Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка компонентів збору, обробки та розповсюдження даних інформаційної системи
(тема)

Виконав:
студент II курсу, групи *СПРм-22-1*
Каряка В.В.
(прізвище, ініціали)
Спеціальність *122 – Комп'ютерні науки*
(код і повна назва спеціальності)
Тип програми *освітньо-наукова*
Освітня програма *Системне проектування*
(повна назва освітньої програми)
Керівник *проф. Саваневич В.Є.*
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри СТ _____ *проф. Гребеннік І.В.*
(підпис) (прізвище, ініціали)

2024 р.

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

15.06.2024 *Каряка В.В.*

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.

Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні.

Попередній захист проведено 12 червня 2024 р.

Керівник кваліфікаційної роботи *Саваневич В.Є.*

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*

Рівень вищої освіти другий (магістерський)

Спеціальність 122 – Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова

Освітня програма Системне проектування
(повна назва)

ЗАТВЕРДЖУЮ

Зав. кафедри *СТ*

проф. Гребеннік І.В.

"1" *квітня* *2024 р.*

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Каряка Віктору Володимировичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Розробка компонентів збору, обробки та розповсюдження даних інформаційної системи*
затверджена наказом по університету від " 1 **"** *квітня* **2024р. №** *259 Ст*

2. Термін подання студентом роботи до екзаменаційної комісії *12 червня 2024р.*

3. Вихідні дані до роботи (проекту) *Розробити компоненти збору, обробки та розповсюдження даних інформаційної системи. Система повинна являти собою веб-сайт з інтерфейсом доступу до бази даних. Перелік використовуваних програмних засобів: ОС Microsoft Windows 11, MariaDB, Postman, Visual Studio Code. Технічне забезпечення: IBM-сумісний ПК з ЦП Intel Core i3 та вище.*

4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ 4.2 Аналіз предметної області, що використовує механізми збору, обробки та розповсюдження інформації 4.3 Аналіз подібних систем збору, обробки та розповсюдження інформації 4.4 Аналіз реалізованої системи 4.5 Визначення сфери застосування інформаційної системи взаємодії з даними 4.6 Постановка задачі 4.7 Розробка вимог до інформаційної системи 4.8 Системне проектування 4.9 Визначення функціональних вимог до інформаційної системи збору, обробки та розповсюдження даних 4.10 Розробка моделі потоків даних (Data Flow Diagram) 4.11 Визначений інтерфейс користувача до інформаційної системи взаємодії з даними 4.12 Діаграма варіантів використання інформаційної системи збору, обробки та розповсюдження даних 4.13 Діаграма послідовності дій (Sequence diagram) 4.14 Проектування діаграми класів системи збору, обробки та розповсюдження даних 4.15 Вибір технологій проектування системи збору, обробки та розповсюдження інформації 4.16 Вибір

архітектури проектування інформаційної системи 4.17 Обґрунтування вибору СУБД 4.18 Вибір структури проектування інформаційної системи 4.19 Моделювання ER-діаграми та створення бази даних інформаційної системи взаємодії з даними 4.20 Розробка моделей та контролерів для інформаційної системи взаємодії з даними 4.21 Розробка інтерфейсу користувачів та створення взаємодії з моделями і контролерами 4.22 Висновки 4.23 Перелік джерел посилання

5. **Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)** 5.1 Контекстна діаграма головного бізнес-процесу, 5.2 Декомпозиція поголовного бізнес-процесу IC 5.3 Декомпозиція бізнес-функції «Взаємодія замовника» 5.4 Декомпозиція бізнес-функції «Взаємодія дослідника» 5.5 Діаграма варіантів використання системи збору, обробки та розповсюдження інформації 5.6 Діаграма послідовності дій для прецеденту «створення замовлення» 5.7 Діаграма класів 5.8 Графічне представлення MVC архітектури 5.9 Технології проектування 5.10 Структура бази даних 5.11 Сторінка реєстрації користувача 5.12 Сторінка введення платіжної інформації 5.13 Створення нового дослідження 5.14 Обладнання 5.15 Список створених досліджень замовника

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів атестаційної роботи	Термін виконання етапів роботи	Примітка
1.	*Отримання завдання кваліфікаційної роботи*	*01.04.2024*	*Виконано*
2.	*Аналіз предметної області, літератури та аналогів існуючих систем з теми атестаційної роботи*	*02-06.04.2024*	*Виконано*
3.	*Вибір мови програмування та системи управління базами даних для розробки функціоналу додатка*	*06-09.04.2024*	*Виконано*
4.	*Розробка вимог до інформаційної системи біржа спостережень*	*09-14.04.2024*	*Виконано*
5.	*Розробка програмного коду інформаційної системи*	*14.04-10.05.2024*	*Виконано*
7.	*Розробка «Посібника користувача»*	*11-18.05.2024*	*Виконано*
8.	*Оформлення пояснювальної записки та документація програмного коду*	*18-28.05.2024*	*Виконано*
9.	*Оформлення графічної частини презентаційних матеріалів, комп'ютерних матеріалів для захисту атестаційної роботи*	*28.05-5.06.24*	*Виконано*
10.	*Представлення на рецензування*	*12.06.24*	*Виконано*

Дата видачі завдання *01.04.2024*

Студент _____ Каряка В.В.
(підпис)

Керівник роботи _____ проф. Саваневич В.Є.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка з кваліфікаційної роботи містить:121 с., 2 табл., 33 рис., 4 додатки, 23 джерел

КОМПОНЕНТ ЗБОРУ, ОБРОБКА, РОЗПОВСЮДЖЕННЯ ДАНИХ, БАЗА ДАНИХ, ІНФОРМАЦІЙНА СИСТЕМА, ПІДПИСНИК, ВИДАВЕЦЬ, ІНТЕРФЕЙС КОРИСТУВАЧА, ОБ’ЄКТ, АРХІТЕКТУРА, ШАБЛОН.

Об’єктом дослідження є розробка компонентів збору, обробки та розповсюдження даних для інформаційної системи.

Предметом дослідження є реалізація компонентів для збору, обробки та розповсюдження даних в інформаційній системи типу «Видавець – підписник», що не потребує прямого зв’язку між підписником (замовник) та видавцем (спостерігач).

Методи дослідження включають в себе аналіз потреб користувачів, дослідження існуючих технологій та методів обробки даних, впровадження та тестування архітектурних рішень та вивчення і впровадження вимог для масштабованості, ефективності та безпеки системи.

В ході дослідження було впроваджено вимоги та задачі до інформаційної системи:

– проведення аналізу предметної області та розробка вимог;

– розробка інтерфейсу користувача;

– створення триланкової архітектури «Клієнт – База Даних – Сервер»;

– реалізація шаблону проектування «Видавець – підписник».

Результатом кваліфікаційної роботи є створені компоненти та модулі, що реалізують механізми збору, обробки та розповсюдження інформації з використанням інтерфейсу користувача для легкого використання.

ABSTRACT

The explanatory note on the qualification work contains: 121 pages, 2 tables, 33 figures, 4 appendices, 23 sources

COMPONENT COLLECTION, PROCESSING, DATA DISTRIBUTION, DATABASE, INFORMATION SYSTEM, SUBSCRIBER, PUBLISHER, USER INTERFACE, OBJECT, ARCHITECTURE, TEMPLATE.

The object of the research is the development of data collection, processing and distribution components for the information system.

The subject of the study is the implementation of components for data collection, processing and distribution in the "Publisher-subscriber" type information system, which does not require direct communication between the subscriber (customer) and the publisher (observer).

Research methods include analysis of user needs, research of existing technologies and methods of data processing, implementation and testing of architectural solutions, and study and implementation of requirements for scalability, efficiency and system security.

In the course of the study, requirements and tasks for the information system were implemented:

– analysis of the subject area and development of requirements;

– user interface development;

- creation of a three-link architecture "Client - Database - Server";

- implementation of the "Publisher - Subscriber" design template.

The result of the qualification work is the created components and modules that implement mechanisms for collecting, processing and distributing information using a user interface for easy use.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ, ТЕРМІНІВ 8

ВСТУП 9

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 10

1.1 Аналіз предметної області, що використовує механізми збору, обробки та розповсюдження інформації 10

1.2 Аналіз подібних систем збору, обробки та розповсюдження інформації 12

1.3 Аналіз реалізованої системи 16

1.4 Визначення сфери застосування інформаційної системи взаємодії з даними 18

1.5 Постановка задачі 18

2 РОЗРОБКА ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ 20

2.1 Системне проектування 20

2.2 Визначення функціональних вимог до інформаційної системи збору, обробки та розповсюдження даних 22

2.3 Розробка моделі потоків даних (Data Flow Diagram) 33

2.4 Визначений інтерфейс користувача до інформаційної системи взаємодії з даними 35

2.5-Діаграма варіантів використання інформаційної системи збору, обробки та розповсюдження даних 37

2.6 Діаграма послідовності дій (Sequence diagram) 40

2.7 Проектування діаграми класів системи збору, обробки та розповсюдження даних 42

3 ОПИС ПРИЙНЯТИХ РІШЕНЬ, ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 45

3.1 Вибір технологій проектування системи збору, обробки та розповсюдження інформації 45
3.2 Вибір архітектури проектування інформаційної системи 46
3.3 Обґрунтування вибору СУБД 48
3.4 Вибір структури проектування інформаційної системи 49
3.5 Моделювання ER-діаграми та створення бази даних інформаційної системи взаємодії з даними 50
3.6 Розробка моделей та контролерів для інформаційної системи взаємодії з даними 54
3.7 Розробка інтерфейсу користувачів та створення взаємодії з моделями і контролерами 60
ВИСНОВОК 71
АПРОБАЦІЇ 72
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 73
ДОДАТОК А. 75
ДОДАТОК Б 86
ДОДАТОК В. 99
ДОДАТОК Г 120

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

Видавець – це особа, яка проводить спостереження відповідно завданням

Підписник – особа, яка подає замовлення на обробку даних

БД – база даних

СУБД – система управління базами даних

ІС – інформаційна система

ПЗ – програмне забезпечення

UML – уніфікована мова моделювання

JSON – текстовий формат обміну даними

API – Application Programming Interface

ВСТУП

Система збору, обробки та розповсюдження даних інформаційної системи представляє собою комплекс технічних та програмних засобів для вирішення ряду запитів автоматизованим способом за допомогою створення замовлень в інформаційній системі та отримання результатів дослідження створюваного запиту на основі обробки даних спостерігачем та надсилання кінцевого результату до інформаційної системи.

Головним аспектом системи, що тісно пов’язана з обробкою даних є база даних. Правильний вибір СУБД та створення бази може вплинути на швидкість роботи та ефективність збереження та цілісного доступу до даних [12]. Обираючи систему управління базами даних для проекту, потрібно враховувати кілька ключових аспектів, що позитивно вплинуть на успішність розробки та її масштабованість.

Метою дослідження є аналіз предметної області та проектування трирівневої архітектури «База даних – Сервер – Клієнт» для інформаційної системи взаємодії з даними.

Використання обраного шаблону дозволяє чітко розділити різні аспекти системи на компоненти, що сприятиме підтримці та розвитку програмного продукту. Цей підхід сприяє масштабованості проекту та дозволяє легко змінювати або оновлювати компоненти системи без перерви в її роботі. Крім того, така архітектура сприяє безпеці та ефективності обробки даних, оскільки дозволить легко оптимізувати будь-які компоненти системи.

Результатом роботи реалізовано веб-додаток з інтерфейсом користувачів та компонентами, що реалізовують механізм збору, обробки та розповсюдження даних в інформаційній системі. Механізми обробки дозволять ефективно та швидко взаємодіяти з системою головним акторам, що дозволить просунути додаток на ринку збору та обробки інформації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області, що використовує механізми збору, обробки та розповсюдження інформації

Головними аспектами у розробці систем та компонентів для неї є ретельний аналіз предметної області, що дозволить визначити потреби користувачів та бізнесу. Цей процес включає визначення функціональних та нефункціональних вимог та аналіз контексту використання системи. Тільки глибоке розуміння цих аспектів допоможе створити інформаційну систему, яка буде відповідати потребам та очікуванням користувачів та забезпечить успішну реалізацію бізнес-процесів та бізнес-функцій.

Розроблюючи систему, необхідно зрозуміти, що представляють механізми збору, обробки та розповсюдження даних в ІС і способи їх використання в створюваному додатку.

Механізм збору даних – це процес, за допомогою якого система отримує та зберігає інформацію для подальшої обробки та аналізу. В ІС механізм може включати такі елементи:

– введення даних користувачами через веб-інтерфейс;

– імпортування початкових та оброблених файлів в систему;

– інтеграції з іншими системами за допомогою API.

Механізм збору даних в інформаційній системі має бути добре розробленим та забезпечувати надійність, цілісність та безпеку отриманих даних. Також важливо враховувати вимоги до швидкості та ефективності збору, особливо в великих та високонавантажених системах.

Механізм обробки даних – це процес дослідження та аналізу об’єкта або купи об’єктів з урахуванням критеріїв завдання, на яке було оформлено підписку для можливості його виконання. Нарешті, остаточні результати

обробки можуть бути візуалізовані та відображені користувачам у зручній формі для подальшої взаємодії з ними.

Механізм розповсюдження даних – включає в себе передачу обробленої інформації як результат дослідження з можливістю завантаження візуального представлення роботи спостерігача. Після цього дані розповсюджуються до призначених замовників, забезпечуючи доступність і цілісність даних.

Проводячи покроковий аналіз предметної області, що використовує зазначені вище механізми потребує докладного вивчення та аналізу різноманітних аспектів з метою зрозуміти потреби, вимоги та особливості галузі, що дозволять програмістам та бізнес-аналітикам сформувати зрозумілий опис кожної функції системи. Саме для реалізації інформаційної системи необхідно поставити та виконати завдання:

– планування та аналіз вимог – на цьому етапі визначаються потреби та вимоги користувачів та створюється план розробки системи;

– проектування системи – обирається архітектура, технології та інструменти майбутнього проекту;

– розробка програмного забезпечення – розроблюється програмне забезпечення відповідно до вимог та специфікацій;

– тестування і виправлення – проводиться тестування системи для перевірки її працездатності та виявлення можливих помилок;

– впровадження та підтримка – проводиться впровадження системи в роботу та надається можливість використання системи користувачам.

Отже, саме для успішної реалізації інформаційної системи важливо проводити планування та виконувати ряд поставлених задач, що враховують в себе різноманітні аспекти розробки та ефективності функціонування системи. Також необхідно детально оцінити потреби користувачів та бізнесу для точного визначення вимог до системи. Проводити постійне виконання завдань, щодо оптимізації структури бази даних та запитів до неї для забезпечення високошвидкісного доступу та обробки великої кількості інформації.

Додатково, слід приділити увагу на оцінку обладнання та інфраструктури, що підтримує розроблену інформаційну систему та забезпечує надійність і високу продуктивність взаємодії акторів з функціоналом системи.

У контексті успішної реалізації ІС, важливим фактором є встановлення системи моніторингу, що дозволить вчасно виявити проблеми з обробкою даних та ефективністю виконання запитів користувачів.

1.2 Аналіз подібних систем збору, обробки та розповсюдження інформації

На просторах інтернету було знайдено інформацію про конкурента для нашої системи, що аналізувала та обробляла інформацію об’єктів сонячної системи. CoLiTec – це система, що тісно пов’язана з астрономією, для використання методів виявлення та моніторингу об’єктів, які зближуються із Землею. Ідея полягала в створенні системи, яка допоможе дослідити космічний об’єкт, що зближується із Землею. Дуже важливими етапом боротьби – це генерація попереджень при виявленні небезпечних об’єктів, що рухаються в напрямку планети.

Після формування «Входу» починається процес обробки цифрових кадрів. Саме виявлення об’єктів, що рухаються, відбувається в декілька етапів. Цикл обробки об’єктів за допомогою програмного забезпечення CoLiTeC представлений на рисунку 1.1 у вигляді блок-схеми.

Наступним кроком система вираховує масив яскравостей кожної серії кадрів отриманих від обсерваторії, що досліджувала космічний простір протягом ночі. Після цього проводиться аналіз та обробка отриманих кадрів та вирівнювання яскравості фону зображення за допомогою службових плагінів в ІС

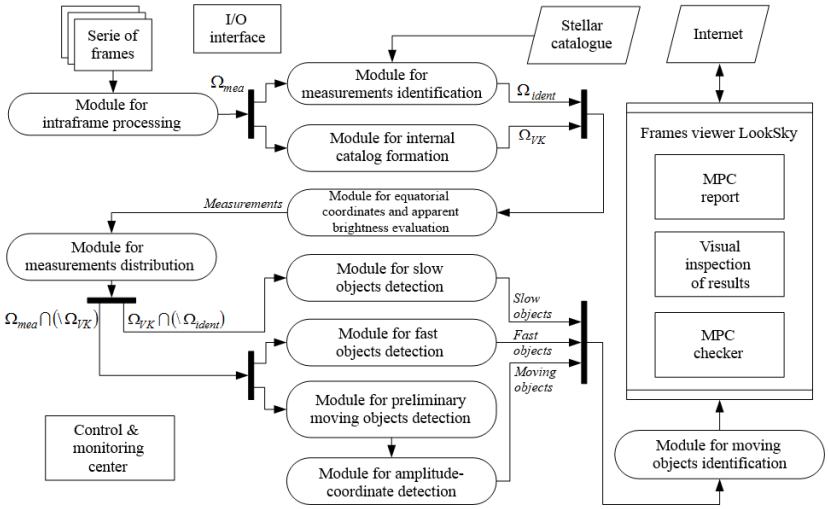

Рисунок 1.1 – Цикл обробки за допомогою ПЗ CoLiTec.

Надалі використовується внутрішньо кадрова обробка, а саме: оцінка положення об’єктів у фіксовані моменти часу. І завершальний етап: між кадрова обробка. Вона використовується для виявлення об'єктів та оцінки параметрів траєкторій їхнього руху об'єктів [1].

Окрім розбору алгоритму роботи необхідно розглянути інтерфейс користувача та його можливості, для того щоб визначити вимоги користувачів та бізнесу до розроблювальної інформаційної системи.

Відкриваючи додаток буде відображено головне меню з інтерфейсом доступу до функцій системи, та коротким описом, що представляє з себе додаток CoLiTec представлений на рисунку 1.2

Рисунок 1.2 – Головне меню додатку CoLiTec

Перелік доступних функцій, що здійснюється через головний інтерфейс вікна (рис. 1.2):

– перехід із режиму обробки на головну сторінку;

– опис програмного забезпечення FrameSmooth з можливістю переходу на офіційний веб-сайт CoLiTec;

– вибір режиму вирівнювання яскравості;

– вибір режиму створення та використання службових майстер кадрів;

– вибір режиму OLDAS для обробки кадрів в режимі реального часу;

– вибір режиму SCRIPT для вирівнювання кадрів із заданою послідовністю операцій;

– панель налаштувань, що згортається;

– шлях до активного конфігураційного файлу;

– скидання всіх параметрів за замовчуванням;

– завантаження конфігураційного файлу;

– вибір мови інтерфейсу;

– вибір кількості ядер для обробки.

Наступне вікно «Яскраве вирівнювання зображень» містить в собі наступні елементи інтерфейсу, що представлено на рисунку 1.3:

– шлях до вхідних файлів – це шлях до файлів, що обробляються;

– шлях до вихідної директорії – це шлях до папки виведення оброблених файлів;

– оброблювати каналами RGB дозволяє обробляти кольорове зображення. На фільтрацію сірих зображень та fits-файлів не впливає;

– розмір маски фільтра – це ширина та висота квадратної маски медіанного фільтра;

– коефіцієнт бінування, операція бінування дозволяє зменшити обсяг використовуваної оперативної пам’яті та час обробки зображень великого об’єму;

– вихідна маска – задається маска для імен файлів з обробленим зображенням.

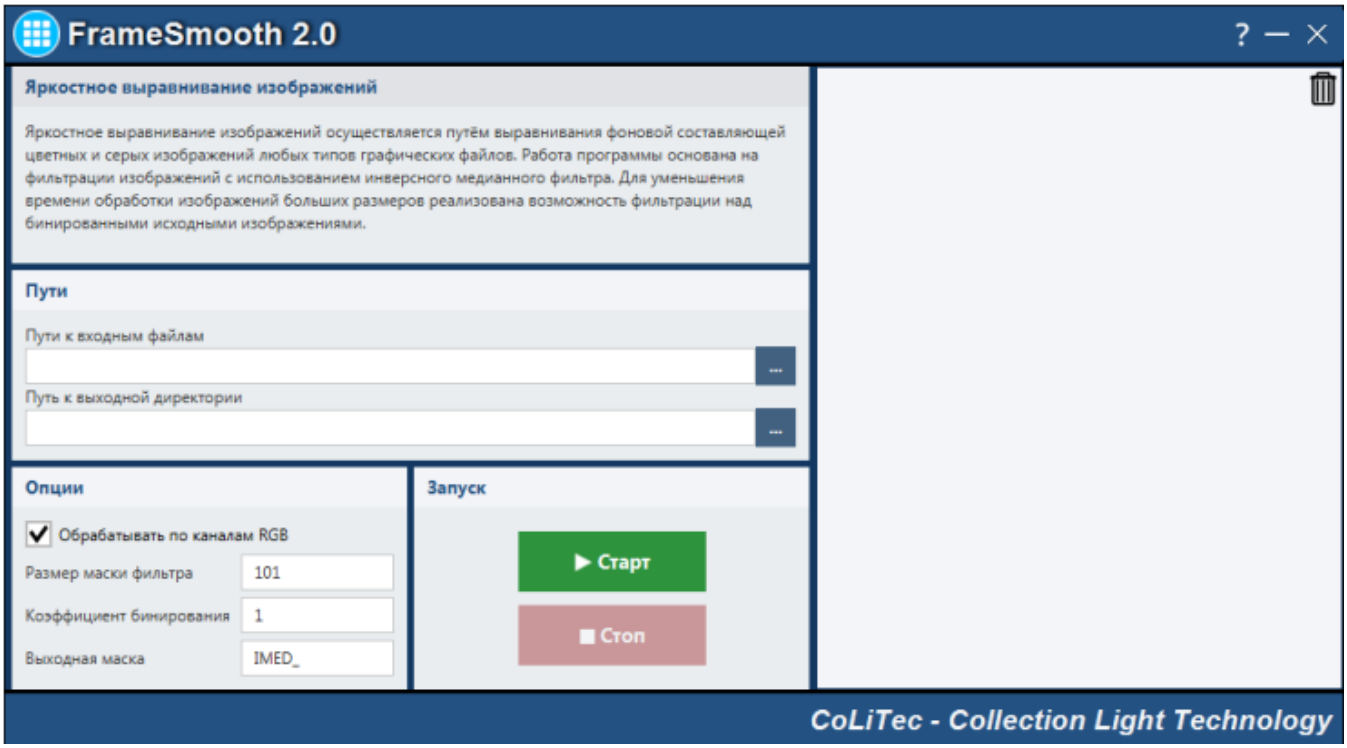

Рисунок 1.3 – інтерфейс користувача «Яскраве вирівнювання зображень»

Розглянемо наступне вікно інтерфейсу та його функції, які призначені для взаємодії з «Конвертування зображення у FIT формат». Даний інтерфейс наведено на рисунку 1.4.

Вікно «Конвертування зображення у FIT формат» містить в собі наступні функції:

– шляхи до вхідних файлів – це шлях до файлів, що оброблюються;

– шлях до вихідної директорії – це шлях до папки виведення оброблених файлів;

– виконати інверсію включає в себе функцію інвертування зображення при конвертації.

– створити кроп – активує функцію створення кропу, що задається координатами верхньої лівої вершини, ширини та висоти;

– параметри кропу – поля для встановлення початкового положення, ширини та висоти.

Рисунок 1.4 – інтерфейс користувача «Конвертування зображення у FIT формат»

Після аналізу інформаційної системи CoLiTec та її функціоналу, стає очевидним необхідність визначення вимог для подальшої розробки інформаційної системи. Вимоги до системи мають бути чітко сформовані та відображати потреби користувачів та бізнес-вимоги [5].

Окрім того, важливо визначити до інтерфейсу користувача, оскільки зручність використання системи визначається його зрозумілістю. Отже, інтерфейс користувача повинен бути легко зрозумілим, легким у навігації та мати привабливий дизайн для користувачів системи.

1.3 Аналіз реалізованої системи

Створена інформаційна система має певний перелік бізнес-функцій, які необхідні для її функціонування за створеним шаблоном проектування «Видавець – підписник». Обраний шаблон дозволяє створювати ефективний механізм для реалізаціє взаємодії між об’єктами без прямого зв’язку між ними. Основна ідея полягає в тому, що один об’єкт (видавці) надсилає повідомлення

на дослідження даних, а інший об’єкт (підписники) отримують ці повідомлення та реагують на них. Механізми та актори, що реалізовані в системі:

– видавець (publisher) – це об’єкт, що створює повідомлення заповнюючи відповідну форму даними для створення нового дослідження за певними критеріями;

– підписник (subscriber) – це об’єкт, який оформлює підписку на виконання певного дослідження, проводячи його за критеріями видавця;

– повідомлення (event) – це подія або сигнал, який передається від видавця до підписника та навпаки. Повідомлення містить інформацію про подію або зміну;

– механізм підписки та відписки (subscribe/unsubscribe) – це методи, які дозволяють відписуватись та підписуватись на отримання повідомлень від видавця;

– Механізм збору даних – це процес, за допомогою якого система отримує та зберігає інформацію для подальшої обробки та аналізу;

– Механізм обробки даних – це процес дослідження та аналізу об’єкта або купи об’єктів з урахуванням критерій завдання, на яке було оформлено підписку для можливості його виконання;

– Механізм розповсюдження даних – включає в себе передачу обробленої інформації як результат дослідження з можливістю завантаження візуального представлення роботи спостерігача.

Перевагою шаблону проектування «Видавець – підписник» включає в себе зменшення кількості залежностей між об’єктами, спрощення розширення, модифікації та підвищення її масштабованості. Це допомагає створити більш гнучкий та легкий код, через те, що зміна в одній частині коду не потребує автоматично перероблювати інший.

Завдяки розділенню обов’язків між видавцем та підписниками, розробники можуть зосередитись на реалізації певної функціональності, що полегшить розробку та підтримку системи.

Узагальнюючи, використання шаблону проектування «Видавець – підписник» дозволяє покращити архітектуру програмного забезпечення, зробити код більш структурованим та легким у розумінні, що сприяє підвищенню продуктивності та зменшенню ризиків при розробці та підтримці проекту.

1.4 Визначення сфери застосування інформаційної системи взаємодії з даними

Інформаційна система, що включає в себе збір, обробку та поширення даних має широкий спектр застосувань, починаючи від особистих досліджень і закінчуючи комерційними замовленнями на аналіз певної категорії даних.

Мета інформаційної системи збору, обробки та розповсюдження даних полягає у створенні ефективної та надійної інфраструктури для збору різноманітних даних, їх обробки з метою аналізу та перетворення в зрозумілу інформацію [8]. У цьому контексті необхідне впровадження механізмів для структурування, фільтрації та розповсюдження даних з метою їхнього подальшого аналізу та використання.

Однак перевагою даної системи є її універсальність, яка дозволяє використовувати її в різних галузях, починаючи від наукових досліджень космічного простору і закінчуючи вивченням різноманітності метеликів в певному регіоні.

1.5 Постановка задачі

Визначені завдання та вимоги до інформаційної системи були успішно поставлені та проаналізовані. Обрані задачі допоможуть побудувати ефективну систему для роботи з даними, а саме: обробка, збір та розповсюдження.

Для досягнення поставлених задач необхідно виконати наступні задачі:

– розробити вимоги до інформаційної системи;

– розробити інтерфейс доступу користувачів до інформаційної системи;

– розробити бізнес-логіку системи, що допоможе взаємодіяти користувачу із додатком;

– розробити серверну частину застосунку, проектування бази даних;

– розробити моделі, що будуть зберігати інформацію та передавати її до інтерфейсу користувача;

– розробити функціонал підписки та відписки на виконання завдання;

– розробка сторінки створення повідомлення, яка відповідає за введення інформації про необхідне дослідження;

– розробка функціоналу, що буде доступний включно для адміністраторів системи, які будуть проводити сертифікацію обладнання;

– розробка функціоналу для спостерігачів системи.

Визначені задачі, які були описані вище, будуть ключовими в процесі створення ефективної та надійної системи для роботи з даними. Розроблена система має відповідати не лише вимогам користувачів, а й вимогам бізнесу, що вважається важливим аспектом при розробці. Дотримання вимог, щодо виконання цих кроків забезпечить високий рівень функціональності та ефективної роботи системи.

2 РОЗРОБКА ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Системне проектування

Переходячи до етапу системного проектування інформаційної системи, що тісно пов’язана з даними. Для успішного системного проектування було успішно проведено аналіз предметної області та обрано майбутні технології для проектування системи. Саме для визначення вимог до інформаційної системи буде використано уніфіковану мову моделювання UML (Unified Modeling Language), за допомогою якої можна створити графічне зображення процесів та об’єктів, що присутні в системі [15].

Для створення клієнтської та серверної частини застосунку було обрано мову програмування PHP та базу даних MySQL.

Створюючи серверну частину було використано середовище виконання коду Nginx, який являє собою високопродуктивний веб-сервер і реверсивний проксі-сервер, який також може виконувати функції балансування навантаження, HTTP-кешування та поштового проксі-сервера [9]. Графічне представлення використання платформи Nginx та PHP представлене на рисунку 2.1

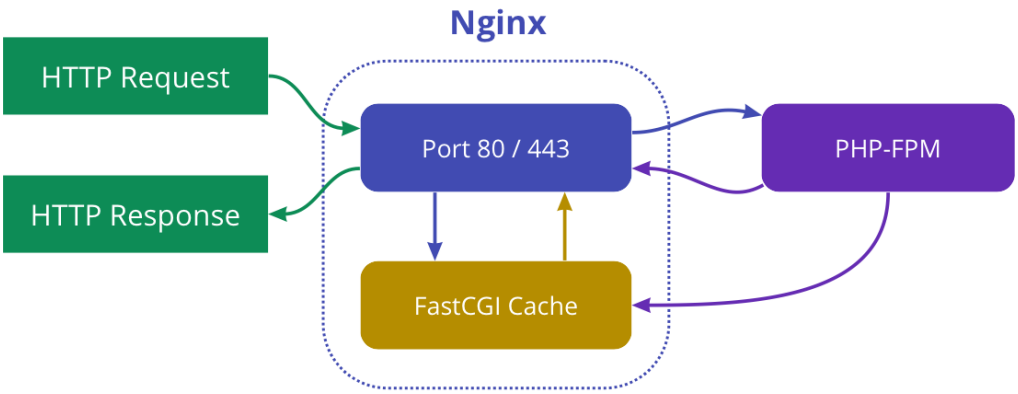

Рисунок 2.1 – Графічне представлення використання Nginx та PHP

Будь-яка задача має бути розподілена на більш менші задачі, тому далі буде наведено основні етапи розділу системного проектування:

– аналіз вимог. На цьому етапі включають потреби користувачів, визначаються функціональні та нефункціональні вимоги до системи, що включають в себе бізнес потреби, вимоги користувачів та ідентифікацію головних функцій системи;

– проектування архітектури системи. Цей етап відповідає за підбір структури системи, включаючи компоненти та взаємозв’язки між собою;

– проектування бази даних. Створення структури бази даних, включаючи таблиці, поля, типи даних та взаємозв’язки між ними;

– проектування інтерфейсів користувача. Реалізація інтерфейсів з якими користувачі будуть взаємодіяти з системою та її функціоналом;

– проектування програмного забезпечення. На цьому етапі розроблюється детальний план системи та її компонентів;

– тестування системи. Проведення тестування системи, щоб перевірити відповідність до поставлених вимог та провести виправлення коду за необхідністю;

– документація. Ключові аспекти проектування необхідно задокументувати, для швидкого ознайомлення із системою.

Наведені етапи необхідні для створення високоякісної та швидкої інформаційної системи, яка буде відповідати потребам та очікуванням користувачів і бізнесу.

Перед початком створення моделей, що будуть детально описувати інформаційну системи збору, обробки та розповсюдження даних необхідно визначити переваги мови моделювання UML.

Мова моделювання UML має великий набір переваг, що робить її популярною серед розробників програмного забезпечення, інженерів, та бізнес-аналітиків:

– стандартизація. Мова моделювання є стандартизованою. Це робить її загальноприйнятою, що дозволяє розробникам різних компаній і команд легко розуміти одне одного;

– універсальність. UML може бути використаний для моделювання різних типів систем, програмного забезпечення та бізнес-процесів [22];

– зручність. Пропонується великий вибір діаграм, які можуть бути використані для різних цілей, таких як моделювання структури, опис поведінки та взаємодії об’єктів;

– спрощення складних процесів. Надається можливість візуалізувати складні концепції і взаємозв’язки у вигляді графічних моделей;

– етапи розробки. UML може використовуватись на різних етапах роботи, для візуалізації будь-яких вимог, що були додані під час розробки чи аналізу процесу.

Підсумовуючи аналіз мови моделювання та її переваг, можна сказати, що вона є потужним інструментом для моделювання системи, процесів та взаємозв’язків. UML дозволяє ефективно розуміти та аналізувати різні частини проектів програмного забезпечення.

2.2 Визначення функціональних вимог до інформаційної системи збору, обробки та розповсюдження даних

Перед визначенням функціональних вимог до інформаційної системи було проведено аналіз предметної області та аналізу аналогічних інформаційних систем, що можуть бути прямими конкурентами для створюваного бізнесу. Отже для отримання лідерських позицій на ринку було розроблено контекстну діаграму IDEF0, що представляє собою головний бізнес-процес.

IDEF0 – це методологія функціонального проектування системи. Використовується для створення функціональної моделі, яка відображає функції та об’єкти системи.

Створюючи нову систему застосування методології IDEF0 є важливим етапом під час визначення вимог і функцій системи, що допоможе в подальшій розробці системи.

Функціональну модель головного бізнес-процесу інформаційної системи збору, обробки та розповсюдження даних наведено на рисунку 2.2.

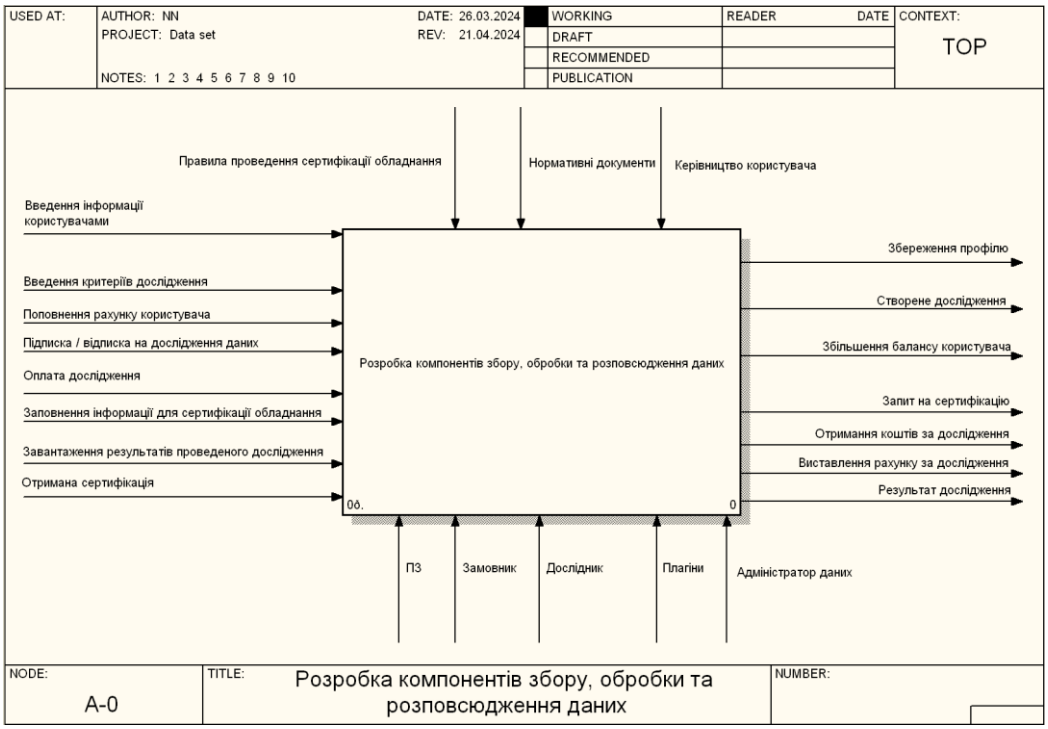

Рисунок 2.2 – Контекстна діаграма головного бізнес-процесу

В ході проектування функціональної моделі було сформовано вхідні та вихідні дані, механізми та управління, які демонструють їх взаємодію з ІС. Стрілки на функціональній моделі представляють дані або матеріальні об'єкти, пов'язані з функціями. Стрілки не являють собою потік або послідовність подій [2]. Вони показують, які дані або матеріальні об'єкти мають надійти на вхід функції для того, щоб ця функція могла виконуватися [16].

Розглянемо всі визначені стрілки на функціональній моделі з визначенням їх видів та описом. Нижче наведено механізми, де механізм – це ресурси необхідні для роботи системи та які не змінюються в процесі роботи:

а) програмне забезпечення (ПЗ);

1) реєстрація та авторизація в інформаційній системі;

2) надсилання запиту на отримання сертифікації свого обладнання;

3) створення запиту на проведення дослідження;

4) поповнення балансу користувача;

5) перегляд створених замовлень;

6) перегляд отриманих результатів створюваного дослідження;

7) надання статусу сертифікації обладнання;

б) плагіни;

1) використання сторонніх API;

2) використання вбудованих API;

в) замовник;

1) реєстрація та авторизація в системі;

2) поповнення балансу користувача для оплати досліджень;

3) створення замовлень з вказанням критеріїв на дослідження даних;

4) перегляд особистих замовлень на дослідження даних;

г) спостерігач;

1) реєстрація та авторизація в системі;

2) введення інформації про своє обладнання;

3) надсилання запиту на отримання сертифікації;

4) можливість підписатися або відписатися від дослідження;

5) перегляд критеріїв дослідження;

6) завантаження та відправка результатів дослідження;

д) адміністратор даних;

1) отримання запитів на видачу сертифікації обладнання спостерігача;

2) перегляд інформації про кожен запит окремо;

3) фільтрація запитів на видачу сертифікації за статусом.

Додатково з механізмами було визначено управління, які зображують правила і обмеження, згідно яких виконується робота:

а) правила проведення сертифікації;

1) забезпечення інформацією спостерігачів системи для отримання сертифікації свого обладнання. Надання актуальної інформації;

2) повідомлення про зміни у проведенні сертифікації обладнання спостерігачів;

б) нормативні документи;

1) надання доступу до нормативних документів користувачам системи;

2) забезпечення актуальної інформації;

3) повідомлення про зміни в нормативних документах та процесі роботи інформаційної системи;

4) надати інформації про обов'язки акторів ІС;

в) керівництво користувача;

1) забезпечити користувачів системи знаннями про використання інформаційної системи відповідно кожної ролі;

2) забезпечити користувачів інформацією про процес роботи зі сторони спостерігача та замовника.

Визначені вхідні та вихідні дані відповідають за інформацію яка надається системі та результати обробки. Для розуміння використання даних було прийняте рішення про проведення декомпозиції функціональної моделі головного бізнес-процесу на бізнес-функції, які забезпечують повноцінне функціонування ІС, що подано на рисунку 2.3. Головний бізнес-процес було поділено на три бізнес-функції:

– реєстрація / авторизація – це бізнес-функція, яка є одною із ключових в інформаційних системах. Функція повинна забезпечувати користувачів зрозумілою формою для проведення авторизації в системі чи реєстрації для отримання доступу до більшої кількості функцій системи. Функціонал неавторизованого користувача досить обмежений, він може переглянути тільки документ «керівництво користувача» в якому подані всі можливі функції та можливості взаємодії з системою та сторінка «Реєстрація» або

«Авторизація». Щодо авторизованого користувача, можна сказати, що його функціонал буде залежати від його ролі в системі;

– взаємодія замовника – мета замовника полягає в створенні замовлення на дослідження або обробку даних, результати якої він зможе отримати після виконання замовлення спостерігачем. Для створення замовлення (дослідження) необхідно перейти до сторінки дослідження та ввести необхідну інформації до відповідних частин форми. Після створення дослідження замовнику необхідно мати на особистому рахунку кошти для оплати замовлення. Для того щоб поповнити рахунок необхідно перейти до певного розділу та провести операцію в певній валюті, після чого, користувач зможе сплачувати за результати свого дослідження;

– взаємодія спостерігача – мета спостерігача полягає в оформленні підписки на виконання дослідження за вказаними критеріями. Перед початком підписки на дослідження спостерігачу необхідно отримати сертифікацію свого обладнання надавши інформацію та фото обладнання, яке буде перевірятися адміністратором даних.

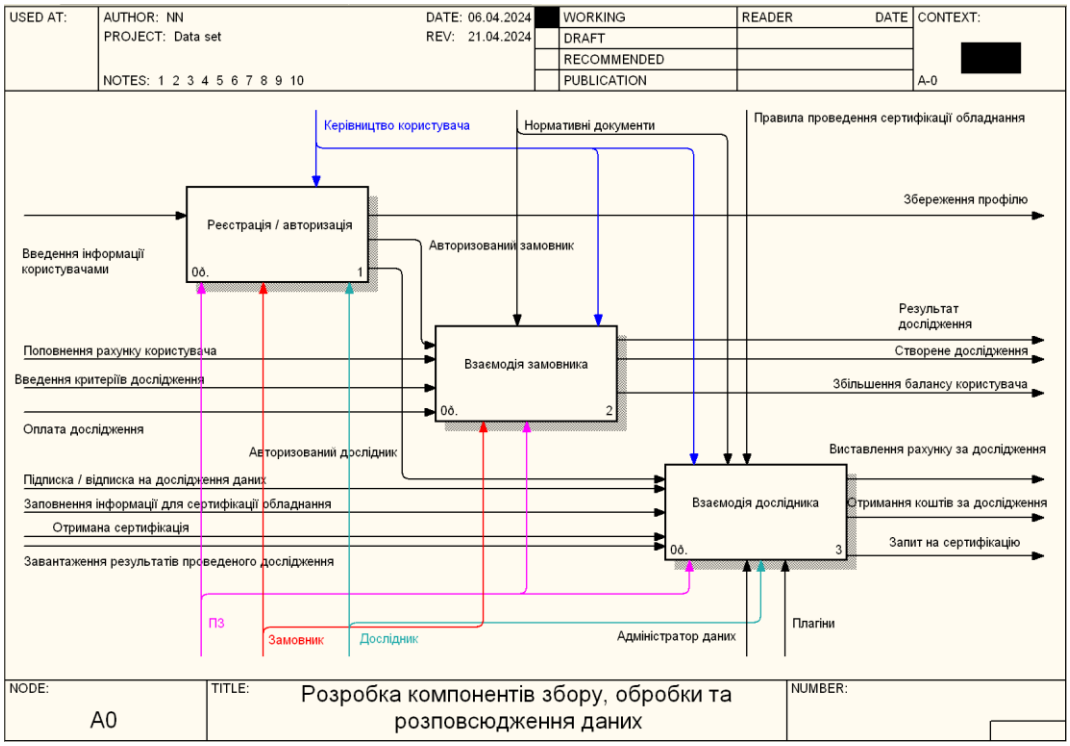

Рисунок 2.3 – Декомпозиція головного бізнес-процесу ІС

Для отримання глибшого розуміння бізнесу та його вимог до інформаційної системи, було вирішено розділити основні бізнес-функції на окремі, менші компоненти [3]. Декомпозиція дозволяє вивчити кожен аспект системи на мікрорівні, що допомагає зрозуміти складні процеси та взаємозв'язки між ними. Завдяки такому розділенню, можна більш ефективно визначити конкретні потреби та вимоги, а також виявити можливості для оптимізації та вдосконалення системи.

Декомпозиція також сприяє кращій організації розробки та тестування, оскільки команди можуть зосередитися на певних компонентах, а не на всій системі одночасно. Цей підхід сприяє більш ефективному плануванню та реалізації проекту, дозволяючи створювати системи, які краще відповідають потребам бізнесу та його користувачів.

Отже, результатом декомпозиції першої бізнес-функції «Реєстрація / авторизація» було отримано наступний результат, що наведено на рисунку 2.4.

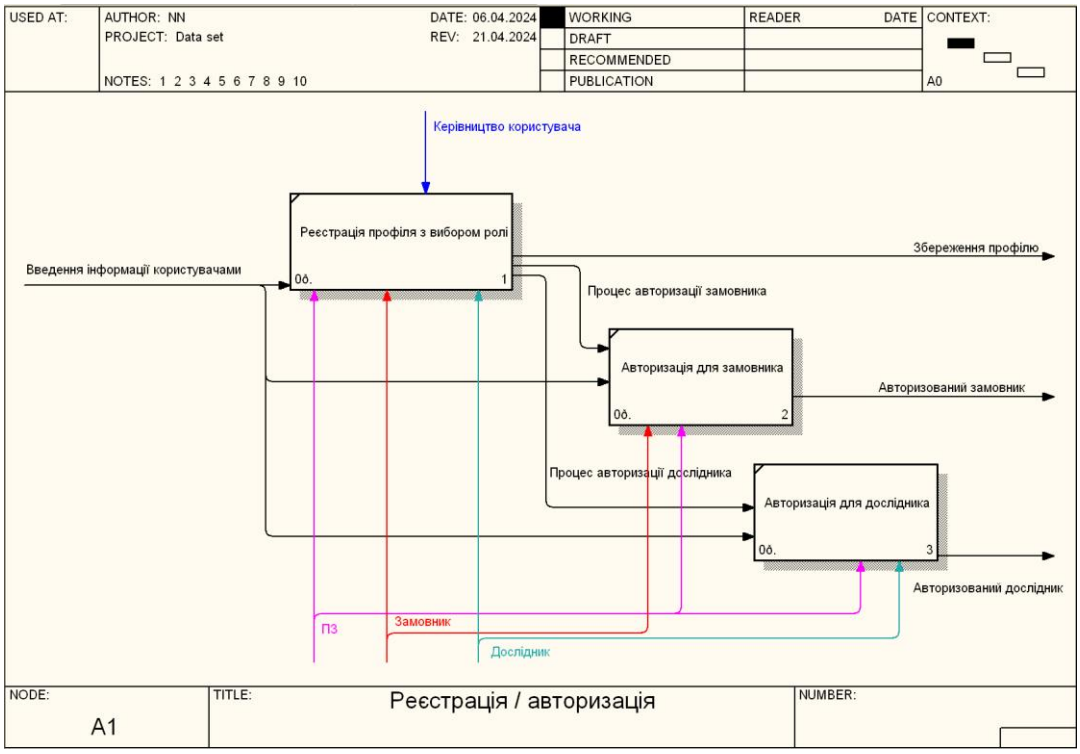

Рисунок 2.4 – Декомпозиція бізнес-функції «Реєстрація / Авторизація»

Розроблена декомпозиція бізнес-функції «Реєстрація та авторизація» містить наступні функціональні блоки:

– реєстрація профіля з вибором ролі. Під час реєстрації профілю важливо забезпечити можливість вибору ролі користувача. Даний функціонал допоможе користувачам визначити свою роль у системі, що, у свою чергу, визначає їхній доступ до певних функцій і можливостей в системі. Обрана роль також впливає на те, які завдання та обов’язки будуть покладені на користувача. Наприклад, адміністратор матиме можливість проводити сертифікацію обладнання дослідників, тоді як спостерігач матиме можливість відправляти запит на отримання сертифікації та можливість виконування дослідження замовників, які в свою чергу створюють замовлення заповнюючи форму та визначаючи критерії для проведення дослідження;

– авторизація для замовника. Авторизація для замовника означає, що після входу в систему користувачеві надається роль "замовник", що відкриває доступ до певного функціоналу або можливостей. Для реалізації цього процесу, потрібно впровадити механізм автентифікації та авторизації, а також визначити, який функціонал буде доступний замовнику;

– авторизація для спостерігача. Авторизація для спостерігача означає, що після входу в систему, користувачу відкривається функціонал відповідно його ролі "спостерігач". Роль спостерігача може включати в себе можливості перегляду критеріїв дослідження на яке була оформлена підписка або за необхідності навіть відмінити підписку, якщо вказана причина відповідає поставленим критеріям.

Наступним кроком було вирішено провести декомпозицію для бізнес-функції «Взаємодія замовника», що передбачає розбивку на дрібніші складові, кожна з яких представляє частину функціоналу замовника. Функція є ключовою для бізнесу, оскільки вона визначає, як замовники будуть взаємодіяти із системою та які можливості будуть використовувати для досягнення своєї мети. Діаграма декомпозиції подана на рисунку 2.5.

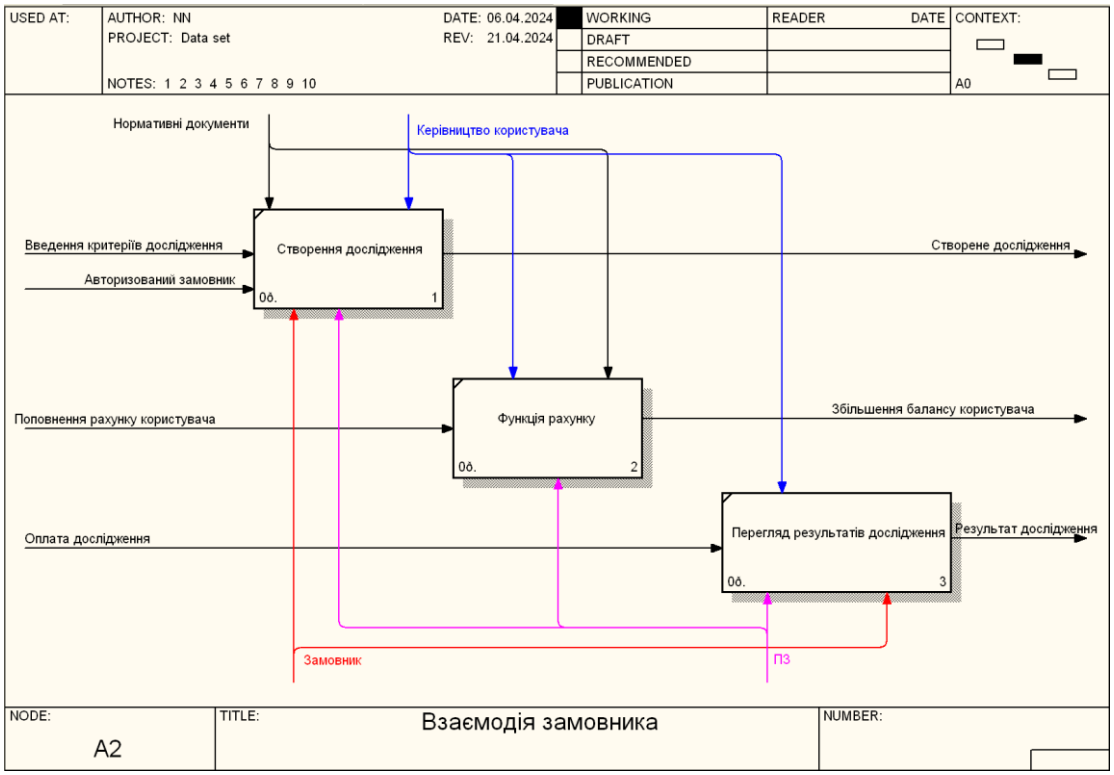

Рисунок 2.5 – Декомпозиція бізнес-функції «Взаємодія замовника»

Розглядаючи функціонал замовника і дослідника необхідно визначити вимоги до ІС та зрозуміти як пов’язані між собою об’єкти та дані системи відповідно кожної ролі. Для цього було проведено декомпозицію функції «Взаємодія замовника», яка містить наступні функціональні блоки:

– створення дослідження. Дана функція передбачає можливість введення інформації та критеріїв створюваного дослідження, є критично важливим компонентом системи, що дозволить взаємодіяти із системою замовникам. Функція реалізована за допомогою веб-технологій на веб-сторінці додатку з використання різноманітних елементів інтерфейсу, що дозволяє користувача легко вводити та змінювати інформацію;

– функція рахунку. Передбачає можливість поповнювати особистий рахунок вводячи інформацію про кількість поповнення з використанням певної валюти, яка є ключовим компонентом для фінансових систем. Ця функція дозволяє користувачам вводити інформацію про суму поповнення та вибрати метод платежу для додавання коштів на особистих рахунок;

– перегляд результатів дослідження. Дана функція представлена у вигляді веб-сторінки, на якій будуть відображатися результати дослідження після його виконання дослідником, що може бути одним із головних елементів для веб-додатків які займаються науковими дослідженням. Вона надає можливості, де замовники, можуть переглядати результати проведеного дослідження, вивчати його деталі та особливості даних.

Остання декомпозиція бізнес-функції «Взаємодія дослідника» представлена на рисунку 2.6, яка передбачає можливості взаємодії з системою для дослідника. Бізнес-функція включає в себе ефективну взаємодію дослідника та системи для виконання завдань на які була оформлена підписка, надсилання запиту на проведення сертифікації свого обладнання за наданням необхідної інформації.

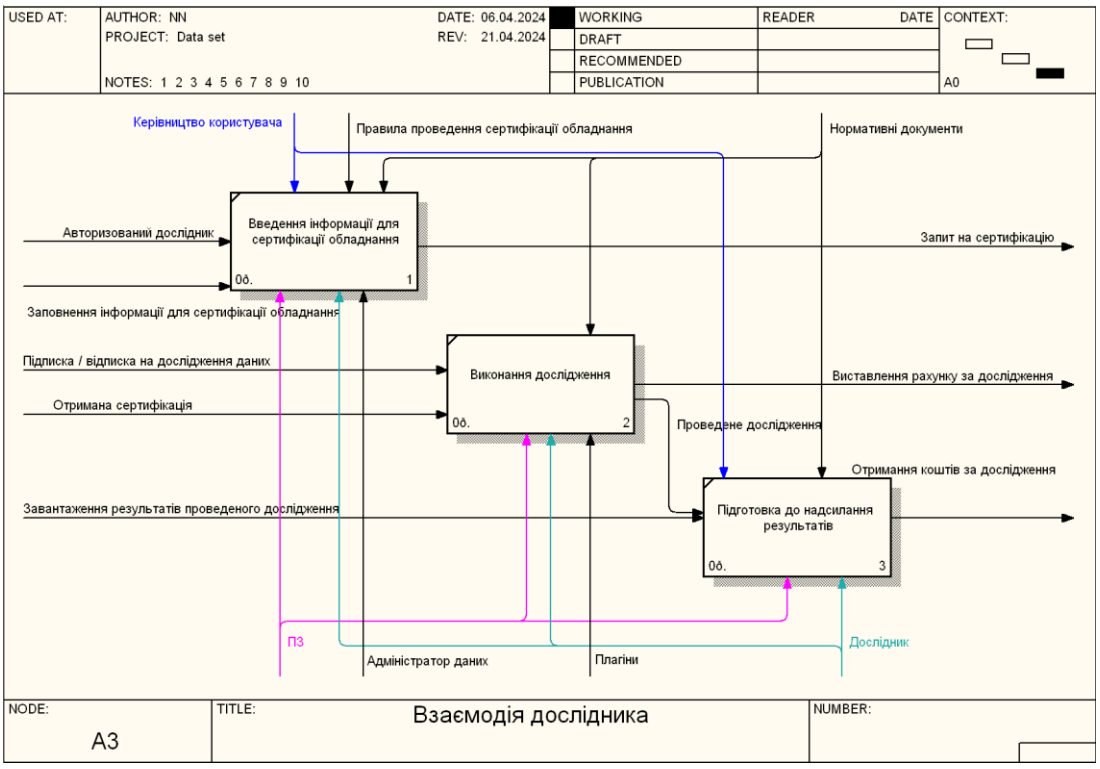

Рисунок 2.6 – Декомпозиція бізнес-функції «Взаємодія дослідника»

Декомпозиція включає в себе наступні функціональні блоки бізнес-функції «Взаємодія дослідника»:

– введення інформації для сертифікації обладнання. Перед початком взаємодії дослідника із системою необхідно отримати сертифікацію свого обладнання, яке відповідає нормам та вимогам бізнесу. Після натискання кнопки «відправити» сформується запит, який буде надісланий адміністратору даних на проведення сертифікації. Цей процес гарантує, що обладнання, яке використовується дослідником, відповідає вимогам безпеки та якості, встановленим організацією або галуззю;

– виконання дослідження. Отримавши позитивний результат запиту сертифікації свого обладнання досліднику відкривається можливість підписки на замовлення. Після успішної сертифікації свого обладнання дослідник отримує право виконувати дослідження, оформлювати підписки на замовлення, встановлювати ціну за свої послуги та, за необхідності, відкликати підписку;

– підготовка до надсилання результатів. Даний процес є завершальним етапом у дослідженні, який передбачає збір, організацію та розповсюдження даних. Оскільки якість та повнота надісланих результатів грає ключову роль у задоволенні замовника, цей етап потребує ретельного підходу.

Підсумуючи результати розробки функціональної моделі та проведенні декомпозиції бізес-функцій можна сказати, що було визначено найголовніші вимоги до інформаційної системи збору, обробки та розповсюдження інформації. Оскільки різні категорії користувачів мають різні вимоги, система має забезпечувати гнучкість та модульність, щоб адаптуватися до потреб кожної групи. Це означає, що кожен актор повинен мати доступ до індивідуального функціоналу в залежності від його ролі в системі. Саме відсутність адаптивного підходу може призвести до зниження ефективності роботи системи та зменшити рівень задоволення користувачів.

Розвиток інформаційної системи має враховувати різні сценарії та передбачати інтеграції та введення нових функцій до системи. Такий підхід доможе стати лідером серед подібних систем та забезпечити стійкість до визначення нових вимог користувачів.

На рисунку 2.7 продемонстровано діаграму дерева вузлів, яка наочно представляє декомпозицію головного процесу та бізнес-функцій системи наукових досліджень. Ієрархічне представлення функціональної моделі допомагає зрозуміти структуру та взаємозв'язки між різними бізнес-функціями, показуючи, як вони підпорядковані головному процесу і як взаємодіють одна з одною.

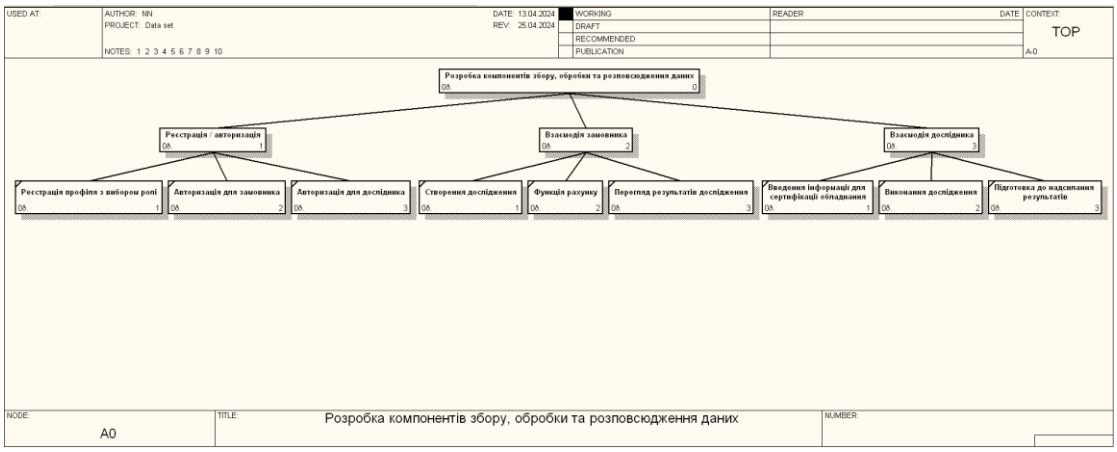

Рисунок 2.7 – Діаграма дерева вузлів

Розробивши та проаналізувавши функціональну модель, було визначено головні функції системи для різних типів акторів, що наведено нижче:

а) незареєстрований користувач;

1) реєстрація в системі;

б) замовник;

1) авторизація в системі;

2) створення замовлення на дослідження даних;

3) поповнення рахунку користувача;

4) оплата дослідження;

5) перегляд результатів дослідження;

в) дослідник;

1) авторизація в системі;

2) створення запиту на сертифікацію обладнання;

3) підписка або відписка від проведення дослідження;

4) виставлення рахунку на виконання дослідження;

5) завантаження звітів дослідження;

г) адміністратор даних;

1) авторизація в системі;

2) перегляд всіх запитів сертифікації обладнання;

3) сортування запитів;

4) видача позитивного або негативного результату, щодо сертифікації обладнання дослідника;

У накових дослідженнях та розробці інформаційних систем, наявність чітко визначених функцій та вимог є критично важливим етапом. Використовуючи визначені функції в системі наукових досліджень, ми можемо охопити всі ключові аспекти нашої діяльності, які необхідні для успішної реалізації ІС. Кожна функція має свою унікальну роль та залежить від інших, створюючи цілісну інформаційну систему для збору, обробки та розповсюдження даних.

2.3 Розробка моделі потоків даних (Data Flow Diagram)

На основі функціональної моделі було прийняте рішення побудувати діаграму потоків даних відповідно до нотації DFD. DFD – це нотація представлення структури процесів, яка не містить логічних операторів [20]. Розроблена діаграма представляє собою графічне представлення процесу, який включає в себе перехід даних в інформаційній системі включаючи як вхідні потоки так і вихідні. Розроблену діаграму потоків даних представлено на рисунку 2.8, на якій визначено чотири зовнішні сутності, які обмінюються потоками даних з інформаційною системою. Отже DFD є ефективним інструментом, оскільки допомогає забезпечити зрозуміле графічне зображення системи та її сутностей.

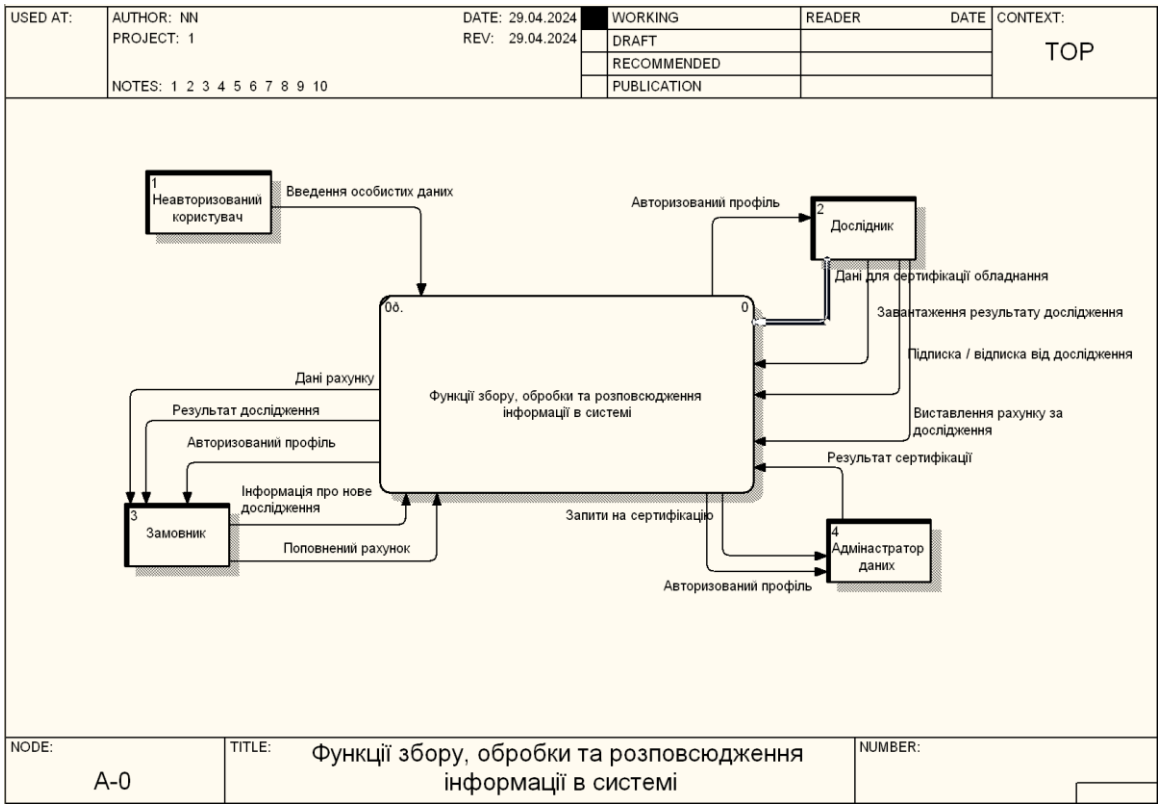

Рисунок 2.8 – Модель потоків даних DFD «Функції збору, обробки та розповсюдження інформації в системі»

Розглянемо детально всі потоки даних, що були визначені в ході проектування моделі потоків даних DFD «Функції збору, обробки та розповсюдження інформації в системі» для взаємодії з чотирма сутностями:

– введення особистих даних – дані користувачів для створення особистого профілю чи авторизацію в системі;

– авторизований профіль – результат авторизації відповідно ролі профілю користувача;

– дані рахунку – інформація для поповнення рахунку, що вводиться через додаток банку;

– результат дослідження – перегляд результату виконаного дослідження дослідником;

– інформація про нове дослідження – обрані критерії для створення нового дослідження;

– поповнений рахунок – отримані кошти, що були переведені через додаток банку;

– дані для сертифікації обладнання – дослідник надає інформацію про своє обладнання яке буде використовувати для проведення досліджень;

– завантаження результату дослідження – провівши дослідження замовника необхідно завантажити результати та надіслати його;

– підписка / відписка від дослідження – дослідник може оформити підписку або підписку від замовлення, що буде свідчить про те, що дослідження взято на виконання;

– виставлення рахунку за дослідження – дослідник визначає ціну за проведення дослідження;

– результат сертифікації – адміністратор даних проводить сертифікацію обладнання дослідника, після чого дослідник отримує статус сертифікації свого обладнання;

– запити на сертифікацію – адміністратор даних може переглядати всі запити незалежно від статусу.

2.4 Визначений інтерфейс користувача до інформаційної системи взаємодії з даними.

Інтерфейс користувача використовується у вигляді веб-сторінок, що розроблені за допомогою HTML, CSS та JavaScript коду. Доступ до сторінок поділяється відповідно до акторів в системі, а саме дослідник, замовник або адміністратор даних.

Сторінки, що доступні адміністраторам системи:

– сторінка авторизації – дозволяє ввести логін та пароль для входу до системи;

– сторінка реєстрації – дозволяє ввести інформацію до форми, що зареєструє в системі після натискання на кнопку «Зареєструватися»;

– сторінка проведення сертифікації – дозволяє переглянути інформацію надіслану на перевірку та видати підтвердження про те, що обладнання відповідає стандартам та вимогам поставленими бізнесом.

Сторінки, що доступні видавцям (дослідникам) системи:

– сторінка авторизації – дозволяє ввести логін та пароль для входу до системи;

– сторінка реєстрації – дозволяє ввести інформації до форми, що зареєструє в системі після натискання на кнопку «Зареєструватися»;

– сторінка перегляду досліджень – надає можливість переглядати дослідження за категоріями, що присутні у системі;

– сторінка оформлення підписки або відписки на спостереження – дозволяє видавцю оформити підписку на виконання завдання, та при непередбачуваній ситуації її відмінити, якщо проблема присутня у списку;

– сторінка відправка даних на сертифікації обладнання – сторінка містить форму, яку необхідно заповнити та отримати підтвердження від адміністратора даних, про те, що обладнання відповідає вимогам і стандартам;

– сторінка відправлення результату спостереження – сторінка містить форму, яку необхідно заповнити видавцю для успішної відправки оброблених даних.

Сторінки, що доступні підписникам (замовникам) системи:

– сторінка авторизації – дозволяє ввести логін та пароль для входу до системи;

– сторінка реєстрації – дозволяє ввести інформації до форми, що зареєструє в системі після натискання на кнопку «Зареєструватися»;

– сторінка створення дослідження – на цій сторінці необхідно вказати інформації та критерії, щодо спостереження та надіслати його;

– сторінка отримання результатів дослідження – дозволяє отримати результати спостереження на яке було створено замовлення та переглянути його;

– сторінка поповнення балансу користувача – у кожного користувача є його особистий баланс, що відображає кількість коштів для оплати досліджень.

На рисунку 2.9 подано візуальне представлення розроблених сторінок інформаційної системи взаємодії з даними з розподілом доступу сторінок відповідно кожного визначеного актора.

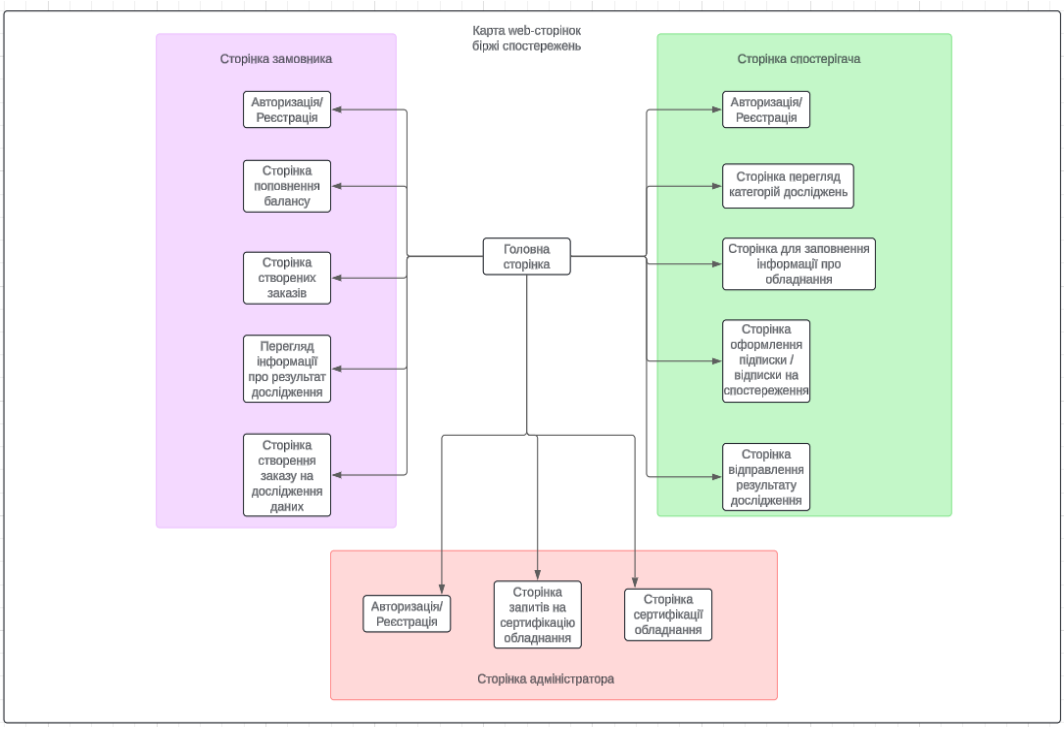

Рисунок 2.9 – Карта web-сторінок

2.5 Діаграма варіантів використання інформаційної системи збору, обробки та розповсюдження даних

Для графічного зображення прецедентів, що доступні акторам системи було використано діаграму варіантів використання. Діаграма варіантів використання – дозволяє уявити типи ролей та їх взаємодії із системою, проте вона не показує послідовність виконання. Головна мета діаграми показати те, що системо може робити з точки зору користувача [24]. Розроблену діаграму варіантів використання продемонстровано на рисунку 2.10.

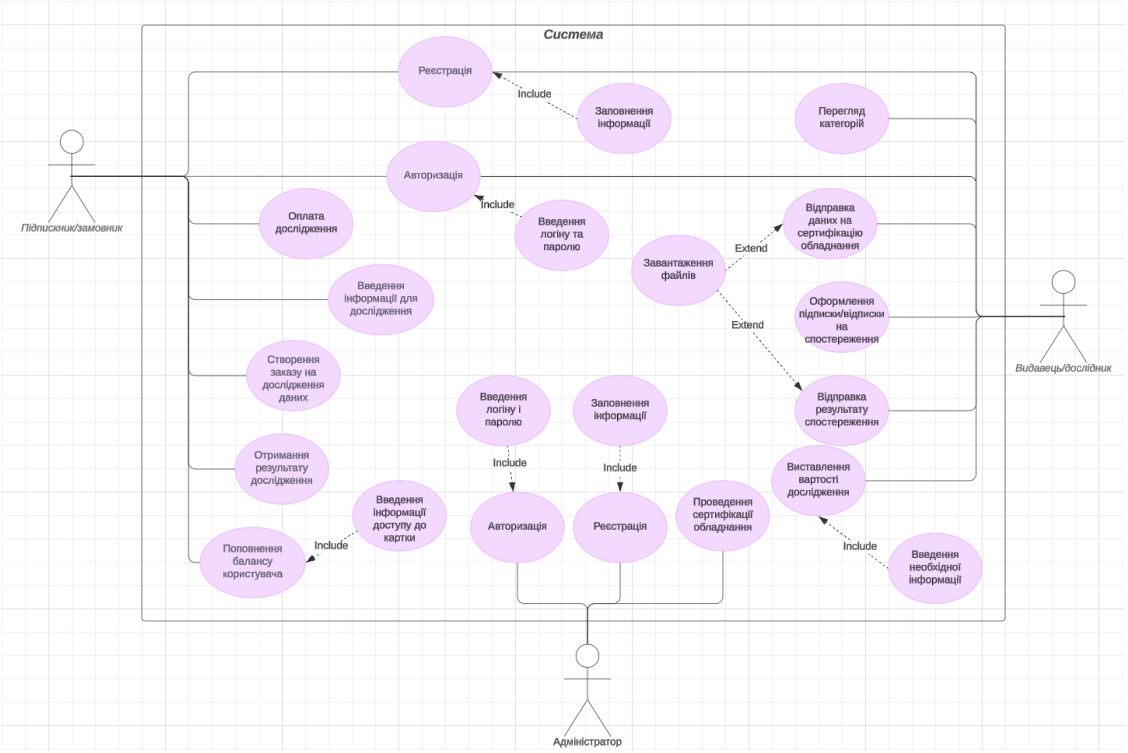

Рисунок 2.10 – Діаграма варіантів використання системи збору, обробки та розповсюдження інформації

Бізнес-функції, що представлені для кожного із акторів на діаграмі та підлягають реалізації наведені далі.

Бізнес-функції підписника/замовника:

– реєстрація;

– авторизація;

– введення інформації для створення дослідження;

– створення заказу на дослідження даних;

– отримання даних дослідження;

– поповнення балансу користувача.

Бізнес-функції видавця/спостерігача:

– реєстрація;

– авторизація;

– оформлення підписки/відписки на спостереження;

– перегляд категорій;

– відправка даних на сертифікацію обладнання;

– відправка результату спостереження.

Бізнес-функції адміністратора:

– реєстрація;

– авторизація;

– проведення сертифікації обладнання.

Визначений функціонал необхідно розробити в повному обсязі, щоб отримати бажаний результат, який можливо покращувати створюючи вбудовані плагіни для обробки даних.

Аналіз діаграми варіантів використання (use case diagram) може стати важливим етапом у проектуванні інформаційних систем. Діаграма допомагає зрозуміти, який функціонал необхідно реалізувати в системи, щоб задовольнити потреби користувачів та досягти поставлених цілей для бізнесу.

Проведемо аналіз компонентів діаграми варіантів використання та на які аспекти вони впливають під час проектування:

– ідентифікація акторів – актори на діаграмі варіантів використання представляють різні категорії користувачів або зовнішні системи, які взаємодіють з певною системою. Визначення акторів допомагає зрозуміти хто буде використовувати систему та які їх вимоги до неї;

– визначення варіантів використання – варіанти використання описують як зовнішні актори взаємодіють із системою для досягнення своїх цілей. Кожен варіант використання має свій опис або послідовність дій та результати виконання;

– взаємодія між варіантами використання – діаграма варіантів використання може показувати взаємозв’язок між різними варіантами використання в системі. Наприклад, варіанти використання можуть доповнювати одне одного;

– бізнес-функції – визначивши варіанти використання, можна зрозуміти, які бізнес-функції необхідно реалізувати. Розуміння визначених бізнес-функцій дозволяє побудувати інформаційну систему, яка відповідає потребам користувачів та вимогам;

– розробка системних вимог – на основі спроектованої діаграми можна розробити системні вимоги до ІС, які продемонструють, те як система повинна функціонувати. Цей етап включає визначення функціональних і нефункціональних вимог;

– комунікація із зацікавленими сторонами – діаграма варіантів використання є ефективним інструментом для комунікації із зацікавленими сторонами. Діаграма надає зрозуміле графічне представлення про те, як система буде функціонувати зі сторони користувача.

Підсумовуючи, аналіз діаграми варіантів використання є важливим кроком під час розробки інформаційних систем, оскільки ми можемо визначити необхідний функціонал та забезпечити комунікацію між учасниками проекту.

2.6 Діаграма послідовності дій (Sequence diagram)

Мета діаграми послідовності полягає в візуальному представленні послідовності дій та взаємодії компонентів або об’єктів в системі за допомогою обміну повідомлень або викликів методів. Діаграма послідовності необхідна для того, щоб показати, як різні елементи системи взаємодіють для виконання певної задачі або бізнес-процесу.

Будь-яка діаграма послідовності має конкретні цілі під час її проектування:

– відображення взаємодії в часі. Діаграма послідовності демонструє порядок взаємодії між об’єктами системи, що допомагає зрозуміти, як відбуваються події;

– візуалізація логіки системи. Дозволяє проілюструвати послідовність кроків в системі для виконання певного функціоналу чи процесу;

– аналіз варіантів використання. Допомагає зрозуміти, як реалізуються варіанти використання системи, забезпечуючи чітке розуміння процесу;

– покращення комунікації в команді. Діаграма є ефективним засобом для комунікації команди та зацікавленими сторонами;

– документація системи. Діаграма послідовності дій є частиною технічної документації, що полегшує підтримку та оновлення системи.

Результатом проведеного моделювання діаграми послідовності дій для прецеденту «створення замовлення на дослідження даних» подано на рисунку 2.11.

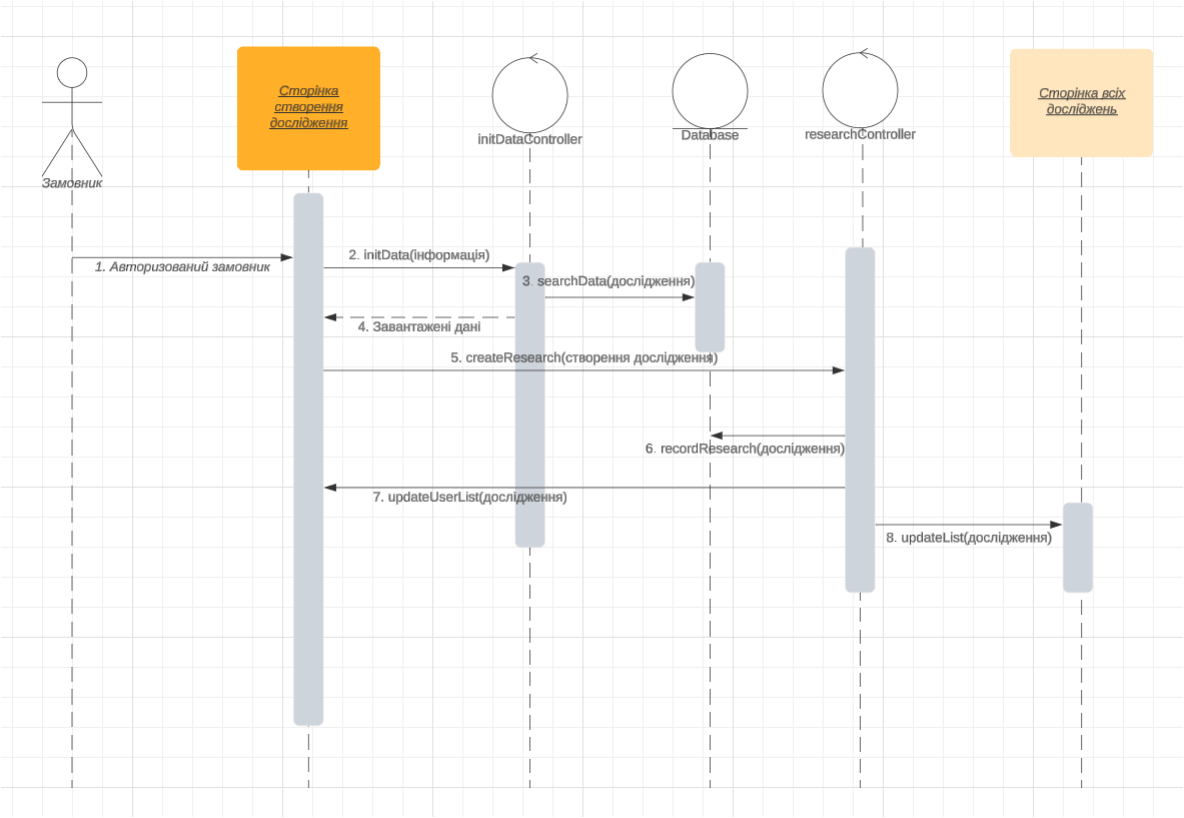

Рисунок 2.11 – Діаграма послідовності дій для прецеденту «створення замовлення на дослідження даних»

На діаграмі для прецеденту «створення замовлення на дослідження даних» було візуально спроектована послідовність дій. Початком створення завжди є авторизований користувач, який відвідав сторінку «створення дослідження». На сторінці «створення дослідження» замовнику необхідно натиснути кнопку «створити» після чого користувачу відкриється модальне вікно з формою для введення критеріїв створюваного дослідження.

Заповнивши форму створення дослідження та натиснувши кнопку починають виконуватися процеси на стороні сервера. Після успішного виконання процесів створене замовлення потрапляє до сторінки всіх досліджень, на які дослідники можуть оформити підписку та почати його виконувати.

2.7 Проектування діаграми класів системи збору, обробки та розповсюдження даних

Проектування та визначення елементів для діаграми класів, її методів, атрибутів та взаємозв’язків між ними може стати важливим етапом для документування та розробки інформаційної системи. Діаграма класів є потужним інструментом для візуалізації принципів об’єктно-орієнтованого програмування (ООП) та забезпечити чітке розуміння структури ІС. Вона дозволяє розробникам та архітекторам системи чітко уявити, як класи взаємодіють між собою, що допомагає покращити модульність системи [13].

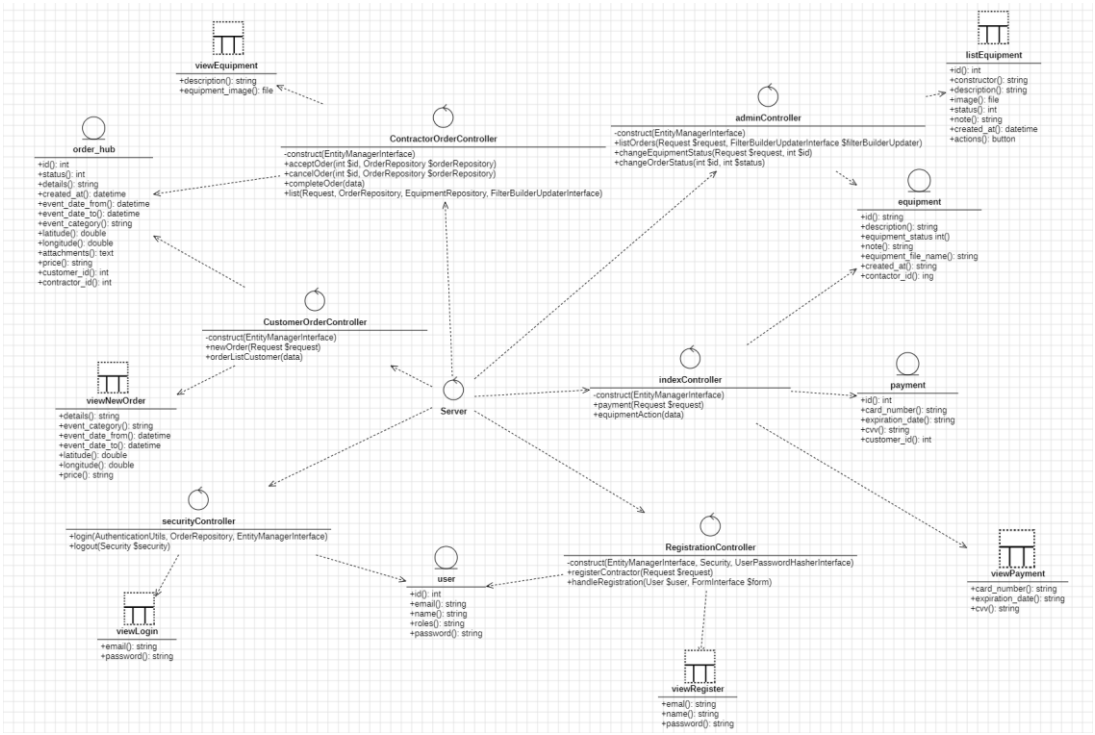

Рисунок 2.12 Розроблена діаграма класів для інформаційної системи взаємодії з даними.

Зробимо короткий опис діаграми класів, який буде використано як частину документації інформаційної системи збору, обробки та розповсюдження інформації та представлена у вигляді таблиці. Короткий опис діаграми класів наведено в таблиці 2.1.

Таблиця 2.1 – Опис контролерів діаграми класів

Контролер	Методи / атрибути	Опис методів	Модель взаємодії
contactorOrderController	construct	Ініціалізація об’єктів для взаємодії системи	Order_hub
	acceptOrder	Підписка на дослідження	
	cancelOrder	Відписка від дослідження	
	completeOrder	Завершення дослідження	
	list	Перегляд всіх доступних досліджень	
customerOrderController	construct		Order_hub
	newOrder	Створення нового замовлення на дослідження даних	
	orderList	Перегляд всіх досліджень	
adminController	construct		Equipment
	listOrders	Перегляд всіх замовлень на дослідження даних	
	changeEquipmentStatus	Проведення сертифікації	
	changeOrderStatus	Зміна статусу замовлення	
securityController	login	Метод авторизації в системі	User
	logout	Вихід із системи	
registrationController	construct		User
	registerCustomer	Реєстрація замовника	
	registerContractor	Реєстрація дослідника	
	handleRegistration	Обробка запиту реєстрації	
indexController	construct		Payment
	payment	Збереження даних картки	
	equipmentAction	Перегляд запитів на проведення сертифікації обладнання	
	indexCustomer	Отримання інформації про авторизованого користувача	

Проектування діаграми може бути дуже важливим етапом для визначення системи через що вона має чисельні переваги, які сприяють розробці ПЗ, так і єдиному розумінню системи учасниками проекту.

– візуалізація структури ПЗ. Діаграма класів дозволяє створити графічне представлення системи, відображаючи класи та зв’язки між об’єктами [23]. Це дозволить розробникам та архітекторам краще розуміти, як відбувається взаємодія в системі;

– модульність і розширюваність. Діаграма дозволяє визначити модулі та їх функціональність в системі, що полегшить розробку та розширення системи;

– зменшення складності. Групування класів, моделей, атрибутів та інших елементів системи допомагає зменшити складність системи, що полегшить розуміння та підтримку ІС;

– взаємодії об’єктів. Визначення взаємозв’язків на діаграмі класів дозволить правильно зрозуміти, як елементи системи використовують одне одного;

– класифікація. Діаграма класів може розподілити елементи за класифікацією та призначенням, що полегшить розуміння системи.

Підсумовуючи, проектування діаграми класів з визначенням необхідних елементів системи є потужним інструментом, який допоможе покращити якість розробки і розуміння інформаційної системи.

3 ОПИС ПРИЙНЯТИХ РІШЕНЬ, ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Вибір технологій проектування системи збору, обробки та розповсюдження інформації

Для проектування інформаційної системи збору, обробки та розповсюдження інформації було обрано мову програмування PHP з використанням фреймворку Symfony та шаблонізатору twig.

PHP – це популярна мова програмування веб-додатків, ідея якої полягає в спрощенні процесу створенні динамічних веб-сторінок. Незважаючи на те, що сучасний PHP представлений як мова загального призначення, найчастіше використовують його для генерації HTML-коду, який потім інтерпретується веб-браузером.

У створюваному проекту присутній фреймворк Symfony, що написаний на PHP. Його мета полягає у прискоренні розробки та підтримки веб-застосунків, а також зменшення часу для розв’язування важких задач [7]. Обираючи технологію для проектування було проаналізовано переваги та недоліки інших PHP-фреймворків, саме: Zend, Yii, CodeIgniter та Laravel. Звертаючи на переваги, було визначено головні причини, через які було обрано Symfony-фреймворк для розробки програмного забезпечення:

– гнучка архітектура. Symfony – представляє добре документований фреймворк, який простий у освоєнні та використанні [18]. Архітектура дозволяє легко розроблювати інформаційні системи найпростішим способом. Symfony включає всі необхідні аспекти, що очікуються від сучасного фреймворку та має чітку структуру, яка полегшить навігацію у своєму коді;

– сумісність. Symfony – дозволяє створити програми, що відповідають потребам будь-якого бізнесу на веб-платформі. Фреймворк надає доступ до

використання окремих компонентів не використовуючи при цьому весь фреймворк;

– екосистема. Фреймворк може надати розробнику доступ до сторонніх плагінів, що називаються Symfony bundles. Для вирішення великої кількості задач, можна знайти необхідний пакет, який допоможе вирішити задачу [19];

– репутація. Запуск проекту відбувся в 2005 році, та набув популярності серед PHP-розробників в усьому світі. На сьогоднішній день фреймворк надає стабільне середовище, яке є визначним на міжнародному рівні. Не менш важливим є активна спільнота Symfony-фреймворку, які беруть постійну участь у вдосконаленні фреймворку, його пакетів та інструментів розробки.

Обраний стек програмування було обґрунтовано на основі аналізу переваг та недоліків інших фреймворків. Symfony визначено як оптимальний вибір для проектування завдяки його гнучкій архітектурі, екосистемі і швидкості розробки ПЗ відповідно потреб бізнесу.

3.2 Вибір архітектури проектування інформаційної системи

Вибір архітектури проектування інформаційної системи є важливим етапом у процесі розробки програмного забезпечення. Цей етап потребує детального аналізу та врахування визначених вимог до системи і потреб користувачів до бізнесу. Обрана архітектура може вплинути на аспекти розробки, гнучкості та масштабованості системи [4].

Архітектура інформаційної системи визначає її організацію та структуру програмного забезпечення, що включає розподіл функцій між елементами системи та способи взаємодії між ними.

Для інформаційної системи збору, обробки та розповсюдження даних було обрано архітектуру проектування MVC (Model-View-Controller). MVC – це архітектурний шаблон, який використовується у веб-розробці для поділу системи на три складові: модель, представлення, контролер. На рисунку 3.1 подано графічне представлення архітектурного шаблону MVC.

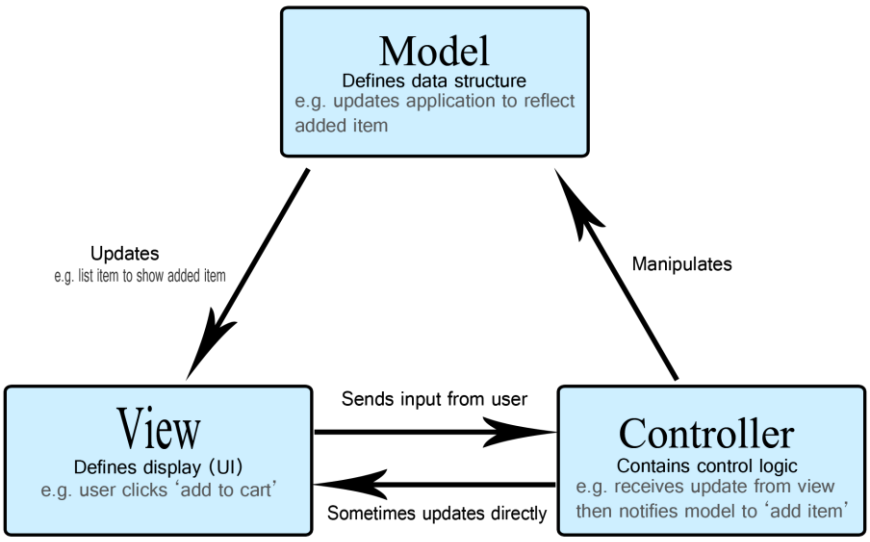

Рисунок 3.1 – Графічне представлення MVC архітектури

Архітектура Model-View-Controller є популярним шаблоном проектування програмного забезпечення. Порівнюючи переваги цього шаблону з іншими, було отримано значно більше переваг, що наведені далі:

– розподіл логіки системи. MVC дозволяє чітко розподілити бізнес-логіку, представлення даних користувачеві та управління взаємодією з користувачем;

– масштабованість. Дозволяє розробити складні інформаційні системи, які легко масштабувати;

– повторне використання коду. Дозволяє використовувати логіку компонента в будь-якій іншій частині системи;

– легке тестування. Розподіл системи на компоненти дозволяє проводити тестування тільки для однієї частини системи;

– підтримка інтерфейсів. Легке створення інтерфейсів системи, наприклад, веб-сторінка або API;

– структура системи. Розподіл компонентів дозволяє надати чітку структуру проекту, що полегшить зберігання та взаємодію з компонентами.

Таким чином, обрана архітектура буде найкращим варіантом для нашої системи, оскільки ми плануємо повторно використовувати логіку компонентів, що полегшить роботу з складними графічними інтерфейсами та бізнес-логікою інформаційної системи.

3.3 Обґрунтування вибору СУБД

Вибір СУБД може стати ключовим фактом в проектуванні бізнес-логіки системи та швидкісного і ефективного отримання даних. Обравши мову програмування необхідно провести аналіз СУБД, які можуть бути обрані для інформаційної системи збору, обробки та розповсюдження інформації. Необхідно звернути увагу на такі фактори, як тип даних, обсяг інформації, потреби в швидкодії операцій та рівень безпеки даних.

Для забезпечення ефективної роботи, було обрано реляційну базу даних MariaDB, що забезпечить структурований підхід до зберігання даних з підтримкою SQL для операцій з даними.

MariaDB – представляє відкриту реляційну СУБД, яка використовується на основі MySQL. Обрана база даних має ряд переваг чому її було обрано для розробки ІС:

– відкритий код та спільнота. Відкритий код дозволяє використовувати його, змінювати та поширювати на активну спільноту розробників, що постійно вдосконалюють СУБД;

– сумісність з MySQL. Розроблені фрагменти коду, що були розроблені для MySQL можна перенести на MariaDB без значних змін;

– висока продуктивність. MariaDB завдяки своїй оптимізації має високий рівень швидкодії та ефективності для виконання запитів БД;

– розширений функціонал. Має більше можливостей використання СУБД аніж MySQL.

На основі проведеного аналізу MariaDB стає кращим варіантом для розробки бази даних інформаційної системи збору, обробки та

розповсюдження інформації, що дасть можливість швидко оброблювати та оновлювати запити на створення досліджень та отримання сертифікації обладнання дослідника.

3.4 Вибір структури проектування інформаційної системи

Інформаційну систему взаємодії з даними було розроблено з використанням веб-технологій та MVC архітектурою, що разом дадуть можливість швидкого та ефективного проектування системи. На рисунку 3.2 подано обрані веб-технології проектування системи.

Рисунок 3.2 – Обрані технології проектування інформаційної системи

Стек проектування LEMP – це комбінація чотирьох технологій з відкритим кодом, які використовуються у веб-розробці [17]. Ці технології включають:

– Linux. Операційна система, що використовується для запуску веб-серверу;

– Nginx. Програмне забезпечення веб-серверу, що оброблює запити HTTP, які надходять від інформаційної системи;

– MySQL. РСУБД – реляційна система управління базами даних, яка зберігає дані системи [21];

– PHP. Мова програмування загального призначення, що використовується для розробки динамічних веб-застосунків.

Даний стек проектування веб-застосунку буде оптимальним для бізнесу по збору, обробки та розповсюдження даних. Оскільки головною бізнес-логікою системи є створення веб-серверу та створення взаємодії даних з користувачами системи, що буде виконано за допомогою швидкого і ефективного отримання інформації з бази даних.

3.5 Моделювання ER-діаграми та створення бази даних інформаційної системи взаємодії з даними

У процесі розробки бази даних було приділено значну кількість часу на визначення типів даних полів та визначення полів з взаємозв’язком таблиць. Перед проектування було проведено аналіз вимог та потреб до інформаційної системи, щоб визначити необхідні сутності для створення таблиць бази даних, для яких визначаємо необхідні поля та їх тип даних.

Окрему увагу надано для визначення взаємозв’язків між створеними таблицями бази даних, що допоможе створити цілісність даних та отримання даних одразу з декількох таблиць. Визначення взаємозв’язків залежить від потреб та бізнес-логіки процесу, наприклад, один до одного, один до багатьох та багато до багатьох.

Для визначення типів взаємозв’язків використовувались методи нормалізації бази даних зі створенням зовнішніх ключів, що забезпечують доступ до зв’язних таблиць.

На рисунку 3.3 подано логічну модель бази даних «order_hub» для інформаційної системи збору, обробки та розповсюдження інформації, що побудована з використанням РСУБД MariaDB.

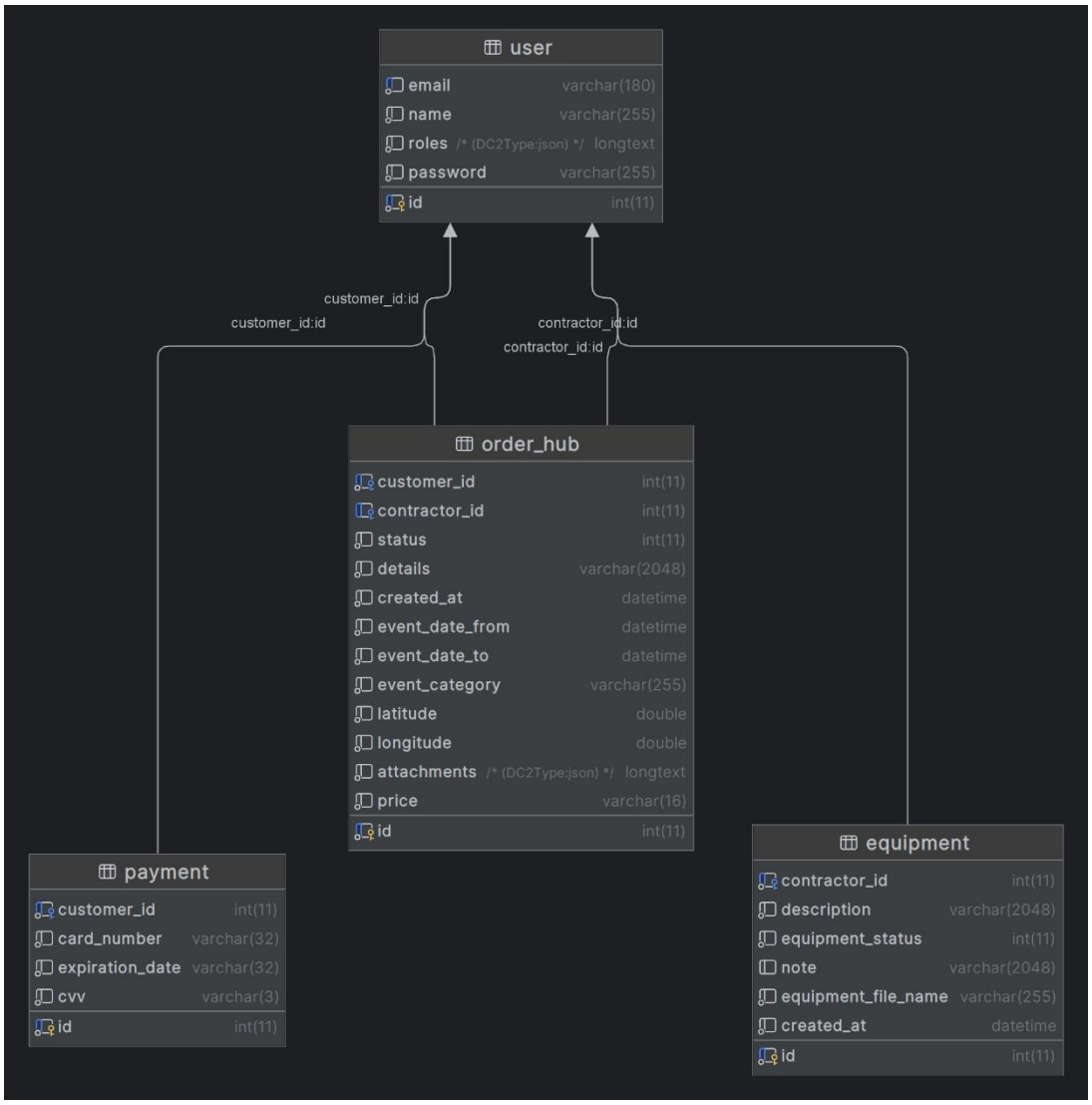

Рисунок 3.3 – Структура розробленої бази даних інформаційної системи збору, обробки та розповсюдження інформації

На логічній моделі, що була розглянута на рисунку 3.3 представлено чотири сутності за допомогою яких відбувається взаємодія та використання даних в програмному забезпеченні. Детальний опис таблиць бази даних наведено в таблиці 3.1.

Таблиця 3.1 – опис таблиць бази даних

№	Назва таблиці	Назва поля	Тип даних	Примітка
1	payment	id	Int(11)	Унікальний ідентифікатор
		cvv	Varchar(3)	Захисний код картки
		expiration_date	Varchar(32)	Термін придатності
		card_number	Varchar(32)	Номер
		customer_id	Int(11)	Ідентифікатор користувача
2	equipment	id	Int(11)	Унікальний ідентифікатор
		created_at	Datetime	Дата створення
		equipment_file_name	Varchar(255)	Завантажені файли обладнання
		note	Varchar(2048)	Помітка
		equipment_status	Int(11)	Статус обладанання
		description	Varchar(2048)	Опис обладнання
		contractor_id	Int(11)	Ідентифікатор дослідника
3	user	id	Int(11)	Унікальний ідентифікатор
		roles	Longtext	Роль користувача
		email	Varchar(180)	Електронна адреса
		password	Varchar(255)	Пароль
		name	Varchar(255)	Ім’я

Кінець таблиці 3.1

№	Назва таблиці	Назва поля	Тип даних	Примітка
4	order_hub	id	Int(11)	Унікальний ідентифікатор
		price	Varchar(16)	Ціна за дослідження
		attachments	Longtext	Додаткові вкладення
		longitude	Double	Довгота
		latitude	Double	Широта
		event_category	Varchar(255)	Категорія дослідження
		event_date_to	Datetime	Кінець дати проведення
		event_date_from	Datetime	Початок дати проведення
		created_at	Datetime	Дата створення
		details	Varchar(2048)	Додаткові деталі
		status	Int(11)	Статус
		contractor_id	Int(11)	Ідентифікатор дослідника, що виконує завдання
		customer_id	Int(11)	Ідентифікатор замовника

Створена база даних, забезпечує ефективну та гнучку структуру, яку можна легко адаптувати під нові зміни та розширення системи в майбутніх оновленнях.

3.6 Розробка моделей та контролерів для інформаційної системи взаємодії з даними

Метою проектування контролерів і моделей є забезпечення програмного забезпечення бізнес-логікою та взаємодією з базою даних процеси, що відбуваються на стороні користувача. Контролери були розроблені з використанням мови програмування PHP та фреймворку Symfony, дозволило значно прискорити процес проектування завдяки бібліотекам, що спрощують роботу [10].

Створення контролера «CustomerOrderController», який допомагає керувати замовленнями користувача має два головні методи newOrder і orderListCustomer. Метод newOrder оброблює запити на створення нових замовлень в ІС. Він включає в себе форму замовлення, яка буде заповнюватись користувачем з використанням веб-інтерфейсу. Відправлені дані замовлення будуть автоматично встановлювати поточного користувача, як замовника цього дослідження після чого буде проводитись перенаправлення користувача на сторінку всіх замовлень даного користувача. Наступний метод orderListCustomer відповідає за відображення всіх створених замовлень поточного користувача на сторінці «Створені замовлення». Дана сторінка реалізовує функції фільтрації особистих замовлень, проведення пагінації та відображення списку всіх замовлень користувача. Отже «CustomerOrderController» забезпечує замовників системи необхідним функціоналом користувачів для взаємодії з системою. Реалізація контролера подана на рисунку 3.4.

```
class CustomerOrderController extends AbstractController
{
    #[Route('/newOrder', name: 'app_customer_new_order')]
    public function newOrder(Request $request): Response
    {
        $form = $this->createForm(OrderFormType::class);
        $form->handleRequest($request);
        if ($form->isSubmitted() && $form->isValid()) {
```

```
            $order = $form->getData();
            $user = $this->getUser();
            $order->setCustomer($user);
            $order->setStatus(OrderStatus::NEW);
            $this->em->persist($order);
            $this->em->flush();
            return $this->redirectToRoute('app_customer_order_list');
        }
        return $this->render('customer/new_order.html.twig', [
            'newOrderForm' => $form,
        ]);
    }

    #[Route('/orderCustomer', name: 'app_customer_order_list',
methods: ['GET'])]
    public function orderListCustomer(
        Request $request,
        OrderRepository $orderRepository,
        FilterBuilderUpdaterInterface $filterBuilderUpdater
    ): Response {
        $form = $this->createForm(OrderFormFilter::class);
        $offset = max(0, $request->query->getInt('offset'));
        $qb = $orderRepository->getOrderQbByCustomer($this->getUser(),
$offset);
        if ($request->query->has($form->getName())) {
            $form->submit($request->query->all($form->getName()));
            $filterBuilderUpdater->addFilterConditions($form, $qb);
        }
        $paginator = new Paginator($qb);
        return $this->render('customer/order_list.html.twig', [
            'orders' => $paginator,
            'previous' => $offset -
OrderRepository::PAGINATOR_PER_PAGE,
            'next' => min(count($paginator), $offset +
OrderRepository::PAGINATOR_PER_PAGE),
            'form' => $form->createView(),
        ]);
    }
}
```

Рисунок 3.4 – Програмний код контролеру «CustomerOrderController», що реалізує головний функціонал для замовника

Функціональність для адміністратора на стороні сервера реалізована за допомогою контролеру «AdminController», що відповідає за управління аспектами адміністрування системи, такими як замовлення та обладнання та редагування їх статусів. Для відображення списків замовлень та зміни його

статусів відповідають методи «listOrders» і «changeOrderStatus», що використовуються на сторінці адміністрування замовлень. Аналогічно методи «listEquipment» і «changeEquipmentStatus» відповідають за обладнання дослідників. Реалізація даного контролера подана на рисунку 3.5.

```
class AdminController extends AbstractController
{
    #[Route('/admin/equipment', name: 'app_admin_equipment', methods:
['GET'])]
    public function listEquipment(Request $request): Response
    {
        $offset = max(0, $request->query->getInt('offset'));
        $paginator = $this->em->getRepository(Equipment::class)-
>list($offset);
        $changeStatusForm = $this-
>createForm(EquipmentChangeStatusFormType::class);
    }
    #[Route('/admin/order', name: 'app_admin_order', methods:
['GET'])]
    public function listOrders(Request $request,
FilterBuilderUpdaterInterface $filterBuilderUpdater): Response
    {
        $form = $this->createForm(OrderFormFilter::class);
        $offset = max(0, $request->query->getInt('offset'));
        $qb = $this->em->getRepository(Order::class)-
>getOrderQbByAdmin($offset);
        if ($request->query->has($form->getName())) {
            $form->submit($request->query->all($form->getName()));
            $filterBuilderUpdater->addFilterConditions($form, $qb);
        }
        $paginator = new Paginator($qb);
    }
}
```

Рисунок 3.5 – Програмний код контролера «AdminController», що відповідає за адміністрування замовлень та обладнання

Розглянемо проектування моделі для збереження даних з веб-сторінки в базі даних. Клас «payment» представляє собою можливість оплати в системі. Даний клас містить методи та поля, що описують оплату досліджень.

Модель містить поля, що так же відображають таблицю бази даних «payment»:

– Id. Унікальний ідентифікатор платежу;

– User. Об’єкт класу User, що представляє поточного користувача;

– cardNumber. Номер картки для оплати дослідження, що був введений;

– expirationDate. Дата завершення дії картки, яка була введена під час оплати дослідження на відповідній веб-сторінці;

– cvv. Введений код платіжної карти.

Наведений клас моделі та її методів використовується для зберігання та обробки інформації про платіж, що забезпечить зв’язок між користувачами та їх платіжними даними. Реалізація даного класу подано на рисунку 3.6.

```
class Payment
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column(type: 'integer')]
    private int $id;
    #[ManyToOne(targetEntity: User::class)]
    #[JoinColumn(name: 'customer_id', referencedColumnName: 'id',
nullable: false)]
    private User $user;
    #[ORM\Column(type: 'string', length: 32)]
    private string $cardNumber;
    #[ORM\Column(type: 'string', length: 32)]
    private string $expirationDate;
    #[ORM\Column(type: 'string', length: 3)]
    private string $cvv;
    public function getId(): int
    {
        return $this->id;
    }
    public function getUser(): User
    {
        return $this->user;
    }
    public function setUser(User $user): void
    {
        $this->user = $user;
    }
    public function getCardNumber(): string
    {
        return $this->cardNumber;
    }
    public function setCardNumber(string $cardNumber): void
```

```
    {
        $this->cardNumber = $cardNumber;
    }
    public function getExpirationDate(): string
    {
        return $this->expirationDate;
    }
    public function setExpirationDate(string $expirationDate): void
    {
        $this->expirationDate = $expirationDate;
    }
    public function getCvv(): string
    {
        return $this->cvv;
    }
    public function setCvv(string $cvv): void
    {
        $this->cvv = $cvv;
    }
}
```

Рисунок 3.6 – Програмний код класу payment

Наступною розробленою моделлю можна продемонструвати equipment. Дана модель допомагає встановлювати та зберігати інформацію до бази даних про обладнання дослідників системи. Вона реалізує поля та методи необхідні для повноцінного функціонування пов’язаних функцій з обладнанням. Реалізація моделі equipment подано на рисунку3.7

```
class Equipment
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column(type: 'integer')]
    private int $id;
    #[ORM\Column(type: 'string', length: 2048)]
    private string $description;

    #[ManyToOne(targetEntity: User::class)]
    #[JoinColumn(name: 'contractor_id', referencedColumnName: 'id',
nullable: false)]
    private User $contractor;
    #[ORM\Column(type: 'integer', enumType: EquipmentStatus::class)]
    private EquipmentStatus $equipmentStatus;
    #[ORM\Column(type: 'string', length: 2048, nullable: true)]
    private ?string $note = null;
```

```
    #[ORM\Column(type: 'string')]
    private string $equipmentFileName;
    #[ORM\Column(type: 'datetime')]
    private DateTime $createdAt;
    public function __construct()
    {
        $this->createdAt = new DateTime();
        $this->equipmentStatus = EquipmentStatus::InReview;
    }
    public function getId(): int
    {
        return $this->id;
    }
    public function getDescription(): string
    {
        return $this->description;
    }
    public function setDescription(string $description): void
    {
        $this->description = $description;
    }
    public function getContractor(): User
    {
        return $this->contractor;
    }
    public function setContractor(User $contractor): void
    {
        $this->contractor = $contractor;
    }
    public function getEquipmentStatus(): EquipmentStatus
    {
        return $this->equipmentStatus;
    }
    public function setEquipmentStatus(EquipmentStatus
$equipmentStatus): void
    {
        $this->equipmentStatus = $equipmentStatus;
    }
    public function getNote(): ?string
    {
        return $this->note;
    }
    public function setNote(?string $note): void
    {
        $this->note = $note;
    }
    public function getEquipmentFileName(): string
    {
        return $this->equipmentFileName;
    }
```

```
    public function setEquipmentFileName(string $equipmentFileName):
void
    {
        $this->equipmentFileName = $equipmentFileName;
    }
    public function getCreatedAt(): DateTime
    {
        return $this->createdAt;
    }
    public function setCreatedAt(DateTime $createdAt): void
    {
        $this->createdAt = $createdAt;
    }
}
```

Рисунок 3.7 – Програмний код моделі equipment

На прикладі класу payment та equipment, які містять в собі реалізацію моделі та методів можна сказати, що таке структурування коду дозволяє чітко розділити логіку роботи з даними від самої моделі забезпечуючи інкапсуляцію даних і захист від некоректного використання. Саме проектування подібних класів може значно спростити написання коду та зробити його більш зрозумілим для інших розробників.

3.7 Розробка інтерфейсу користувачів та створення взаємодії з моделями і контролерами

Розробка інтерфейсу користувачів є самою важливою частиною проектування інформаційної системи. Присутність графічного інтерфейсу забезпечує зручну і ефективну можливість взаємодіяти з функціями системи, що були визначені та розроблені для бізнесу. Графічний інтерфейс, дозволяє легко та інтуїтивно взаємодіяти із системою, забезпечуючи швидкий доступ до даних.

Успішний інтерфейс повинен бути естетично привабливим і функціональним, забезпечуючи зручність і простоту використання. Спираючись на потреби та вимоги користувачів необхідно створити

інтерфейс, який мінімізує помилки та підвищить продуктивність взаємодії з системою.

Доступ до індивідуального функціоналу акторів інформаційної системи відбувається після реєстрації та авторизації. Даний процес є важливим, оскільки забезпечує безпеку та персоналізацію використання системи. На рисунку 3.8 продемонстровано сторінку реєстрації користувача.

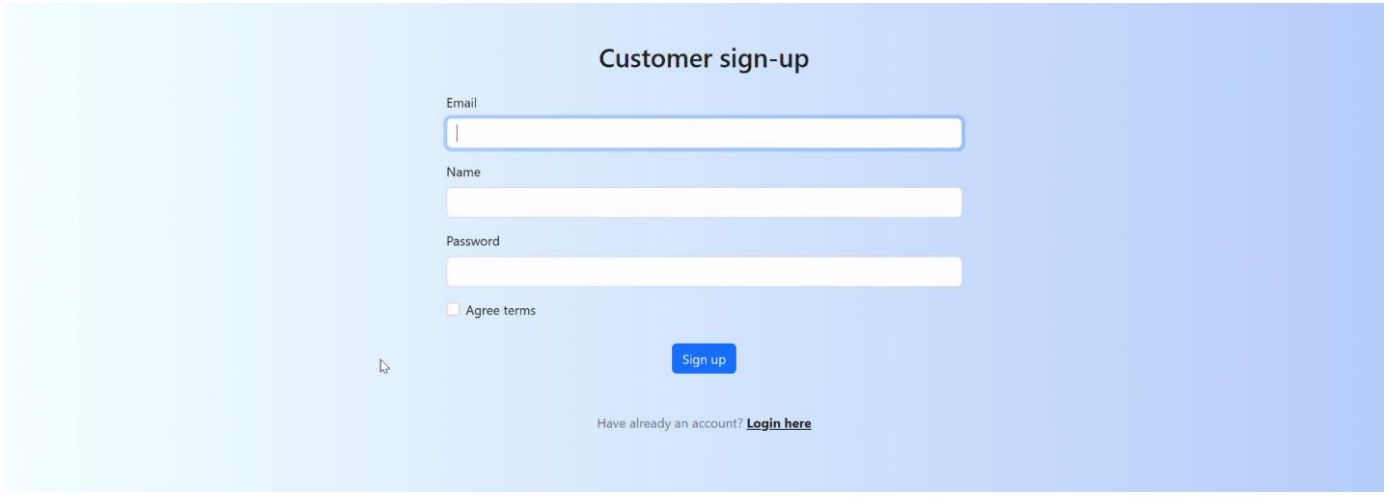

Рисунок 3.8 – Сторінка реєстрації нового користувача системи

Реалізація сторінки «реєстрація нового користувача» розроблена за допомогою шаблонізатору twig з використанням мови програмування PHP. На рисунку 3.9 подано програмний код реалізованої сторінки.

```
{% extends 'base.html.twig' %}
{% block title %}Register customer{% endblock %}
{% block body %}
    <div class="container mt-5">
        <div class="row justify-content-center align-items-center">
            <div class="col-md-6">
                <div class="text-center">
                    <h2 class="mb-4">Customer sign-up</h2>
                </div>
                {{ form_start(registrationForm) }}
                <div class="mb-3">
                    {{ form_errors(registrationForm) }}
                    {{ form_row(registrationForm.email) }}
                </div>
                <div class="mb-3">
                    {{ form_row(registrationForm.name) }}
```

```
                </div>
                <div class="mb-3">
                    {{ form_row(registrationForm.plainPassword) }}
                </div>
                <div class="mb-3">
                    <div class="form-check">
                        {{ form_row(registrationForm.agreeTerms) }}
                    </div>
                </div>
                <div class="text-center" style="margin-top: 30px">
                    <button type="submit" class="btn btn-primary">Sign
up</button>
                </div>
                <p class="text-center text-muted mt-5 mb-0">Have
already an account?
                    <a href="{{url('app_login')}}" class="fw-bold
text-body">
                        <u>Login here</u>
                    </a>
                </p>
                {{ form_end(registrationForm) }}
            </div>
        </div>
    </div>
{% endblock %}
```

Рисунок 3.9 – Програмний код сторінки «реєстрація нового користувача»

Ввівши інформацію для створення особового профілю, саме електронну адресу, ім’я та пароль ми можемо надіслати запит на реєстрацію, після чого на основі введених даних буде зареєстровано нового користувача системи. Пройшовши даний етап, ми отримаємо роль замовника та будемо перенаправлені на сторінку введення платіжної інформації своєї картки, для майбутньої оплати проведених досліджень.

Сторінка «payment» потребує введення номеру картки, дати її закінчення та cvv-коду. Графічний інтерфейс користувача даної сторінки подано на рисунку 3.10.

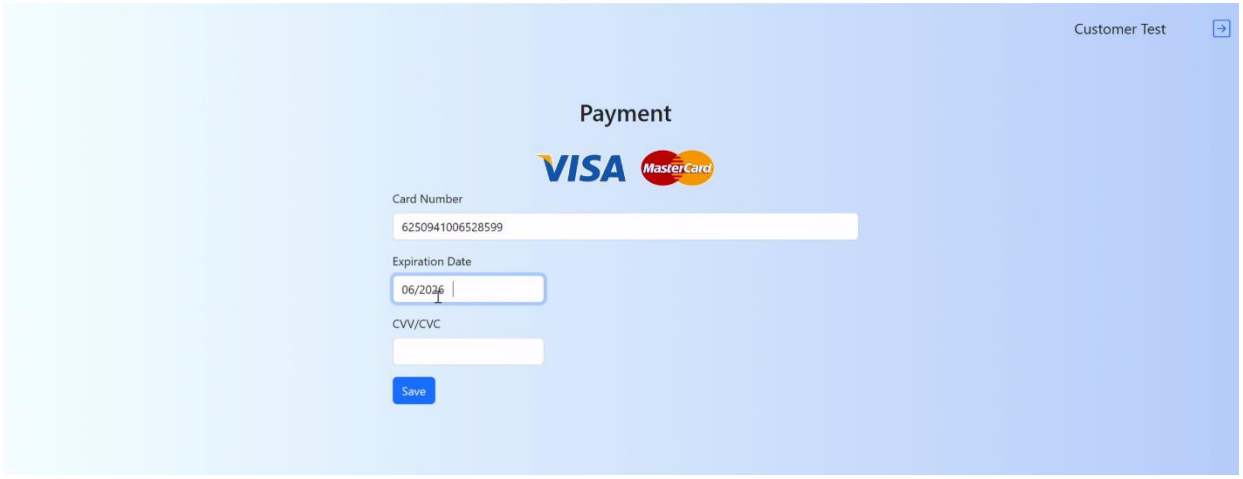

Рисунок 3.10 – Сторінка введення платіжної інформації

Реалізована сторінка містить форму з можливістю введення необхідної інформації. Сторінка реалізована з використання шаблонізатору twig, що використовує HTML та мови програмування PHP. Програмний код сторінки подано на рисунку 3.11.

```
{% extends 'base.html.twig' %}
{% block title %}Payment{% endblock %}
{% block body %}
    <div class="container mt-5">
        <div class="row justify-content-center align-items-center">
            <div class="col-md-6">
                <div class="text-center">
                    <h2 class="mb-4">Payment</h2>
                </div>
                <div class="text-center">
                    <img style="width: 250px" src="{{ asset('visa-and-
mastercard-logo.png') }}" class="rounded mx-auto d-block">
                </div>
                {{ form_start(form) }}
                <div class="mb-3">
                    {{ form_row(form.cardNumber) }}
                </div>
                <div class="input-group mb-3">
                    {{ form_row(form.expirationDate) }}
                </div>
                <div class="input-group mb-3">
                    {{ form_row(form.cvv) }}
                </div>
                <button type="submit" class="btn btn-
primary">Save</button>
                {{ form_end(form) }}
```

```
            </div>
        </div>
    </div>
{% endblock %}
```

Рисунок 3.11 – Програмний код сторінки «введення платіжної інформації»

Сторінка створення замовлення є обов'язковим елементом інтерфейсу користувача для замовника системи, оскільки його мета полягає в створенні нових замовлень на дослідження даних. Сторінка «створення досліджень» містить форми для введення головної інформації про дослідження, такі як деталі, дати проведення, опис, ціна та за необхідністю координати. Це дозволить дослідникам чітко зрозуміти, що потрібно виконати та на який результат чекає замовник. Сторінка створення нового замовлення наведена на рисунку 3.12.

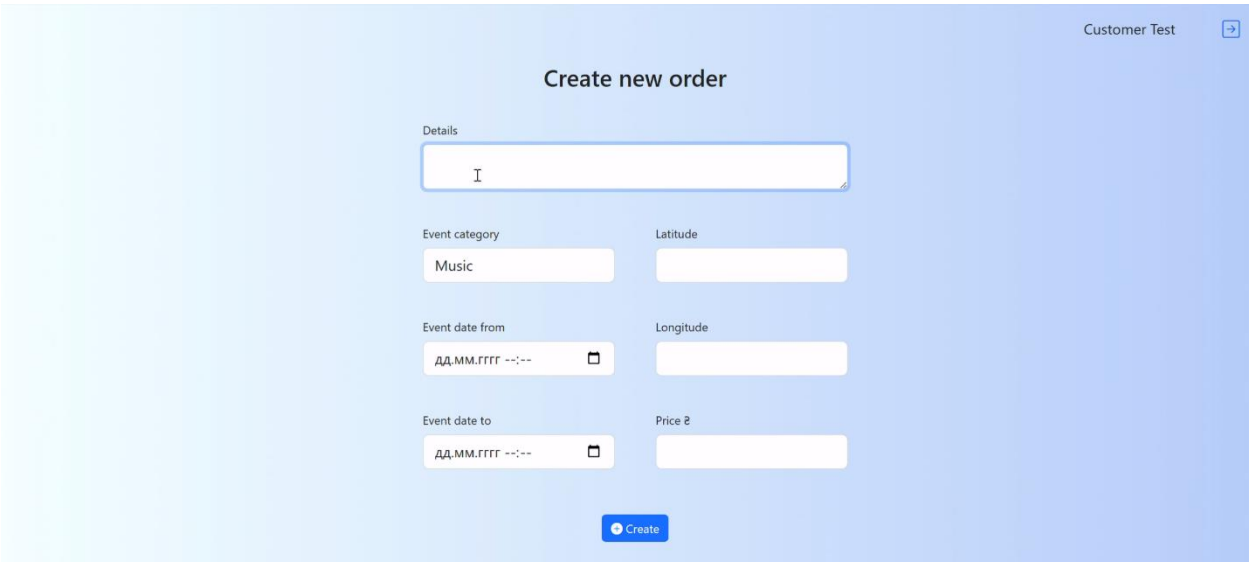

Рисунок 3.12 – Сторінка «створення нового дослідження»

Реалізована сторінка містить форму з можливістю введення необхідної інформації. Сторінка реалізована з використанням шаблонізатору twig, що використовує HTML та мови програмування PHP. Програмний код сторінки наведено в лістингу 3.13.

```
{% extends 'base.html.twig' %}
```

```
{% block title %}Create new order{% endblock %}

{% block body %}
    <div class="container mt-7">
        <div class="row justify-content-center align-items-center">
            <div class="col-md-6">
                <div class="text-center">
                    <h2 class="mb-4">Create new order</h2>
                </div>
                {{ form_errors(newOrderForm) }}
                {{ form_start(newOrderForm) }}
                <div class="container">
                    <div class="row">
                        <div class="mb-3">
                            <div class="p-3">
                                {{ form_row(newOrderForm.details) }}
                            </div>
                        </div>
                    </div>
                    <div class="row">
                        <div class="col">
                            <div class="mb-3">
                                <div class="p-3">
                                    {{
form_row(newOrderForm.eventCategory) }}
                                </div>
                            </div>
                            <div class="mb-3">
                                <div class="p-3">{{
form_row(newOrderForm.eventDateFrom) }}</div>
                            </div>
                            <div class="mb-3">
                                <div class="p-3">{{
form_row(newOrderForm.eventDateTo) }}</div>
                            </div>
                        </div>
                        <div class="col">
                            <div class="mb-3">
                                <div class="p-3">{{
form_row(newOrderForm.latitude) }}</div>
                            </div>
                            <div class="mb-3">
                                <div class="p-3">{{
form_row(newOrderForm.longitude) }}</div>
                            </div>
                            <div class="mb-3">
                                <div class="p-3">{{
form_row(newOrderForm.price) }}</div>
                            </div>
                        </div>
                    </div>
```

```
                </div>
                <div class="text-center" style="margin-top: 30px">
                    <div class="btn-group">
                        <button type="submit" class="btn btn-primary">
                            <i class="bi bi-plus-circle-fill"></i>
                            Create
                        </button>
                    </div>
                </div>
                {{ form_end(newOrderForm) }}
            </div>
        </div>
    </div>
{% endblock %}
```

Рисунок 3.13 – Програмний код сторінки «створення нового дослідження»

Розглянемо деякий функціонал, що доступний тільки для зареєстрованих дослідників ІС взаємодії з даними. Зареєструвавшись та авторизувавшись в системі досліднику необхідно надати інформацію про особисте обладнання, що буде використовуватись для проведення досліджень. Для цього йому необхідно відвідати сторінку «Обладнання» на якій необхідно надати опис свого обладнання та прикріпити фотографії для додаткового підтвердження.

Після заповнення всіх необхідних даних та завантаження фото і документів дослідник відправляє інформацію на перевірку. Адміністратор даних отримує запит на проведення сертифікації обладнання та має можливість перевірити надану інформацію та визначити тип статусу обладнання дослідника.

В залежності від статусу обладнання можна зрозуміти чи була отримана сертифікація чи ні, що в свою чергу надає можливість оформлювати підписку на проведення досліджень із використанням зареєстрованого обладнання.

На рисунку 3.14 подано сторінку надання інформації про обладнання.

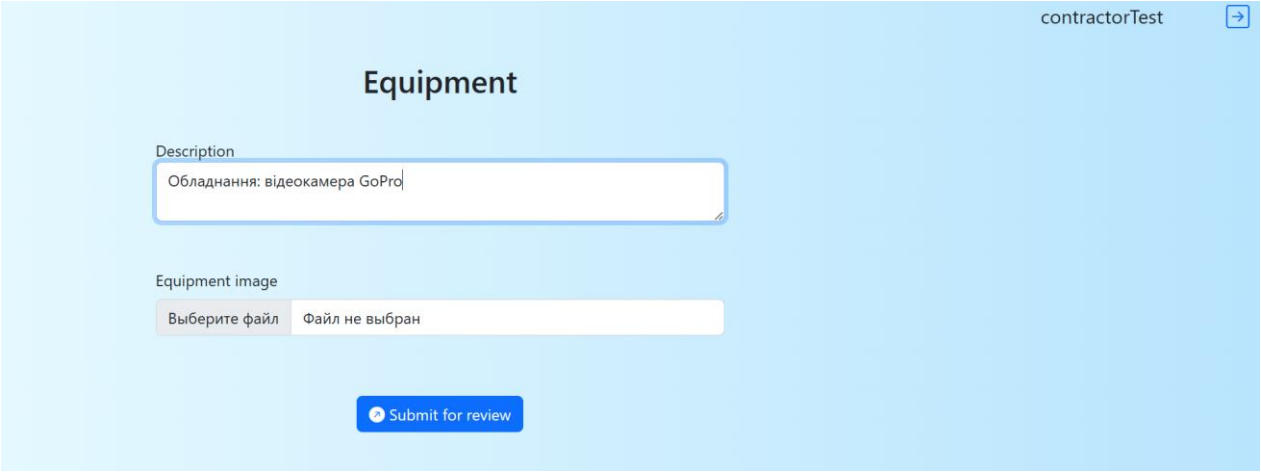

Рисунок 3.14 – Сторінка «Надання інформації про особисте обладнання»

Реалізована сторінка містить форму з можливістю введення необхідної інформації. Сторінка реалізована з використання шаблонізатору twig, що використовує HTML та мови програмування PHP. Програмний код сторінки подано на рисунку 3.15.

```
{% extends 'base.html.twig' %}
{% block stylesheets %}
    {{ parent() }}
    <style type="text/css">
        body {
            background-image: linear-gradient(to right, #f3ffff,
#b8e4fd);
        }
    </style>
{% endblock %}
{% block body %}
    <div class="container mt-7">
        <div class="row justify-content-center align-items-center">
            <div class="col-md-6">
                <div class="text-center">
                    <h2 class="mb-4">Equipment</h2>
                </div>
                {{ form_errors(equipmentForm) }}
                {{ form_start(equipmentForm) }}
                <div class="container">
                    <div class="row">
                        <div class="mb-3">
                            <div class="p-3">
```

```
                                {{ form_row(equipmentForm.description)
}}
                            </div>
                        </div>
                    </div>
                    <div class="row">
                        <div class="mb-3">
                            <div class="p-3">
                                {{ form_row(equipmentForm.equipment)
}}
                            </div>
                        </div>
                    </div>
                </div>
                <div class="text-center" style="margin-top: 30px">
                    <div class="btn-group">
                        <button type="submit" class="btn btn-primary">
                            <i class="bi bi-arrow-up-right-circle-
fill"></i>
                            Submit for review
                        </button>
                    </div>
                </div>
                {{ form_end(equipmentForm) }}
            </div>
        </div>
    </div>
{% endblock %}
```

Рисунок 3.15 – Програмний код сторінки «Надання інформації про особисте обладнання»

Перегляд особистих замовлень та результатів дослідження є кінцевим етапом взаємодії з системою для замовників. Форматування результатів дослідження та повторне їх використання може стати необхідною вимогою для користувачів системи. Графічний інтерфейс списку особистих досліджень дозволяє переглядати деталі кожного замовлення та отримані результати. Такий функціонал дозволяє відстежувати прогрес виконання досліджень. Сторінка особистих замовлень подана на рисунку 3.16, яка є важливою частиною інтерфейсу для користувачів ІС.

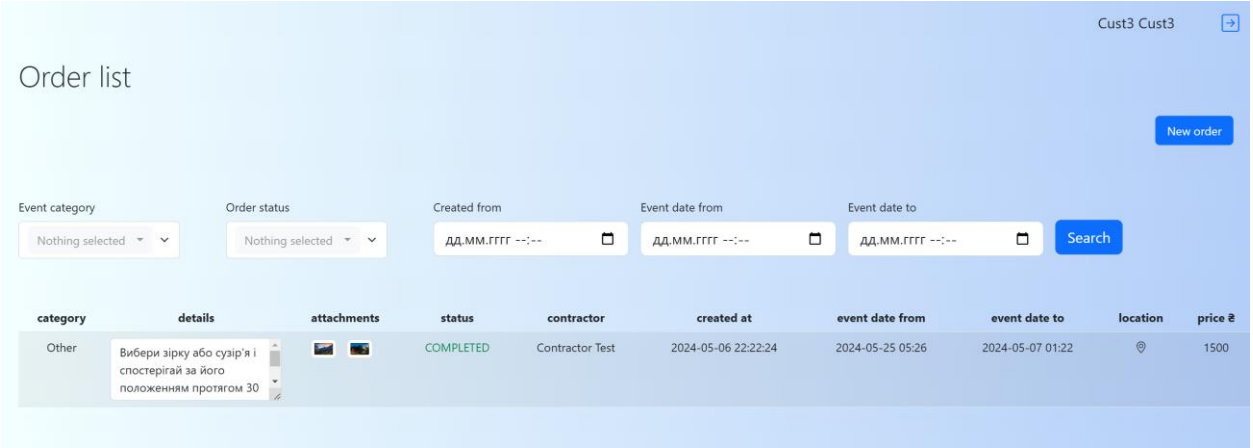

Рисунок 3.16 – Сторінка «список створених досліджень замовника»

Реалізована сторінка містить детальну інформацію про створені замовлення. Сторінка реалізована з використання шаблонізатору twig, що використовує HTML та мови програмування PHP. Програмний код сторінки подано на рисунку 3.17.

```
{% extends 'base.html.twig' %}
{% block title %} Order list {% endblock %}
{% block body %}
    <h1 class="display-6">Order list</h1>
    <nav class="nav nav-pills justify-content-end p-4">
        <a class="nav-link active" aria-current="page"
href="{{url('app_customer_new_order')}}">New order</a>
    </nav>
    {% include 'order_filters.html.twig' %}
    <table class="table table-striped table-hover">
        <thead>
        <tr>
            <th>category</th>
            <th style="width: 250px">details</th>
            <th>attachments</th>
            <th>status</th>
            <th>contractor</th>
            <th>created at</th>
            <th>event date from</th>
            <th>event date to</th>
            <th>location</th>
            <th>price ₴</th>
        </tr>
        </thead>
        <tbody>
        {% for order in orders %}
```

```
            <tr>
                <td>{{ order.eventCategory.value }}</td>
                <td>
                    <textarea class="form-control" rows="3">{{
order.details }}</textarea>
                </td>
                <td>
                    {% if order.contractor is not empty %}
                        {{ order.contractor.name }}
                    {% endif %}
                </td>
                <td>{{ order.createdAt.format('Y-m-d H:i:s') }}</td>
                <td>{{ order.eventDateFrom.format('Y-m-d H:i') }}</td>
                <td>{{ order.eventDateTo.format('Y-m-d H:i') }}</td>
                <td>
                    <i class="bi bi-geo-alt" data-bs-toggle="modal"
data-bs-target="#mapModal" onclick="showMap({{ order.latitude }}, {{
order.longitude }})"></i>
                </td>
                <td>{{ order.price }}</td>
            </tr>
        {% endfor %}
        </tbody>
    </table>
{% endblock %}
```

Рисунок 3.17 – Програмний код сторінки «список створених досліджень замовника»

Проектування інтерфейсу користувача для інформаційної системи збору, обробки та розповсюдження даних має забезпечувати зрозумілий та ефективний доступ до функціоналу системи. Одним із важливих аспектів проектування системи є створення замовлень на різні теми та повторне використання результатів проведеного дослідження.

ВИСНОВОК

Проводячи аналіз предметної області було детально розглянуто існуючі аналоги по збору, обробки та розповсюдження інформації. Головним конкурентом нашої системи в області космічного простору є CoLiTec, яка має велику кількість плагінів та функцій щодо спостереження в автоматизованому режимі.

Проектуючи інформаційну систему було детально проведено функціональне моделювання бізнес-процесу та бізнес-функцій системи, які були визначені в ході роботи. Саме проектування бізнес-функцій та проведення їх декомпозицій дозволило чітко зрозуміти та визначити вимоги до бізнес-логіки системи та проведення моделювання процесів і їх взаємозв’язків за допомогою використання уніфікованої мови моделювання (UML). Даний підхід забезпечив структурованість і впорядкованість під час розробки, що є важливим аспектом під час створення надійної системи [14].

Написання серверної частини застосунку було виконано з використанням PHP та фреймворку Symfony. Використання Symfony дозволило скористатися потужними інструментами для розробки, такими як управління маршрутизацією, автентифікацією та авторизацією користувачів, а також інтеграцією з базами даних через Doctrine ORM [11].

Клієнтська частина застосунку була реалізована з використанням шаблонізатора Twig, який забезпечує зручність і гнучкість у створенні інтерфейсу користувача [6].

Підсумовуючи, детальне проектування бізнес-функцій з використанням технологій та інструментів для написання програмного забезпечення дозволило створити потужну та гнучку до змін інформаційну систему.

АПРОБАЦІЇ

Розробка компонентів збору, обробки та розповсюдження даних інформаційної системи пройшла повний комплекс, що включав в себе наукову і практичну частину.

Метою кваліфікаційної роботи було створення ефективної та гнучкої інформаційної системи збору, обробки та розповсюдження даних. Проведення апробація для даної теми була необхідна для визначення та підтвердження новизни та необхідності системи для проведення результатів дослідження в будь-якій сфері.

Обрану тему для апробації було представлено для конференції «Інформаційні інтелектуальні системи» в м. Харків, 16-18 квітня 2024 року.

У доповіді було подано тему «Розробка компонентів збору, обробки та розповсюдження даних інформаційної системи».

В ході конференції було обговорено переваги та недоліки технологій та отримано рекомендації, щодо покращення функціональності розробленої інформаційної системи:

Апробації результатів дослідження дозволила об’єктивно визначити значущість інформаційної системи збору, обробки та розповсюдження даних для користувачів і бізнесу. Зокрема, було підкреслено такі ключові аспекти як можливість адаптації системи до різних потреб користувачів та специфічних бізнес-процесів і визначено високий потенціал використання системи в різних галузях, включаючи бізнес, науку та промисловість.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Петров Е.Г., Новожилова М.В., Гребеннік І.В., Соколова Н.А. Методи та засоби прийняття рішень у соціально-економічних та технічних системах: Херсон: Олді – плюс, 2003. – 380 с.

2. І.В.Гребеннік, М.Ю.Вишняк, В.Г.Іванов, З.А.Імангулова, Н.І.Калита Елементи системного проектування (за редакцією І.В.Гребенніка): Навч. посібник. – Харків: ХНУРЕ, 2016. – 322 с.

3. Наконечний О. Г., Гребеннік І. В., Романова Т. Є., Тевяшев А. Д., Методи прийняття рішень: Навч. посібник. – Харків: ХНУРЕ, 2016. – 132 с.

4. Гребеннік І.В., Коваленко А.І., Міщеряков Ю.В., Решетнік В.М., Титов С.В. Системне програмування – Х.:ХНУРЕ, 2017 – 374 с. // I. Grebennik, A. Kovalenko, I. Mishcheriakov, V.Reshetnik, S.Titov System programming: Tutorial. Kh.: NURE, 2018 – 374 p.

5. Нефьодов Л. І., Невлюдов І. Ш., Безкоровайний В. В. CALS-технології і системи: навч. посібник. Харків: ХНУРЕ, 2021. 272 с.

6. Documentation - Twig - The flexible, fast, and secure PHP template engine. Home - Twig - The flexible, fast, and secure PHP template engine. URL: https://twig.symfony.com/doc/ (date of access: 27.05.2024).

7. Symfony Documentation. Symfony, High Performance PHP Framework for Web Development. URL: https://symfony.com/doc/current/index.html (date of access: 27.05.2024).

8. Каряка В.В. Розробка компонентів збору, обробки та розповсюдження даних інформаційної системи: 28-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. матеріалів форуму. Т. 6., – Харків: ХНУРЕ. 2024. С. 185-186 PHP: PHP Manual - Manual. PHP: Hypertext Preprocessor. URL: https://www.php.net/manual/en/ (date of access: 27.05.2024).

9. PHP: Classes and Objects - Manual. PHP: Hypertext Preprocessor. URL: https://www.php.net/manual/en/language.oop5.php (date of access: 27.05.2024).

10. Laravel vs Symfony: The Best PHP Framework in 2024?. The Official Cloudways Blog. URL: https://www.cloudways.com/blog/laravel-vs-symfony/ (date of access: 27.05.2024).

11. Б.А Новиков, Е.А Горшкова, Н.Г Графеева. Основы технологий баз данных. 2-ге вид. ДМК Пресс, 2020. 583 с.

12. Моралес Дж. What is UML Class Diagram including UML Class Diagram Maker. MindOnMap | Free Mind Mapping Tool to Draw Ideas Easily Online. URL: https://www.mindonmap.com/uk/blog/what-is-uml-class-diagram/ (date of access: 27.05.2024).

13. Діаграми UML для моделювання процесів і архітектури проекту. Evergreen - web розробка і діджиталізація бізнесу за допомогою AI продуктів. URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення: 27.05.2024).

14. UML діаграми, їхні основні типи та процес розроблення. FoxmindEd. URL: https://foxminded.ua/uml-diagramy/ (дата звернення: 27.05.2024).

15. OTUS. Главные причины, почему мы разрабатываем веб-приложения на Symfony. Хабр. URL: https://habr.com/ru/companies/otus/articles/564532/ (дата звернення: 27.05.2024).

16. What is LEMP Stack? - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/what-is-lemp-stack/ (date of access: 27.05.2024).

17. Controller (Symfony Docs). Symfony, High Performance PHP Framework for Web Development. URL: https://symfony.com/doc/current/controller.html (date of access: 27.05.2024).

18. Creating and Using Templates (Symfony Docs). Symfony, High Performance PHP Framework for Web Development. URL: https://symfony.com/doc/current/templates.html (date of access: 27.05.2024).

19. Головна | Elib LNTU. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/Кондіус%202%20готовва/page9.html (дата звернення: 27.05.2024).

20. Documentation for MySQL-server: офіційна документація. URL: https://www.mysql.com/ (дата звернення: 01.05.2024).

21. Introduction | v6 | StarUML documentation. Introduction | v6 | StarUML documentation. URL: https://docs.staruml.io/ (дата звернення: 26.05.2024).

22. UML Class Diagram Tutorial. Lucidchart. URL: https://www.lucidchart.com/pages/uml-class-diagram (дата звернення: 26.05.2024).

23. UML Use Case Diagram Tutorial. Lucidchart. URL: https://www.lucidchart.com/pages/uml-use-case-diagram (дата звернення: 26.05.2024).