Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

за спеціальності 121 – Інженерія програмного забезпечення

На тему: Розробка сервісу електронної черги для вантажних терміналів аграрних компаній (елеваторів)

Засвідчую, що в цій кваліфікаційній роботі немає
запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-20-2
_____/І.В.Недвига/

Керівник
кваліфікаційної роботи _____ /Н.Н.Шаповалова/

Завідувач кафедри _____ /А.М.Стрюк/

Кривий Ріг
2024

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: бакалавр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А.М.Стрюк

«__» _____ 20__р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ-20-2 Недвигі Івану Вікторовичу

1. Тема: Розробка сервісу електронної черги для вантажних терміналів аграрних компаній (елеваторів). Затверджено наказом по КНУ №__ від «__» _____ 2024р.
2. Термін подання студентом закінченої роботи : «__» _____ 2024р.
3. Вихідні дані по роботі: обсяг вантажопотоку щоденно, середню тривалість очікування автомобілів, та кількість та розміщення вантажних точок на терміналі.
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): Зміст пояснювальної записки передбачає визначення потреб користувачів та їх очікувань від електронної черги, аналіз поточних проблем у чергуванні на вантажних терміналах, розробку функціональних можливостей системи, встановлення алгоритму управління вантажопотоком, визначення технічних вимог до програмного забезпечення та оцінку ефективності впровадження системи електронної черги в аграрних компаніях.
5. Перелік ілюстративного матеріалу: знімки екранну, блок-схеми, ілюстрації алгоритмів роботи.

Календарний план**:**

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Збір даних та аналіз теми	29.01.2024 – 04.02.2024
2	Аналіз існуючих аналогів	07.02.2024 – 09.02.2024
3	Підготовка матеріалів першого розділу	09.02.2024 – 12.02.2024
4	Створення блок-схем та проектування логіки програмної частини проекту	18.02.2024 – 26.02.2024
5	Аналіз та розробка користувацького інтерфейсу	05.03.2024 – 07.03.2024
6	Підготовка матеріалів другого розділу	08.03.2024 – 23.03.2024
7	Підготовка матеріалів третього розділу	24.03.2024 – 28.03.2024
8	Реалізація програмної частини	01.04.2024 – 23.04.2024
9	Створення пояснювальної записки	27.04.2024 – 23.05.2024

Дата видачі завдання: «28» січня 2024р.

Студент _____/ І.В.Недвига

Керівник роботи _____/Н.Н.Шаповалова

РЕФЕРАТ

ЕЛЕКТРОННА ЧЕРГА, ВАНТАЖНІ ТЕРМІНАЛИ, АГРАРНІ КОМПАНІЇ, ОПТИМІЗАЦІЯ ПРОЦЕСІВ, АВТОМАТИЗАЦІЯ, УПРАВЛІННЯ ВАНТАЖОПОТОКОМ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ТЕХНІЧНІ ВИМОГИ, ЕФЕКТИВНІСТЬ, ІНТЕГРАЦІЯ СИСТЕМ.

Пояснювальна записка: 34 с., 12 рис., 1 дод., 11 джерел.

У сучасному швидкому темпі життя, де ефективність та оперативність стають ключовими факторами успіху, оптимізація вантажних терміналів аграрних компаній є надзвичайно актуальною проблемою. Необхідність ефективного управління вантажопотоком та чергуванням у таких терміналах стає важливим завданням для забезпечення неперервного та швидкого обігу товарів. Довгі черги, затримки та неефективне управління часом можуть впливати на конкурентоспроможність підприємств.

Метою цієї роботи є розробка та обґрунтування концепції електронної черги для вантажних терміналів аграрних компаній. Ця кваліфікаційна робота спрямована на автоматизацію та оптимізацію процесів чергування вантажів, зменшення часу очікування для клієнтів та підвищення загальної продуктивності терміналу.

У ході дослідження буде проведений аналіз поточного стану чергування на вантажних терміналах аграрних компаній, включаючи вивчення проблемних аспектів та недоліків існуючих систем. На основі отриманих даних буде розроблений концепт електронної черги, враховуючи потреби користувачів та особливості вантажопотоку.

ABSTRACT

ELECTRONIC QUEUE, CARGO TERMINALS, AGRICULTURAL COMPANIES, PROCESS OPTIMIZATION, AUTOMATION, CARGO FLOW MANAGEMENT, SOFTWARE, TECHNICAL REQUIREMENTS, EFFICIENCY, SYSTEM INTEGRATION.

Thesis in 34 p., 0 table, 12 figure, 1 appendix, 11 source.

In today's fast-paced life, where efficiency and effectiveness are becoming key success factors, optimization of cargo terminals of agricultural companies is an extremely important issue. The need to efficiently manage cargo flow and duty in such terminals is becoming an important task to ensure continuous and fast turnover of goods. Long queues, delays, and inefficient time management can affect the competitiveness of enterprises.

The purpose of this work is to develop and justify the concept of an electronic queue for cargo terminals of agricultural companies. This qualification work is aimed at automating and optimizing cargo queuing processes, reducing waiting time for customers, and increasing overall terminal productivity.

The study will analyze the current state of queuing at the cargo terminals of agricultural companies, including the study of problematic aspects and shortcomings of existing systems. Based on the data obtained, the concept of an electronic queue will be developed, taking into account the needs of users and the peculiarities of cargo flow.

ЗМІСТ

ВСТУП

В епоху динамічного розвитку інформаційної сфери, електронні сервіси стають невід'ємною частиною повсякденного життя, оптимізуючи та вдосконалюючи роботу в багатьох галузях. Сільське господарство, як один із ключових секторів економіки, також не оминуло цю тенденцію. Елеватори – вантажні термінали аграрних компаній – відіграють важливу роль у логістичних ланцюгах постачання зерна та олійних культур. Їхня ефективна робота сприяє безперебійному функціонуванню всієї галузі.

Однією з проблем, з якими стикаються елеватори, є черги вантажних автомобілів, що очікують на завантаження або розвантаження. Це призводить до втрат часу, ресурсів та, як наслідок, до зниження прибутковості. Використання традиційних методів управління чергами, таких як ведення паперових журналів та ручне розподіл черги, часто виявляється неефективним та схильним до людської помилки.

Впровадження сервісу електронної черги для вантажних терміналів аграрних компаній може стати дієвим рішенням цієї проблеми. Такий сервіс дозволить оптимізувати процес управління чергою, значно скоротивши час очікування та підвищивши пропускну здатність елеватора. Ця дипломна робота присвячена розробці сервісу електронної черги для вантажних терміналів аграрних компаній.

В рамках роботи буде проведено аналіз існуючих систем управління чергами, досліджені методи та технології, що використовуються для їх створення, а також розроблена власна модель сервісу електронної черги з урахуванням специфіки роботи елеваторів.

1 СУЧАСНИЙ СТАН І АКТУАЛЬНІСТЬ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.1 Актуальність теми кваліфікаційної роботи

Актуальність теми "Розробка сервісу електронної черги для вантажних терміналів аграрних компаній" обумовлена наступними важливими факторами:

1. Зростання ролі електронних сервісів в оптимізації роботи аграрних компаній.

В сучасному світі електронні сервіси стають невід'ємною частиною повсякденного життя, проникаючи в усі сфери, включаючи агросектор. Використання електронних рішень дозволяє оптимізувати багато рутинних процесів, економити час та ресурси, а також підвищувати загальну ефективність роботи. Елеватори, як вантажні термінали аграрних компаній, також не оминули цю тенденцію. Впровадження електронних сервісів на елеваторах може значно покращити їхню роботу, зробити її більш прозорою та зручною для клієнтів.

2. Необхідність вдосконалення методів управління чергами на вантажних терміналах елеваторів.

Однією з проблем, з якими стикаються елеватори, є черги вантажних автомобілів, що очікують на завантаження або розвантаження. Ці черги призводять до втрат часу, ресурсів та, як наслідок, до зниження прибутковості. Традиційні методи управління чергами, такі як ведення паперових журналів та ручне розподіл черги, часто виявляються неефективними та схильними до людської помилки.

3. Відсутність на ринку комплексних рішень для електронної черги, що відповідають специфіці роботи елеваторів.

На даний момент на ринку існують різні системи електронної черги, але вони не завжди відповідають специфічним потребам елеваторів. Ці системи часто розроблені для загального використання і не враховують особливості

роботи елеваторів, такі як різні типи вантажів, що приймаються, динаміка завантаження та розвантаження, необхідність пріоритетизації певних категорій вантажних автомобілів тощо.

4. Зростання обсягів перевезень зерна та олійних культур.

В останні роки спостерігається значне зростання обсягів перевезень зерна та олійних культур. Це призводить до збільшення навантаження на елеватори та, як наслідок, до збільшення часу очікування вантажних автомобілів. Впровадження сервісу електронної черги може допомогти впоратися з цим зростаючим навантаженням та забезпечити більш ефективне обслуговування клієнтів.

5. Підвищення вимог до рівня обслуговування клієнтів.

Сучасні аграрні компанії прагнуть до того, щоб надавати своїм клієнтам максимально якісний сервіс. Це включає в себе не тільки якість продукції, але й зручність та швидкість обслуговування. Впровадження сервісу електронної черги може значно покращити рівень обслуговування клієнтів на елеваторах, зробити його більш прозорим та зручним.

1.2 Цілі та завдання кваліфікаційної роботи

Метою цієї кваліфікаційної роботи є розробка сервісу електронної черги для вантажних терміналів аграрних компаній, який дозволить оптимізувати процес управління чергою, значно скоротивши час очікування та підвищивши пропускну здатність елеватора.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Визначити функціональні вимоги до сервісу електронної черги.
2. Розробити архітектуру сервісу, використовуючи DDD-підхід.
3. Розробити модуль реєстрації та авторизації користувачів.
4. Спроектувати та реалізувати інтерфейс користувача сервісу.

В результаті виконання цієї кваліфікаційної роботи очікується отримати:

1. Детально продуману та чітко описану модель сервісу електронної черги, що включає в себе всі описані вище компоненти та їхні характеристики.
2. Функціонуючий сервіс електронної черги, розроблений на основі моделі та впроваджений на елеваторі.
3. Протестований сервіс, з оціненою працездатністю та ефективністю.
4. Дипломну роботу, де буде описано процес дослідження, розробки та впровадження сервісу, а також представлені результати дослідження.
5. Презентацію, яка візуально та інформативно представляє ключові результати кваліфікаційної роботи.

1.3 Критичний аналіз літературних джерел за темою

Проаналізувавши значну кількість літературних джерел, зокрема наукових статей та публікацій, я можу зробити наступні висновки щодо поточної ситуації з розробкою сервісу електронної черги для вантажних терміналів аграрних компаній:

Актуальність розробки сервісу електронної черги:

- Зростання обсягів вантажоперевезень: Зростання обсягів виробництва та експорту сільгосппродукції призводить до збільшення кількості вантажних автомобілів, які очікують на обслуговування на елеваторах [5].
- Неефективність існуючих систем управління чергою: Традиційні системи управління чергою, засновані на ручному записі та веденні журналів, не справляються з зростаючим потоком вантажних автомобілів [6].
- Економічні втрати від простою: Час очікування в черзі на елеваторі призводить до значних економічних втрат для аграрних компаній та перевізників [7].

Переваги впровадження сервісу електронної черги:

- *Зменшення часу очікування:* Автоматизація та оптимізація процесу управління чергою дозволить значно скоротити час очікування вантажних автомобілів [8].
- *Підвищення пропускної здатності елеватора:* Ефективне розподілення черги та моніторинг її стану дозволять збільшити кількість обслуговуваних вантажних автомобілів за одиницю часу [9].
- *Економія коштів:* Зменшення простою вантажних автомобілів та оптимізація роботи елеватора призведуть до значної економії коштів для аграрних компаній [10].
- *Покращення логістики:* Електронна черга дозволить оптимізувати маршрути перевезень та планувати завантаження елеватора, що покращить загальну логістику [11].

2 РОЗРОБКА ФУНКЦІОНАЛЬНОЇ СХЕМИ І АЛГОРИТМУ

2.1 Розробка та докладний опис функціональної схеми програми

Функціональна схема програми Echerga описує зв'язки між її компонентами та чітко окреслює їх організацію в логічні блоки або модулі. Ця схема слугує ключем до розуміння функціоналу програми, демонструючи порядок виконання дій та взаємодію між її частинами.

Функціональна схема програми описує головний функціонал ПЗ та складається з наступних компонентів(пунктів):

Функціональна схема Echerga складається з чотирьох основних компонентів:

1. Закрита веб-частина Адміністратора: Цей компонент призначений для адміністрування системи та надає доступ до наступних функцій:
   - Реєстрація та авторизація адміністраторів
   - Управління користувачами (водіями)
   - Управління елеваторами
   - Перегляд та аналіз даних
   - Налаштування системи
2. Закрита веб-частина Перевізника*:* Цей компонент призначений для управління роботою перевізників та надає доступ до наступних функцій:
   - Реєстрація та авторизація перевізників
   - Запис у чергу
   - Зворотний зв'язок
3. Закрита веб-частина Диспетчера: Цей компонент призначений для моніторингу та управління чергами на елеваторах, а також для координації роботи водіїв та перевізників. Він надає доступ до наступних функцій:
   - Моніторинг статусу черг
   - Управління чергами
   - Надсилання повідомлень водіям та перевізникам
   - Перегляд історії записів

- Зворотний зв'язок

4. Відкрита частина для Водіїв: Цей компонент призначений для водіїв та надає доступ до наступних функцій:

- Реєстрація та авторизація водіїв
- Запис у чергу
- Перегляд статусу черги
- Перегляд історії записів
- Зворотний зв'язок

Структурна система взаємодії сутностей функціональних компонентів описаних раніше - зображена на рисунку 2.1.

Рисунок 2.1 – Взаємодія об’єктів функціональних компонентів

Компоненти функціональної схеми Echerga взаємодіють один з одним за допомогою наступних механізмів:

1. API: Кожен компонент має API, який використовується для доступу до його функцій та даних.

Echerga використовує локальний API для взаємодії між компонентами. Цей API базується на бібліотеці Mediator, яка полегшує синхронну та асинхронну комунікацію між компонентами.

1. Структура API:

- Кожен компонент має свій API: Кожен компонент має набір методів, які описують його функціональність. Ці методи доступні іншим компонентам через API.
- Типізація: API використовує систему типізації для забезпечення безпеки та надійності. Це гарантує, що методи API використовуються правильно і що дані передаються у відповідному форматі.
- Документація: API добре документований, що полегшує його використання іншими розробниками.

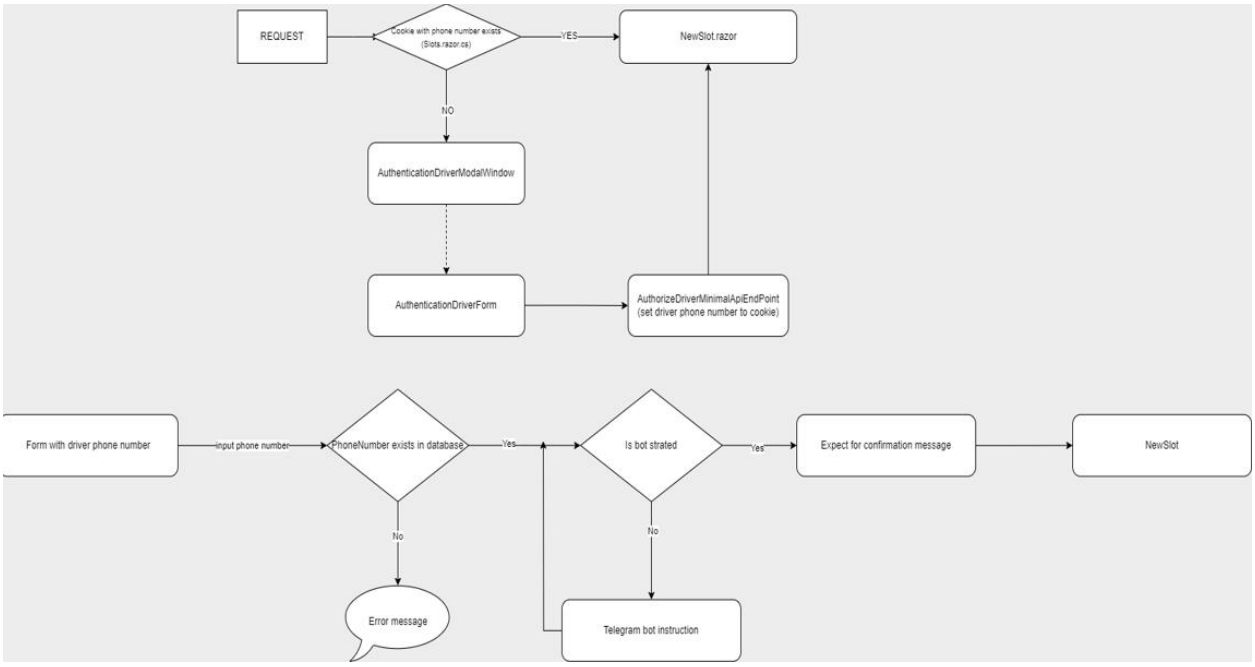

Рисунок 2.2 – Головний процесс основного компоненту відповідального за реєстрації слоту у черзі

2.2 Розробка та докладний опис алгоритму роботи програми

У сучасному світі, де час є цінним ресурсом, оптимізація будь-яких процесів стає ключовим фактором успіху. Це твердження особливо актуальне для сфери аграрного бізнесу, де чітка координація та ефективне управління логістичними операціями є запорукою рентабельності та конкурентоспроможності.

Електронна черга для вантажних терміналів аграрних компаній - це інноваційне рішення, яке покликане значно спростити та оптимізувати процес бронювання слотів у черзі. Ця система ґрунтується на динамічному алгоритмі вирахування вільних слотів, що забезпечує прозорість, зручність та гнучкість для користувачів.

Даний розділ присвячений детальному опису алгоритму роботи програми, розкриваючи його ключові етапи та принципи функціонування.

Основний акцент буде зроблено на динамічному вирахуванні вільних слотів у черзі, адже саме цей аспект є основоположним для проекту "Ечерга". Завдяки динамічному підходу, система гарантує користувачам актуальну інформацію про доступні слоти, що дає їм можливість планувати свої дії та економити час.

Ознайомлення з алгоритмом роботи програми допоможе краще зрозуміти принципи її функціонування та оцінити її переваги для оптимізації логістичних процесів у аграрній сфері.

Алгоритм бронювання слотів:

1. Завантаження інформації про операцію:
   - Отримання ID операції з запиту.
   - Завантаження інформації про операцію з бази даних, включаючи робочі періоди.
   - Перевірка, чи існують робочі періоди для обраної операції та чи обраний часовий діапазон не є поточним моментом.
2. Завантаження зарезервованих слотів:

- Отримання списку зарезервованих слотів для обраної операції та часового діапазону.
- Сортування зарезервованих слотів за датою прибуття та порядком прибуття.
- Перетворення списку зарезервованих слотів у формат DTO.

3. Додавання зарезервованих слотів до результату:
   - Додавання списку зарезервованих слотів до результату.
4. Перевірка наявності вільних слотів:
   - Перевірка наявності робочих періодів для обраної операції.
   - Перевірка, чи обраний часовий діапазон не є поточним моментом.
5. Ітерація по всіх можливих датах прибуття:
   - Перебір всіх можливих дат прибуття в обраному часовому діапазоні.
   - Перевірка, чи дата прибуття не є поточним моментом.
6. Підрахунок кількості зарезервованих слотів:
   - Визначення кількості зарезервованих слотів для поточної дати прибуття.
7. Перевірка наявності вільних слотів:
   - Порівняння кількості зарезервованих слотів з максимальною кількістю пунктів обслуговування для даної операції.
8. Додавання вільних слотів до результату:
   - Якщо кількість зарезервованих слотів менша за максимальну кількість пунктів обслуговування, додавання до результату вільних слотів для поточної дати прибуття.
9. Повернення результату:
   - Повернення результату, який містить інформацію про зарезервовані та вільні слоти.

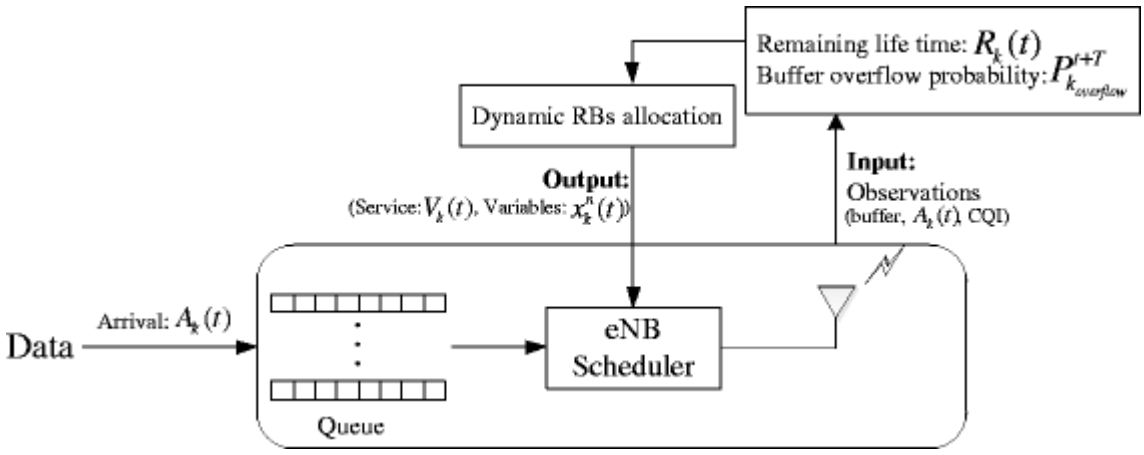

Рисунок 2.3 – Основний процес роботи алгоритму бронювання слотів у черзі

2.3 Розробка інтерфейсу програми

2.3.1 Головний інтерфейс сторінки "Базиси"

Сторінка "Базиси" є головним інтерфейсом користувача, де представлені всі доступні базиси для бронювання слотів у черзі.

Інтерфейс сторінки повинен включати:

- Список доступних базисів:
  - Перелік всіх активних базисів з можливістю сортування та фільтрації за різними критеріями (наприклад, за назвою, типом операції, часом роботи).
  - Для кожного базису коротка інформація, така як назва, опис, поточний стан черги (кількість вільних та зайнятих слотів).
- Можливість перегляду черги:
  - Детальний опис черги для обраного базису, включаючи список зарезервованих та вільних слотів.
  - Інформація про кожен слот у черзі (час прибуття, номер транспортного засобу, статус).
  - Можливість оновлення інформації про чергу в режимі реального часу.

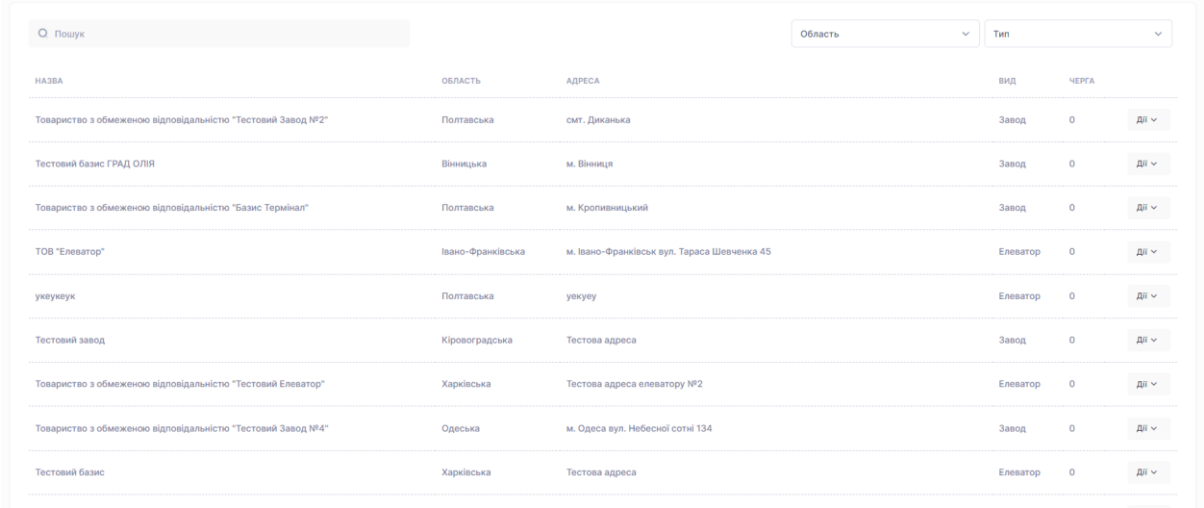

Рисунок 2.4 – Головний інтерфейс сторінки "Базиси"

2.3.2 Сторінка "Загальна черга базису"

Сторінка "Загальна черга базису" призначена для відображення загальної черги для обраного базису.

Інтерфейс сторінки включає в себе наступну інформацію:

- Список зарезервованих слотів:
  - Детальний список всіх зарезервованих слотів у черзі для обраного базису.
  - Для кожного слоту інформація про час прибуття, номер транспортного засобу, статус (у дорозі/прибув), тип вантажу, назву операції (розвантаження/навантаження).
  - Можливість сортування та фільтрації слотів за різними критеріями (наприклад, за часом прибуття, статусом, типом вантажу, назвою операції).
- Пошук:
  - Функція пошуку, що дозволяє користувачам знаходити слоти за номером транспортного засобу, часом прибуття або іншими критеріями.
- Фільтрація:
  - Можливість фільтрувати слоти за типом вантажу та назвою операції (розвантаження/навантаження).
- Оновлення інформації:

○ Механізм оновлення інформації про чергу в режимі реального часу.

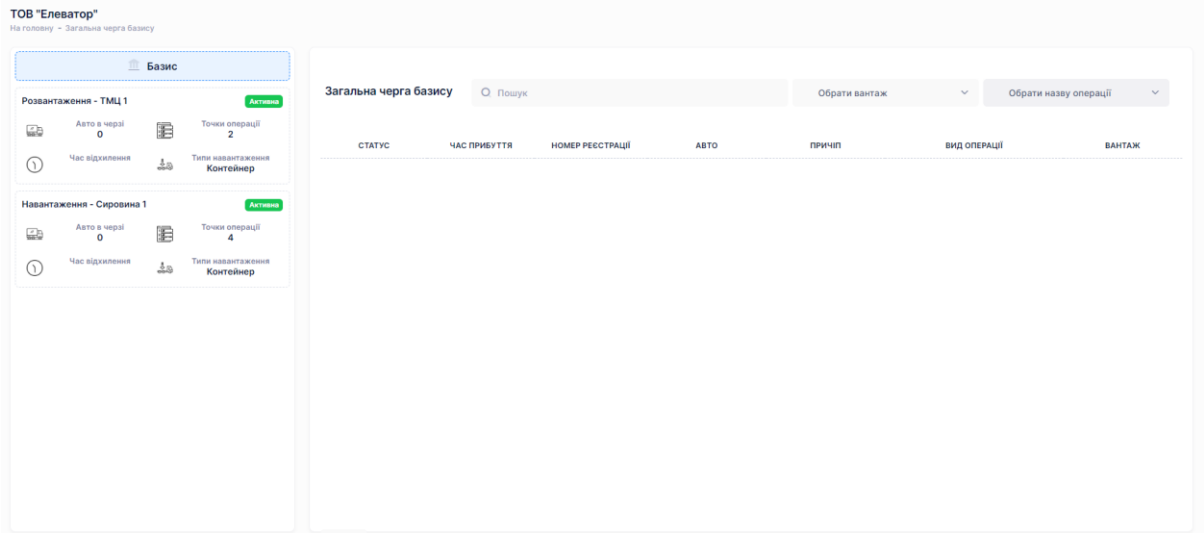

Рисунок 2.5 – Сторінка "Загальна черга базиси"

2.3.3 Сторінка "Черга операції"

Інтерфейс сторінки включає в себе наступну інформацію:

- Календар:

○ Інтерактивний календар, який дозволяє користувачам вибирати будь-який день місяця.

○ Для кожного дня календаря інформація про вільні та зарезервовані слоти для обраної операції (розвантаження/навантаження).

○ Візуальне представлення кількості вільних та зарезервованих слотів (наприклад, за допомогою кольорів або графіків).

○ Можливість переходу до сторінки загальної черги базису для обраного дня.

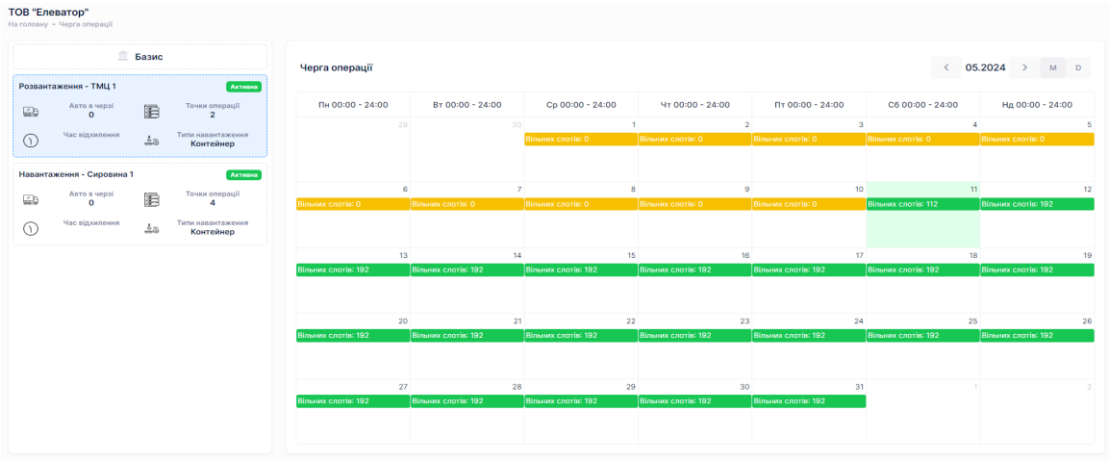

Рисунок 2.6 – Сторінка "Загальна черга операції"

2.3.4 Спливаюче вікно "Аутентифікації водія під час реєстрації у черзі"

Аутентифікація водія відбувається у два етапи:

1 - Підтвердження номеру телефону у системі(тобто спочатку водій повинен зареєструватися в системі як водій за обраним перевізником, щоб успішно пройти 1-ий етап аутентифікації)

2 - Підтвердження номеру телефону через телеграм(водій повинен бути присутнім у нашому телеграм боті задля того, щоб ми йому змогли відправити повідомлення з можливістю “Підтвердити” або “Скасувати” процесс підтвердження).

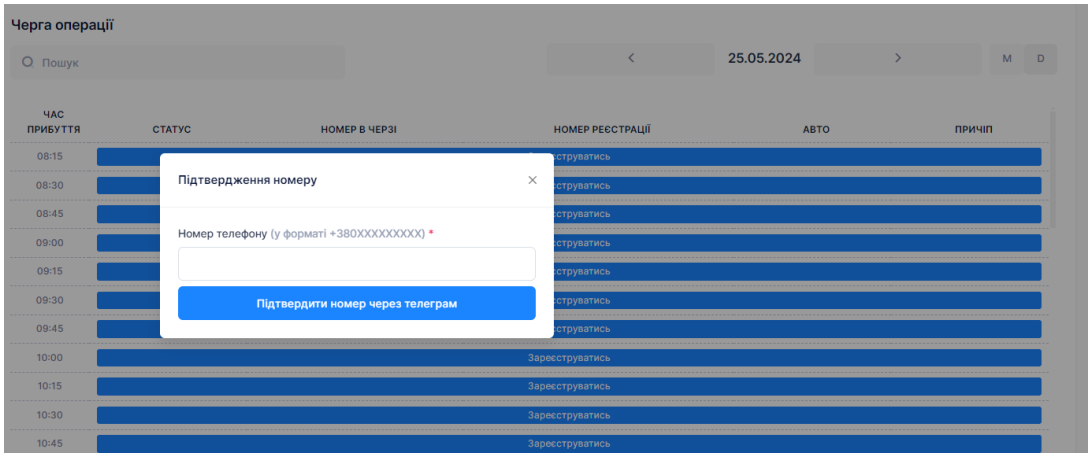

Рисунок 2.7 – Попап № 1 "Підтвердження номеру телефону у системі"

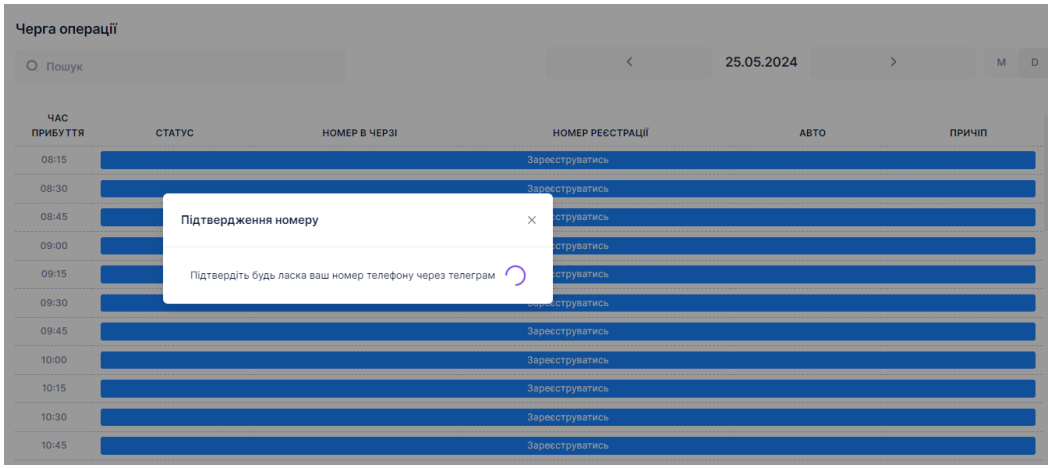

Рисунок 2.8 – Попап № 2 "Підтвердження номеру телефону у телеграмі"

2.3.5 Спливаюче вікно "Реєстрації у черзі"

Після успішного проходження 2-х факторного етапу аутентифікації, - водія буде відображений наступне модальне вікно “Реєстрація у черзі”, де водію необхідно виконати наступні дії задля успішної реєстрації у черзі:

1 - Обрати перевізника за яким водій буде ставати у чергу(нагадаю що один водій можу бути закріплений за декількома перевізниками).

2 - Обов’язково вибрати вантажний автомобіль з селекту “Автомобіль”.

3 - За потребності вибрати причіп з селекту “Причіп”.

4 - Вказати назву та код вантажовідправника.

5 - Завантажити ТТН.

6 - Після виконання всіх вище описаний етапів - натиснути на кнопку “Зареєструватися у черзі”.

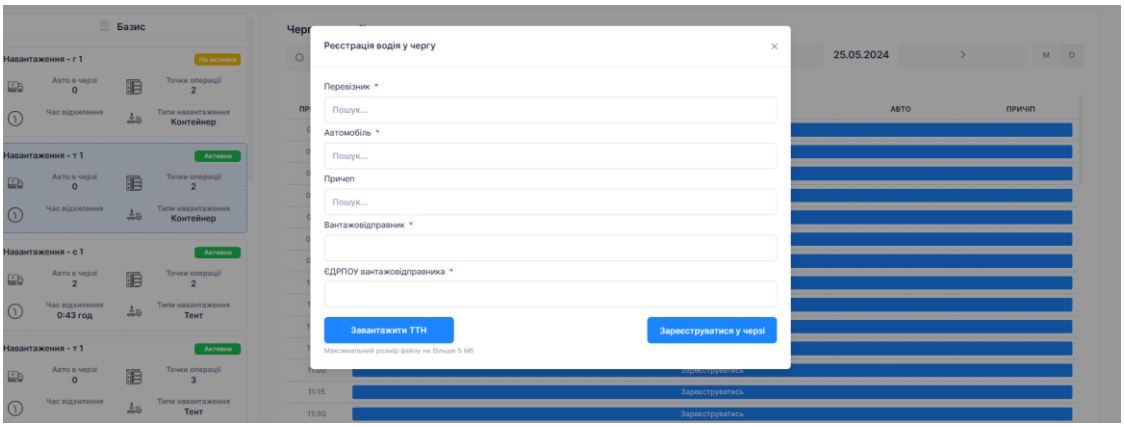

Рисунок 2.9 – Попап "Реєстрація у черзі"

3 РОЗРОБКА БАЗИ ДАНИХ І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз обраної середи програмування

Даний проект написаний на платформі .NET, а саме на актуальній версії 8.0, яка надає розробникам широкий спектр можливостей для швидкої та оптимізованої розробки додатків. У якості технологій був вибраний наступний стек:

1. BlazorServer

BlazorServer - це фреймворк для створення інтерактивних веб-додатків за допомогою C# замість традиційного JavaScript. Він працює на сервері, обробляючи події та оновлюючи інтерфейс користувача через SignalR, що дозволяє створювати високоефективні додатки з мінімальною затримкою.

2. Serilog

Serilog - це бібліотека для ведення логів у .NET, яка дозволяє записувати структуровані логи в різні сховища, такі як файли, бази даних, хмарні сервіси тощо. Вона підтримує гнучку конфігурацію і розширюваність, що робить її потужним інструментом для моніторингу та відлагодження додатків.

3. Entity Framework Core (EF Core)

EF Core - це об’єктно-реляційний маппер (ORM) для .NET, який дозволяє розробникам працювати з базами даних, використовуючи об’єктно-орієнтований підхід. Він підтримує міграції, забезпечує гнучку конфігурацію моделей та полегшує доступ до даних.

4. Mediator

Mediator - це бібліотека, яка реалізує паттерн медіатора для .NET. Вона спрощує обмін повідомленнями між різними компонентами системи, що допомагає зменшити зв’язність коду і покращує його модульність та підтримуваність.

5. AutoMapper

AutoMapper - це бібліотека для автоматичного мапінгу об’єктів. Вона дозволяє швидко перетворювати об’єкти одного типу на інші, зменшуючи рутинний код і помилки, пов’язані з ручним мапінгом. Це особливо корисно при роботі з DTO та моделями домену.

3.2 Розробка бази даних

В якості бази даних для цього проекту було обрано PostgreSQL. Це потужна, надійна та відкрита реляційна база даних, яка має ряд переваг:

- Висока продуктивність: PostgreSQL забезпечує високу продуктивність при обробці великих обсягів даних і складних запитів завдяки оптимізації плану виконання запитів, використанню індексів і паралельному виконанню запитів.
- Розширюваність: Система підтримує створення власних типів даних, функцій, операторів і індексів, що дозволяє адаптувати базу даних під специфічні потреби додатку.
- Відповідність стандартам: PostgreSQL підтримує більшість SQL стандартів, що забезпечує сумісність і переносимість додатків.
- Надійність та безпека: Вбудовані механізми резервного копіювання, відновлення даних, транзакційність (ACID) і розширені функції безпеки роблять PostgreSQL дуже надійною базою даних.
- Велика спільнота та підтримка: Активна спільнота розробників і користувачів, а також велика кількість документації та ресурсів для підтримки і розвитку проектів.

База даних містить одну схему public (паблік) з наступними основними таблицями:

- Користувачі (Users):

Таблиця зберігає інформацію про користувачів системи, включаючи їх ідентифікатори, імена, електронні адреси, ролі та інші атрибути, необхідні для управління користувачами.

- Базиси (Basises):

Таблиця містить дані про базиси - основні точки або центри, які використовуються в системі. Це можуть бути склади, логістичні центри або інші важливі об’єкти.

- Перевізники (Carriers):

Таблиця зберігає інформацію про перевізників, які здійснюють транспортні послуги. Вона включає дані про назву компанії, контактну інформацію та інші важливі відомості.

- Водії (Drivers):

Таблиця містить дані про водіїв, які працюють з перевізниками. Тут зберігається інформація про імена водіїв, контактні дані, ліцензії та інші характеристики.

- Слоти (Slots):

Таблиця використовується для управління слотами, тобто часовими інтервалами або резервуванням ресурсів. Вона включає дані про час початку та закінчення слотів, доступність та пов'язані ресурси.

- Транспортні засоби (AutoVehicles):

Таблиця зберігає інформацію про транспортні засоби, що використовуються в системі. Включає дані про марки та моделі автомобілів, реєстраційні номери, вантажопідйомність та інші характеристики.

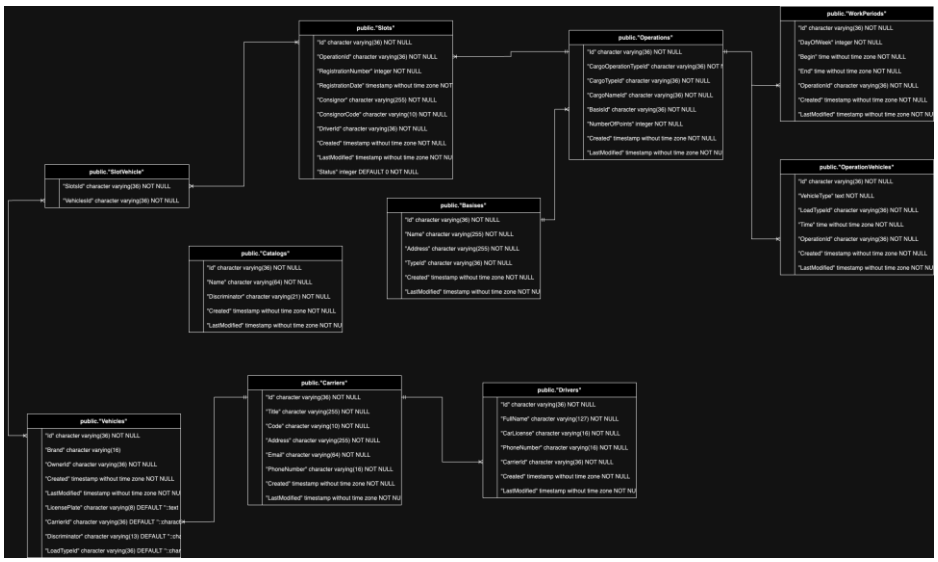

Рисунок 3.1 – Структура таблиць публічної схеми(public schema)

3.3 Програмна реалізація основних функцій

Програмна реалізація основних функцій проекту виконана за допомогою бібліотеки Mediator, яка реалізує патерн CQRS (Command Query Responsibility Segregation). Використання Mediator у порівнянні з традиційними сервісами має кілька важливих переваг.

Переваги використання бібліотеки Mediator

- Зниження зв'язності коду:

Використання патерну медіатора дозволяє значно знизити зв'язність коду. Замість того, щоб кожен компонент безпосередньо взаємодіяв з іншими компонентами або сервісами, усі команди і запити проходять через медіатор. Це спрощує підтримку і модифікацію коду, оскільки зміни в одному компоненті не вимагають змін в інших.

- Покращення масштабованості:

Патерн CQRS, на якому базується Mediator, розділяє операції на команди (commands) і запити (queries). Це дозволяє більш гнучко масштабувати різні частини системи відповідно до їх навантаження. Наприклад, можна окремо оптимізувати обробку команд і запитів, використовуючи різні техніки кешування або балансування навантаження.

- Чітке розмежування відповідальності:

CQRS чітко розділяє читання даних (queries) від їх модифікації (commands), що спрощує розуміння і підтримку коду. Кожен обробник запиту чи команди має одну конкретну відповідальність, що робить систему більш зрозумілою і легкою для тестування.

- Полегшення тестування:

Завдяки чіткому розмежуванню команд і запитів, кожен обробник можна тестувати окремо, що полегшує написання модульних тестів і знижує вартість підтримки коду. Медіатор дозволяє ізолювати логіку обробки запитів від інших частин системи, що зменшує залежності і спрощує тестування.

- Гнучкість і розширюваність:

Mediator забезпечує високу гнучкість і розширюваність системи. Додавання нових функцій або змінення існуючих вимагає мінімальних змін у коді. Нові команди або запити можна легко додати, просто створивши нові обробники, не змінюючи існуючу інфраструктуру.

- Покращення підтримуваності:

Оскільки Mediator організовує код у невеликі, ізольовані обробники, він значно покращує підтримуваність коду. Кожен обробник фокусується на виконанні однієї конкретної задачі, що робить код більш модульним і зрозумілим для розробників.

- Реалізація основних функцій з використанням Mediator

У нашому проекті, основні функції реалізовані шляхом створення окремих команд і запитів для кожної операції. Наприклад, створення нового користувача реалізовано через команду CreateUserCommand, яка обробляється відповідним обробником CreateUserCommandHandler. Для отримання списку користувачів використовується запит GetUsersQuery, який обробляється обробником GetUsersQueryHandler.

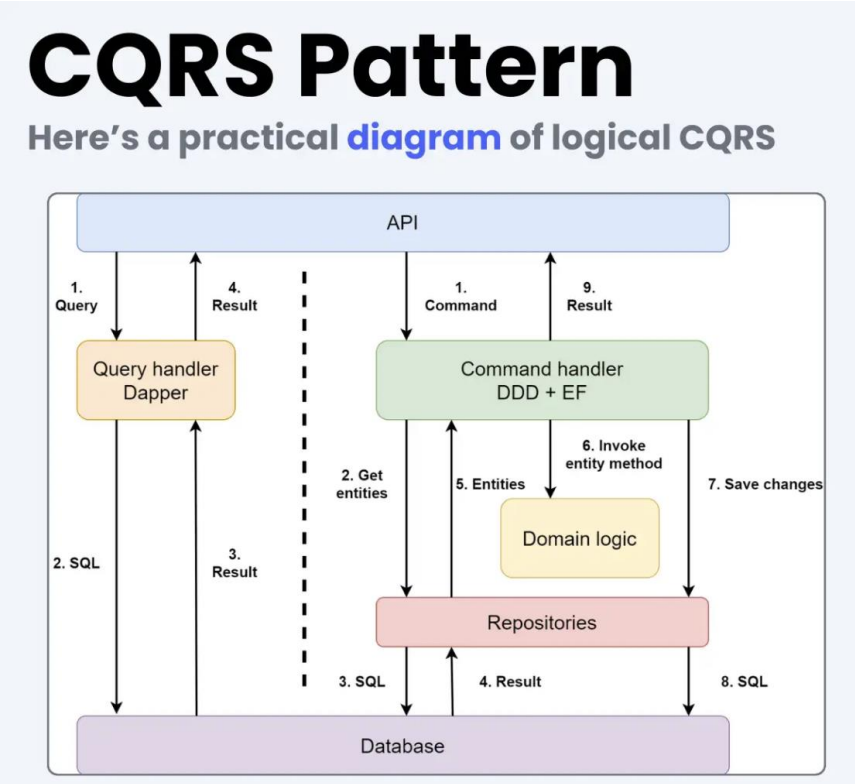

Рисунок 3.2 – Схема роботи CQRS патерну на основі Mediator

Отже, це підхід дозволяє нашій системі легко розширювати функціональність системи, додаючи нові команди і запити, не змінюючи існуючий код. Також це спрощує діагностику і усунення несправностей, оскільки кожна команда і запит обробляються окремими, добре визначеними обробниками.

3.4 Методика роботи користувача

Основна методика роботи користувача в нашій системі реалізована через сутність "Водій". Процес роботи водія складається з наступних кроків:

- Вхід на веб-частину системи:

Коли водій заходить на відкриту веб-частину системи, він має змогу обрати базис (логістичний центр або інший важливий об'єкт), куди він планує приїхати і стати в чергу.

- Вибір базису:

Водій обирає конкретний базис зі списку доступних, на який йому потрібно приїхати. Це дозволяє системі зрозуміти, куди саме спрямовується водій.

- Вибір операції:

Після вибору базису водій обирає операцію, яку він хоче виконати (наприклад, завантаження, розвантаження тощо). Це допомагає визначити, яку саме дію водій планує здійснити на вибраному базисі.

- Інтерактивний календар:

Після вибору операції водієві відкривається інтерактивний календар. У цьому календарі відображаються всі вільні та зайняті слоти на обраний день. Це дозволяє водієві проаналізувати доступні варіанти і вибрати зручний для себе час.

- Вибір вільного слоту:

Водій вибирає зручний вільний слот у календарі. Це резервує час для його приїзду та виконання обраної операції на базисі.

- Процес аутентифікації та реєстрації:

Після вибору слоту водій проходить процеси аутентифікації та реєстрації, які були детально описані раніше. Це включає введення особистих даних, підтвердження контактної інформації та інші необхідні кроки для верифікації користувача.

- Підтвердження бронювання:

Після виконання всіх дій система отримує всю необхідну інформацію про новий зареєстрований слот. Водій отримує підтвердження про успішне бронювання слоту, що завершує процес реєстрації.

- Переваги даної методики:

Простота і зручність:

Інтерфейс системи забезпечує інтуїтивно зрозумілу навігацію, що полегшує водіям вибір базису, операції та слоту.

- Ефективне управління часом:

Інтерактивний календар дозволяє водіям бачити вільні і зайняті слоти в реальному часі, що допомагає ефективно планувати свій робочий день і мінімізувати час очікування.

- Автоматизація процесів:

Система автоматизує процеси аутентифікації, реєстрації та бронювання, що знижує кількість ручної роботи і підвищує точність даних.

- Покращена комунікація:

Водії отримують миттєві підтвердження про свої бронювання, що забезпечує прозорість і покращує комунікацію між водіями і адміністрацією базисів.

Ця методика роботи користувача забезпечує високу ефективність і зручність використання системи для водіїв, покращуючи організацію їх роботи та знижуючи час на обробку даних.

ВИСНОВОК

Виконуючи цю кваліфікаційну роботу, я отримав значний досвід і знання у розробці програмного забезпечення, зокрема в області створення сучасних веб-додатків на платформі .NET. Робота над проектом дозволила мені поглибити свої знання та навички в наступних напрямках:

- Розробка на платформі .NET:

Я досконало освоїв основи платформи .NET, включаючи останню версію 8.0, та застосування її переваг для створення продуктивних та оптимізованих додатків. Зокрема, я навчився використовувати BlazorServer для побудови інтерактивних веб-інтерфейсів.

- Бази даних:

Вибір і використання PostgreSQL як основної бази даних дозволив мені на практиці освоїти роботу з цією потужною системою управління базами даних. Я навчився проектувати схему бази даних, створювати і оптимізувати таблиці, а також використовувати функції та індекси для підвищення продуктивності.

- Патерн CQRS та бібліотека Mediator:

Застосування патерну CQRS і бібліотеки Mediator стало важливою частиною мого навчання. Я зрозумів, як розділяти операції на команди та запити для покращення масштабованості та підтримуваності системи, а також отримав практичний досвід у впровадженні цього патерну.

- Інтерактивний інтерфейс користувача:

Створення інтерактивного календаря для вибору слотів допомогло мені розвинути навички роботи з користувацьким інтерфейсом та динамічним відображенням даних. Це також покращило моє розуміння UX/UI дизайну, орієнтованого на зручність користувача.

- Автентифікація та безпека:

Я отримав цінний досвід у реалізації механізмів автентифікації та безпеки користувачів, що включає верифікацію даних, роботу з особистою інформацією та забезпечення захищеного доступу до системи.

- Логування та моніторинг:

Використання бібліотеки Serilog для ведення логів навчило мене основам структурованого логування та моніторингу додатків, що є критично важливим для відлагодження і підтримки програмного забезпечення.

- Робота в команді та управління проектами:

Виконання цієї роботи дало мені можливість вдосконалити навички роботи в команді, планування та управління проектами. Я зрозумів важливість комунікації, розподілу задач та контролю за виконанням завдань для успішного завершення проекту.

У підсумку, виконання цієї кваліфікаційної роботи стало для мене важливим етапом у професійному розвитку. Я навчився застосовувати сучасні технології та підходи у розробці програмного забезпечення, що дозволяє мені впевнено рухатись далі у кар'єрі програміста.

ПЕРЕЛІК ПОСИЛАНЬ

1. Designing and Implementing Queue Management Systems. https://q-net.pro/?gad_source=1&gclid=CjwKCAjwrvyxBhAbEiwAEg_KgsjDTB_53X4IwtwOMaJ0myxPwayIey0ZTlH6bn_KAGphMRwZ5t_3iRoCO_oQAvD_BwE

2. The Art of Queueing Theory.https://www.cse.fau.edu/~bob/publications/encyclopedia.pdf

3. Design and Analysis of a Dynamic Queueing System for Real-Time Data Processing. https://www.jstor.org/stable/2628412

4. Herlihy, M., Shavit, N., Luchangco, V., & Spear, M. (2019). Priority queues. *The Art of Multiprocessor Programming*. https://doi.org/10.1214/aoms/1177705990.

5. Haviv, M., & Oz, B. (2018). Self-Regulation of an Unobservable Queue. *Manag. Sci.*, 64, 2380-2389. https://doi.org/10.1287/mnsc.2017.2728.

6. Hu, L. (2015). A Docking System of Supply and Demand for Agricultural Products Based on Intelligent Mobile Terminal. *Journal of Huizhou University*.

7. Renyuan, D., & Yang, L. (2008). The Researches of Agile Logistics System of Fresh Agricultural Products Based on E-commerce. *2008 International Conference on Information Management, Innovation Management and Industrial Engineering*, 2, 520-523. https://doi.org/10.1109/ICIII.2008.53.

8. Yang, H., Xiong, S., Frimpong, S., & Zhang, M. (2020). A Consortium Blockchain-Based Agricultural Machinery Scheduling System. *Sensors (Basel, Switzerland)*, 20. https://doi.org/10.3390/s20092643.

9. Freeman, A. (2020). Using Blazor Server, Part 1. , 865-892. https://doi.org/10.1007/978-1-4842-5440-0_33.

10. Kozak, K., & Smołka, J. (2020). Analysis of the Blazor framework in client-hosted mode. , 16, 269-273. https://doi.org/10.35784/JCSI.2019.

11. Taipalus, T. (2019). Teaching Tip: A Notation for Planning SQL Queries. *J. Inf. Syst. Educ.*, 30, 160-166.

ДОДАТОК А – РИСУНКИ

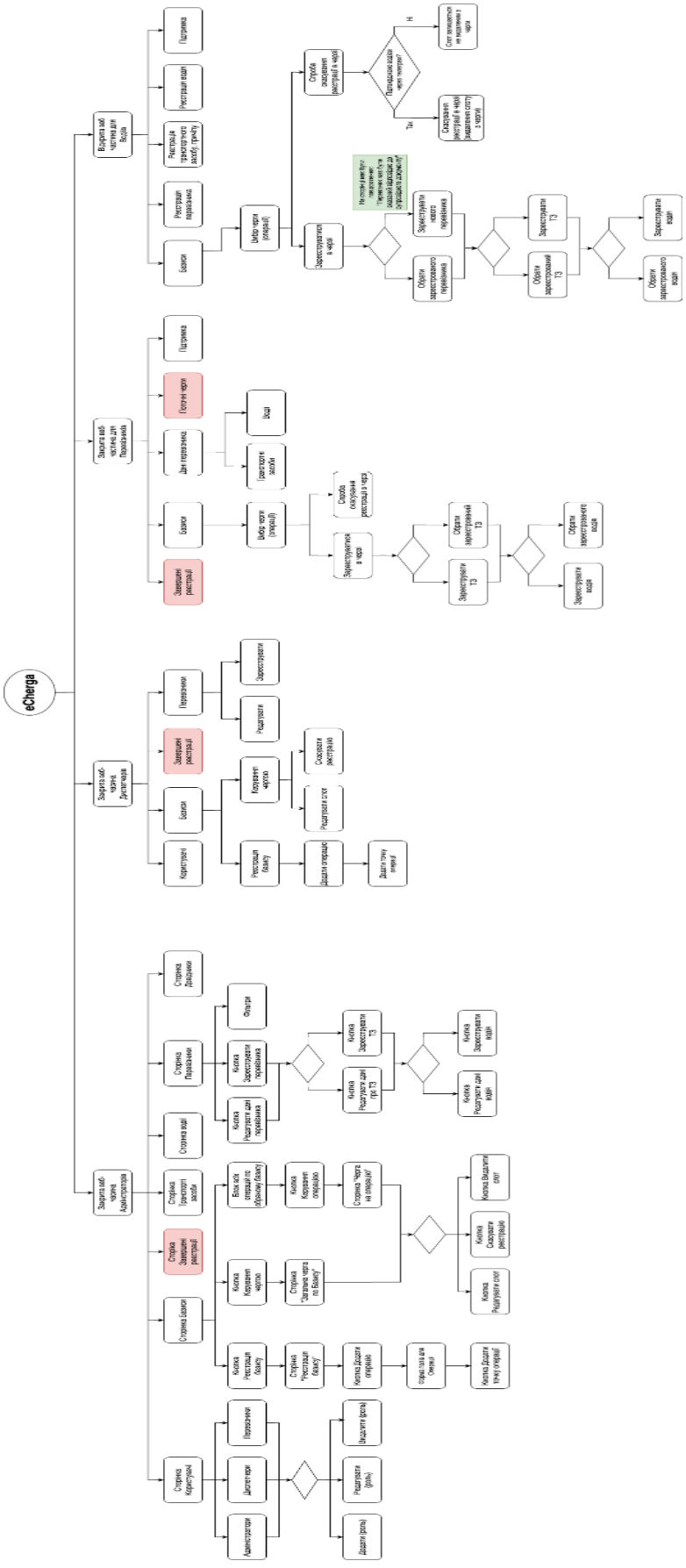

Рисунок 1 – Функціональна схема програми