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 та дослідження інформаційної системи для моніторингу

та координації гуманітарної допомоги
(тема)

Виконав:
студент *II* курсу, групи *СПРм-22-2*

Максимов Я.О.
(прізвище, ініціали)

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

Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Системне проектування*
(повна назва освітньої програми)

Керівник *доц. Петрова Р. В.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *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.

Дата 17.06.24 Максимов Я.О.

Підпис 

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.

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

Попередній захист проведено «17» червня 2024 р.

Керівник кваліфікаційної роботи доц. Петрова Р.В 

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*

Кафедра *Системотехніки*

Рівень вищої освіти *другий (магістерський)*

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

Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Системне проектування*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Максимову Яну Олеговичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *«Розробка та дослідження інформаційної системи для моніторингу та координації гуманітарної допомоги»*
затверджена наказом університету від *«01» квітня 2024 р.* № *259 Ст*
2. Термін подання студентом роботи до екзаменаційної комісії *«18» червня 2024 р.*
3. Вихідні дані до роботи *Об'єкт дослідження – інформаційна система для моніторингу та координації гуманітарної допомоги. Предмет дослідження – методи моніторингу ефективності гуманітарної допомоги. Об'єкт розробки – додаток для оптимізації ефективності моніторингу та координації гуманітарної допомоги. Технічне забезпечення: IBM-сумісний персональний комп'ютер. Перелік використовуваних програмних засобів: ОС Microsoft Windows 10, мова або пакет імітаційного моделювання.*
4. Перелік питань, що потрібно опрацювати в роботі *4.1 Перелік скорочень та термінів. 4.2 Вступ. 4.3 Аналіз предметної області. 4.4 Бізнес-можливості. 4.5 Бізнес-вимоги. 4.6 Бізнес-ризики. 4.7 Особливості продукту. 4.8 Визначення основних бізнес-функцій. 4.9 Визначення функцій інтерфейсу користувальницької частини. 4.10 Організаційна структура. 4.11 Аналіз існуючих ПЗ. 4.12 Переваги програмного забезпечення 4.13 Постановка мети і задач кваліфікаційної роботи. 4.14 Загальні вимоги до розроблювальної системи. 4.15 Аналіз проблемних сторін прототипу та шляхи їх вирішення. 4.16 Серверна частина програми. 4.17 Клієнтська частина програмного застосунку. 4.18 Проектування архітектури системи. 4.19 Обґрунтування вибору мови програмування. 4.20 Обґрунтування вибору СУБД. 4.21 Створення функціональної моделі ІС. 4.22 Склад сервісів ПЗ. 4.23 Визначення функцій серверної частини ІС. 4.24 UML – моделювання клієнтської частини ІС. 4.25 USE CASE діаграма. 4.26 Діаграма послідовності. 4.27 Діаграма класів. 4.28 Опис технічної бази та середовища розробки. 4.29 Створення бази даних. 4.30 Розробка бізнес-логіки системи. 4.31 Заповнення даними БД. 4.32 Розробка інтерфейсу клієнської частини. 4.33 Тестування розробленого ПЗ. 4.34 Висновки. 4.35 Керівництво користувача. 4.36. Текст програми.*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п. 5 включається до завдання за рішенням випускової кафедри) *5.1 Схема бази дани. 5.2 Діаграма IDEF0. 5.3 Діаграма декомпозиції. 5.4 USE CASE діаграма. 5.5 Діаграма послідовності. 5.6 Діаграма класів.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Отримання та аналіз завдання, уточнення плану роботи*	*13.01.24*	виконано
2	*Аналіз формування заявки на допомогу як об'єкта*		виконано
	автоматизації	*26.01.24*	виконано
3	*Огляд існуючих моделей та методів моніторингу та*		виконано
	координації гуманітарної допомоги	*01.02.24*	виконано
4	*Розробка вимог до інформаційної системи*	*20.02.24*	виконано
5	*Розробка програмного забезпечення задачі*	*14.03.24*	виконано
6	*Проведення експериментальних досліджень*	*26.03.24*	виконано
7	*Оформлення пояснювальної записки*	*04.04.24*	виконано
8	*Підготовка презентації*	*01.05.24*	виконано
9	*Подання закінченої роботи науковому керівникові*	*02.06.24*	виконано
10	*Подання роботи на рецензування*	*14.06.24*	виконано
11	*Попередній захист*	*17.06.24*	виконано
12	*Подання роботи до екзаменаційної комісії*	*18.06.24*	виконано

Дата видачі завдання *«13» січня 2024 р.*

Студент _____ Макс
(підпис)

Керівник роботи _____ *доц. Петрова Р. В.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Звіт з передатестаційної практики містить: 74 с., 35 рис., 1 табл., 25 джерел.

ІНФОРМАЦІЙНА СИСТЕМА, ЖИТТЄВИЙ ЦИКЛ, ВОЛОНТЕР, КОРИСТУВАЧ, ВОДІЙ, UML, DFD, ERD, БАЗА ДАНИХ, СЕРВЕР, СИСТЕМА, ДОДАТОК.

Об’єкт дослідження – інформаційна система, призначена для збору, обробки, аналізу та координації даних, пов'язаних з наданням гуманітарної допомоги.

Предмет дослідження – методи моніторингу ефективності гуманітарної допомоги з використанням інформаційної системи.

Мета роботи – розробка та дослідження окремих елементів і функцій інформаційної системи для підвищення ефективності моніторингу та координації гуманітарної допомоги в реальному часі, з використанням інтегрованого підходу.

Система буде розроблятись з використанням патерну репозиторію, де за клієнтську частину відповідає технологія створення веб-додатків Asp .NetMVC, що включає в себе написання сторінок мовою HTML та підключенням css (таблиць стилей) та js. В якості системи управління базами даних було обрано SQL Server Management Studio — система управління базами даних, яка розробляється корпорацією Microsoft.

Об’єктом розробки є додаток для оптимізації ефективності моніторингу та координації гуманітарної допомоги, який може бути використаний волнтерськими організаціями або іншими установами у яких є потреба розширити сферу використання не тільки в живу, а і у Інтернеті.

ABSTRACT

The report on pre-certification practice contains: 74 pages, 35 figures, 1 table, 25 sources.

INFORMATION SYSTEM, LIFE CYCLE, VOLUNTEER, USER, DRIVER, UML, DFD, ERD, DATABASE, SERVER, SYSTEM, APP.

The object of the study is an information system designed for the collection, processing, analysis and coordination of data related to the provision of humanitarian aid.

The subject of the research is methods of monitoring the effectiveness of humanitarian aid using an information system.

The purpose of the work is the development and research of individual elements and functions of the information system to increase the effectiveness of monitoring and coordination of humanitarian aid in real time, using an integrated approach.

The system will be developed using a pattern repository, where the technology for creating Asp .NetMVC web applications is responsible for the client part, which includes writing pages in HTML and connecting css (style sheets) and js. The database management system will be SQL Server Management Studio, a database management system developed by Microsoft Corporation.

The object of development is an application for optimizing the effectiveness of monitoring and coordination of humanitarian aid, which can be used by volunteer organizations or other institutions that need to expand the scope of use not only live, but also on the Internet.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

СУБД – Система управління базами даних.

БД – База даних.

Транзакція – Група послідовних операцій.

Фреймворк – Набір інструментів, бібліотек та правил, який використовується для створення програмних додатків.

ІС – Інформаційна система.

UML – Уніфікована мова моделювання (Unified Modeling Language).

IDEF0 – Методологія функціонального моделювання і графічного опису процесів, призначена для формалізації і опису бізнес-процесів.

SQL – Діалогова мова програмування для здійснення запиту і внесення змін до бази даних, а також керування базами даних.

HACS – Система для моніторингу та координації гуманітарної допомоги (Humanitarian Aid Coordination System).

ERD – Модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків (Entity-Relation Diagram).

DFD – Методологія графічного структурного аналізу (Data Flow Diagram).

Користувач – Людина яка робить запит на доставку та отримання гуманітарної допомоги.

Волонтер – Працівник волонтерської організації, який оброблює запити на отримання гуманітарної допомоги.

Водій - Працівник волонтерської організації, який доставляє гуманітарну допомогу.

MVP – Продукт з мінімальним функціоналом (Minimum viable product).

WA – Рішення яке включає веб-додаток.

SADT – Методологія структурного аналізу і проектування (Structured Analysisand Design Technique).

ПЗ – Програмне забезпечення.

ВСТУП

Розвиток інформаційних систем та впровадження їх у повсякденне життя стало невід'ємною частиною розвитку інформаційних технологій. В зв'язку з ситуаціями, які відбуваються в наше сьогодення, створення та впровадження систем для волонтерських організацій є основним принципом для допомоги людям.

Бувають випадки, коли в певних тяжких ситуаціях необхідно організувати та допомогти населенню, яке опинилося в осередку лиха. Потреба в допомозі може бути зумовлена з різних причин, таких як: війна, епідемія серед населення, випадках коли небезпека може бути природнього або техногенного характеру, які загрожують або можуть загрожувати життю людей. Загалом, коли виникають подібні ситуації і вони стають масовими, чітка та скоординована взаємодія між різними суб'єктами, такими як місцеві чи міжнародні організації, приватні компанії, волонтери, є дуже важливою і необхіною. Сучасні технології, які можуть обробляти великі обсяги даних, забезпечують швидку та ефективну взаємодію між суб'єктами та населенням, яке потребує допомоги. Сучасні системи, які займаються моніторингом та координацією гуманітарної допомоги мають велике значення у вирішенні непередбачуваних ситуацій і допомагають швидкому їх подоланню.

Сьогодні існує достатня кількість різноманітних організацій і компаній або невеликих груп волонтерів, які пропонують свою допомогу для надання гуманітарної допомоги людям, які постраждали від лиха. Але практично жодна гуманітарна організація чи група людей не в змозі задовольнити всі гуманітарні запити, тому виникає потреба в створені системі в якій необхідно об'єднати наявні в цих організаціях ресурси реагування на надзвичайні ситуації.

Розроблювальна система передбачає наявність центральної системи, яка буде виконувати головні функції, такі як моніторинг та доставка гуманітарної допомоги постраждалим. Завдяки великим системам обробки даних можна автоматизувати координацію реалізації різних видів гуманітарної допомоги. За

допомогою таких систем можна забезпечити наскрізну оперативну видимість, забезпечуючи швидке реагування та кращу якість надання допомоги населенню. Гуманітарна координація має вирішальне значення для здатності задовільнити населення в їх потребах та ефективно діяти під час стихійних лих чи конфліктів.

Метою роботи є розробка інформаційної системи за допомогою якої буде відбуватися моніторинг та координація гуманітарної допомоги населенню. В розроблювальній системі буде можливість долучитись в допомозі для надання гуманітарної допомоги постраждалим, а також запросити допомогу, яку людина потребує.

Для досягнення поставленої мети потрібно виконати:

- аналіз існуючих інформаційних систем, виділення їх переваг та недоліків;
- аналіз засобів проектування інформаційних систем;
- аналіз систем автоматизації бізнес-процесів;
- проектування інформаційної системи для моніторингу та координації гуманітарної допомоги;
- автоматизацію процесів проектування інформаційної системи для моніторингу та координації гуманітарної допомоги в системі HACS (Humanitarian Aid Coordination System).

Об’єкт дослідження – інформаційна система для моніторингу та координації гуманітарної допомоги.

Предмет дослідження – методи моніторингу ефективності гуманітарної допомоги з використанням інформаційної системи.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ СИСТЕМ

1.1 Аналіз предметної області

У сучасному світі несподівані гуманітарні виклики та події можуть призвести до несподіваних ситуацій, які вимагають оперативного реагування та оперативного втручання для налагодження ситуації. Організаційна координація та співпраця між різними суб’єктами має вирішальне значення в складних ситуаціях для забезпечення швидкої та ефективної доставки гуманітарної допомоги. Метою такої взаємодії є забезпечення оперативної допомоги постраждалим через технічну інфраструктуру системи, що підтримується наданням ресурсів та координацією дій.

Однією з проблем існуючих систем подачі заявок на гуманітарну допомогу є їх багаторівневий характер і відсутність прозорості. Клієнту часто доводиться шукати відповідну організацію чи волонтера, які можуть надати необхідну допомогу, перш ніж подати запит. Також існують проблеми в управлінні та обробці великої кількості запитів, які надходять від клієнтів, що може призвести до проблем з логістикою та доставкою ресурсів до місця призначення. Іншою поширеною проблемою є труднощі спілкування через месенджер або телефонні дзвінки, що ускладнює моніторинг та координацію доступних ресурсів.

Враховуючи вищезазначені проблеми, вирішенням яких може стати інформаційна система моніторингу та координації гуманітарної допомоги, можна спростити багатоетапний процес, а саме прискорити взаємодію між організаціями та клієнтами які потребують допомоги, забезпечивши єдину платформу для реєстрації, взаємодії та обробки запитів на гуманітарну допомогу. В системі потрібно забезпечити прозорість та ефективну координацію допомоги, надаючи доступ до інформації про наявність тих чи інших ресурсів, їх статус та місцезнаходження. Інтеграція інструментів комунікації та систем моніторингу

стану може полегшити комунікацію між учасниками та забезпечити ефективнішу координацію дій.

Таким чином, впровадження системи моніторингу та координації гуманітарної допомоги може полегшити управління та організацію процесу доставки допомоги до потерпілих, скоротити час реагування на кризові ситуації та покращити результати.

1.1.1 Бізнес-можливості

В сучасному світі виникає велика потреба в простій та надійній системі, яка зможе допомогти постраждалим отпримати необхідну допомогу. Гуманітарна допомога є різновидом благодійництва і має спрямовуватися відповідно до обставин, об’єктивних потреб, згоди її отримувачів та за умови дотримання вимог статті 4 Закону України "Про благодійництво та благодійні організації". Відповідно до Закону України "Про гуманітарну допомогу" гуманітарна допомога – це є цільова адресна безоплатна допомога в грошовій або натуральній формі, у вигляді безповоротної фінансової допомоги або добровільних пожертвувань, або допомога у вигляді виконання робіт, надання послуг, що надається іноземними та вітчизняними донорами із гуманних мотивів отримувачам гуманітарної допомоги в Україні або за кордоном, які потребують її у зв’язку з соціальною незахищеністю, матеріальною незабезпеченістю, важким фінансовим становищем, виникненням надзвичайного стану, зокрема внаслідок стихійного лиха, аварій, епідемій і епізоотій, екологічних, техногенних та інших катастроф, які створюють загрозу для життя і здоров’я населення, або тяжкою хворобою конкретних фізичних осіб.

Розроблювальна система буде відноситись до некомерційних організацій, в якої на меті є швидке надання безоплатної допомоги постраждалим.

На сьогоднішній день існує кілька програмних рішень для автоматизації координації з надання гуманітарної допомоги, що включають замовлення гуманітарної допомоги - Благороб, ZayavaInfo, єДопомога (Волонтерська). Вони

використовують програмний додаток з великою кількістю форм для отримання допомоги.

Система Humanitarian Aid Coordination System повинна позбавити користувачів даної проблеми і надати їм менш складне і набагато більш зручне використання системи для отримання гуманітарної допомоги. Для MVP робочий процес складається з наступних кроків:

1. На сайті користувач може ознайомитися з наявними видами допомоги і доступними в даний час.

2. Користувач самостійно обирає допомогу, щоб оформити отримання допомоги йому потрібно вказати свої дані для отримання і відправити заявку.

3. Адміністратор сайту, тобто волонтер, перевіряє наявність допомоги яку обрав користувач, звіряється з даними та обробляє заявку.

4. Волонтер-водій переглядаючи наявні заяки може самостійно вибрати, які він зможе виконати.

4. Користувач отримує відповідь від адміністратора.

5. Після цього допомога резервується і заявка користувача з’являється в його особистому кабінеті, де він може моніторити статус та місце знаходження допомоги.

6. Волонтер-водій може відразу закривати декілька заявок, після отримання користувачем допомоги, заявка отримує статус «Завершена».

7. Також адміністратори можуть оновлювати та додавати нові дані про допомогу.

1.1.2 Бізнес-вимоги

Для системи з моніторингу та координації гуманітарної допомоги головною метою є забезпечення ефективного функціонування та досягнення бізнес-цілей організації. Ось деякі з найважливіших бізнес-вимог розроблювальної інформаційної системи:

Таблиця 1.1 – Бізнес вимоги.

Автоматизація основних процесів	В системі повинні бути автоматизовані основні процеси замовлення, розподілу, відстеження та доставки гуманітарної допомоги, щоб забезпечити швидке реагування на кризові ситуації.
Інтеграція з суміжними системами	Система повинна мати змогу інтеграції з іншими системами, такими як системамою управління запасами допомоги, системами моніторингу, геопросторовими системами для забезпечення повноцінного функціонування та обміну даними.
Забезпечити безпеку даних	Безпека даних є дуже важливою для гуманітарних організацій, тому в системі повинно бути шифрування даних та контроль доступу, а також засоби забезпечення конфідеційності.
Простий інтерфейс користувача	Інтерфейс користувача повинен бути інтуїтивно зрозумілим та зручним у використанні.
Масштабованість системи	Система повинна мати змогу масштабуватись, оскільки обсяги гуманітарних операцій можуть змінюватися в залежності від ситуації. Система повинна бути здатна обробляти великі обсяги даних, бути надійною та доступною в будь-який час.
Аналітика та звітність системи	Система повинна забезпечити можливість аналізу даних та створення звітів для оцінки ефективності гуманітарних операцій.

1.1.3 Бізнес-ризики

Під час впровадження системи моніторингу та координації гуманітарної допомоги можуть виникнути різноманітні бізнес-ризики, які потрібно врахувати та уникати їх. Ось деякі з них, які можуть вплинути на нашу систему:

- Зловмисники можуть використати систему для незаконного доступу до конфіденційної інформації про гуманітарні операції. Порушення безпеки може призвести до розголошення особистої інформації та втрати довіри з боку стейкхолдерів.
- Відмова серверверу або ПЗ можуть призвести до перебоїв в роботі системи у важливий момент, що може ускладнити надання гуманітарної допомоги та порушити процес координації.
- Помилки при введенні даних можуть призвести до невірних рішень та неправильного розподілу допомоги і це може призвести до зайвих витрат та зниження ефективності надання гуманітарної допомоги.
- Недостатня підготовленість персоналу може призвести до неправильного використання системи та збільшення помилок та недоліків в роботі системи.
- Введення нової системи може створити юридичні ризики, такі як порушення законодавства про захист даних, відповідальність за помилки в розподілі допомоги або порушення прав людини.

1.1.4 Особливості продукту

Web-додаток HACS:

- WA-1: Register
- WA-2: Log In
- WA-3: Main Page
- WA-4: My Profile
- WA-5: Aid Page

- WA-6: Aid History
- WA-7: Manage Users
- WA-8: Aid Management

1.2 Визначення основних бізнес-функцій інформаційної системи

Метою створення інформаційної системи з моніторингу та координації гуманітарної допомоги є автоматизація її бізнес-функцій:

– Аналіз та збір інформації про потреби, які виникають на місцях подій для забезпечення своєчасної допомоги.

– Спрямування та організація ресурсів різних волонтерських груп та гуманітарних організацій для максимальної ефективності впливу на ситуацію.

– Доцільне виконання розподілу необхідних ресурсів на основі нагальних потреб.

– Забезпечення ефективної комунікації між учасниками системи, для високої координації дій.

– Моніторинг переміщення допомоги, її використання та ефективності, а також контроль за виконанням завдань з доправленням допомоги.

Загалом кожен з перерахованих бізнес-функцій входить і включається в процес моніторингу та координації гуманітарної допомоги.

При автоматизації бізнес-процесів можна виділити ряд користувачів, які будуть взаємодіяти з системою, а саме: гість, користувач, волонтер та волонтер-водій.

Гість – умовна роль, котра надається відвідувачу веб-додатку за замовчуванням, доки він не авторизується. Даний тип користувача має переважно оглядові можливості взаємодії з системою: дізнаватись про те як можна оттримати допомогу або як стати волонтером. Він не може оформити отримання допомоги поки не авторизується. Даний функціонал стане доступний лише після авторизації.

Користувач – роль, яка надається відвідувачу після авторизації у системі. Це особа, яка потребує в гуманітарній допомозі. Вона може бути постраждалим від різних катастроф, конфліктів, бути біженцем або безхатьком. Після реєстрації, користувач може обрати допомогу, вказавши свої потреби, свої особисті дані та місце перебування.

Волонтер – роль, яка надається одному з членів організації. Це може бути одна чи група осіб, які добровільно надають свою допомогу та зусилля для надання гуманітарної допомоги. Вони можуть працювати в різних рівнях надання домоги, таких як розподіл допомоги та логістика. Волонтери мають окремий доступ до системи для підветрдження наявності обраної допомоги та комунікації з іншими учасниками в операції з надання допомоги.

Волонтер-водій – роль, надається тій людині яка самостійно виявила бажання стати водієм з доправленням гуманітарної допомоги. Це волонтери, які мають транспортні засоби та готові доправляти допомогу до місць де її потребують. Вони авторизуються у системі для перегляду доступних заявок на допомогу та забезпечують доставку до потрібного місця.

1.3 Визначення функцій інтерфейсу користувальницької частини інформаційної системи

Інтерфейс, що представляє собою доступ до ІС, являє собою веб-додаток. Веб-додаток включає декілька веб-сторінок, контент та доступ до яких залежить від ролі користувача, які задовольняють бізнес-функції користувачів у доступі до інформації та її зміні. Зі сторони користувача, інтерфейс повинен бути інтуїтивно зрозумілим.

Головна сторінка веб-додатку – сторінка яка має простий та зрозумілий інтерфейс, там де можна створити запит на допомогу або стати волонтером чи водієм-волонтером, а також інформація про організацію. Головна сторінка містить також можливість авторизації користувача або реєстрації в системі. Для

того щоб отримати всі можливості в системі, потрібно зараєструватися. Форма реєстрації матиме схожі поля для заповненя, такі як: прізвище, ім’я, по батькові, область в якій людина очікує або буде надавати допомогу, місто, номер телефону, тип документу, серія або номер паспорту, пароль. Але форма матиме відмінність, яка заключатиметься в тому що людина яка потребує допомогу потрібно буде підтвердити свою особу паспортними даними, а водію-волонтеру потрібно додати своє водійське посвідчення, яке затвердить його можливість керувати транспортними засобами.

На другій сторінці, де буде форма з вибором та типом гуманітарної допомоги, користувач зможе обрати самостійно ті товари в яких він потребує та обрати її варіанти доставки.

В особистому кабінеті користувача є змога самостіно створити нове замовлення, де з’явиться форма з категоріями та типом допомоги. Також в особистому кабінеті користувач має змогу переглянути статус та місце знаходження свого замовлення. Для того щоб комунікація між потребуючим та водієм була максимально ефективною, потребуючий може перглянути додаткову інформацію, таку як: хто доставляє йому допомогу, о котрій годині та номер автомобіля та її модель.

На відміну від користувача, який потребує допомогу, особистий кабінет організації має відмінності, в ньому присутні 4 вкладки: відкриті, підверджені, які знаходяться в роботі та виконані заявки. Також організація може переглянути більш детальну інформацію про те кому вони доставляють та що міститься в заявці.

Для водія-волнтера особистий кабінет матиме також 4 вкладки, але відміність буде полягати у відображені завки, буде показано додаткову інформацію про кількість цих речей. Це зроблено для того щоб водій мав змогу розрахувати кількість речей, котру він може помістити в свою автівку.

Одним із головних завдань системи це є надання можливості моніторингу статусу замовлення, що в свою чергу забезпечує його прозорість виконання.

1.4 Організаційна структура

Організаційна структура представляє собою сукупність впорядкованих та взаємопов’язаних елементів, що забезпечує їх чітке функціонування один з одним та успішну роботу системи як одного цілого. Кожен з елементів блок-схеми – це окремі працівники, служби та інші ланки, задіяні в діяльності організації, а відносини між ними підтримуються завдяки зв’язкам.

Елементи схеми пов’язані один з одним за допомогою декількох видів зв’язків. Горизонтальні зв’язки мають характер погодження, а вертикальні – це зв’язки підпорядкування, необхідність в них виникає при ієрархічності управління, тобто за наявності декількох рівнів управління.

Організаційна структура з моніторингу та координації гуманітарної допомоги зображена на рисунку 1.1:

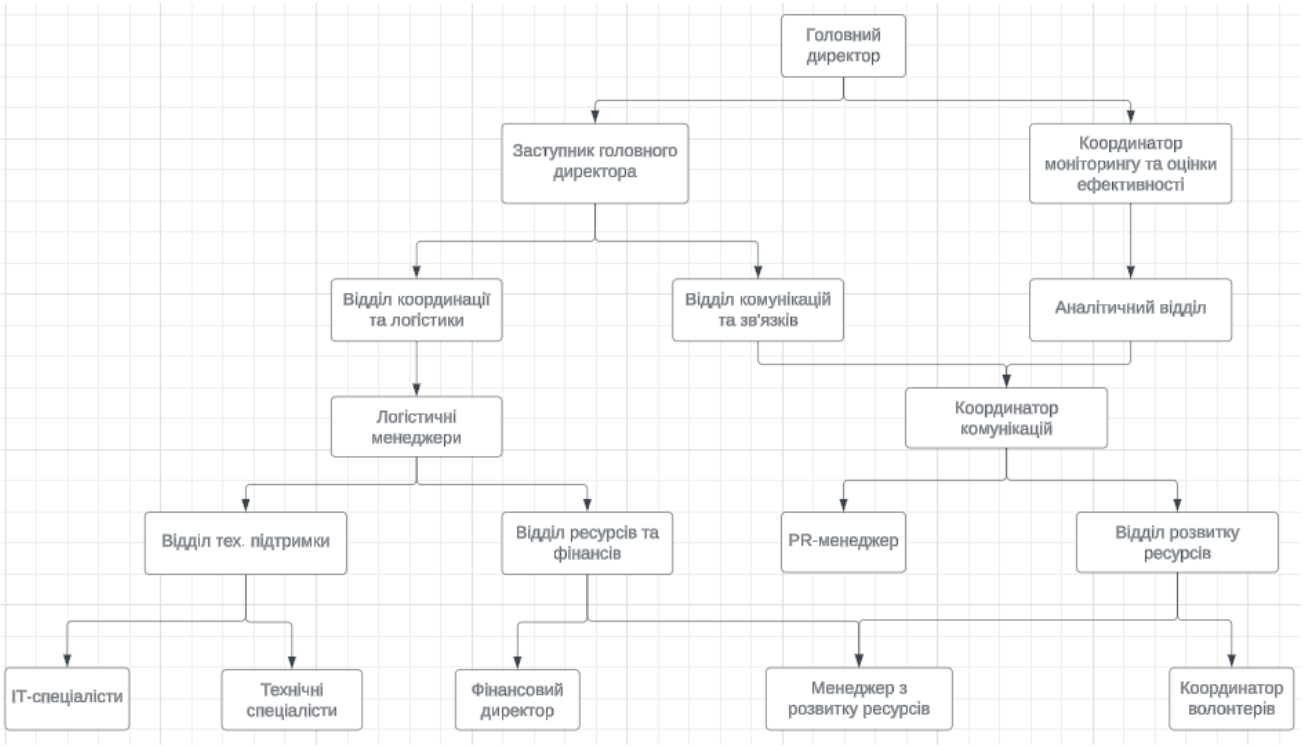


Рис 1.1 – Організаційна структура

Очолює дану структуру головний директор – він відповідає за загальне керівництво та стратегічне планування організації, розробляє ключові стратегічні рішення та встановлює головні напрями розвитку організації.

Заступник головного директора – допомагає головному директору у керівництві організацією, він відповідає за конкретні функціональні групи та проекти.

Відділ моніторингу та оцінки – це група людей, які займаються зібранням, аналізом та оцінкою інформації про гуманітарні потреби та ситуації на місцях подій. Надає перевірені дані та аналітичні звітності для прийняття правильних рішень.

Відділ координації та логістики – організовує та керує ресурсами для забезпечення ефективного виконання операцій, включає в себе траспортування та розподіл гуманітарної допомоги.

Відділ комунікацій та зв’язків – забезпечує ефективну комунікацію з усіма зацікавленими сторонами, включаючи громадськість, урядові структури та інші організації.

Відділ технічної підтримки – розробляє та впроваджує інформаційні системи для моніторингу та координації гуманітарної допомоги. Забезпечує технічну підтримку та обслуговування обладнання та ПЗ.

Відділ ресурсів та фінансів – керує фінансами організації та розробляє бюджети для гуманітарних проектів. Займається збором коштів та ресурсів для підтримки гуманітарних операцій.

Відділ розвитку ресурсів – набирає, організовує та керує волонтерами, які допомагають у виконанні гуманітарних завдань. Здійснює планування та координацію дій волонтерів для ефективної роботи в гуманітарних проектах.

Система організації має відповідати також принципу забезпечення контролю. Обов'язковими елементами тут мають бути внутрішній і зовнішній аудити. Підрозділи внутрішнього і зовнішнього аудиту покликані контролювати законність виконуваних заходів, їхню відповідність нормам та інструкціям.

Внутрішня упорядкованість та узгодженість внутрішніх підрозділів організації забезпечуються за допомогою підпорядкування працівників правилам діяльності. Для реалізації даного принципу організації з моніторингу та координації гуманітарної допомоги розробляють певні розпорядження (статут, положення про відділи і служби, кваліфікаційні характеристики, посадові інструкції). Кожен співробітник повинен знати свої обов'язки, мати певні знання й уміння, виконувати правила, наведені в цих та інших документах.

Забезпечення організації оперативною і достатньою інформацією, що необхідна для того, щоб вчасно приймати необхідні організаційні й економічні заходи, в організаційному відношенні реалізується шляхом створення в організації особливих підрозділів (групи людей), що займаються збором і обробкою відповідної інформації.

Слід зазначити, що організаційна структура забезпечує координацію всіх функцій управління, визначає права і обов’язки на управлінських рівнях. Саме від правильності її побудови залежить ефективність діяльності організації, її виживання і процвітання. Структура для конкретної фірми визначає організаційну поведінку її співробітників, тобто стиль управління, якість та ефективність праці колективу.

1.5 Аналіз існуючих програмних забезпечень

Перед початком написання цієї роботи, було проведено дослідження та огляд існуючих систем гуманітарної допомоги в Україні. Цей аналіз став особливо важливим у зв'язку з тим, що країна зазнає важкої гуманітарної кризи внаслідок війни. Відразу ж стало очевидним, що існуючі рішення не завжди відповідають потребам користувачів, а особливо актуальним стало питання про швидкий та зручний доступ до інформації про допомогу.

Програмне забезпечення для системи з моніторингу та координації гуманітарної допомоги – це рішення для управління системою, яке призначене для автоматизації процесів збору, аналізу, координації та моніторингу

гуманітарної допомоги. Завдяки ПЗ відбувається спрощення процесу взаємодії між різними сторонами та полегшення доступу до допомоги для тих хто її потребує.

Далі детально про існуючі програмні засоби.

1.5.1 Платформа «єДопомога (Волонтерська)»

Платформа «єДопомога (Волонтерська)» – розроблений Міністерством соціальної політики України за підтримки Міністерства цифрової трансформації України та Програми розвитку ООН в Україні за фінансової підтримки Швеції з метою забезпечення нагальних потреб громадян, які постраждали внаслідок війни в країні [1] (рис. 1.2).

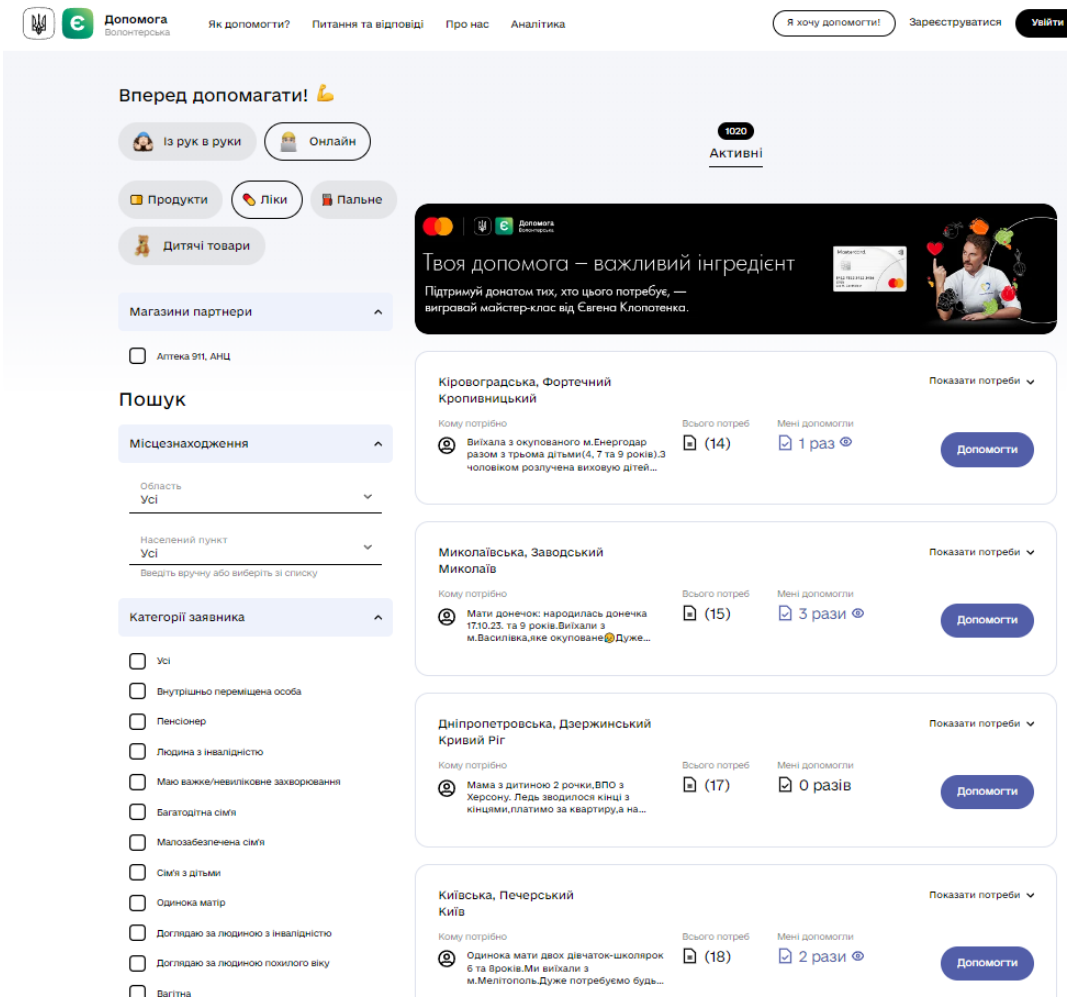


Рис 1.2 – Платформа «єДопомога (Волонтерська)»

Ця платформа також взаємодіє з Дією, що є однією з головних переваг цього веб-сайту. Користувачам доступно як подання, так і отримання фінансової та волонтерської допомоги. На сайті користувачі можуть створювати запити на отримання конкретної допомоги, такої як ліки, одяг, продукти харчування, а волонтери можуть обирати, кому з цих заявок вони бажають передати необхідні ресурси особисто. У системі також передбачена можливість надання фінансової допомоги у випадках, коли немає можливості доставити допомогу особисто. Ця платформа має ряд партнерів, які за кошти, виділені волонтерами, надають можливість використовувати їх як "подарункові сертифікати". Іншими словами, особа може придбати продукти на певну суму у партнерському магазині (рис. 1.3).

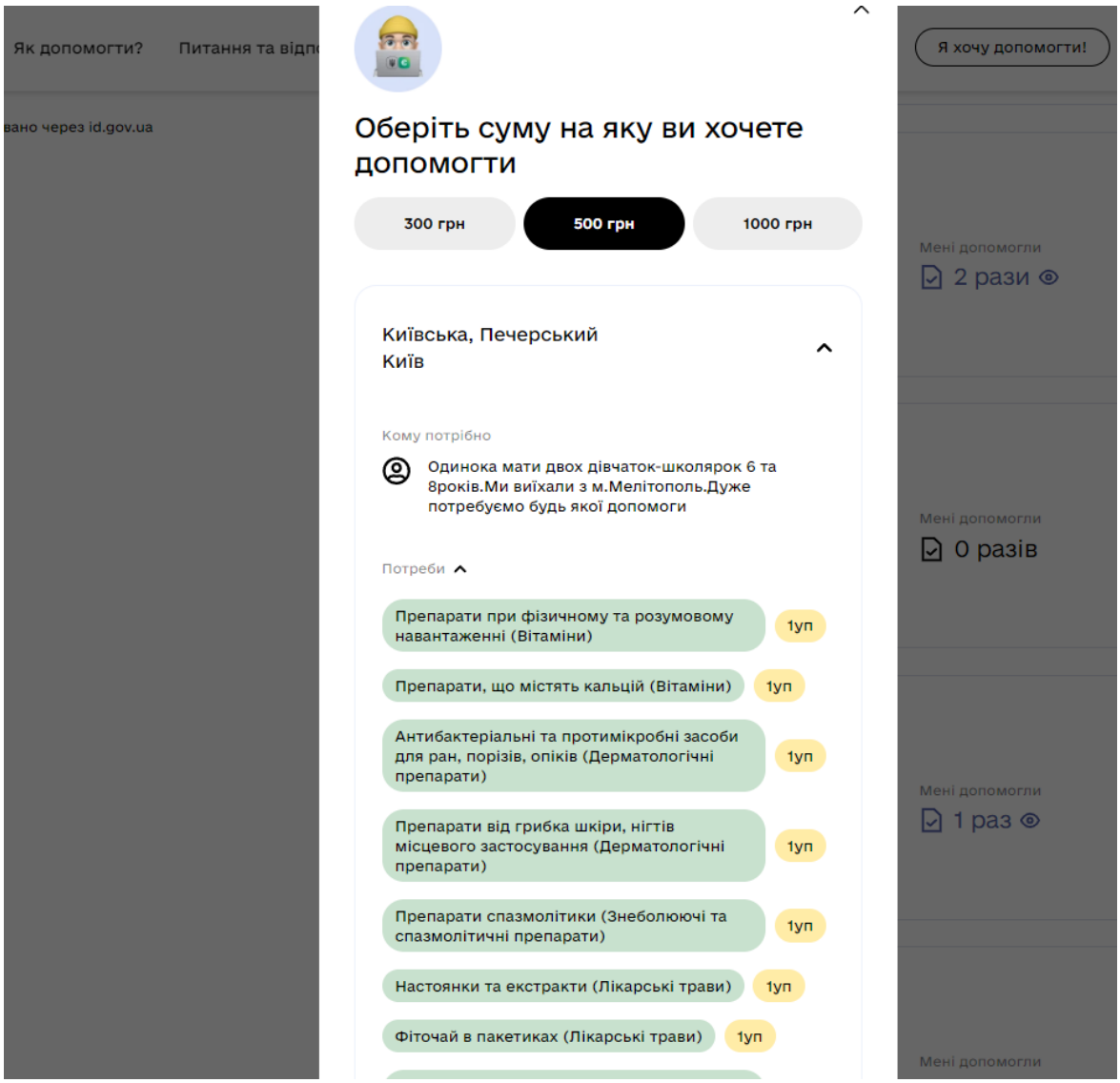


Рис 1.3 – Грошова допомога на сертифікат магазинів партнерів

Основні переваги цієї платформи:

– Взаємодія з застосунком «Дія».

– Присутня база партнерів із мережі продуктових товарів та аптек.

– Грошова допомога в онлайн. Волонтер може допомогти, тобто задонативши не виходячи з дому.

– Зручний та зрозумілий інтерфейс.

У даного сайту присутня схожість до розроблювальної системи даної роботи. Але присутні деякі відмінності, такі як:

– Неавтоматизована система для масових випадків. Волонтеру потрібно власноруч налаштовувати фільтр по своїм критеріям, кому і як саме надати ту чи іншу допомогу.

– Волонтер чи організації можуть відправляти запитувані ресурси тільки особисто, з рук в руки.

1.5.2 Благодійний фонд «Благороб»

Благодійний фонд «Благороб» – це благодійний фонд, який має на меті забезпечити допомогу тим, хто опинився в складних життєвих обставинах у Харкові або Харківській області. Наша місія - підтримувати та допомагати тим, хто потребує допомоги, надаючи необхідні ресурси, які можуть включати фінансову допомогу, послуги, необхідні предмети для життя, медичну або соціальну підтримку та інше. Ми співпрацюємо з волонтерами, партнерами та спонсорами для забезпечення максимального покриття потреб наших бенефіціарів. Наш фонд прагне створити світ, де кожна людина має можливість налагоджувати стабільне та щасливе життя, незалежно від її обставин [2] (рис. 1.4.).

Рис 1.4 – Благодійний фонд «Благороб»

Також на сайті присутня можливість для підтримки на рахунок фонду, що допоможе підвищити фінансову можливість волонтерів цієї організації (рис. 1.5).

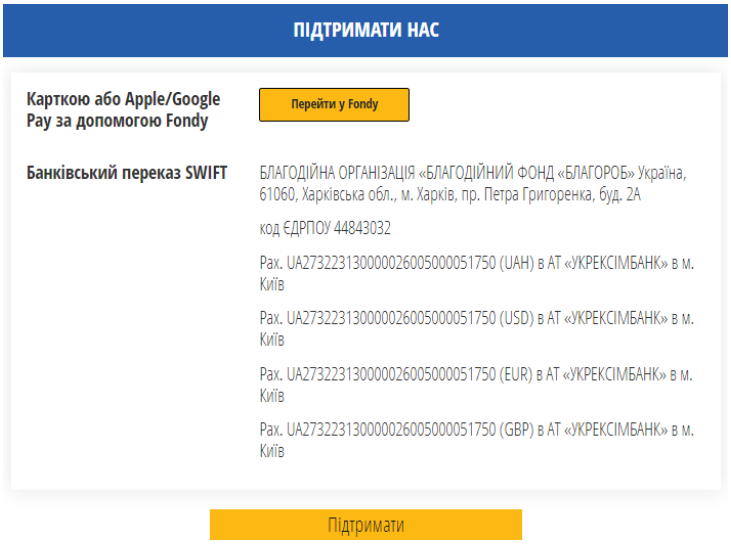


Рис 1.5 – Благодійний внесок на рахунок фонду

Проаналізувавши даний сайт фонду, можна виділити окремі співпадання з розроблювальною системою, а саме:

– Допомога розділена за типами її надання.

– Доступна можливість не тільки отпримати допомогу, а ще запропонувати.

Недоліками данної системи:

– Система не може поширюватись на другі регіони України, отримати допомогу мають можливість тільки жителі Харкова та області.

– В системі неможливо відслідкувати статус свого замовлення, комунікація відбувається виключно через пошту.

1.5.3 Платформа «ZayavaInfo»

На платформі «ZayavaInfo», розмістила свою допомогу громадська організація «Їжа Харків'янам», яка надає адресну гуманітарну допомогу продуктами харчування, ліками, питною водою, засобами гігієни, памперсами, дитячим харчуванням тощо (рис. 1.6.).

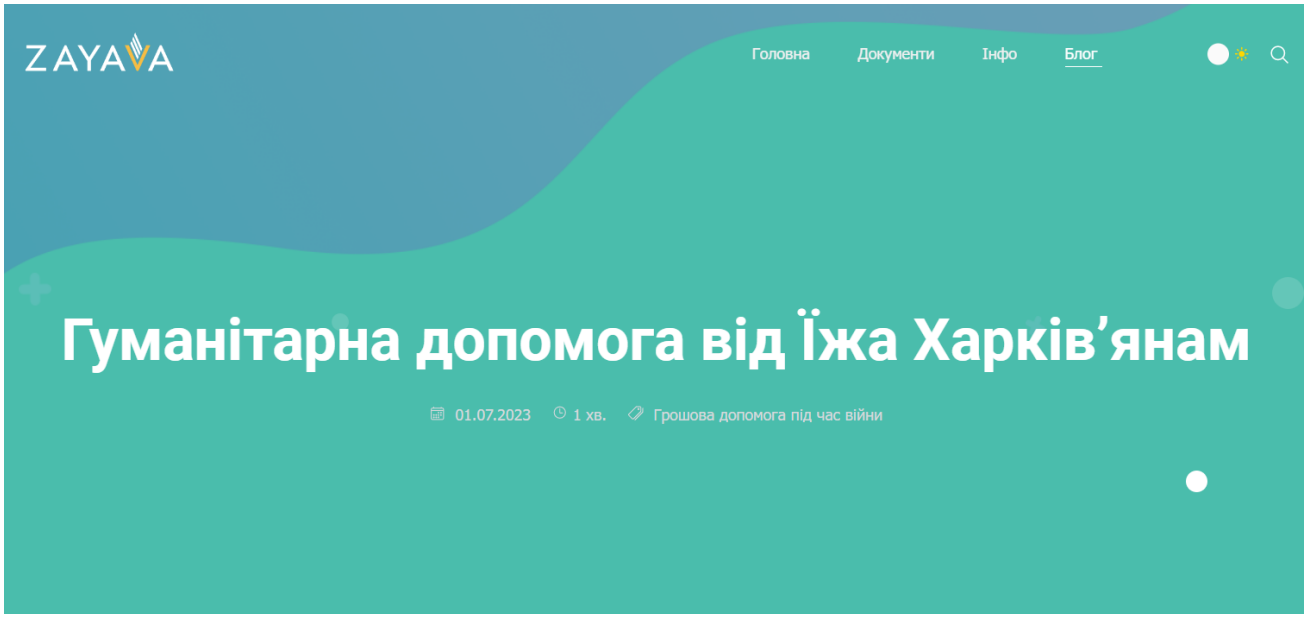


Рис 1.6 – Громадська організація «Їжа Харків'янам»

Проаналізувавши даний сайт організації, можна виділити співпадання з розроблювальною системою, а саме те що допомога розділена за типами і людина яка потребує допомоги може самостійно обрати, що їй потрібно.

Недоліками данної системи:

– Отримати допомогу від організації можуть громадяни, які знаходяться у м. Харків та які підпадають під певні категорії соціально незахищених верств.

– В системі неможливо відслідкувати статус свого замовлення, комунікація відбувається виключно через телефонну розмову;

– Необхідно погодитися на фото- або відеозйомку під час отримання допомоги та дозволити розміщувати такі матеріали на офіційних сторінках організації або у соціальних мережах.

– Отримати допомогу можна лише один раз.

1.6 Переваги програмного забезпечення

Зважаючи на складність та масштаби гуманітарних операцій, ПЗ для системи моніторингу та координації гуманітарної допомоги надає численні переваги:

– Підвищення ефективності використання ресурсів. ПЗ дозволяє ефективно використовува та координувати гуманітарну допомогу, такі як медичні засоби, продукти харчування, речі і тд. Це допомагає забезпечувати необхідну допомогу вчасно та ефективно.

– Підвищення продуктивності. ПЗ автоматизує багато рутинних операцій, що дозволяє працівникам організації сконцентруватися на більш важливих завданнях, таких як аналіз потреб на місцях подій та прийняття стратегічних рішень.

– Покращення зв’язку та обміну інформацією. Програмне забезпечення надає можливість співпрацювати з різними організаціями, що допомагає

покращити координацію гуманітарних заходів та спільно вирішувати важливі проблеми.

– Зменшення часу реакції на події. Завдяки ПЗ можна швидко реагувати на гуманітарні кризи та події, оскільки воно дозволяє отримувати, аналізувати та реагувати на інформацію в реальному часі.

– Забезпечення безпеки даних. ПЗ може забезпечити захист конфіденційної інформації та особистих даних, що є важливим аспектом в гуманітарних операціях, де часто обробляється конфідеційна інформація про постраждалих.

– Підвищення іміджу та довіри до організації. Використання сучасного ПЗ для координації гуманітарних заходів свідчить про професіоналізм та високий рівень організації, що може підвищити її імідж та довіру громадськості.

1.7 Постановка мети і задач кваліфікаційної роботи

На даний час існує багато програмних рішень для координації гуманітарної допомоги, але вони не завжди задовольняють потребам користувачів. Багато систем мають досить не зрозумілий та складний інтерфейс для користувача. Все що потрібно користувачу – це мати змогу швидко та зручно оформити допомогу і отримати її якомога скоріше. Зі сторони керівництва – це управління робочими заявками та перегляд необхідної статистики, також додавання та оновлення допомоги.

Актуальність роботи випливає з того, що незважаючи на існуючі методи управління з моніторингу та координації допомоги, вони потребують певної автоматизації та удосконалення.

Основною метою кваліфікаційної роботи є покращення ефективності процесів оформленя, моніторингу та координації гуманітарної допомоги шляхом розробки компонентів системи автоматизації цих процесів.

Треба розробити інформаційну систему з моніторингу та координації гуманітарної допомоги, яка допоможе в автоматизації надання допомоги.

Це повинно пройти в декілька етапів, а саме:

– Визначення вимог до ІС: визначення функціональних та нефункціональних вимог, таких як можливість реєстрації заявок на допомогу, відстеження статусу, координація волонтерів.

– Проектування системи: розробити архітектуру системи, включаючи БД, логіку самої програми та інтерфейс користувача. Також потрібно визначити інфраструктуру для зберігання та обробки даних.

– Розробка ПЗ: потрібно розробити систему відповідно до визначених вимог та проектування.

– Тестування ПЗ: провести тестування ПЗ для перевірки його на коректну працездатність та відповідності вимогам.

– Впровадження та підтримка системи: після завершення тестування систему впровадити в роботу та підтримувати в процесі експлуатації.

– Аналіз та удосконалення системи: Після впровадження системи в роботу, потрібно провести аналіз її роботи та можливості її удосконалити.

Така система значно допоможе автоматизувати процеси оформлення допомоги, координації та моніторингу, що значно покращить ефективність та швидкість реагування на гуманітарні кризи.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ ПОБУДОВИ СИСТЕМ

2.1 Загальні вимоги до розроблювальної автоматизованої системи

Автоматизація процесів моніторингу та координації гуманітарної допомоги стає все більш актуальною в умовах сьогодення. Використання автоматизованих програмних систем, які можуть в реальному часі обробляти великі обсяги даних, стає ключовим фактором для підвищення ефективності логістики допомоги. Такі системи мають складну архітектуру, що дозволяє їм не лише адаптуватися до поточних потреб, а ще прогнозувати можливі зміни в процесах, оптимізуючи їх до встановлених критеріїв.

В таких системах наявні високопродуктивні обчислювальні ресурси, які містять одне або декілька програмних рішень для досягнення ефективності. Це в свою чергу створює надійну платформу, яка здатна витримати високий рівень навантаження і забезпечити роботу системи в будь-яких ситуаціях. Така платформа необхідна для задоволення потреб користувачів у неперервному доступі до інформації та допомоги. Ключовими аспектами є також забезпечення високої пропускної спроможності та безпеки даних в мережі Інтернет.

Забезпечення доступу користувачів до системи через Інтернет, в свою чергу дозволяє значно розширити її функціональні можливості. Користувачі можуть взаємодіяти з системою в будь-який час та будь-де, з використанням будь-яких пристроїв. Це не тільки збільшує зручність та продуктивність користування, але й відкриває нові можливості для організації, дозволяючи організації оперативно реагувати на зміни у потребах населення та покращувати якість надання допомоги.

Завдяки цьому важливою стає не лише технічна реалізація системи, але й її інтеграція з іншими бізнес-процесами організації, включаючи управління відносинами з клієнтами та планування допомоги постраждалим. Інтеграція

дозволяє забезпечити цілісний підхід до управління просцесом отримання допомоги, від моменту оформлення заявки на отримання допомоги до її виконання та аналізу ефективності надання допомоги.

Сучасні автоматизовані системи з моніторингу та координації гуманітарної допомоги включають передові рішення у сфері кібербезпеки. З цього виходить, що на кожному етапі обробки та отримання інформації відбувається її захист від несанкціонованого доступу, забезпечуючи конфіденційність і цілісність даних. Користувачі в свою чергу стають активними учасниками процесу, маючи можливість не тільки отримати інформацію але ще й вносити зміни та налаштувати параметри в системі під свої потреби.

Одним з основних викликів при впровадженні та експлуатації таких систем є їх масштабованість та гнучкість. Бізнес-середовище постійно змінюється, тому системи мають бути спроектовані таким чином, щоб легко адаптуватися до нових умов, збільшення обсягів даних або змін у процесах управління ланцюгами поставок. Така адаптивність забезпечується за рахунок модульної архітектури та використання хмарних технологій, що дозволяє легко масштабувати ресурси в залежності від поточних потреб.

Особливу увагу потрібно буде приділити розвитку інструментів аналітики та інтелектуального аналізу даних. Завдяки збору та обробці великих обсягів інформації, організація краще зрозуміє потреби користувачів та ефективніше реагувати на кризові ситуації. Це дозволить підвищити рівень задоволеності користувачів і зміцнити конкурентоспроможність на ринку.

Успіх у розробці та впровадженні інформаційних систем для моніторингу та координації гуманітарної допомоги залежитиме не тільки від технологічних інновацій, а ще й від глибокого розуміння соціальних аспектів, що впливають на розподіл і використання допомоги у кризових ситуаціях. Завдяки врахування таких аспектів, разом із технологічним прогресом, це дозволить створити ефективні та стійкі системи, які в свою чергу здатні відповідати на виклики сьогодення.

2.2 Аналіз проблемних сторін прототипу та шляхи їх вирішення

Розробка ПЗ включає в себе застосування різноманітних методологій, кожна з яких має певні особливості та призначена для конкретного типу проекту. Вибір підходу для розробки потребує аналізу та розуміння ключових вимог проекту, таких як цілі проекту, специфікаця, а ще орієнтація на потреби користувачів.

Першим кроком є ідентифікація основних завдань проекту, які повинна вирішувати система та визначення основних характеристик системи. Це включає в себе не лише технічні аспекти, а ще й оцінку користувачів, для того щоб забезпечити зручність і інтуїтивність роботи з системою. Ще важливим аспектом є те, як система буде представляти дані та інформацію для користувачів, забезпечуючи їх доступність та зрозумілість.

Розробка прототипу системи потребує врахування цілого ряду факторів, таких яких:

– Ясне визначення, які саме задачі має вирішувати система і який функціонал буде необхідний для її функціонування.

– Детальний опис технічних вимог та специфікацій, що враховує потреби користувачів та організації.

– Розробка користувальницького інтерфейсу, який був би не тільки функціональним, але й зручним для кінцевого користувача, включаючи в себе зрозуміле відображення інформації, яку потребує користувач.

Подолання проблем, які пов'язані з розробкою прототипу, передбачає в собі активну взаємодію зі стейкхолдерами проекту, збір інформації від користувачів системи та гнучке впровадження змін у процесі розробки системи. В свою чергу це дозволить своєчасно коригувати проектні рішення, адаптувати систему до вимог ринку та забезпечити її високу якість.

Нижче наведено детальний опис про кожен підхід.

2.2.1 Серверна частина програми

Серверна частина програми грає ключову роль у забезпеченні безперебійної та ефективної роботи системи. Вона відповідає за обробку запитів від клієнтської частини, управління БД, забезпечення безпеки даних та інтеграцію з іншими сервісами та системами. При реалізації ПЗ є два архітектурних підходи розробки серверної частини програмного застосунку: мікросервіси та моноліт.

Мікросервісна архітектура (рис. 2.1.) полягає у розбитті ПЗ на невеликі та незалежні між собою модулі, які виконують конкретні бізнес-функції та комунікують між собою через добре визначений набір правил та протоколів. Кожен мікросервіс працює як незалежний сервіс, який має свою власну БД та може бути розгорнутий незалежно від інших частин системи.

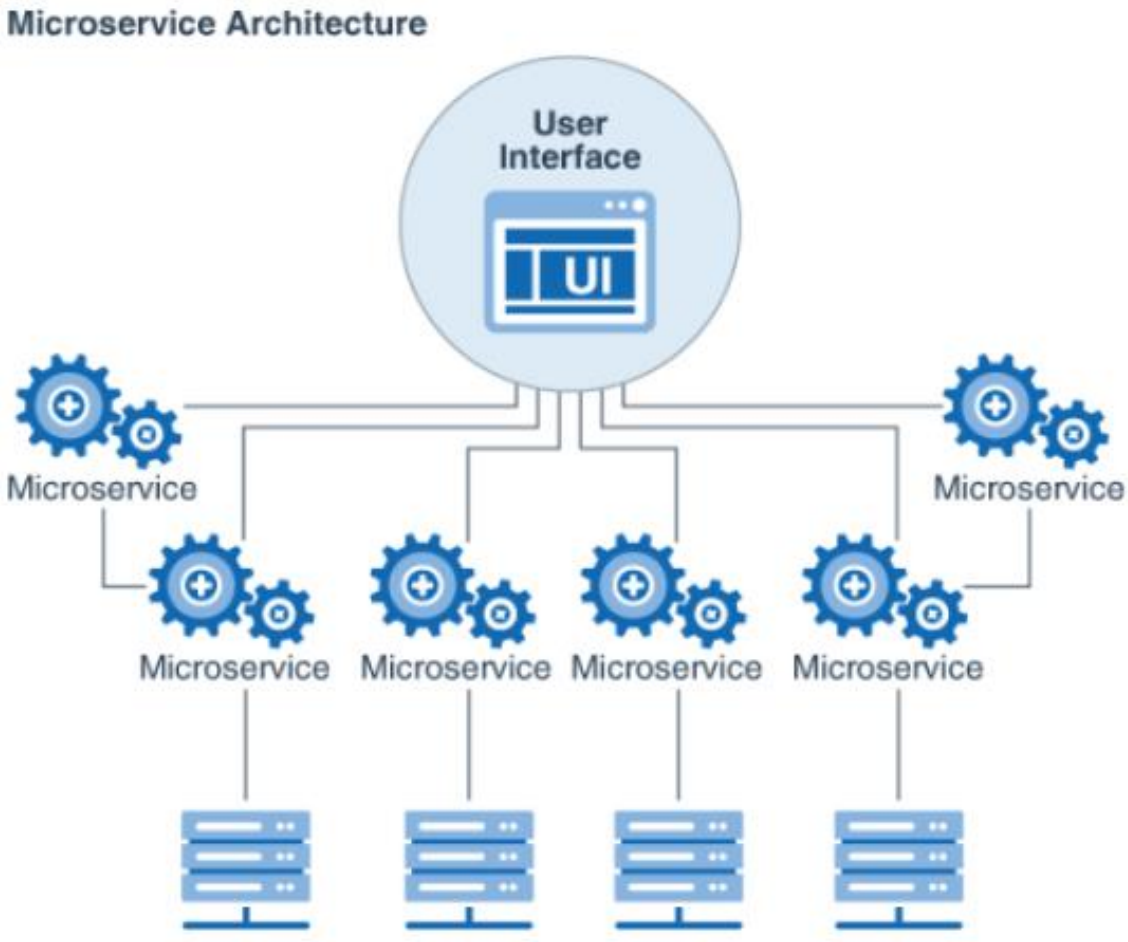


Рис 2.1 – Мікросервісна архітектура

Переваги мікросервісної архітектури:

– Нові функції можна додавати або видаляти, розробляючи нові мікросервіси або модифікуючи вже існуючі без впливу на саму системи.

– Є можливість масштабувати окремі компоненти системи залежно від нагальних потреб, без необхідності масштабування всієї системи.

– Помилка в будь-якому мікросервісі не веде до зупинки всієї системи.

– Можна використовувати різні технології та мови програмування для різних мікросервісів залежно від вимог та завдань системи.

Недоліки мікросервісної архітектури:

– Зростання кількості мікросервісів веде до складнощів у їх координації, спостереження та забезпечення їх безпеки.

– Якщо взаємодія між мікросервісами відбувається через мережу, то можуть збільшитись затримки порівняно з викликами в межах одного процесу.

– У разі використання окремих БД для кожного мікросервісу, то виникає потреба в синхронізації даних між ними.

Метою цієї архітектури є створення слабо пов’язаних служб і комунікація відіграє головну роль у досягненні цього. В архітектурі мікросервісів є чотири способи, завдяки яким служби можуть комунікувати між собою, кожен із способів має свої переваги та недоліки:

– Синхронний, комунікація відбувається в режимі реального часу.

– Асинхронний, комунікація відбувається незалежно від часу.

– Гібрид, який підтримує обидва способи.

– Подійно-орієнтована архітектура взаємодія.

Детальний опис того, як працює кожен із цих варіантів:

У синхронному способі комунікації один мікросервіс відправляє запит до іншого мікросервісу і очікує на відповідь перед продовженням своєї роботи. З цього виходить, що процеси відбуваються в режимі реального часу. Перевагами синхронної комунікації є легкість розуміння та реалізації, а ще тут присутня тісна інтеграція з іншими сервісами. А недоліками є залежність від доступності та

продуктивності інших сервісів та високий ризик затримок, тому що один сервіс чекає відповіді від іншого.

Асинхронний спосіб комунікації дозволяє сервісам відправляти повідомлення без очікування зворотної відповіді. Сервіс може продовжувати свою роботу, поки інший сервіс обробляє запит. Перевагами є зниження залежності між сервісами та можливість ефективної обробки запитів за рахунок буферизації повідомлень. Недоліками є складність стеження та діагностики, тому що взаємодя не є лінійною.

Гібридний підхід поєднує в собі елементи синхронної та асинхронної комунікації, дозволяючи системі використовувати переваги обох методів залежно від контексту. Однозначною перевагою є гнучкість у виборі підходу комунікації та збалансоване співвідношення. Недоліками є складність архітектури та управління, тому що потрібно забезпечити ефективну інтеграцію різних методів комунікації.

У подійно-орієнтованій архітектурі мікросервіси реагують на події, які генеруються іншими сервісами. Такий підхід дозволяє сервісам залишатися слабо зв'язаними, оскільки вони не викликають один одного безпосередньо, а лише прослуховують та обробляють події. Перевагою є висока гнучкість, оскільки сервіси можуть додаватися, оновлюватися або видалятися без зупинки роботи інших компонентів системи. Недоліком є те що виникає потреба в надійному механізмі керування подіями та забезпеченні їх доставки.

Протилежною до мікросервісної архітектури є монолітна архітектура. Монолітна архітектура (рис. 2.2.) передбачає розробку ПЗ як єдиного цілого, де всі компоненти системи, такі як: інтерфейс користувача, бізнес-логіка, доступ до даних, тісно пов’язані та працюють у межах одного процесу або проекту.

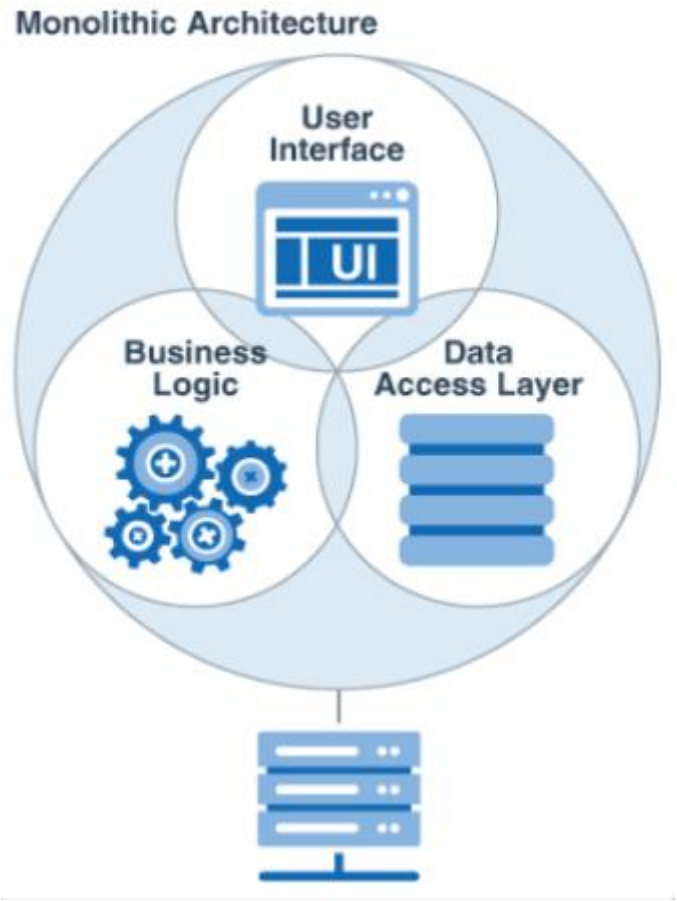


Рис 2.2 – Монолітна архітектура

Переваги монолітної архітектури:

– Завдяки одному коду та меншій кількості залежностей, розробка та розгортання монолітного застосунку є простішим та більш швидким.

– Керування одним застосунком та його конфігурацією є менш складним, ніж координація між кількома мікросервісами, як у мікросервісній архітектурі.

– У межах одного процесу внутрішні виклики функцій є швидшими, ніж мережеві виклики між мікросервісами, що призводить до кращої загальної продуктивності проекту.

– Забезпечення цілісності в монолітній архітектурі є значно простішим, оскільки всі операції з даними виконуються в рамках однієї БД.

Недоліки монолітної архітектури:

– Масштабування монолітного проекту часто вимагає масштабування цілого застосунку.

– Внесення змін до однієї частини може вплинути на інші частини проекту, що робить процес оновлення більш ризикованим.

– Усі компоненти монолітної архітектури зазвичай використовують одну технологічну стеку, що обмежує можливості вибору технологій та інструментів.

– По мірі зростання розміру монолітного застосунку, його кодова база стане дуже складною для підтримки.

Монолітна архітектура характеризується своєю єдиною та неділимою структурою, вона може бути розроблена за допомогою різних підходів, кожен з яких має свої особливості та відмінності, свої переваги та обмеження. Далі наведенний типи реалізації монолітної архітектури:

1. Традиційний моноліт – це моноліт архітектури, де бізнес-логіка, інтерфейс користувача та доступ до даних інтегровані в один застосунок. В свою чергу це дозволяє легко розробити, тестувати та розгорнути застосунок як одне ціле. Перевагами традиційного моноліту є простота розробки та тестування завдяки єдиному середовищу виконання. А недоліками є складність масштабування окремих компонентів застосунку та високий ризик впливу змін у одній частині застосунку на інші частини.

2. Багаторівнева архітектура – це архітектура, яка передбачає розділення програмного застосунку на шарів, де кожен шар відповідає за певний аспект функціональності. Хоча шари тісно інтегровані, таке розділення сприяє кращій організації програми. Перевагами багаторівневої архітектури є покращена організація коду програми та спрощення масштабування та оновлення окремих шарів застосунку. Недоліками багаторівневої архітектури є висока залежність між шарами, що в свою чергу ускладнює розробку та впровадження змін в застосунку.

3. Модульна монолітна архітектура – така архітектура включає в себе розбиття застосунку на незалежні між собою модулі, кожен з яких в свою чергу може бути розроблений, тестований та розгорнутий окремо один від одного.

Водночас усі модулі збираються разом і працюють як єдиний застосунок. Перевагами є можливість паралельної роботи над різними модулями командою розробників та вища стійкість до відмов, оскільки помилка в одному модулі менш ймовірно призведе до збою всього застосунку. Недоліками такої архітектури є потреба в додаткових зусиллях для координації залежностей між модулями та їх інтеграції, а ще можуть виникнути складнощі у розгортанні, якщо не було належно враховано залежності між модулями.

4. Сервіс-орієнтована монолітна архітектура – такий підхід все ще вважається монолітним, але він включає в себе розробку внутрішніх сервісів, які в свою чергу взаємодіють один з одним через внутрішній набір правил. Це схоже на мікросервісну архітектуру, але відмінність заключається в тому що все виконується в межах одного процесу. Перевагами такої архітектури є покращене розподілення відповідальності та модульності всередині застосунку, що спрощує розуміння та підтримку коду, та вища ефективність використання ресурсів порівняно з розподіленими мікросервісами, оскільки всі компоненти працюють в одному процесі. Недоліками є обмеження у масштабуванні окремих компонентів або сервісів застосунку, оскільки вони все ще є частиною одного застосунку. Також виникає потреба в розробці та підтримці внутрішніх механізмів комунікації та координації між сервісами.

Вибір підходу до реалізації монолітної архітектури залежить від багатьох факторів, таких як розмір та складність проекту, вимоги до швидкості розробки та розгортання, а також здатність команди управляти складною системою. Хоча монолітна архітектура часто має критику в свою сторону за неможливість легкого масштабувати та складнощі внесення змін в застосунок. Монролітна архітектура може бути дуже ефективною при вірному підході до розробки та використанні сучасних практик програмування, які підтримують чітке відділення компонентів і модульність.

При розробці системи потрібно приділити особливу увагу вибору місця розміщення його на цифрових ресурсів. Серед найбільш вживаних варіантів розміщення проектів є розміщення на хмарній інфраструктурі та на внутрішньому

сервері. Хоча обидва варіанти надають можливості для зберігання даних, обробки запитів та забезпечення безпеки даних, важливо зрозуміти унікальні переваги та обмеження кожного з них.

Хмарний сервер – це технологія обчислень та зберігання даних, яка є надійною та безпечною для бізнесу. Хмарний сервер розроблений для забезпечення легкого та доступного доступу до програм і ресурсів без необхідності внутрішньої інфраструктури чи обладнання [3].

Переваги хмарних серверів:

– В таких серверах легко адаптується під змінні потреби проекту, дозволяючи швидко збільшувати або зменшувати ресурси.

– Оплата відбувається лише за використані ресурси.

– Високий рівень доступності та автоматичне резервне копіювання даних в проекті.

Недоліки хмарних серверів:

– Якщо на стороні хмарного провайдера вимкнеться Інтернет, користувач не матимете доступу до інформації.

– Дуже мало контролю над фізичною інфраструктурою та потенційні питання з конфіденційністю даних.

– Витрати будуть зростати при збільшенні навантаження на систему.

Внутрішній сервер зазвичай є апаратним забезпеченням, створеним самостійно, влаштованим у певній конфігурації та призначеним для досягнення певної мети в бізнесі. Розміщення проекту на власних серверах передбачає купівлю та обслуговування обладнання власними силами або за допомогою найнятих фахівців, які працюють в цій сфері. Все це знаходиться під безпосереднім контролем організації.

Переваги внутрішніх серверів:

– У внутрішніх серверів є повне управління обладнанням та програмним забезпеченням.

– В таких серверах присутня можливість застосування спеціалізованих рішень для забезпечення безпеки даних.

– Доступ до системи та даних відбувається незалежно від зовнішніх факторів.

Недоліки внутрішніх серверів:

– У внутрішніх серверах виникає необхідність самостійно забезпечувати резервне копіювання та відновлення даних у разі перебоїв.

– Внутрішні сервери потребують регулярного обслуговування та оновлення власного обладнання та ПЗ.

– Фізичне розширення ресурсів може бути обмеженим або вимагати додаткових інвестицій від організації.

– Виникають значні інвестиції у придбання та налаштування обладнання для роботи.

Вибір між хмарними серверами та внутрішніми серверами залежить від таких факторів: бюджет організації, потреби у безпеці даних, гнучкість у масштабуванні та ресурси для управління IT-інфраструктурою.

Хмарні сервера ідеально підходять для стартапів та проектів зі змінним навантаженням, де швидкість розгортання та гнучкість є критично важливими. В той час, організації з високими вимогами до контролю над даними та безпекою даних можуть віддавати перевагу внутрішнім серверам, незважаючи на більш високі витрати та необхідність забезпечення надійного управління.

Потрібно ретельно оцінити потреби проекту та довгострокову стратегію розвитку перед прийняттям рішення. Кожен варіант має свої переваги та виклики, і ідеальний вибір зазвичай знаходиться у балансі між цими факторами.

Для програмних застосунків робоча інформація зберігається у БД. Для програмної взаємодії з БД необхідно визначитися з типом системи керування БД. Існують реляційні і нереляційні СКБД. Нереляційні СКБД поділяються на підтипи такі як: документо-орієнтовані, графові, сховище ключ-значення.

Реляційні БД організовують інформацію у форматі таблиць з рядками та колонками, де зв'язки між різними таблицями створюються через первинні ключі. Такий підхід дозволяє ефективно здійснювати складний пошук і аналіз даних.

Переваги реляційних БД:

– В таких БД наявна структурованість і чітка організація даних.

– У реляційних БД використовується мова SQL для запитів, що дозволяє легко працювати з даними.

– Висока надійність і безпека даних.

– Підтримка транзакцій, що забезпечує цілісність даних.

Недоліки реляційних БД:

– Виникає складність у впровадженні змін через складну схему даних.

– Виникає обмежена масштабованість при роботі з великими обсягами даних.

– Присутні великі витрати на ліцензування і обслуговування ПЗ.

Нереляційні бази даних (NoSQL) – є менш структуровані та забезпечують більшу гнучкість та адаптивність [4]. Нереляційні БД пропонують гнучке зберігання даних, яке не обмежується схемою таблиць, вони можуть бути документо-орієнтованими, у вигляді графіків, або організованими як сховища ключ-значення, що забезпечує оптимальні рішення для різних типів даних і завдань.

Переваги нереляційних БД:

– В таких БД присутня здатність зберігати великі обсяги даних без необхідності визначати їх структуру.

– Висока масштабованість і гнучкість в умовах роботи з великими наборами невпорядкованих даних.

– Досить велика швидкість обробки, завдяки оптимізованим механізмам зберігання.

Недоліки нереляційних БД:

– Відсутня наявність стандартизованої мови запитів подібно до SQL.

– Під час роботи виникають потенційні труднощі в забезпеченні узгодженості даних на високому рівні масштабованості через відмову від жорстких схем.

При виборі між реляційними та нереляційними системами керування БД, потрібно враховувати специфіку проекту, його масштаб, вимоги до обробки та зберігання даних. Реляційні СКБД ідеально підходять для таких ситуацій, коли необхідна висока цілісність даних та чітка структура даних, а також для випадків, коли заплановано активне використання складних запитів та аналізу даних. А нереляційні СКБД в свою чергу краще адаптовані для проектів, що вимагають великої гнучкості, швидкої масштабованості та ефективної роботи з великими обсягами неструктурованих або слабо структурованих даних. Тому можна зробити певний висновок щодо використання тої чи іншої системи керування БД. Чим більший набір даних, тим більша ймовірність, що нереляційна база даних краще підійде. Нереляційні бази даних можуть зберігати необмежену кількість даних будь-якого типу і мають гнучкість для зміни типу даних.

2.2.2 Клієнтська частина програмного застосунку

Клієнтська частина програми – це інтерфейс, через який користувачі взаємодіють із програмним застосунком, його прийнято ще називати фронтендом. Розробка фронтенду вимагає врахування візуального дизайну та функціональності системи, щоб забезпечити її зручність, інтуїтивність використання клієнтом та швидкість роботи. Існують такі підходи та технології для реалізації клієнтської частини програми:

1. Традиційний веб-сайт – це веб-сайт, який складається з набору статичних сторінок, що відображаються в браузері. Вони розробляються за допомогою HTML, CSS та JavaScript для додавання інтерактивності. Контент на таких сайтах не змінюється динамічно в залежності від дій користувача.

Переваги:

– Простота розробки та розгортання таких застосунків.

– Висока швидкість завантаження через мінімальний обсяг динамічного контенту.

– Добре індексується пошуковими системами.

Недоліки:

– Вони обмежені можливостями інтерактивності та персоналізації даних.

– Ними тяжко управляти великими сайтами через статичну природу контенту.

2. Рендерінг на стороні серверу (Server-Side Rendering) – веб-сторінка генерується на сервері кожен раз, коли користувач використовує. Сервер обробляє запит, виконує необхідні операції з БД і в кінці генерує кінцевий HTML і відправляє його клієнту.

Переваги:

– Початкова швидкість завантаження краща, а ніж з клієнтським рендерингом.

– Поліпшена доступність для систем які займаються пошуком інформації, оскільки весь контент вже генерується на сервері.

Недоліки:

– При кожному запиті потрібна нова генерація сторінки, що в свою чергу може підвищити навантаження на сервер.

– Зниження загальної швидкості сайту для користувачів.

3. Мобільний застосунок – розробляються виключно для використання на смартфонах або планшетах. Такі застосунки зазвичай є нативними, тобто вони розроблені конкретно для якоїсь платформи (Android або iOS), або крос-платформними, що дозволяє запускати їх на різних ОС.

Переваги:

– Більш глибока інтеграція з апаратним забезпеченням та операційною системою в цілому.

– Висока швидкість роботи та інтерактивність такої системи.

– Можливість роботи офлайн з такими застосунками.

Недоліки:

– Високі витрати на розробку таких застосунків та підтримку окремих версій для різних платформдля забезпечення крос-платформеності.

– Необхідність завантаження та інсталяції користувачами.

4. Настільний додаток – це такі програми, що встановлюються та виконуються на ПК або ноутбуці, незалежно від веб-браузера, який встановлений на них. Вони можуть бути розроблені для різних ОС, таких як Windows, macOS, або Linux. Вони часто використовуються для завдань, що вимагають інтенсивних обчислень, великого обсягу локального зберігання або специфічного апаратного інтерфейсу.

Переваги:

– В них присутня висока продуктивність та швидкість роботи завдяки прямому доступу до апаратних ресурсів комп'ютера.

– Є можливість роботи без постійного з'єднання з Інтернетом.

– Присутні можливості для персоналізації інтерфейсу та функціоналу під специфічні задачі користувача.

Недоліки:

– Виникає необхідність в окремій розробці та підтримки для кожної ОС, що збільшує витрати організації.

– Є потреба у встановленні та оновленні ПЗ користувачем, що може бути незручно.

Вибір підходу до реалізації клієнтської частини програмного застосунку може залежити від таких факторів як: цільова аудиторія застосунку, типи пристроїв, вимоги до продуктивності та доступності системи.

2.3 Проектування архітектури системи

Програмна система буде складатися з 3 компонентів: веб-додаток, сервер та сервер БД, ці компоненти будуть постійно взаємодіяти між собою.

Монолітний додаток - це додаток, що представляється через єдине розгортання. Програма на Node з одним входом є прикладом такого додатку. У контексті розробки ПЗ, монолітний підхід описує програму як одне ціле. Основна ідея монолітного ПЗ полягає в тому, що різноманітні складові програми

інтегровані у єдину систему на одній платформі. Монолітний додаток складається з БД, інтерфейсу користувача та серверного додатку. Всі частини ПЗ уніфіковані, а всі його функції керуються в одному місці.

Монолітна архітектура підходить для роботи з невеликими проектами, тому багато стартаперів вибирають такий підхід при створенні власного програмного продукту. Складові монолітного ПЗ взаємопов'язані і взаємозалежні, що допомагає ПЗ бути автономним. Всвою чергу монолітна архітектура є рішенням для побудови невеликих додатків.

Виходячи з вище сказаного, можна виділити основні плюси такої архітектури для розробки власного програмного продукту:

– Проста розробка та розгортання: є багато інструментів, які можна об’єднати для сприяння розвитку. В монолітній архітектурі дії виконуються з одним каталогом, який забезпечує легше розгортання. В монолітній архітектурі розробникам не потрібно розгортати або оновлювати окремо, оскільки вони можуть робити це одночасно і заощадити часу та грошовий ресурс.

– Легка інтеграція системи моніторингу: більшість додатків залежать аналізу трафіку та обмеження швидкості. Програми розроблені на монолітній архітектурі значно полегшують ці проблеми, завдяки тому що у них єдина база з кодом. Легко підключати будь-які компоненти, коли все працює в тому ж самому додатку.

– Відносно краща продуктивність: якщо монолітні програми побудовані належним чином, то вони будуть більш ефективнішими ніж програми на основі мікросервісу. Наприклад, програма з мікросервісною архітектурою може потребувати 30 викликів API для 30 різних мікросервісів, наприклад для завантаження кожного екрану, що очевидно призводе до менш продуктивної системи. Монолітна програма, у свою чергу дозволяює швидко спілкуватися між компонентами ПЗ через спільний код і пам'ять.

При проектуванні інтерфейсу користувача потрібно користуватися такими правилами, які представлені нижче:

1. Звести до мінімуму навантаження на користувачів, зробивши об'єкти, дії та параметри помітними для нього. Користувач не повинен згадувати

інформацію з однієї частини діалогового вікна до іншої. Інструкції з використання системи повинні бути помітними або доступними, коли це необхідно.

2. Потрібно підбирати слова, фрази та концепції, знайомі та прості для користувача, представляючи інформацію логічно та послідовно.

3. Потрібно уникати невизначеності щодо значення різних термінів, ситуацій або команд.

4. Потрібно надати користувачеві можливість легкого виходу з будь-якого небажаного стану без проходження складних діалогів за допомогою чітких опцій для скасування або повторення дій.

5. Прискорювачі, невидимі для користувача-початківця, часто можуть прискорити взаємодію для досвідченого користувача, тому система може обслуговувати як недосвідчених, так і досвідчених користувачів. Дозвольте користувачам налаштовувати часті дії.

6. Потрібно уникати непотрібної інформації в діалогових вікнах, щоб не відволікати увагу користувача від важливих даних.

7. Повідомлення про помилки мають бути зрозумілі для користувача, вони повинні вказувати на суть проблеми та пропонувати конструктивне рішення.

8. Якщо систему можна використовувати без документації, користувачеві може знадобитися допомога. Подібна інформація повинна бути зручною для пошуку, орієнтованої на завдання, утримувати список конкретних кроків, які необхідно виконати, і не повинна бути занадто великою.

9. Потрібно допомогти користувачам розпізнавати, діагностувати та усувати помилки. Повідомлення про помилки мають бути викладені простою мовою, точно вказувати на проблему та конструктивно пропонувати рішення.

Переглядаючи правила, які варто враховувати, щоб інтерфейс був інтуїтивно зрозумілий був розроблений макет інтерфейсу користувача для веб-додатку, який представлений на рисунку 2.3.

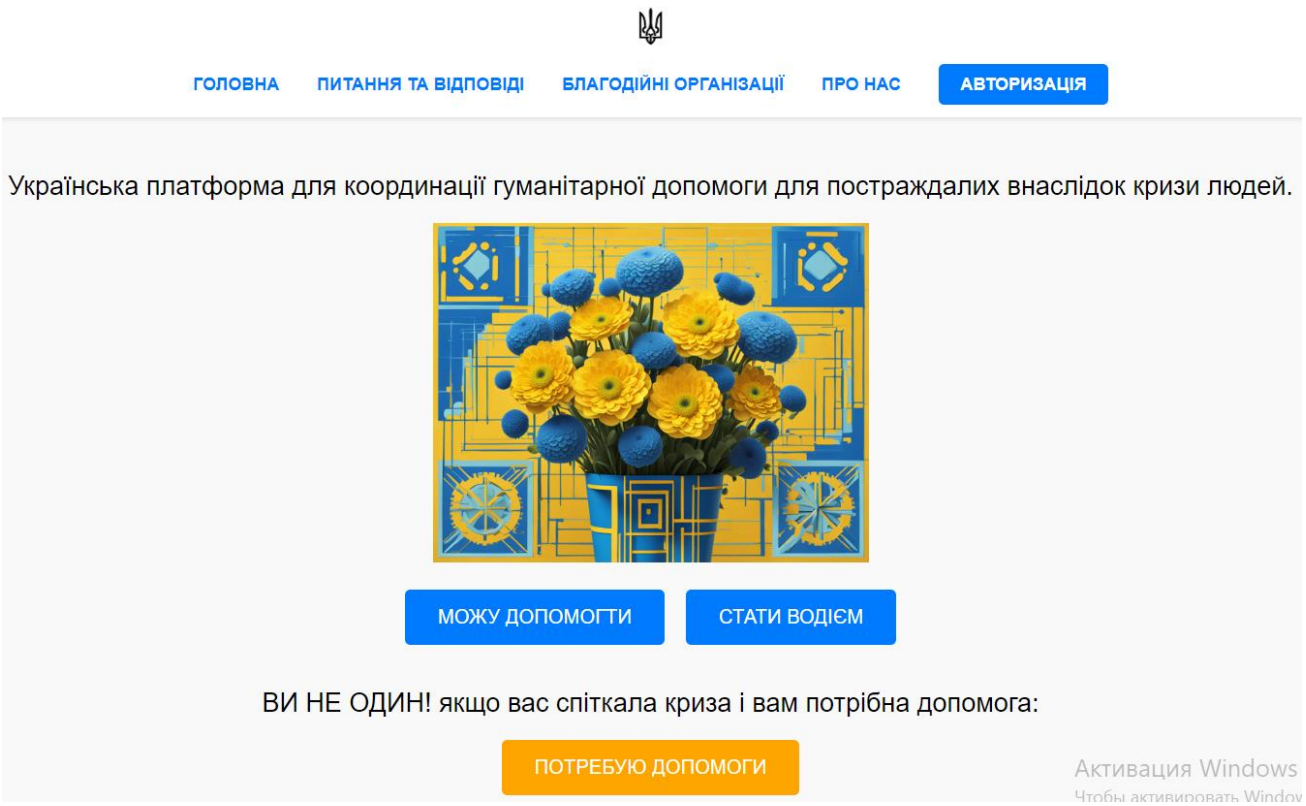


Рисунок 2.3 – Інтерфейс користувача

2.4 Обґрунтування вибору мови програмування

Для створення інформаційної системи була обрана мова програмування C# – це сучасна об'єктно-орієнтована та типобезпечна мова програмування. C# дозволяє розробникам створювати різні типи безпечних та надійних програм, що виконуються в .NET. C# відноситься до широко відомого сімейства мов C.

Написання програм на мові C# забезпечує надійність і стійкість цих додатків завдяки таким функціям мови, як прибирання непотрібного, що автоматично звільняє пам'ять яка зайнята невикористаними об'єктами. Обробка винятків, що надає структурований і розширений підхід до виявлення помилок і їх відновлення. А типобезпечна структура мови унеможливлює читання з неініціалізованих змінних, індексацію масивів за межами їх кордонів або виконання неперевірених зведень типів.

Програми C# виконуються в .NET, віртуальній системі виконання, що викликає середовище виконання (CLR) та набір бібліотек класів. Середовище

CLR – це реалізація загальномовної інфраструктури мови (CLI), що є міжнародним стандартом від корпорації Майкрософт. CLI є основою для створення середовищ виконання та розробки, у яких мови та бібліотеки прозоро працюють одна з одною.

Вихідний код, написаний мовою C# компілюється в проміжну мову (IL), що відповідає специфікаціям CLI. Код на мові IL та ресурси, у тому числі растрові зображення та рядки, зберігаються у збірці, зазвичай з розширенням .dll. Складання містить маніфест з інформацією про типи, версії, мову та параметри для цієї збірки.

Під час виконання програми C# збірка завантажується у середу CLR. Середовище CLR виконує компіляцію з коду мовою IL в інструкції машинної мови. Середовище CLR також виконує інші операції, такі як: автоматичне складання сміття, обробку винятків та управління ресурсами. Код, який виконується середовищем CLR, іноді називають "керованим кодом". "Некерований код" компілюється на машинну мову, призначену для конкретної платформи.

Забезпечення взаємодії між мовами є ключовою особливістю .NET. Код IL, створений компілятором C# відповідає специфікації загальних типів. Одна збірка може містити кілька модулів, написаних різними мовами .NET, і всі типи можуть посилатися один на одного, якби вони були написані однією мовою.

Крім служб часу виконання .NET також включає розширені бібліотеки. Ці бібліотеки підтримують безліч різних робочих навантажень. Вони впорядковані за просторами імен, які надають різні корисні можливості: від операцій файлового введення та виведення до керування рядками та синтаксичного аналізу XML, від платформ веб-додатків до елементів керування Windows Forms. Зазвичай програма C# активно використовують бібліотеку класів .NET для вирішення типових завдань.

Завдяки тому, що C# – це об'єктно-орієнтована мова, це надає можливість легко будувати абстракції у вигляді наборів класів, що значно полегшує процес

розробки додатків. Окрім цього C# має досить багато фреймворків, що можна використати дляполегшення написання коду.

2.5 Обґрунтування вибору СУБД

В якості системи управління БД був обраний SQL Server Management Studio – це інтегроване середовище для управління будь-якою інфраструктурою SQL, від SQL Server до баз даних SQL Azure. SSMS надає засоби для налаштування, спостереження та адміністрування екземплярів SQL Server та баз даних. Використовуйте SSMS для розгортання, моніторингу та оновлення компонентів рівня даних, що використовуються програмами, та створення запитів та скриптів.

SSMS можна використовувати для запиту, проектування та управління базами даних та сховищами даних, де б вони не знаходилися на локальному комп'ютері або у хмарі.

Основною перевагою даного програмного продукту є те, що воно добре поєднується з продуктами компанії Microsoft, а саме мовою програмування C#, на якій і буде написана дана ІС.

Окрім цього SQL Server Management Studio досить легка у використанні. В ній вбудований оглядач об'єктів, який дозволяє переглядати всі об'єкти сервера, і надає графічний інтерфейс для управління цими об'єктами. Таким чином за допомогою оглядача об'єктів, ви легко можете подивитися які бази даних, таблиці, функції і так далі є, які користувачі створені, які пов'язані сервери налаштовані.

3 РОЗРОБКА АРХІТЕКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Створення функціональної моделі ІС

Методологія SADT є сукупністю методів, правил і процедур, призначених для побудови функціональної моделі об'єкта будь-якої предметної області. Функціональна модель SADT відображає багатофункціональну структуру об'єкта, тобто, вироблені їм дії та зв'язки між цими діями. Основні елементи цієї методології ґрунтуються на таких концепціях:

– графічне уявлення блочного моделювання. Графіка блоків і дуг SADT-діаграми відображає функцію у вигляді блоку, а інтерфейси входу/виходу представляються дугами, що відповідно входять в блок і виходять з нього. Взаємодія блоків один з одним описуються за допомогою інтерфейсних дуг, що виражають "обмеження", які в свою чергу визначають, коли і яким чином функції виконуються та управляються;

– суворість та точність. Виконання правил SADT вимагає достатньої строгості та точності, не накладаючи водночас надмірних обмежень на дії аналітика.

Правила SADT включають:

– обмеження кількості блоків на кожному рівні декомпозиції (правило 3-6 блоків);

– зв'язність діаграм (номери блоків);

– унікальність міток та найменувань (відсутність повторюваних імен).

Перший етап – створення контекстної діаграми (рис. 3.1). На ній визначено головний процес ІС, а саме «Моніторинг та координація гуманітарної допомоги». На вхід поступає запит користувача на отримання допомоги. До керуючих механізмів належать данні про допомогу, політика розподілу допомоги та

міжнародні стандарти гуманітарної діяльності. Механізмами виступають: волонтери, волонтери-водії та сама ІС. На виході повинно бути отримано отримання допомоги.

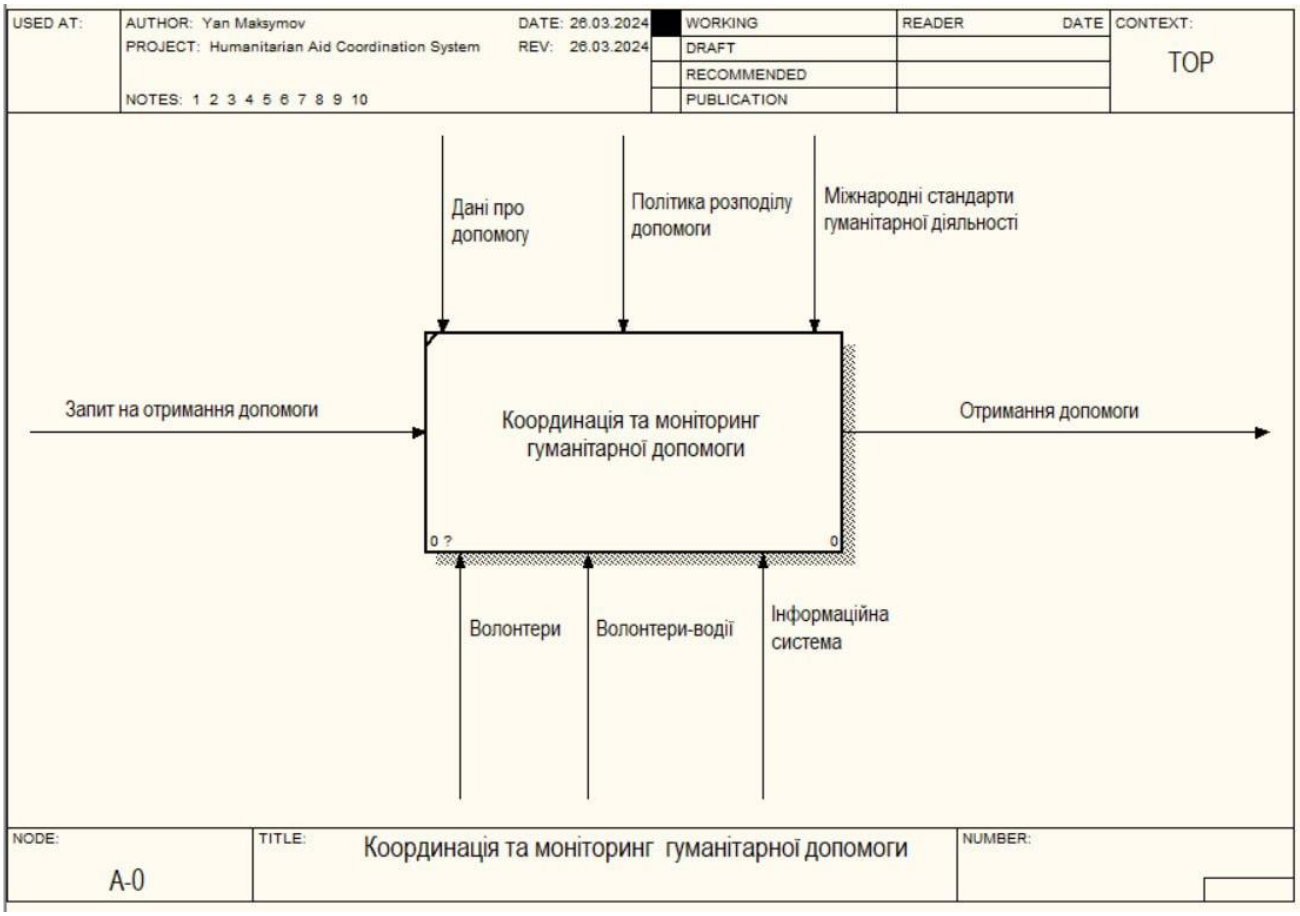


Рисунок 3.1 – Діаграма IDEF0 головного процесу

Декомпозуємо великий процес на більш дрібні. І маємо декілька процесів, таких як: формування заявки на отримання допомоги, оцінка та розподіл заявок, відбір заявок та планування доставки, доставка гуманітарної допомоги, моніторинг статусу заявки . На рисунку 3.2 представлено декомпозицію головного процесу.

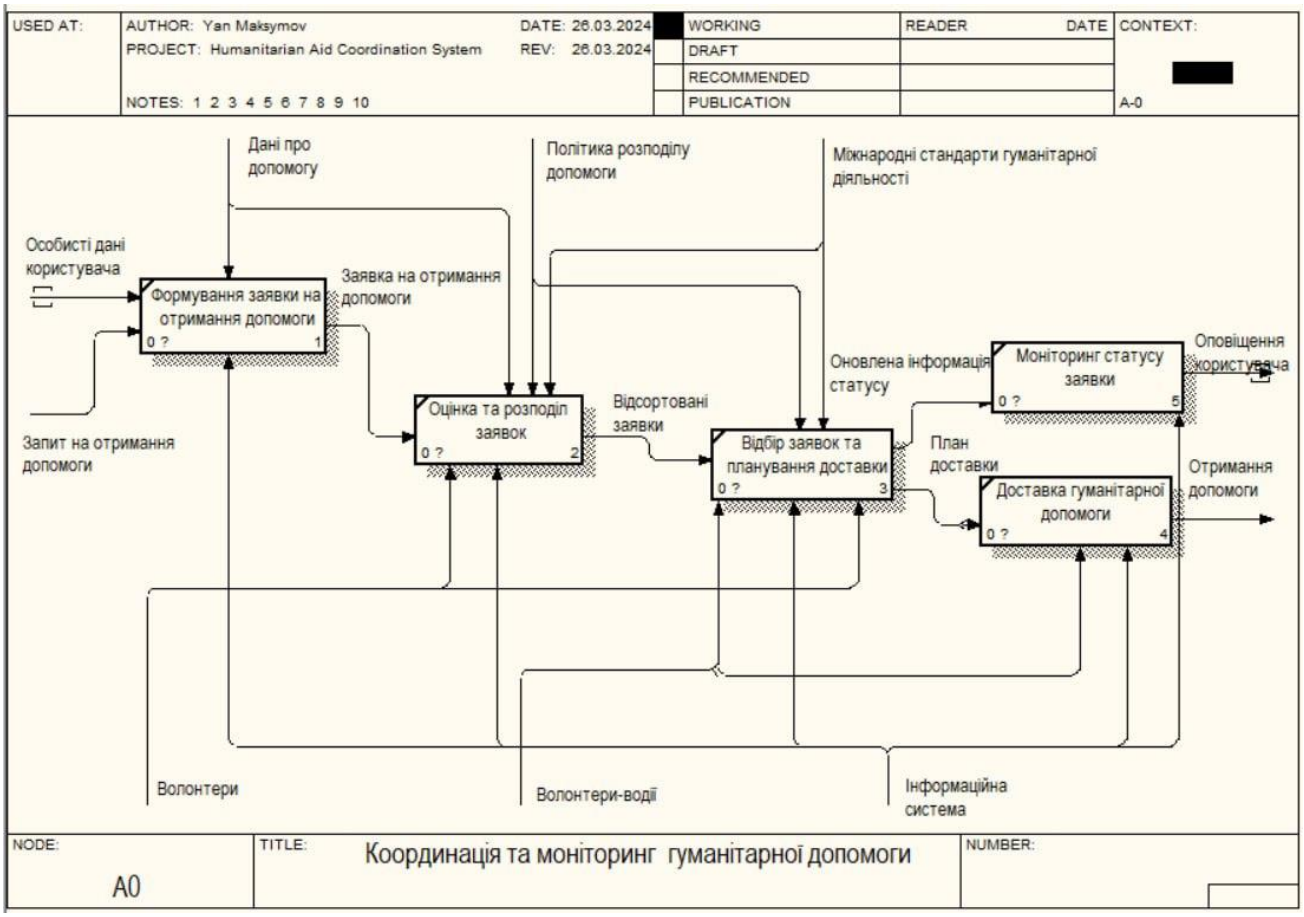


Рисунок 3.2 – Діаграма декомпозиції головного процесу

Стрілки залишилися ті ж самі, але кількість варіацій того з чим вони з'єднуються збільшилась. Щоб оформити заявку на отримання допомоги потрібно: надати запит на отримання допомоги і особисті дані користувача, заяку оцінють та перевіряють волонтери, якщо все гаразд вони її передають до виконання, якщо ні, то вони уточнюють потрібні дані, після чого волонтер-водій відбирає заявки які він зможе закрити та планує свою доставку, головний процес вважається виконаним, якщо користувач отримав свою допомогу.

3.2 Склад сервісів ПЗ

На рисунку 2.8 представлено структурну схему складу сервісів, що будуть розроблені в процесі кваліфікаційної роботи.

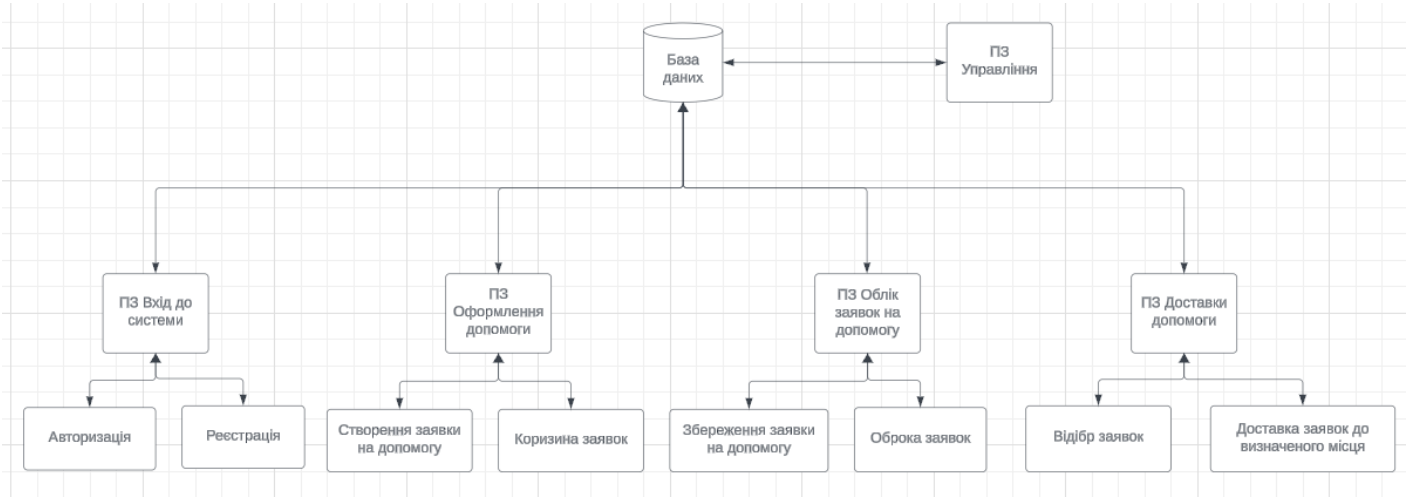


Рисунок 3.3 – Структурна схема складу сервісів

3.3 Визначення функцій серверної частини ІС

У серверній частині є такі функції:

- Реєстрація в системі.
- Авторизація.
- Створення заявки на допомогу.
- Перегляд власних заявок.
- Відстеження статусу заявки.
- Оновлення інформації про допомогу.
- Отримання інформації про допомогу.
- Перегляд заявок для водія.
- Прийняття заявок до роботи.

3.4 UML - моделювання клієнтської частини ІС

Визначившись з основним функціоналом системи можна розробити діаграми, які будуть показувати взаємодію між усіма компонентами системи.

При виконанні проекту важливе значення має моделювання предметної галузі. Зараз для моделювання предметної галузі та розробки структури проекту можна використовувати різноманітні CASE-засоби (IDEF0, DFD, IDEF3). Але одне з найбільших значень для створення цієї спільноти має використання уніфікованої мови моделювання – UnifiedModelingLanguage, UML.

UML - це уніфікована графічна мова моделювання для опису, візуалізації, проектування та документування систем. UML покликаний підтримувати процес моделювання програмних систем, організовувати взаємозв'язок концептуальних і програмних понять, відображати проблеми масштабування складних систем. Найчастіше вона використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю. Була створена для визначення, візуалізації, проектування й документування в основному програмних систем. Сама по собі не є мовою програмування, але в засобах виконання UML-моделей, як інтерпретованого коду, можлива кодогенерація.

Моделі на UML використовуються на всіх етапах життєвого циклу програмних систем, починаючи з бізнес-аналізу і закінчуючи супроводом системи. Різні організації можуть застосовувати UML на свій розсуд в залежності від своїх проблемних областей і використовуваних технологій.

3.4.1 USE CASE діаграма

Функціональність, що забезпечується системою описується за допомогою функціональної моделі, яка відображає системні прецеденти, системне оточення і зв'язку між цими прецедентами. Основне завдання моделі прецедентів - являти собою єдиний засіб, що дає можливість кінцевому користувачеві і розробнику спільно обговорювати функціональність і поведінку системи.

Створення моделі прецедентів починається на стадії задумки з вибору акторів і визначення загальних принципів функціонування системи. Потім на етапі опрацювання модель доповнюється детальною інформацією до існуючих прецедентів.

На рисунку 3.4 представлена діаграма прецедентів. Було вияснено що з системою буде взаємодіяти користувач, волонтер та водій.

Користувач може зареєструватися в системі та авторизуватися, якщо він вже зареєстрований, також він має можливість запросити допомоги від організації, а саме створити заявку на отримання допомоги. В особистому кабінеті користувач може переглядати статус своєї заявки.

Волонтер – це особа, яка є членом гуманітарної організації, в свою чергу він може створювати пропозиції надання допомоги для користувачів. Також він займається прийняттям заявок на допомогу від користувачів та передає далі до виконання доправлення гуманітарної допомоги.

Водій – це особа, яка доставляє гуманітарну допомогу до місця її потребування. Водій в особистому кабінеті може переглядати існуючи заявки від користувачів та братися до їх доставки. Також він може самостійно створити власної пропозиції на доправлення гуманітарної допомоги.

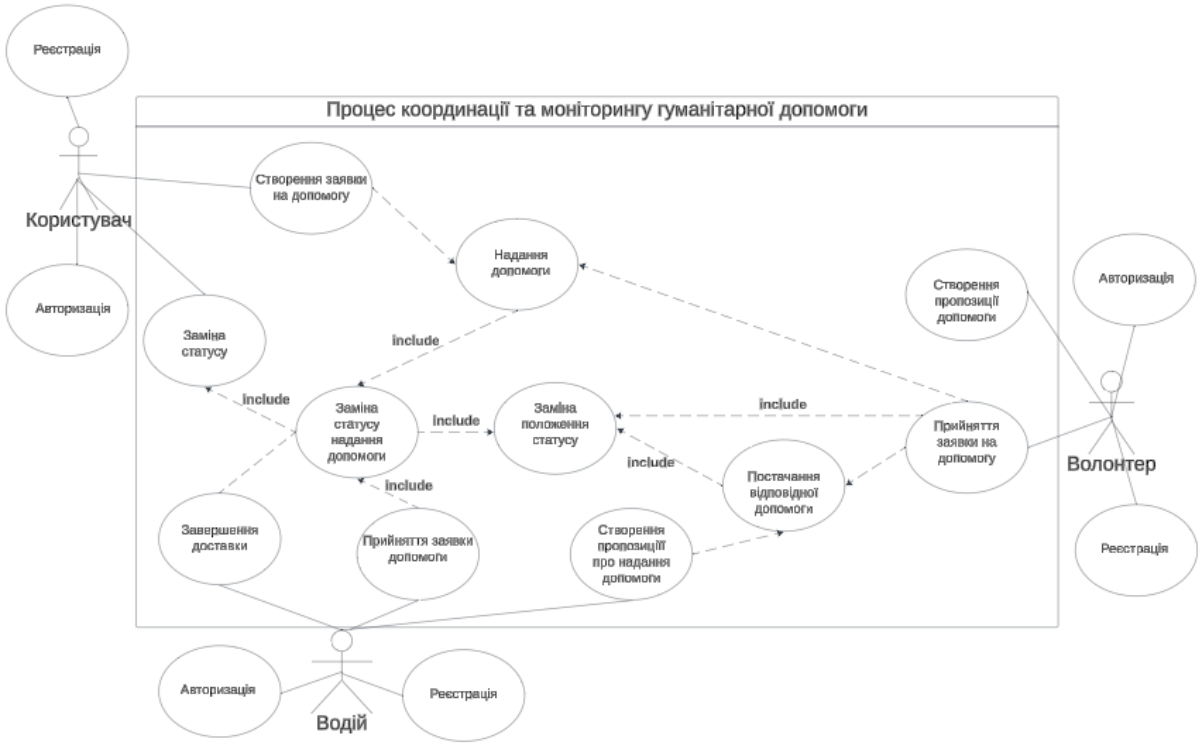


Рисунок 3.4 – USE CASE діаграма

3.4.2 Діаграма послідовності (Sequence diagram)

Діаграма послідовності - це послідовність подій, необхідних для забезпечення необхідного поводження. Потік подій описується в термінах того, «що» система повинна робити, а не «як» вона повинна це робити. Тобто він описується на мові предметної області, а не термінами реалізації.

Потік подій повинен визначати:

- коли і як прецедент починається і закінчується;
- як він взаємодіє з актором;
- які дані йому потрібні;
- нормальну послідовність подій для прецеденту;
- опис потоків в альтернативних і виняткових сі-ситуаціях.

На рисунку 3.5 представлена діаграма послідовності моніторингу та координації гуманітарної допомоги.

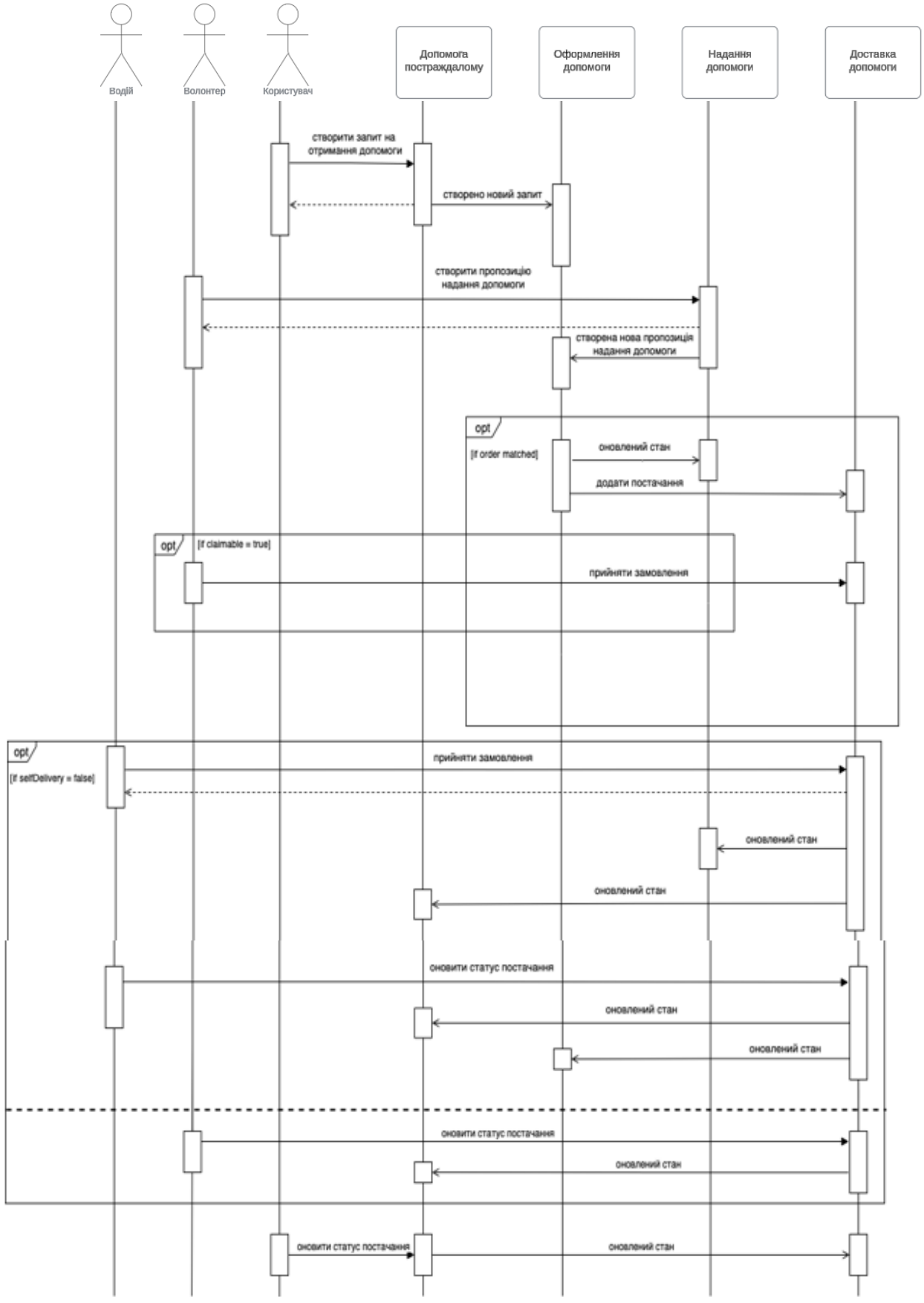


Рисунок 3.5 – Діаграма послідовності

3.4.3 Діаграма класів

Ціллю створення діаграми класів є відображення класів, що присутні у інформаційній системі, операцій над ними, взаємодій та атрибутів класів. Вона дозволяє формалізувати логічну модель на рівні структурних елементів системи, тобто класів та надає статичне представлення про структуру програми – з яких класів вона складається, які зв’язки між цими класами, з чого складається кожний клас.

Модель кожного класу є шаблоном майбутнього об’єкту, який створюється на основі цього класу. Кожна програма може містити чисельну кількість різних об’єктів. Як відомо, об’єкти мають стан та поведінку, відповідно до концепції об’єктно-орієнтованого програмування. На рівні класу стан описується полями даних, а поведінка – методами. Реалізована діаграма класів має визначати структуру даних майбутнього програмного додатку, який отримує замовник.

Формуючи модель даних, відповідно до предметної області та бізнес логіки, варто виділити три ключові сутності: запит на отримання допомоги AidOrder, пропозиції забезпечення допомоги AidProvision, пропозиції доставки допомоги AidSupply.

Для проекту була розроблена наступна діаграма класів (рис 3.6).

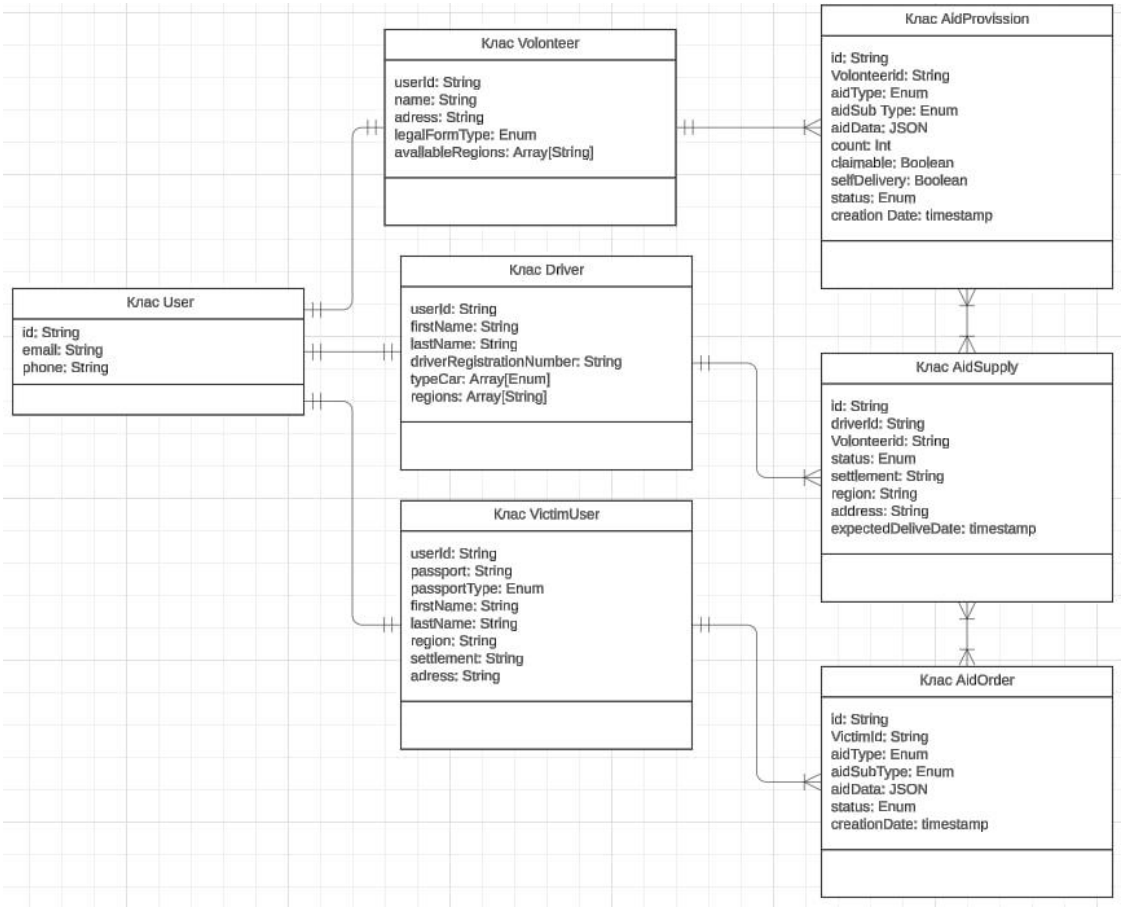


Рисунок 3.6 – Діаграма класів ІС

Під час аналізу вимог до розробки проекту ІС, мною було прийняте рішення щодо реалізації усіх класів у вигляді прототипів сторінок, кожна з яких відіграє ключову роль в системі.

4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСТОСУНКУ

4.1 Опис технічної бази та середовища розробки

Розроблювана система є комплексним рішенням, яке вимагає постійного використання обчислювальних ресурсів для стабільної роботи в мережі. Для ефективної роботи необхідні сучасні комп'ютери, які можуть бути представлені у вигляді обслуговуваного сервера або ресурсів у хмарному середовищі.

При використанні серверних рішень, мінімальні технічні характеристики обчислювальної машини такі:

– архітектура процесора x64;

– мінімум 256 МБ оперативної пам’яті для запуску одного мікросервісу;

– не менше 1 ГБ місця на жорсткому диску;

– операційна система на базі Microsoft (наприклад Windows 10).

Усі розроблені сервіси базуються на технологічному стеку .NET та написані на мові програмування C#. Для запуску програмного забезпечення необхідно встановити відповідне програмне середовище: .NET SDK.

Для доступу до БД у C# використовується Entity Framework (EF) Core. EF Core - це ORM (Object-Relational Mapper), який дозволяє розробникам працювати з БД, використовуючи об'єктно-орієнтовані принципи. Він підтримує як реляційні, так і нереляційні бази даних. Основні переваги EF Core включають автоматичне створення та міграцію бази даних, підтримку запитів LINQ, а також спрощення роботи з даними.

EF Identity - це бібліотека, що інтегрується з EF Core для керування користувачами, ролями та аутентифікацією. Вона забезпечує потужні механізми для створення та керування обліковими записами користувачів, а також для впровадження аутентифікації та авторизації в додатках.

Принцип Model First передбачає спочатку створення моделей даних у коді, а потім генерування бази даних на основі цих моделей. Це дозволяє зосередитися на доменній моделі додатку, а не на структурі БД. Перевагами такого підходу є простота модифікації моделей, автоматична синхронізація з базою даних та покращена підтримка міграцій.

4.2 Створення бази даних для платформи SQL Server Management Studio

Для створення бази даних необхідно виконати скрипт в середовищі SQL Server Management Studio. Фізична модель бази даних в середовищі SQL Server Management Studio представлена на рисунку 4.1.

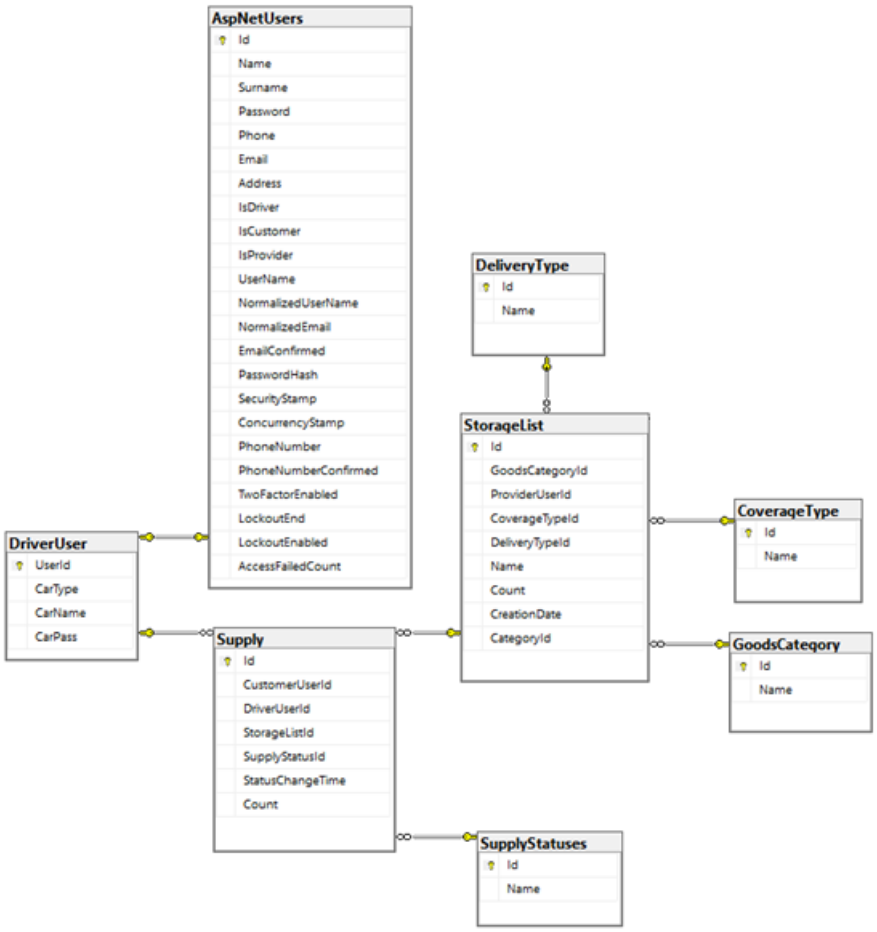


Рисунок 4.1 – Фізична модель БД в середовищі SQL Server Management Studio

4.3 Розробка бізнес-логіки системи

Виділемо основні методи, які реалізують бізнес-логіку та описують головні процеси інформаційної системи:

– Метод що описує оформлення запиту на допомогу представлено на рисунку 4.2.

```
public IActionResult MakeOrder([FromBody] SelectHelpViewModel model)
{
    using (VolunteerOrgContext db = new VolunteerOrgContext())
    {
        var user = db.User.Where(u => u.Email == User.Identity.Name).FirstOrDefault();
        var storage = db.StorageList.Where(i => i.Id == model.StorageId).FirstOrDefault();
        storage.Count = storage.Count - model.Count;


        var newSupply = new Supply()
        {
            Id = Guid.NewGuid().ToString(),
            CustomerUserId = user.Id,
            DriverUserId = null,
            StatusChangeTime = DateTime.Now,
            Count = model.Count,
            StorageListId = storage.Id,
           SupplyStatusId = "1"
        };

        db.Supply.Add(newSupply);
        db.SaveChanges();
        return Json(new { success = true });
    }
}
```

Рисунок 4.2 – Метод оформлення запиту на допомогу

– Метод що описує заміни статусу допомоги на рисунку 4.3.

```
public IActionResult ChangeStatus(string supplyId ,string statusId)
{
    using (VolunteerOrgContext db = new VolunteerOrgContext())
    {
        var user = db.User.Where(u => u.Email == User.Identity.Name).FirstOrDefault();
        var supply = db.Supply.Where(s => s.Id == supplyId).FirstOrDefault();
        supply.SupplyStatusId = statusId;
        supply.StatusChangeTime = DateTime.Now;
        db.SaveChanges();
        return Json(new { success = true });
    }
}
```

Рисунок 4.3 – Метод заміни статусу допомоги

– Метод що описує надання допомоги представлено на рисунку 3.5.

```
public IActionResult AddNewItem([FromBody] CreateSupplyViewModel model)
{
    using (VolunteerOrgContext db = new VolunteerOrgContext())
    {
        var user = db.User.Where(u => u.Email == User.Identity.Name).FirstOrDefault();
        var newItem = new StorageList()
        {
            Id = Guid.NewGuid().ToString(),
            Count = model.Count,
            Name = model.Name,
            CoverageTypeId = model.CoverageTypeId,
            GoodsCategoryId = model.CategoryId,
            ProviderUserId = user.Id,
            DeliveryTypeId = model.DeliveryTypeId,
            CreationDate = DateTime.Now
        };
        db.StorageList.Add(newItem);
        db.SaveChanges();
        var supplyStatuses = db.SupplyStatuses.Select(s => new { s.Id, s.Name }).ToList();

        // Повернення JSON відповіді з новим товаром та списком статусів
        return Json(new
        {
            success = true,
            newItem = new
            {
                StorageId = newItem.Id,
                UserSurname = user.Surname,
                UserName = user.Name,
                UserEmail = user.Email,
                DriverSurname = "",
                DriverName = "",
                DriverEmail = "",
                CustomerSurname = "",
                CustomerName = "",
                CustomerEmail = "",
                AddressFrom = user.Address,
                AddressTo = "",
                GoodName = newItem.Name,
                GoodCategoryName = newItem.GoodsCategoryId,
                Count = newItem.Count,
                CoverageType = newItem.CoverageTypeId,
                StatusName = "На складі"
            },
            supplyStatuses
        });
    }
}
```

Рисунок 4.4 – Метод надання допомоги

4.4 Заповнення даними БД

Для перевірки коректності БД заповнимо її даними.

	Id	Name	Surname	Pass...	Phone	Email	Address	IsDriver	IsCustomer	IsProvider	UserName	NormalizedUserName	PasswordHash
1	1...	Максим	Власенко	Yan....	0984357843	maksym.vlasenko@ukr.net	м. Часів Яр	0	1	1	maksym.vlasenko@ukr.net	MAKSYM.VLASENKO@UKR.NET	AQAAAAEAACcQAAAA
2	2..	Григорій	Гришин	Yan....	0650985876	grishin.grugorii@gmail.com	м. Слов'янськ	0	1	1	grishin.grugorii@gmail.com	GRISHIN.GRUGORII@GMAIL.COM	AQAAAAEAACcQAAAA
3	4..	Максим	Федоров	Yan....	0985684376	maksym12fedorov@ukr.net	м. Київ	0	1	1	maksym12fedorov@ukr.net	MAKSYM12FEDOROV@UKR.NET	AQAAAAEAACcQAAAA
4	7..	Дмитро	Кравченко	Krav...	0670985987	kravchenko.dmitrii@gmail.com	м. Ірпінь	1	1	1	kravchenko.dmitrii@gmail.com	KRAVCHENKO.DMITRII@GMAIL.COM	AQAAAAEAACcQAAAA
5	9..	Степан	Громов	Yan....	0680941556	stepan.gromov@ukr.net	м. Київ	1	1	1	stepan.gromov@ukr.net	STEPAN.GROMOV@UKR.NET	AQAAAAEAACcQAAAA
6	a..	Максим	Самойленко	Yan....	0985486456	samoulenkomaksim21@ukr.net	м. Львів	0	1	1	samoulenkomaksim21@ukr.net	SAMOULENKOMAKSIM21@UKR.NET	AQAAAAEAACcQAAAA
7	d..	Ігор	Конкур	Igor....	0680941443	igor.konk@gmail.com	м. Слов'янськ	0	1	1	igor.konk@gmail.com	IGOR.KONK@GMAIL.COM	AQAAAAEAACcQAAAA
8	e..	Влад	Іванов	Vlad...	09859876385	vlad.ivanov21@gmail.com	м. Слов'янськ	1	1	1	vlad.ivanov21@gmail.com	VLAD.IVANOV21@GMAIL.COM	AQAAAAEAACcQAAAA

Рисунок 4.5 - Заповнена таблиця «Користувачі»

	UserId	CarType	CarName	CarPass
1	7c839a7d-8c2f-4a04-8e86-5a860b7b660f	Грузове	Mercedes-Benz	DI432424
2	9c200570-9a60-4d93-a1ca-4a071c6360af	Універсал	Volvo	DI534234
3	e87b4792-caa2-486d-a841-415b8970861c	Універсал	Volvo	DI31233

Рисунок 4.6 - Заповнена таблиця «Водії»

	Id	CustomerUserId	DriverUserId	StorageListId	SupplyStatusId	StatusChangeTime	Count
1	027ca6c7...	1daaf461-57ae-4b...	NULL	148d6eca-8d...	1	2024-06-11 08:06:05...	20
2	1ececf6d-...	dccdb996-b8a2-47...	e87b4792-caa2-48...	4c076a0f-8ce...	7	2024-06-11 08:21:04...	20
3	3ddb21f2-...	229a1a5b-4c72-4...	NULL	18c337e4-48c...	1	2024-06-11 08:07:23...	2
4	43810d1e...	229a1a5b-4c72-4...	NULL	a84f8682-b8c...	1	2024-06-10 14:01:49...	10
5	542dba10...	1daaf461-57ae-4b...	7c839a7d-8c2f-4a...	01a19438-34...	4	2024-06-11 08:15:28...	10
6	5f3f9d6c-...	1daaf461-57ae-4b...	7c839a7d-8c2f-4a...	4c076a0f-8ce...	7	2024-06-11 08:21:05...	20
7	70381a38...	229a1a5b-4c72-4...	7c839a7d-8c2f-4a...	2d96d1f8-3b7...	7	2024-06-11 08:16:14...	10
8	7048a2de...	229a1a5b-4c72-4...	NULL	ff10e17e-b36...	1	2024-06-11 08:07:40...	10
9	712d68df-...	1daaf461-57ae-4b...	NULL	fbe1bbdc-cd0...	1	2024-06-11 08:05:37...	10
10	75f4a3c4-...	229a1a5b-4c72-4...	7c839a7d-8c2f-4a...	01a19438-34...	3	2024-06-11 08:21:51...	4
11	7e2a2de9...	1daaf461-57ae-4b...	NULL	a4c0de8b-7d...	1	2024-06-11 08:17:09...	10
12	aa84f4f7-...	1daaf461-57ae-4b...	NULL	0084135d-85...	1	2024-06-11 08:16:54...	50
13	ba613849...	dccdb996-b8a2-47...	e87b4792-caa2-48...	e8427383-69...	7	2024-06-11 08:21:12...	100
14	c1ac5753...	1daaf461-57ae-4b...	7c839a7d-8c2f-4a...	e3541b92-f77...	3	2024-06-11 12:20:56...	5
15	cdbe64db...	dccdb996-b8a2-47...	e87b4792-caa2-48...	1fa2fd43-be7...	6	2024-06-11 08:20:54...	10
16	cfb6ffff-8a...	1daaf461-57ae-4b...	7c839a7d-8c2f-4a...	efb734bd-e46...	4	2024-06-11 08:15:20...	10
17	d30998d5...	1daaf461-57ae-4b...	NULL	148d6eca-8d...	1	2024-06-11 08:16:57...	50

Рисунок 4.7 - Заповнена таблиця «Доставка»

	Id	Name
1	1	Одяг
2	2	Медикаменти
3	3	Їжа
4	4	Електроніка
5	5	Засоби гігієни

Рисунок 4.8 - Заповнена таблиця «Категорії допомоги»

	Id	Name
1	0	На складі
2	1	Зарезервовано користувачем
3	2	Пошук водія
4	3	Отримано водієм
5	4	У дорозі
6	5	Готове до отримання
7	6	Відхилено
8	7	Отримано

Рисунок 4.9 - Заповнена таблиця «Статуси доставки»

	Id	GoodsCategoryId	ProviderUserId	CoverageTypeId	DeliveryTypeId	Name	Count	CreationDate
1	0084...	5	4ca73b1a-8c78-...	1	2	Підгузки	50	2024-06-11 07:48:11.9815256
2	01a1...	1	4ca73b1a-8c78-...	1	1	Взуття	10	2024-06-10 15:26:33.5885195
3	0a4e...	4	4ca73b1a-8c78-...	1	2	Павербанки	20	2024-06-10 20:26:06.1999655
4	13c63...	5	4ca73b1a-8c78-...	1	2	Шампунь (шт)	120	2024-06-10 20:28:01.0926542
5	148d...	5	9c200570-9a60...	1	2	Туалетний папір	130	2024-06-11 07:50:55.8439054
6	18c33...	1	a056511a-d8f4-...	1	2	Зимові рукавиці	48	2024-06-11 07:44:47.9256852
7	1bc2a...	1	a056511a-d8f4-...	1	2	Одяг для вагітних	10	2024-06-11 07:45:16.4740662
8	1fa2fd...	5	4ca73b1a-8c78-...	1	2	Підгузки	90	2024-06-11 07:48:11.9815389
9	2c4fc...	3	9c200570-9a60...	2	1	Готова їжа (порції)	100	2024-06-10 15:31:51.6567680
10	2d96...	2	7c839a7d-8c2f-...	1	1	Антибіотики	0	2024-06-10 13:53:03.7388681
11	3b2a...	1	9c200570-9a60...	1	2	Гелі	46	2024-06-11 07:54:50.7605539
12	4c076...	5	4ca73b1a-8c78-...	1	2	Мило	260	2024-06-10 20:26:52.1408993
13	4dd8...	3	a056511a-d8f4-...	3	1	Дитяче харчування	36	2024-06-10 20:41:06.8732834
14	554a...	3	9c200570-9a60...	1	2	Фрукти в сиропі	100	2024-06-11 07:58:21.8308452
15	556e...	1	e87b4792-caa2...	1	2	Чоловіча нижня білизна	40	2024-06-10 20:24:08.6262154
16	56a7...	3	9c200570-9a60...	1	2	Рибні консерви	40	2024-06-11 07:56:58.0642111
17	803b...	5	4ca73b1a-8c78-...	1	2	Підгузки	100	2024-06-11 07:48:11.9815339

Рисунок 4.10 - Заповнена таблиця «Список допомоги»

	Id	Name
1	1	Самостійно
2	2	Потребую доставки

Рисунок 4.11 - Заповнена таблиця «Типи доставки»

	Id	Name
1	1	По всій країні
2	2	По місту
3	3	По області

Рисунок 4.12 - Заповнена таблиця «Покриття доставки»

4.5 Розробка інтерфейсу клієнтської частини системи

Як зазначалося раніше, система була розроблена з використанням патерну репозиторію, де за клієнтську частину відповідає технологія створення веб-додатків ASP.NetMVC. ASP.Net – це технологія для розробки веб-застосунків, яка включає написання сторінок мовою HTML з використанням динамічного коду на мові C#. Крім того, ця технологія включає мову Razor, що дозволяє швидко інтегрувати серверний код у розмітку, значно прискорюючи розробку додатків. Для відображення інформації були розроблені сторінки з підключенням CSS (таблиць стилів) та JavaScript.

У веб-додатку буде декілька сторінок, а саме:

– головна сторінка;

– сторінка реєстрації;

– сторінка авторизації;

– особистий кабінет користувача, в свою чергу на цій сторінці є вкладки де розміщена інформація про допомогу, потреби та склад волонтерської допомоги;

– сторінка з детальною інформацію заявок на допомогу для водія.

На головній сторінці розміщена загальна інформація про волонтерську організацію, яка займається доправленням допомоги постарждалим. На головній сторінці розміщена панель навігаційних елементів, завдяки цим елементам користувач може перейти до реєстрації або авторизації на веб-ресурсі і в подальшому оформити заявку на допомогу або доєднатися до волонтерського руху в якості волонтера або стати водієм.

Головна сторінка організації представлена на рисунку 4.13.

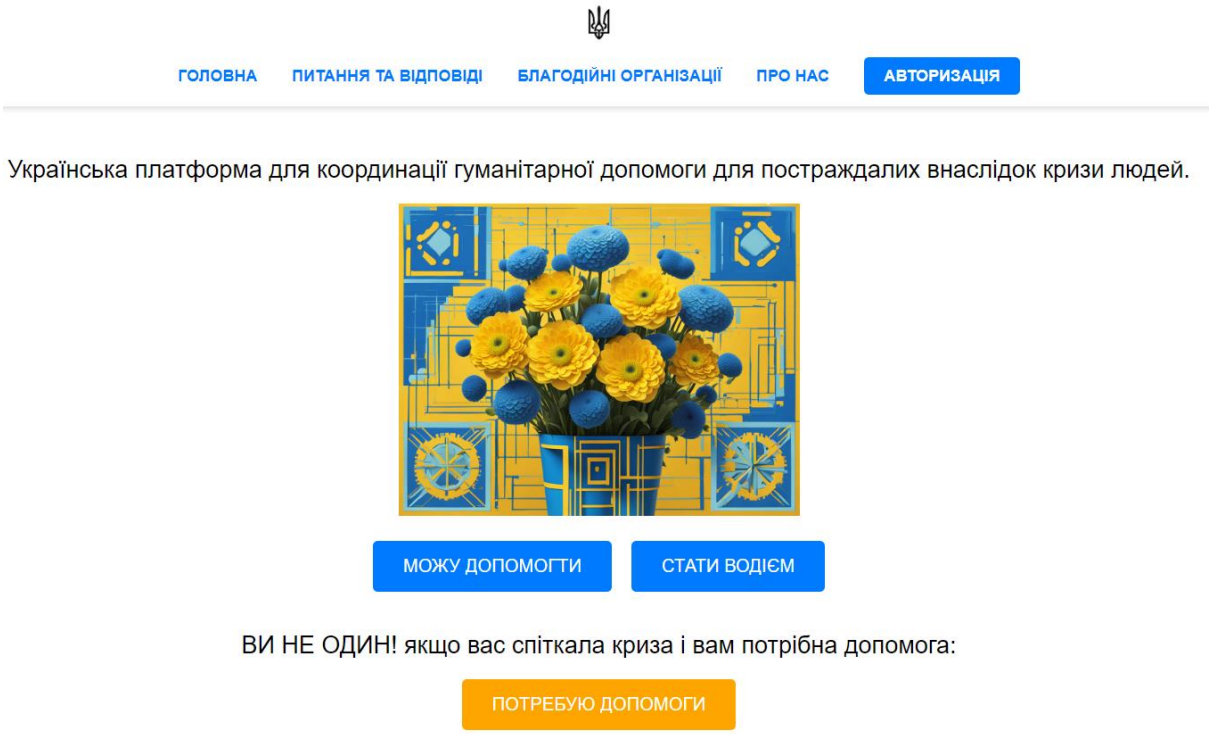


Рисунок 4.13 – Головна сторінка

Сторінка з реєстрацією має форму та поля для реєстрації користувача. Вона представлена на рисунку 4.14.

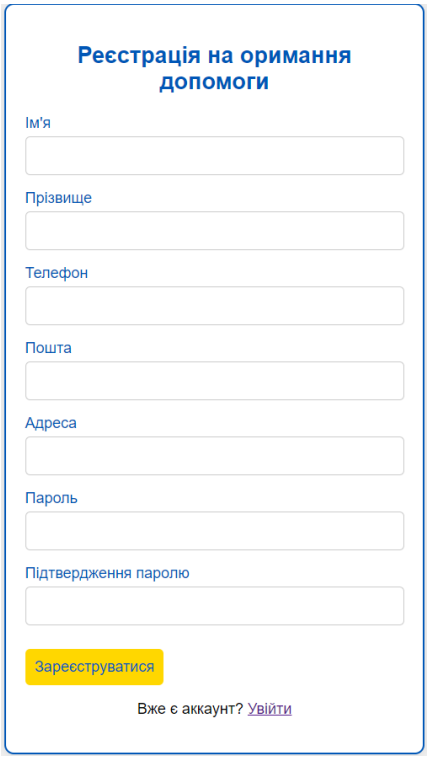


Рисунок 4.14 - Форма реєстрації

Сторінка для авторизації має два поля, це логін та пароль після введення та перевірки даних буде виконано вхід до системи.

В особистому кабінеті користувача розміщені такі сторінки: «Мої потреби», «Вибір допомоги», «Можу допомогти», «Мій склад», «Можу доставити».

Каталог допомоги можна переглянути на вкладці «Вибір допомоги». На цій вкладці людина яка потребує допомоги може обрати ті товари які їй потрібні. Також, на цій вкладці показано хто надає допомогу, в якій кількості та покриття доставки. Сторінка каталогу допомоги зображена на рисунку 4.15.

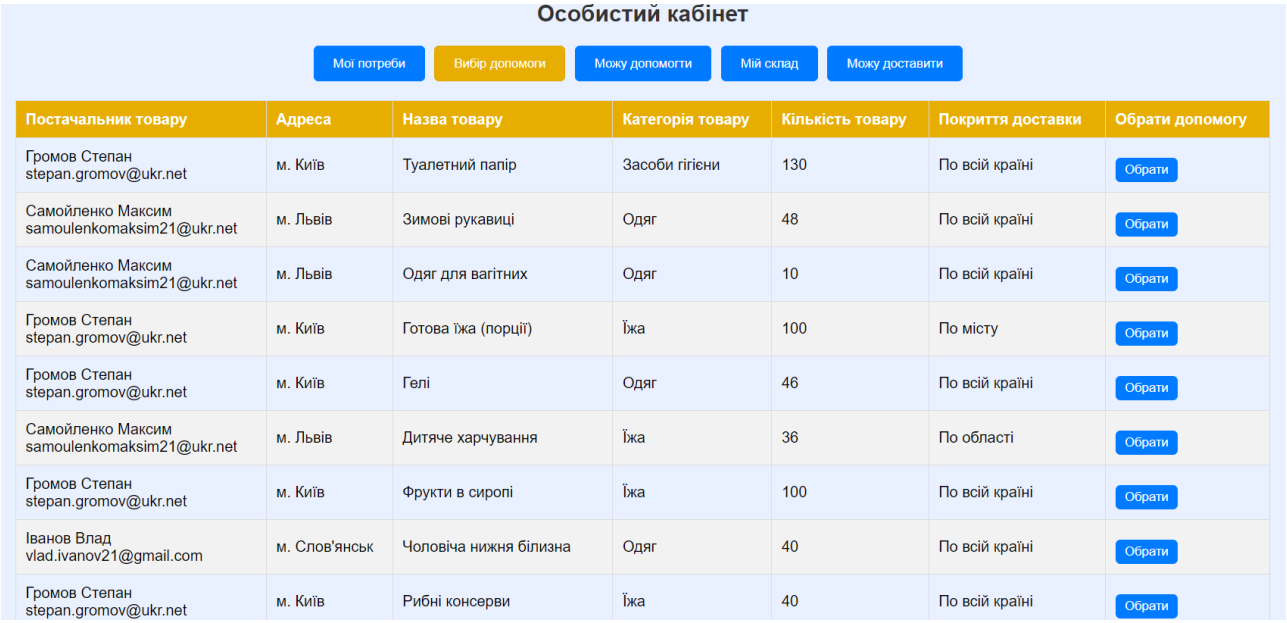
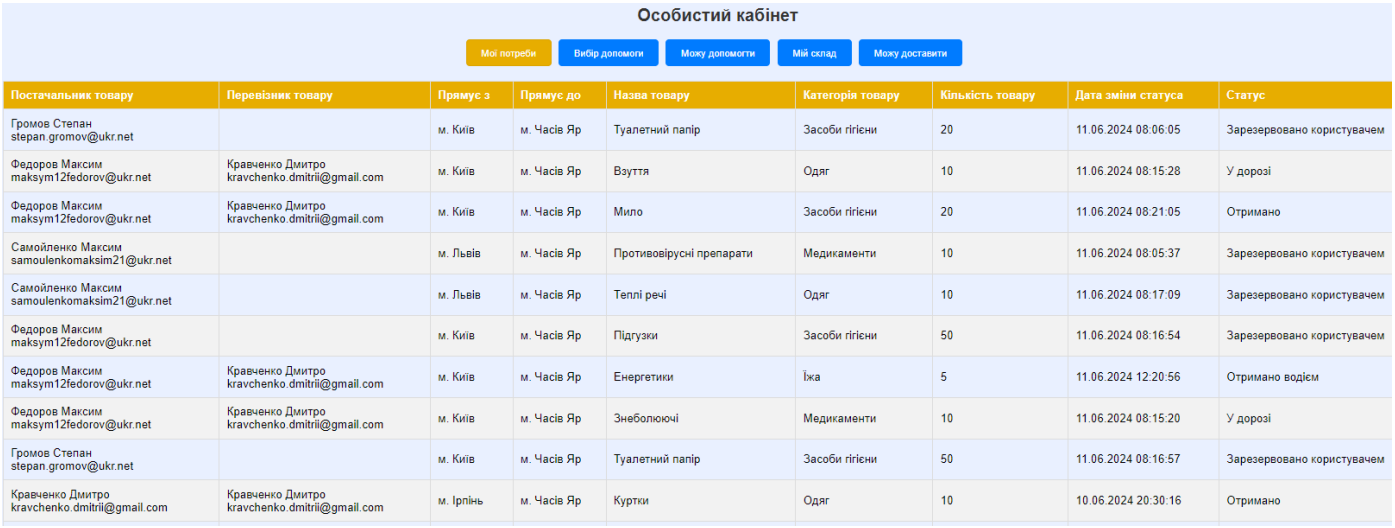


Рисунок 4.15 – Каталог допомоги

Після того як людина обрала товари, на вкладці «Мої потреби» вона може переглянути їх та перевірити дату заміни статуса та статус доставки. Сторінка каталогу потреб в допомозі зображена на рисунку 4.16.



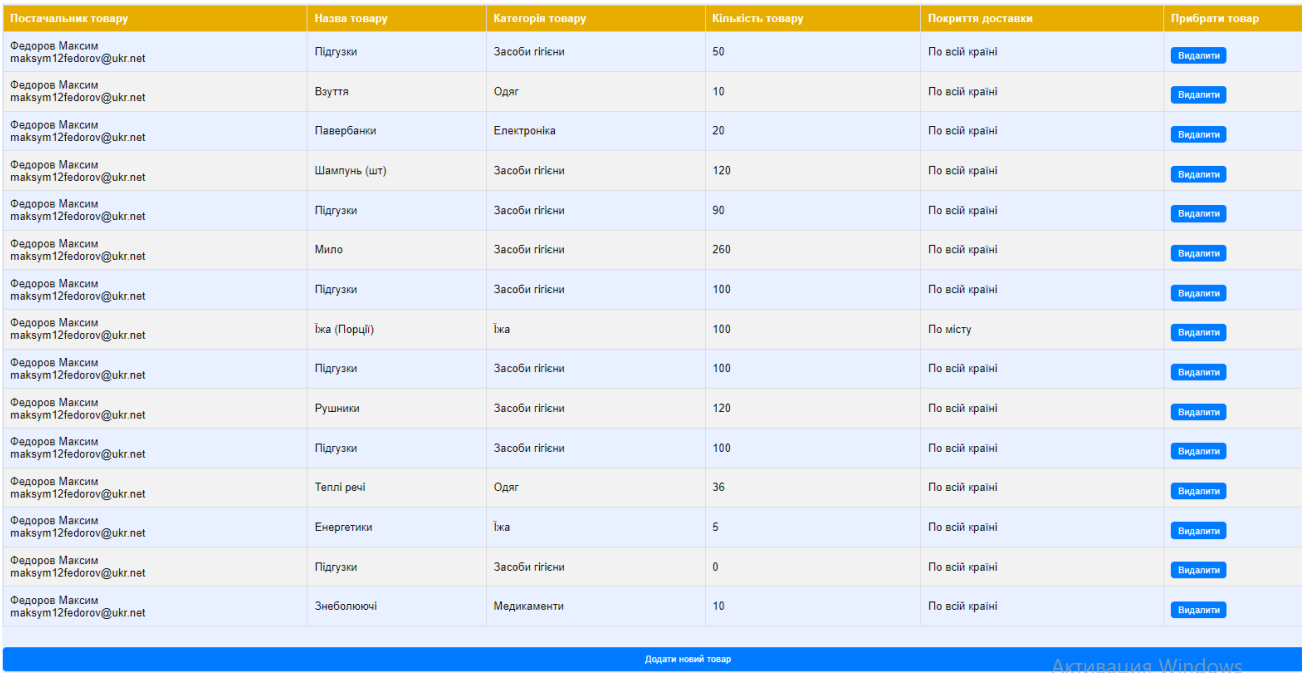
Особистий кабінет

Мої потреби | Вибір допомоги | Можу допомогти | Мій склад | Можу доставити

Постачальник товару	Перевізник товару	Прямує з	Прямує до	Назва товару	Категорія товару	Кількість товару	Дата зміни статуса	Статус
Громов Степан stepan.gromov@ukr.net		м. Київ	м. Часів Яр	Туалетний папір	Засоби гігієни	20	11.06.2024 08:06:05	Зарезервовано користувачем
Федоров Максим maksym12fedorov@ukr.net	Кравченко Дмитро kravchenko.dmitrii@gmail.com	м. Київ	м. Часів Яр	Взуття	Одяг	10	11.06.2024 08:15:28	У дорозі
Федоров Максим maksym12fedorov@ukr.net	Кравченко Дмитро kravchenko.dmitrii@gmail.com	м. Київ	м. Часів Яр	Мило	Засоби гігієни	20	11.06.2024 08:21:05	Отримано
Самойленко Максим samoulenkomaksim21@ukr.net		м. Львів	м. Часів Яр	Противовірусні препарати	Медикаменти	10	11.06.2024 08:05:37	Зарезервовано користувачем
Самойленко Максим samoulenkomaksim21@ukr.net		м. Львів	м. Часів Яр	Теплі речі	Одяг	10	11.06.2024 08:17:09	Зарезервовано користувачем
Федоров Максим maksym12fedorov@ukr.net		м. Київ	м. Часів Яр	Підгузки	Засоби гігієни	50	11.06.2024 08:16:54	Зарезервовано користувачем
Федоров Максим maksym12fedorov@ukr.net	Кравченко Дмитро kravchenko.dmitrii@gmail.com	м. Київ	м. Часів Яр	Енергетики	Їжа	5	11.06.2024 12:20:56	Отримано водієм
Федоров Максим maksym12fedorov@ukr.net	Кравченко Дмитро kravchenko.dmitrii@gmail.com	м. Київ	м. Часів Яр	Знеболюючі	Медикаменти	10	11.06.2024 08:15:20	У дорозі
Громов Степан stepan.gromov@ukr.net		м. Київ	м. Часів Яр	Туалетний папір	Засоби гігієни	50	11.06.2024 08:16:57	Зарезервовано користувачем
Кравченко Дмитро kravchenko.dmitrii@gmail.com	Кравченко Дмитро kravchenko.dmitrii@gmail.com	м. Ірпінь	м. Часів Яр	Куртки	Одяг	10	10.06.2024 20:30:16	Отримано

Рисунок 4.16 – Каталог потреб

В особистому кабінеті також є вкладка «Мій склад», вона створена для людей які хочуть доєднатися до волонтерської організації та надати допомогу. На цій вкладці волонтер може додати різні категорії товару та вказати їх кількість, після чого вони з’являться в особистому кабінеті людини яка потребує допомоги. Сторінка додавання товарів зображена на рисунку 4.17.



Постачальник товару	Назва товару	Категорія товару	Кількість товару	Покриття доставки	Прибрати товар
Федоров Максим maksym12fedorov@ukr.net	Підгузки	Засоби гігієни	50	По всій країні	Видалити
Федоров Максим maksym12fedorov@ukr.net	Взуття	Одяг	10	По всій країні	Видалити
Федоров Максим maksym12fedorov@ukr.net	Павербанки	Електроніка	20	По всій країні	Видалити
Федоров Максим maksym12fedorov@ukr.net	Шампунь (шт)	Засоби гігієни	120	По всій країні	Видалити
Федоров Максим maksym12fedorov@ukr.net	Підгузки	Засоби гігієни	90	По всій країні	Видалити
Федоров Максим maksym12fedorov@ukr.net	Мило	Засоби гігієни	260	По всій країні	Видалити
Федоров Максим maksym12fedorov@ukr.net	Підгузки	Засоби гігієни	100	По всій країні	Видалити
Федоров Максим maksym12fedorov@ukr.net	Їжа (Порції)	Їжа	100	По місту	Видалити
Федоров Максим maksym12fedorov@ukr.net	Підгузки	Засоби гігієни	100	По всій країні	Видалити
Федоров Максим maksym12fedorov@ukr.net	Рушники	Засоби гігієни	120	По всій країні	Видалити
Федоров Максим maksym12fedorov@ukr.net	Підгузки	Засоби гігієни	100	По всій країні	Видалити
Федоров Максим maksym12fedorov@ukr.net	Теплі речі	Одяг	36	По всій країні	Видалити
Федоров Максим maksym12fedorov@ukr.net	Енергетики	Їжа	5	По всій країні	Видалити
Федоров Максим maksym12fedorov@ukr.net	Підгузки	Засоби гігієни	0	По всій країні	Видалити
Федоров Максим maksym12fedorov@ukr.net	Знеболюючі	Медикаменти	10	По всій країні	Видалити

Додати новий товар

Рисунок 4.17 – Сторінка додавання допомоги

Сторінка «Можу доставити» створена для водіїв-волонтерів, які в свою чергу доправляють гуманітарну допомогу до потерпілих. Для того щоб стати водієм в організації, потрібно зареєструватися як водій та вказати свій автомобіль. Сторінка реєстрації водіїв зображена на рисунку 4.18.

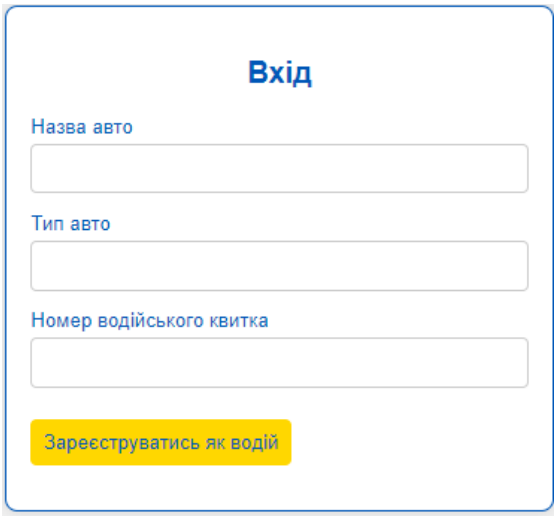
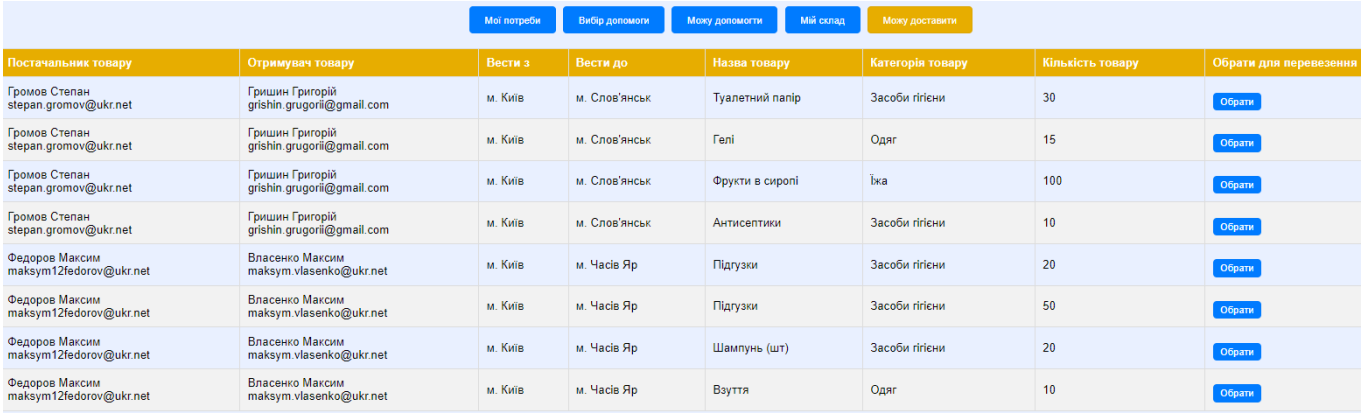


Рисунок 4.18 – Сторінка реєстрації водіїв

Після реєстрації водій у вкладці «Можу доставити» обирає доступні для нього доставки, які він зможе виконати, це зображено на рисунку 4.19.



Постачальник товару	Отримувач товару	Вести з	Вести до	Назва товару	Категорія товару	Кількість товару	Обрати для перевезення
Громов Степан stepan.gromov@ukr.net	Гришин Григорій grishin.grugorii@gmail.com	м. Київ	м. Слов'янськ	Туалетний папір	Засоби гігієни	30	Обрати
Громов Степан stepan.gromov@ukr.net	Гришин Григорій grishin.grugorii@gmail.com	м. Київ	м. Слов'янськ	Гелі	Одяг	15	Обрати
Громов Степан stepan.gromov@ukr.net	Гришин Григорій grishin.grugorii@gmail.com	м. Київ	м. Слов'янськ	Фрукти в сиропі	Їжа	100	Обрати
Громов Степан stepan.gromov@ukr.net	Гришин Григорій grishin.grugorii@gmail.com	м. Київ	м. Слов'янськ	Антисептики	Засоби гігієни	10	Обрати
Федоров Максим maksym12fedorov@ukr.net	Власенко Максим maksym.vlasenko@ukr.net	м. Київ	м. Часів Яр	Підгузки	Засоби гігієни	20	Обрати
Федоров Максим maksym12fedorov@ukr.net	Власенко Максим maksym.vlasenko@ukr.net	м. Київ	м. Часів Яр	Підгузки	Засоби гігієни	50	Обрати
Федоров Максим maksym12fedorov@ukr.net	Власенко Максим maksym.vlasenko@ukr.net	м. Київ	м. Часів Яр	Шампунь (шт)	Засоби гігієни	20	Обрати
Федоров Максим maksym12fedorov@ukr.net	Власенко Максим maksym.vlasenko@ukr.net	м. Київ	м. Часів Яр	Взуття	Одяг	10	Обрати

Рисунок 4.19 – Сторінка водія-волонтера

На сторінці «Можу допомогти» зображена вся інформація з надання гуманітарної допомоги, а саме: постачальник, перевізник, отримувач, адреса відправки, адреса отримувача, назва товару, категорія до якої належить товар,

кількість, покриття доставки, дата, статус заявки. Сторінка обробки заявок на допомогу зображена на рисунку 4.20.

Рисунок 4.20 – Сторінка обробки допомоги

Детальний опис взаємодії користувачів з інтерфейсом можна знайти у посібнику користувача.

4.6 Тестування розробленого програмного забезпечення

В результаті проведеного тестування програмного забезпечення було виявлено, що система відповідає висунутим вимогам, а усі системи працюють справно.

Таким чином дана система має наступний працюючий функціонал:

- Реєстрація / вхід.
- Реєстрація водіїв.
- Отримання даних про тип та кількість допомоги.
- Оформлення заявки на допомогу.
- Перегляд своїх заявок на допомогу.
- Додавання нових типів допомоги.
- Зміна статусу заявок на допомогу.

ВИСНОВКИ

В ході роботи було проаналізовано предметну область з моніторингу та координації гуманітарної допомоги, проведено аналіз вже існуючих програмних забезпечень для отримання гуманітарної допомоги. У розробленій системі було описано та автоматизовано основний бізнес-процес – моніторинг та координація гуманітарної допомоги. Користувачі системи поділяються на три типи: користувачі (потребують допомоги), волонтери (надають допомогу) та водії-волонтери (доставляють гуманітарну допомогу) кожен з яких певним чином включений в процес моніторингу та координації за певним алгоритмом та володіє своїм набором функціональних можливостей.

Користувач може подавати запити на отримання допомоги та перегляд статусу вже існуючих заявок. Волонтери можуть надавати гуманітарну допомогу та розподіляти її між користувачами. Водії в свою чергу доправляють допомогу до місця її потреби.

В результаті роботи були побудовані такі діаграми:

- Функціональна модель IDEF0.
- Use Case діаграма.
- Діаграма послідовностей.
- Діаграма класів.

Також була побудована організаційна структура та сруктурна схема складу сервісів, був зроблений інтерфес користувачів.

Таким чином дана інформаційна система є придатною до використання в предметній області з моніторингу та координації гуманітарної допомоги.

Розроблена система пройшла апробацію, взявши участь у 7-й міжнародній студентській науковій конференції «Сучасні аспекти та перспективні напрямки розвитку науки». В рамках цієї участі було опубліковано тези доповіді на тему «Інформаційна система для моніторингу та координації гуманітарної допомоги».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мартін Ф. С. UML основи / Фаулер Сеймс Мартін., 2004. – 356 с. – (3).

2. Веллінг Л. MySQL. Навчальний посібник.- К: Люк Веллінг. – Київ: Зоря, 2007. – 214 с. – (1). – (1).

3. Благороб – 2023 – URL: https://blagorob.com.ua (дата звернення 13.05.2024)

4. ZayavaInfo – 2024 – URL: https://zayava.info/vydacha-produktovyh-naboriv-u-kharkiv (дата звернення 13.05.2024)

5. єДопомога – 2022 – URL: https://social.edopomoga.gov.ua/en/auth/sign-in (дата звернення 14.05.2024)

6. ДСТ 34.003-90 - Автоматизовані системи. Терміни й відділення.

7. РД 50-680-88 - Керівний документ по стандартизації. Методичні вказівки. Автоматизовані системи. Основні положення.

8. Бондаренко М.Ф., Соловйова Є.А. Моделювання та проетування бізнес-систем: методи, стандарти, технології. - Харьків: СМІТ, 2004. - 272с

9. Томашевський О.М. та ін.. Інформаційні технології та моделювання бізнес процесів. Навчальний посібник.- К.: Видавництво «Центр учбової літератури», 2012. – 296с.

10. Monolithic vs. microservices architecture – 2024 – URL: https://www.outsystems.com/blog/posts/monoliths-or-microservices-make-both-your-domain (дата звернення 20.05.2024)

11. Поняття хмарних сервісів та їх основні типи – 2024 – URL: https://www.techopedia.com/definition/29017/cloud- services (дата звернення 22.05.2024)

12. Борисенко В.П., Левікін В.М., Пономарьов Ю.А. Об’єктно-орієнтований аналіз та проектування ІУС на основі UML. – Харків: ХНУРЕ, 2004. – 80 с.

13. Lambert M. Surhone, Miriam T. Timpledon, Susan F. Marseken.

Sequence Diagram, 2010. – 128 с.

14. Діаграми відношень сутностей (ERD) – 2019 – URL: https://lucidchart.zendesk.com/hc/ru/articles/207299756-Діаграми-відношень-сутностей-ERD- (дата звернення 24.05.2024)

15. UML Sequence diagram – 2021 – URL: https://flexberry.github.io/ru/fd_sequencediagram.htmlhttps://www.outsystems.com/blog/posts/monoliths-or-microservices-make-both- your-domain (дата звернення 25.05.2024)

16. Петрова Р.В., Морозова А.І. Іноваційний реінжинірінг бізнес-процесів в інформаційному середовищі // Вісник Хмельницького національного університету. Серія: Технічні науки. –2019, № 1 (269). – С. 211−215.

17. Томка Ю.Я., Ушенко Ю.О. Основи ASP.NET MVC. Навчальний посібник.- К.: / Томка Ю.Я., Ушенко Ю.О. – Чернівці, Чернівецький нац. ун-т, 2018. – 730 с.

18. Gerardus Blokdyk. Microsoft Sql Server Management Studio a clear and concise reference, 2021. – 312 с.

19. Настройка моделі в ASP.NET Core Identity – 2021 – URL: https://learn.microsoft.com/ /aspnet/core/security/authentication/customize-identity-model?view=aspnetcore-8.0 (дата звернення 29.05.2024)

20. Jon P. Smith. Entity Framework Core in Action, 2021. – 624 с.

21. Haris Tsetsekas. Object-Oriented Programming Exercises with C#, 2023. – 171 с.

22. Alan Beaulieu. Learning SQL: Generate, Manipulate, and Retrieve Data, 2020. – 342 с.

23. Joe Reis. Matt Housley. Fundamentals of Data Engineering, 2022. – 743 с.

24. Mathias Wesk. Business Process Management, 2007. – 426 с.

25. Daniel Shiffman. The Nature of Code: Simulating Natural Systems with Processing, 2012. – 520с.