Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка методів і програмних засобів CRM системи мобільного електротранспорту»

Виконав: студент II курсу
групи 2ПІ-22м спеціальності
121 – Інженерія програмного забезпечення
Гронюк Р.О.
Керівник: д.т.н., професор каф. ПЗ
Ліщинська Л.Б.
«15» грудня 2023 р.
Опонент: к.т.н, доцент каф. ЗІ
Дудатьєв А.В.
«15» грудня 2023 р.

Допущено до захисту
Завідувач кафедрою ПЗ
д.т.н., проф., Романюк О.Н.
«15» грудня 2023 р.

ВНТУ – 2023

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
_____Романюк О.Н.
«19 » вересня 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Гронюку Руслану Олександровичу

1. Тема роботи – розробка методів і програмних засобів CRM системи мобільного електротранспорту.

Керівник роботи: Ліщинська Людмила Броніславівна, д.т.н., проф. каф. ПЗ, затверджені наказом вищого навчального закладу від «18» вересня 2023 р. № 247.

2. Строк подання студентом роботи 5 грудня 2023 р.

3. Вихідні дані до роботи:
Операційна система – Windows, Mac OS.
Мови програмування – Java, SQL.
Технології – Spring boot, Maven, Thymeleaf, Git.
База даних PostgreSQL.
Середовище розробки IntelliJ IDEA (Ultimate Edition).

4. Зміст розрахунково-пояснювальної записки: аналіз сучасних методів і програмних засобів автоматизації відносин з клієнтами; моделювання автоматизації відносин з клієнтами та проектування програмного засобу crm системи мобільного електротранспорту; розробка програмного засобу crm системи мобільного електротранспорту; тестування програмного засобу crm системи мобільного електротранспорту.

5. Перелік ілюстративного матеріалу: схеми функціонування та організації CRM систем, діаграми проектування (класів, послідовності, активності та прецедентів), скріншоти процесів реалізації в середовищі

розробки та системах контролю версій, скріншоти створення та підключення бази даних, скріншоти тестування програмного забезпечення.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Ліщинська Л.Б., д.т.н., проф. кафедри ПЗ	19.09.23	01.12.23
5	Кавецький В.В., к.т.н., доц. кафедри ЕПВМ	25.11.23	01.12.23

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів	Примітка
1	Аналіз сучасних методів і програмних засобів автоматизації відносин з клієнтами	20.09.2023 – 30.09.2023	вик
2	Моделювання автоматизації відносин з клієнтами та проектування програмного засобу crm системи мобільного електротранспорту	30.09.2023 – 20.10.2023	вик
3	Розробка програмного засобу crm системи мобільного електротранспорту	20.10.2023 – 20.11.2023	вик
4	Тестування програмного засобу crm системи мобільного електротранспорту	20.11.2023 – 25.11.2023	вик
5	Економічна частина	25.11.2023 – 01.12.2023	вик

Студент _____ **Гронюк Р.О.**

(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Ліщинська Л.Б.**

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК. 004

Гронюк Р.О. Розробка методів і програмних засобів CRM системи мобільного електротранспорту. Магістерська кваліфікаційна робота зі спеціальності 121 - інженерія програмного забезпечення, освітня програма - програмна інженерія. Вінниця: ВНТУ, 2023. 138 с.

На укр.мові. Бібліогр.: 20 назв; рис. 27; табл. 16, к-сть стр. до додатків 95.

У магістерській кваліфікаційній роботі зроблено аналіз сучасних методів та засобів автоматизації відносин з клієнтами.

Запропоновано використати операційну модель для створення CRM системи мобільного електротранспорту. Розброблено методи та моделі для розрахунку необхідної потужності мобілного електротранспорту та енергитчної ємності акумуляторної батареї.

Розроблено архітектуру CRM системи мобільного електротранспорту, спроектовано структуру бази даних та основні модулі.

Для реалізації CRM системи мобільного електротранспорту було обрано мову програмування Java та середовище розробки IntelliJ IDEA (Ultimate Edition).

Проведено економічний розрахунок для визначення фінансової доцільності створення та впровадження методів і програмних засобів CRM системи мобільного електротранспорту.

Ключові слова: CRM система, автоматизація відносин з клієнтами, методи і програмні засоби CRM.

ABSTRACT

Hroniuk R.O. Development of CRM methods and software for the mobile electric transport system. Master's qualification thesis on specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2023. 138 p.

In the Ukrainian language. Bibliography: 20 titles; fig. 27; table 16.

In the master's qualification work, an analysis of modern methods and means of automating relations with clients was made.

It is proposed to use an operational model to create a CRM system of mobile electric transport. Methods and models for calculating the required power of mobile electric transport and the energy capacity of the battery are developed.

The CRM architecture of the mobile electric transport system was developed, the database structure and main modules were designed.

The Java programming language and the IntelliJ IDEA (Ultimate Edition) development environment were chosen for the implementation of the CRM system of mobile electric transport.

An economic calculation was carried out to determine the financial feasibility of creating and implementing CRM methods and software for the mobile electric transport system.

Keywords: CRM system, automation of customer relations, CRM methods and software tools.

Зміст

ВСТУП

Обґрунтування вибору теми дослідження. Абревіатура CRM походить від англійського Customer Relationship Management, що в перекладі означає «управління взаємо-відносинами з клієнтами». Звідси слідує, що CRM система – набір технологій, які дозволяють автоматизувати бізнес-процеси, які торкаються взаємодії з клієнтами.

CRM включає в себе ідеологію і технології створення історії взаємин клієнта і фірми, що дозволяє більш чітко планувати бізнес і підвищувати його стійкість. Зі зростанням бізнесу та масштабуванням продажів, взаємини з клієнтами все відчутніше впливатимуть на всі робочі процеси: від розробки продуктів до їх реалізації. Тому, чим раніше впровадити в діяльність підприємства CRM системи – тим швидше зможете надавати клієнтам ефективні рішення, раціонально управляти часом, оптимізувати бізнес-процеси, просувати бренд на ринку та, як наслідок, збільшувати прибуток підприємства.

Сучасні реалії, в яких сьогодні функціонує бізнес, свідчать про те, що успішними можуть бути лише ті компанії, які зуміли побудувати ефективні маркетингові комунікації, поставивши на перше місце інтереси споживача. Розуміння того факту, що клієнти – основа будь-якого бізнесу, заради чого починається діяльність та багато в чому залежить успіх та прибутковість компанії і є вже запорукою зростання ефективності та підняття на новий рівень.

Питання CRM-систем досліджувались багатьма вченими, зокрема Бутенко Н., Берестова Т., Вишлінський Г., Вишневська М., Гужва В.,Ганущак-Єфіменко Л. М., Гордієнко Д.О., Литвинова О. В., Лобань О. О., Островерхов В. М., Сьомкіна Т. В., Шпортько Г., Янчук Т.В. та інші [2].

Актуальність. В Україні CRM системи це великий та швидкозростаючий ринок програмного забезпечення, а тому розробка методів і програмних засобів CRM системи мобільного електротранспорту є актуальною для ведення ефективного та прибуткового бізнесу[3]. Але представлені в Україні CRM системи, в основному, спрямовані на великі

підприємства із готовими рішеннями, які важко адаптувати під потреби малого та середнього бізнесу та конкретний вид діяльності, а також підтримувати та розвивати фахівцями самого підприємства.

Тому актуальним є питання створення методів розрахунку потужності мобільного електротранспорту та розрахунку енергетичної ємності акумуляторної батареї, щоб покращити ефективність автоматизації відносин з клієнтами в галузі мобільного електротранспорту.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою досліджень є підвищення ефективності взаємовідносин підприємства з клієнтами.

Основними заданнями досліджень є:

- проаналізувати сучасні методи і програмні засоби автоматизації відносин з клієнтами;
- розробити метод та модель розрахунку потужності мобільного електротранспорту за вхідними даними клієнта та параметрами електротранспорту;
- розробити метод та модель розрахунку енергетичної ємності акумуляторної батареї за значенням потужності та бажаною відстанню пробігу на одному заряді батареї;
- розробити та протестувати програмний засіб CRM системи мобільного електротранспорту;
- розрахувати економічну ефективність.

Об'єктом дослідження цієї магістерської кваліфікаційної роботи є бізнес-процеси взаємовідносин підприємства з клієнтами.

Предмет досліджень – методи та засоби організації бізнес процесів взаємовідносин підприємства з клієнтами.

Методи дослідження. Було використано метод теорії систем, теорії інформаційних систем, а також аналізу, синтезу та алгоритмізації.

Наукова новизна отриманих результатів:

удосконалено модель операційної CRM системи, яка на відміну від існуючої використовує методи розрахунку потужності мобільного електротранспорту та енергетичної ємності акумуляторної батареї за вхідними даними замовника і технічними параметрами транспортного засобу, що дозволяє покращити систему взаємовідносин з клієнтами в галузі мобільного електротранспорту.

Практична цінність отриманих результатів полягає в тому, що на основі досліджень в магістерській кваліфікаційній роботі запропоновано програмні засоби для автоматизації взаємовідносин клієнтів в галузі мобільного електротранспорту.

Апробація результатів. Основні положення магістерської кваліфікаційної роботи обговорювалися на Міжнародній науково-практичній Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ 2023».

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто., а також було опубліковано порівняльний аналіз методів і програмних засобів автоматизації відносин з клієнтами в збірнику Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ 2023» [1].

Публікації. Результати досліджень опубліковано у збірнику міжнародної науково-практичної Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ 2023» [1].

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ І ПРОГРАМНИХ ЗАСОБІВ АВТОМАТИЗАЦІЇ ВІДНОСИН З КЛІЄНТАМИ

1.1 Аналіз предметної галузі та особливості процесів автоматизації відносин з клієнтами

Абревіатура CRM походить від англійського Customer Relationship Management, що в перекладі означає «управління взаємо-відносинами з клієнтами». Звідси слідує, що CRM система – набір технологій, які дозволяють автоматизувати бізнес-процеси, які торкаються взаємодії з клієнтами[2,3].

Існує три основні види CRM систем: операційні, аналітичні та колаборацийні. Хоча вони мають різні набори функцій і цілей, кожна з них має ту саму мету: покращити відносини з клієнтами та розвивати бізнес [4,5].

Операційні CRM-системи є ключовою складовою сучасних бізнес-процесів, оскільки вони спрямовані на автоматизацію операційних аспектів взаємодії з клієнтами. Ці системи допомагають підприємствам ефективно керувати процесами продажу, маркетингу та обслуговування клієнтів. Основні характеристики операційних CRM-систем включають:

1. Управління продажами:
   - ведення потенційних клієнтів – зберігання та оновлення інформації про потенційних клієнтів, включаючи контактні дані та інтереси;
   - трекінг угод – моніторинг стадій угод, від початкового контакту з клієнтом до укладення угоди.
2. Маркетингові автоматизації:
   - сегментація клієнтів – розподіл клієнтів на групи з метою більш ефективного спрямування маркетингових кампаній;
   - електронна розсилка – відправлення автоматизованих листів, спрямованих на залучення нових клієнтів або утримання існуючих.
3. Служба підтримки та обслуговування клієнтів:

- тікет-системи – обробка запитань та скарг клієнтів через створення тікетів і ведення їхньої історії;
- база знань – надання клієнтам доступу до інформації та рішень щодо продуктів чи послуг.

4. Автоматизація контактів:
   - управління контактами – зберігання та впорядкування інформації про клієнтів, включаючи історію взаємодії та комунікації.
5. Управління інвентарем:
   - моніторинг запасів – системи, що ведуть облік товарів або послуг, контролюючи їхню доступність та рух по ланцюжку постачань.
6. Аналіз та звітність:
   - аналіз результативності – здійснення аналізу даних для визначення ефективності продажів, маркетингових кампаній та обслуговування клієнтів;
   - звітність – надання звітів та статистики для прийняття обґрунтованих стратегічних рішень.

Операційні CRM-системи спрямовані на автоматизацію і поліпшення роботи всіх відділів компанії, які взаємодіють з клієнтами. Це допомагає підприємствам підтримувати ефективні та продуктивні відносини з клієнтами, що є важливим елементом успішної діяльності. [6].

Аналітичні CRM-системи використовуються для аналізу та інтерпретації великої кількості даних, що збираються підприємством про його клієнтів та їхню взаємодію з компанією. Ці системи надають глибокі інсайти, сприяють прийняттю стратегічних рішень та оптимізації бізнес-процесів. Основні характеристики аналітичних CRM-систем включають:

1. Аналіз даних клієнтів:
   - сегментація та класифікація – розділення клієнтів на групи за різними критеріями для більш ефективного управління взаємодією.
   - прогнозування попиту – використання даних для передбачення майбутніх потреб та змін в попиті.

2. Аналітика взаємодії з клієнтами:
   - моніторинг соціальних мереж – слідкування за активністю та відгуками клієнтів в соціальних мережах для оцінки репутації компанії;
   - аналіз відгуків та скарг – виявлення патернів та тенденцій в зворотньому зв'язку від клієнтів.
3. Здійснення аналізу продажів:
   - аналіз конверсії – визначення ефективності різних каналів продажів та ідентифікація можливостей для покращення;
   - аналіз корзини покупок – вивчення структури замовлень для виявлення схожих та взаємозалежних продуктів.
4. Прогностичний аналіз:
   - прогнозування витрат – оцінка та прогнозування витрат клієнтів на певний час;
   - прогнозування доходів – прогнозування майбутніх прибутків від різних сегментів клієнтської бази.
5. Постановка завдань для маркетингових кампаній:
   - оптимізація рекламних зусиль – аналіз результатів маркетингових кампаній для підвищення їхньої ефективності;
   - персоналізовані рекомендації – використання даних для створення персоналізованих пропозицій для клієнтів.
6. Стратегічне планування:
   - розвиток стратегій – надання інформації для розробки стратегій залучення нових клієнтів та утримання існуючих;
   - оцінка конкуренції – аналіз даних для порівняння з конкурентами та виявлення можливостей покращення.

Аналітичні CRM-системи допомагають компаніям розуміти свою клієнтську базу, ефективно реагувати на зміни в ринковому середовищі та приймати обгрунтовані стратегічні рішення. [7,8].

Колаборaційні CRM-системи спрямовані на поліпшення комунікації та співпраці всередині організації, зокрема між різними відділами, які взаємодіють з клієнтами. Ці системи дозволяють різним командам ефективно обмінюватися інформацією, спільно працювати над проектами та забезпечувати єднану картину відносин з клієнтами. Основні характеристики колаборaційних CRM-систем включають:

1. Об'єднання комунікації:
   - спільні чати та обговорення – можливість обговорювати клієнтів, проекти та інші аспекти взаємодії з клієнтами в режимі реального часу;
   - спільний доступ до інформації – забезпечення широкого доступу різних відділів до спільних ресурсів та даних.
2. Спільна робота над завданнями:
   - робочі групи та проекти – створення спільних проектів, які об'єднують представників різних відділів для вирішення конкретних завдань;
   - обмін завданнями та відстеження виконання – можливість призначати завдання, відстежувати їх виконання та надавати звіти.
3. Спільний доступ до інформації про клієнтів:
   - об'єднані профілі клієнтів – єдина база даних, яка містить повну інформацію про взаємодію клієнта з різними відділами;
   - історія взаємодії – можливість переглядати історію комунікації та торгових операцій з клієнтом.
4. Системи внутрішнього спілкування:
   - внутрішні форуми та борди – середовище для обміну ідеями, надання відгуків та співпраці внутрішньою командою;
   - внутрішні новини та оголошення – механізми для сповіщення співробітників про важливі події та оновлення взаємодії з клієнтами.
5. Інтеграція з іншими інструментами:

- синхронізація з електронною поштою та календарями – забезпечення спільної роботи з електронною поштою та графіком подій;
- інтеграція з іншими корпоративними системами – можливість обмінюватися даними з іншими системами, такими як ERP (Enterprise Resource Planning).

Колаборaційні CRM-системи важливі для забезпечення єдиної платформи для командної роботи та взаємодії всередині організації. Вони сприяють зміцненню співпраці та підвищенню ефективності роботи всіх відділів, які мають стосунки з клієнтами. [9,10].

На рисунку 1.1 зображено організацію і функціонування основних видів CRM систем.

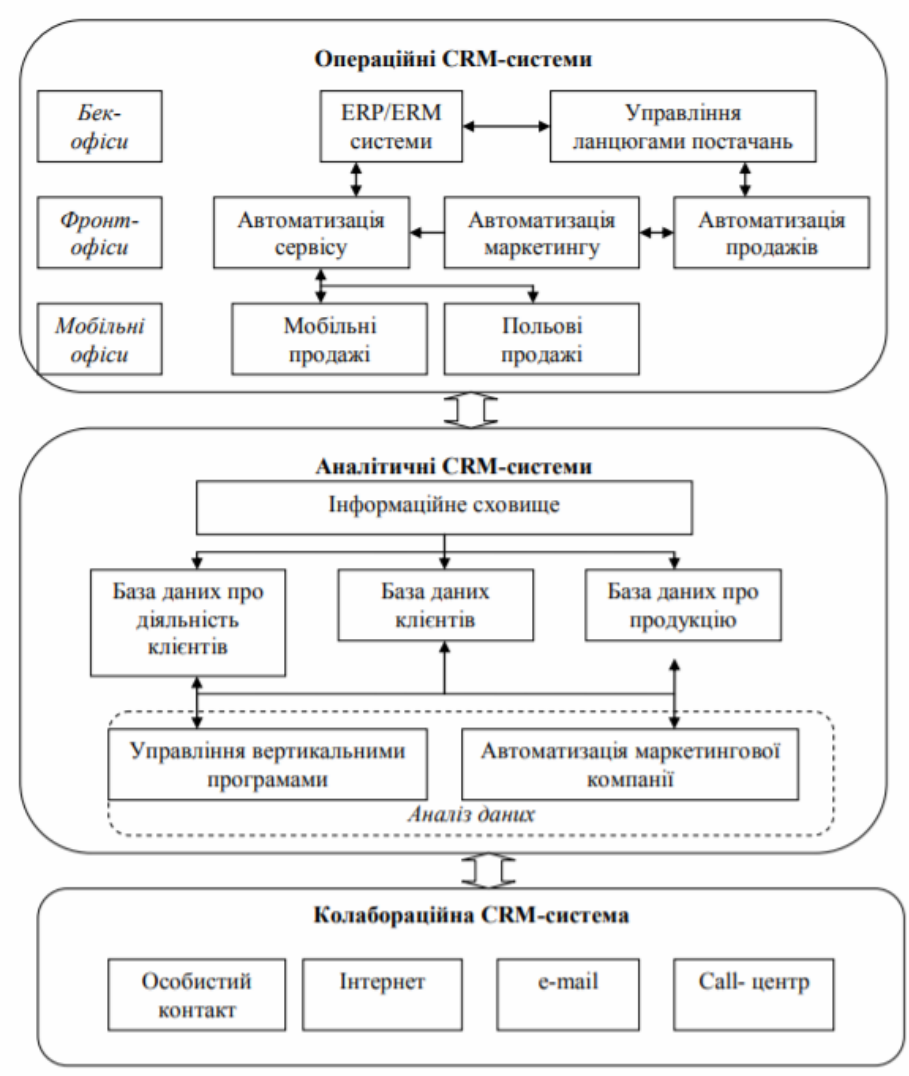


Рисунок 1.1 – Організація і функціонування CRM систем

Проблема номер 1 для сучасного бізнесу – це перетворення просто відвідувачів на постійних клієнтів. Є хибне рішення, що вкладаючи величезні кошти лише в залучення відвідувачів – маркетинг, рекламу, SEO-просування ви досягнете результатів[11,12]. Але цього замало – необхідно ретельно підходити до цього процесу, аналізувати, проводити аналітику та тільки після цього приймати рішення щодо впровадження вище перерахованих засобів зацікавлення клієнтів у своїх продуктах та послугах[13,14]. Саме впровадження CRM системи дає можливість бізнесу в повній мірі та максимально ефективно реалізувати ці завдання[15,16].

Завдяки використанню CRM компанія може зберігати та аналізувати історію взаємин з клієнтами, що дає можливість ретельніше та точніше планувати бізнес і забезпечувати підвищення його стабільності[17,18]. Зі зростанням бізнесу та масштабуванням продажів, взаємини з клієнтами все відчутніше впливатимуть на всі робочі процеси: від розробки продуктів до їх реалізації. Тому, чим раніше впровадити в діяльність підприємства CRM системи – тим швидше зможете надавати клієнтам ефективні рішення, раціонально управляти часом, оптимізувати бізнес-процеси, просувати бренд на ринку та, як наслідок, збільшувати прибуток підприємства[19,20].

На рисунку 1.2 зображено переваги та можливості CRM систем.

Враховуючи вподобання сучасного клієнта, підприємці повинні розуміти, що 75% споживачів потребують ідеального обслуговування[21,22]. По певним прогнозам на кінець 2023 року 81% організацій будуть використовувати AICRM системи для удосконалення взаємодії з клієнтами, тобто використання штучного інтелекту[23,24]. Підприємства, які впровадили в діяльність CRM-системи та використовують збільшують конверсію лідів на 17% та продуктивність роботи на 21%.

Сьогодні в України CRM – великий та швидкозростаючий ринок програмного забезпечення. Якщо підраховувати ефективність від систем, то повернення інвестицій (ROI) в розмірі 8.71 долара на кожний витрачений

долар. За цих умов термін окупності впровадження даних технологій становить близько року[25].

Рисунок 1.2 – Переваги і можливості CRM систем

Розроблювана нами CRM система мобільного електротранспорту є MVP продуктом, функціонал якого можна в подальшому розширювати в залежності від бажань конкретного замовника. Також, завдяки використанню мови програмування Java, дана система придатна для горизонтального масштабування в залежності від навантаження, що в свою чергу забезпечує використання даного продукту підприємствами різної величини.

1.2 Порівняльний аналіз існуючих програмних засобів автоматизації відносин з клієнтами

Сьогодні на світовому ринку представлено багато різних CRM систем в хмарному середовищі та щомісячною абонентською платою в середньому від 15 до 100 $ за користувача. Найкращі 5 CRM систем 2023: Salesforce: Best for

Small Businesses, Pipedrive: Best for Visual Sales Pipelines, monday Sales CRM: Best for Project Management, Oracle NetSuite CRM: Best for E-Commerce, Freshsales: Best for Communication[26].

Проаналізувавши CRM системи в Україні та провівши дослідження серед 46 підприємств України, які користуються даними системами було складено список топових систем, станом на січень 2023 року.

Creatio – це комплексна платформа для автоматизації бізнес-процесів і управління відносинами з клієнтами (CRM). Пропонує повний набір функцій управління відносинами з клієнтами, включаючи управління контактами, можливостями, замовленнями та іншими елементами життєвого циклу клієнта. Дозволяє автоматизувати різні бізнес-процеси, забезпечуючи ефективну організацію робочих процесів в компанії. Включає інструменти для створення, впровадження та відстеження маркетингових кампаній та стратегій. Управління відділом продажів, прогнозування, обробка замовлень та інші функції, спрямовані на підвищення ефективності продажів. Інструменти для надання послуг клієнтам, відстеження запитань та вирішення проблем. Вбудовані засоби аналітики та звітності для візуалізації та аналізу даних. Підтримка мобільних платформ для доступу та управління даними з різних пристроїв. Можливість інтеграції з іншими системами, такими як електронна пошта, соціальні мережі, бухгалтерія, електронні підписи тощо. Ціна: від 25$/місяць за одного користувача[27].

SalesDrive – CRM система управління продажу товарами в інтернет-магазинах. Можна налаштувати сервіс під конкретний бізнес з різних галузей. Має різні засоби інтеграції, а саме: телефонією, SMS, інтернет-магазином, Укрпоштою, Новою Поштою, Instagram, Facebook, Viber, Telegram, Email. Приватбанк, Монобанк, РРО, склад, рахунки, витрати. Prom, Rozetka, OpenCart, WordPress, Хорошоп. Ціна: 3-9 користувачів – 279 грн./міс, пакет 50 користувачів – 5080 грн./міс.[28].

LP-CRM – створена для продажу товарів через сайти-лендінги, великі торгові інтеграційні системи та інтернет системи. Переваги LP-CRM: Зручно

відслідковувати замовлення, зміну їхніх статусів, а також можливість створювати товарно транспортні накладні, оскільки в цій CRM є інтеграція зі службами доставки України: «Нова Пошта», «Укрпошта» та «Justin». Присутня інтеграція з популярними українськими маркетплейсами Prom та Rozetka дозволяє переносити всі замовлення з них у CRM без ризику їх втрати. Інтеграція з ПРРО Checkbox дозволяє створювати електронні чеки та одразу реєструвати їх у ДПС. Ціна: $9.99 на місяць – включає двох користувачів та підтримує до 200 замовлень[29].

KeepinCRM – зручна система для виконання автоматизації відносин з клієнтами. Має всі необхідні інструменти для вдення лідів до фінансових операція. Також, постійні оновлення та доступна вартість. Ціна: 1 користувач безкоштовно, кожен наступний – 250 грн. на місяць[30].

HugeProfit це система управління товарним бізнесом, яка призначена для автоматизації основних фінансових процесів: обліку залишків товару, контролю продажів та наочного відображення доходів та витрат. Дає можливість повністю автоматизувати бізнес: від закупки товару до отримання його покупцем. Ціна: від 99 грн. Є пробна версія тривалістю місяць, а також можливість отримати 300 грн бонусів[31].

Pipedrive – це CRM-система, яка спеціалізується на оптимізації процесів продажу та управлінні відносинами з клієнтами для підприємств. Система дозволяє вам ефективно відстежувати угоди та переговори, а також визначати етапи продажу. Інтеграція з електронною поштою, можливість відправляти листи безпосередньо з системи та налаштовувати нагадування. Ведення бази даних клієнтів та можливість класифікації контактів. Можливість сегментувати клієнтів за різними критеріями для більш ефективного управління. Надає засоби для аналізу продажів, відстеження результативності команди та визначення ключових показників ефективності. Доступ до системи через мобільні додатки для зручності роботи в дорозі. Підтримка інтеграції з різними іншими інструментами та сервісами, такими як Google Apps, Mailchimp, Slack і багато інших. Можливість створення та відстеження завдань, а також інтеграція

з календарем. Забезпечення безпеки та конфіденційності даних користувачів. Засоби для спільної роботи команди, обміну інформацією та коментування угод. Ціна: Від $19 на місяць[32].

CleverBOX: CRM – готовий набір інструментів для керування салоном краси, клінікою, центром. Більше 100 модулів для ефективного управління компанією від запису в календарі до інтеграції з вайбер та телеграм, складським обліком і фінансовим обліком. Автоматизуємо до 70% ключових процесів підприємства. Ціна: від $36 на місяць[33].

PERFECTUM CRM+ERP – український продукт. Система для всієї компанії може покривати всі процеси компанії. Має корпоративні та галузеві рішення та доступна у хмарній або коробковій версіях системи з мобільним додатком. Модульна структура дозволяє розширювати функціонал та кастомізувати її під себе. Ціна: від 175 грн. за 1 користувача[34].

KeyCRM – українська CRM система із фокусом на товарний бізнес. Переваги KeyCRM: готові модулі для автоматизації торгівлі з локальними (Prom.ua, Rozetka) та глобальними e-commerce платформами (eBay, Etsy, Amazon, Shopify, Woocommerce, PrestaShop, Magento), які доступні одразу після підключення та підтягують не тільки замовлення, а також синхронізують залишки та прораховають комісії маркетплейсів. Вбудовані інтеграції з українськими (Нова пошта, Укрпошта, Justin) та зарубіжними (USPS, DHL, UPS, FedEx, WesternBid, SellerOnline, SkladUSA) службами доставки/посередниками. Офіційна інтеграція з Instagram (діректі, сторіз, коментарі – в вікні CRM-системи) та месенджерами Viber та Telegram: чати з клієнтами та оформлення замовлення прямо всередині CRM. Ціна: від 19$. Безкоштовний доступ до 30 днів одразу після реєстрації[35].

Метою створення CRM системи мобільного електротранспорту є підвищення ефективності взаємовідносин підприємства з клієнтами, що дасть можливість створити свою інформаційну базу даних про клієнтів та забезпечити ефективний механізм управління маркетингом, продажами та сервісом.

1.3 Постановка задачі досліджень

Аналіз предметної галузі та особливості процесів автоматизації відносин з клієнтами дав можливість зрозуміти, яку потрібно обрати модель CRM системи для автоматизації бізнес-процесів в галузі мобільного електротранспорту. Також, було виявлено, чого не вистачає в існуючих методах і засобах, що представлені сьогодні на ринку України для покращення нашої системи.

Для ефективної взаємодії з клієнтами у сфері мобільного електротранспорту потрібні грамотні кваліфіковані консультанти, які швидко та якісно можуть допогти клієнту обрати відповідний мобільний електротранспорт, що відповідатиме вимогам та потребам замовника. Але 1 консультант одночасно здатний працювати з 1 клієнтом, а сучасні методи та засоби автоматизації роботи з клієнтами дають можливість якісно та швидко обслуговувати одночасно тисячі клієнтів з різних куточків країни, що дає можливість бізнесу збільшувати обсяги реалізації своїх продуктів та послуг, що в свою чергу забезпечує зростання прибутків.

Метою створення нашої CRM системи мобільного електротранспорту є вирішення наступних задач для автоматизації:

- збір даних про клієнтів системи;
- зручна та швидка самостійна реєстрація користувачів;
- пошук продуктів та послуг за каталогом;
- пошук продуктів та послуг за введеною назвою;
- фільтрація за відповідними параметрами;
- розрахунок потрібної потужності за даними користувача;
- розрахунок потрібної ємності батареї живлення за даними користувача;
- створення замовлення;
- опрацювання замовлення.

Основні нефункціональні вимоги нашої системи:

- надійність;
- швидкість;
- маштабованість;
- розширюваність функціоналу;
- відсутність помилок.

Основні функціональні вимоги:

- незареєстрований користувач може зареєструватися в системі;
- зареєстрований користувач може авторизуватися в системі за своїм логіном та паролем;
- авторизовані та неавторизовані користувачі можуть здійснювати пошук необхідних продуктів та послуг;
- авторизовані користувачі можуть залишати свої відгуки у системі;
- авторизовані користувачі можуть відстежувати історію своїх замовлень.

1.4 Висновки

Отже, в даному розділі було проведено аналіз сучасних методів і програмних засобів автоматизації відносин з клієнтами та розглянуто актуальність CRM систем у галузі ведення сучасного бізнесу. Зроблено аналіз існуючих програмних засобів автоматизації відносин з клієнтами в Україні та сформовано список топових систем, станом на січень 2023 року.

В результаті проведеного аналізу пропонується використати операційну модель CRM системи для вдосконалення методів автоматизації відносин з клієнтами в галузі мобільного електротранспорту.

2 МОДЕЛЮВАННЯ АВТОМАТИЗАЦІЇ ВІДНОСИН З КЛІЄНТАМИ ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ CRM СИСТЕМИ МОБІЛЬНОГО ЕЛЕКТРОТРАНСПОРТУ

2.1 Удосконалення методу автоматизації відносин з клієнтами

Важливими характеристиками мобільного електротранспорту, які забезпечують відповідність вимогам клієнта є потужність та ємність акумуляторної батареї. Вони безпосередньо впливають на швидкісно-динамічні показники та дальність пробігу на одному заряді акумуляторної батареї залежно від масо-габаритних показників і бажаної швидкості руху. А тому нам необхідно розробити методи обрахунку потужності та ємності акумуляторної батареї.

Для розробки методу відповідних розрахунків необхідно використати формули з розділів фізики – механіка та електродинаміка[36,37].

Оскільки, електродвигуни мають відповідні коефіцієнти корисної дії (ККД), то нам необхідно розрахувати номінальну потужність (формула 2.1):

$$P\text{н} = \frac{P}{\eta}, \tag{2.1}$$

де Р – потужність, яка необхідна для подолання всіх сил протидії руху електротранспорту, Вт;

η – коефіцієнти корисної дії.

Потужність Р (потужність, яка необхідна для подолання всіх сил протидії руху електротранспорту) розраховується за формулою 2.2:

$$P = F\text{ст} \cdot \nu, \tag{2.2}$$

де Fст – статична сила навантаження, Н;

ν – швидкість руху електротранспорту, м/с.

Статична сила навантаження Fст розраховується за формулою 2.3:

$$F\text{ст} = F\text{тр} + F\text{оп}, \tag{2.3}$$

де Fтр – сила тертя об асфальт коліс мобільного електротранспорту, Н;

Fоп – сила опору повітря, Н.

Сила тертя розраховується за формулою 2.4:

$$F\text{тр} = \mu \cdot m \cdot g \ , \tag{2.4}$$

де μ – коефіцієнт взаємодії шин коліс з асфальтом;

m – загальна маса електротранспорту з водієм, кг;

g – прискорення вільного падіння, м/с^2.

Сила опору повітря *F*оп розраховується за формулою 2.5:

$$F\text{оп} = Cx \cdot S \cdot \rho \cdot \left(\frac{v^2}{2}\right) , \tag{2.5}$$

де *Cx* – коефіцієнт фронтального опору;

S – фронтальна площа перерізу, м^2;

ρ – густина повітря, кг/м^3;

ν – швидкість руху електротранспорту, м/с.

Фронтальну площу *S* визначимо за формулою 2.6:

$$S = 0{,}5 \cdot h \cdot w \ , \tag{2.6}$$

де h – висота розташування водія на електротранспорті, м;

w – ширина розташування водія на електротранспорті, м.

За приведеними вище формулами ми зможемо вирахувати необхідну потужність мобільного електротранспорту, яка потрібна клієнту. А далі на основі цього показника ми можемо розрахувати необхідну ємність акумуляторної батареї для потрібної відстані, яку може подолати електротранспорт з водієм на одному заряді (формула 2.7):

$$Q = P\text{н} \cdot t \ , \tag{2.7}$$

де Рн – номінальна потужність електротранспорту, Вт;

t – час роботи електротранспорту, год.

Оскільки, для мобільного електротранспорту є актуальними швидкість руху та відстань, яку він може проїхати з цією швидкістю, то ми можемо виразити час через відношення відстані (S, км) до швидкості руху (ν, км/год) і

тоді ми отримаємо наступну формулу 2.8 для розрахунку ємності акумуляторної батареї:

$$Q = P\text{н} \cdot (\frac{S}{\nu}) , \quad (2.8)$$

Тепер нам потрібно вирахувати необхідну енергетичну ємність акумуляторної батареї в ампер годинах (А · год) за формулою 2.9:

$$Q = \frac{Q}{U} , \quad (2.9)$$

де Q – енергетична ємність акумуляторної батареї, Вт · год;

U – робоча напруга електротранспорту, В.

2.2 Удосконалення моделі автоматизації відносин з клієнтами

На основі формул, що використані для розрахунку необхідної потужності мобільного електротранспорту і енергетичної ємності акумуляторної батареї визначаємо набір вхідних даних, щоб створити моделі розрахунків.

Отже, для розрахунку потужності електротранспорту нам потрібні наступні вхідні дані:

- h – висота розташування водія на електротранспорті (1,8 м), м;
- w – ширина розташування водія на електротранспорті (0,7 м), м;
- ν – швидкість руху електротранспорту, м/с;
- ρ – густина повітря (1,29 кг/м^3), кг/м^3;
- *Cx* – коефіцієнт фронтального опору (*Cx* = 1,1);
- g – прискорення вільного падіння (9,8 м/с^2);
- m – загальна маса електротранспорту з водієм, кг;
- μ – коефіцієнт взаємодії шин коліс з асфальтом (0,0035);
- η – коефіцієнти корисної дії (середнє значення електродвигунів, що використоуються для створення мобільного електранспорту ≤ 0,8).

Тепер проаналізуємо дані і визначимо, які з них ми можемо винести в константи.

Для розрахунку фронтальної площі ми можемо використовувати середню висоту розташування водія на електротранспорті – 1,8 м та середню ширину розташування водія на електротранспорті – 0,7 м. Підставляємо дані у формулу розрахунку фронтальної площі (2.6) і розраховуємо значення:

$S = 0{,}5 * 1{,}8 * 0{,}7 = 0.63$ м^2

Нам відомі значення коефіцієнта фронтального опору ($Cx = 1{,}1$) та густина повітря ($\rho = 1{,}29$ кг/м^3), тому у формулі розрахунку опору повітря (2.5) ми можемо вирахувати добуток коефіцієнта фронтального опору, фронтальної площі перерізу та густини повітря, представивши цей його у вигляді коефіцієнта $f_{оп}$:

$f_{оп} = 1{,}1 * 0{,}63 * 1{,}29 = 0{,}89$

Тепер підставимо значення $f_{оп}$ у формулу сили опору повітря (2.10):

$$Fоп = fоп \cdot (\frac{v^2}{2}), \quad (2.10)$$

де $f_{оп}$ – коефіцієнт для розрахунку опору повітря (0,89);

v – швидкість руху електротранспорту, м/с.

У формулі розрахунку сили тертя (2.4) нам відомі значення коефіцієнту взаємодії шин коліс з асфальтом та прискорення вільного падіння, тому представимо їх добуток у вигляді коефіцієнта $f_{тр}$:

$f_{тр} = 0{,}0035 * 9{,}8 = 0{,}343$

Тепер підставимо значення $f_{тр}$ у формулу сили тертя (2.11):

$$Fтр = fтр \cdot m, \quad (2.11)$$

де $f_{тр}$ – коефіцієнт для розрахунку сили тертя (0,343);

m – загальна маса електротранспорту з водієм, кг;

Для зручності розрахунків та створення моделі розрахунку номінальної потужності можемо винести в константи наступні коефіцієнти:

- корисної дії;
- опору повітря;
- сили тертя.

У таблиці 2.1 представлені константи для розрахунку номінальної потужності мобільного електротранспорту.

Таблиця 2.1 - Константи розрахунку номінальної потужності

Константа	Опис	Значення
η	Коефіцієнти корисної дії	0,8
$f_{оп}$	Коефіцієнт для розрахунку опору повітря	0,89
$f_{тр}$	Коефіцієнт для розрахунку сили тертя	0,0343

У таблиці 2.2 представлені змінні вхідні дані для розрахунку номінальної потужності мобільного електротранспорту.

Таблиця 2.2 – Дані для розрахунку номінальної потужності

Дані	Опис
ν	швидкість руху електротранспорту, м/с
$m_т$	маса електротранспорту, кг;
$m_в$	маса водія, кг;
m	загальна маса електротранспорту з водієм, кг ($m = m_т + m_в$)

На основі констант та змінних вхідних даних можемо вивести формулу для нашої моделі розрахунку номінальної потужності електротранспорту (2.12):

$$P_н = \frac{\left(f_{тр} \cdot (m_т + m_в) + f_{оп} \cdot \left(\frac{\nu^2}{2}\right)\right) \cdot \nu}{\eta}, \quad (2.12)$$

де $f_{тр}$ – коефіцієнт для розрахунку сили тертя (0,343);

$m_т$ – маса електротранспорту, кг;

$m_в$ – маса водія, кг;

$f_{оп}$ – коефіцієнт для розрахунку опору повітря (0,89);

ν – швидкість руху електротранспорту, м/с;

η – коефіцієнти корисної дії.

Тепер для прикладу розрахуємо потужність за такими вхідними даними:

- маса електротранспорту = 35 кг;
- маса водія = 82 кг;
- швидкість руху = 50 км/год = 13,9 м/с.

Підставимо ці дані та константи у формулу моделі розрахунку номінальної потужності електротранспорту:

$$P\text{н} = \frac{\left(0{,}343 \cdot (35 + 82) + 0{,}89 \cdot \left(\frac{13{,}9^2}{2}\right)\right) \cdot 13{,}9}{\eta} = 1567 \text{ Вт}$$

Ці розрахунки та отримане значення використаємо потім для створення автоматичних юніт тестів.

В таблиці 2.3 представлено необхідні вхідні дані для розрахунку ененергетичної ємності акумуляторної батареї в ампер годинах (А · год):

Таблиця 2.3 – Вхідні дані для розрахунку ененергетичної ємності акумуляторної батареї

Дані	Опис
S	Відстань пробігу на одному заряді батареї, км
ν	Швидкість руху, км/год
Рн	Номінальна потужність електротранспорту, Вт
U	Напруга живлення електротранспорту, В

Тепер на основі вище згаданих формул 2.8 та 2.9 та вхідних даних вивдемо формулу 2.13 для моделі розрахунку ененергетичної ємності акумуляторної батареї в ампер годинах:

$$Q = \frac{P\text{н} \cdot \left(\frac{S}{\nu}\right)}{U}, \quad (2.13)$$

Для прикладу розрахуємо ененергетичну ємность акумуляторної батареї за наступними значеннями вхідних дани:

- відстань пробігу на одному заряді батареї = 100 км;

- швидкість руху = 50 км/год;
- номінальна потужність електротранспорту = 1567 Вт;
- напруга живлення електротранспорту = 60 В;

Підставимо вхідні дані у формулу моделі розрахунку ененергетичної ємності акумуляторної батареї та розрахуємо значення:

$$Q = \frac{1567 \cdot (\frac{100}{50})}{60} = 52{,}2 \text{ А} \cdot \text{год}$$

Ці розрахунки та отримане значення використаємо потім для створення автоматичних юніт тестів.

2.3 Проектування програмного засобу автоматизації відносин з клієнтами

В даному проекті на самому високому рівні архітектура представлена у вигляді моделі Клієнт-Сервер (рис. 2.1).

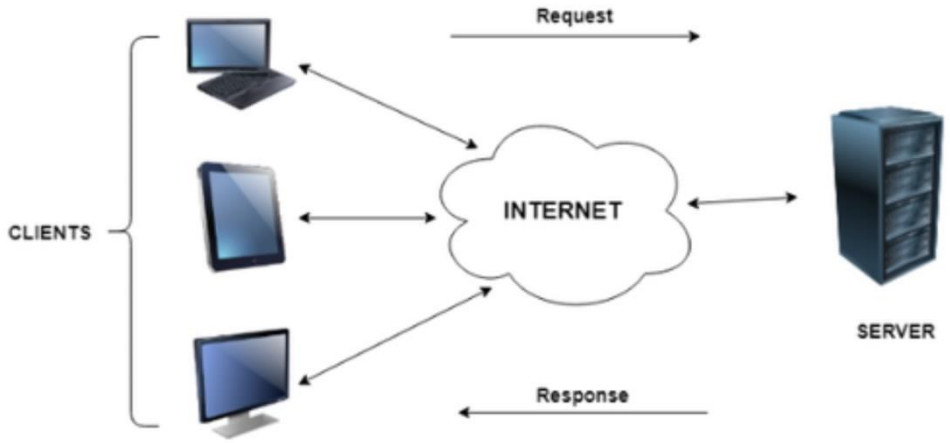


Рисунок 2.1 – Модель Клієнт-Сервер

Архітектура нашої CRM системи, що буде працювати на сервері, представлена у вигляді у вигляді монолітної архітектури з логічними шарами.

На рисунку 2.2 зображена архітектура розгортання системи.

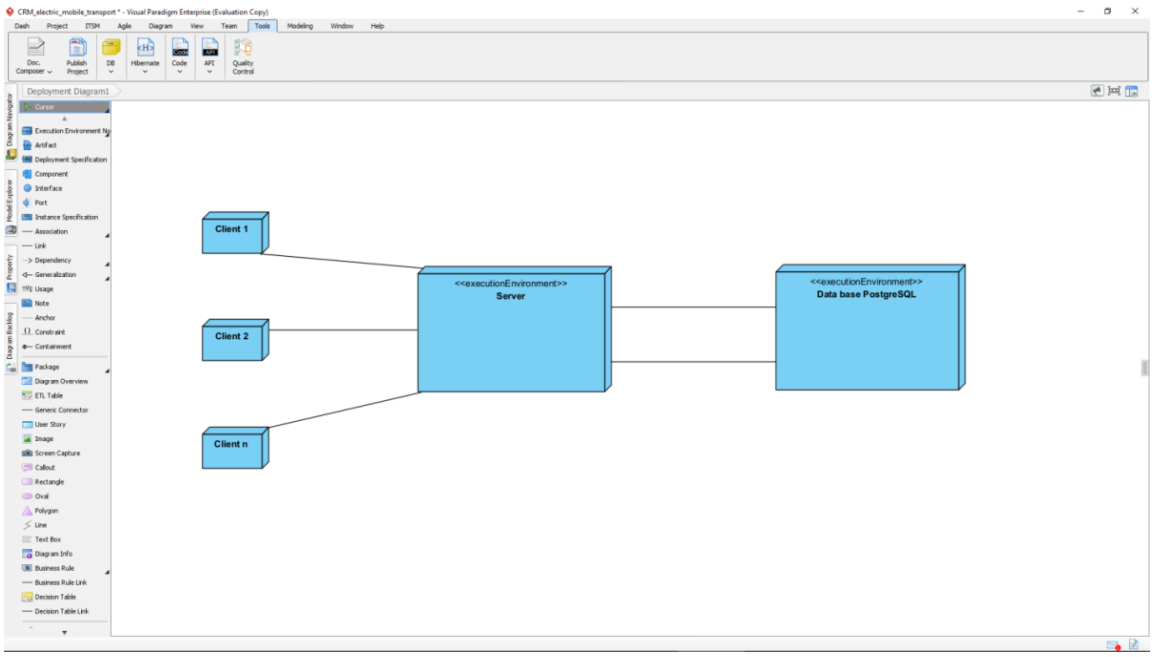


Рисунок 2.2 – Архітектура розгортання системи

Система розділена на 4 основні компоненти (рис. 2.3):

1) Entity – сунтності об’єктів системи.
2) Repository – відповідає за роботою з базою даних.
3) Controller – доступ до системи через http запити.
4) Service – бізнес логіка.

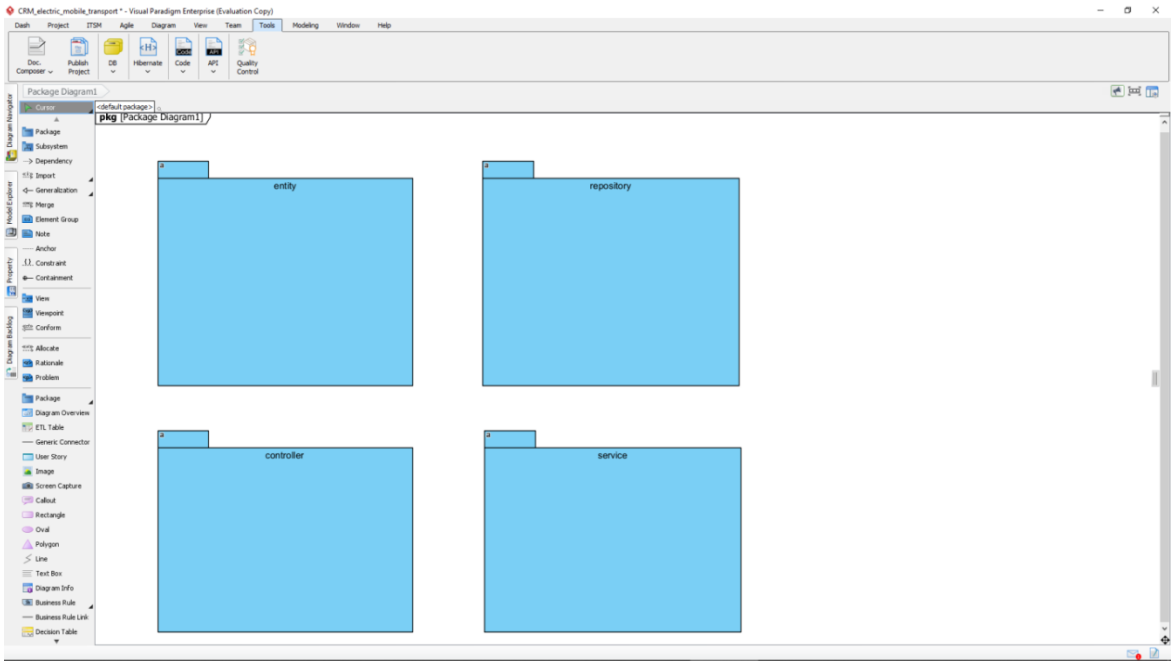


Рисунок 2.2 – Основні компоненти системи

При проектуванні більш детальної архітектури системи були використані різні підходи, але основним з них був шаблон проектування MVC.

Модель-Перегляд-Контролер (Model-View-Controller, MVC) - це підхід до проектування, який розділяє систему на три компоненти: модель, представлення та контролер. Модель відповідає за даний, представлення - за відображення даних користувачу, а контролер - за обробку введення користувача та зв'язок між моделлю та представленням [38].

Сервіс-Орієнтований Архітектурний Підхід (Service-Oriented Architecture, SOA) - це підхід до проектування, в якому система розбивається на набір сервісів, кожен з яких виконує певну функцію. Сервіси взаємодіють між собою за допомогою стандартизованих протоколів, таких як SOAP або REST, що забезпечує гнучкість та розширюваність системи [39].

Діаграма прецедентів - це візуальний засіб моделювання в програмному проектуванні, який використовується для опису поведінки системи з точки зору користувачів або зовнішніх систем. Головною метою створення діаграми прецедентів є визначення функціональної поведінки системи, зокрема, які функції системи повинні бути доступні для користувачів, які користувачі можуть використовувати ці функції та як вони можуть взаємодіяти з системою[40].

Діаграма прецедентів є важливим інструментом при проектуванні програмного забезпечення, оскільки дозволяє розуміти потреби користувачів та визначити вимоги до системи. Вона є ефективним засобом спілкування між розробниками та замовниками проекту, дозволяючи зрозуміти, як буде використовуватися система та які функції вона повинна мати. Крім того, діаграма прецедентів допомагає визначити переваги та недоліки різних альтернативних рішень та спрямовує процес розробки на реалізацію потреб користувачів.

У результаті було створено діаграму прецедентів (рисунок 2.3) для CRM системи мобільного електротранспорту, що включає у себе акторів (таблиця 2.4) та прецедентів (таблиця 2.5).

Таблиця 2.4. Характеристика акторів

Актор	Короткий опис
Адміністратор	Працівник компанії, який авторизується, здійснює глобальне керування системою, створює нових користувачів та налаштовує їхні ролі.
Менеджер замовлень	Працівник компанії, що авторизується, опрацьовує замовлення, додає нові товари та послуги в систему.
Менеджер клієнтів	Працівник компанії, який авторизується, читає відгуки клієнтів та відповідає на них.
Клієнт	Людина зацікавлена в товарах та послугах компанії, яка реєструється в системі, авторизується, здійснює пошук товарів та послуг, створює замовлення, пише відгуки про систему.

На основі вище визначених акторів опишемо прецеденти представлення в таблиці 2.5.

Таблиця 2.5 − Характеристика прецедентів

Прецедент	Короткий опис
Реєстрація	Створення нового користувача в системі з унікальним логіном та паролем, базовими правами звичайного користувача, з певним набором мінімальної інформації (ПІБ, електронна пошта, мобільний телефон, адреса та ін..). Нового користувача в системі може створити інший користувач з правами адміна або неавторизований користувач, який надішле свої дані (логін, пароль, ПІБ, електронна пошта, мобільний телефон) на відповідній сторінці реєстрації на сайті системи
Авторизація	Кожен користувач системи за своїм логіном та паролем здійснює вхід до системи. Після успішного входу користувачу доступний певний функціонал системи відповідно до наданих йому прав.

Продовження таблиці 2.5

Прецедент	Короткий опис
Створення замовлення	Виконується авторизованими та неавторизованими користувачами. Після того як знайдено потрібний товар, користувач виставляє необхідну кількість (не може перевищувати доступну кількість для замовлення) та натискає кнопку «Додати в кошик». Оформлення замовлення здійснюється на сторінці кошика за допомогою натискання кнопки «Оформити замовлення». Якщо користувач авторизований, то автоматично заповнюються потрібні дані на основі інформації, що збережена в системі про користувача. Якщо користувач не авторизований, то йому потрібно внести необхідні дані. Коли потрібні дані для оформлення замовлення внесені, то користувач тисне кнопку «Надіслати замовлення» і замовлення з відповідним статусом зберігається в системі.
Опрацювання замовлення	Оператор відстежує на сторінці опрацювання замовлень появу нових замовлень. Як тільки з’являється замовлення із статусом «Нове», оператор телефонує на номер мобільного, який має бути в інформації про замовлення та після узгодження всієї інформації з замовником переводить замовлення в статус «Підтверджено». Потім оператор збирає всі необхідні товари для конкретного замовлення в окремий ящик або пакет, роздруковує інформацію про замовлення та наліплює її на ящик чи пакет . Після цього через свій інтерфейс змінює статус замовлення на «Готове». Клієнт приходить на склад, проводить оплату та отримує замовлення. Після видачі замовлення оператор змінює статус на «Видано».
Створення відгуків	Виконується лише авторизованим Клієнтом на сторінці «Відкгуки»
Відповіді на відгуки	Виконується Менеджером клієнтів на сторінці «Відкгуки».

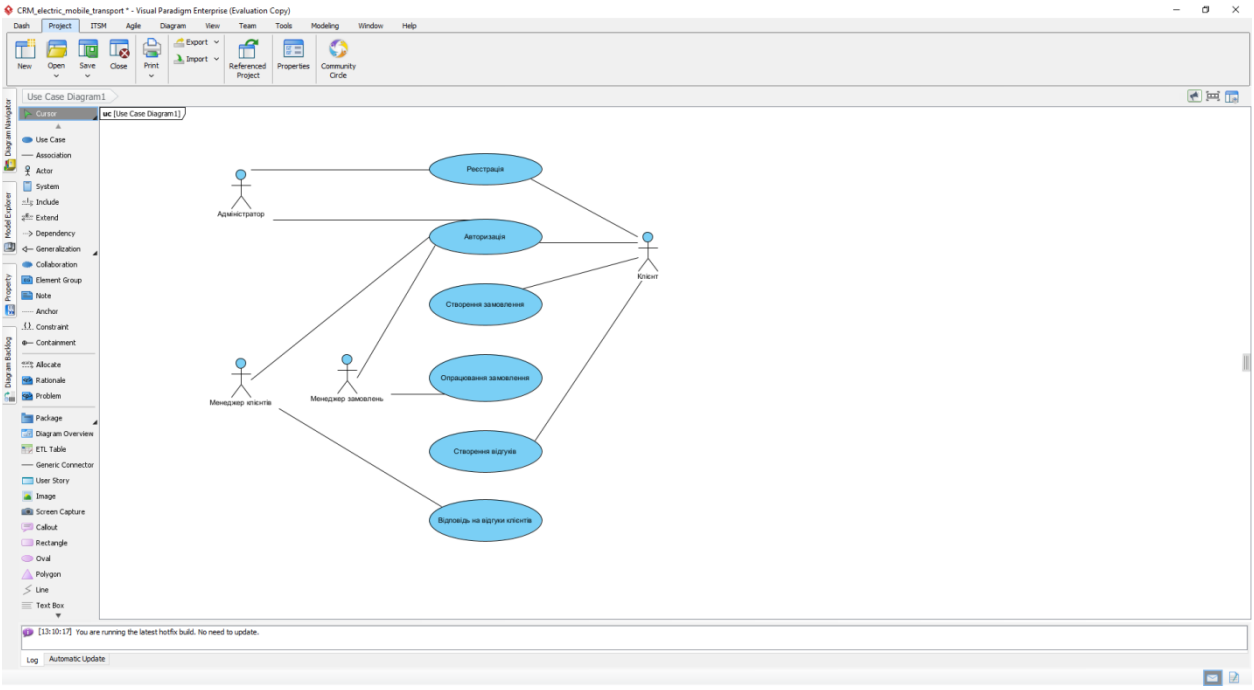


Рисунок 2.3 – Головна діаграма прецедентів

Діаграма класів - це візуальний засіб моделювання в програмному проектуванні, який використовується для відображення класів системи, їх взаємозв'язки та атрибути. Головною метою створення діаграми класів є визначення структури системи, зокрема, які класи існують, які поля та методи вони містять, та як вони пов'язані між собою [41].

Вони є однією з ключових діаграм в проектуванні об'єктно-орієнтованих систем, що дозволяє візуалізувати класи, інтерфейси, атрибути, методи і взаємозв'язки між ними.

Основна мета діаграми класів у контексті моделювання програмного забезпечення полягає у візуалізації структури класів системи та взаємодії між ними. Діаграма класів надає абстрактний огляд системи, вказуючи на класи, їх взаємозв'язки, атрибути та методи. Діаграма класів відображає класи системи та їх залежності, що дозволяє розробникам отримати загальний огляд проекту. Діаграми класів можуть служити документацією для системи, що допомагає командам розробників та іншим зацікавленим сторонам краще розуміти структуру та функціональність програмного забезпечення.

Опис класів системи наведено у додатку А.

У додатку Б зображено діаграму основних класів системи.

Далі було проведено більш детальне проектування на основі діаграм взаємодії.

Діаграма взаємодії є важливим інструментом моделювання у сфері розробки програмного забезпечення. Її основна мета полягає в описі взаємодії між об'єктами системи на основі послідовності повідомлень, які вони обмінюються між собою [42].

Вона дозволяє показати послідовність взаємодії між об'єктами в системі та ілюструвати порядок виконання повідомлень між ними. Це допомагає розробникам програмного забезпечення зрозуміти, як система працює та які процеси відбуваються в системі під час взаємодії між її об'єктами. Основні завдання діаграми взаємодії включають:

- показати послідовність взаємодії між об'єктами системи;
- визначити порядок виконання повідомлень між об'єктами;
- деталізувати специфікацію поведінки системи та розкрити її взаємодію між об'єктами;
- допомогти виявити можливі помилки в проекті та підвищити якість.

Діаграма взаємодії використовується в різних етапах розробки програмного забезпечення, включаючи аналіз вимог, проектування системи та тестування. Вона є важливим інструментом для комунікації між розробниками та замовниками, оскільки дозволяє легко зрозуміти взаємодію між різними частинами системи.

У результаті було створено діаграму послідовності (рисунок В.1) та діаграму активності (рисунок Г.1) для системи, що описують процес повідомлення якими обмінюються об’єкти.

Проектування системи виконане шляхом аналізу та інтеграції діаграм станів, діаграм активності, діаграм послідовності, діаграм класів, діаграм прецедентів, діаграм компонентів та діаграм пакетів і розгортання. Кожна з цих діаграм має свою мету та значення, і їхнє поєднання допомагає розкрити повну

картину функціональності системи та виявити можливі проблеми, що потребують вирішення.

Діаграми прецедентів дозволяють ідентифікувати акторів та їхні дії в системі. Ці діаграми допомагають зрозуміти функціональні вимоги до системи та інтерфейсу користувача.

Діаграми класів описують структуру системи та взаємозв'язки між класами. Ці діаграми допомагають зрозуміти архітектуру системи та її модульну структуру.

Діаграми активності описують взаємодію об'єктів в системі та показують, як об'єкти співпрацюють для досягнення певної мети. Ці діаграми допомагають зрозуміти взаємодію між об'єктами та процеси в системі.

Діаграми послідовності описують послідовність взаємодій між об'єктами в системі. Ці діаграми допомагають зрозуміти взаємодію між об'єктами та порядок виконання дій в системі.

Діаграми станів описують поведінку об'єктів в системі в різних станах. Ці діаграми допомагають зрозуміти, як система реагує на різні події та які зміни відбуваються в системі у різних станах.

Діаграма компонентів дозволяє розглянути систему з відкритої та модульної перспективи, що полегшує розуміння та підтримку системи в майбутньому.

Діаграма пакетів дозволяє відображати ієрархічну структуру пакетів, компонентів та модулів системи[43].

Поєднання різних типів діаграм при проектуванні системи дозволяє отримати комплексний огляд системи та її взаємодії з користувачами і іншими системами. Крім того, це дає змогу зрозуміти, які об'єкти потрібні для реалізації системи та які зв'язки між ними необхідні.

Таким чином, поєднання діаграм допомагає зрозуміти систему як цілісну сутність, виявити проблеми та потенційні помилки в розробці системи та зробити необхідні корективи до проектування для досягнення більш ефективного та оптимального результату.

2.4 Висновки

Отже, в даному розділі був розрозроблений метод розрахунку необхідної потужності мобільного електротранспорту з використанням фізичних формул. На основі цього методу було створено модель для автоматизації розрахунку необхідної потужності мобільного електротранспорту за вхідними даними клієнта та параметрами електротранспорту.

Також, було розрорблено метод розрахунку енергетичної ємності акумуляторної батареї з використанням фізичних формул. На його основі було створено модель для автоматизації розрахунку необхідної енергетичної ємності акумуляторної батареї для електротранспорту за розрахованою потужністю та даними клієнта, щодо бажаної відстані пробігу мобільного електротранспорту на одному заряді акумуляторної батареї.

Також, було спроектовано програмний засіб автоматизації відносин з клієнтом з використанням UML діаграм.

3 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ CRM СИСТЕМИ МОБІЛЬНОГО ЕЛЕКТРОТРАНСПОРТУ

3.1 Аналіз та обґрунтування вибору способів реалізації програмного засобу

3.1.1 Обґрунтування вибору мови програмування

Java та C++ — це дві популярні мови програмування, і вони мають свої особливості та використання[44]. Нижче подано порівняльну характеристику Java та C++ за різними аспектами:

1. Призначення:
   - Java – зазвичай використовується для веб-розробки, , серверних програм, мобільних додатків (Android), корпоративних застосунків та веб-служб;
   - C++ – використовується для системного програмування, розробки операційних систем, ігор, вбудованих систем, високоефективних додатків.
2. Система типів:
   - Java – об'єктно-орієнтована мова з автоматичним управлінням пам'яттю, відсутність вказівників;
   - C++ – об'єктно-орієнтована та процедурна мова з можливістю вручну керувати пам'яттю та використовувати вказівники.
3. Управління пам'яттю:
   - Java – використовує автоматичне управління пам'яттю (garbage collection);
   - C++ – розробник самостійно відповідає за виділення та звільнення пам'яті.
4. Виконання коду:
   - Java – використовується віртуальна машина Java (JVM), що забезпечує переносимість між платформами;

- C++ – компілюється безпосередньо в машинний код для конкретної платформи, що може забезпечувати вищу ефективність.

5. Багатозадачність:
   - Java – вбудована підтримка багатозадачність та багатопоточність;
   - C++ – також підтримує многозадачність, але розробник повинен вручну керувати потоками та синхронізацією.
6. Винятки:
   - Java – використовує обробку винятків (exception handling);
   - C++ – також має механізм обробки винятків, але також може використовувати традиційний підхід з кодами помилок.
7. Стандартна бібліотека:
   - Java – має обширну стандартну бібліотеку, що забезпечує готові класи для різних завдань;
   - C++ – також має потужну стандартну бібліотеку, але розробникам надається більше контролю над тим, які частини бібліотеки використовувати.
8. Платформенна незалежність:
   - Java – висока сумісність завдяки віртуальній машині Java.
   - C++ – потребує компіляції для конкретної платформи, менша сумісність.

Отже, враховуючи те, що Java призначена для написання серверних програм, має автоматичне управління пам’ятю, вбудовану багатозадачність та багатопоточність, а також високу сумісність з різними операційними сисетами, то для програмної реалізації було обрано мову програмування Java у поєднанні з сучасним java фреймворком Spring boot, який є зручною та гнучкою системою для розробки та тестування систем з різною архітектурою та завданнями[45]. Це дасть можливість створити високопродуктивний сервер з можливістю вертикального та горизонтального масштабування. Також це дає перевагу у зручності підтримки та розширюваності бажаного функціоналу в процесі експлуатації нашої системи.

3.1.2 Обґрунтування вибору бази даних

PostgreSQL та MySQL — це дві популярні об'єктно-реляційні системи управління базами даних (СУБД). Обидві системи мають свої переваги та особливості, і вибір між ними може залежати від конкретних вимог та вподобань користувача[46].

Вибір між PostgreSQL та MySQL для проекту на Java з Spring Boot може залежати від конкретних вимог проекту, але обидві бази даних є популярними та широко використовуваними, а також обидві можуть успішно взаємодіяти з Java та Spring Boot.

Ось кілька основних аспектів, які можна врахувати при виборі між PostgreSQL та MySQL для вашого проекту:

1. Визначення типів даних – PostgreSQL має більше вбудованих типів даних та дозволяє користувачам визначати свої власні типи. Якщо вам потрібно використовувати специфічні типи даних, PostgreSQL може бути більш гнучким.
2. Підтримка SQL-стандартів – обидві системи підтримують багато SQL-стандартів, але PostgreSQL відомий своєю високою ступенем відповідності до стандартів.
3. Реплікація та висока доступність – якщо вам важливі висока доступність та реплікація, слід розглянути деталі конкретних можливостей PostgreSQL та MySQL у цих питаннях. PostgreSQL володіє потужними механізмами реплікації[12].
4. Інтеграція з Java та Spring Boot – обидві бази даних мають драйвери JDBC для взаємодії з Java та Spring Boot. Вибір може залежати від вашого досвіду, зручності роботи з певною базою даних, або від ваших особистих вподобань.
5. Спільнота та підтримка – обидві бази даних мають великі та активні спільноти. Вам може бути зручніше вибирати ту базу даних, до якої ви вже звикли або яку ви вважаєте більш зручною у використанні.

6. Розширені можливості – PostgreSQL зазвичай вважається більш потужною для складних операцій JOIN та інших продвинутих можливостей SQL.

Враховуючи ці аспекти, обидві бази даних можуть бути відмінними виборами для проектів на Java з Spring Boot. Але PostgreSQL володіє потужнішими механізмами реплікації, що є плюсом для надійності, а також в плані продуктивності є більш потужною для складних операцій JOIN. PostgreSQL використовує ліцензію PostgreSQL, яка є вільною та відкритою.

Отже, для нашого проекту було обрано базу даних PostgreSQL.

3.1.3 Обґрунтування вибору середовища розробки

Для реалізації розробки CRM системи мобільного електротранспорту було обрано сучасне середовище розробки IntelliJ IDEA (Ultimate Edition). Даний продукт є лідером в галузі інструментального програмного забезпечення та використовується компаніями різної величини в багатьох країнах світу.

IntelliJ IDEA (Ultimate Edition) має цілий ряд переваг представлених нище.

Підтримка різних операційних систем:

- Windows;
- Linux;
- MacOS.

Ергономічне середовище:

- продумана в кожному аспекті та готова до використання одразу після встановлення;
- щвидкий доступ до всіх функцій та вбудованих інструментів, необхідних розробнику;
- широкі можливості індивідуального налаштування;

- можливість налаштувати середовище відповідно до свого робочого процесу: задати клавіші, встановити плагіни, налаштувати інтерфейс на свій розсуд і т.д.

Легко стартувати:

- швидкий запуск та створення першого проекту;
- у майстрі New Project можна вказати потрібну мову, інструменти збирання (наприклад Maven або Gradle) та версію JDK;
- якщо потрібний JDK ще не встановлений, IDE завантажить його з інтернету;
- інші параметри будуть задані автоматично;
- можна відкривати інші проекти, імпортувати існуючі проекти Maven або Gradle, а також вилучати проекти із систем контролю версій.

Комбінації клавіш різних дій:

- у IntelliJ IDEA передбачені поєднання клавіш практично для будь-яких дій - від перегляду недавніх файлів до запуску та налагодження проекту;
- одне з універсальних поєднань - подвійний Shift (Search Everywhere);
- ця функція дозволяє знайти будь-які об'єкти в проекті або за його межами;
- діапазон пошуку може включати все: від файлів, дій, класів та символів до налаштувань, елементів інтерфейсу та історії з Git.

Спеціальні можливості:

- IntelliJ IDEA пропонує різні спеціальні можливості, які можуть знадобитися;
- сумісні програми читання з екрана;
- налаштувати кольори різних елементів інтерфейсу;
- додати контрастності смуг прокручування;
- змінити розмір вікон і шрифту в редакторі.

Глибокий аналіз коду:

- IntelliJ IDEA створена в першу чергу для розробки на Java та Kotlin – але розуміє багато інших мов програмування: Groovy, Scala, JavaScript, TypeScript та SQL, і пропонує інтелектуальну допомогу в написанні коду на кожному з них;

- початкова індексація вихідного коду дозволяє IDE створити віртуальну мапу проекту. Використовуючи інформацію віртуальної мапи, миттєво виявляє помилки, пропонує варіанти автодоповнення коду з урахуванням контексту, виконує рефакторинг тощо.

Запуск, тестування та налагодження:

- в IntelliJ IDEA вбудований ефективний набір інструментів для налаштування параметрів запуску та збирання програми;

- налагодження коду, а також застосування та розробки тестів JUnit прямо в IDE.

Вбудовані інструменти та інтеграція:

- IntelliJ IDEA пропонує важливі вбудовані інструменти та можливості інтеграції, завдяки яким ви можете працювати у звичному середовищі, не перемикаючись між різними програмами.

Інтеграція із системами контролю версій:

- IntelliJ IDEA за промовчанням підтримує найпопулярніші системи контролю версій, такі як Git, Subversion, Mercurial та Perforce;

- проект із системи контролю версій можна клонувати прямо на початковому екрані, проаналізувати різницю між двома версіями, керувати гілками, записувати та відправляти зміни, вирішувати конфлікти злиття, переглядати історію змін.

Фреймворки JVM:

- IntelliJ IDEA Ultimate забезпечує першокласну підтримку провідних фреймворків та технологій для розробки сучасних додатків та мікросервісів;

- в IDE вбудована підтримка Spring та Spring Boot, Jakarta EE, JPA, Reactor та інших фреймворків.

Веб-разработка:

- IntelliJ IDEA Ultimate включає всю функціональність WebStorm – інтегрованого середовища розробки для JavaScript і пов'язаних технологій, включаючи TypeScript, React, Vue, Angular, Node.js, HTML і файлів стилів.

Розгортання:

- IntelliJ IDEA Ultimate пропонує інтеграцію з найпопулярнішими системами управління контейнерами: Kubernetes та Docker;
- крім того є сторонні плагіни для розгортання коду в Amazon Web Services, Google Cloud та Azure.

Віддалена розробка та спільна робота:

- IntelliJ IDEA пропонує розробникам все потрібне для віддаленої роботи;
- IDE допомагає продуктивно працювати в команді, де б ви не знаходилися, і писати код на ноутбуці будь-якої потужності, адже вся ресурсомістка обробка виконується на віддаленому сервері [47].

Отже, IntelliJ IDEA – провідна IDE для розробки на Java та Kotlin, яка допомагає працювати продуктивніше за рахунок інтелектуальної допомоги в написанні коду, надійних рефакторингів, швидкої навігації за кодом, широкого набору вбудованих інструментів розробника, підтримки веб- та корпоративної розробки та багатьох інших корисних можливостей.

3.2 Розробка програмного засобу

У нас вже сформовані вимоги та зроблено проектування нашої системи і ми можемо тепер розпочати програмну реалізацію нашої CRM системи мобільного електротранспорту.

Оскільки, ми маємо створити web-сервер на мові Java з використанням Spring boot, то заходимо на web-ресурс Spring Initializr [48] (рис. 3.1) і створюємо наш проект обраючи проект Maven, мову програмування Java та потрібну версію Spring boot.

Також заповнюємо метадані для нашого проекту:

1. Group
2. Artifact
3. Name
4. Description
5. Package name

Потім обираємо потрібні нам залежності, генеруємо проект і завантажуємо його на свій комп’ютер.

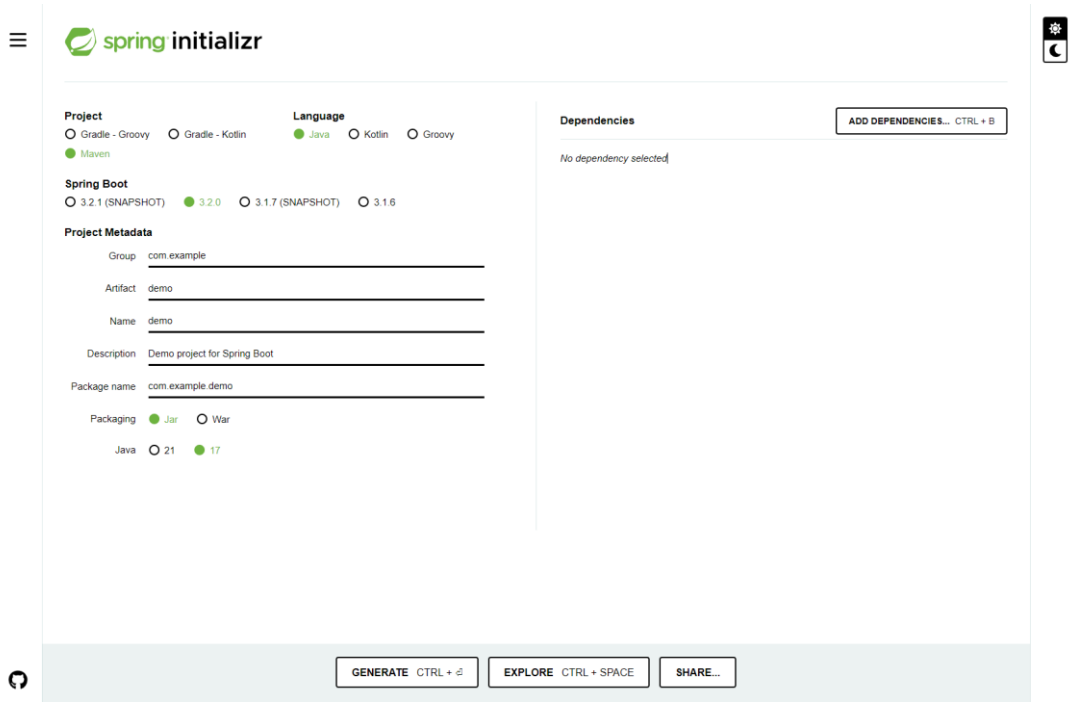


Рисунок 3.1 – Web-ресурс Spring Initializr

У розділі проектування було вказано, що ми обрали для розробки нашого продукту середовище розробки IntelliJ IDEA. Тому відкриваємо наш проект в IntelliJ IDEA (рис. 3.2).

Створюємо необхідні пакети для логічних шарів нашого програмного засобу (рис. 3.3):

- config
- controller
- entity

- repository
- user
- util

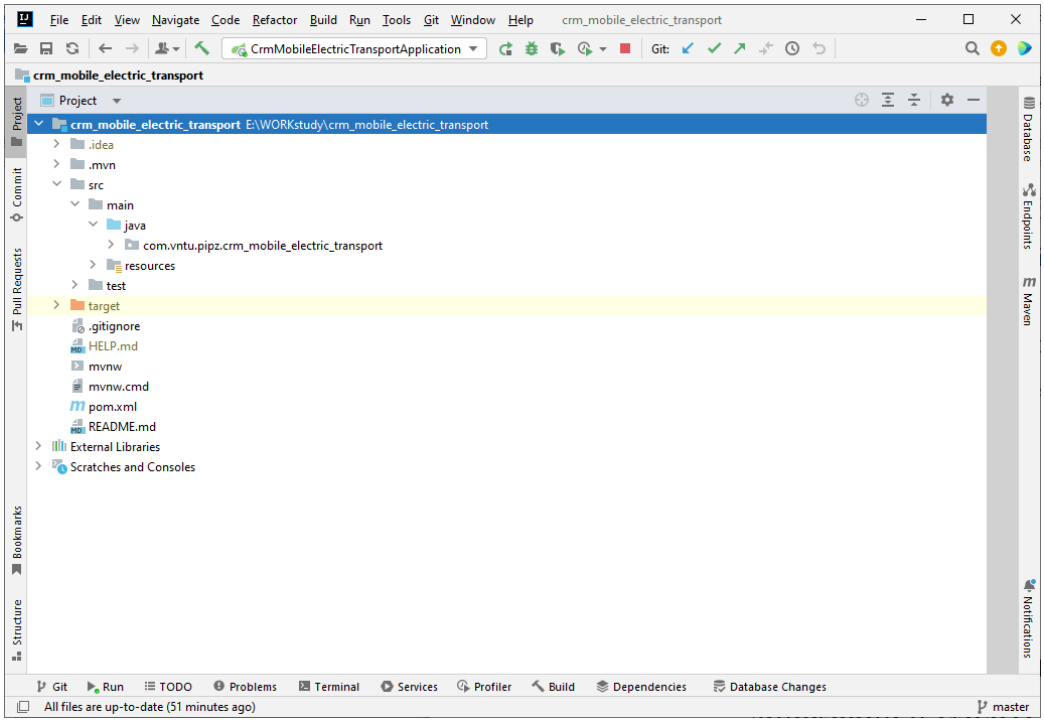


Рисунок 3.2 – Відкритий проект в IntelliJ IDEA

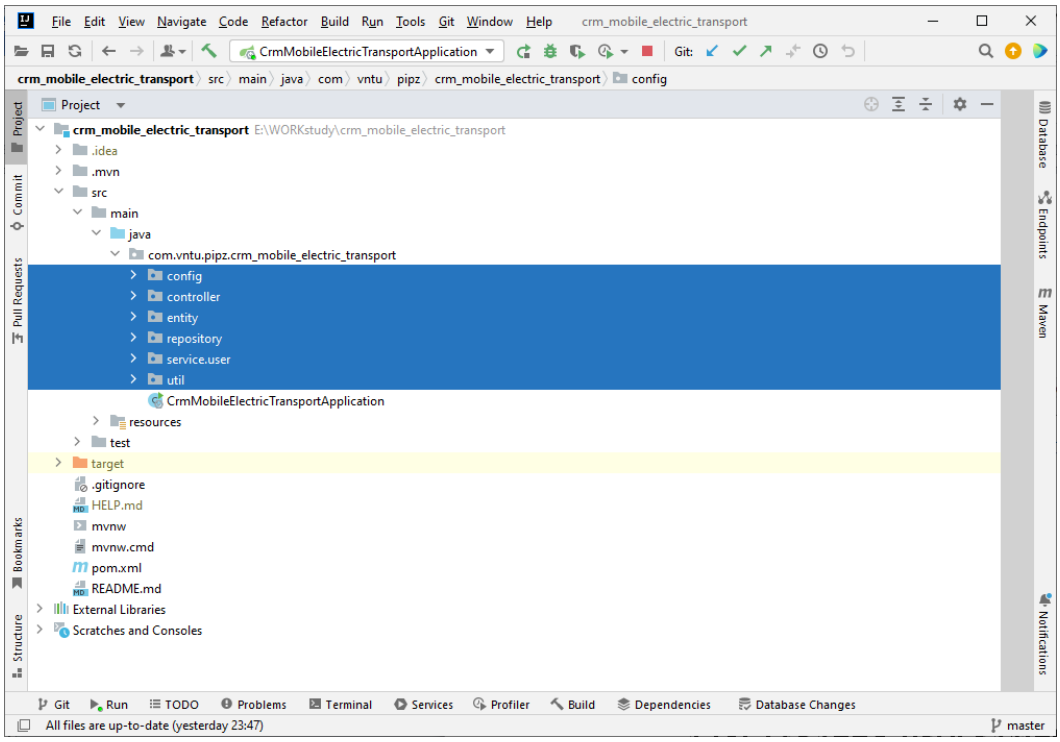


Рисунок 3.3 – Пакети CRM системи мобільного електротранспорту

Для зручності розробки та контролю версій програмного продукту було використано систему Git (рис. 3.4), як локальний репозиторій контролю версій. А також завантажили наш проект на віддалений репозиторій контролю версій GitHub (рис. 3.5).

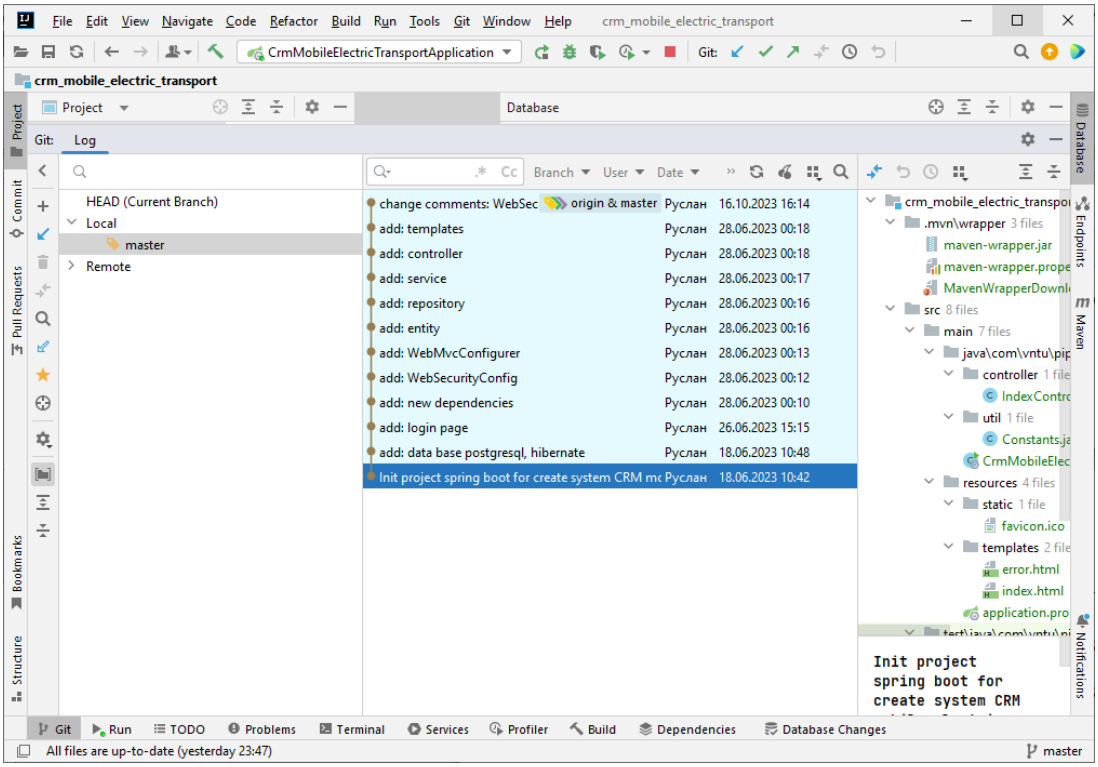


Рисунок 3.4 – Локальний репозиторій контролю версій Git

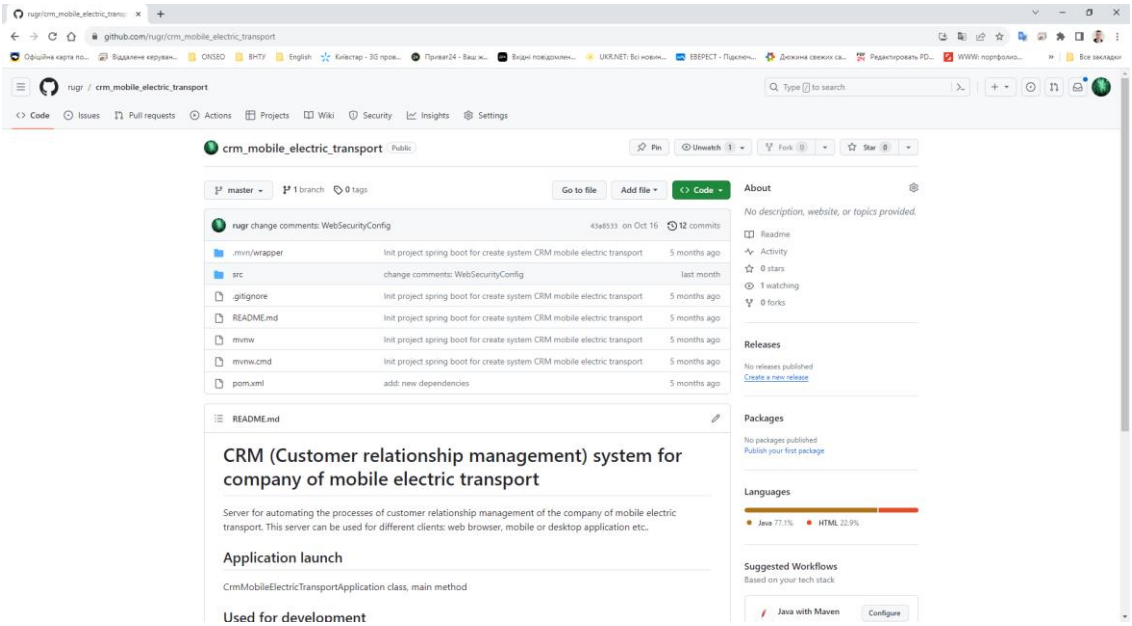


Рисунок 3.5 – Віддалений репозиторій контролю версій GitHub

Локально періодично робились коміти в локальному репозиторії контролю версій Git і потім всі зміни періодично зберігались у віддаленому репозиторії контролю версій GitHub (рис. 3.6).

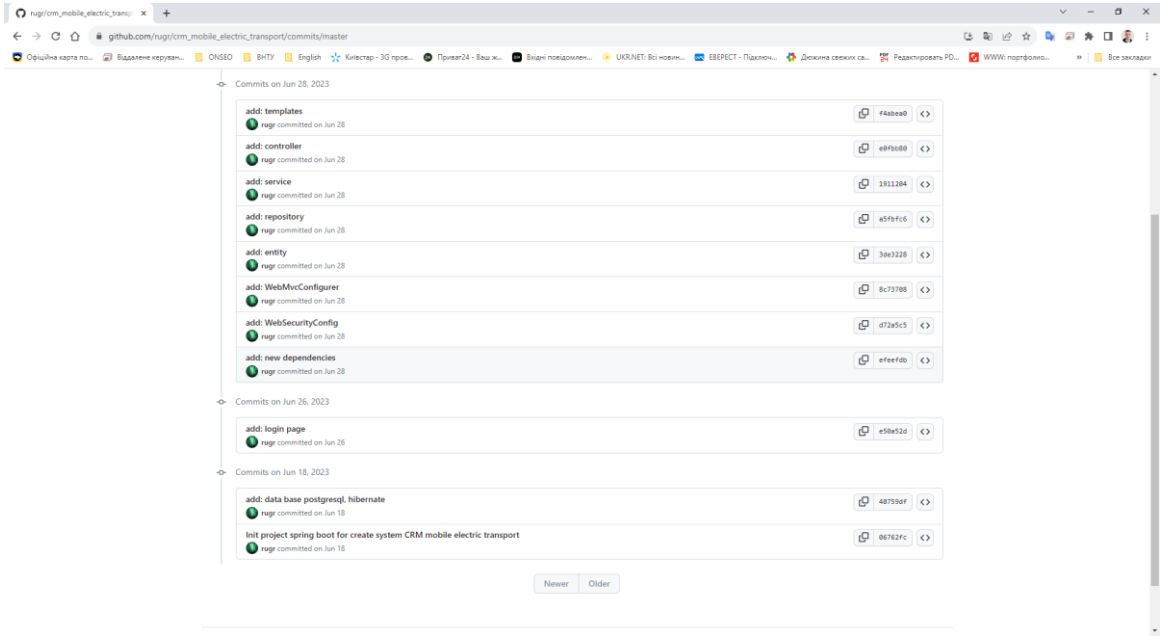


Рисунок 3.6 – Коміти в системі контролю версій GitHub

Далі було налаштовано файл Pom.xml та успішно завантажено всі необхідні залежності для нашого проекту з репозиторія Maven (рис. 3.7).

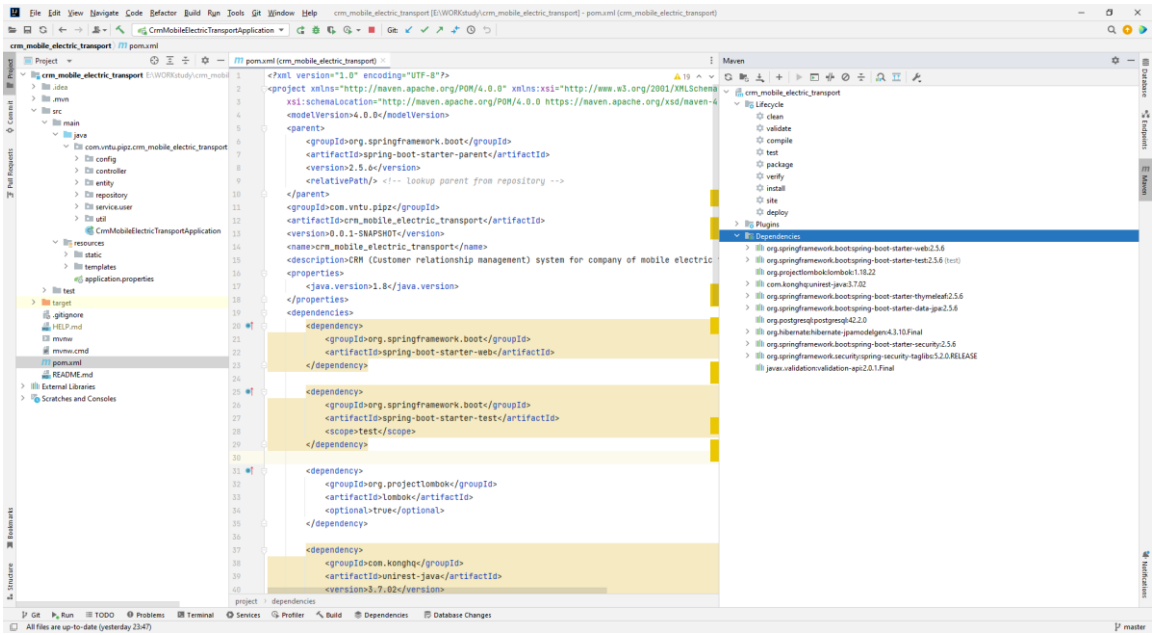


Рисунок 3.7 – Файл Pom.xml та завантаженыі залежності

Для створення нашої системи ми використали базу даних PostgreSQL. За допомогою клієнта pgAdmin ми створили базу даних (рис. 3.8). А для підключення нашого проекту до створеною бази даних в PostgreSQL були зроблені необхідні налаштування у файлі application.properties (рис. 3.9).

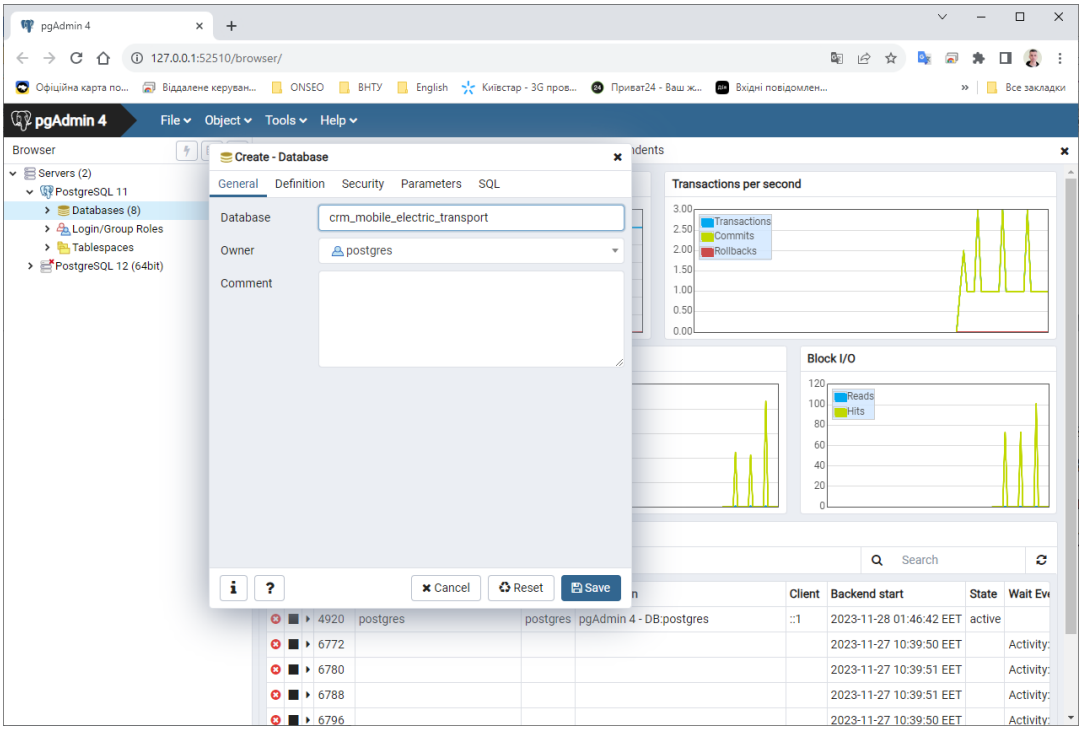


Рисунок 3.9 – Створення бази даних в PostgreSQL

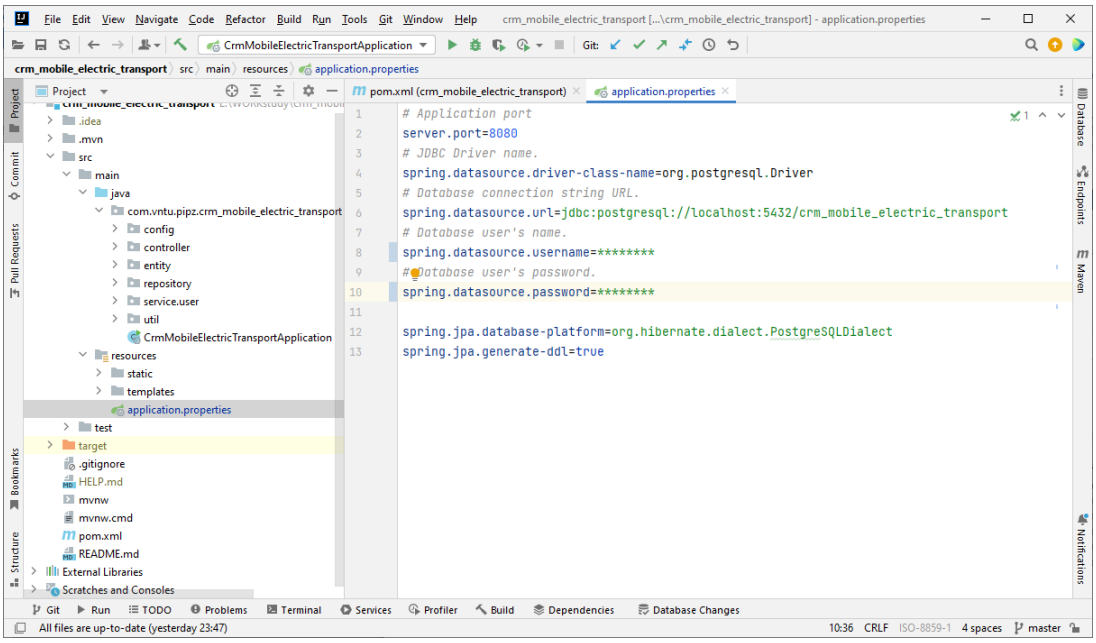


Рисунок 3.8 – Налаштування підключення проекту до бази даних

Потім ми створили класи основних сутностей для нашої системи (рис. 3.10), а також було створено відповідні таблиці в базі даних PostgreSQL (рис. 3.11).

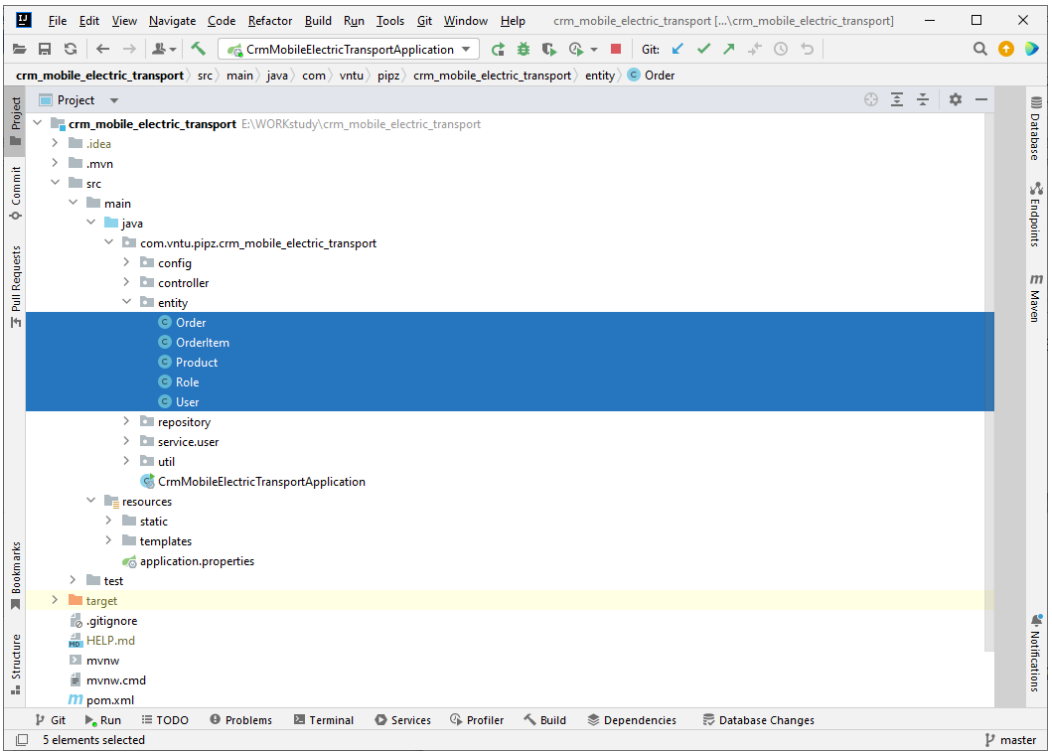


Рисунок 3.10 – Класи сутностей в проекті

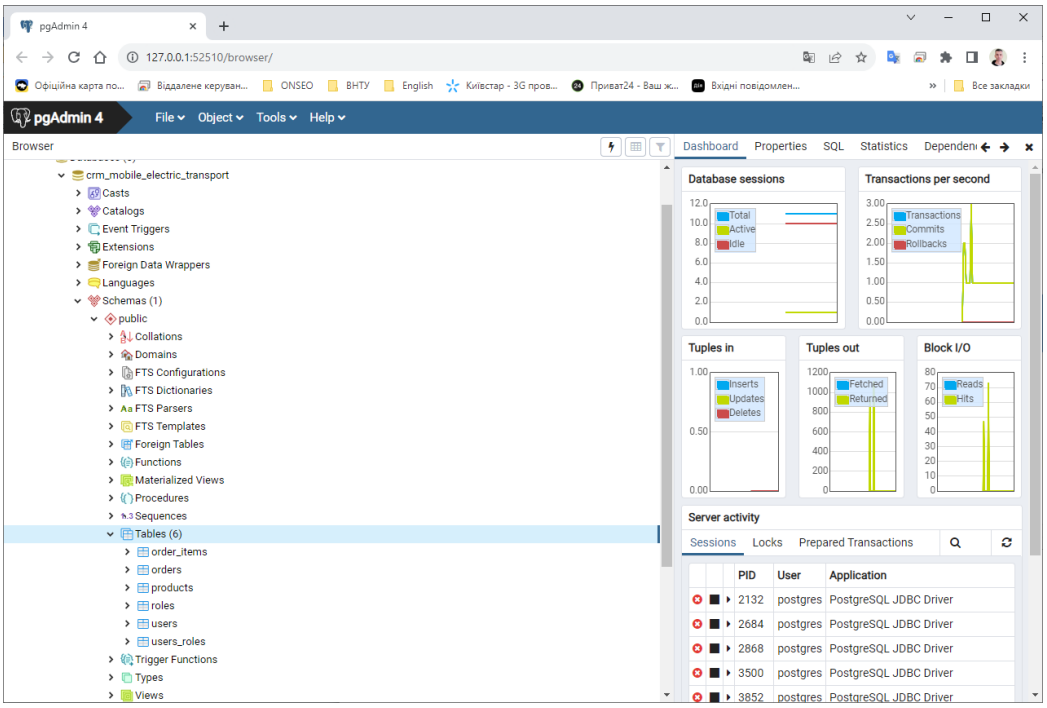


Рисунок 3.11 – Таблиці сутностей системи в базі даних

Для взаємодії з базою даних були створені інтерфейси репозиторіїв (рис. 3.12).

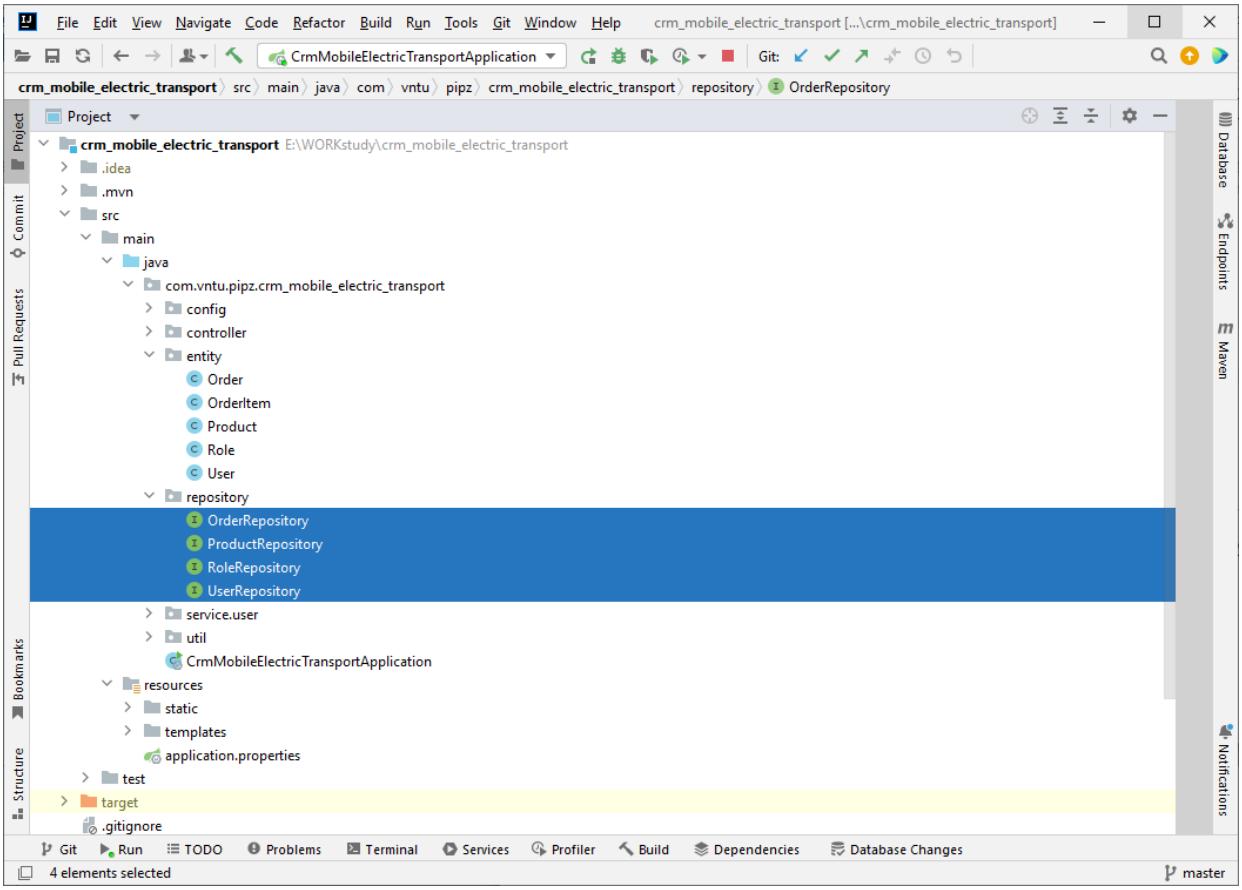


Рисунок 3.12 – Інтерфейси репозиторіїв для взаємодії з базою даних

Потім були реалізовані класи сервісів для виконання процесів бізнес логіки нашої системи.

Далі були реалізовані моделі розрахунку бажаної потужності мобільного електротранспорту та енергетичної ємності акумуляторної батареї.

Для реалізації розрахунку бажаної потужності мобільного електротранспорту було використано константи та дані з розділу 2 і на основі формули для цієї моделі розрахунку було створено клас розрахунку потужності мобільного електротранспорту (рис. 3.13).

Також, з використанням даних та формули для моделі розрахунку енергетичної ємності акумуляторної батареї з розділу 2 було створено модель розрахунку енергентичної ємності (рис. 3.14).

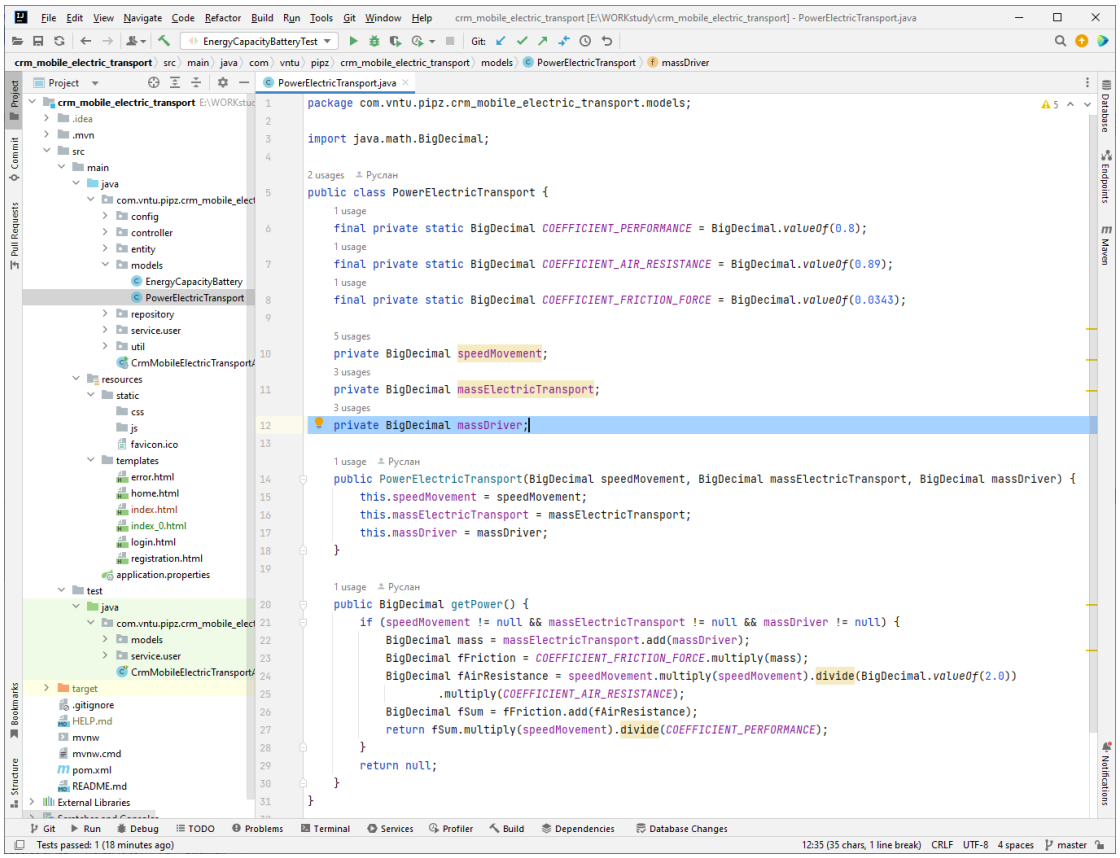


Рисунок 3.13 – Клас-модель розрахунку потужності

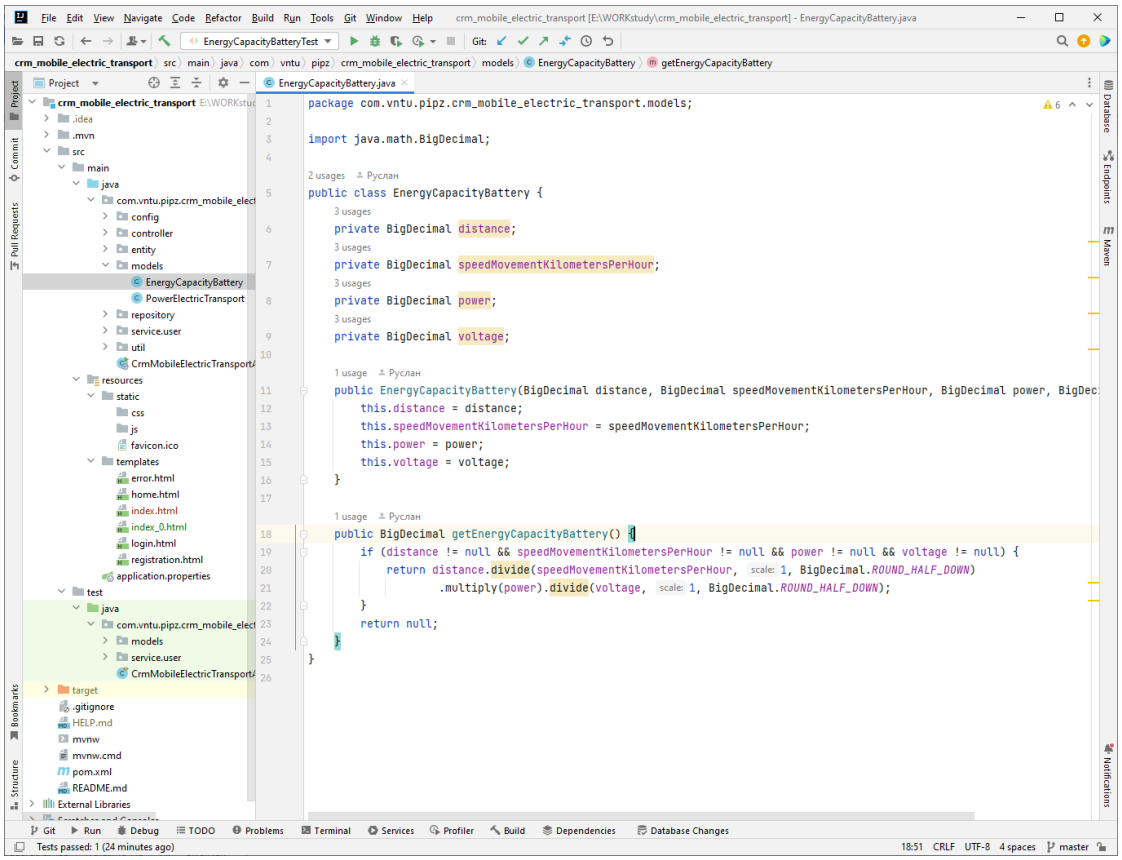


Рисунок 3.14 – Клас-модель розрахунку ємності

Наступним етапом була реалізація класів контролерів для обміну http запитами з програмами клієнтами.

В якості web-клієнта нашої системи було розроблено HTML шаблони сторінок на базі шаблонізатора Thymeleaf, щоб можна було продемонструвати роботу нашого web-сервера (рис. 3.15).

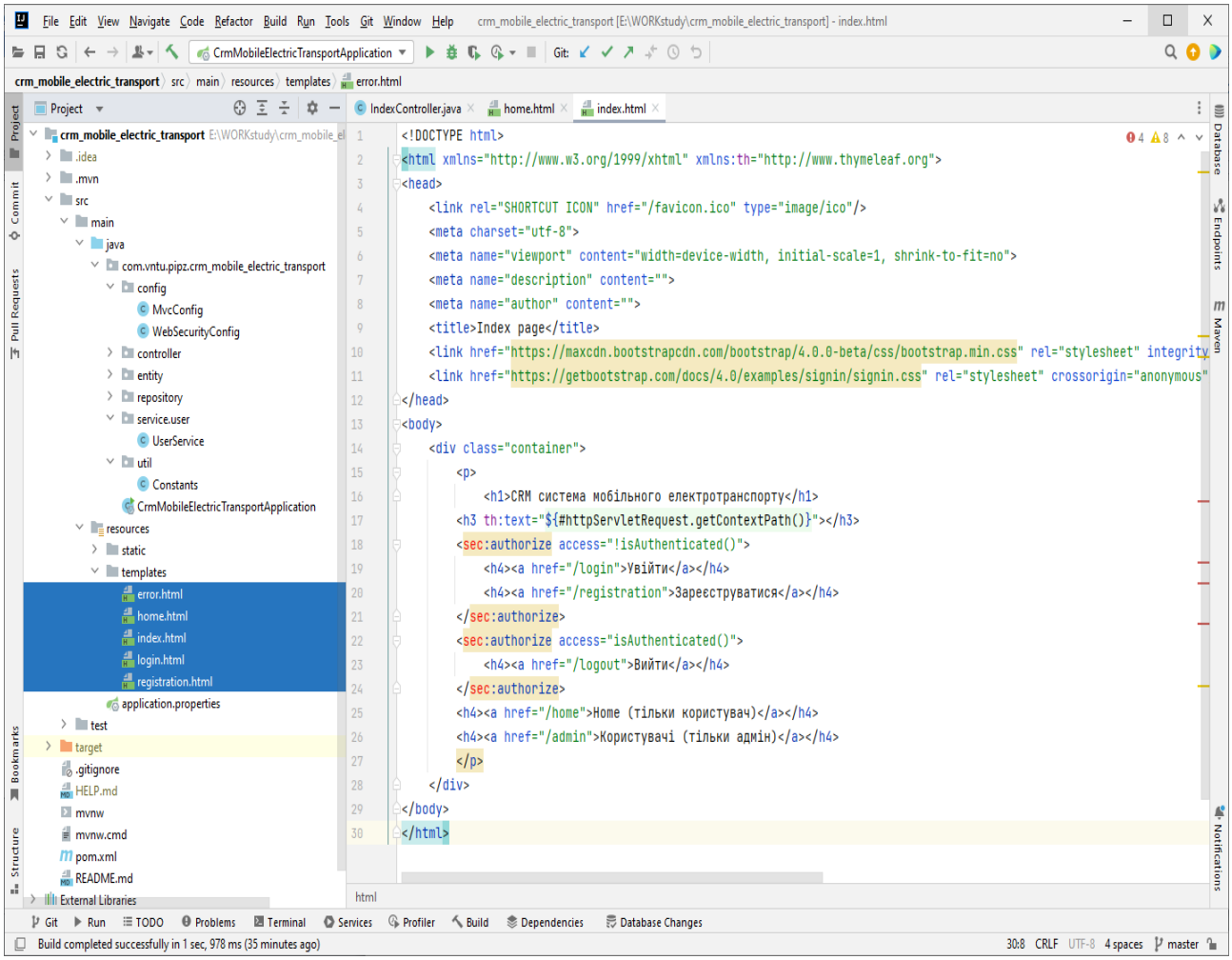


Рисунок 3.15 – HTML шаблон початкової сторінки

Також, було підключено та сконфігуровано Spring boot security для необхідних налаштувань безпеки нашої системи (рис. 3.16).

Весь створений програмний код всіх об’єктів нашої системи більш детально представлений в лістингу (додаток Д).

На рисунку 3.17 представлено успішний старт нашої системи в середовищі розробки IntelliJ IDEA.

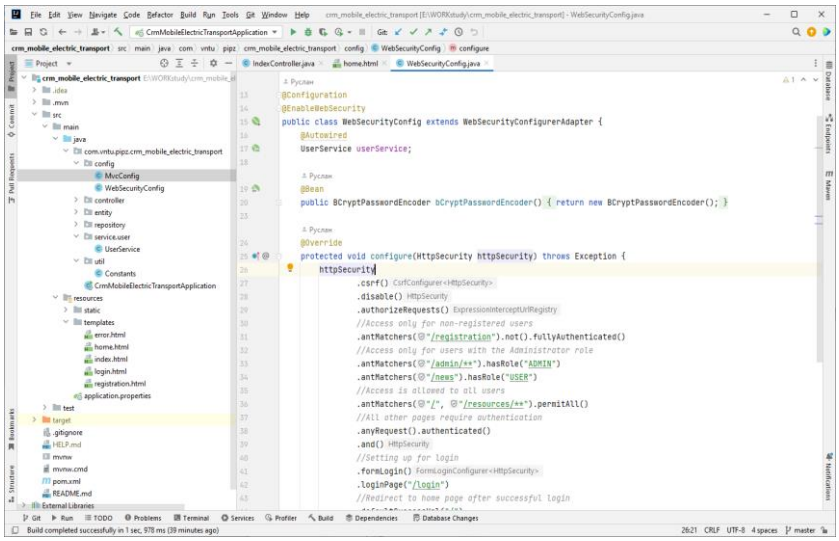


Рисунок 3.16 – Налаштування WebSecurityConfig

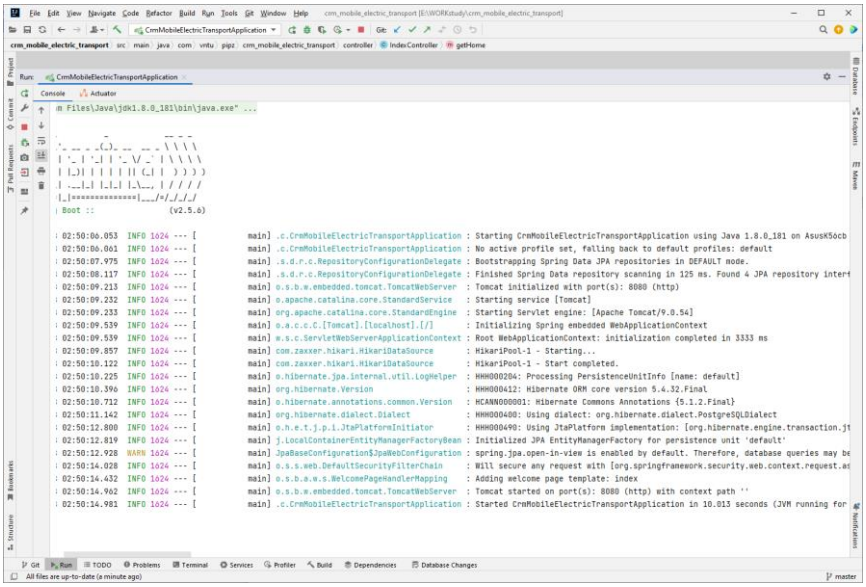


Рисунок 3.17 – Старт CRM системи в IntelliJ IDEA

3.3 Висновки

Отже, в цьому розділі був проведений аналіз та зроблений вибір мови програмування (Java), бази даних (PostgreSQL) та середовища розробки (IntelliJ IDEA).

Також, був описаний процес програмної реалізації CRM мобільного електротранспорту.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСОБУ CRM СИСТЕМИ МОБІЛЬНОГО ЕЛЕКТРОТРАНСПОРТУ

4.1 Аналіз методів і засобів тестування

4.1.1 Етапи тестування

Методологія тестування програмного забезпечення - це систематичний підхід до планування, виконання та аналізу тестів для перевірки якості програмного продукту. Цей процес включає в себе ряд етапів та кроків, які допомагають виявляти помилки та недоліки в програмному забезпеченні. Основні етапи методології тестування можна описати наступним чином [49,50]:

1. Аналіз вимог:
   - ретельний аналіз вимог допомагає зрозуміти очікувані функції та властивості програмного продукту;
   - визначення тестових випадків на основі вимог дозволяє створити план тестування.
2. Планування тестування:
   - визначення обсягу тестування, вибір методів тестування (функціональне, навантажувальне, відмовостійкість, тощо);
   - розробка тестових сценаріїв і тест-кейсів, визначення критеріїв завершення тестування.
3. Розробка тестових сценаріїв та тест-кейсів:
   - створення конкретних тестових сценаріїв для перевірки різних аспектів програмного продукту;
   - визначення вхідних даних, кроків виконання, очікуваних результатів.
4. Виконання тестів:
   - запуск тестових сценаріїв та тест-кейсів;
   - запис результатів тестування, виявлення і реєстрація помилок.

5. Відстеження помилок:
   - створення звітів про виявлені помилки;
   - відстеження процесу виправлення помилок та їхньої перевірки.
6. Автоматизація тестування:
   - використання інструментів автоматизації для виконання повторюваних тестових сценаріїв;
   - Зменшення ручної праці та підвищення ефективності тестування.
7. Валідація та верифікація:
   - валідація - перевірка того, чи програма виконує очікувані функції для кінцевого користувача;
   - верифікація - перевірка відповідності програми специфікаціям.
8. Завершення тестування та звітність:
   - оцінка результатів тестування.
   - підготовка звітів про якість та стабільність програмного продукту.

Процес методології тестування є ітеративним, і його можна повторювати при внесенні змін у програмний продукт або вимоги. Методологія тестування є важливою частиною життєвого циклу розробки програмного забезпечення і сприяє забезпеченню високої якості та надійності продукту.

4.1.2 Методологія тестування

У термінології професійного тестування програмного забезпечення є визначення «тестування білого ящика» і «тестування чорного ящика»[51].

Метод тестування "білого ящика" (white-box testing) програмного забезпечення передбачає аналіз внутрішньої структури програми, коду та алгоритмів з метою виявлення помилок, оптимізації та покращення якості коду. Цей метод також відомий як структурне тестування чи тестування на рівні коду. Основні характеристики методу «білого ящика» включають:

1. Знання внутрішньої структури:
   - тестувальники знають код, структуру даних та логіку програми;

- тдатність розуміти алгоритми та процеси в програмі.

2. Використання коду для створення тестів:
   - тестувальники розробляють тести на основі структури коду;
   - включає тестування функцій, логічних гілок, меж, табличного тестування і т.д.
3. Критерії покриття коду:
   - визначення метрик покриття коду, таких як покриття рядків коду, гілок, умов, інструкцій тощо;
   - гарантує повноту тестування в межах визначених критеріїв.
4. Тестування структури баз даних:
   - включає перевірку ефективності та правильності взаємодії програми з базою даних.
5. Тестування інтеграції:
   - перевірка правильності взаємодії компонентів програми, а також їхніх інтерфейсів.
6. Виділення помилок та вразливостей:
   - може допомогти виявити помилки, які можуть бути пов'язані з неправильним виконанням коду.
   - допомагає в ідентифікації потенційних вразливостей безпеки.
7. Автоматизація тестування:
   - велика частина тестів може бути автоматизована за допомогою спеціалізованих інструментів.

Метод "білого ящика" є ефективним для тестування та виявлення проблем на ранніх етапах розробки, коли доступ до внутрішньої структури програми легко здійснюється. Однак його обмеженням є те, що він часто не виявляє проблеми, пов'язані зі специфікацією та зовнішнім поведінкою програми. Тому часто використовують комбінацію методів "білого ящика" і "чорного ящика" для більш повного тестування програмного продукту.

Метод тестування "чорного ящика" (black-box testing) передбачає випробування програмного забезпечення без врахування внутрішньої структури

коду, алгоритмів та логіки роботи програми. Такий підхід фокусується на тестуванні зовнішнього поведінки системи на основі вхідних даних та очікуваних результатів. Основні характеристики методу "чорного ящика" включають:

1. Не залежить від внутрішньої реалізації:
   - тестувальники не мають доступу до коду програми або його внутрішньої структури;
   - тестування здійснюється лише на основі вхідних та вихідних даних.
2. Орієнтоване на вимоги та специфікації:
   - використовується для перевірки відповідності програми вимогам та специфікаціям;
   - тестувальники розглядають програму як чорний ящик, не знаючи, яким чином вона працює всередині.
3. Тестування функціональності:
   - перевірка функціональності програми без знання її внутрішньої реалізації;
   - переконання в тому, що програмне забезпечення виконує очікувані дії за заданими умовами.
4. Тестування взаємодії та інтеграції:
   - випробування взаємодії між різними компонентами системи;
   - перевірка сумісності та інтеграції з іншими частинами програмного забезпечення.
5. Тестування продуктивності та навантаження:
   - випробування роботи програми при різних умовах завантаження;
   - перевірка швидкодії та ефективності системи.
6. Тестування безпеки:
   - аналіз вразливостей програмного забезпечення з точки зору зовнішнього зловживання;
   - виявлення можливих точок входу для атак.
7. Тестування сумісності:

- перевірка того, як програма працює на різних операційних системах, пристроях чи в різних середовищах.

8. Автоматизоване тестування:
   - можливість автоматизувати багато тестових сценаріїв, що полегшує повторне тестування.

Метод "чорного ящика" дозволяє визначити, чи програма працює так, як очікується від користувача, і виявити проблеми на рівні зовнішньої функціональності та інтерфейсів. Однак цей підхід може не виявити деякі внутрішні проблеми, які можуть бути виявлені методом "білого ящика". Тому комбінація обох методів забезпечує більш повне тестування програмного продукту.

4.1.3 Рівні тестування

Тестування програмного забезпечення включає різні рівні, які дозволяють перевірити його функціональність, продуктивність та інші характеристиких[52].

Модульне тестування є одним з основних рівнів тестування програмного забезпечення і спрямоване на перевірку індивідуальних модулів чи компонентів програми. Модуль може бути найменшою одиницею програми, яка може бути самостійною, тестирувати або обслуговувати. Основні принципи та характеристики модульного тестування включають:

1. Орієнтація на фрагменти коду – тестування проводиться на рівні індивідуальних модулів або функцій програмного коду.
2. Автоматизація – модульні тести часто автоматизовані для ефективності та повторюваності.
3. Ізоляція – модулі тестуються ізольовано, щоб впевнитися, що їхні функції працюють коректно незалежно від інших частин програми.
4. Вхідні та вихідні дані – тести включають в себе введення конкретних вхідних даних та перевірку очікуваних вихідних результатів.

5. Спрощення виявлення помилок – модульне тестування допомагає виявити та виправити помилки на ранніх етапах розробки, що зменшує вартість виправлення проблем в подальшому.
6. Підтримка рефакторингу – дозволяє розробникам змінювати або вдосконалювати код (рефакторити) і перевіряти, чи це не порушує існуючу функціональність.
7. Сприяє відлагодженню – допомагає виявляти й усувати помилки та недоліки на ранніх етапах розробки, полегшуючи відлагодження.

Модульне тестування є важливою складовою розробки програмного забезпечення, оскільки воно сприяє підвищенню надійності і якості коду, а також забезпечує більш ефективне управління ризиками в розробці програм.

Інтеграційне тестування програмного забезпечення (Integration Testing) є наступним рівнем після модульного тестування і спрямоване на перевірку взаємодії між різними модулями або компонентами програми, коли вони об'єднуються. Основні аспекти інтеграційного тестування включають:

1. Тести інтерфейсів – перевірка взаємодії між різними модулями через їхні інтерфейси.
2. Взаємодія між модулями – тести оцінюють, як модулі взаємодіють один з одним, особливо коли вони об'єднуються для створення більш складної функціональності.
3. Підсистеми та компоненти – перевірка взаємодії підсистем, компонентів або сервісів в межах програми.
4. Передача даних – тести перевіряють правильність передачі даних між різними частинами програми.
5. Тестування функціональності об'єднаних частин – перевірка, чи працює функціональність вже об'єднаних частин програми як очікується.
6. Тестування інтерфейсів зовнішніх систем – якщо програма взаємодіє з іншими системами, тестування може включати перевірку цієї взаємодії.

7. Регресійне тестування – перевірка, чи нові зміни не порушують функціональність, яка вже була протестована.
8. Тестування виняткових ситуацій – тести оцінюють реакцію системи на несподівані обставини або помилки.

Інтеграційне тестування допомагає впевнитися в тому, що різні компоненти програми працюють правильно разом, забезпечуючи коректну та стабільну роботу програмного забезпечення. Цей етап тестування є важливим для виявлення проблем, які можуть виникнути в результаті взаємодії між різними частинами системи.

Системне тестування програмного забезпечення (System Testing) – це рівень тестування, який спрямований на перевірку функціональності і характеристик системи в цілому. Основні аспекти системного тестування включають:

1. Функціональність – перевірка того, чи виконує програмне забезпечення всі функції, визначені в специфікаціях та вимогах.
2. Визначення відповідності вимогам – впевненість у тому, що програмне забезпечення відповідає всім вимогам замовника чи користувача.
3. Продуктивність – оцінка швидкодії та ефективності програми в умовах реального використання.
4. Стабільність – тестування стійкості програмного забезпечення в умовах різноманітних сценаріїв використання.
5. Безпека – перевірка системи на вразливість до зловмисних атак та забезпечення безпеки даних.
6. Тестування сумісності – перевірка взаємодії програми з різними операційними системами, браузерами та іншими зовнішніми факторами.
7. Тестування відмовостійкості – визначення того, як система реагує на відмови та відновлення після них.
8. Тестування відновлення – перевірка можливості відновлення системи після збоїв або відмов.

9. Тестування користувацького інтерфейсу – перевірка зручності та інтуїтивності інтерфейсу користувача.
10. Тестування документації – впевненість, що документація, пов'язана з програмним забезпеченням, відповідає реальному стану системи.

Системне тестування проводиться в середовищі близькому до реального, з метою перевірки всіх аспектів функціональності і надійності програмного продукту. Цей етап тестування допомагає виявити проблеми, які можуть виникнути тільки при взаємодії різних частин системи в комплексі.

Динамічне тестування програмного забезпечення — це вид тестування, який включає в себе виконання програми або системи та оцінку її поведінки в реальному часі. Цей підхід активно використовується для перевірки функціональності, продуктивності та надійності програм, а також для виявлення помилок та дефектів у коді.

Альфа-тестування є одним із етапів тестування програмного забезпечення і проводиться після внутрішнього (внутрішньої альфа-версії) та перед бета-тестуванням. Це етап тестування, який включає в себе реальних користувачів або внутрішніх співробітників компанії-розробника.

Бета-тестування програмного забезпечення – це етап тестування, який відбувається після альфа-тестування та перед випуском фінальної версії програми. Під час бета-тестування програма стає доступною для обмеженого кола зовнішніх користувачів або тестерів. Головною метою бета-тестування є збір повідомлень про можливі помилки, отримання відгуків користувачів і перевірка працездатності програми в реальних умовах використання.

Основні аспекти бета-тестування включають:

1. Обмежений доступ – програма доступна обмеженому колу зовнішніх користувачів або тестерів, які мають можливість використовувати програму в реальних умовах.
2. Розгортання в реальному середовищі – випробування програми в реальних умовах використання, що може призвести до виявлення

проблем, які було важко або неможливо визначити на етапі внутрішнього тестування.

3. Збір відгуків користувачів – отримання відгуків користувачів щодо функціональності, ефективності, інтерфейсу та інших аспектів програми.
4. Виявлення та виправлення дефектів – виявлення потенційних дефектів і помилок, які не було виявлено під час внутрішнього тестування.
5. Оцінка стійкості і продуктивності – оцінка стабільності та продуктивності програми при реальному використанні.
6. Підготовка до релізу – збір інформації, яка допоможе вдосконалити продукт перед його офіційним випуском.

Бета-тестування є важливим кроком у виправленні помилок і вдосконаленні програмного забезпечення перед його остаточним випуском на ринок. Відгуки користувачів, отримані під час бета-тестування, можуть бути використані для покращення функціональності і вирішення проблем.

Приймальне тестування (Acceptance Testing) – це фаза тестування програмного забезпечення, яка визначає, чи відповідає продукт заданим вимогам та чи може бути прийнятий замовником або користувачем. Це останній етап тестування перед впровадженням програми в експлуатацію. Приймальне тестування може включати декілька різновидів тестів:

1. Тестування відповідності вимогам (Requirements-based testing) – перевірка, чи функціонал програми відповідає усім вимогам та специфікаціям, що були визначені на етапі розробки.
2. Тестування коректності функціональності – перевірка правильності реалізації основних функцій програми з точки зору користувача.
3. Тестування відповідності користувацьким потребам – перевірка, чи програма відповідає очікуванням та потребам користувачів.
4. Тестування сценаріїв використання – перевірка, як програмне забезпечення працює в реальних умовах використання, включаючи різні сценарії та послідовності взаємодії.

5. Тестування продуктивності – оцінка швидкодії та продуктивності програми під час реального використання.
6. Тестування безпеки – визначення, наскільки система захищена від зовнішніх загроз та атак.
7. Тестування встановлення та відновлення – перевірка процедур встановлення та відновлення програмного забезпечення.
8. Тестування взаємодії з іншими системами – перевірка, як програма взаємодіє з іншими системами, якщо це є частиною її функціоналу.

Приймальне тестування може бути проведене замовником або представниками замовника, і його успішне завершення підтверджує готовність програмного забезпечення до впровадження в експлуатацію. У разі виявлення проблем на цьому етапі може бути прийняте рішення про внесення коректив до програми або відкладення впровадження до виправлення дефектів.

Регресійне тестування програмного забезпечення (Regression Testing) – це вид тестування, спрямований на перевірку того, чи нові зміни в програмному коді не порушують існуючий функціонал. Його основною метою є впевненість у тому, що внесені зміни або додатки нового функціоналу не вплинули на вже перевірені та працюючі частини програми.

Основні аспекти регресійного тестування включають:

1. Перевірка існуючого функціоналу – тестування основного функціоналу програми для виявлення можливих проблем, які можуть виникнути через внесені зміни.
2. Тестування інтеграції – перевірка взаємодії між різними компонентами або модулями після внесених змін.
3. Повторне виконання попередніх тестів – запуск тестів, які були успішно пройдені на попередніх етапах розробки для перевірки, чи продовжують вони працювати коректно.
4. Автоматизація тестування – використання автоматизованих тестів для швидшого та ефективнішого виконання регресійних тестів.

5. Виявлення нових дефектів – перевірка, чи внесені зміни не викликали появу нових дефектів або небажаних побічних ефектів.
6. Тестування сумісності – перевірка, чи внесені зміни не вплинули на сумісність програми з іншими системами чи платформами.
7. Оновлення тестових кейсів – оновлення тестових кейсів у випадку внесення змін у функціонал програми.

Регресійне тестування особливо важливе при регулярних випусках нових версій програмного забезпечення чи при внесенні змін у код під час розробки. Воно допомагає підтримувати якість програми в умовах швидких змін і забезпечує вчасне виявлення та виправлення проблем.

Димове тестування (Smoke Testing), також відоме як Build Verification Testing (BVT) або Sanity Testing, є видом тестування програмного забезпечення, який призначений для швидкої перевірки основної функціональності та стабільності системи. Це перевірка, яка виконується перед розпочаттям більш глибокого тестування для переконання, що базові функції працюють вірно.

Основні характеристики димового тестування включають:

1. Спрощений сценарій – димове тестування виконує спрощений сценарій або невеликий набір сценаріїв для перевірки базової функціональності.
2. Перевірка основних функцій – основна мета - перевірити, чи може програмне забезпечення запускатися та чи працюють основні функції без серйозних помилок.
3. Безпека та стабільність – перевірка стабільності програми та виявлення критичних проблем або дефектів, які можуть перешкоджати нормальному функціонуванню системи.
4. Швидке виконання – димове тестування проводиться дуже швидко, і його результати використовуються для визначення готовності системи до більш детального тестування.
5. Перевірка збирання – впевнення, що зібраний білд (версія програми) встановлюється та запускається на початковому етапі тестування.

Димове тестування допомагає виявити великі проблеми або критичні дефекти на ранніх етапах розробки або перед важливими випусками програмного продукту. Це основний крок для впевненості в тому, що основна функціональність працює, і тестувальна команда може перейти до більш розгорнутого тестування.

4.1.4 Види тестування

Існує багато видів тестування програмного забезпечення, які використовуються для різних цілей та на різних етапах розробки[53].

Функціональне тестування програмного забезпечення (Functional Testing) – це вид тестування, який спрямований на перевірку функціональності програми для того, щоб переконатися, що вона працює відповідно до визначених вимог та очікувань. Цей вид тестування оцінює те, як програма взаємодіє з користувачем, базується на вхідних даних та поводиться у різних сценаріях використання.

Основні аспекти функціонального тестування включають:

1. Тестування основної функціональності – перевірка відповідності програми визначеним вимогам та специфікаціям.
2. Тестування входів і виводів – перевірка коректності обробки вхідних даних та вірності вивідних результатів.
3. Тестування сценаріїв використання – оцінка роботи програми в реальних умовах, включаючи різні сценарії використання.
4. Тестування користувацького інтерфейсу – перевірка зручності та ефективності інтерфейсу для користувачів.
5. Тестування функцій управління – перевірка функцій управління, таких як додавання, видалення, редагування об'єктів.
6. Тестування системи оповіщення та повідомлень – перевірка системи оповіщення та повідомлень на коректність та актуальність.

7. Тестування дотримання стандартів і гарантій якості – перевірка того, чи програма відповідає стандартам та нормативам, а також чи гарантує високу якість виконання.
8. Тестування виняткових ситуацій – перевірка поведінки програми в умовах виняткових ситуацій або помилок.
9. Тестування сумісності – перевірка сумісності програми з різними операційними системами, пристроями та іншими складовими системи.

Функціональне тестування може бути виконане як вручну, так і автоматизовано, використовуючи спеціальні інструменти для створення та виконання тестових сценаріїв. Воно є важливим компонентом процесу забезпечення якості програмного забезпечення і допомагає виявляти та виправляти дефекти, що стосуються функціональності продукту.

Тестування продуктивності програмного забезпечення (Performance Testing) є важливим етапом у забезпеченні якості програми та її ефективності під час роботи в реальних умовах. Метою цього виду тестування є визначення та оцінка різних аспектів продуктивності, таких як швидкодія, стійкість, навантаження та ефективність в різних умовах.

Основні види тестування продуктивності включають:

1. Тестування швидкодії (Speed Testing) – визначення часу реакції програми на конкретні дії користувача або події.
2. Тестування завантаження (Load Testing) – визначення того, як програмне забезпечення працює при підвищеному навантаженні, яке перевищує очікувані користувацькі вимоги.
3. Тестування стресостійкості (Stress Testing) – випробування програми на максимальному або надмірному навантаженні для визначення меж її працездатності та стійкості.
4. Тестування масштабованості (Scalability Testing) – оцінка здатності програми масштабуватися та обробляти зростаюче число користувачів або об'єм даних.

5. Тестування продуктивності баз даних – визначення швидкодії та надійності бази даних при роботі з великою кількістю даних та великою кількістю одночасних запитів.
6. Тестування пропускної здатності (Throughput Testing) – визначення кількості операцій, які програма може обробляти за певний період часу.
7. Тестування ефективності роботи пам'яті – визначення використання оперативної пам'яті та роботи програми при великому обсязі даних.
8. Тестування стійкості (Reliability Testing) – визначення стійкості програми до великої тривалості роботи без помилок та витоків пам'яті.

Ці види тестування допомагають розробникам та тестувальникам зрозуміти, як програма веде себе під час реального використання та як добре вона справляється з різними умовами навантаження. Тестування продуктивності є важливою частиною процесу забезпечення якості програмного забезпечення, особливо для великих та складних систем.

Стресове тестування програмного забезпечення (Stress Testing) – це вид тестування продуктивності, який випробовує програму або систему при екстремальних умовах навантаження, що перевищують межі її нормального функціонування. Метою стресового тестування є виявлення меж працездатності, ідентифікація слабких місць та визначення того, як програмне забезпечення реагує на надмірне навантаження.

Основні аспекти стресового тестування включають:

1. Велике навантаження – створення умов для обробки програмою або системою максимально великої кількості користувачів, запитів чи транзакцій.
2. Перевищення ресурсів – забезпечення ситуацій, коли програма витрачає всі доступні ресурси, такі як процесорний час, оперативна пам'ять, дисковий простір і мережева пропускна здатність.

3. Тривале навантаження – випробування стійкості програми або системи під час тривалого навантаження, що може визначатися годинами або навіть днями.
4. Зміна навантаження – створення ситуацій, коли навантаження змінюється динамічно, від низького до високого та навпаки, для визначення, як система адаптується до зміни обсягу роботи.
5. Тестування в крайніх умовах – створення умов, коли в систему введені непридатні або неправдоподібні дані.
6. Виявлення ідентифікації слабких місць – виявлення дефектів, помилок або несправностей, які можуть виникнути під час стресового навантаження.
7. Оцінка відновлюваності – перевірка того, наскільки добре програма відновлюється після завершення екстремальних умов навантаження.

Стресове тестування дозволяє визначити межі витривалості і працездатності програмного забезпечення та ідентифікувати проблеми, які можуть виникнути під час великого навантаження або стресових умов. Це важливий етап у впровадженні надійних та стабільних програм, особливо в ситуаціях, коли система використовується в умовах великого обсягу користувачів або великого обсягу даних.

Тестування навантаження програмного забезпечення (Load Testing) – це вид тестування продуктивності, який визначає, як програма чи система поводиться під час обробки очікуваного навантаження. Основна мета полягає в тому, щоб визначити максимальні межі працездатності, ідентифікувати слабкі місця та з'ясувати, чи може програма витримати навантаження, що перевищує звичайне користувацьке використання.

Основні аспекти тестування навантаження включають:

1. Максимальне навантаження – створення умов, при яких програма опиняється під максимальним навантаженням, яке може виникнути при реальному використанні.

2. Тестування під піковим навантаженням – визначення того, як програма веде себе під час виникнення пікового навантаження або підвищеного трафіку.
3. Завантаження системи в терміні – випробування програми протягом тривалого періоду, щоб визначити, чи збільшується споживання ресурсів та чи зберігається стійкість системи.
4. Тестування максимального одночасного користування – визначення максимальної кількості користувачів, яких може обслуговувати програма одночасно без втрати продуктивності.
5. Виявлення меж ресурсів – визначення того, як програма веде себе при досягненні ліміту ресурсів, таких як CPU, пам'ять, мережа тощо.
6. Тестування змінюваного навантаження – створення сценаріїв, коли навантаження динамічно змінюється, щоб оцінити адаптивність системи.
7. Тестування в період піків користування – проведення тестування у періоди максимального користування, такі як розпродажі, рекламні акції тощо.
8. Виявлення ідентифікації слабких місць – виявлення проблем, які можуть виникнути під час великого обсягу одночасного користування.

Цей вид тестування допомагає розробникам та тестувальникам переконатися, що програма або система може ефективно функціонувати під час максимального або надзвичайного користувацького навантаження, а також ідентифікувати та усунути можливі проблеми продуктивності.

Тестування стабільності програмного забезпечення (Stability Testing) спрямоване на визначення того, наскільки програма або система залишається стабільною та надійною під час тривалої роботи чи під навантаженням. Мета полягає в тому, щоб виявити можливі проблеми, такі як витоки пам'яті, некоректне вивільнення ресурсів, падіння системи або інші аномалії, які можуть виникнути під час тривалого функціонування програми.

Основні аспекти тестування стабільності включають:

1. Тривалість роботи – визначення того, як програма веде себе під час тривалого періоду безперервної роботи.
2. Тестування роботи під навантаженням – випробування програми або системи під значним навантаженням, щоб переконатися, що вона залишається стабільною.
3. Виявлення витоків пам'яті – перевірка, чи програма правильно використовує та вивільнює пам'ять, та виявлення можливих витоків.
4. Тестування в умовах обмежених ресурсів – спроби виконати програму в умовах обмежених ресурсів, таких як обмежена кількість пам'яті або обчислювальна потужність.
5. Перевірка стабільності при змінах у середовищі – випробування програми під час зміни умов середовища, таких як зміни конфігурації, апдейти операційної системи чи інші фактори.
6. Тестування збереження стану – перевірка, чи програма може коректно відновлювати свій стан після можливих збоїв чи падінь.
7. Тестування міжпрограмних взаємодій – випробування стабільності програми в умовах взаємодії з іншими програмами або компонентами системи.
8. Виявлення критичних помилок – ідентифікація та усунення критичних помилок, які можуть призвести до падінь чи некоректної роботи програми.

Тестування стабільності допомагає забезпечити, що програмне забезпечення може ефективно працювати в реальних умовах та підтримувати стабільну роботу протягом тривалого періоду часу. Особливо важливо проводити таке тестування для програм, які піддаватимуться тривалому користуванню або мають велику кількість користувачів.

Тестування зручності використання програмного забезпечення (Usability Testing) спрямоване на оцінку того, наскільки програма є зручною та легкою у використанні для кінцевих користувачів. Мета цього виду тестування полягає в

тому, щоб забезпечити, що програма відповідає потребам користувачів і надає їм задоволення від взаємодії з нею.

Основні аспекти тестування зручності використання включають:

1. Навігація – оцінка легкості та зрозумілості навігаційної структури програми.
2. Графічний інтерфейс – оцінка дизайну та естетичних характеристик графічного інтерфейсу, включаючи колірну схему, шрифти та іконки.
3. Взаємодія з користувачем – визначення того, наскільки ефективно користувачі можуть взаємодіяти з програмою.
4. Зрозумілість введення та виведення – перевірка того, наскільки зрозуміло користувачеві, як вводити дані та сприймати виведення програми.
5. Доступність – оцінка того, наскільки програма доступна для користувачів з різними потребами та особливостями.
6. Документація та допомога – перевірка наявності та якості документації, а також наявності систем підтримки та допомоги користувачам.
7. Простота використання – оцінка того, наскільки легко нові користувачі можуть оволодіти програмою без додаткового навчання.
8. Тестування на різних платформах – визначення того, як добре програма працює на різних пристроях та платформах.
9. Інтернаціоналізація та локалізація – перевірка того, наскільки програма підтримує різні мови та регіональні налаштування.

Тестування зручності використання є важливим етапом в процесі забезпечення якості програмного забезпечення, оскільки забезпечує, що програма відповідає очікуванням та потребам користувачів. Зручна та легка у використанні програма сприяє підвищенню задоволення користувачів і може

Тестування інтерфейсу користувача (UI testing або GUI testing) – це процес визначення, чи працює графічний інтерфейс програмного забезпечення

відповідно до специфікацій та очікувань користувачів. Це включає в себе оцінку вигляду, поведінки та взаємодії всіх компонентів і елементів у програмі.

Основні аспекти тестування інтерфейсу користувача включають:

1. Вигляд та розміщення елементів – перевірка вигляду та розташування всіх елементів інтерфейсу, таких як кнопки, поля введення, меню і т.д.
2. Керування елементами – перевірка можливості взаємодії з елементами інтерфейсу, такими як клікання, введення тексту, перетягування та ін.
3. Навігація – тестування можливостей користувача пересуватися між різними сторінками, вікнами, вкладками та іншими частинами інтерфейсу.
4. Сумісність з браузерами та платформами – перевірка того, як добре інтерфейс працює на різних браузерах та платформах.
5. Реакція на різні роздільні здатності – тестування реакції інтерфейсу на зміни роздільної здатності екрану та розмір вікна.
6. Тестування обробки помилок – перевірка коректності повідомлень про помилки та їх обробки в інтерфейсі.
7. Безпека і конфіденційність – тестування на витоки конфіденційної інформації та забезпечення, що інтерфейс не вразливий для атак.
8. Доступність – перевірка того, наскільки інтерфейс доступний для користувачів з різними потребами та можливостями.
9. Локалізація та інтернаціоналізація – тестування того, як програма взаємодіє з різними мовами та регіональними налаштуваннями.
10. Тестування реакції на зміни у конфігурації – перевірка того, як програма взаємодіє з різними конфігураціями обладнання та налаштувань.

Тестування інтерфейсу користувача є ключовою частиною процесу забезпечення якості програмного забезпечення, оскільки графічний інтерфейс є основним шляхом взаємодії користувачів з програмою.

Тестування безпеки програмного забезпечення (Security Testing) – це процес визначення вразливостей та потенційних ризиків безпеки в

програмному забезпеченні. Мета тестування безпеки полягає в тому, щоб виявити та усунути потенційні слабкі місця, які можуть бути використані зловмисниками для несанкціонованого доступу, втрати конфіденційності, цілісності чи доступу до ресурсів системи.

Основні аспекти тестування безпеки включають:

1. Тестування вразливостей – визначення потенційних вразливостей в програмному забезпеченні, таких як недоліки в коді, невірні налаштування безпеки або недостатня обробка введених даних.
2. Тестування на проникнення (Penetration Testing) – спроби проникнути в систему чи додаток з метою виявлення слабкі місця та визначення, як зловмисники можуть отримати несанкціонований доступ.
3. Аудит безпеки коду – аналіз вихідного коду програми для виявлення можливих проблем безпеки та рекомендацій щодо виправлення.
4. Тестування аутентифікації та авторизації – перевірка механізмів аутентифікації (підтвердження ідентичності користувача) та авторизації (контроль доступу до ресурсів).
5. Тестування мережевої безпеки – аналіз мережевої архітектури та виявлення можливих атак або слабкі місця в мережевій безпеці.
6. Тестування безпеки введених даних – виявлення можливостей для атак на введені дані, такі як SQL-ін'єкції, кросс-сайтовий скриптінг тощо.
7. Тестування безпеки сесій та кукі-файлів – визначення можливих атак на сесії користувачів та захист від них.
8. Тестування на захист від витоку інформації – визначення та усунення можливих витоків конфіденційної інформації.
9. Тестування на захист від зловживання функціоналу – визначення та захист від можливостей для зловживання функціональністю програми.
10. Тестування на захист від DoS-атак (Denial of Service) – аналіз та захист від атак, спрямованих на переповнення ресурсів системи для відмови у обслуговуванні.

Тестування безпеки є критичним етапом в життєвому циклі розробки програмного забезпечення, оскільки допомагає попередити можливі загрози та забезпечити високий рівень безпеки для користувачів та даних.

Тестування локалізації програмного забезпечення – це процес перевірки того, наскільки ефективно програма підтримує мови та регіональні налаштування, забезпечуючи коректне відображення тексту, форматування дат та чисел, підтримку різних алфавітів та інші аспекти, які забезпечують коректне використання програми в різних культурних та мовних середовищах.

Основні етапи тестування локалізації включають:

1. Перевірка перекладу – перевірка точності та коректності перекладів усіх текстових рядків програми на вибрані мови.
2. Форматування дат та чисел – перевірка того, як програма відображає та обробляє дати та числа з урахуванням локальних налаштувань.
3. Взаємодія з алфавітами різних мов – тестування коректності введення, відображення та обробки тексту на різних алфавітах, таких як кирилиця, латиниця, китайська, японська тощо.
4. Локалізовані графічні елементи – перевірка, як коректно відображаються графічні елементи та зображення, які можуть бути змінені в залежності від мови.
5. Тестування різних кодувань символів – перевірка коректності відображення та обробки тексту в різних кодуваннях символів, таких як UTF-8, UTF-16, інші многобайтові та однобайтові кодування.
6. Адаптація до місцевих звичаїв та стандартів – перевірка, як програма адаптується до місцевих звичаїв, таких як форматування валюти, використання метричної чи британської системи мір, відміток часу та ін.
7. Тестування мовних перемикачів – тестування можливості користувача змінювати мову інтерфейсу, переключати регіональні налаштування та перевірка коректності змін.

8. Адаптивність до розмірів тексту – тестування того, як програма адаптується до різних розмірів тексту в залежності від вибраної мови.

Тестування локалізації важливо для забезпечення того, що програма буде коректно та зручно працювати для користувачів у різних культурних та мовних середовищах. Забезпечення високої якості локалізації покращує користувацький досвід та зростання популярності продукту на ринку.

Тестування сумісності програмного забезпечення – це процес визначення того, як ефективно програма працює на різних платформах, операційних системах, браузерах, пристроях чи середовищах взагалі. Мета цього виду тестування полягає в забезпеченні того, щоб програма була функціональною та ефективною на різних конфігураціях та для різних типів користувачів.

Основні аспекти тестування сумісності включають:

1. Операційні системи – перевірка сумісності програми з різними операційними системами, такими як Windows, macOS, Linux, iOS, Android тощо.
2. Версії операційних систем – тестування програми на різних версіях операційних систем, оскільки програми можуть вести себе по-різному на різних версіях.
3. Браузери – визначення того, як програма взаємодіє з різними веб-браузерами, такими як Chrome, Firefox, Safari, Edge, Internet Explorer тощо.
4. Розширення та оновлення браузерів – перевірка, як програма реагує на оновлення та нові версії веб-браузерів та їх розширення.
5. Апаратні платформи – тестування на різних апаратних платформах, таких як комп'ютери, ноутбуки, смартфони, планшети та інші.
6. Роздільна здатність екрану – визначення того, як програма виглядає та працює на різних роздільних здатностях екрану.
7. Мережеві умови – тестування програми в різних умовах мережі, включаючи швидкість з'єднання та стабільність мережі.

8. Сумісність зі стороннім програмним забезпеченням – перевірка того, як програма взаємодіє з іншими програмами або сервісами, зокрема, визначення сумісності з популярними програмами.
9. Апаратні вимоги – визначення, чи відповідає програма апаратним вимогам для різних конфігурацій.
10. Тестування на різних мовах та локалізації – перевірка, як програма працює на різних мовах та підтримує місцеві налаштування.

Тестування сумісності важливо для забезпечення того, що програма буде доступною та ефективною для широкого кола користувачів, які можуть використовувати різні пристрої та операційні системи.

4.2 Етапи тестування програмного засобу

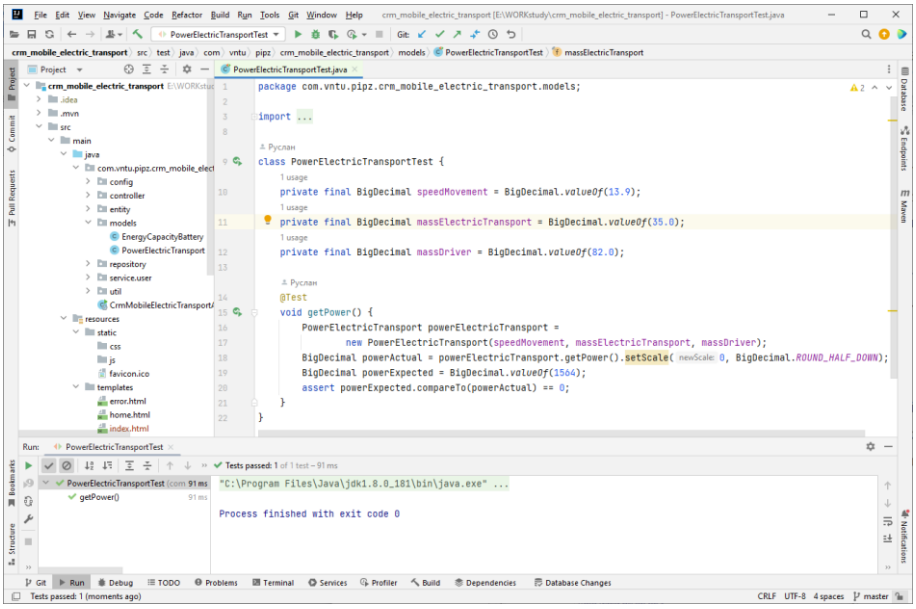
Як видно з проведеного аналізу методології – процес тестування програмного забезпечення вимагає великих ресурсних затрат. У компаніях, які спеціалізуються на розробці програмного забезпечення, цей процес може займати до 60 - 70 відсотків часу та ресурсів.

Виходячи із задач поставлених у даній кваліфікаційній роботі нам потрібно створити прототип робочої версії програмного засібу з мінімальним запланованим функціоналом та мініміальними затратами.

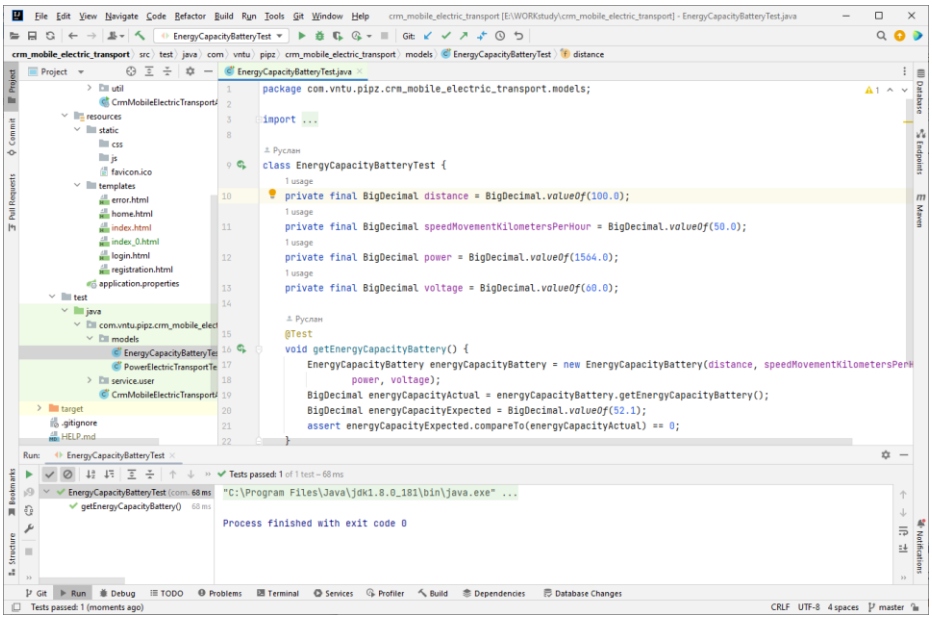
Оскільки, в нашому випадку відсутній штат працівників для проведення тестування програмного забезпечення, то на першому етапі тестування ми застосували принцип «тестування білого ящика", який був виконаний розробником в процесі реалізації програмного засобу. Для цього було використане автоматичне модульне тестування, за допомогою створених розробником «юніт-тестів». Головною цілю модульного тестування нашого програмного засобу – було покрити тестами 80-90 відсотків публічних методів наших класів, які відповідають за бізнес логіку нашої системи. Це дало можливість на етапі збірки артефакту виявляти певні невідповідності в

функціонуванні нашого продукту та виправляти їх. Також, це дає можливість виявити завчасно помилки в процесі розширення та модифікації функціоналу модулів нашого програмного засобу (рис. 4.5).

Модульними тестами було покрито наші моделі для розрахунку потужності електротранспорту та енергетичної ємності батареї (рис. 4.2 та 4.3).



Рисунку 4.2 – Успішне виконання тесту розрахунку потужності



Рисунку 4.3 – Успішне виконання тесту розрахунку ємності батареї

Наступним етапом було мануальне тестування кожного ендпойнта модулів нашої системи за допомогою Postman платформи API для розробників.

Це дало можливість перевірити коректність функціонування нашого програмного засобу (рис. 4.6).

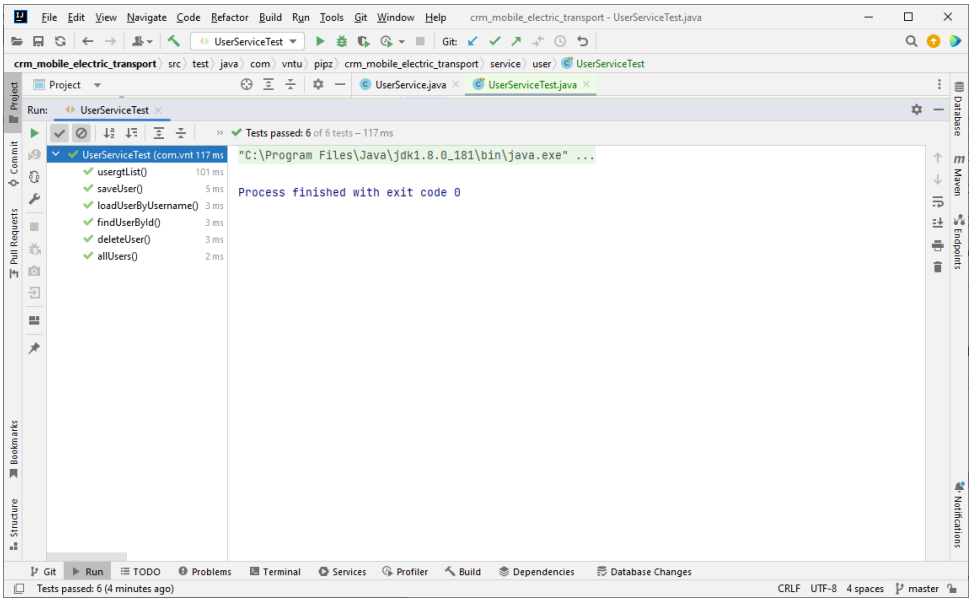


Рисунок 4.5 – Успішне виконання юніт-тестів класу UserService

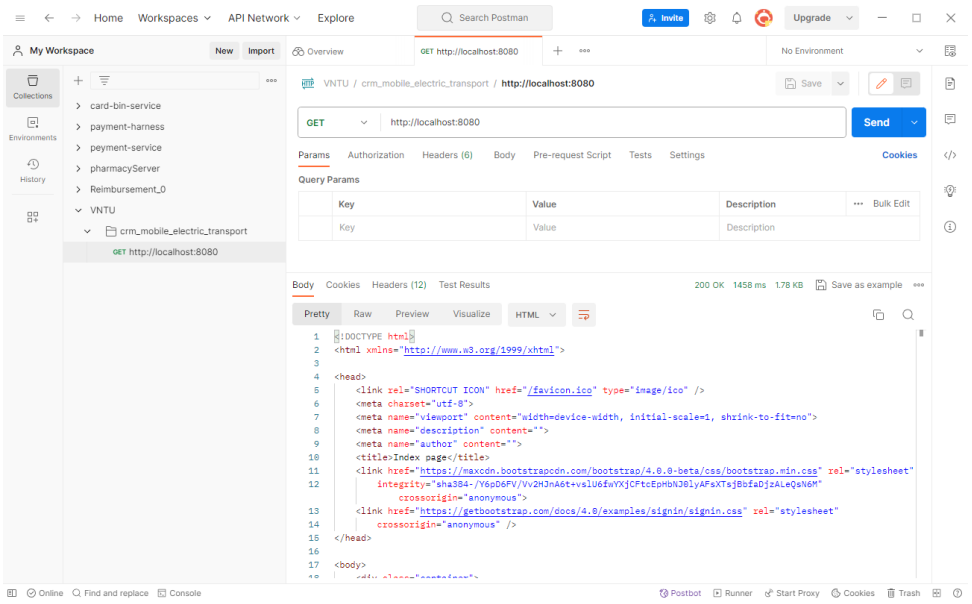


Рисунок 4.6 – Тестування ендпойнтів за допомогою Postman

Після тестування ендпойнтів, в нашій системі був реалізований шар відображення, що дало можливість в якості клієнта нашого web-сервера CRM

системи використати web-браузер. На цьому етапі було здійснено мануальне тестування користувацького інтерфейса в процесі взаємодії з нашою системою з використанням принципу «чорного ящика».

4.3 Висновки

Отже, в даному розділі було проаналізовано методологію тестування програмного забезпечення. Це дало можливість обрати оптимальні шляхи для тестування програмного забезпечення CRM системи мобільного електротранспорту.

Також, було здійснено поетапне тестування нашого програмного забезчення з використанням принципу «білого ящика», де використовувалися модульне тестування юніт-тестами та мануальне тестування ендпойнтів за допомогою Postman платформи API.

За принципом «чорного ящика» було зроблено тестування користувацького інтерфейсу з використанням web-браузера, в якості клієнта нашої системи.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного аудиту є оцінювання комерційного потенціалу впровадження методів та засобів, розробленої CRM системи мобільного електротранспорту.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів Вінницького національного технічного університету к.т.н., доцента Майданюка В. П., к.т.н., доцента Ракитянську Г. Б., к.т.н., доцента Рейду О.М з кафедри програмного забезпечення.

Аудит науково-технічної розробки та її комерційного потенціалу проведено за допомогою таблиці 5.1, застосовуючи п’ятибальну шкалу оцінювання за 12-ма критеріями оцінки.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
1	2	3	4	5	6
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження табл. 5.1.

1	2	3	4	5	6
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження табл. 5.1.

1	2	3	4	5	6
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

В таблиці 5.2 наведено результати оцінювання науково-технічного рівня і комерційного потенціалу розробки.

Таблиця 5.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1. Майданюк В. П.	2. Ракитянська Г. Б.	3. Рейда О. М.
	Бали, виставлені експертами:		
1	2	4	1
2	3	2	3
3	3	2	2
4	2	1	3
5	1	3	3
6	3	3	3
7	1	4	4
8	2	3	2
9	3	3	3
10	3	3	2
11	3	3	4
12	4	1	1
Сума балів	$СБ_1$=31	$СБ_2$=33	$СБ_3$=30
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{30+32+31}{3} = 31$		

В таблиці 5.3 наведено шкалу оцінки комерційного потенціалу розробки.

Таблиця 5.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього

41-48	Високий

За результатами розрахунків, наведених в таблиці 5.2 та шкалою оцінки наведеної в таблиці 5.3 можна зробити висновок щодо рівня комерційного потенціалу розробки. Середньоарифметична сума балів, виставлених експертами склала 31, що відповідає рівню «вище середнього».

Даний рівень комерційного потенціалу можливий за рахунок попиту на системи автоматизації взаємодії з клієнтами, що дає можливість бізнесу збільшувати об’єми реалізції продукції, що в свою чергу збільшує їх прибуток.

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи можна розрахувати за наступними статтями:

- Витрати на оплату праці;
- Витрати на інструментальне та інфраструктурне програмне забезпечення;
- Амортизаційні відрахування;
- Енергія для науково-виробничих цілей.

5.2.1 Основна заробітна плата

Заробітна плата кожного із залучених осіб визначається за формулою (5.1)

$$З_0 = \sum_{i=1}^{K} \frac{M_{ni} * t_i}{T_p} \text{ (грн)} \qquad (5.1)$$

Де k – кількість посад працівників, залучених до процесу дослідження і розробки;

M_{ni} – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$T_р$– число робочих днів в місяці; приблизно $T_р = 22$;

t - кількість робочих днів роботи працівника.

Для проектування і розробки CRM системи мобільного було залучено наступних робітників: Solution Architect, Software Engineer та Quality Assurance Engineer. Посадові оклади, число днів роботи та витрати на компенсацію наведено в таблиці 5.4

Таблиця 5.4 - Компенсація спеціаліста в дослідницькій установі

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Solution Architect	45000	2045	44	90000
Software Engineer	40000	1818	88	160000
Quality Assurance Engineer	35000	1591	88	140000
Всього				390000

5.2.2 Розрахунок додаткової заробітної плати робітників

Додаткова заробітна плата $З_д$ всіх робітників, які приймали участь в розробці нового технічного рішення розраховується за формулою (5.2) як 10 - 15 % від основної заробітної плати робітників як премія.

На даному підприємстві додаткова заробітна плата нараховується в розмірі 10% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{доп}}{100\%} \quad (5.2)$$

$$З_д = 0{,}1 * 390000 = 39000 \text{ грн}$$

5.2.3 Відрахування на соціальні заходи

Нарахування на заробітну плату $Н_{ЗП}$ робітників, які брали участь у виконанні роботи, розраховуються за формулою (5.3):

$$З_н = (З_о + З_р + З_д) * \frac{Н_{ЗП}}{100} \text{ (грн)} \quad (5.3)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

$Н_{ЗП}$ – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Основна ставка єдиного внеску на загальнообов'язкове державне соціальне страхування на 2023 рік – 22 %, тоді:

$$З_н = (390000 + 39000) * 0.22 = 94380 (грн)$$

5.2.4 Сировина та матеріали

Дана стаття витрат включає витрати на матеріали, пристрої, засоби, які використовують при виготовленні одиниці продукції. Розраховуються, згідно їх номенклатури, за формулою:

$$К = \sum_{i=1}^{n} Н_i \cdot Ц_i \cdot К_i , \quad (5.4)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.;

$Ц_i$ – покупна ціна комплектуючих *i*-го найменування, грн.;

$К_i$ – коефіцієнт транспортних витрат (1,1…1,15).

Потрібно закладати витрати на доставку у вигляді коефіцієнту транспортних витрат – 1.1. Інформацію про використані матеріали та комплектуючі наведено у таблиці 5.5.

Таблиця 5.5 – Матеріали, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн.	Витра-чено	Вартість витраченого матеріалу, грн.
Папір офісний А4 Zoom білий 250 аркушів	205	3	615
Тонер ColorWay для принтера HP LaserJet 1018	129	1	129
Набір маркерів кольорових Stanger 8 шт.	180	1	180
Всього			924
З врахуванням коефіцієнта транспортування			1016

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування, верстатів, пристроїв, інструментів, приладів, стендів, апаратів, механізмів, іншого спецобладнання, необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення. Вартість спецустаткування визначається за прейскурантом гуртових цін або за даними базових підприємств за відпускними і договірними цінами. До балансової вартості устаткування окрім прейскурантної вартості входять витрати на його транспортування і монтаж, тому ці витрати беруться додатково в розмірі 10…12% від вартості устаткування. Балансову вартість спецустаткування розраховують за формулою:

$$В_{спец} = \sum_{i=1}^{k} Ц_і \cdot C_{пр.і} \cdot К_і, \quad (5.5)$$

Таблиця 5.6 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кіль-кість, шт	Ціна за одиницю, грн	Вартість, грн
Комп’ютер	3	40000	120000
Ноутбук	1	25000	25000
Всього			145000
З врахуванням коефіцієнта транспортування			159500

5.2.6 Витрати на програмне забезпечення

До даної статті входять витрати на програмне забезпечення, необхідне для проектування та розробки CRM системи мобільного електротранспорту. Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{i=1}^{k} Ц_{іпрг} \cdot C_{пргі} \cdot К_і, \quad (5.6)$$

де $Ц_{іпрг}$ – ціна придбання/використання одиниці програмного засобу цього виду, грн;

$С_{пргі}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо ($К_і$ = 1,10…1,12).

k – кількість найменувань програмних засобів.

Отримані результати наведено в таблиці 5.7.

Таблиця 5.7 – Витрати на використання програмних

Найменування інструментального ПЗ для підписки	Час використання, місяців	Ціна за місяць, грн	Вартість, грн
Visual Paradigm Enterprise	2	3660	7320
IntelliJ IDEA	6	2800	16800
Всього			24120

5.2.7 Амортизація обладнання

До даної статті включаються амортизаційні відрахування по кожному виду обладнання, устаткування яке використовувалось для проектування та розробки CRM системи мобільного електротранспорту.

$$А_{обл} = \frac{Ц_б}{Т_в} * \frac{t_{вик}}{12} \quad (5.7)$$

де $Ц_б$ – балансова вартість даного виду обладнання (приміщень), грн.;

$t_{вик}$ – час користування;

$Т_в$ – термін використання обладнання (приміщень), цілі місяці.

Для розробки та хостингу продукту використовувався персональний комп’ютер вартістю 120000 грн. Для тестування використовувався ноутбук вартістю 25000 грн. Амортизаційні відрахування наведено в таблиці 5.8.

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Комп’ютер	120000	5	4	8000
Ноутбук	25000	4	4	2083
Офісне приміщення	2000000	30	4	22222
Всього				32306

5.2.8 Енергія для науково-виробничих цілей

До даної статті відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

$$B_e = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_e \cdot К_{впі}}{\eta_i} \quad (5.8)$$

де W_{yt} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Вартість електроенергії становить 7.6 грн за кВт в 2023 році. При розробці системи використовувалось 2 комп’ютери з сумарною потужністю 0.9 кВт, освітлення, вентиляція та кондиціювання має сумарну потужність 0.5 кВт. Сумарні витрати на електроенергію становлять:

$$B_e = \frac{(0.9 + 0.5) \cdot 792 \cdot 7.6 \cdot 0.45}{0.76} = 4990$$

5.2.9 Службові відрядження

Стаття включає витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень. Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою 5.9:

$$B_{св} = \left(З_о + З_р\right) \cdot \frac{Н_{св}}{100} \quad (5.9)$$

де $Н_{св}$ – норма нарахування за статтею «Інші витрати».

$$B_{св} = 390000 * 0.2 = 78000 \text{ грн}$$

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Дана стаття витрат включає витрати на проведення досліджень, що не можуть бути виконані штатними працівниками або наявним обладнанням організації, а виконуються на договірній основі іншими підприємствами, установами і організаціями незалежно від форм власності та позаштатними працівниками. Такі витрати розраховуються як 30…45% від суми основної заробітної плати дослідників та робітників за формулою 5.10.

$$B_{сп} = \left(З_о + З_р\right) \cdot \frac{Н_{сп}}{100} \quad (5.10)$$

де $Н_{пв}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації».

$$В_{сп} = 390000 * 0.30 = 117000 \text{ грн}$$

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у попередніх статтях витрат і можуть бути віднесені безпосередньо на собівартість розробки за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати робітників за формулою:

$$І_в = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100} \quad (5.11)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$$І_в = 390000 * 0.5 = 195000 \text{грн}$$

5.2.12 Загальновиробничі витрати

Дана стаття витрат охоплює витрати на управління організацією, оплату службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати $В_{нзв}$ можна прийняти як 120% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$В_{нзв} = \left(З_о + З_р\right) \cdot \frac{Н_{нзв}}{100}, \quad (5.12)$$

де $Н_{нзв}$ – норма нарахування за статтею «Загальновиробничі витрати».

$$В_{нзв} = 390000 * 1.2 = 486000 \text{грн}$$

5.2.13 Загальні витрати

Сума всіх статей витрат дає в результаті витрати на проведення дослідження та розробку CRM системи мобільного електротранспорту знань і розраховується за формулою 5.13:

$$В = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв} \quad (5.13)$$

Таблиця 5.9 – Суми за розрахованими статтями витрат

Назва статті витрат	Сума, грн.
5.2.1 Основна заробітна плата	390000
5.2.2 Розрахунок додаткової заробітної плати робітників	39000
5.2.3 Відрахування на соціальні заходи	94380
5.2.4 Сировина та матеріали	1016
5.2.5 Спецустаткування для наукових (експериментальних) робіт	159500
5.2.6 Витрати на програмне забезпечення	24120
5.2.7 Амортизація обладнання	32306
5.2.8 Енергія для науково-виробничих цілей	4990
5.2.9 Службові відрядження	78000
5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації	117000
5.2.11 Інші витрати	195000
5.2.12 Загальновиробничі витрати	486000
Всього:	1621312

$$В = 390000 + 39000 + 94380 + 1016 + 159500 + 24120 + 32306 + 4990 + 78000 + 117000 + 195000 + 486000 = 1621312 \text{ грн}$$

Загальні витрати ЗВ на завершення роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В}{\eta}, \quad (5.14)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії дослідного зразка, то коефіцієнт β = 0.5.

Звідси:

$$ЗВ = \frac{1621312}{0.5} = 3242624 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки

В цьому підрозділі розрахуємо прогнозований прибуток від реалізації розробленої CRM системи.

Оскільки, від моменту вкладення інвестицій до отримання прибутку минає певний період, то потрібно враховувати також девальвацію.

Прогнозований термін комерціалізації нашого продукту складає 3 роки.

Для розрахунку збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки використовується формула 5.15:

$$\Delta П_i = \sum_1^n \left(\Delta Ц_o \cdot N + Ц_o \cdot \Delta N\right)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100}\right) \quad (5.15)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році (у нашому випадку припустимо, що ціна нашої системи зросте на 2000 грн.);

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження нової розробки (у нас це значення буде початком реалізації системи – 1 шт.);

Цо – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів розробки (оціночна вартість нашої системи складає 10000 грн);

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки (припустимо, що за 1-й рік буде реалізовано 1000 систем, за 2-й рік – 5000 шт., за 3-й рік – 10000 шт.;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки (вище було зазначено – 3 роки);

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт *л* = 0,8333.

ρ – коефіцієнт, який враховує рентабельність продукту. р = 0,25;

ϑ – ставка податку на прибуток. У 2023 році – 18%.

$$\Delta П_1 = (2000 \cdot 1 + 10000 \cdot 1000) \cdot 0.833 \cdot 0.25 \cdot \left(1 - \frac{18}{100}\right) = 1707992 \text{ грн.}$$

$$\Delta П_2 = (2000 \cdot 1 + 10000 \cdot 5000) \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = 8538592 \text{ грн.}$$

$$\Delta П_3 = (2000 \cdot 1 + (10000 \cdot 10000) \cdot 10800) \cdot 0.833 \cdot 0.25 \cdot \left(1 - \frac{18}{100}\right) = 17076842 \text{ грн.}$$

Далі за формулою 5.16 розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} \quad (5.16)$$

де ΔПі – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15. В Україні рівень інфляції за підсумком 2023 року склав 10.6%, прогнозований рівень на 2024 рік – 8.5% ;

t – період часу (в роках) від моменту початку впровадження розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$\text{ПП} = \frac{1707992}{1+0.1} + \frac{8538592}{(1+0.085)^2} + \frac{17076842}{(1+0.085)^3}$$
$$= 1552720 + 7253152 + 13369598 = 22175469 \text{ грн.}$$

За формулою 5.16 необхідно розрахувати величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ} \qquad (5.17)$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{\text{інв}}$ =2...5, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2{,}5 \cdot 3242624 = 8106560 \text{ грн}$$

Тоді абсолютний економічний ефект $E_{\text{абс}}$ або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації розробки становитиме:

$$E_{\text{абс}} = \text{ПП} - PV \qquad (5.17)$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн;

PV – теперішня вартість початкових інвестицій, грн.

$$E_{\text{абс}} = 22175469 - 8106560 = 14068909 \text{ грн.}$$

Розрахована величина економічного ефекту має велике додатне значення, отже це свідчить про потенціал зацікавленості інвесторів у впровадженні та комерціалізації розробки.

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Для остаточного прийняття рішення про впровадження розробки та виведення її на ринок необхідно розрахувати внутрішню економічну дохідність $E_в$ або показник внутрішньої норми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою 5.19:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.19}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн; PV – теперішня вартість початкових інвестицій, грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки. 2,735496785

$$E_в = \sqrt[3]{1 + \frac{14068909}{8106560}} - 1 = \sqrt[3]{2{,}7355} - 1 = 1{,}398552379 - 1 = 0.40 = 40\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою 5.20:

$$\tau = d + f, \tag{5.20}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d= (0,14…0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{min} = 0.17 + 0.07 = 0.24$$

Так як $E_в > \tau_{min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Далі розраховується період окупності інвестицій, вкладених у реалізацію проекту за формулою 5.21.

$$T_{ок} = \frac{1}{E_в} \tag{5.21}$$

де Ев – внутрішня економічна дохідність вкладених інвестицій.

$$\text{Т}_{\text{ок}} = \frac{1}{0.40} = 2.5 \text{ роки} = 30 \text{ місяців} = 2 \text{ роки та 6 місяці}$$

Отже, якщо $\text{Т}_{\text{ок}} \leq$ 3 років, то це свідчить про позитивний комерційний потенціал програмного продукту і може спонукати потенційних інвесторів профінансувати впровадження цієї CRM системи та виведення її на ринок.

5.5 Висновки

За результатами опитування 3-х незалежних експертів було встановлено значення комерційного потенціалу для CRM системи мобільного електротранспорту – вище середнього.

Було встановлено прогнозовані витрати на проектування та розробку CRM системи за відповідними статтями – 1621312 грн., а загальна величина витрат становить 3242624 грн..

Встановлені значення економічної ефективності та терміну окупності свідчать про те, що розробка CRM системи мобільного електротранспорту є економічно вигідною та може зацікавити потенційних інвесторів. Прогнозований чистий прибуток від реалізації системи протягом 3-х років склав 14068909 грн., а термін окупності інвестицій становить 2 роки та 6 місяці.

ВИСНОВКИ

Отже, в даній магістерській кваліфікаційній роботі був проведений аналіз сучасних методів і програмних засобів для автоматизації відносин з клієнтами та запропоновано обрати операційний вид CRM системи для вдосконалення методів і програмних засобів CRM системи мобільного електротранспорту.

Також, було розроблено методи для розрахунку необхідної потужності мобільного електротранспорту та енергетичної ємності акумуляторної батареї з використанням фізичних формул з розділів механіки та електродинаміки. На основі цих методів були виведені формули та створені моделі розрахунку вище зазначених показників за вхідними параметрами клієнта та відповідними характеристиками мобільного електротранспорту.

Далі було описано процес розробки програмного забезпечення і базуючись на аналізі та обгрунтуванні способів реалізації було обрано мову програмування Java, базу даних PostgreSQL та середовище розробки IntelliJ IDEA. Також, було спроектовано загальну архітектуру, основні модулі та бізнес логіку.

Потім було зроблено аналіз методів і засобів тестування програмного забезпечення. Для тестування нашої системи були застосовані два основних принципи тестування програмного забезпечення – «білого ящика» та «чорного ящика». На першому етапі, для автоматичного тестування модулів нашої системи, були написані юніт-тести. Після створення та тестування модулів проводилось мануальне тестування ендпойнтів нашої системи. І на завершальному етапі було проведено мануальне тестування користувацького інтерфейсу шару відображення за допомогою web-браузера.

За результатами розрахунків економічної ефективності було встановлено, що розробка методів і програмних засобів CRM системи мобільного електротранспорту є економічно вигідною та може мати комерційний успіх.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гронюк Р.О., Ліщинська Л.Б. Порівняльний аналіз методів і програмних засобів автоматизації відносин з клієнтами. Електронні інформаційні ресурси: створення, використання, доступ: Збірник матеріалів Міжнародної науково-практичної Інтернет-конференції 20-21 листопада 2023 р. Суми/Вінниця: НІКО/ВНТУ, 2023. С.89-91. URL: https://drive.google.com/file/d/1oVmxS3W_sEQPjes9S9AzWDaJxDxi6I0X/view (Lastaccessed: 04.12.2023).
2. Впровадження CRM-систем як засіб підвищення ефективності маркетингової діяльності, 2023. URL: http://www.economyandsociety.in.ua/index.php/journal/article/view/2269/2192 (Lastaccessed: 05.12.2023).
3. TQM systems. Впровадження CRM-системи: роль CRM-технологій у підвищенні ефективності бізнесу. URL: https://tqm.com.ua/ua/likbez/crm-systemy/rol-vprovadzhennia-crm (Lastaccessed: 04.12.2023).
4. Customer Relationship Management Research from 2000 to 2020: An Academic Literature Review and Classification, 2021. URL: https://www.researchgate.net/publication/349609303_Customer_Relationship_Management_Research_from_2000_to_2020_An_Academic_Literature_Review_and_Classification (Lastaccessed: 05.12.2023).
5. Аналіз функцій та принципів розроблення CRM-систем. URL: https://openarchive.nure.ua/server/api/core/bitstreams/7bd6b092-3c29-4902-922a-c2b963e7a37e/content (Lastaccessed: 05.12.2023).
6. Особливості застосування crm-систем у маркетинговому та логістичному управлінні закладами послуг, 2023. URL: http://e-forum.lntu.edu.ua/index.php/ekonomichnyy_forum/article/view/375/379 (Lastaccessed: 05.12.2023).
7. Роль crm системи в управлінні взаєминами з клієнтами у готельному бізнесі, 2023. URL:

http://www.vtei.com.ua/doc/2023/vatra2004/zb22_177.pdf#page=322 (Lastaccessed: 05.12.2023).

8. CRM аналітика сучасного маркетолога, 2023. URL: https://er.knutd.edu.ua/bitstream/123456789/23890/1/DOMIN_2023_P071-073.pdf (Lastaccessed: 05.12.2023).
9. Розвиток та імперативисrm-систем в управлінні транспортним підприємством, 2023. URL: https://amtp.org.ua/index.php/journal2/article/view/538/458 (Lastaccessed: 05.12.2023).
10. Впровадження crm-систем як засіб підвищення ефективності маркетингової діяльності, 2023. URL: https://www.economyandsociety.in.ua/index.php/journal/article/view/2269 (Lastaccessed: 05.12.2023).
11. Удосконалення системи управління комунікації організації зі споживачами шляхом автоматизації процесу надання послуг у середовищі CRM, 2023. URL: https://er.chdtu.edu.ua/handle/ChSTU/4575 (Lastaccessed: 05.12.2023).
12. Прогнозування впливу ефективності бізнес-процесів на конкурентостійкість підприємств електронної торгівлі, 2023. URL: https://www.economyandsociety.in.ua/index.php/journal/article/view/2858 (Lastaccessed: 05.12.2023).
13. Трансформація системи управління персоналом підприємств в умовах постіндустріального розвитку суспільства, 2023. URL: https://em.duit.in.ua/index.php/home/article/view/91 (Lastaccessed: 05.12.2023).
14. CRM-системи як інструменти обліково-аналітичного забезпечення економічної безпеки підприємств, 2023. URL: https://em.duit.in.ua/index.php/home/article/view/66 (Lastaccessed: 05.12.2023).
15. Innovative principles of using crm systems: methodology at different levels of branded niche industries, 2023. URL: https://visnyk.nuwm.edu.ua/index.php/econ/article/view/1248 (Lastaccessed: 05.12.2023).

16. Підвищення якості автоматизації бізнес-процесів шляхом розроблення й тестування методу оцінювання ефективності використання crm-платформ на базі застосування системи salesforce, 2023. URL: http://bulletin.khadi.kharkov.ua/article/view/257398 (Lastaccessed: 05.12.2023).
17. Теоретичні основи формування діджитал стратегії в умовах vuca-світу, 2023. URL: http://vtei.com.ua/doc/11konf/zb10.pdf#page=251 (Lastaccessed: 05.12.2023).
18. Цифровізація маркетингової діяльності підприємства, 2023. URL: https://www.economyandsociety.in.ua/index.php/journal/article/view/2608 (Lastaccessed: 05.12.2023).
19. CRM-системи в управлінні маркетинговими ризиками проєктів ризикозахищеності банків, 2022. URL: http://rinek.onu.edu.ua/article/view/274373 (Lastaccessed: 05.12.2023).
20. Інформаційні технології в системі управління взаємовідносин з клієнтами, 2023. URL: http://journals.khnu.km.ua/vestnik/wp-content/uploads/2022/07/2022-306-41.pdf (Lastaccessed: 05.12.2023).
21. Особливості застосування цифрових технологій «інтернет речей» та новітніх систем у бізнесі, 2022. URL: https://journal.eae.com.ua/index.php/journal/article/view/150/131 (Lastaccessed: 05.12.2023).
22. Стан та тенденції сучасних досліджень з маркетингу: бібліометричний огляд, 2022. URL: https://www.mdt-opu.com.ua/index.php/mdt/article/view/286 (Lastaccessed: 05.12.2023).
23. Маркетинг в цифрової економіці, 2022. URL: http://dees.iei.od.ua/index.php/journal/article/view/68 (Lastaccessed: 05.12.2023).
24. Сценарне планування як інструмент безперервного бізнесу: web-технології (комплексний підхід), 2022. URL: http://journals.maup.com.ua/index.php/it/article/view/2098 (Lastaccessed: 05.12.2023).
25. Впровадження crm-систем як засіб підвищення ефективності маркетингової

діяльності, 2023. URL: https://www.economyandsociety.in.ua/index.php/journal/article/view/2269/2192 (Lastaccessed: 05.12.2023).

26. Best CRM Softwareof 2023, 2023 URL: https://www.businessnewsdaily.com/7839-best-crm-software.html (Lastaccessed: 05.12.2023).
27. NO-CODE платформа для автоматизації процесів та CRM з максимальним рівнем свободи. URL: https://www.creatio.com/ua/ (Lastaccessed: 05.12.2023).
28. CRM для продажу товарів, для інтернет-магазину. URL: https://salesdrive.ua/ (Lastaccessed: 05.12.2023).
29. CRM для товарного бізнесу. URL: https://lp-crm.biz/ (Lastaccessed: 05.12.2023).
30. Зрозуміла CRM для керування компанії. URL: https://keepincrm.com/ (Lastaccessed: 05.12.2023).
31. CRM system for commodity business. URL: https://h-profit.com/ (Lastaccessed: 05.12.2023).
32. CRM-платформа для розвитку вашого бізнесу. URL: https://www.pipedrive.com/uk (Lastaccessed: 05.12.2023).
33. Більше, ніж просто CRM. Одна програма для управління. URL: https://cleverbox-crm.com/ (Lastaccessed: 05.12.2023).
34. Perfectum CRM+ERP система для всієї компанії. URL: https://ua.keycrm.app/ (Lastaccessed: 05.12.2023).
35. Все, що потрібно підприємцю. Підключай. Перемагай. URL: https://ua.keycrm.app/ (Lastaccessed: 05.12.2023).
36. Механіка. Молекулярна фізика і термодинаміка. URL: https://pdf.lib.vntu.edu.ua/books/2015/Kycheryk_P1_2006_532.pdf (Lastaccessed: 05.12.2023).
37. Курс фізики електрика і магнетизм. URL: https://pdf.lib.vntu.edu.ua/books/2015/Byshok_P2_2003_278.pdf (Lastaccessed: 05.12.2023)

38. MVC Architecture, 2022. URL: https://www.educative.io/blog/mvc-tutorial (Last accessed: 29.05.2023).
39. Service Oriented Architecture (SOA), 2022. URL: https://www.javatpoint.com/service-oriented-architecture (Last accessed: 29.05.2023).
40. Purposeof Precedence Diagram Method, 2023. URL: https://acqnotes.com/acqnote/tasks/precedence-diagram-method-pdm. (Last accessed: 29.05.2023).
41. UML Class Diagram, 2023. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/ (Last accessed: 29.05.2023).
42. Purpose of Interaction Diagrams, 2023. URL: https://www.tutorialspoint.com/uml/uml_interaction_diagram.htm (Last accessed: 29.05.2023).
43. Visual Paradigm, 2022. URL: https://www.visual-paradigm.com/tutorials/ (Last accessed: 05.12.2023).
44. Energy efficiency of the Visitor Pattern: contrasting Java and C++ implementations, 2023 URL: https://link.springer.com/article/10.1007/s10664-023-10387-8 (Lastaccessed: 05.12.2023).
45. Spring Boot, 2023 URL: https://spring.io/projects/spring-boot (Last accessed: 05.12.2023).
46. Comparative analysis of selected databases on the example of a proprietary web application, 2023. URL: https://ph.pollub.pl/index.php/jcsi/article/view/3668/4164 (Lastaccessed: 05.12.2023).
47. IntelliJ IDEA, 2023 URL: https://www.jetbrains.com/idea/features/ (Last accessed: 05.12.2023).
48. Spring Initializr: 2023 URL: https://start.spring.io/ (Last accessed: 29.05.2023).
49. A reliable and an efficient web testing system: 2020 URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3341095 (Last accessed: 05.12.2023).

50. A Study of Software Testing: Categories, Levels, Techniques, and Types: 2020 URL: https://www.researchgate.net/publication/342579919_A_Study_of_Software_Testing_Categories_Levels_Techniques_and_Types (Last accessed: 05.12.2023).
51. A Comparative Study of White Box, Black Box and Grey Box Testing Techniques: 2012 URL: https://www.researchgate.net/publication/270554162_A_Comparative_Study_of_White_Box_Black_Box_and_Grey_Box_Testing_Techniques (Last accessed: 05.12.2023).
52. Comprehensive study of software testing: Categories, levels, techniques, and types: 2019 URL: https://www.researchgate.net/publication/337331361_Comprehensive_study_of_software_testing_Categories_levels_techniques_and_types (Last accessed: 05.12.2023).
53. IEEE Standard Glossary of Software Engineering Terminology, 2023 URL: https://standards.ieee.org/ieee/610.12/855/ (Last accessed: 05.12.2023).

ДОДАТКИ

Додаток А. Таблиці опису класів системи

Таблиця А.1 – Опис класу CrmMobileElectricTransportApplication

Параметр	**Значення**
Коментар	Головний клас, який є точкою старту системи
Атрибути	
Операції	**…**

Таблиця А.2 – Опис класу User

Параметр	**Значення**
Коментар	Клас, який представляє сутність користувача системи з певним набором даних та ролей доступу
Атрибути	id – унікальний ідентифікатор користувача password - пароль username – унікальний логін passwordConfirm – підтвердження пароля roles – набір ролей користувача
Операції	isAccountNonExpired – перевіряє чи не закінчився термін дії користувача isAccountNonLocked – перевірка заблокованості isCredentialsNonExpired – перевірка закінчення прав доступу isEnabled – перевірка стану влкючення getAuthorities – отримати дані авторизації користувача

Додаток Б. Діаграма класів

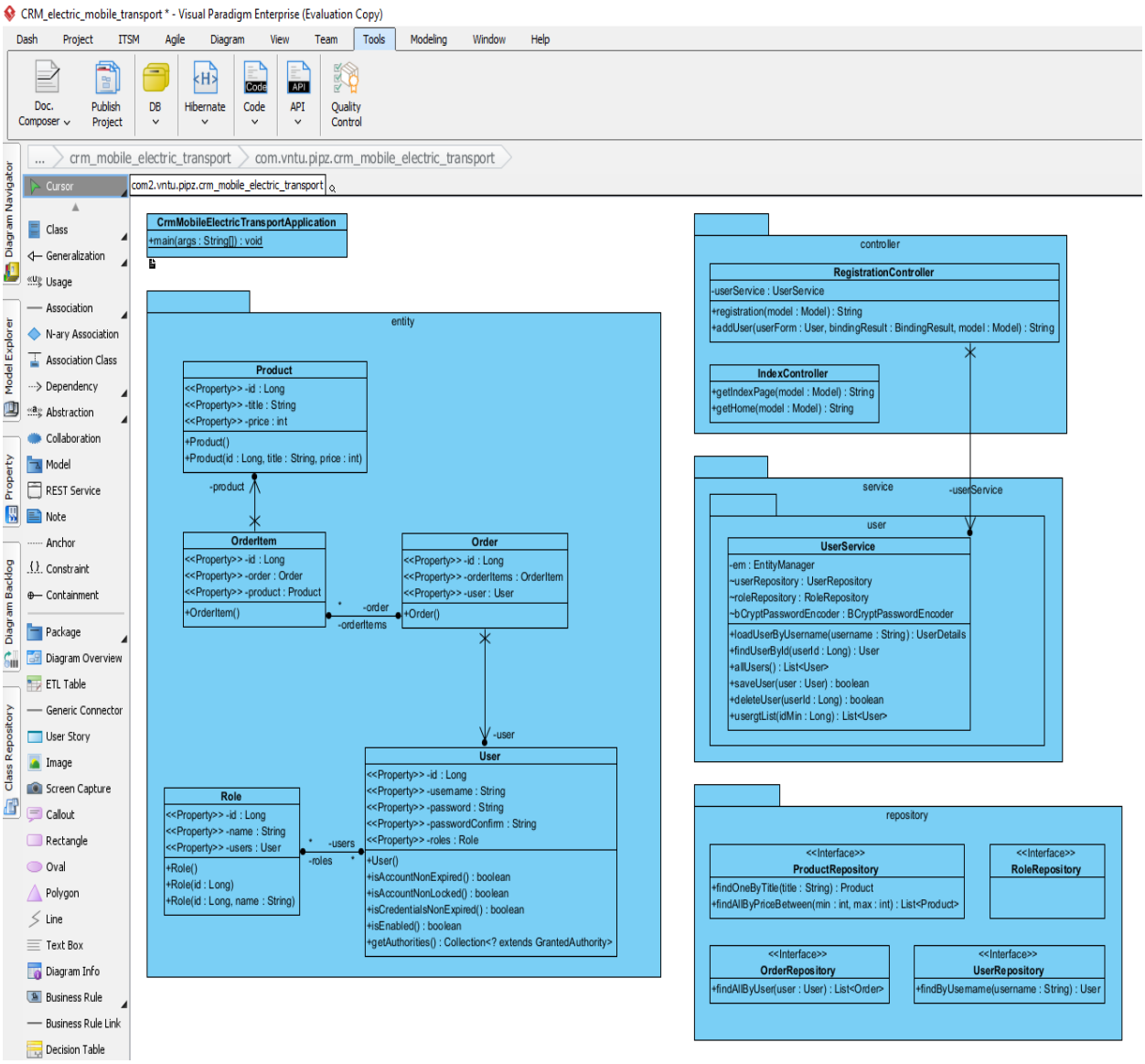


Рисунок Б.1 – Діаграма класів

Додаток В. Діаграма послідовності

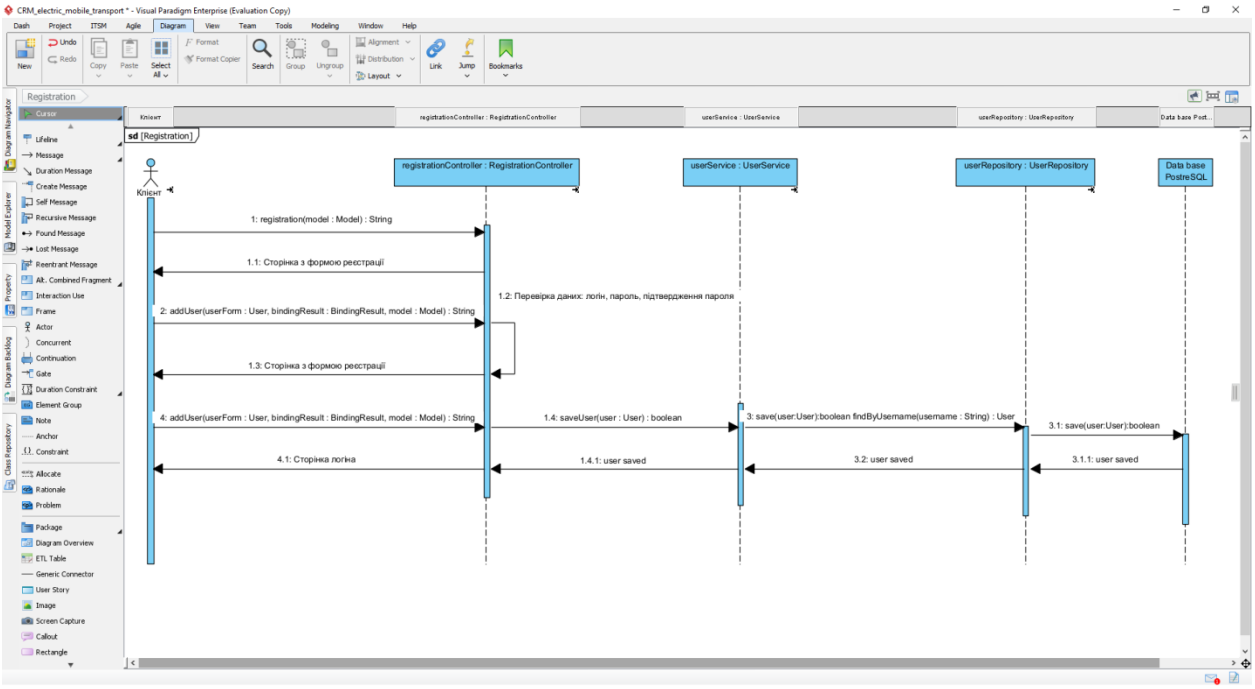


Рис. В.1 – Діаграма послідовності реєстрації нового користувача

Додаток Г. Діаграма активності

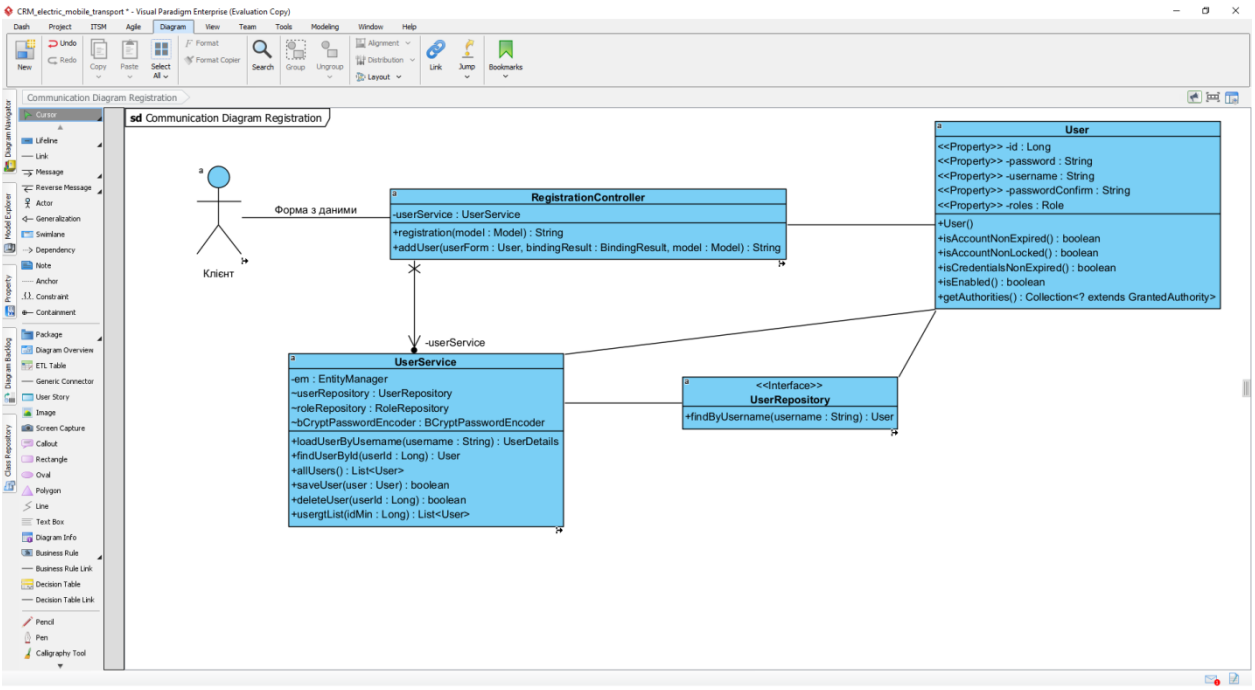


Рис. Г.1 – Діаграма кооперації для сценаріїв прецедентів

Додаток Д. Лістинг

```
/*
 *
 *
 *
 *
 *
 * Copyright (c) 2023, Ruslan Gronyuk
 *
 * All rights reserved.
 *
 *
 */

package com.vntu.pipz.crm_mobile_electric_transport;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

/**
 *This program was created to obtain customer relationship management
 * for company of mobile electric transport
 * */

@SpringBootApplication
public class CrmMobileElectricTransportApplication {

    public static void main(String[] args) {
        SpringApplication.run(CrmMobileElectricTransportApplication.class, args);
    }

}

package com.vntu.pipz.crm_mobile_electric_transport.entity;

import org.springframework.security.core.GrantedAuthority;
```

```
import org.springframework.security.core.userdetails.UserDetails;

import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.validation.constraints.Size;
import java.util.Collection;
import java.util.Set;

@Entity
@Table(name = "users")
public class User implements UserDetails {
    @Id
    @SequenceGenerator(name = "users_seq", sequenceName = "users_seq",
allocationSize = 1)
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"users_seq")
    private Long id;
    @Size(min=2, message = "Не меньше 5 знаков")
    private String username;
    @Size(min=2, message = "Не меньше 5 знаков")
    private String password;
    @Transient
    private String passwordConfirm;
    @ManyToMany(fetch = FetchType.EAGER)
    private Set<Role> roles;

    public User() {
    }

    public Long getId() {
        return id;
    }
```

```
public void setId(Long id) {
   this.id = id;
}

@Override
public String getUsername() {
   return username;
}

@Override
public boolean isAccountNonExpired() {
   return true;
}

@Override
public boolean isAccountNonLocked() {
   return true;
}

@Override
public boolean isCredentialsNonExpired() {
   return true;
}

@Override
public boolean isEnabled() {
   return true;
}

public void setUsername(String username) {
   this.username = username;
}

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
   return getRoles();
}
```

```
    @Override
    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getPasswordConfirm() {
        return passwordConfirm;
    }

    public void setPasswordConfirm(String passwordConfirm) {
        this.passwordConfirm = passwordConfirm;
    }

    public Set<Role> getRoles() {
        return roles;
    }

    public void setRoles(Set<Role> roles) {
        this.roles = roles;
    }
}
```

```
package com.vntu.pipz.crm_mobile_electric_transport.entity;

import com.vntu.pipz.crm_mobile_electric_transport.entity.User;
import org.springframework.security.core.GrantedAuthority;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import java.util.Set;

@Entity
```

```
@Table(name = "roles")
public class Role implements GrantedAuthority {
    @Id
    private Long id;
    private String name;
    @Transient
    @ManyToMany(mappedBy = "roles")
    private Set<User> users;
    public Role() {
    }

    public Role(Long id) {
        this.id = id;
    }

    public Role(Long id, String name) {
        this.id = id;
        this.name = name;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Set<User> getUsers() {
        return users;
    }
```

```
    public void setUsers(Set<User> users) {
        this.users = users;
    }

    @Override
    public String getAuthority() {
        return getName();
    }
}
```

```
package com.vntu.pipz.crm_mobile_electric_transport.entity;

import javax.persistence.*;

@Entity
@Table(name = "products")
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;

    @Column(name = "title")
    private String title;

    @Column(name = "price")
    private int price;

    public Product() {}

    public Product(Long id, String title, int price) {
        this.id = id;
        this.title = title;
        this.price = price;
    }

    public Long getId() {
```

```
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public int getPrice() {
        return price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    @Override
    public String toString() {
        return "Product [" +
                "id=" + id +
                ", title='" + title + '\'' +
                ", price=" + price +
                ']';
    }
}
```

```
package com.vntu.pipz.crm_mobile_electric_transport.entity;

import javax.persistence.*;

@Entity
@Table(name = "order_items")
```

```
public class OrderItem {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;

    @ManyToOne
    @JoinColumn(name = "order_id")
    private Order order;

    @ManyToOne
    @JoinColumn(name = "product_id")
    private Product product;

    public OrderItem() {
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public Order getOrder() {
        return order;
    }

    public void setOrder(Order order) {
        this.order = order;
    }

    public Product getProduct() {
        return product;
    }

    public void setProduct(Product product) {
```

```
        this.product = product;
    }

    @Override
    public String toString() {
      return product.toString();
    }
}
```

```
package com.vntu.pipz.crm_mobile_electric_transport.entity;

import javax.persistence.*;
import java.util.List;

@Entity
@Table(name = "orders")
public class Order {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;

    @OneToMany(mappedBy = "order", cascade = CascadeType.ALL)
    private List<OrderItem> orderItems;

    @ManyToOne
    @JoinColumn(name = "user_id")
    private User user;

    public Order() {
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
```

```
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public List<OrderItem> getOrderItems() {
        return orderItems;
    }

    public void setOrderItems(List<OrderItem> orderItems) {
        this.orderItems = orderItems;
    }

    @Override
    public String toString() {
        return "Order [" +
                "id=" + id +
                ", orderItems=" + orderItems +
                ", user=" + user.getUsername() +
                ']';
    }
}
```

```
package com.vntu.pipz.crm_mobile_electric_transport.repository;

import com.vntu.pipz.crm_mobile_electric_transport.entity.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
}
```

```
package com.vntu.pipz.crm_mobile_electric_transport.repository;

import com.vntu.pipz.crm_mobile_electric_transport.entity.Role;
```

```
import org.springframework.data.jpa.repository.JpaRepository;

public interface RoleRepository extends JpaRepository<Role, Long> {
}
```

```
package com.vntu.pipz.crm_mobile_electric_transport.repository;

import com.vntu.pipz.crm_mobile_electric_transport.entity.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface ProductRepository extends JpaRepository<Product, Long> {

    Product findOneByTitle(String title);

    List<Product> findAllByPriceBetween(int min, int max);
}
```

```
package com.vntu.pipz.crm_mobile_electric_transport.repository;

import com.vntu.pipz.crm_mobile_electric_transport.entity.Order;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.security.core.userdetails.User;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface OrderRepository extends JpaRepository<Order, Long> {

    List<Order> findAllByUser(User user);
}
```

```
package com.vntu.pipz.crm_mobile_electric_transport.service.user;

import com.vntu.pipz.crm_mobile_electric_transport.entity.Role;
import com.vntu.pipz.crm_mobile_electric_transport.entity.User;
```

```
import com.vntu.pipz.crm_mobile_electric_transport.repository.RoleRepository;
import com.vntu.pipz.crm_mobile_electric_transport.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import java.util.Collections;
import java.util.List;
import java.util.Optional;

@Service
public class UserService implements UserDetailsService {
    @PersistenceContext
    private EntityManager em;
    @Autowired
    UserRepository userRepository;
    @Autowired
    RoleRepository roleRepository;
    @Autowired
    BCryptPasswordEncoder bCryptPasswordEncoder;

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        User user = userRepository.findByUsername(username);

        if (user == null) {
            throw new UsernameNotFoundException("User not found");
        }

        return user;
    }

    public User findUserById(Long userId) {
```

```
        Optional<User> userFromDb = userRepository.findById(userId);
        return userFromDb.orElse(new User());
    }

    public List<User> allUsers() {
        return userRepository.findAll();
    }

    public boolean saveUser(User user) {
        User userFromDB = userRepository.findByUsername(user.getUsername());

        if (userFromDB != null) {
            return false;
        }

        user.setRoles(Collections.singleton(new Role(1L, "ROLE_USER")));
        user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));
        userRepository.save(user);
        return true;
    }

    public boolean deleteUser(Long userId) {
        if (userRepository.findById(userId).isPresent()) {
            userRepository.deleteById(userId);
            return true;
        }
        return false;
    }

    public List<User> usergtList(Long idMin) {
        return em.createQuery("SELECT u FROM User u WHERE u.id > :paramId",
User.class)
                .setParameter("paramId", idMin).getResultList();
    }
}

/*
 * @author  Ruslan Gronyuk
 */
```

```
package com.vntu.pipz.crm_mobile_electric_transport.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class IndexController {
    @GetMapping(value = "/")
    public String getIndexPage(Model model) {
        return "index";
    }

    @GetMapping(value = "/home")
    public String getHome(Model model) {
        return "home";
    }
}
```

```
package com.vntu.pipz.crm_mobile_electric_transport.controller;

import com.vntu.pipz.crm_mobile_electric_transport.service.user.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class AdminController {
    @Autowired
    private UserService userService;

    @GetMapping("/admin")
    public String userList(Model model) {
        model.addAttribute("allUsers", userService.allUsers());
```

```
        return "admin";
    }

    @PostMapping("/admin")
    public String  deleteUser(@RequestParam(required = true, defaultValue = "" )
Long userId,
                              @RequestParam(required = true, defaultValue = "" ) String action,
                              Model model) {
        if (action.equals("delete")){
            userService.deleteUser(userId);
        }
        return "redirect:/admin";
    }

    @GetMapping("/admin/gt/{userId}")
    public String  gtUser(@PathVariable("userId") Long userId, Model model) {
        model.addAttribute("allUsers", userService.usergtList(userId));
        return "admin";
    }
}
```

```
package com.vntu.pipz.crm_mobile_electric_transport.controller;

import com.vntu.pipz.crm_mobile_electric_transport.entity.User;
import com.vntu.pipz.crm_mobile_electric_transport.service.user.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;

import javax.validation.Valid;

@Controller
public class RegistrationController {

    @Autowired
```

```
    private UserService userService;

    @GetMapping("/registration")
    public String registration(Model model) {
        model.addAttribute("userForm", new User());

        return "registration";
    }

    @PostMapping("/registration")
    public String addUser(@ModelAttribute("userForm") @Valid User userForm,
BindingResult bindingResult, Model model) {

        if (bindingResult.hasErrors()) {
            return "registration";
        }
        if (!userForm.getPassword().equals(userForm.getPasswordConfirm())){
            model.addAttribute("passwordError", "Паролі не співпадають");
            return "registration";
        }
        if (!userService.saveUser(userForm)){
            model.addAttribute("usernameError", "Користувач з таким іменем вже
існує");
            return "registration";
        }

        return "redirect:/home";
    }
}
```

```
package com.vntu.pipz.crm_mobile_electric_transport.config;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class MvcConfig implements WebMvcConfigurer {

    @Override
```

```
    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController("/login").setViewName("login");
        registry.addViewController("/home").setViewName("home");
    }
}
```

```
package com.vntu.pipz.crm_mobile_electric_transport.config;
import com.vntu.pipz.crm_mobile_electric_transport.service.user.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.Authenticatio
nManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecuri
ty;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConf
igurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    UserService userService;
    @Bean
    public BCryptPasswordEncoder bCryptPasswordEncoder() {
        return new BCryptPasswordEncoder();
    }
    @Override
    protected void configure(HttpSecurity httpSecurity) throws Exception {
        httpSecurity

                .csrf()

                .disable()
```

```
            .authorizeRequests()
            //Access only for non-registered users
            .antMatchers("/registration").not().fullyAuthenticated()
            //Access only for users with the Administrator role
            .antMatchers("/admin/**").hasRole("ADMIN")
            .antMatchers("/news").hasRole("USER")
            //Access is allowed to all users
            .antMatchers("/", "/resources/**").permitAll()
            //All other pages require authentication
            .anyRequest().authenticated()
            .and()
            //Setting up for login
            .formLogin()
            .loginPage("/login")
            //Redirect to home page after successful login
            .defaultSuccessUrl("/")
            .permitAll()
            .and()
            .logout()
            .permitAll()
            .logoutSuccessUrl("/");
    }
    @Autowired
    protected void configureGlobal(AuthenticationManagerBuilder auth) throws
Exception {
auth.userDetailsService(userService).passwordEncoder(bCryptPasswordEncoder());
    }
}
```

ДОДАТОК Е. Технічне завдання

127

ДОДАТОК Е. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор Романюк О.Н.
«19» 09 2023 року

Технічне завдання
на магістерську кваліфікаційну роботу «Розробка методів і програмних засобів CRM системи мобільного електротранспорту»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
_____д.т.н., проф. каф. ПЗ Ліщинська Л.Б.
"19" 09 2023 р.
Виконав:
_____студент гр. 2ПІ-22м Гронюк Р.О.
"19" 09 2023 р.

Вінниця – 2023 рік

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів і програмних засобів CRM системи мобільного електротранспорту».

Галузь застосування – програмне забезпечення для автоматизації взаємовідносин з клієнтами.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 247 від 18.09.2023 ректора по ВНТУ про затвердженян тем МКР.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є підвищення ефективності взаємовідносин підприємства з клієнтами.

Призначення роботи – розробка методів та програмних засобів для автоматзації відносин з клієнтами.

4. Вхідні дані для проведення НДР

Методи розрахунку необхідної потужності мобільного електротранспорту та необхідної енергентичної ємності акумуляторної батареї за фізичними формулами з розділів механіки та електродинаміки.

5. Технічні вимоги

Операційна система – Windows, Mac OS.
Мови програмування – Java, SQL.
Технології – Spring boot, Maven, Thymeleaf, Git.
База даних PostgreSQL.
Середовище розробки IntelliJ IDEA (Ultimate Edition).
Мають бути реалізовані ендпойнти:

- початкова сторінка;
- авторизація;
- реєстрація користувача;
- адміністрування.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;
– технічне завдання;
– лістинги програми.

7. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

8. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів	Примітка
1	Аналіз сучасних методів і програмних засобів автоматизації відносин з клієнтами	20.09.2023 – 30.09.2023	Вик.
2	Моделювання автоматизації відносин з клієнтами та проектування програмного засобу crm системи мобільного електротранспорту	30.09.2023 – 20.10.2023	Вик.
3	Розробка програмного засобу crm системи мобільного електротранспорту	20.10.2023 – 20.11.2023	Вик.
4	Тестування програмного засобу crm системи мобільного електротранспорту	20.11.2023 – 25.11.2023	Вик.
5	Економічна частина	25.11.2023 – 01.12.2023	Вик.

9. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Захист магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Є. Протокол перевірки МКР на плагіат

130

ДОДАТОК Є. Протокол перевірки МКР на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Розробка методів і програмних засобів CRM системи мобільного електротранспорту.**

Тип роботи: кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 2ПІ – 22м

Науковий керівник: д.т.н. проф. Ліщинська Л. Б.

Unicheck	
Оригінальність	**93 %**
Схожість	7 %

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Гронюк Р. О.

Керівник роботи _____ Ліщинська Л. Б.

ДОДАТОК Ж. Ілюстративна частина

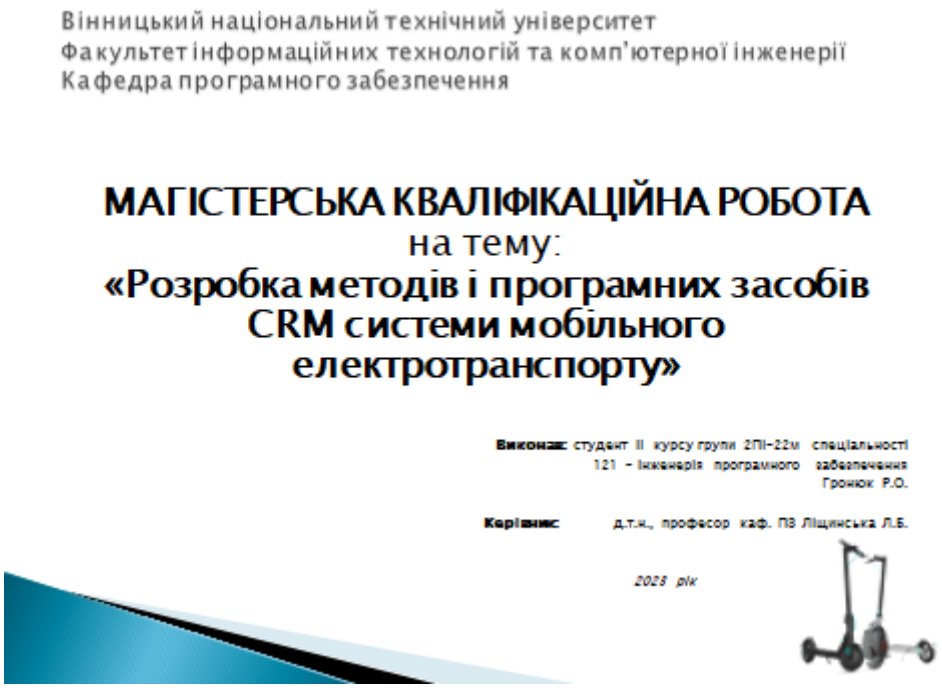


Рисунок Ж.1 – Плакат 1

Рисунок Ж.2 – Плакат 2

МЕТА ТА ЗАВДАННЯ

- **Метою** досліджень є підвищення ефективності взаємовідносин підприємства з клієнтами.

Основними заданнями досліджень є:

- проаналізувати сучасні методи і програмні засоби автоматизації відносин з клієнтами;
- розробити метод та модель розрахунку потужності мобільного електротранспорту за вхідними даними клієнта та параметрами електротранспорту;
- розробити метод та модель розрахунку енергетичної ємності акумуляторної батареї за значенням потужності та бажаною відстанню пробігу на одному заряді батареї;
- розробити та протестувати програмний засіб CRM системи мобільного електротранспорту;
- розрахувати економічну ефективність.

Рисунок Ж.3 – Плакат 3

ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕНЬ

- **Об'єктом дослідження** цієї магістерської кваліфікаційної роботи є бізнес-процеси взаємовідносин підприємства з клієнтами.
- **Предмет досліджень** - методи та засоби організації бізнес процесів взаємовідносин підприємства з клієнтами.

Рисунок Ж.4 – Плакат 4

НАУКОВА НОВИЗНА

- удосконалено модель операційної CRM системи, яка на відміну від існуючої використовує методи розрахунку потужності мобільного електротранспорту та енергетичної ємності акумуляторної батареї за вхідними даними замовника і технічними параметрами транспортного засобу, що дозволяє покращити систему взаємовідносин з клієнтами в галузі мобільного електротранспорту

Рисунок Ж.5 – Плакат 5

ОСНОВНІ ВИДИ CRM СИСТЕМ

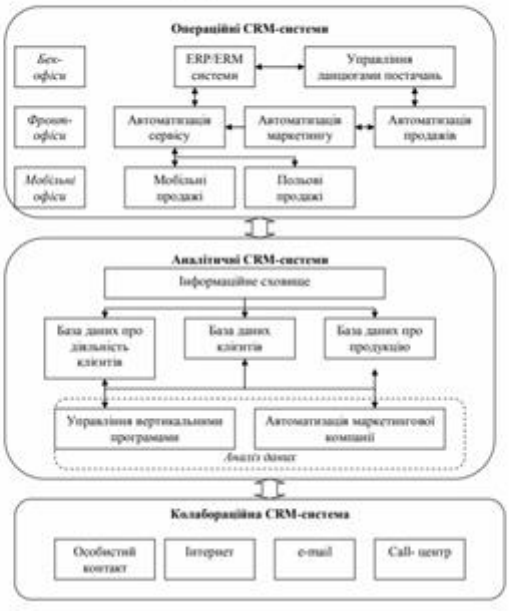


Рисунок Ж.6 – Плакат 6

ТОПОВІ CRM СИСТЕМИ НА РИНКУ УКРАЇНИ

- **1. Creatio** – це комплексна платформа для автоматизації бізнес-процесів і управління відносинами з клієнтами (CRM).
- **2. SalesDrive** – CRM система управління продажу товарами в інтернет-магазинах.
- **3. LP-CRM** – створена для продажу товарів через сайти-лендінги, великі торгові інтеграційні системи та інтернет системи.
- **4. KeepinCRM** – зручна система для виконання автоматизації відносин з клієнтами.
- **5. HugeProfit** – це система управління товарним бізнесом, яка призначена для автоматизації основних фінансових процесів.
- **6. Pipedrive** – це CRM-система, яка спеціалізується на оптимізації процесів продажу та управлінні відносинами з клієнтами для підприємств.

Рисунок Ж.7 – Плакат 7

ТОПОВІ CRM СИСТЕМИ НА РИНКУ УКРАЇНИ

- **7. CleverBOX**: CRM – готовий набір інструментів для керування салоном краси, клінікою, центром.
- **8. PERFECTUM CRM+ERP** – український продукт. Система для всієї компанії може покривати всі процеси компанії.
- **9. LP-CRM** – створена для продажу товарів через сайти-лендінги, великі торгові інтеграційні системи та інтернет системи.
- **10. KeyCRM** – українська CRM система із фокусом на товарний бізнес.

Рисунок Ж.8 – Плакат 8

МЕТОД РОЗРАХУНКУ ПОТУЖНОСТІ

Константи розрахунку номінальної потужності електротранспорту

Константа	Опис	Значення
η	Коефіцієнти корисної дії	0,8
$f_{оп}$	Коефіцієнт для розрахунку опору повітря	0,89
$f_{тр}$	Коефіцієнт для розрахунку сили тертя	0,343

Дані для розрахунку номінальної потужності електротранспорту

Дані	Опис
v	швидкість руху електротранспорту, м/с
$m_т$	маса електротранспорту, кг;
$m_в$	маса водія, кг;
m	загальна маса електротранспорту з водієм, кг ($m = m_т + m_в$)

$$Pн = \frac{\left(f\text{тр} \cdot (m\text{т} + m\text{в}) + f\text{оп} \cdot \left(\frac{v^2}{2}\right)\right) \cdot v}{\eta}$$

Рисунок Ж.9 – Плакат 9

МЕТОД РОЗРАХУНКУ ЄМНОСТІ БАТАРЕЇ

Вхідні дані для розрахунку ененергетичної ємності акумуляторної батареї

Дані	Опис
S	Відстань пробігу на одному заряді батареї, км
v	Швидкість руху, км/год
Рн	Номінальна потужність електротранспорту, Вт
U	Напруга живлення електротранспорту, В

$$Q' = \frac{Pн \cdot \left(\frac{S}{v}\right)}{U}$$

Рисунок Ж.10 – Плакат 10

Рисунок Ж.11 – Плакат 11

Рисунок Ж.12 – Плакат 12

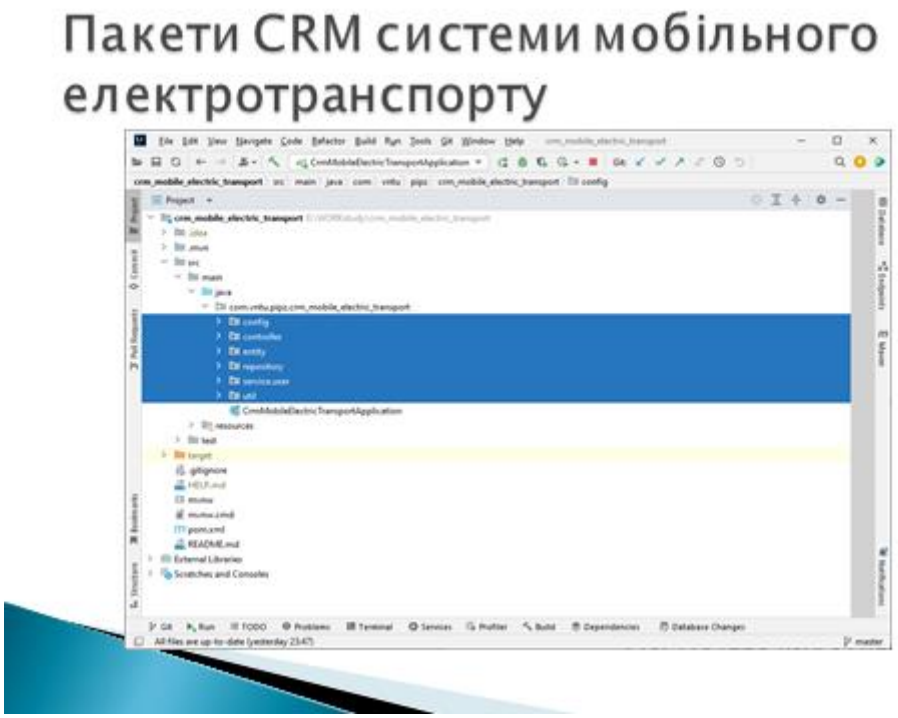


Рисунок Ж.13 – Плакат 13

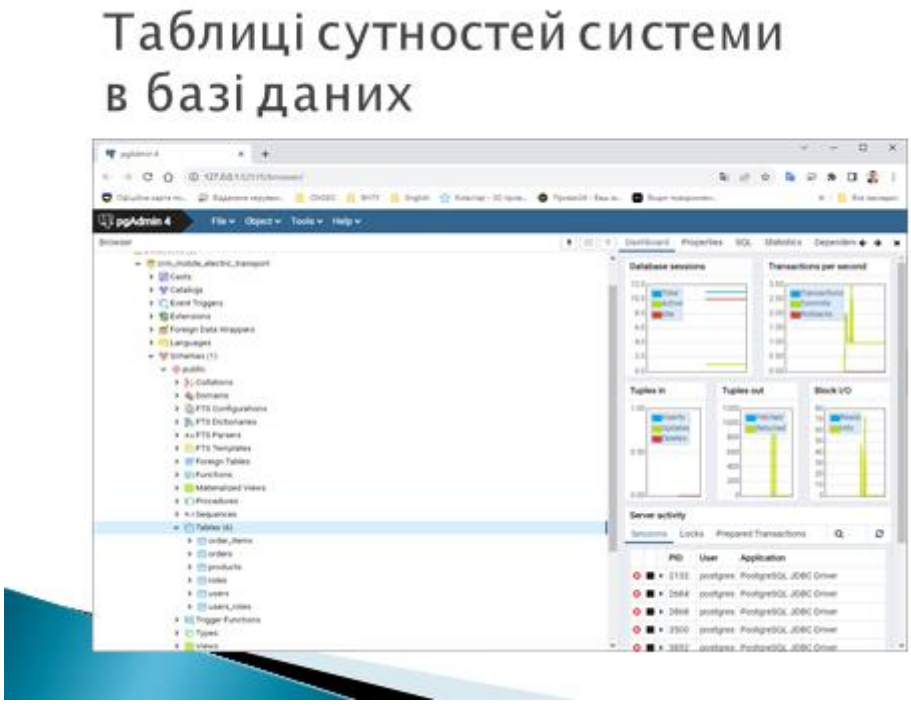


Рисунок Ж.14 – Плакат 14

ВИСНОВКИ

- Проведений аналіз сучасних методів і програмних засобів для автоматизації відносин з клієнтами та запропоновано обрати операційний вид CRM системи для вдосконалення методів і програмних засобів CRM системи мобільного електротранспорту.
- Розроблено методи для розрахунку необхідної потужності мобільного електротранспорту та енергетичної ємності акумуляторної батареї з використанням фізичних формул з розділів механіки та електродинаміки. На основі цих методів були виведені формули та створені моделі розрахунку вище зазначених показників за вхідними параметрами клієнта та відповідними характеристиками мобільного електротранспорту.
- Описано процес розробки програмного забезпечення і базуючись на аналізі та обгрунтуванні способів реалізації було обрано мову програмування Java, базу даних PostgreSQL та середовище розробки IntelliJ IDEA. Також, було спроектовано загальну архітектуру, основні модулі та бізнес логіку.
- Зроблено тестування програмного засобу.
- За результатами розрахунків економічної ефективності було встановлено, що розробка методів і програмних засобів CRM системи мобільного електротранспорту є економічно вигідною та може мати комерційний успіх.

Рисунок Ж.15 – Плакат 15