма діяльності – це графічне зображення послідовності дій або процесу, яке допомагає ілюструвати, як відбуваються певні операції чи події в рамках конкретної діяльності. Вона часто використовується для моделювання бізнес-процесів, системного аналізу, розробки програмного забезпечення та інших областей. Діаграми діяльності допомагають краще розуміти послідовність подій і можуть бути використані для уточнення, оптимізації або автоматизації процесів.

Основні елементи діаграми діяльності включають в себе:

- початок та кінець: ці символи позначають початок і закінчення діаграми;
- діяльності (дії): ці блоки представляють конкретні дії, які виконуються під час процесу, вони зазвичай позначаються прямокутниками і містять опис дії;
- рішення (розгалуження): ці блоки використовуються для умовних операторів, які визначають шляхи в процесі в залежності від певних умов, зазвичай вони позначаються ромбами;
- сполучення (з’єднання): вони використовуються для з’єднання різних шляхів в процесі або для повернення до попереднього кроку;
- паралельні процеси: якщо в процесі відбуваються кілька дій одночасно, то вони можуть бути позначені паралельними лініями;
- параметри (дані): в діаграмі діяльності можуть бути вказані дані, які використовуються в процесі.

Діаграма діяльності допомагає легше розуміти, аналізувати та комунікувати процеси, що відбуваються в конкретній діяльності чи системі. Вона може бути використана для покращення ефективності та організації робочих процесів.

Наведемо діаграму діяльності, що описує модель поведінки варіанта використання «Візуалізація даних». Діаграма представлена на рисунках 2.2 та 2.3.

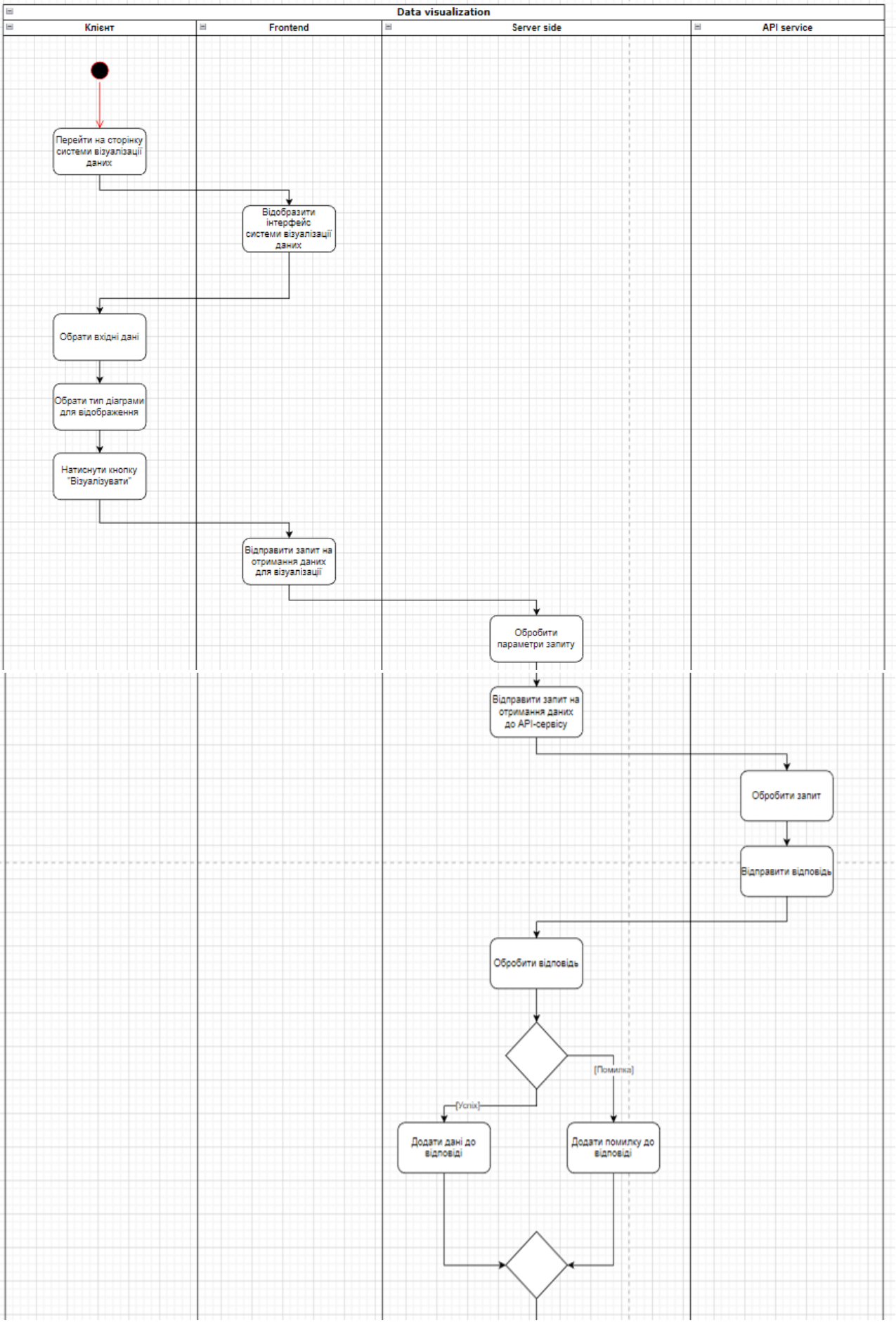

Рисунок 2.2 – Діаграма діяльності

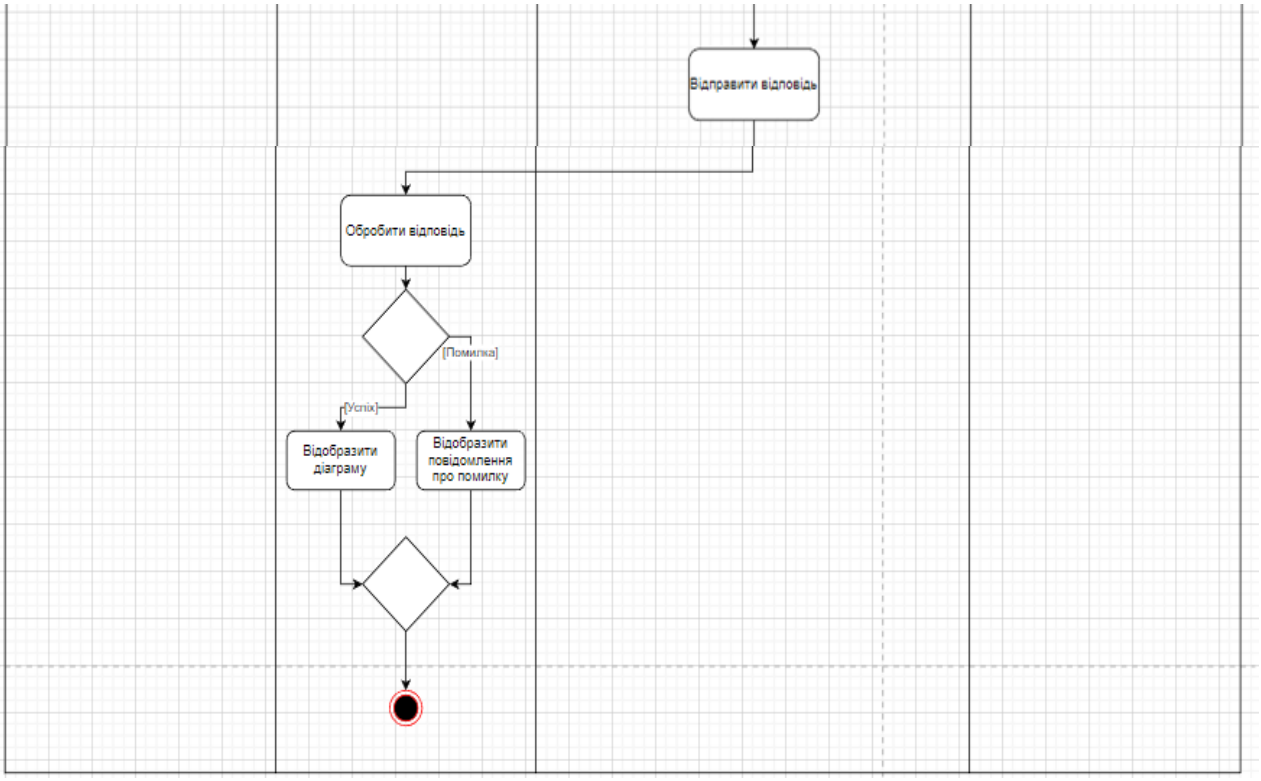

Рисунок 2.3 – Діаграма діяльності

2.4 Діаграма послідовності

Діаграма послідовності, також відома як UML (Unified Modeling Language) діаграма послідовності, це графічний спосіб моделювання послідовних взаємодій між різними об’єктами або компонентами в системі чи програмі. Вона дозволяє ілюструвати, як об’єкти спілкуються один з одним і які повідомлення передаються між ними в рамках конкретного сценарію або операції.

Діаграма послідовності допомагає моделювати та аналізувати взаємодії між об’єктами в системі або програмі та визначає послідовність подій. Це корисний інструмент для розробки програмного забезпечення, проєктування систем, аналізу бізнес-процесів і документації взаємодій між різними частинами системи.

На рисунку 2.4 описана діаграма послідовності прецедента «Візуалізувати дані».

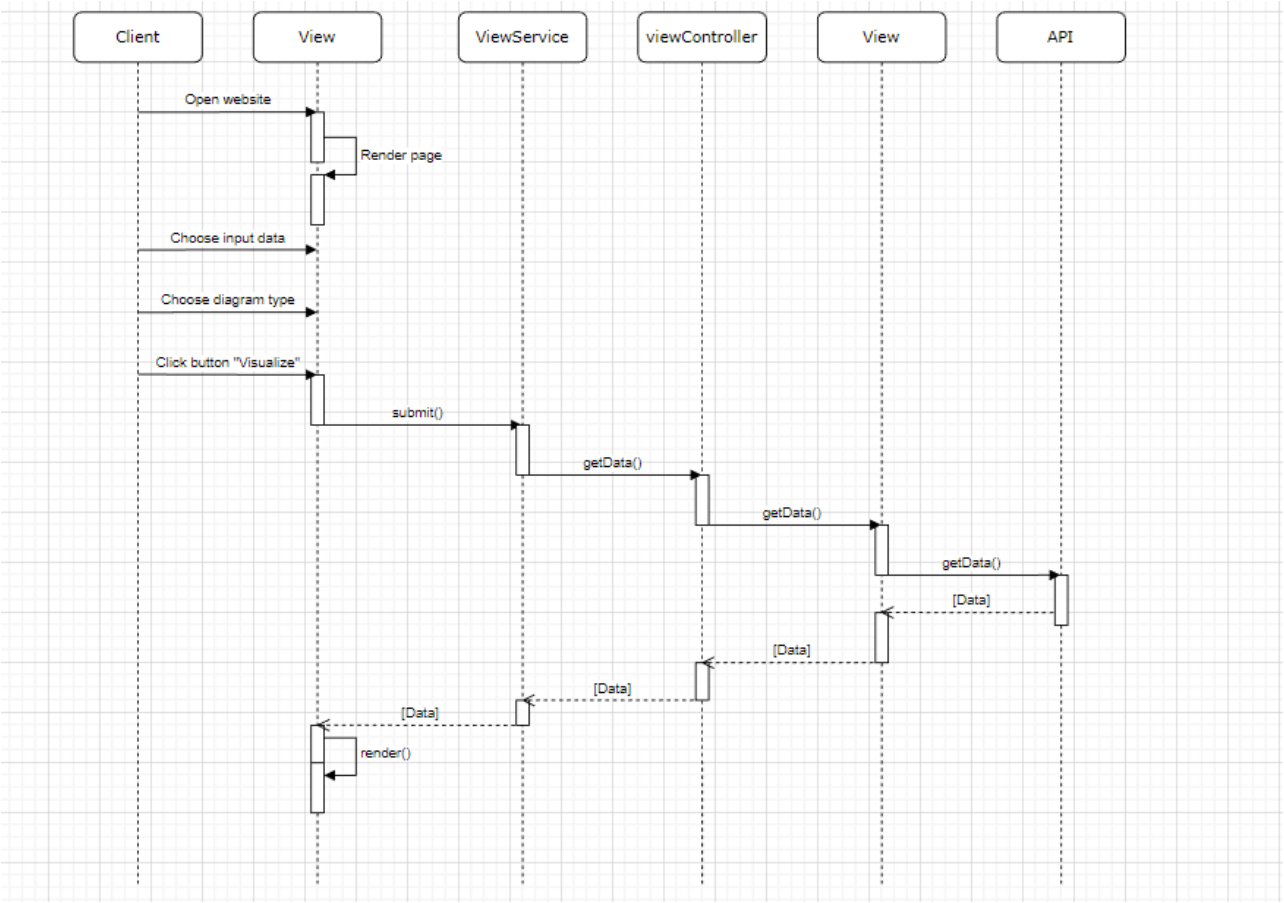

Рисунок 2.4 – Діаграма послідовності

Розглянемо основні елементи діаграми послідовності.

Об'єкти. Об'єкти або учасники взаємодій представляються у верхній частині діаграми. Це можуть бути конкретні об'єкти, такі як користувачі, системи, класи програмного забезпечення, або будь-які інші учасники взаємодії.

Повідомлення. Повідомлення, які передаються між об'єктами, показані у вигляді стрілок, які йдуть від одного об'єкта до іншого. Повідомлення описують операції чи методи, які викликаються між об'єктами.

Часова лінія. Часова лінія показує послідовність подій і взаємодій між об'єктами на діаграмі. Це допомагає визначити, коли події відбуваються у відношенні до один одного.

Загальні фрейми. Загальні фрейми використовуються для визначення більших блоків функціональності в діаграмі послідовності, які можуть бути подробніше розглянуті в окремих частинах діаграми.

2.5 Діаграма розгортання

Діаграма розгортання – це один з типів діаграм UML (Unified Modeling Language), який використовується для моделювання архітектури розгортання системи або програмного забезпечення на рівні апаратного забезпечення. Ця діаграма допомагає показати, як різні компоненти програмного забезпечення, апаратне забезпечення та інші ресурси розгортані фізично та як вони взаємодіють один з одним для забезпечення функціональності системи.

На рисунку 2.5 наведено діаграму розгортання системи візуалізації даних.

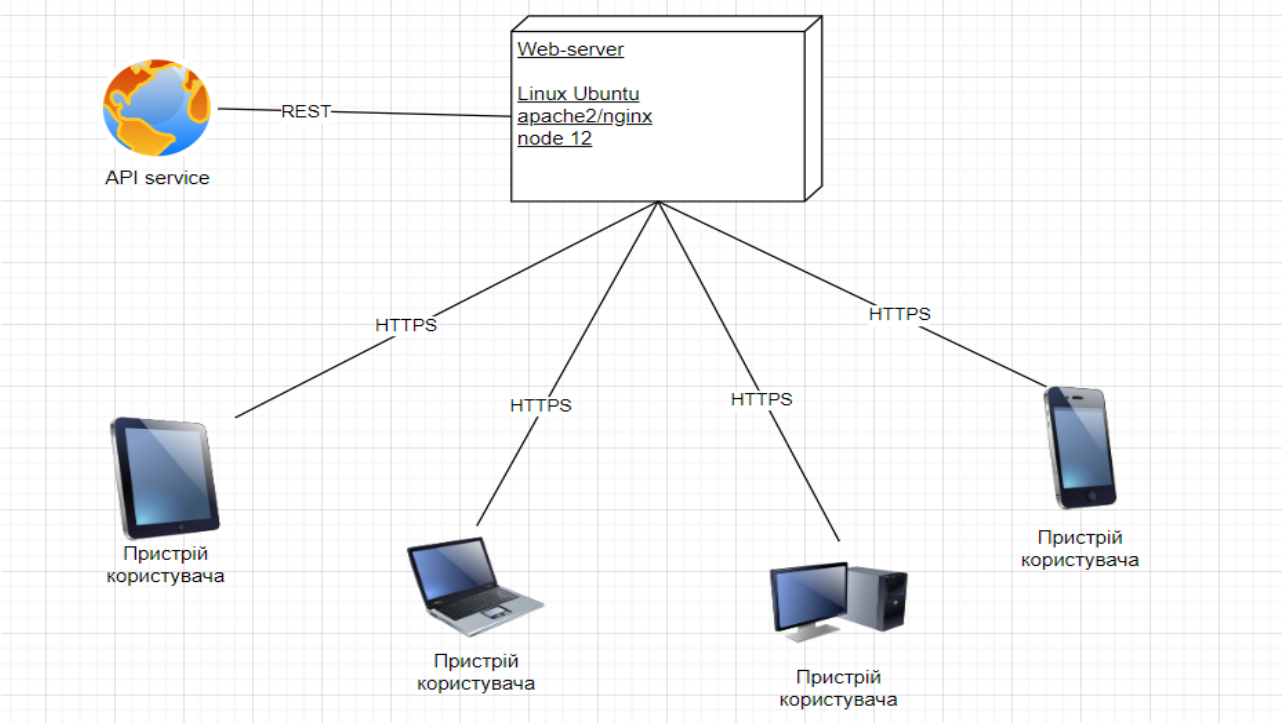

Рисунок 2.5 – Діаграма розгортання

Основні елементи діаграми розгортання включають в себе:

– вузли: вузли представляють апаратне забезпечення або інші обчислювальні ресурси, такі як сервери, комп’ютери, мобільні пристрої тощо (вони зображені у вигляді квадратів або ромбів);

– артефакти: артефакти позначають програмні компоненти, які розгортані на вузлах (це можуть бути файли, програмні модулі, бази даних та інші

ресурси);

- зв’язки: зв’язки між вузлами та артефактами показують, як програмні компоненти розподіляються на різних вузлах, а також вказують способи зв’язку та комунікації між ними;
- анотації: анотації можуть використовуватися для надання додаткової інформації щодо апаратного забезпечення, програмних компонентів та інших елементів на діаграмі.

Діаграма розгортання корисна під час проєктування та аналізу архітектури системи, особливо у великих розподілених системах. Вона допомагає розробникам та архітекторам системи краще розуміти, як компоненти взаємодіють між собою та де вони фізично розгорнуті.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Опис інструментів розробки

Для реалізації були використані Laravel та пакет charts.

Charts – це пакет для Laravel, який дозволяє створювати графіки та діаграми в Laravel додатку [12].

Laravel – відкритий фреймворк для розробки веб-додатків, написаний на мові програмування PHP [15].

3.2 Laravel-модель

Модель відповідає за обробку та управління даними додатку. Це можуть бути взаємодії з базою даних, зберігання та маніпуляція даними, а також бізнес-логіка, пов’язана з цими даними [1, 2].

На рисунку 3.1 наведено приклад моделі.

```
public static function getData($dataSource)
{
    $apiUrl = '';

    // Зчитуємо значення з .env
    if ($dataSource === 'binance') {
        $apiUrl = env('BINANCE_API_URL');
    } elseif ($dataSource === 'minfin') {
        $apiUrl = env('MINFIN_API_URL');
    }

    // Викликаємо API лише якщо маємо URL
    if (!empty($apiUrl)) {
        $client = new Client();
        $response = $client->get($apiUrl);
        return json_decode($response->getBody(), true);
    }

    // Повертаємо порожній масив у випадку відсутності URL
    return [];
}
```

Рисунок 3.1 – Приклад моделі

Повний код моделі наведено в Додатку А.

Модель може включати схеми, моделі даних та методи для отримання/збереження/оновлення/видалення даних, включаючи правила валідації, взаємодію з базою даних та зовнішніми сервісами [8, 11].

3.3 Laravel-контролер

Відповідає за обробку вхідних запитів та взаємодію з моделлю та поданням. Контролер містить логіку, яка обробляє запити користувача, виконує необхідні операції з моделлю та відправляє дані до відповідного подання для відображення результату [4, 7].

Основна задача контролера полягає в тому, щоб приймати запити, зчитувати параметри, викликати відповідні методи моделі для обробки даних та підготовки результату, а потім передавати цей результат до відповідного подання для відображення. Контролер також може відповідати за маршрутизацію, визначення шляхів та керування потоком програми [11, 12].

На рисунку 3.2 наведено приклад контролеру.

```
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\View\View
     */
    public function showChart(Request $request)
    {
        $dataSource = $request->input('data_source', 'binance');
        $chartType = $request->input('chart_type', 'pie');
        $data = $this->getDataFromModel($dataSource);
        $chart = $this->createChart($chartType, $data);

        return view('chart', compact('chart'));
    }
```

Рисунок 3.2 – Приклад контролеру

Повний код контроллеру наведено в Додатку Б.

3.4 Laravel-шаблон

Відповідає за візуалізацію даних. Може містити логіку, яка обробляє передані дані за допомогою інструментів мови шаблону blade [11, 12].

На рисунку 3.3 наведено приклад шаблону.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Chart PDF</title>
    {!! Charts::assets() !!}
</head>
<body>
    <div>
        {!! $chart->render() !!}
    </div>
</body>
</html>
```

Рисунок 3.3 – Приклад шаблону

Повний код шаблону наведено в Додатку В.

3.5 Тестування проєкту

3.5.1 Unit-тест

Unit-тестування є процесом перевірки, чи працюють окремі компоненти програмного забезпечення (функції, класи, модулі) вірно і очікувано.

Це важлива частина розробки програмного забезпечення, оскільки добре написані та надійні тести допомагають виявити помилки та сприяють збереженню коректної роботи програми при змінах в коді [5, 11].

Приклади такого тестування наведено на рисунках 3.4 та 3.5.

```
public function testGetDataFromBinance()
{
    // Задаємо значення в .env для тестування Binance
    putenv('BINANCE_API_URL=https://api.binance.com');

    $data = DataVisualization::getData('binance');

    $this->assertIsArray($data);
    $this->assertNotEmpty($data);
}
```

Рисунок 3.4 – Тестування Binance

```
public function testGetDataFromMinfin()
{
    // Задаємо значення в .env для тестування Minfin
    putenv('MINFIN_API_URL=https://api.minfin.com.ua');

    $data = DataVisualization::getData('minfin');

    $this->assertIsArray($data);
    $this->assertNotEmpty($data);
}
```

Рисунок 3.5 – Тестування Minfin

3.5.2 Integration-тест

Інтеграційне тестування в Laravel – це вид тестування, який спрямований на перевірку взаємодії різних компонентів Laravel-додатка, таких як маршрути, контролери, міграції баз даних і інші складові, які працюють разом.

Інтеграційні тести допомагають переконатися, що всі частини додатка правильно взаємодіють одна з одною та виконують очікувані функції [6, 11].

Розглянемо приклад інтеграційного тесту (рис. 3.6).

```
public function testShowChart()
{
    $this->seed();

    $response = $this->get('/chart');

    $response->assertStatus(200)
        ->assertSee('Chart Display');
}

public function testExportPdf()
{
    $this->seed();

    $response = $this->post('/export-pdf');

    $response->assertStatus(200)
        ->assertHeader('Content-Type', 'application/pdf')
        ->assertHeader('Content-Disposition', 'attachment; filename=chart.pdf');
}
```

Рисунок 3.6 – Інтеграційне тестування

3.6 Керівництво користувача

3.6.1 Рівень підготовки користувача

Користувач сайту повинен володіти певною кваліфікацією. Навички користувача для роботи з ПК, та навички роботи з web-браузером. Знайомство з Керівництвом користувача.

3.6.2 Підготовка до роботи

Запуск системи. Доступ до сайту здійснюється через мережу Інтернет за допомогою звичайного web-браузера. Адреса сайту в мережі Інтернет: https://localhost:8080. Для коректної роботи клієнтської частини повинен використовуватися браузер Google Chrome, Mozilla Firefox, Opera, Safari.

При вході на Сайт користувач потрапляє на сторінку з інструментами для генерації відображення. На Сайті розрізняються наступні групи користувачів: користувач – особа, що має доступ до функцій користувача на сторінці сайту.

3.6.3 Керування джерелом вхідних даних

Після входу на сайт, система відображає форму для вибору параметрів візуалізації даних (рис. 3.7). Перший селектор відповідає за джерело вхідних даних. На вибір пропонується два джерела даних, пов’язаних з курсами валют різних країн та криптовалюти (рис. 3.8).

Рисунок 3.7 – Форма параметрів візуалізації даних

Рисунок 3.8 – Перелік джерел вхідних даних

3.6.4 Керування діаграмами

Після вибору джерела вхідних даних, необхідно визначитися з типом діаграми для їх візуалізації. На вибір пропонуються шість видів діаграм (рис. 3.9), що надають можливість відображувати та аналізувати дані з різних кутів.

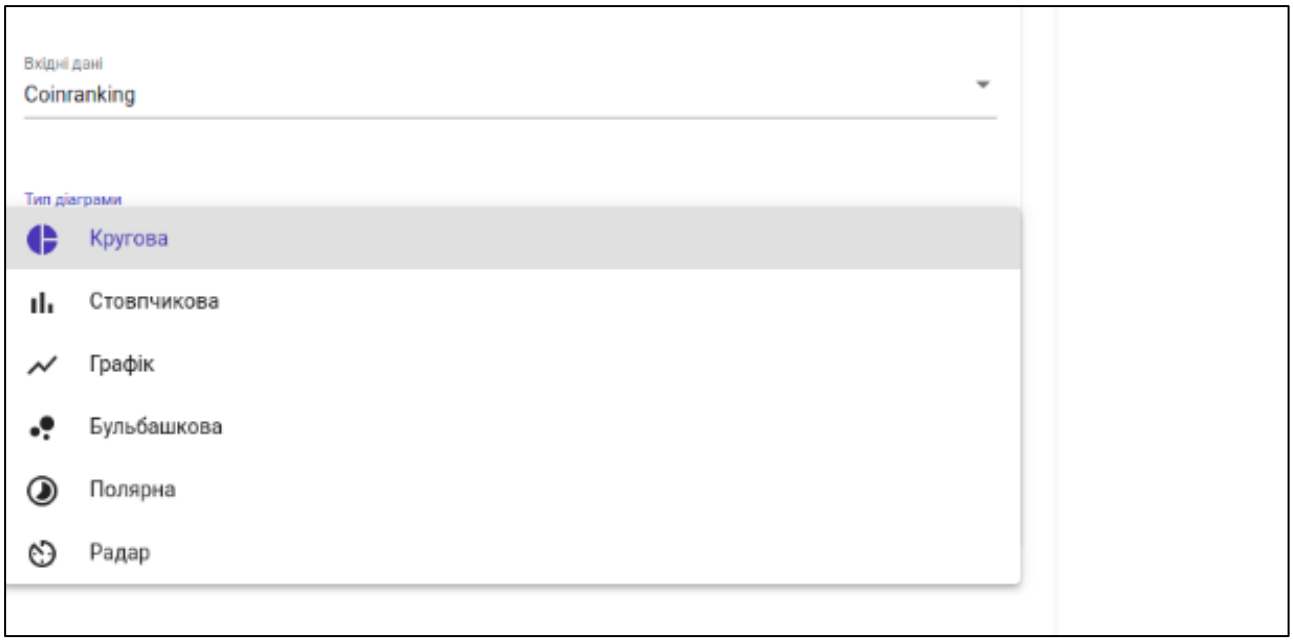

Рисунок 3.9 – Перелік типів діаграм

Кругова діаграма (pie chart) використовується для графічного відображення даних у вигляді кругової частки або сектора. Вона дозволяє ілюструвати пропорції часток даних відносно цілого.

Стовпчикова діаграма (bar chart) використовується для графічного відображення даних у вигляді стовпців, які представляють різні категорії та їх відносні значення. Вона дозволяє порівнювати значення різних категорій та аналізувати їх відносність один до одного.

Діаграма-графік (line chart) використовується для візуалізації залежності між значеннями на осі X (наприклад, час) та відповідними значеннями на осі Y. Вона дозволяє відслідковувати зміну значень впродовж певного періоду часу або відображати будь-яку іншу прогресію.

Бульбашкова діаграма (bubble chart) використовується для візуалізації трьох вимірних даних. Вона подібна до точкової діаграми, але додає третій вимір – розмір (або вагу) елементу. Кожен елемент діаграми представляється у вигляді круга (бульбашки), розмір якого відображає третій вимір даних.

Полярна діаграма (polar chart), також відома як кругова діаграма або вітрова розсіювання, використовується для візуалізації даних, які мають циклічну або кругову структуру. Вона зображає дані на круговій сітці з полюсами, що відповідають різним значенням на осі X, а радіуси кіл відображають значення на осі Y.

3.6.5 Візуалізація відображення

Після вибору джерела вхідних даних та типу діаграми для відображення, потрібно натиснути на кнопку «Візуалізувати дані». Система відобразить обраний тип діаграми, опираючись на джерело вхідних даних, в правій частині сторінки (рис. 3.10).

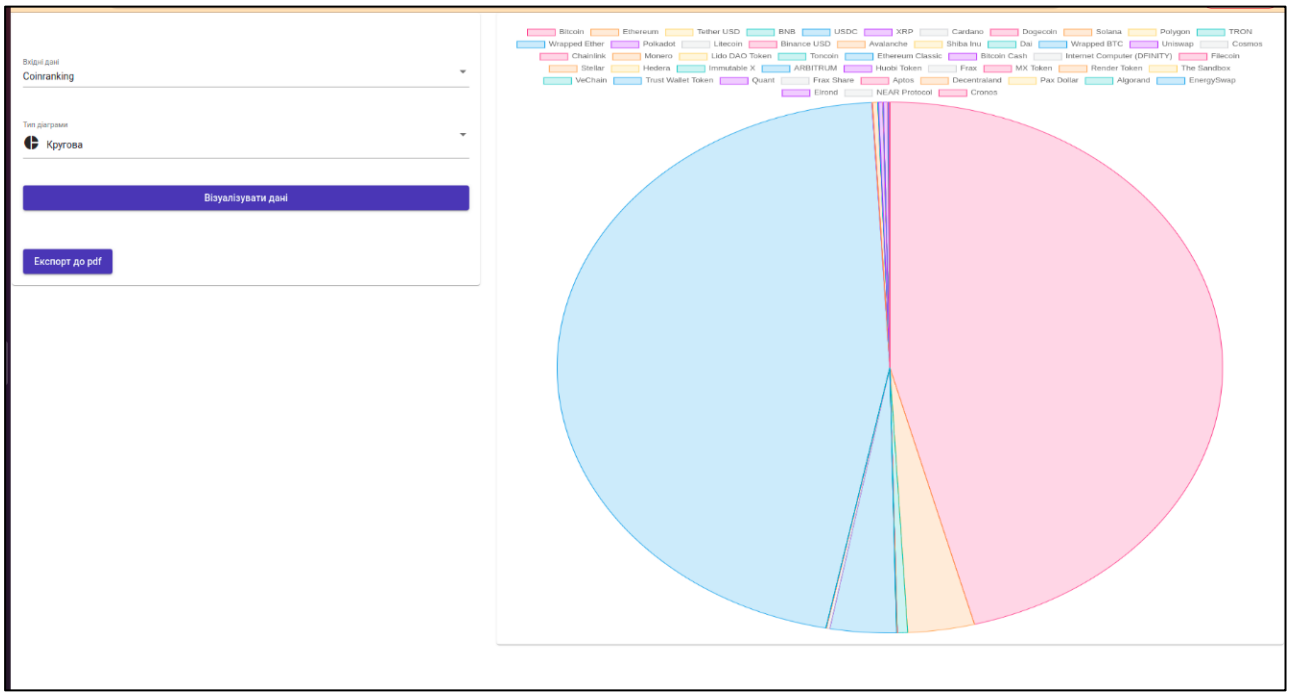

Рисунок 3.10 – Приклад візуалізації даних за допомогою кругової діаграми

Залежно від потреб та актуальності результату відображення, можна змінити параметри та проаналізувати, який вид діаграми найбільш вдало візуалізує вхідні дані. На рисунках 3.11 – 3.13 зображено приклади візуалізації наборів даних за допомогою різних типів діаграм.

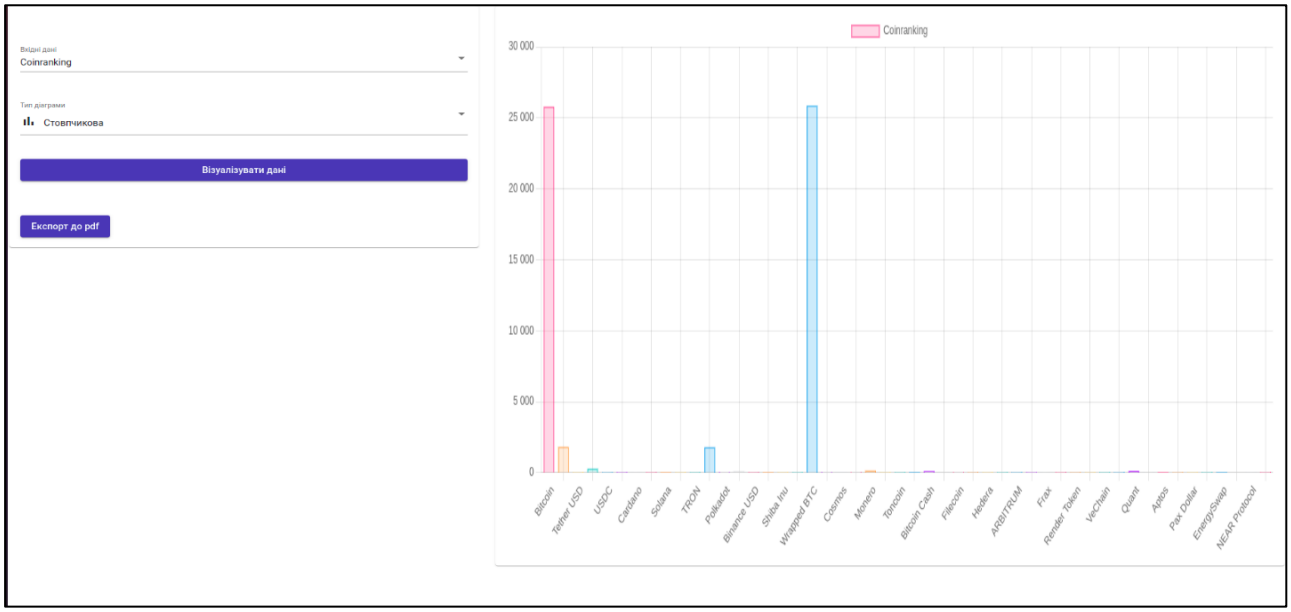

Рисунок 3.11 – Візуалізація даних за допомогою стовпчикової діаграми

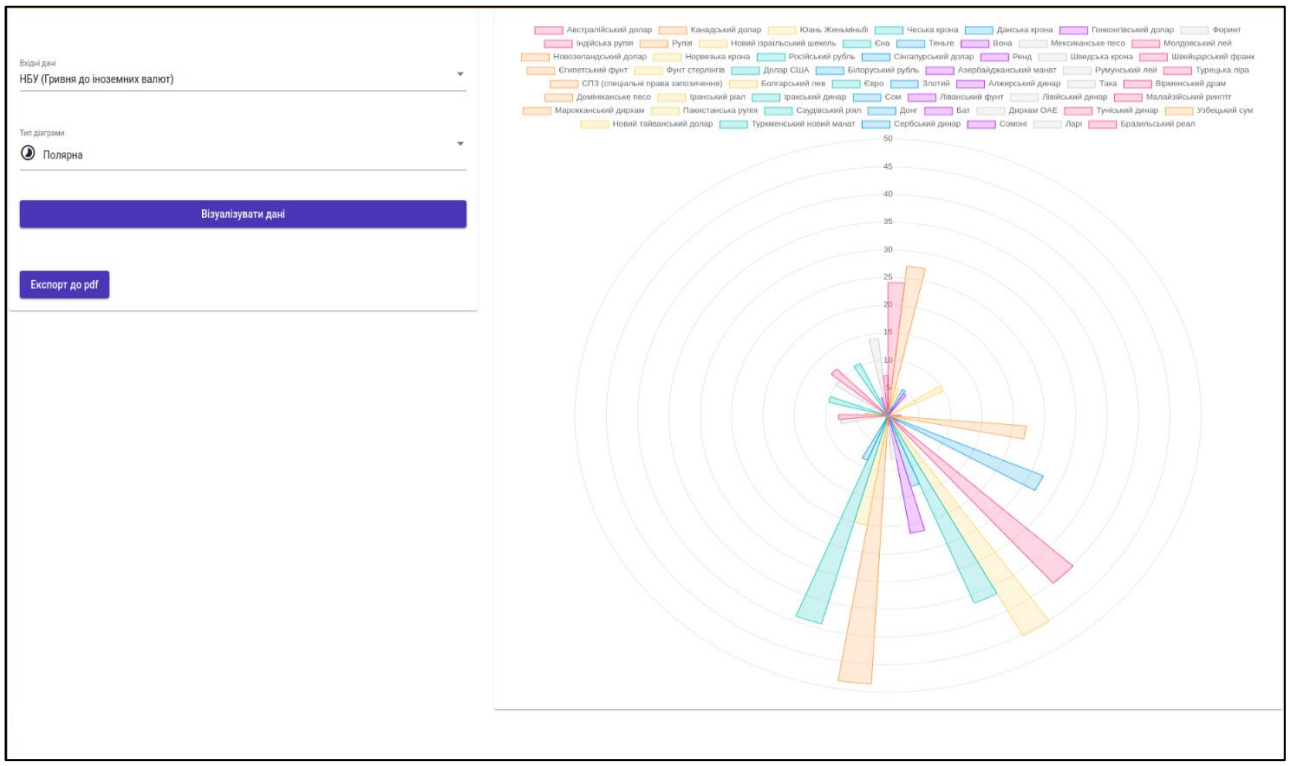

Рисунок 3.12 – Візуалізація даних за допомогою полярної діаграми

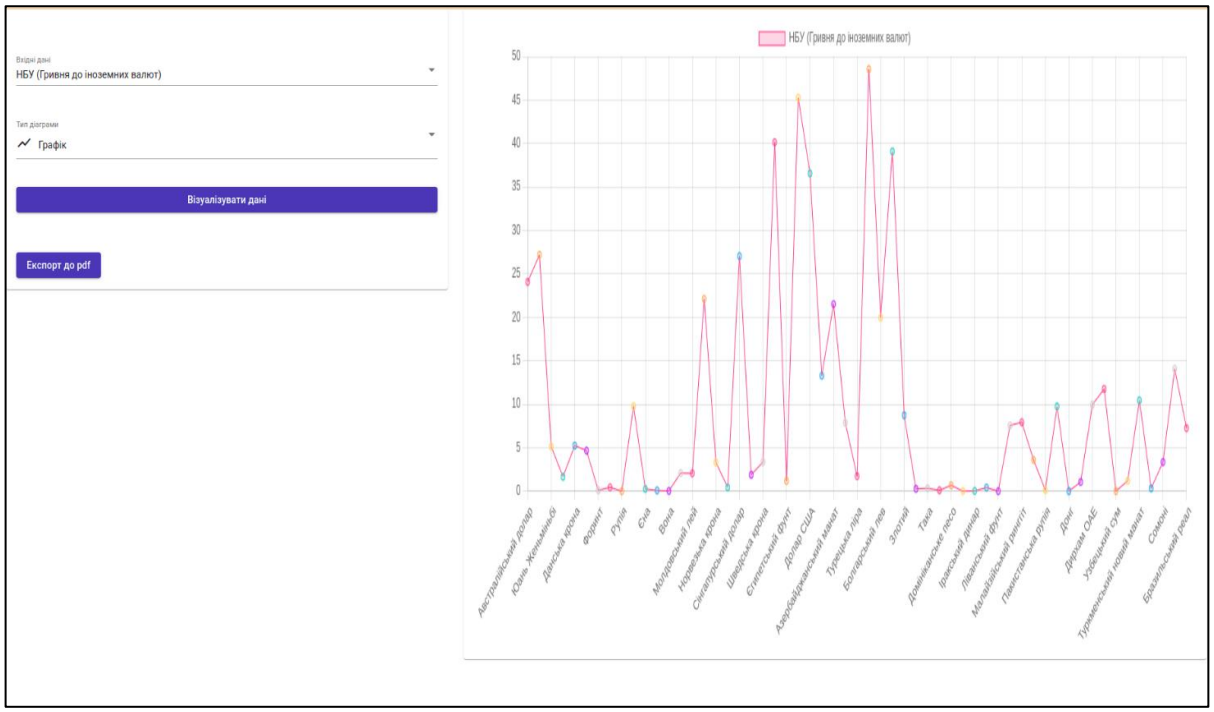

Рисунок 3.13 – Візуалізація даних за допомогою графіку

3.6.6 Фільтрація вхідних даних

Також система надає можливість фільтрації вхідних даних у кругових, полярних да радарних діаграмах (рис. 3.14).

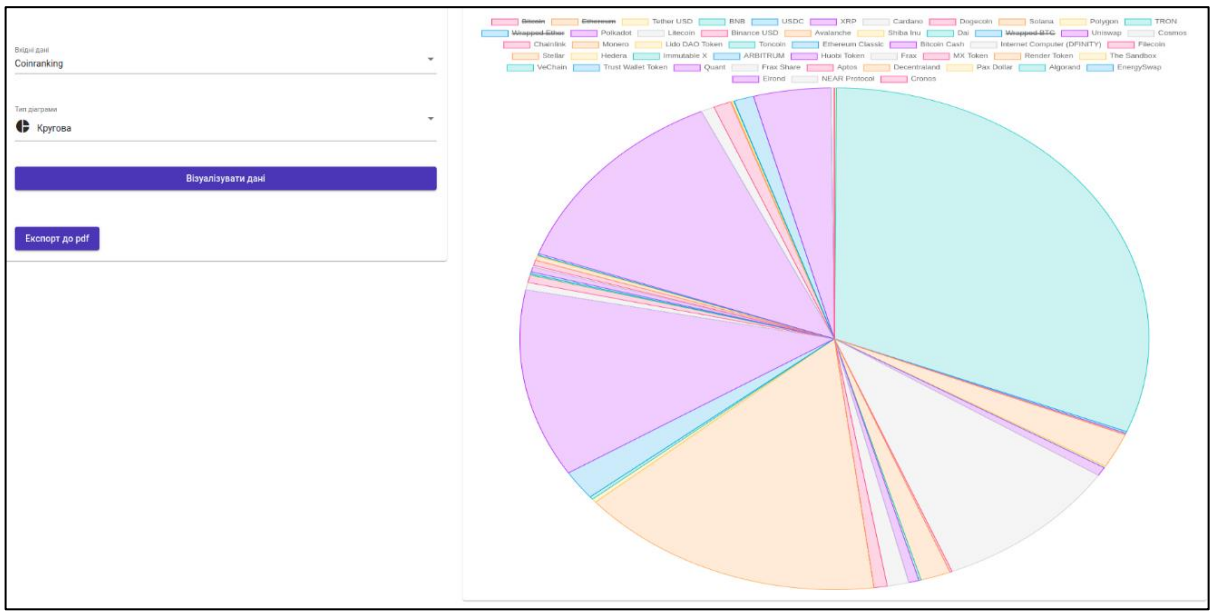

Рисунок 3.14 – Фільтрація даних

Для цього необхідно натиснути на назву елементу, який необхідно приховати або показати.

Прихований елемент не приймає участь у відображенні та його назва перекреслена.

Це дає можливість порівняти та проаналізувати конкретні елементи або з меншою вагою.

3.6.7 Експорт даних до pdf

Для експорту даних до pdf, необхідно натиснути на кнопку «Експорт до pdf», система завантажить файл з короткою назвою джерела даних (див. рис. 3.15).

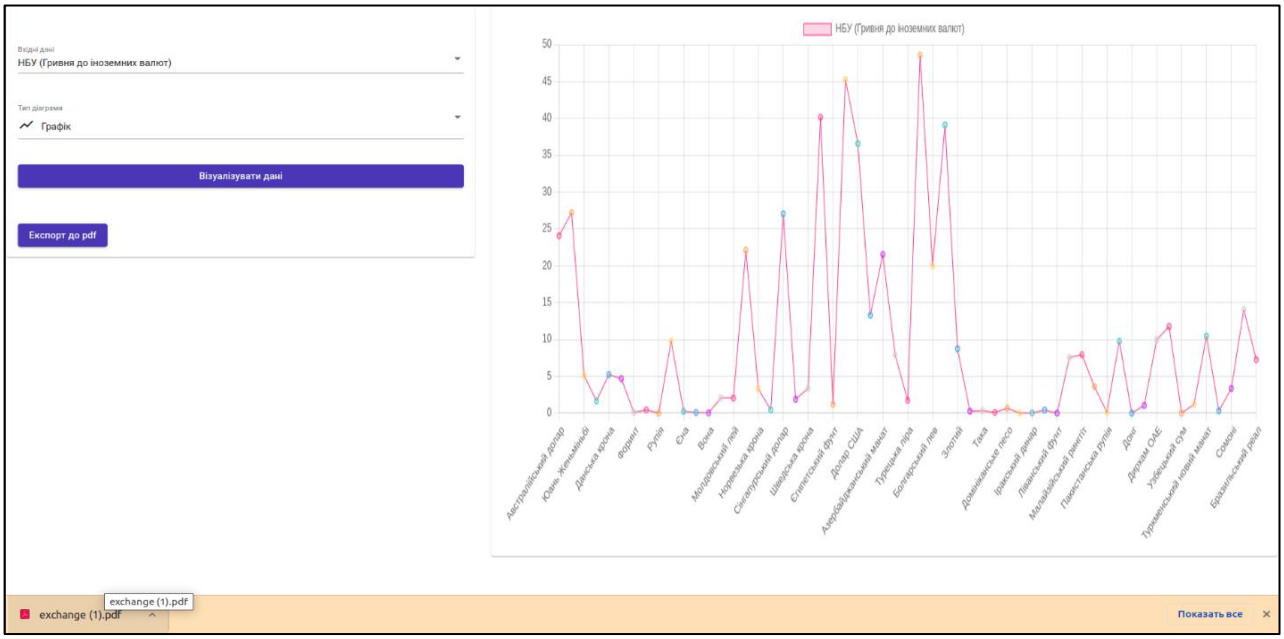

Рисунок 3.15 – Завантаження файлу

Файл відобразиться в новому вікні браузера, користувач може його завантажити на пристрій або роздрукувати (див. рис. 3.16).

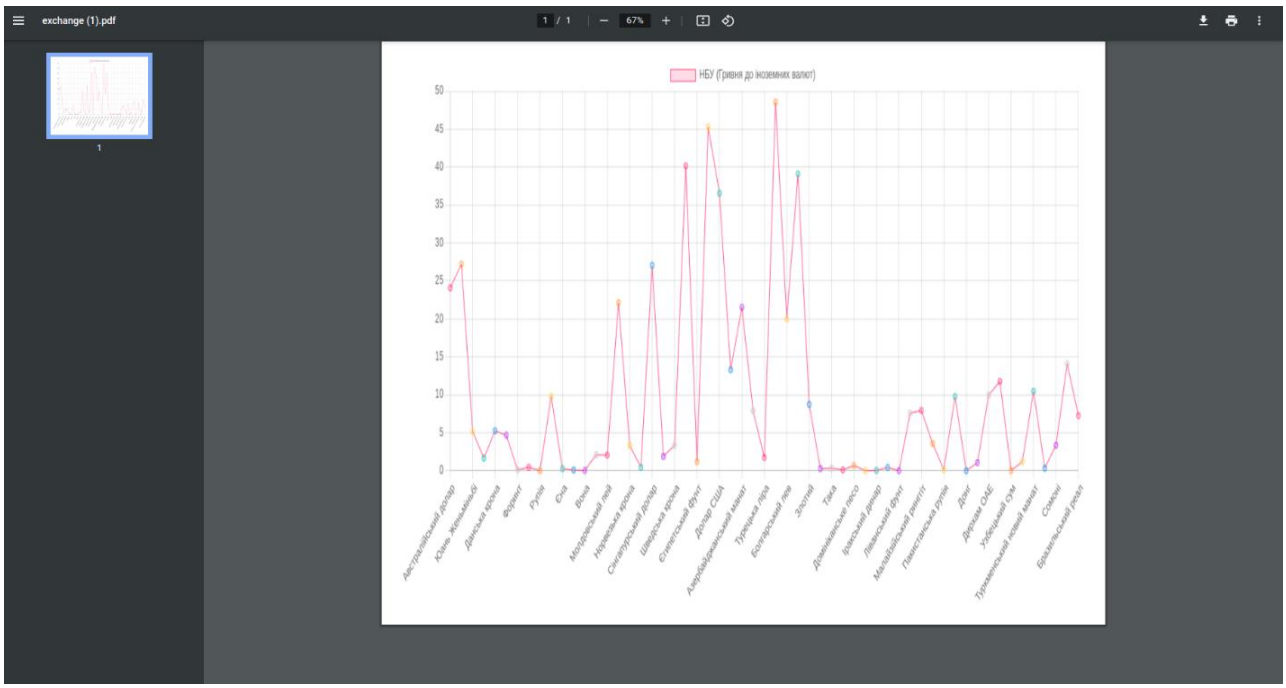

Рисунок 3.16 – Перегляд файлу

ВИСНОВКИ

В результаті роботи було написано технічне завдання на розробку веб-застосунку візуалізації даних. Для створення цієї системи були обрані фреймворк Laravel як і зі сторони клієнта, так і зі сторони сервера, за його широкі можливості у сфері створення web-систем.

У відповідності з метою кваліфікаційної роботи було розроблено веб-застосунок візуалізації даних із застосуванням наступних технологій:

- Laravel для реалізації backend і frontend;
- charts для візуалізації даних.

У відповідності з поставленими задачами були виконані наступні етапи створення системи:

- сформовані вимоги до системи (функціональні та нефункціональні (інтерфейс, кросбраузерність, безпека, продуктивність)), також проведено огляд предметної області та порівняння аналогів;
- спроєктована та побудована структура системи (побудовані діаграми прецедентів, діяльності, послідовності та розгортання; надано детальний опис прецедентів);
- реалізовано веб-застосунок візуалізації даних (наведена інструкція по створенню компонентів системи, надано керівництво користувача та структура проєкту);
- протестована робота системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Scholtens A. Mastering Laravel. Sas155, 2023. 102 p.
2. Ashley A. Consuming APIs in Laravel: Build Robust and Powerful API Integrations For Your Laravel Projects With Ease. Independently published, 2023. 551 p.
3. Chartio Documentation. URL: https://chartio.com/docs/ (дата звернення: 23.08.2023).
4. Correa D., Vallejo P. Practical Laravel: Develop clean MVC web applications. Independently published, 2022. 177 p.
5. Daubois A., Windler C. Clean Code in PHP: Expert tips and best practices to write beautiful, human-friendly, and maintainable PHP. Packt Publishing, 2022. 264 p.
6. Delcione Lopes da Silva. Framework PHP Laravel 8 & AJAX. CBL, 2022. 445 p.
7. Dwivedi R. Laravel Essentials: A Step-by-Step Guide for Web Developers: “Laravel by Example: Building Real-World Applications”. Independently published, 2023. 269 p.
8. Gaitatzis T. Learn REST APIs: Your guide to how to find, learn, and connect to the REST APIs that powers the Internet of Things revolution. BackupBrain Press, 2019. 109 p.
9. Google Data Studio. URL: https://lookerstudio.google.com/overview (дата звернення: 21.08.2023).
10. Infogram. URL: https://infogram.com/ (дата звернення: 23.08.2023).
11. Laravel 8 Developer Documentation. URL: https://laravel.com/docs/8.x/ (дата звернення: 15.09.2023).
12. Laravel Charts Documentation. URL: https://charts.erik.cat/installation.html (дата звернення 10.10.2023).
13. Plotly Chart Studio. URL: https://chart-studio.plotly.com/feed/#/ (дата звернення: 23.08.2023).
14. Power BI. URL: https://www.microsoft.com/en-us/power-

platform/products/power-bi/ (дата звернення: 23.08.2023).

15. Stauffer M. Laravel: Up & Running: A Framework for Building Modern PHP Apps. O’Reilly Media, 2019. 544 p.
16. Tableau Public. URL: https://public.tableau.com/app/discover (дата звернення: 13.08.2023).
17. Visme. URL: https://www.visme.co/ (дата звернення: 23.08.2023).

ДОДАТОК А

Laravel-модель

```
namespace App\Models;

use GuzzleHttp\Client;

class DataVisualization
{
    /**
        * @param  string  $dataSource
     * @return array
     */
    public static function getData($dataSource)
    {
        $apiUrl = ``;
        // Зчитуємо значення з .env
        if ($dataSource === `binance`) {
            $apiUrl = env(`BINANCE_API_URL`);
        } elseif ($dataSource === `minfin`) {
            $apiUrl = env(`MINFIN_API_URL`);
        }
        // Викликаємо API лише якщо маємо URL
        if (!empty($apiUrl)) {
            $client = new Client();
            $response = $client->get($apiUrl);
            return json_decode($response->getBody(), true);
        }
```

```
        // Повертаємо порожній масив у випадку відсутності URL
        return [];
    }
}
```

ДОДАТОК Б

Laravel-контролер

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Charts;
use PDF;
use App\Models\DataVisualization;

class DataVisualizationController extends Controller
{
    /**
     * Створення об`єкта графіка з вибраного типу.
     *
     * @param  string  $chartType
     * @param  array  $data
     * @return \Charts\Builder
     */
    private function createChart($chartType, $data)
    {
        $chart = null;

        switch ($chartType) {
            case `pie`:
                $chart = Charts::create(`pie`, `highcharts`)
                    ->title(`Pie Chart`)
                    ->labels(array_keys($data))
```

```
        ->values(array_values($data));
    break;

case `bar`:
    $chart = Charts::create(`bar`, `highcharts`)
        ->title(`Bar Chart`)
        ->labels(array_keys($data))
        ->values(array_values($data));
    break;

case `line`:
    $chart = Charts::create(`line`, `highcharts`)
        ->title(`Line Chart`)
        ->labels(array_keys($data))
        ->values(array_values($data));
    break;

case `bubble`:
    $chart = Charts::create(`bubble`, `highcharts`)
        ->title(`Bubble Chart`)
        ->labels(array_keys($data))
        ->values(array_map(function ($value) {
            return [$value, $value * 2, $value * 0.5];
        }, array_values($data)));
    break;

case `polar`:
    $chart = Charts::create(`polar`, `highcharts`)
        ->title(`Polar Chart`)
        ->labels(array_keys($data))
```

```
            ->values(array_values($data));
        break;
    default:
        $chart = Charts::create(null, `highcharts`);
  }

  return $chart;
}

/**
 * Отримання даних для графіка з моделі.
 *
 * @param  string  $dataSource
 * @return array
 */
private function getDataFromModel($dataSource)
{
  return DataVisualization::getData($dataSource);
}

/**
 * Відображення графіка на основі введених параметрів користувачем.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\View\View
 */
public function showChart(Request $request)
{
  $dataSource = $request->input(`data_source`, `binance`);
  $chartType = $request->input(`chart_type`, `pie`);
```

```
        $data = $this->getDataFromModel($dataSource);
        $chart = $this->createChart($chartType, $data);

        return view(`chart`, compact(`chart`));
    }
}
```

ДОДАТОК В

Laravel-шаблон

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Data Visualization</title>
    {!! Charts::assets() !!}
</head>
<body>
    <div style="float: left; width: 40%;">
        <form action="{{ url(`/chart`) }}" method="get">
            <label for="data_source">Вхідні дані</label>
            <select name="data_source" id="data_source">
                <option value="binance">Coinranking</option>
                <option value="minfin">НБУ (Гривня до іноземних валют)</option>
            </select>

            <label for="chart_type">Тип даних</label>
            <select name="chart_type" id="chart_type">
                <option value="pie">Кругова</option>
                <option value="bar">Стовпчикова</option>
                <option value="line">Графік</option>
                <option value="bubble">Бульбашкова</option>
                <option value="polar">Полярна</option>
            </select>
```

```
            <button type="submit">Візуалізувати дані</button>
        </form>

        <form action="{{ url(`/export-pdf`) }}" method="post">
            @csrf
            <button type="submit">Експорт до PDF</button>
        </form>
    </div>

    <div style="float: right; width: 60%;">
        <div>
            {!! $chart->render() !!}
        </div>
    </div>
</body>
</html>
```