МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАДИМА ГЕТЬМАНА»
навчально-науковий інститут
«Інститут інформаційних технологій в економіці»

Кафедра інформаційних систем в економіці

галузь знань 12 «Інформаційні технології»

спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНИЙ БАКАЛАВРСЬКИЙ ПРОЄКТ

на тему

ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ З ОБЛІКУ ТА АНАЛІЗУ ПРОКАТУ ВЕЛОСИПЕДІВ

студента Гордієнка Мілана Парвез _____

Науковий керівник: к.е.н., проф.

_____ Ситник Н.В.

Кваліфікаційний бакалаврський проєкт допущений до захисту в Екзаменаційній комісії з атестації здобувачів вищої освіти
В.о. завідувача кафедри: к.е.н., доцент

_____ Тішков Б.О.

Київ 2022

Міністерство освіти і науки України
Державний вищий навчальний заклад
«Київський національний економічний університет
імені Вадима Гетьмана»
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»
денна форма навчання

Затверджую:
в.о. завідувача кафедри_____Тішков Б.О.
"____" _____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на кваліфікаційний бакалаврський проєкт

на тему: «Проєктування інформаційної системи з обліку та аналізу прокату велосипедів»
Гордієнка Мілана Парвез
Тему кваліфікаційного бакалаврського проєкту затверджено наказом ректора від «21» грудня 2021 р. № 1916-ст.

кваліфікаційний бакалаврський проєкт виконується на матеріалах
мережі прокату велосипедів

План на кваліфікаційний бакалаврський проєкт

Розділ І *Характеристика та аналіз предметної області*.
Розділ ІІ *Розробка вимог і моделювання інформаційної системи.*
Розділ ІІІ *Проєктування та реалізація компонентів системи.*

Об'єкт дослідження *інформаційні процеси, що характеризують систему обліку та аналізу прокату велосипедів*
Предмет дослідження *підходи та інформаційні технології використання засобів та методів ведення обліку та аналізу прокату велосипедів*
Мета кваліфікаційного бакалаврського проєкту
розробити інформаційну систему з *обліку та аналізу прокату велосипедів*

Конкретні завдання, які студент повинен виконати для досягнення поставленої мети:

У розділі I *Дослідити та характеризувати процеси обліку та аналізу прокату велосипедів. Провести аналіз існуючих підходів до автоматизації досліджуваної задачі.*

У розділі II *Провести вивчення та аналіз бізнес-вимог до проєктуємої системи. Розробити постановку задачі, навівши її характеристику, перелік вхідних і вихідних повідомлень та алгоритм її вирішення. Розробити інформаціну модель задачі. Провести моделювання системи з використанням інструментарію побудови UML-схем.*

У розділі III *Запроєктувати реляційну базу даних інформаційної системи, визначити основні вимоги до технічного забезпечення та представити його конфігурацію. Розробити програмне забезпечення для реалізації системи обліку та аналізу прокату велосипедів. Результати реалізації проілюструвати у відповідних додатках.*

Завдання підготував

науковий керівник **Ситник Ніна Василівна**

(підпис) (прізвище, ім'я, по-батькові)

"22" грудня 2021 р.

Завдання одержав

студент **Гордієнко Мілан Парвез**

(підпис) (прізвище, ім'я, по-батькові)

"_____"_____ 2021 р.

.

АНОТАЦІЯ

кваліфікаційного бакалаврського проєкту студента 4 курсу Навчально-наукового інституту «Інститут інформаційних технологій в економіці»

Гордієнка Мілана Парвез, виконаної на тему:

«Проєктування інформаційної системи з обліку та аналізу прокату велосипедів»

Київ: кафедра інформаційних систем в економіці, 2022 р.

У кваліфікаційному бакалаврському проєкті здійснено проєктування та реалізація інформаційної системи з обліку та аналізу прокату велосипедів для централізованої організації пунктів прокату велосипедів з деякими особливостями велосипедної бібліотеки. Досліджено сучасні інформаційні системи, які використовуються для обліку та аналізу прокату. Наведено історичну довідку про системи спільного використання велосипедів та їх першу появу. Об’єктом дослідження є організація обліку та аналізу роботи мережі пунктів прокату, а предметом дослідження є рішення завдань обліку прокату, розрахунок коштів та визначення доходу від прокату, проведення статистики популярних типів велосипедів та брендів, обсяги прокату велосипедів за пунктами прокату. На основі досліджень створено модель предметної області, визначено бізнес-вимоги, функціональні та нефункціональні вимоги до системи, її користувачів, основні терміни та поняття. Розглянуто програмні засоби, технології та методології, за допомогою яких можна вирішити задачу. Проведено моделювання поведінки системи, визначені варіанти використання. Розроблено тест-кейси по кожному тестовому випадку та наведено стислий опис тестів. Здійснено верифікування проєктних рішень через трасування описаних вимог до системи та зв’язків між ними.

РЕФЕРАТ

Кваліфікаційний бакалаврський проєкт містить 88 сторінок, 13 таблиць, 47 рисунків, список літератури з 28 найменувань, 3 додатків.

ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ З ОБЛІКУ ТА АНАЛІЗУ ПРОКАТУ ВЕЛОСИПЕДІВ

ІС, ІНФОРМАЦІЙНА СИСТЕМА, ПРОЄКТУВАННЯ, ООП, ПРОКАТ, ВЕЛОСИПЕД, ОРЕНДА, MICROSOFT VISUAL STUDIO, WINDOWS FORMS, .NET FRAMEWORK, C#, ОБЛІК, АНАЛІЗ.

Предметом дослідження є рішення завдань обліку прокату, розрахунок коштів та визначення доходу від прокату, проведення статистики популярних типів велосипедів та брендів, обсяги прокату велосипедів за пунктами прокату, підходи та інформаційні технології, що використовуються для вирішення цих завдань.

Об’єктом дослідження виступає проєктування інформаційної системи з обліку та аналізу прокату велосипедів.

Мета кваліфікаційного бакалаврського проєкту полягає в проєктуванні та розробці інформаційної системи з обліку та аналізу прокату велосипедів.

Апаратні та програмні засоби, що використовувались при проєктуванні: Microsoft Visual Studio, Microsoft SQL Server, SQL Server Management Studio, ER-діаграми, діаграми класів, станів, дій, прецедентів, послідовності, Windows Forms .NET Framework.

Результати досягнуті в процесі роботи – спроєктовано та реалізовано ІС з обліку та аналізу прокату велосипедів з використанням реляційної бази даних в середовищі СУБД.

Одержані результати можуть бути використані мережами систем спільного використання велосипедів з централізованою організацією та підтримкою можливостей велобібліотеки. Вона легко інтегрується в роботу та адаптується до міських та позаміських (лісових, гірських) умов. Має можливості до розширення, а також може вміщувати у своїх масивах даних багато записів різноманітних типів велосипедів багатьох відомих брендів та вести їх облік і аналіз.

ЗМІСТ

ВСТУП

Велосипед є одним з найпопулярніших засобів пересування у всьому світі. Багато людей щодня пересувається саме на велосипеді та використовують його не тільки для здійснення прогулянок, а й для поїздок на роботу або безпосереднього виконання робочих обов'язків.

Система мережі пунктів прокату надає можливість орендувати велосипед у пункті прокату, а після поїздки повернути його у будь-який інший пункт. Такі системи існують у більшості великих міст світу. Їх метою є надання мешканцям та гостям міста доступного велотранспорту для коротких поїздок, а також покращення екології та зменшення заторів на дорогах міста.

Зрозуміло, що згодом з'являється багато різних систем спільного використання велосипедів, які в свою чергу розширяються. Це передбачає збільшення кількості вхідних заявок та появу нових пунктів, що створює додаткове навантаження на менеджерів та ускладнює облік оренди та прокату, тому спеціалізоване програмне забезпечення стає необхідністю. З його допомогою можна значно прискорити багато бізнес-процесів та полегшити обробку запитів.

На сьогоднішній день існує велика кількість програмних засобів автоматизації та обліку прокату. Організації почали зводити до мінімуму документи на паперових носіях та розпочали використовувати електронний документообіг, автоматизацію різних бізнес-процесів. Але слід враховувати, що кожна організація може мати свої особливості, які не дозволяють їй працювати вже з представленими на ринку програмними продуктами.

Без автоматизації процесів введення, виведення, зберігання, обробки, реєстрації необхідної інформації, дані можуть бути втрачені, а ефективність їх пошуку буде дуже низькою.

В основному система обліку та аналізу прокату покликана допомогти зацікавленим особам задовольнити свої потреби та виконати бізнес-цілі. На сам перед така система автоматизує процеси, на які витрачається багато часу. Вона може допомогти вести облікову документацію та аналітику, що можна

буде використовувати для визначення кращого шляху розвитку бізнесу велопрокату, проводити оцінку велопарку, дізнатися найкращі місця для розміщення пунктів мережі. Вона також покликана полегшити рутинні процеси та зменшити кількість паперової документації. Все це підвищує ефективність роботи мережі пунктів прокату велосипедів, чим забезпечує підвищення зацікавленості для потенційних клієнтів.

Автоматизація за допомогою інформаційної системи необхідна на тих ділянках діяльності, де її поява збільшить швидкість виконання завдань і, відповідно, підвищить ефективність праці. Наочно це можна показати, наприклад, на роботі адміністратора компанії прокату: адміністратор без встановленого програмного забезпечення багато працюватиме і йому знадобиться більше часу, тому що кожного разу при запиті будь-якого звіту доведеться проводити розрахунки самостійно. Але якщо впровадити програму для автоматизації обліку послуг з прокату велосипедів, то будь-який звіт можна буде сформувати за допомогою натискання однієї кнопки.

РОЗДІЛ 1. ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1. Характеристика предметної галузі та об’єкта дослідження

Прокат (оренда) - це терміновий різновид договору оренди, відповідно з яким особою, яка здійснює підприємницьку діяльність, тимчасово у користування орендарю передається майно.

Дуже важливою частиною оренди є збір та обробка облікових даних, які спрямовані на вирішення поставленого управлінського завдання, що у широкому розумінні зветься економічним чи управлінським обліком.

Цей облік покликаний збирати фінансову та виробничу інформації з подальшим її аналізом для подальшого використання у прийнятті стратегічних рішень керівництвом компанії.

Слід зазначити, що інформаційна система повинна добре співвідноситися з організаційною структурою мережі прокату велосипедів, адже це запорука злагодженої та ефективної її роботи як основного облікового та аналітичного механізму. Безпосередньо, головним користувачем системи повинен бути адміністратор, який входить до відділу обліку прокату велосипедів мережі; а також аналітик відділу статистики та маркетингу. Ці два відділи є складовою бек-офісу мережі прокату велосипедів та виконують облікову, аналітичну та статистичну функції. У кожного відділу є свій керівник, які підпорядковуються директору мережі прокату велосипедів. Схема організаційної структури представлена на рисунку 1.1.

Рисунок 1.1 – Схема організаційної структури

Система обліку є найважливішою частиною бізнес-інструментів. Без цієї системи неможливо впоратися з обліком та аналізом прокату.

Основні функції, які повинні виконуватися системою з обліку та аналізу прокату велосипедів мають бути такими:

- ведення бази клієнтів пункту прокату, їх контактної інформації;
- облік усіх об’єктів пункту прокату – складається довідник об’єктів із необхідними характеристиками кожної одиниці: найменування, марка, модель, рік виготовлення, тип та інше;
- підготовка аналітичних звітів за об’єктами прокату, клієнтами, орендою;

Для вирішення задачі обліку та аналізу прокату велосипедів треба проаналізувати історію та організацію роботи систем спільного використання велосипедів. Такі системи можна поділити на декілька типів:

- Приватна система [1]. Приватний велопрокат розрахований передовсім на одиночні поїздки пішохідними зонами, парками чи набережними. Основний його принцип — взяття велосипеда за певну

оплату на декілька годин і повернення його по закінченні поїздки в те саме місце.

– Централізована система [2]. Централізована система має єдиного велооператора. Вона може бути створена як за муніципальною ініціативою, так і за приватною. Як комерційною, так і некомерційною. Ця мережа має численні пункти з великою кількістю велосипедів. Основний принцип, на відміну від приватного велопрокату, взяття велосипеда в одному пункті і залишення в іншому.
– Депозитна система [3]. Цей вид передбачає оренду велосипеда після внесення депозиту, який після поїздки і повернення велосипеда на станцію — повертається власникові. Депозит свого роду є гарантією для велооператора, що велоорендар обов'язково поверне майно.

Слід виділити окремий інноваційний проєкт системи спільного використання велосипедів, який передбачає, що жителям і гостям міста надається можливість користуватися велосипедами, подібно до користування книгами в громадських бібліотеках. На відміну від звичайної системи, бібліотечна має так званий асортимент — десятки велосипедів різного типу й виду: від міських до гірських, від механічних до електричних.

Отже, для вирішення задачі найкращим варіантом вважається створення системи обліку та аналізу прокату для централізованої системи спільного використання велосипедів з деякими можливостями велосипедної бібліотеки. Оскільки така система є найпоширенішою та зручною, а велобібліотека надає можливість створення асортименту велосипедів. Така спільна система буде працювати як централізована, але з великим асортиментом велосипедів. Наприклад, пункти прокату такої системи можуть знаходитися за містом та надавати можливість орендувати гірські велосипеди для прогулянок у лісовій місцевості. Тобто, такі пункти можуть працювати також біля різних культурних та природних пам’яток та інших туристичних об’єктів, які знаходяться не тільки в містах, а і у гірській та лісовій місцевостях.

Облік та аналіз прокату для такої системи дуже важливий, тому що вона має багато різних типів велосипедів та місць, де може здійснюватися прокат, а отже управлінський облік та його аналіз саме те, що допоможе у прийнятті вірних рішень для розвитку та подальшого розширення компанії з велопрокату.

Клієнтові, щоб скористатися послугами велопрокату потрібно, попередньо зареєструвавшись, обрати потрібний велосипед та у додатку зайти у камеру, якою просканувати штрих-код велосипеду. Далі він обирає час оренди та підтверджує замовлення, після чого велосипед буде розблоковано на обраний проміжок часу, до закінчення якого, клієнт повинен буде повернути велосипед на цей або інший пункт прокату цієї ж мережі. Усі ці дані система з обліку та аналізу повинна буде обробляти та подавати у вигляді звітів та результатів аналізу.

Основуючись на перелічених процесах система повинна опрацьовувати мінімум шість об’єктів, такі як: Оренда, Клієнт, Велосипед, Бренд, Пункт Прокату, Місто. Кожен з цих об’єктів характеризується певним набором параметрів, що описують деяку частину предметної області, яку представляє даний об’єкт.

1.2. Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі

У теперішніх реаліях подібні системи розробляються та використовуються переважно як десктопні програми, встановлені на робочих станціях користувачів системи і поєднаних однією локальною мережею, яка з’єднана з іншими мережами. Технології, які використовуються та надаються можливість створювати подібні системи можуть різнитися, але найкращим варіантом можна вважати Windows Forms. Це інтелектуальна клієнтська технологія для .NET Framework. Windows Forms дозволяє розробляти прості в розгортанні програми з графічним інтерфейсом, здатні працювати при наявності або відсутності підключення до Інтернету.

Windows Forms використовується разом з Microsoft Visual Studio, – лінійкою продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення.

Windows Forms включає широкий набір елементів управління, які можна додавати на форми: текстові поля, кнопки, списки, що розкриваються, перемикачі та навіть веб-сторінки.

За допомогою конструктор Windows Forms перетягування в Visual Studio можна легко створювати додатки. Досить виділити елемент керування курсором і помістити його в потрібне місце на формі.

Зважаючи на те, що система повинна буде аналізувати дані з БД та виводити їх на екран, але вже в заданому структурованому вигляді, форми у Windows Forms являють собою дуже зручний інструмент. Для кожного процесу є можливість створити свою форму, на якій будуть знаходитися усі необхідні функції для роботи з цим процесом та формування звітності.

На сьогодні представлено великий вибір програмних засобів, для автоматизації різних видів діяльності, у тому числі й у сфері прокату. Для того, щоб вирішити поставлені задачі, необхідно провести аналіз аналогічних програмних засобів.

EasyPro [4]**.** Програма обліку прокату від компанії Easy Software. Призначена для автоматизації пункту або мережі прокату. Її можна використовувати для оренди будь-якого виду обладнання або інвентарю:

- велосипедів, роликів та ковзанів, лиж та сноубордів та іншого спортивного інвентарю;
- автомобілів, мотоциклів, скутерів;
- будівельної техніки, спецтехніки, будівельного обладнання та електроінструментів
- суконь, костюмів та обладнання для проведення заходів;
- дитячих іграшок;
- відео та аудіо техніки;
- та будь-якого іншого товару;

Ціна базової версії програми для обліку прокату становить 3655,52 гривень. Ціна версії з функціоналом сервісу становить 4763,25 гривень. Передбачена передплата за хостинг програми на сервері, сума якої залежить від кількості користувачів, які працюють із програмою. Ціна для 1 користувача 110,77 гривень у місяць. На рисунку 1.1 представлено інтерфейс програми.

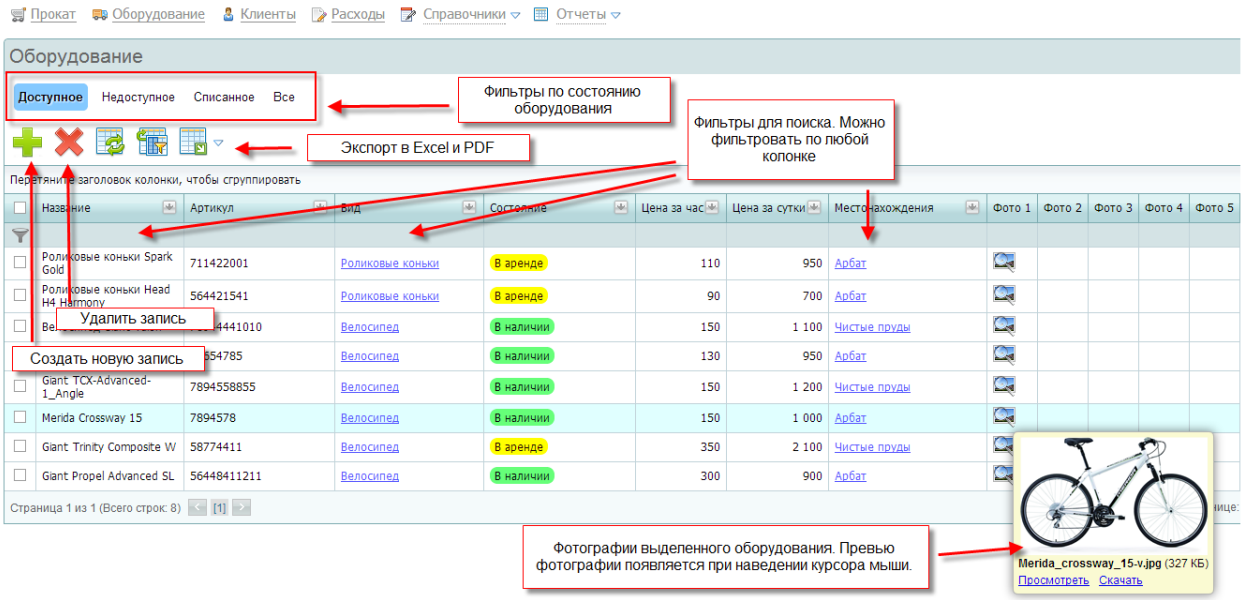

Рисунок. 1.2 – Інтерфейс EasyPro

Основні можливості програми для обліку прокату:

- зручний каталог обладнання, налаштований з урахуванням специфіки користувача;
- база клієнтів;
- чорні списки клієнтів;
- швидкий пошук потрібних товарів на вибрані дати;
- нагадування про прострочені прокати та інші події;
- можливість роботи з комп'ютера чи планшета;
- облік витрат;
- оцінка ефективності реклами;
- продаж товарів;
- необхідні звіти;
- експорт даних до Excel, PDF, Word;
- друк документів та звітності;

- можливість інтеграції програми з сайтом користувача;
- СМС та e-mail розсилання;
- можливість настроювання програми з урахуванням особливостей бізнесу користувача.

Переваги:

- великий набір звітів, що настроюються, експорт в різні формати;
- не потребує встановлення;
- скорочення витрат часу на облік;
- система працює через інтернет на хмарному сервері.

Недоліки:

- додаткова плата за кожне робоче місце;
- надлишковий функціонал.

Висновок: дане рішення має ряд переваг, однак його використання недоцільне з огляду на наявні недоліки, насамперед щодо вартості продукту, впровадження, супроводу та непотрібних функцій.

РемОнлайн [5]**.** Комплексна програма для обліку оренди і прокату, яка являє собою хмарний сервіс, для роботи з яким потрібен тільки інтернет. Вона має зручний облік, аналіз асортименту з виявленням популярних та рентабельних товарів, брендів. Можна відстежувати показники бізнесу в зручному форматі графіків за допомогою аналітичного звіту. На рисунку 1.2 представлено інтерфейс РемОнлайн.

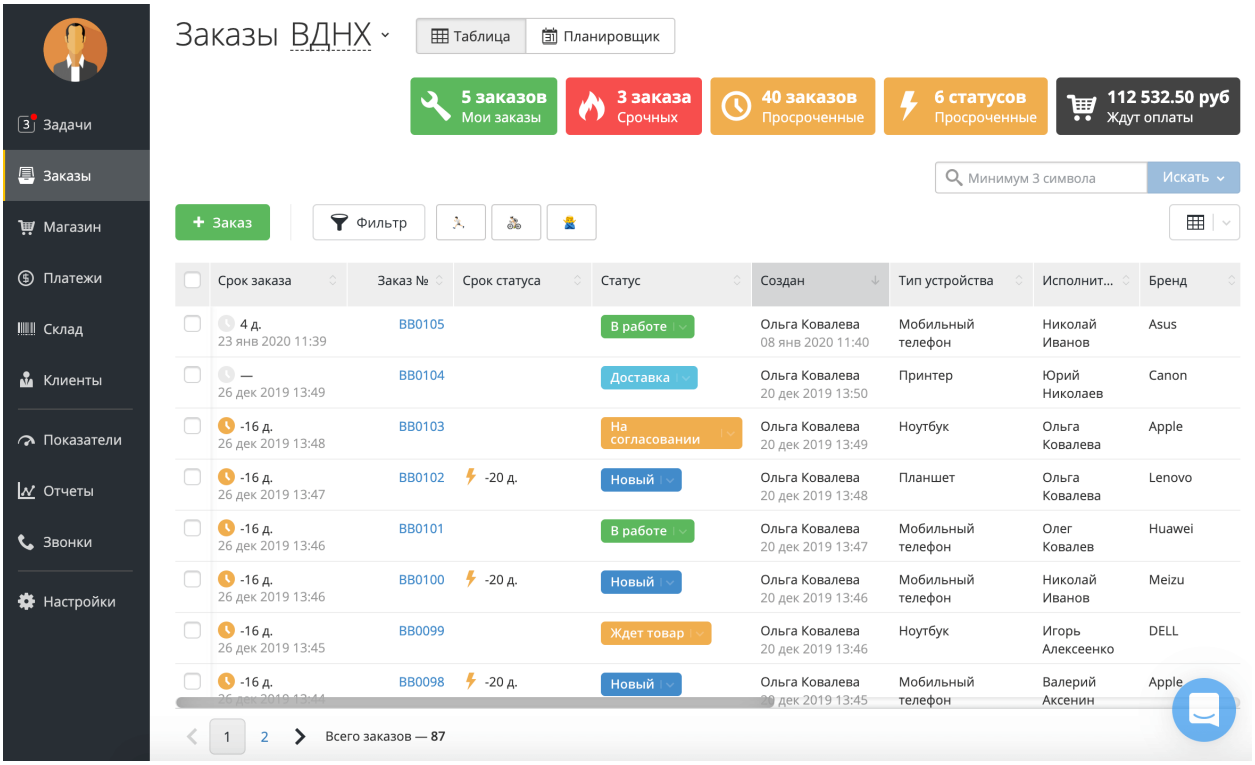

Рисунок. 1.3 – Інтерфейс РемОнлайн

Наявні тарифи:

– Хобі: від 278,07 гривень/місяць. Додаткова плата за співробітника 61,79 гривень/місяць. Максимум 5 співробітників 100 замовлень/продажів на місяць. Не більше однієї локації (пункту прокату). Є 14 днів пробного періоду.
– Стартап: від 587,04 гривень/місяць. Додаткова плата за співробітника 123,59 гривень/місяць та 278,07 гривень/місяць за локацію. Без обмежень кількості замовлень/продажів, співробітників та локацій.
– Бізнес: від 896 гривень/місяць. Додаткова плата за співробітника 123,59 гривень/місяць та 587,04 гривень/місяць за локацію. Без обмежень.

Основні можливості програми:

– оформлення замовлень;
– облік оренди та прокату;
– складський облік;
– автоматичні повідомлення клієнтів;
– миттєве формування та друк документів;
– контроль роботи менеджерів;

– автоматизація обліку та оренди різних видів товарів.

Переваги:

– інтуїтивно зрозумілим інтерфейсом, який легко адаптувати під свій бізнес;
– легка інтеграція з іншими сервісами для обліку і зв'язку;
– зручна перевірка статусу замовлень і наявних прокатних товарів;
– автоматизація звітності та відправки повідомлень;
– полегшена аналітика та оцінка ефективності бізнесу.

Недоліки:

– надлишковий функціонал;
– додаткова плата за співробітника та локацію;
– обмеження співробітників у стартовому тарифі;
– переплата за надлишковий функціонал;
– наявність тільки підписки;
– підходить більше для складського обліку на підприємстві.

Висновки: програма має великий корисний функціонал, але не настільки необхідний, щоб переплачувати. Вона доволі зручна, але заточена більше для обліку товарів на складі та роботи з клієнтами.

Торгсофт [6]**.** Програма підходить як для одного пункту прокату чи магазинів, так і для мережі. Працює без підключення до інтернету, якщо використовується один пункт. Має декілька модулів, один з яких прокат товару. Дозволяє вести облік товару, який знаходиться або планується бути в прокаті, заборонити реалізацію товару, який знаходиться в прокаті, роздрукувати договір прокату, видаткову накладну, акт виконаних робіт, фінансовий документ прийому застави, фінансовий документ повернення застави. На рисунку 1.3 представлено інтерфейс Торгсофт.

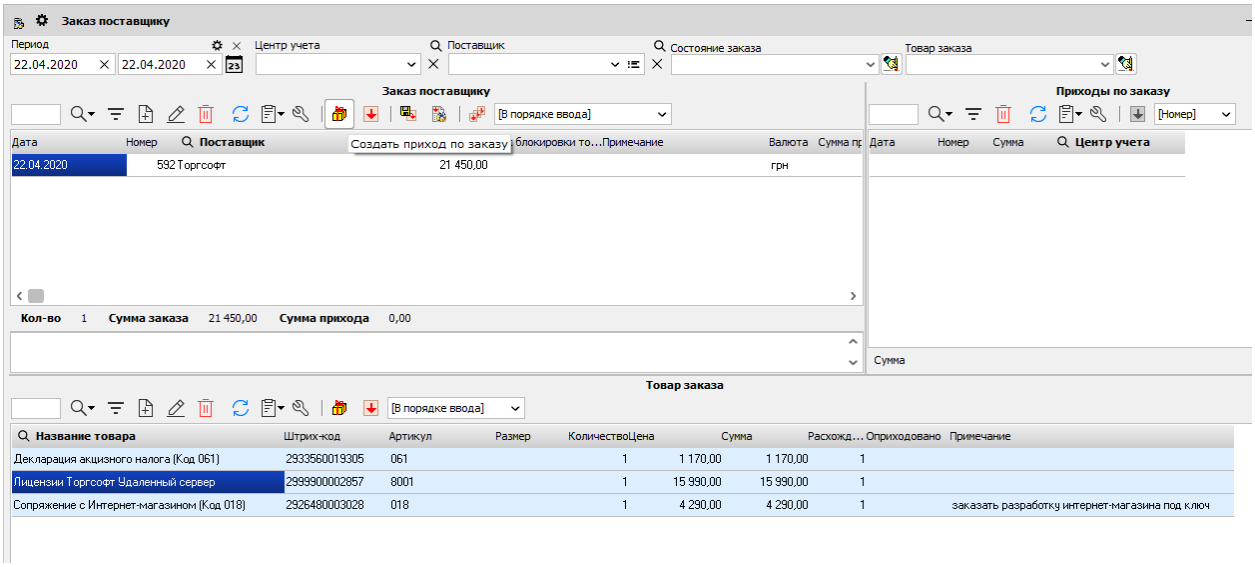

Рисунок. 1.4 – Інтерфейс Торгсофт

Ціни:

- Торгсофт-Онлайн: від 590 гривень.
- Торгсофт-Старт: 5290 гривень – безстроково, 2860 гривень – оренда на один рік.
- Торгсофт-Ультра: від 7560 гривень за 7 робочих місць до 10990 гривень за одне робоче місце. Оренда на один рік: від 3780 гривень за 7 робочих місць до 5495 гривень за одне робоче місце.

Основні можливості програми для обліку і аналізу:

- аналіз результатів торгівлі (прокату) по мережі або центрам обліку: валовий дохід, витрати, дохід;
- результати торгівлі (прокату) за видами товарів (велосипедів) з фільтрами по центрам обліку, клієнтам і виробникам;
- аналіз розміру виручки за групами товарів, інформація для звітності.
- порівняння двох періодів;
- перегляд виторгу магазину в розрізі постачальників товару;
- перегляд виторгу магазину по сезонах;
- побудова рейтингу найбільш продаваних (прокатних) товарів за період, де був проданий товар і в якій кількості – в цифрах і у вигляді графіка.
- аналіз того, наскільки вигідно продавати (прокатувати) той чи інший товар.

Переваги:

- готове рішення для підприємця й не вимагає доопрацювань;
- немає щорічної підписки, ліцензія не має обмеження за часом;
- безкоштовні оновлення програми;
- демоверсія на 30 днів;
- надає дві години гарантійної технічної підтримки для допомоги клієнтам для віддаленого налаштування та консультацій по програмі;
- немає складної термінології, а для роботи не потрібні спеціальні бухгалтерські знання;
- понад 250 навчальних відео, вичерпна інструкція, інформація на сайті;
- оперативна техпідтримка онлайн 7 днів на тиждень.

Недоліки:

- надлишковість;
- орієнтована більше під торгові точки та магазини, а модуль прокату з'явився як додаткова функція.

Висновки: програма з великою кількістю аналітичних та облікових функцій, які реалізовані в одному модулі, має матеріал для навчання, техпідтримку та проста у використанні. Але більше підходить для обліку продажів мереж магазинів, ніж для обліку прокату велосипедів.

Після проведення аналізу аналогів можна зробити висновки, що більшість програм реалізована з повним функціоналом, який призначений не тільки для обліку та аналізу прокату, а також для продажів, роботи з клієнтами, обліку продажів, інвентаризації та ін. У зв'язку з цим, а також великою вартістю без можливості покупки лише потрібного функціоналу, виникла потреба створити власну систему з обліку та аналізу прокату велосипедів.

У таблиці 1.1 наведена порівняльна характеристика існуючих програмних засобів.

Таблиця 1.1

АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ ЗАСОБІВ ТА ЇХ ПОРІВНЯННЯ

Критерій	EasyPro	РемОнлайн	Торгсофт
Інтерфейс	Зручний	Зручний	Зручний
Функціонал	Великий	Великий	Великий
Техпідтримка	Відсутня	Присутня	Присутня (постійна)
Процес впровадження	Безкоштовно	Безкоштовно	Входить у ціну
Демоверсія	Відсутня	14 днів	30 днів
Функції обліку та аналізу	Присутні у достатній кількості	Присутні та націлені на складський облік	Окремий модуль, з великою кількістю функцій
Ціна	від 3655,52 грн. до 4763,25 грн. / 110 грн в місяць на одного	від 278,07 грн/місяць до 896 грн/місяць + доплата за користувачів	від 3780 грн. до 10990 грн.

РОЗДІЛ 2. РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ

2.1. Аналіз і специфікація вимог до інформаційної підсистеми

Перш за все треба визначити зацікавлених осіб та їх бізнес-цілі, щоб сформувати бізнес-вимоги, які б забезпечували цінність та описували характеристики системи з точки зору кінцевого користувача. Система не може існувати, якщо в цьому немає потреби. У кожної зацікавленої особи є свої цілі, які висвітлюють їх зацікавленість у розробці системи, яка б змогла ці цілі задовольнити.

До зацікавлених осіб можна віднести:

- Директор мережі пунктів прокату велосипедів;
- Адміністратор мережі пунктів прокату велосипедів;
- Аналітик відділу статистики та маркетингу мережі пунктів прокату велосипедів;
- Клієнт мережі пунктів прокату велосипедів.

Проаналізувавши потреби кожної зацікавленої особи можна визначити такі бізнес-вимоги:

Директор мережі пунктів прокату велосипедів:

- Збільшення масштабів мережі та кількості замовлень завдяки автоматизованому аналізу велопрокату. На основі проведених досліджень та аналізування можна буде створювати плани на подальше розширення мережі.
- Ревізія велопарку та пошук найкращої за обсягами замовлень марки велосипеду. Пошук та використання найкращих марок велосипедів збільшить прибуток, а списання непопулярних марок зменшить зайві трати.
- Аналіз місць та тенденцій, пошук найкращих варіантів розташування пунктів прокату велосипедів. Пошук найпопулярніших місць збільшить обсяг замовлень.

Адміністратор мережі пунктів прокату велосипедів:

- Полегшення ведення обліку замовлень прокату велосипедів. Пришвидшить облік замовлень та покращить ефективність обліку.
- Полегшення ведення довідників клієнтів та пунктів прокату мережі. Пришвидшить реєстрацію нових клієнтів та пунктів прокату, забезпечить швидкий пошук потрібних клієнтів та пунктів прокату.
- Полегшення ведення обліку та моніторингу стану наявних велосипедів у кожному пункті прокату. Забезпечить швидкий облік наявних велосипедів, додання нових або списання старих та непридатних за станом велосипедів.

Аналітик відділу статистики та маркетингу мережі пунктів прокату велосипедів:

- Полегшення у складанні аналітичної звітності. Ефективне складання звітів за раніше створеними шаблонами.
- Ефективний аналіз статистики популярних велосипедів та пунктів прокату за районами міста. Ефективний аналіз велопарку конкретних пунктів прокату та їх популярності дозволить ефективно управляти усією мережею, закриваючи неприбуткові пункти, які несуть збитки. Ефективний збір даних про те, в яких частинах міста найкращий обсяг прокату дозволить покращити планування подальшого розвитку мережі.

Клієнт мережі пунктів прокату велосипедів:

- Поява нових пунктів прокату у районі клієнта. Поява пунктів прокату біля з місцем проживання, роботи та іншої діяльності клієнта забезпечить зручне пересування.
- Надання акційних пропозицій для постійних клієнтів. Знижки, промокоди та акції будуть заохочувати постійних клієнтів частіше користуватися послугами мережі пунктів прокату.

При формуванні вимог слід визначити, що саме та як буде виконувати система і хто буде мати можливість користуватися функціями системи.

Адміністратор повинен мати можливість переглядати, додавати, змінювати та видаляти дані прокату за допомогою форм редагування для кожної таблиці з відповідними полями. Після обрання потрібної таблиці на формі вибору таблиці, відкривається форма з відповідною таблицею. На формі є можливість обрати декілька дій: додати, видалити, змінити та вийти. Після натискання кнопки "вийти" відбувається повернення до форми вибору таблиці.

При натисканні кнопок "Додати", "Видалити" та "Змінити" відкриваються відповідні форми редагування таблиць з відповідними до обраної таблиці полями та двома кнопками: "Ок", що відповідає за збереження внесених даних або видалення, "Повернутися", що відповідає за вихід з форми редагування та повернення до форми з обраною таблицею.

Адміністратор повинен мати можливість проаналізувати наявні дані за критеріями ціни, дати, пункту прокату, моделі та бренду велосипеда. При натисканні кнопки "Аналіз" на головній формі, відкривається форма аналізу даних, на якій можна обрати потрібний критерій у відповідному полі та натиснути кнопку "Аналіз", після чого на екран буде виведена таблиця, яка відповідає виборці.

Адміністратор повинен мати можливість зберегти результати аналізу у форматі файлу Excel. Після проведення аналізу на формі аналізу даних, треба натиснути кнопку "Зберегти у Excel". Відкриється вікно шляху збереження, де можна вказати назву файлу та місце збереження.

Сумуючи вищесказане можна виділити такі загальні вимоги до інформаційної системи з обліку та аналізу прокату велосипедів.

Функціональні вимоги:

- Req A - Облік даних прокату;
    - Req A.1 – Додавання даних;
    - Req A.2 – Змінення даних;
    - Req A.3 – Видалення даних.
- Req B – Аналіз даних прокату.
    - Req B.1 – Аналіз за критерієм;

        - Req B.1.1 – Аналіз за ціною та датою;
        - Req B.1.2 – Аналіз за маркою та брендом велосипеда;
        - Req B.1.3 – Аналіз за пунктом прокату, ціною та маркою велосипеда.
    - Req B.2 – Збереження аналізу в файл Excel.

Нефункціональні вимоги:

- В системі повинна бути можливість переглядати дані з БД у табличному форматі;
- В системі повинна бути можливість швидко формувати звіт на основі аналізу даних;
- В системі повинно бути передбачено відображення валюти – грн;
- Система повинна бути розділена на різні модулі, які не будуть залежати один від одного;
- В системі повинен бути обробник винятків, який буде відстежувати правильність введення даних;
- Файл збереженого звіту аналізу даних повинен мати формат xls, xlsx;
- Система повинна давати можливість переглядати дані з БД;
- Система повинна застерігати від видалення записів за неіснуючим ключем;
- В системі всі поля вводу даних повинні відповідати своєму типу даних.

На рисунку 2.1 представлено діаграму вимог з поданням ієрархії вимог та зв’язками між ними.

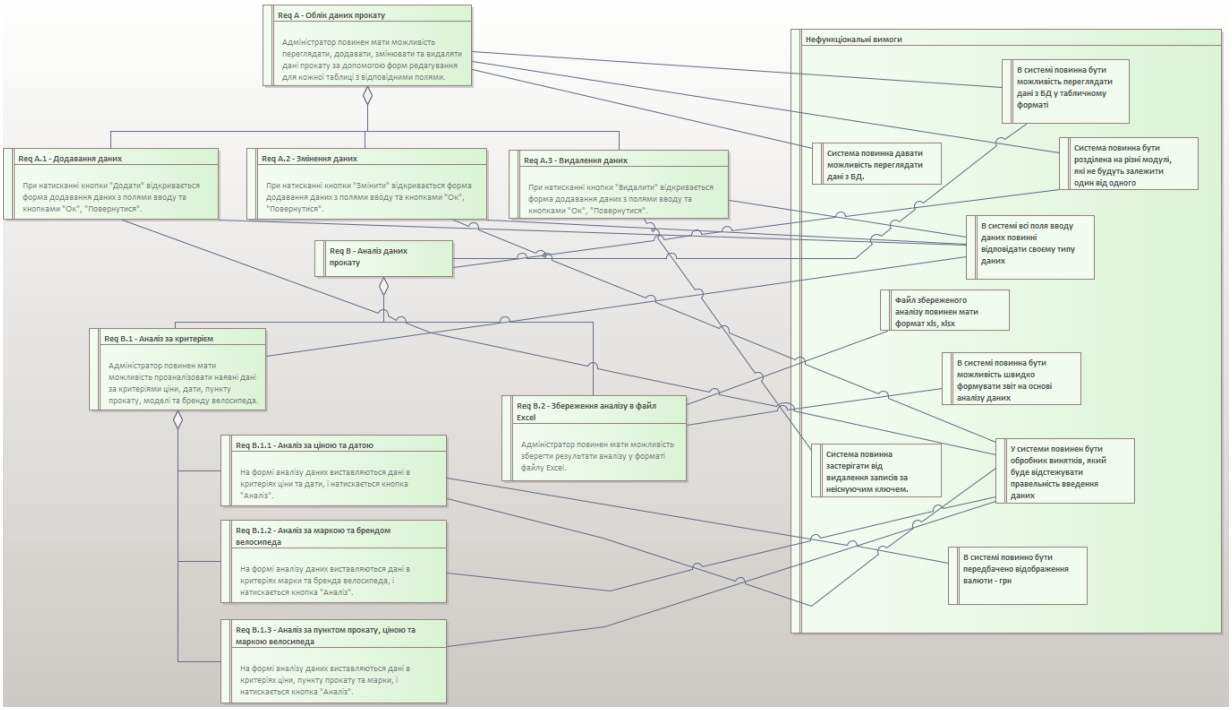

Рисунок 2.1 – Діаграма вимог з поданням ієрархії вимог та зв’язками між ними

В таблиці 2.1 представлено специфікації вимог.

Таблиця 2.1

СПЕЦИФІКАЦІЇ ВИМОГ

№/ вимоги	Стисла назва вимоги	Повна назва вимоги	Тип	Статус	Складність	Пріоритетність
Req A	Облік даних прокату	Адміністратор повинен мати можливість переглядати, додавати, змінювати та видаляти дані прокату за допомогою форм редагування для кожної таблиці з відповідними полями.	Функціональний	Затверджено	Низька	Високий
Req A.1	Додавання даних	При натисканні кнопки "Додати" відкривається форма додавання даних з полями вводу та кнопками "Ок", "Повернутися".	Функціональний	Обов’язково	Низька	Високий
Req A.2	Змінення даних	При натисканні кнопки "Змінити" відкривається форма додавання даних з полями вводу та кнопками "Ок", "Повернутися".	Функціональний	Обов’язково	Низька	Високий
Req A.3	Видалення даних	При натисканні кнопки "Видалити" відкривається форма	Функціональний	Обов’язково	Низька	Високий

		додавання даних з полями вводу та кнопками "Ок", "Повернутися".				
Req B	Аналіз даних прокату	Адміністратор повинен мати можливість проводити аналіз даних прокату за допомогою форми аналізу	Функціональний	Затверджено	Середня	Високий
Req B.1	Аналіз за критерієм	Адміністратор повинен мати можливість проаналізувати наявні дані за критеріями ціни, дати, пункту прокату, моделі та бренду велосипеда.	Функціональний	Обов'язково	Середня	Високий
Req B.1.1	Аналіз за ціною та датою	На формі аналізу даних виставляються дані в критеріях ціни та дати, і натискається кнопка "Аналіз".	Функціональний	Затверджено	Середня	Середній
Req B.1.2	Аналіз за маркою та брендом велосипеда	На формі аналізу даних виставляються дані в критеріях марки та бренда велосипеда, і натискається кнопка "Аналіз".	Функціональний	Затверджено	Середня	Середній
Req B.1.3	Аналіз за пунктом прокату, ціною та маркою велосипеда	На формі аналізу даних виставляються дані в критеріях ціни, пункту прокату та марки, і натискається кнопка "Аналіз".	Функціональний	Затверджено	Середня	Середній
Req B.2	Збереження аналізу у файл Excel	Адміністратор повинен мати можливість зберегти результати аналізу у форматі файлу Excel.	Функціональний	Затверджено	Низька	Середній
Req N.1	Перегляд даних у табличному у форматі	В системі повинна бути можливість переглядати дані з БД у табличному форматі	Нефункціональний	Обов'язково	Середня	Середній
Req N.2	Формування звіту аналізованих даних	В системі повинна бути можливість швидко формувати звіт на основі аналізу даних	Нефункціональний	Затверджено	Середня	Середній
Req N.3	Відображення валюти	В системі повинно бути передбачено відображення валюти – грн	Нефункціональний	Затверджено	Низька	Низький
Req N.4	Розподілення на незалежні модулі	Система повинна бути розділена на різні модулі, які не будуть залежати один від одного	Нефункціональний	Обов'язково	Середня	Високий
Req N.5	Оброблення винятків	В системі повинен бути обробник винятків, який буде відстежувати	Нефункціональний	Затверджено	Середня	Середній

		правильність введення даних				
Req N.6	Формат файлу звіту	Файл збереженого звіту аналізу даних повинен мати формат xls, xlsx	Нефункціональний	Затверджено	Низька	Середній
Req N.7	Перегляд даних з БД	Система повинна давати можливість переглядати дані з БД	Нефункціональний	Затверджено	Низька	Високий
Req N.8	Заборонено видаляти записи за неіснуючим ключем	Система повинна застерігати від видалення записів за неіснуючим ключем	Нефункціональний	Обов'язково	Низька	Високий
Req N.9	Відповідність типу даних	В системі всі поля вводу даних повинні відповідати своєму типу даних	Нефункціональний	Обов'язково	Середня	Високий

2.2. Постановка задачі та алгоритм розв'язання задачі

2.2.1 Постановка задачі.

Характеристика задачі. Система призначена для ведення обліку та аналізу прокату велосипедів. Вона повинна: працювати з базою даних пунктів прокату; мати можливість зчитувати, обробляти, редагувати, додавати та видаляти інформацію; проводити аналіз за критеріями та виводити зведення даних; зберігати інформацію у вигляді Excel файлу.

Система обробляє дані, які надаються від головної системи спільного використання велосипедів протягом кожного дня. Дані можуть оновлюватися у будь який момент часу, так як клієнти здійснюють прокат велосипедів протягом усього дня у різний проміжок часу.

Вихідна інформація використовуються для статистичного огляду, аналізування роботи пунктів прокату велосипедів, визначення доходу від прокату, попиту за марками велосипедів, обсягами прокату.

Взаємозв'язок між задачами можна прослідкувати на інформаційній моделі задачі (див. рис. 2.2).

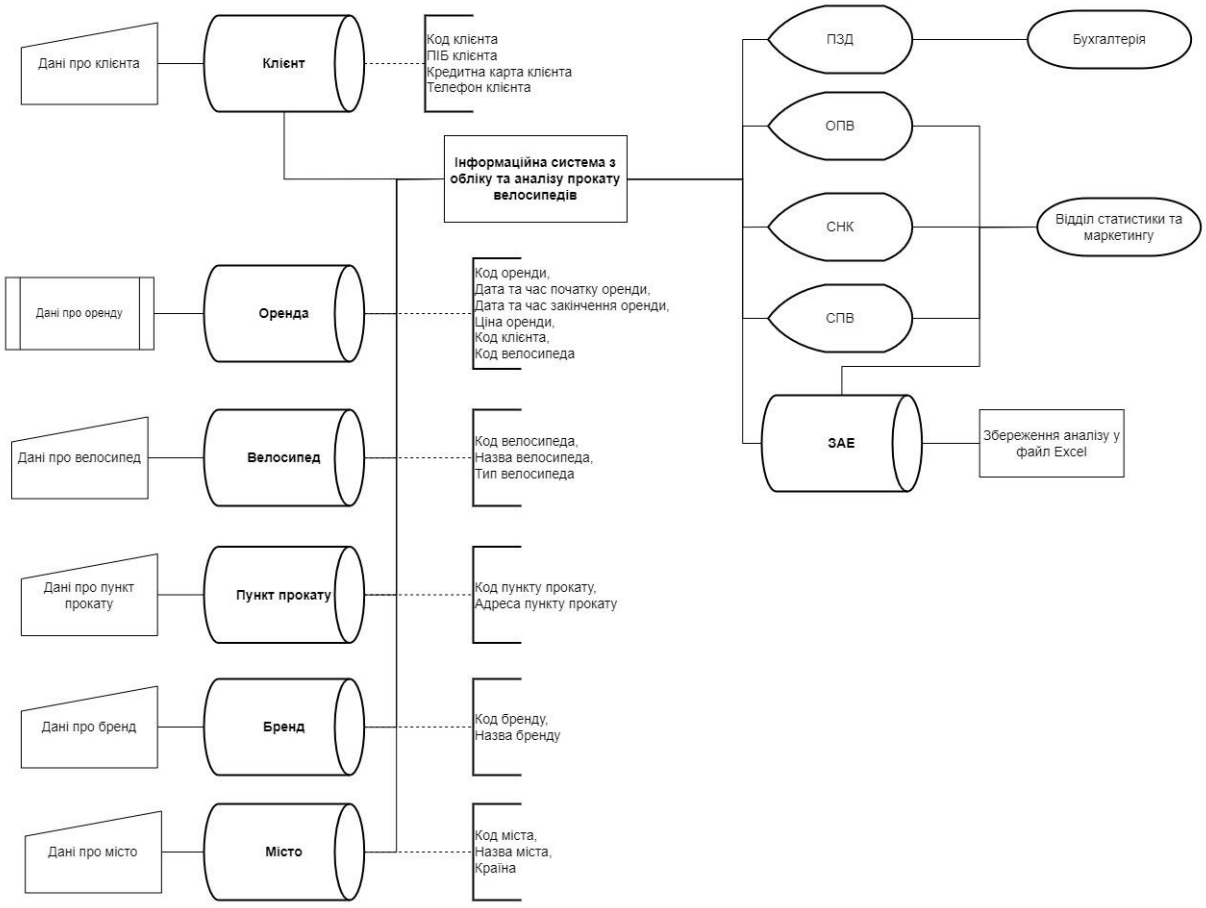

Рисунок 2.2 – Інформаційна модель задачі

Вихідна інформація. До вихідних повідомлень належать підрахунок доходу від прокату, збір статистики реєстрації нових клієнтів, статистики популярних велосипедів та пунктів прокату та її аналіз, обсяги прокату велосипедів за районами, експорт звіту у Excel файл.

Перелік вихідних повідомлень та їх характеристик наведений у таблиці 2.2.

Таблиця 2.2

ПЕРЕЛІК І ОПИС ВИХІДНИХ ПОВІДОМЛЕНЬ

№ з/п	Назва вихідного повідомлення	Ідентифікатор	Форма подання і вимоги до неї	Періодичність видання	Термін видання і допустимий час затримки	Користувачі інформації
1	2	3	4	5	6	7
1	Прибуток за день	ПЗД	Екранна форма	Раз на день	Кожного дня до 20-00 год.	Бухгалтерія
2	Обсяги прокату велосипедів	ОПВ	Екрана форма	Раз на тиждень	Кожного понеділка до 18-00 год.	Відділ статистики та маркетингу
3	Статистика нових клієнтів	СНК	Екрана форма	Раз на місяць	30 (31) число до 14-00 год.	Відділ статистики

						та маркетингу
4	Статистика популярних велосипедів та прокатних пунктів	СПВ	Екрана форма	Раз на місяць	30 (31) число до 18-00 год.	Відділ статистики та маркетингу
5	Звіт аналізу в Excel	ЗАЕ	Файл Excel	Раз на місяць	30 (31) число до 18-00 год.	Відділ статистики та маркетингу

Вхідна інформація. Система працюватиме з даними прокату, клієнтів, та велосипедів. Після здійснення клієнтом оренди велосипеду на одному з пунктів прокату у таблицю бази даних заноситься запис, де вказується велосипед, який був орендований, код клієнта, дата та час початку оренди. Після закінчення оренди в запис додається дата та час закінчення оренди. Розраховуються термін оренди, на основі якого формується ціна і потім списується з картки клієнта (див рис. 2.4). Усі дані про клієнта заносяться самим клієнтом під час реєстрації у системі, яка відповідає за безпосередню взаємодію з ним – мобільний або веб-додаток. У свою чергу система обліку та аналізу користується цими даними.

До переліку вхідних повідомлень відносяться дані клієнта, який збирається орендувати велосипед:

– Прізвище, ім’я, по батькові;
– Дані кредитної карти.
– Номер телефону

Також, до вхідних повідомлень відносяться дані про велосипеди та бренди:

– Назва велосипеду;
– Тип велосипеду;
– Назва бренду.

Вони заносяться до бази даних, коли на пунктах прокату з’являються нові екземпляри велосипедів, яких раніше не було.

Очевидно, що при відкриті нового пункту, вхідними даними також будуть дані про пункт прокату з такими атрибутами:

– Назва пункту прокату;

– Адреса пункту прокату.

Окрім цього, вхідними даними також можуть бути назва та країна розташування міста, де працюють пункти прокату мережі.

Перелік усіх вхідних повідомлень та їх характеристик наведений у таблиці 2.3.

Таблиця 2.3

ПЕРЕЛІК І ОПИС ВХІДНИХ ПОВІДОМЛЕНЬ

№ з/п	Назва вхідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Джерело
1	2	3	4	5	6
1	Дані про клієнта	ClientData	В електронному вигляді	Під час реєстрації	Форма реєстрації в мобільному або веб-додатку
2	Дані про оренду	RentalData	В електронному вигляді	Після оформлення прокату	Форма оформлення прокату в мобільному або веб-додатку
3	Дані про велосипед	BicycleData	В електронному вигляді	Під час внесення нового екземпляру велосипеда	Форма додавання/редагування даних
4	Дані про пункт прокату	DepartData	В електронному вигляді	Після відкриття нового пункту прокату	Форма додавання/редагування даних
5	Дані про бренд	BrandData	В електронному вигляді	Під час внесення нового бренду	Форма додавання/редагування даних
6	Дані про місто	CityData	В електронному вигляді	Під час внесення нового міста	Форма додавання/редагування даних

2.2.2 Алгоритм розв’язання задачі.

Математичний опис. В таблиці 2.4 представлено атрибути, які характеризують предметну область.

Таблиця 2.4

АТРИБУТИ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ЇХ ХАРАКТЕРИСТИКИ

Назва атрибута	Ідентифікатор	Умовне позначення	Характеристика атрибута
Код велосипеда	BICYCLE_ID	i	якісний, груповий
Код пункту прокату	DEPART_ID	p	якісний, груповий
Код оренди	RENTAL_ID	x	якісний, груповий
Марка велосипеду	BICYCLE_NAME	m	якісний, груповий
Ціна за разову поїздку	PRICE_ONE	O	кількісний, розрах.

Ціна за хв оренди	PRICE_MINUTE	M	кількісний, норм-розціночний
Плата за відкріплення велосипеда	COST	C	кількісний, норм-розціночний
Сума доходу	SUM	S	кількісний, розрах.
Дата	D	d	якісний, груповий
Кількість хвилин оренди велосипеда	BICYCLE_MINUTE	Y	кількісний, норм-розціночний
Коефіцієнт популярності велосипеда	MODEL_POP	MP	кількісний, норм-розціночний
Кількість хвилин оренди усіх велосипедів	MIN_SUM	MS	кількісний, розрах.
Дата та час початку прокату	DATE_FROM	F	кількісний, норм-розціночний
Дата та час закінчення прокату	DATE_TO	T	кількісний, норм-розціночний
Термін оренди	PERIOD	P	кількісний, розрах.
Підсумкові дані оренди	COUNT_DATA	U	кількісний, розрах.

P_i – термін оренди і-го велосипеда = (дата та час закінчення прокату і-го велосипеда – дата та час початку прокату і-го велосипеда) * 1440 – кількість хвилин в одній добі.

$$P_i = (T_i - F_i) * 1440, \quad (2.1)$$

O_i – ціна за поїдку і-го велосипеда = ціна за хв оренди і-го велосипеда * термін оренди і-го велосипеда + плата за відкріплення і-го велосипеда.

Плата за відкріплення і-го велосипеда – C_i

Ціна за хв оренди і-го велосипеда – M_i

$$O_i = M_i * P_i + C_i, \quad (2.2)$$

$S_{ip}d$ – сума доходу за d-дату і-го велосипеда р-го пункту прокату розраховується за формулою:

$$S_{ip}d = \sum_{x=1}^{n} O_{xip}d = O_{1ip}d + O_{2ip}d + \cdots + O_{nip}d, \quad (2.3)$$

Кількість хвилин оренди за d-дату і-го велосипеда р-го пункту прокату розраховується за формулою:

$$Y_{ip}d = \sum_{x=1}^{n} P_{xip}d = P_{1ip}d + P_{2ip}d + \cdots + P_{nip}d, \quad (2.4)$$

Кількість хвилин оренди усіх велосипедів за d-дату р-го пункту прокату розраховується за формулою:

$$MS_p d = \sum_{i=1}^{n} Y_{ip} d = Y_{1p} d + Y_{2p} d + \cdots + Y_{np} d, \quad (2.5)$$

де i – код велосипеда.

$MP_{ip} d$ - коефіцієнт популярності i-го велосипеда p-го пункту прокату за d-дату = кількість хвилин оренди за d-дату i-го велосипеда p-го пункту прокату / кількість хвилин оренди усіх велосипедів за d-дату p-го пункту прокату.

$$MP_{ip} d = \frac{Y_{ip} d}{MS_p d}, \quad (2.6)$$

$U_{xpm} d$ – підсумкові дані від прокату всіх велосипедів за d-дату p-го пункту прокату m-марки велосипеда отримуються за формулою:

$$U_{xpm} d = \sum_{x=1}^{n} U_{xpm} d = U_{1pm} d + U_{2pm} d + \cdots + U_{npm} d, \quad (2.7)$$

де x – код оренди,
n – верхній ліміт

Отримати підсумкові дані від прокату всіх велосипедів за d-дату, які входять у певний діапазон ціни можна за формулою:

$$U_x d = x_a d \le x_a d, x_{a+1} d \dots x_{b-1} d, x_b d \le x_b d, \quad (2.8)$$

де x_a – початкова ціна,
x_b – кінцева ціна

Отримати підсумкові дані від прокату всіх велосипедів m-марки b-бренда можна за формулою:

$$U_{xmb} = \sum_{x=1}^{n} U_{xmb} = U_{1mb} + U_{2mb} + \cdots + U_{nmb}, \quad (2.9)$$

Отримати підсумкові дані від прокату всіх велосипедів за d-дату m-марки p-го пункту прокату, які входять у певний діапазон ціни можна за формулою:

$$U_{xpm} d = x_{apm} d \le x_{apm} d, x_{a+1pm} d \dots x_{b-1pm} d, x_{bpm} d \le x_{bpm} d, \quad (2.10)$$

де x_{apm} – початок діапазону,
x_{bpm} – кінець діапазону

Алгоритм розв'язання задачі на ЕОМ. Алгоритми розв'язання задачі. Основні алгоритми розв'язання задачі представлені на рисунках далі.

На рисунку 2.3 представлено схема алгоритму обліку даних прокату.

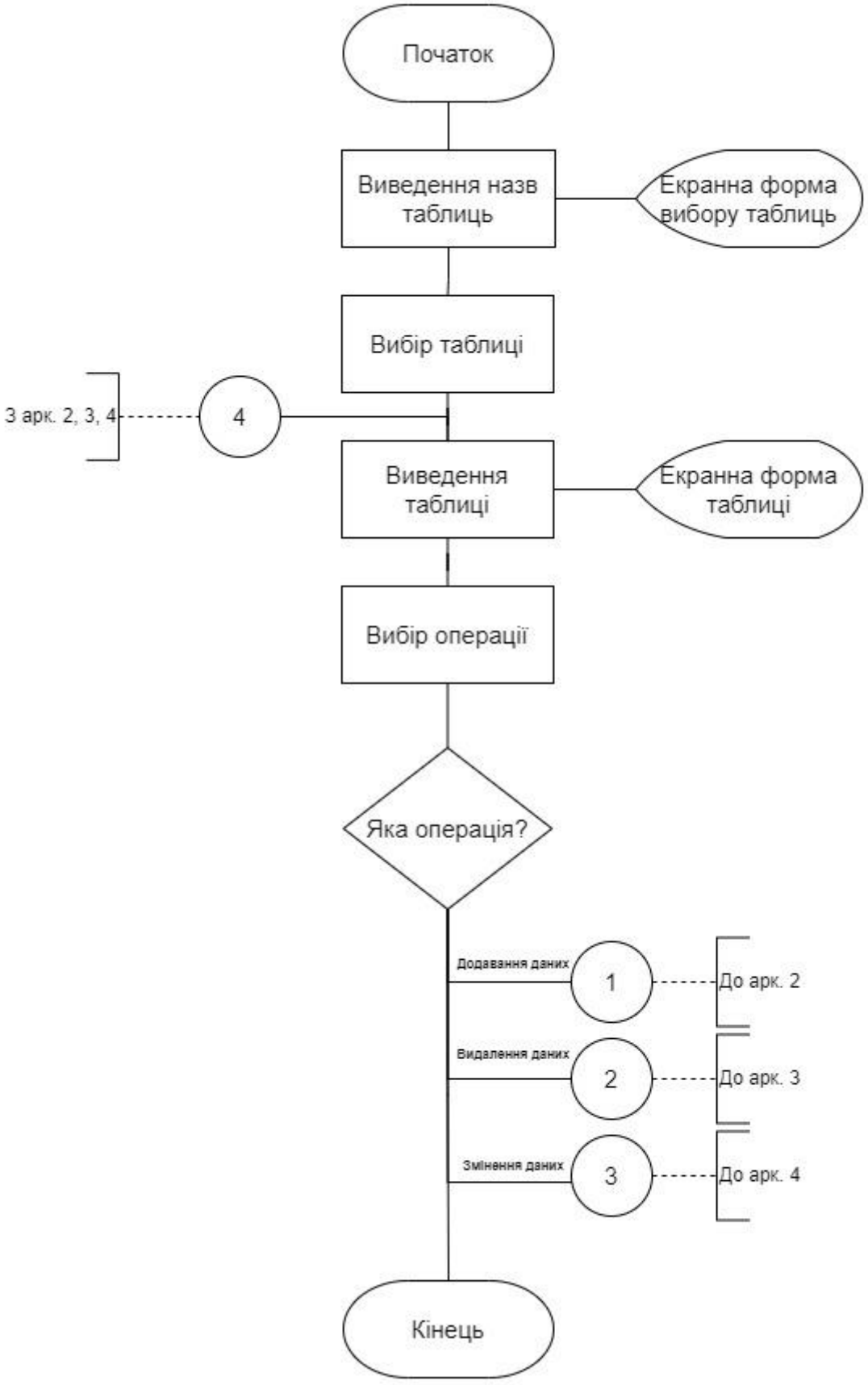

Рисунок 2.3. – Схема алгоритму обліку даних прокату. Арк. 1

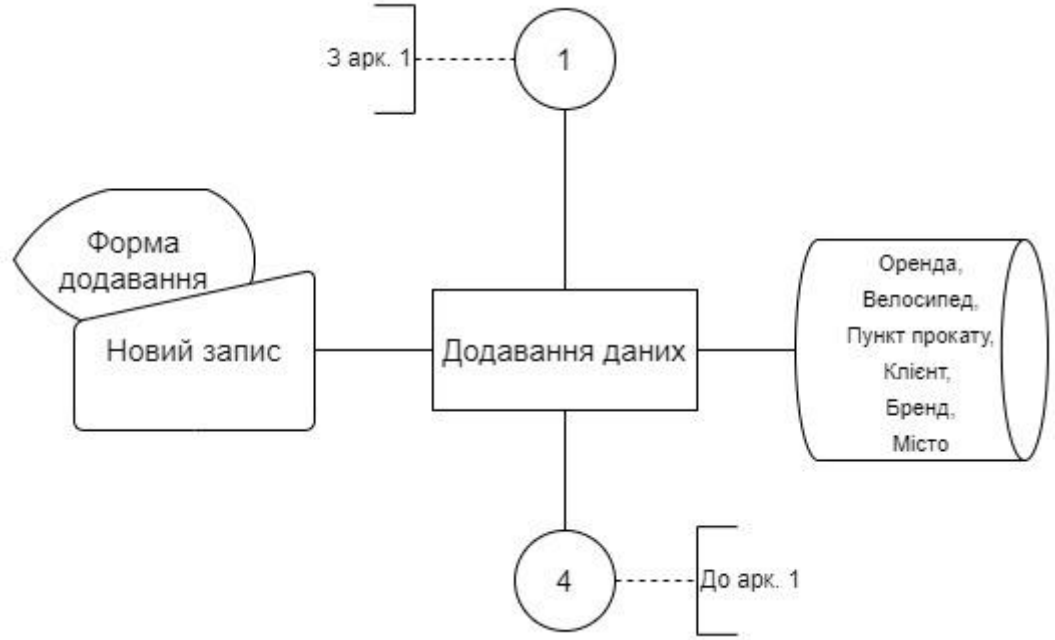

Рисунок 2.3. Арк. 2

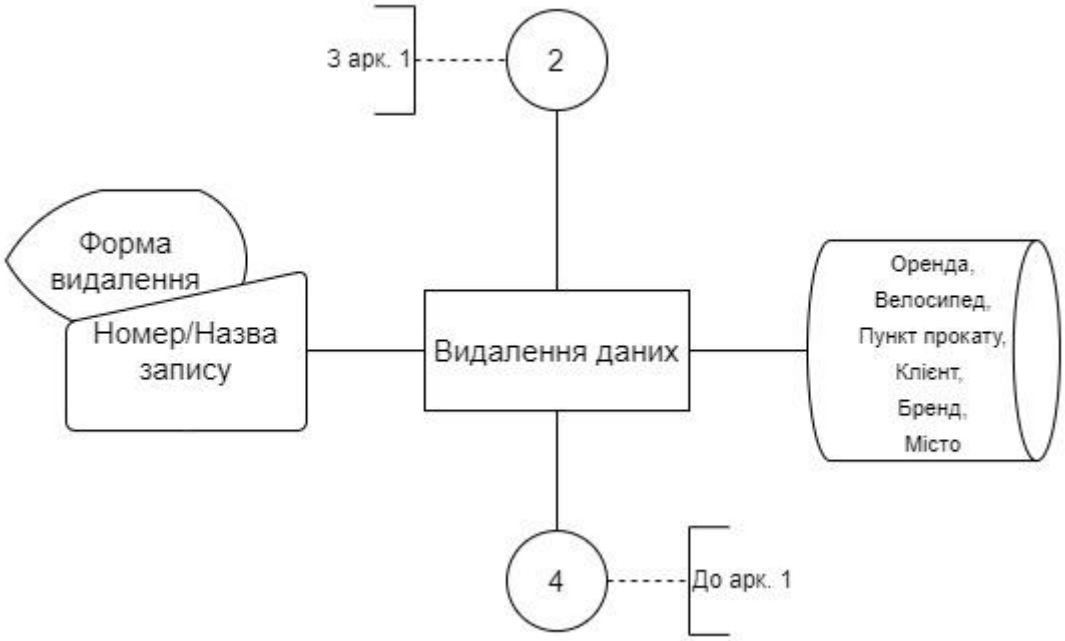

Рисунок 2.3. – Арк. 3

Рисунок 2.3. – Арк. 4

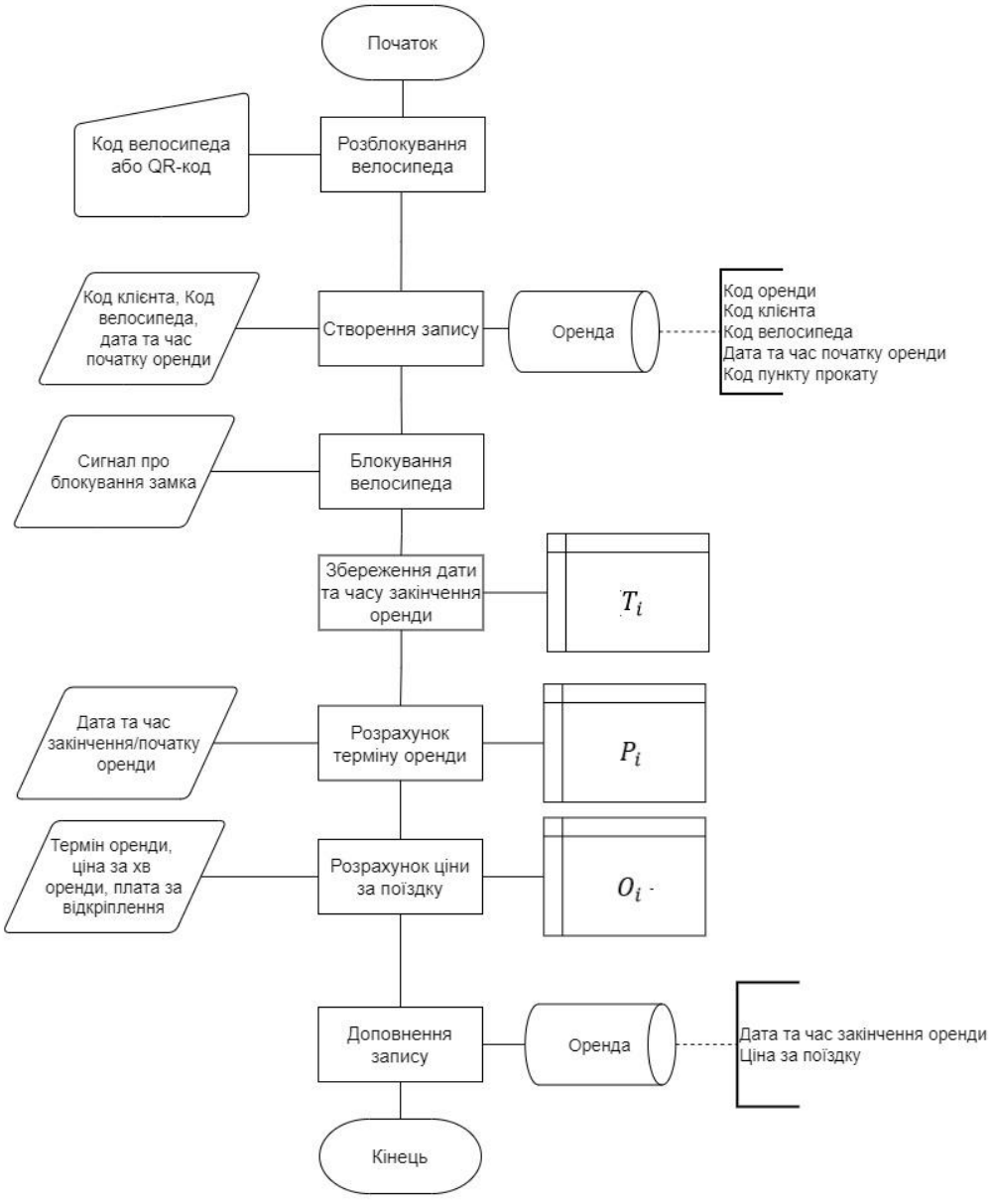

Рисунок 2.4. – Схема алгоритму автоматичного додавання даних оренду

У системі є можливість робити аналіз даних за критеріями. На рисунку 2.5 представлено схему алгоритму аналізу за критеріями. Де критеріями виступають аналіз за ціною та датою, аналіз за маркою велосипеда та брендом, аналіз за пунктом прокату, ціною та маркою велосипеда. Тут можна розрахувати дохід за певну дату, обсяг прокату певного велосипеду та знайти популярний велосипед. Кожний аналіз можна зберегти у файл Excel.

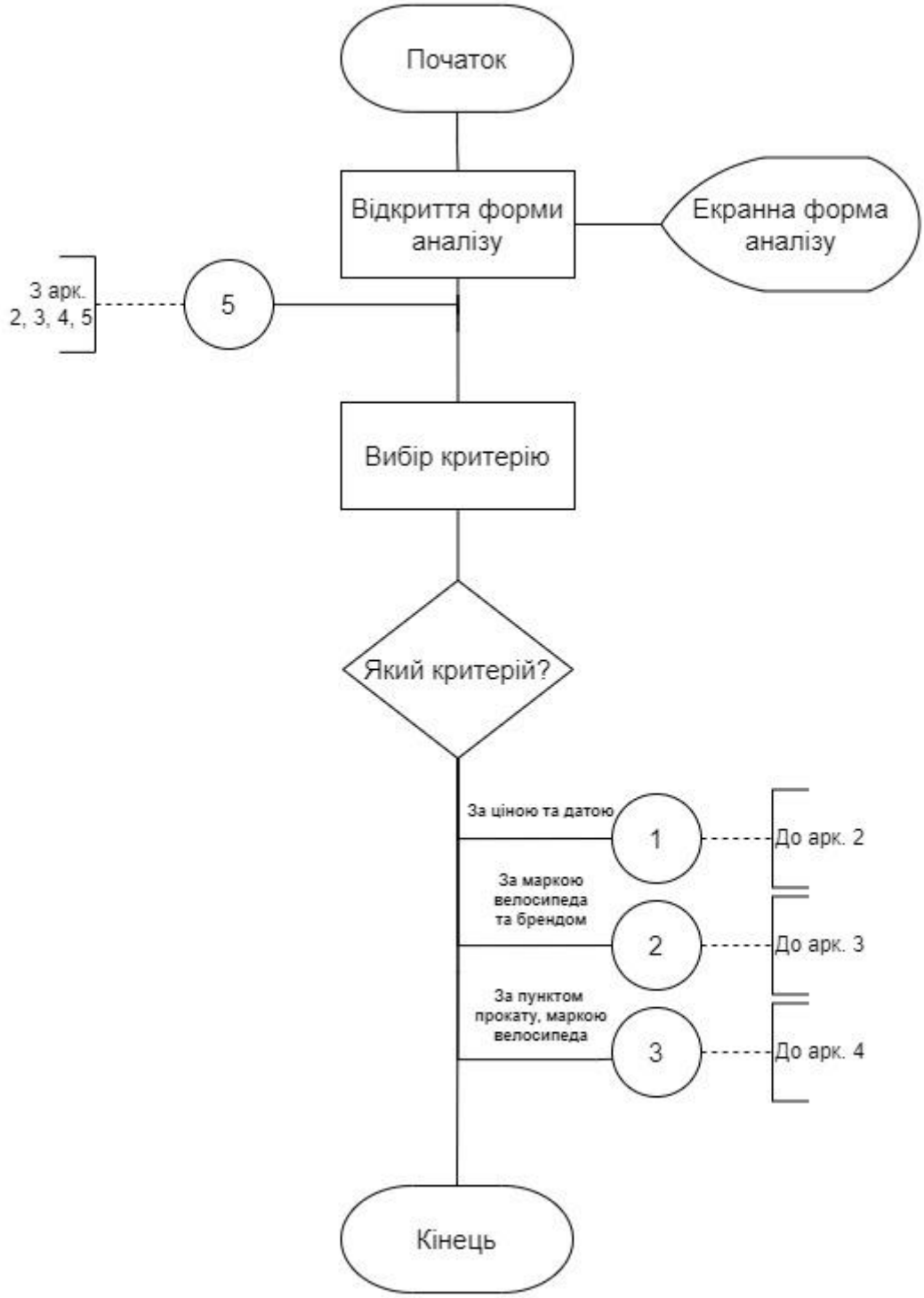

Рисунок 2.5. – Схема алгоритму аналізу за критеріями. Арк. 1

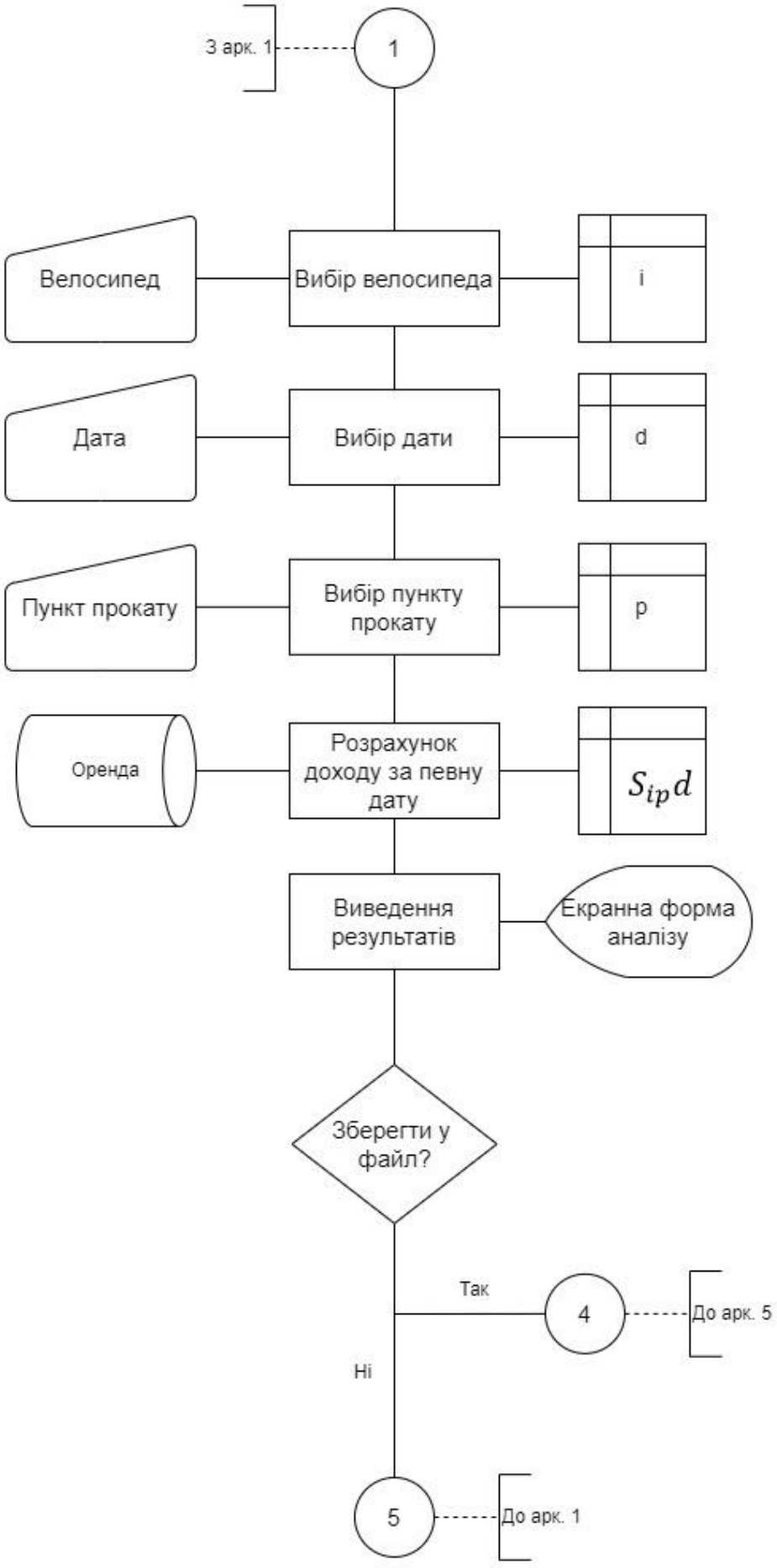

Рисунок 2.5. – Арк. 2

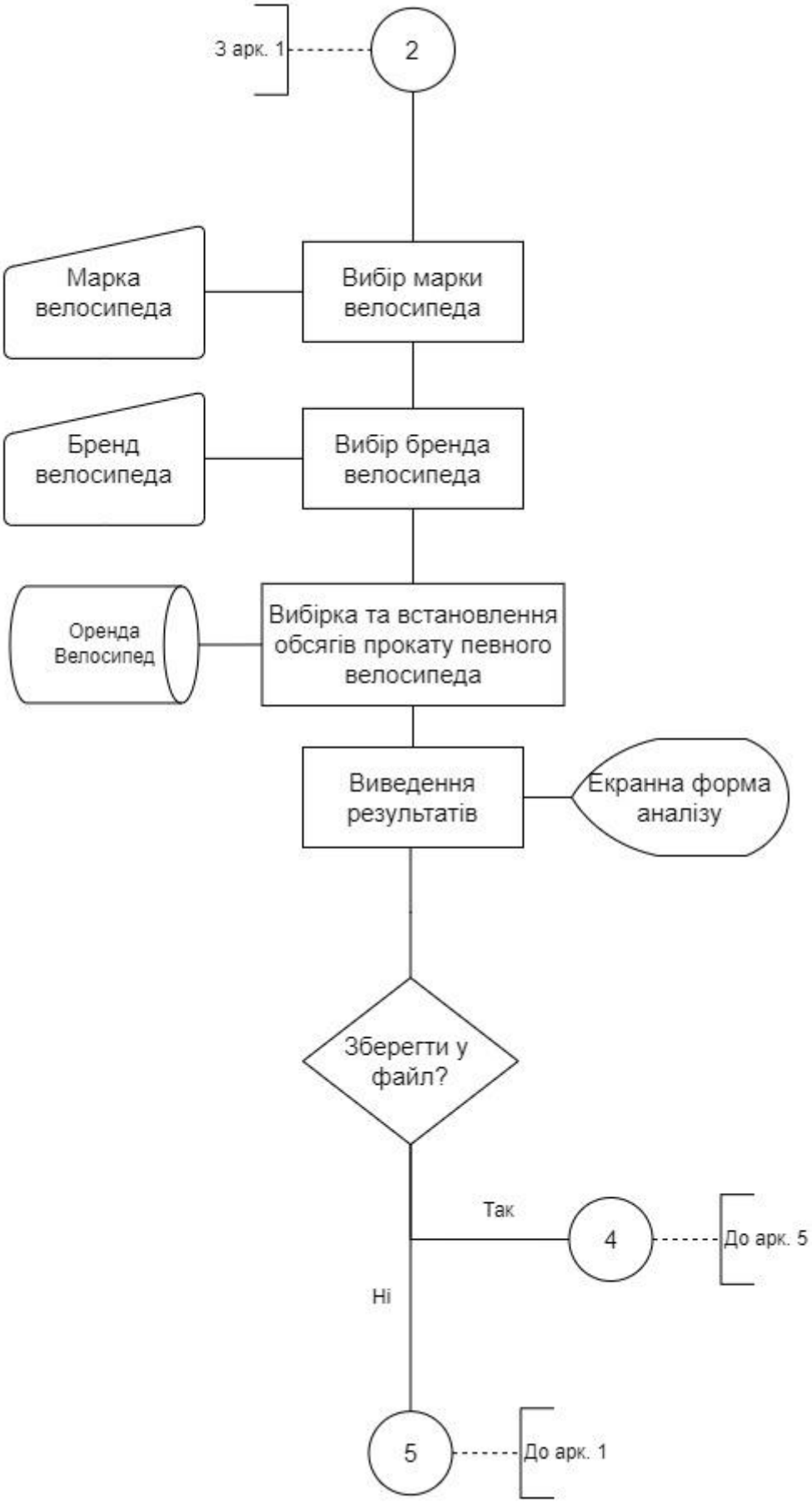

Рисунок 2.5. – Арк. 3

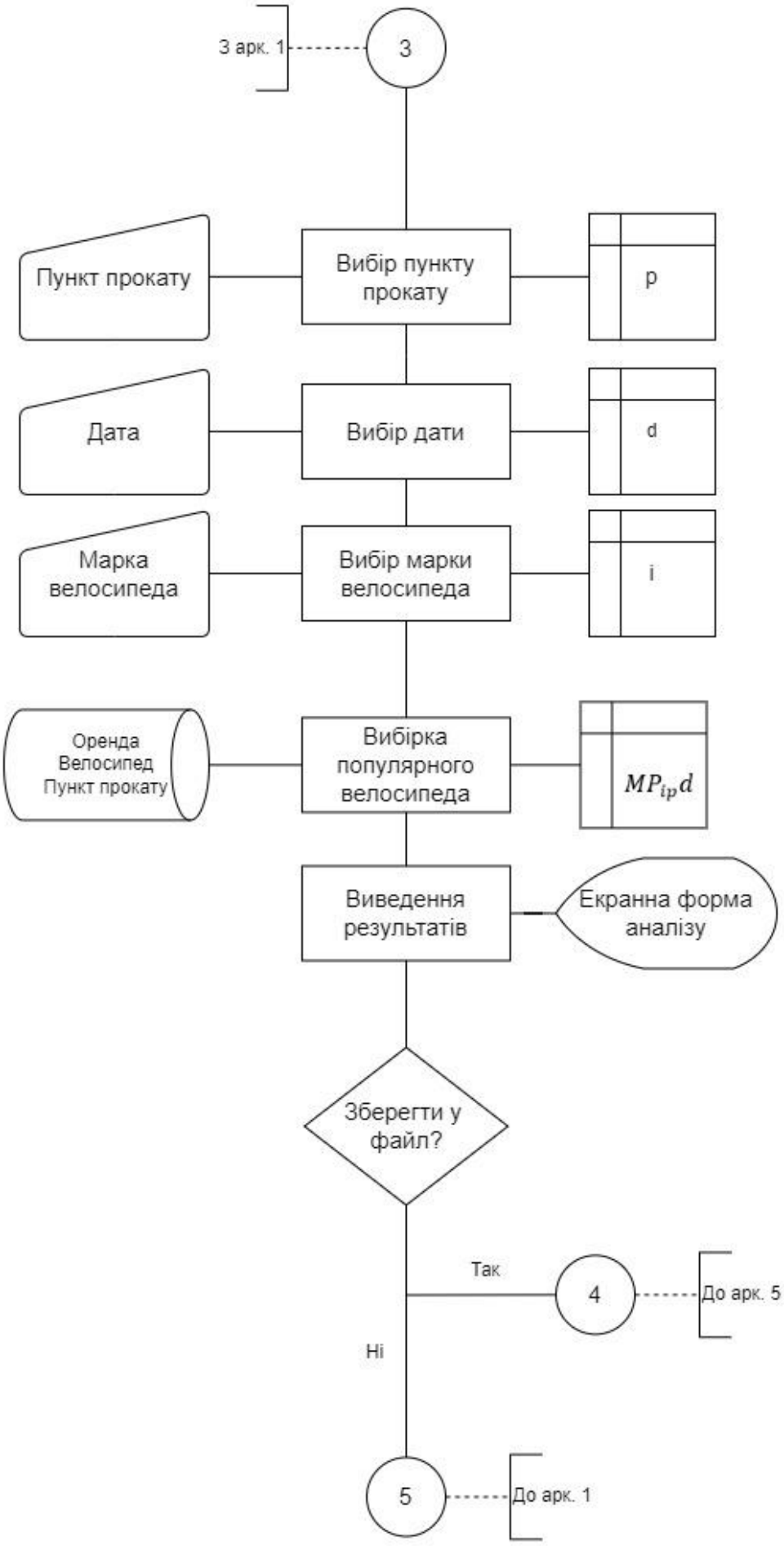

Рисунок 2.5. – Арк. 4

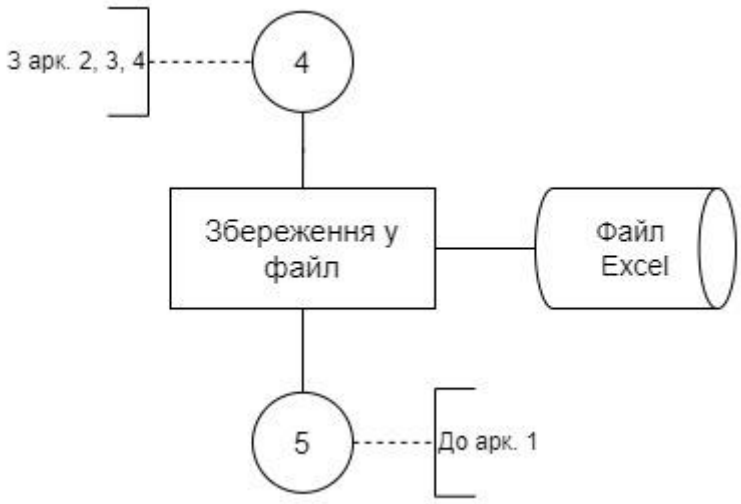

Рисунок 2.5. – Арк. 5

2.3. Моделювання інформаційної підсистеми

Моделювання поведінки системи.

Щоб визначити та описати функціональність інформаційної системи треба створити діаграму прецедентів та для кожного прецедента навести свою діаграму послідовності.

На рис. 2.6 представлено діаграму прецедентів (варіантів використання), а на рисунках 2.7, 2.8, 2.9, 2.10 – діаграми послідовності.

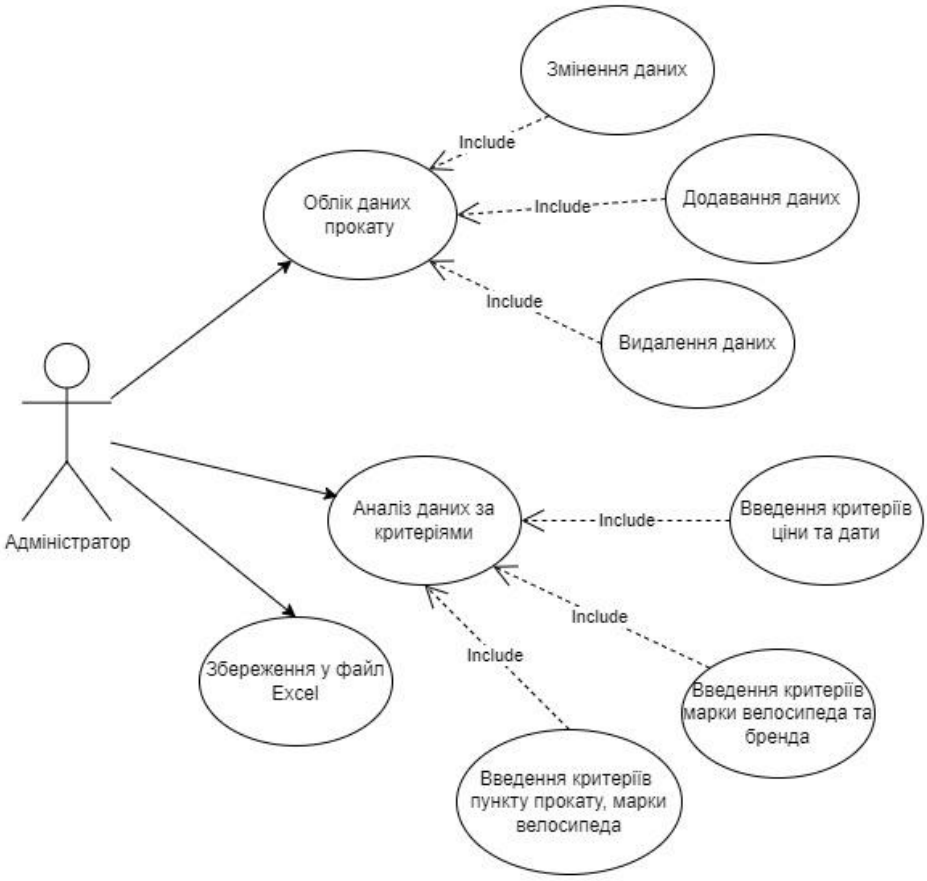

Рисунок 2.6. – Діаграма прецедентів

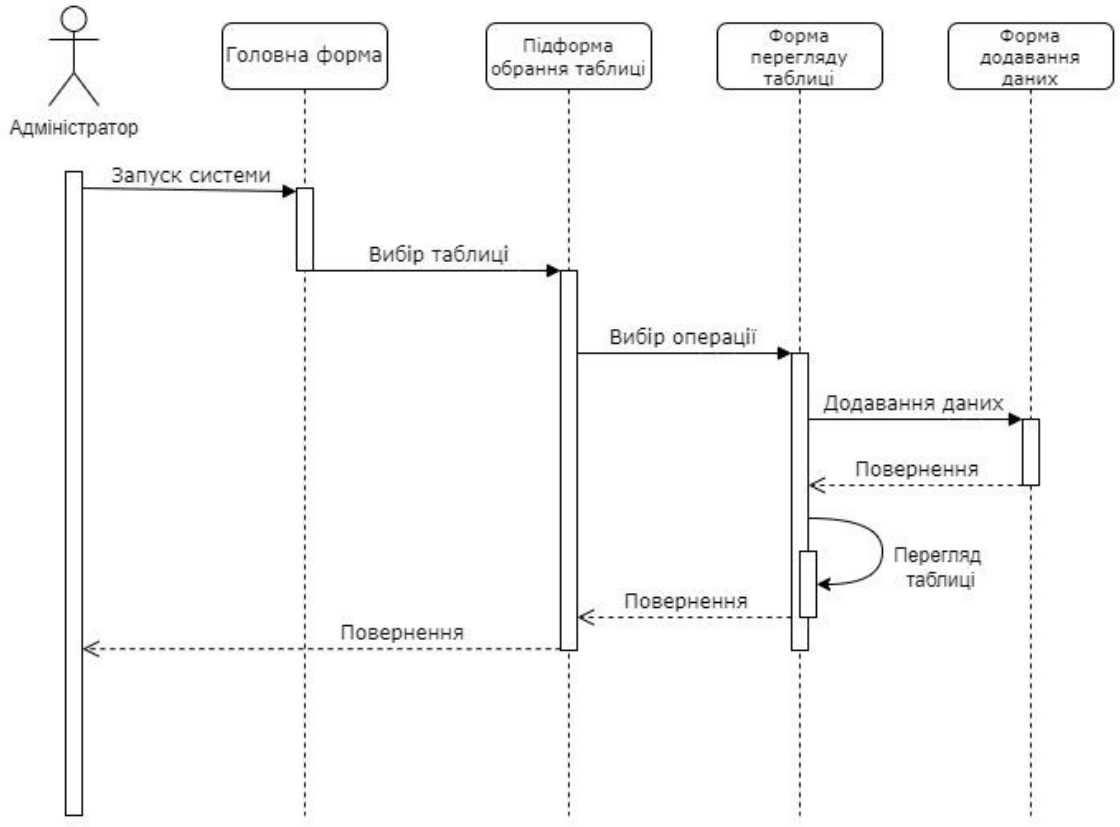

Рисунок 2.7. – Діаграма послідовності для прецеденту «Додавання даних»

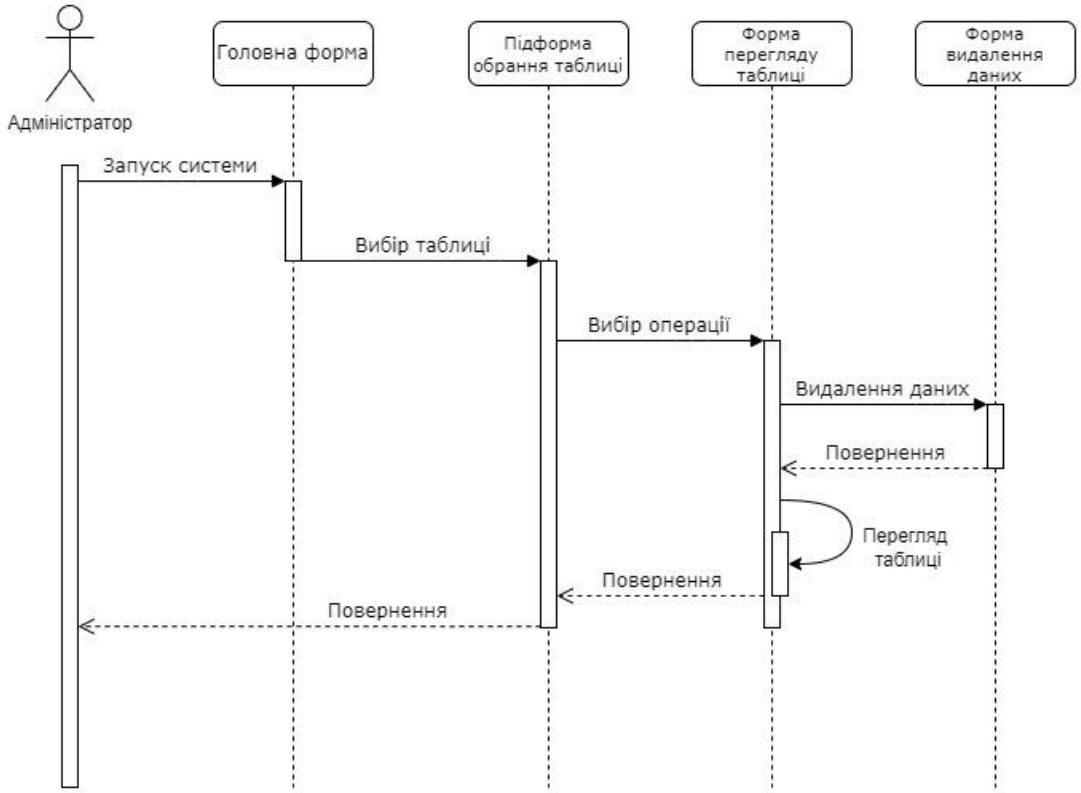

Рисунок 2.8. – Діаграма послідовності для прецеденту «Видалення даних»

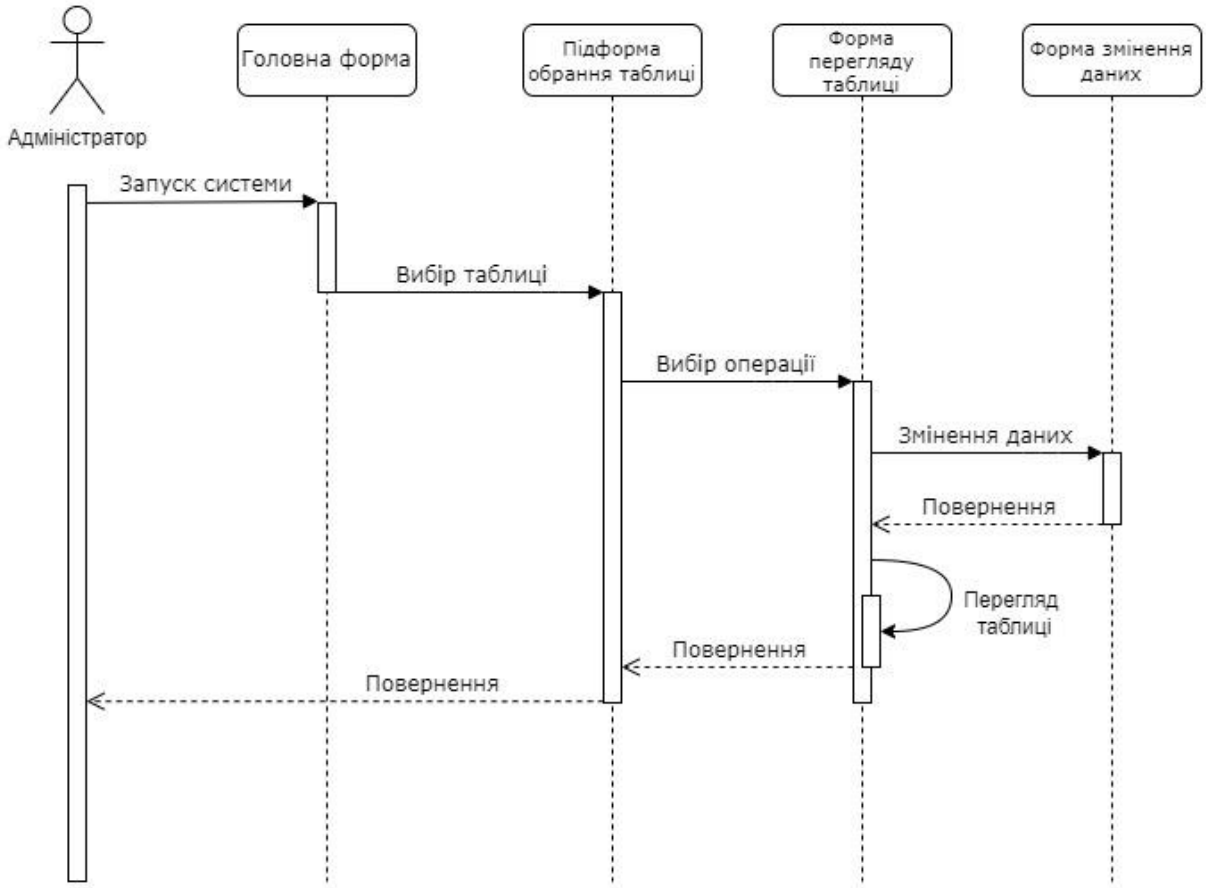

Рисунок 2.9. – Діаграма послідовності для прецеденту «Змінення даних»

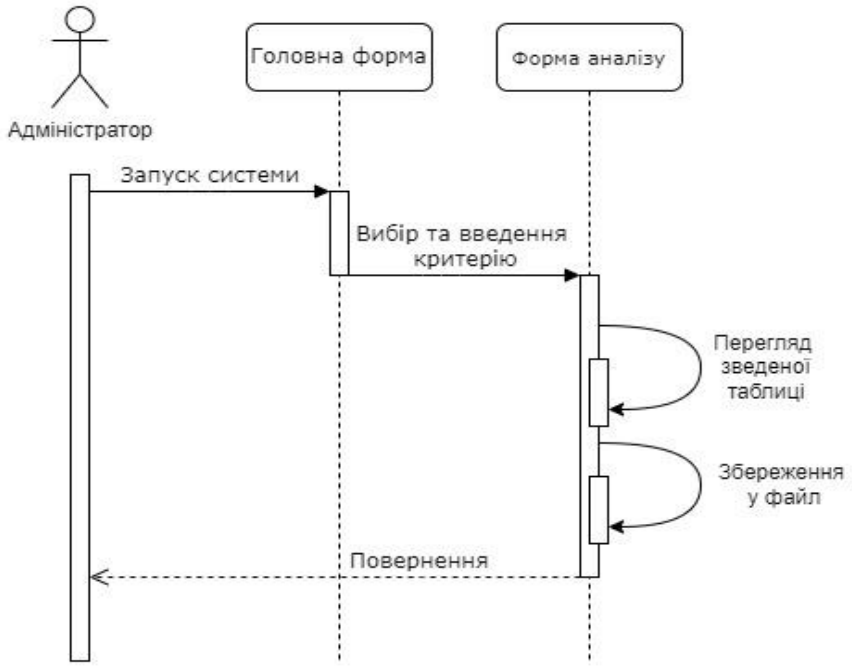

Рисунок 2.10. – Діаграма послідовності для прецедентів «Аналіз за критеріями» та «Збереження у файл Excel»

Щоб графічно зобразити виконаний набор дій системи та розглянути варіацію діаграми стану було створено діаграму діяльності (дій). Діаграма дій представлена на рисунку 2.11.

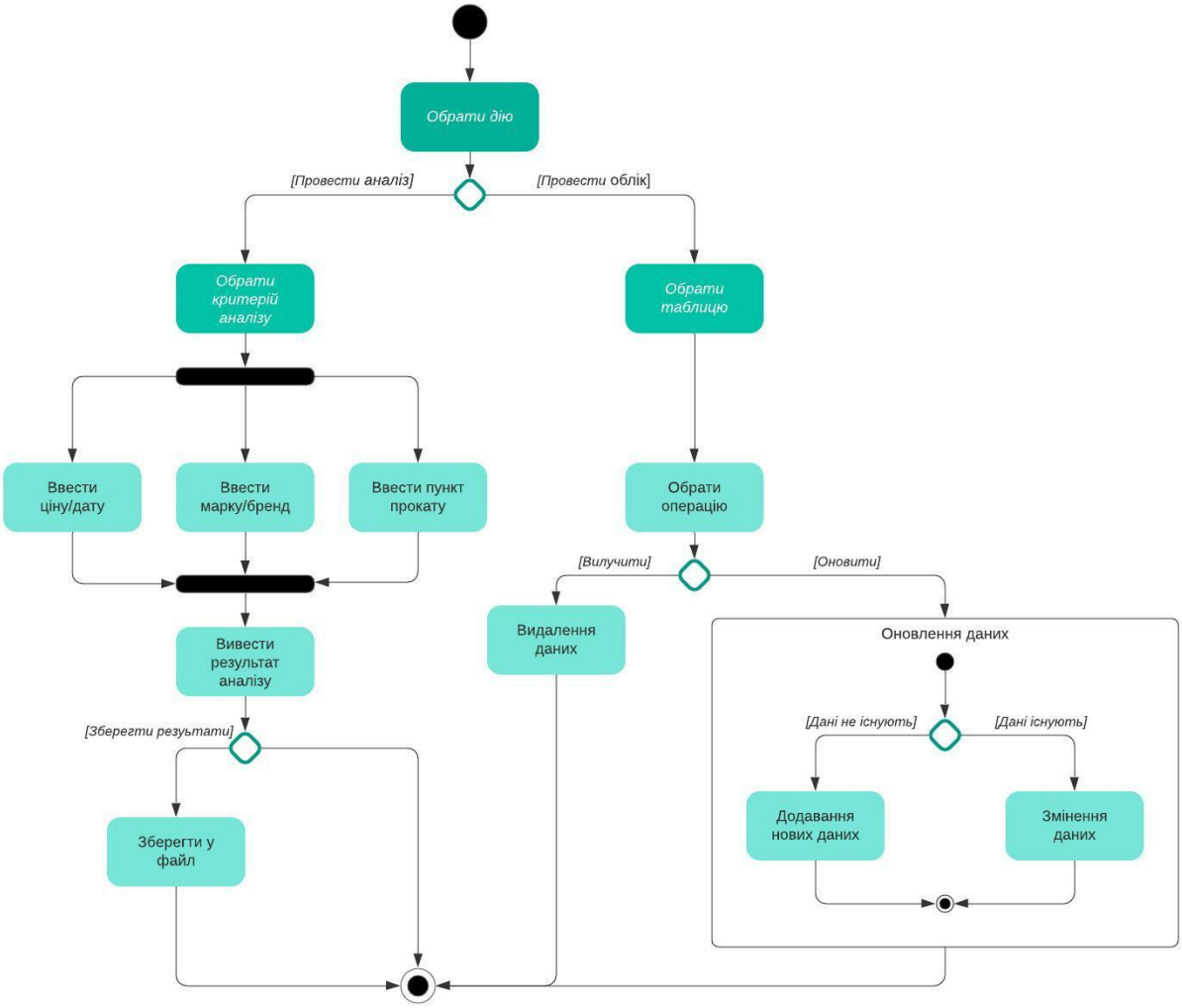

Рисунок 2.11. – Діаграма дій

Моделювання структури системи.

Система буде будуватися на основі об’єктно-орієнтованого підходу – методу програмування, що заснований на поданні програми у вигляді об’єктів, кожен з яких є екземпляром класу.

Основними складовими ООП є:

– Клас – користувацький тип даних, який виступає шаблоном з набором атрибутів та методів.

– Об’єкт – окремий екземпляр класу.

На рисунку 2.12 представлено діаграму класів. Для реалізації логіки системи було розроблено набір з 4 класів, а для візуалізації взаємодії з користувачем 10 класів, які є нащадками класу форма та реалізовують користувацький інтерфейс.

– RentalAccounting – реалізує облік даних: додавання, видалення,

змінення;

– RentalAnalysis – реалізовує аналіз даних за критеріями та їх представлення в результуючій таблиці, а також використовує клас ExcelReport для створення звіту.
– IReport – інтерфейс з методом збереження даних у файл, де ExcelReport – реалізація методу інтерфейсу.

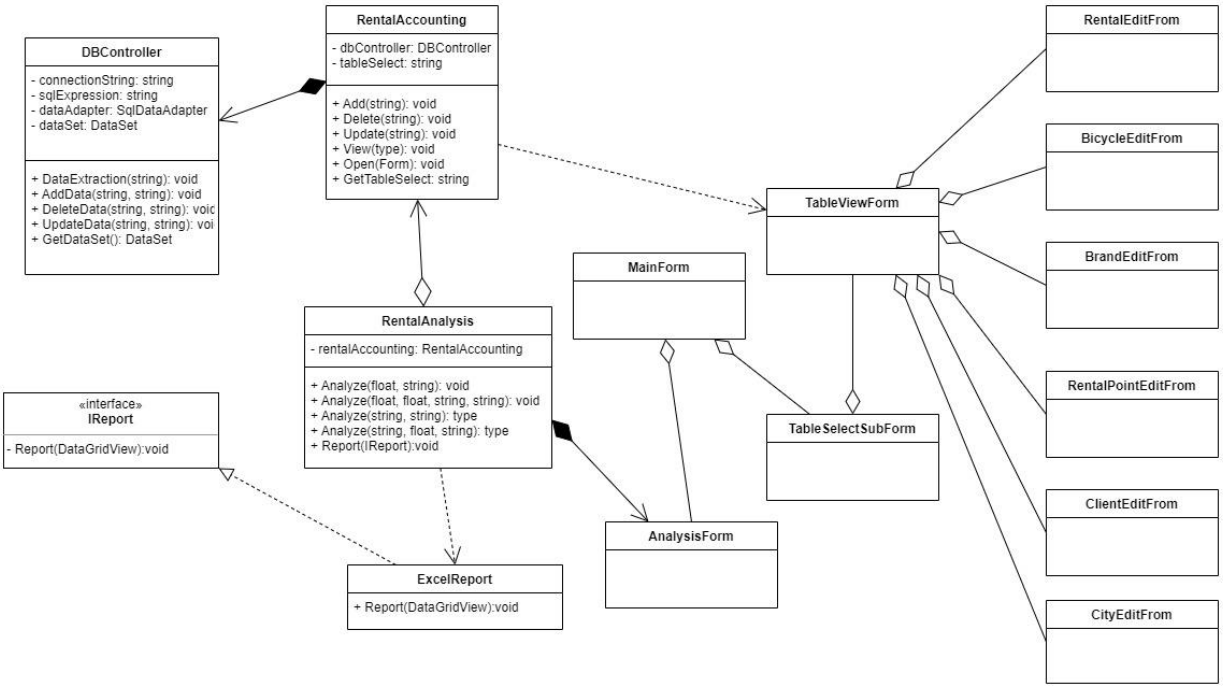

Рисунок 2.12. – Діаграма класів

Розподіл вимог за компонентами системи.

Щоб перевірити правильність виконання кожної функції системи слід провести тестування, розробивши тест-кейси. За допомогою них можна описати сукупність кроків, умов і параметрів, які необхідні для перевірки реалізації функції.

В таблиці 2.5 наведено всі розроблені тест-кейси.

Таблиця 2.5

ТАБЛИЦЯ ТЕСТ-КЕЙСІВ

Дія	Очікуваний результат	Результат тесту
Введення в поле коду існуючого значення		
1. Запустити програму 2. Відкрити форму редагування	Виведення повідомлення "Запис з таким кодом вже існує!"	Пройшов

3. У поле коду ввести цифри 4. Натиснути кнопку "Ок" 5. Вводимо значення "1" (вже існує)		
Введення в поле коду літер		
1. Запустити програму 2. Відкрити форму редагування 3. У поле коду ввести літери 4. Натиснути кнопку "Ок" 5. Вводимо значення "код"	Виведення повідомлення "Поле коду може містити лише цифри!"	Пройшов
Введення в поле коду цифр		
1. Запустити програму 2. Відкрити форму редагування 3. У поле коду ввести цифр 4. Натиснути кнопку "Ок" 5. Вводимо значення "1"	Виведення повідомлення "Запис створений!"	Пройшов
Залишити поле коду пустим		
1. Запустити програму 2. Відкрити форму додавання даних 3. Заповнити усі поля окрім поля коду 4. Натиснути кнопку "Ок"	Виведення повідомлення "Запис створено!"	Пройшов
Залишити усі поля пустими		
1. Запустити програму 2. Відкрити форму редагування 3. Натиснути кнопку "Ок	Виведення повідомлення "Поля повинні бути заповнені!"	Пройшов
Введення в поле бренда велосипеда цифр		
1. Запустити програму 2. Відкрити форму аналізу даних 3. Введення в поле бренда велосипеда цифр 4. Натиснути кнопку "Аналіз" 5. Вводимо "924"	Виведення повідомлення "Поле бренду велосипеда містить лише текстове значення"	Пройшов
Введення в поле марка велосипеда цифр		
1. Запустити програму 2. Відкрити форму аналізу даних 3. Введення в поле марки велосипеда цифр 4. Натиснути кнопку "Аналіз" 5. Вводимо "1343"	Виведення повідомлення "Поле марка велосипеда містить лише текстове значення"	Пройшов
Введення в поле пункту прокату цифр		
1. Запустити програму 2. Відкрити форму аналізу даних 3. Введення в поле пункту прокату цифр 4. Натиснути кнопку "Аналіз" 5. Вводимо "547"	Виведення повідомлення "Поле пункту прокату містить лише текстове значення"	Пройшов
Введення неправильного діапазону ціни		
1. Запустити програму	Виведення повідомлення	Пройшов

2. Відкрити форму аналізу даних 3. Ввести в поле "Від" більше число 4. Ввести в поле "До" менше число 5. Натиснути кнопку "Аналіз" 6. В поле "Від" вводимо 1000, а в поле "До" вводимо 500.	"Неможна вибрати діапазон ціни від більшого до меншого"	
Введення неправильної дати		
1. Запустити програму 2. Відкрити форму аналізу даних 3. Ввести в поле "Від" більшу дату 4. Ввести в поле "До" меншу дату 5. Натиснути кнопку "Аналіз" 6. В поле "Від" вводимо 22.01.2022, а в поле "До" вводимо 20.01.2022	Виведення повідомлення "Некоректне введення діапазону дати"	Пройшов
Залишити поле коду пустим		
1. Запустити програму 2. Відкрити форму видалення даних 3. Заповнити усі поля окрім поля коду 4. Натиснути кнопку "Ок"	Виведення повідомлення "Поле коду не може бути пустим!"	Пройшов
Правильний формат файлу		
1. Запустити програму 2. Відкрити форму аналізу 3. Виконати аналіз 4. Натиснути кнопку "Зберегти у файл Excel" 4. Введення назви файлу у вікні збереження Провідника	Формат запропонованих файлів відображається як ".xls, .xlsx"	Пройшов
Неможливо зберегти результати аналізу без його проведення		
1. Запустити програму 2. Відкрити форму аналізу 3. Натиснути кнопку "Зберегти у файл Excel"	Виведення повідомлення. "Щоб зберегти результати аналізу спочатку треба обрати потрібні критерії та виконати аналіз."	Пройшов

Щоб верифікувати проєктні рішення слід провести трасування описаних вимог до системи та їх співвіднесення з прецедентами, варіантами тестування, елементами користувацького інтерфейсу та зв’язками між ними.

На рисунку 2.13 представлена діаграма трасування вимог до системи, прецедентів, тест-кейсів та елементів інтерфейсу для обліку даних прокату.

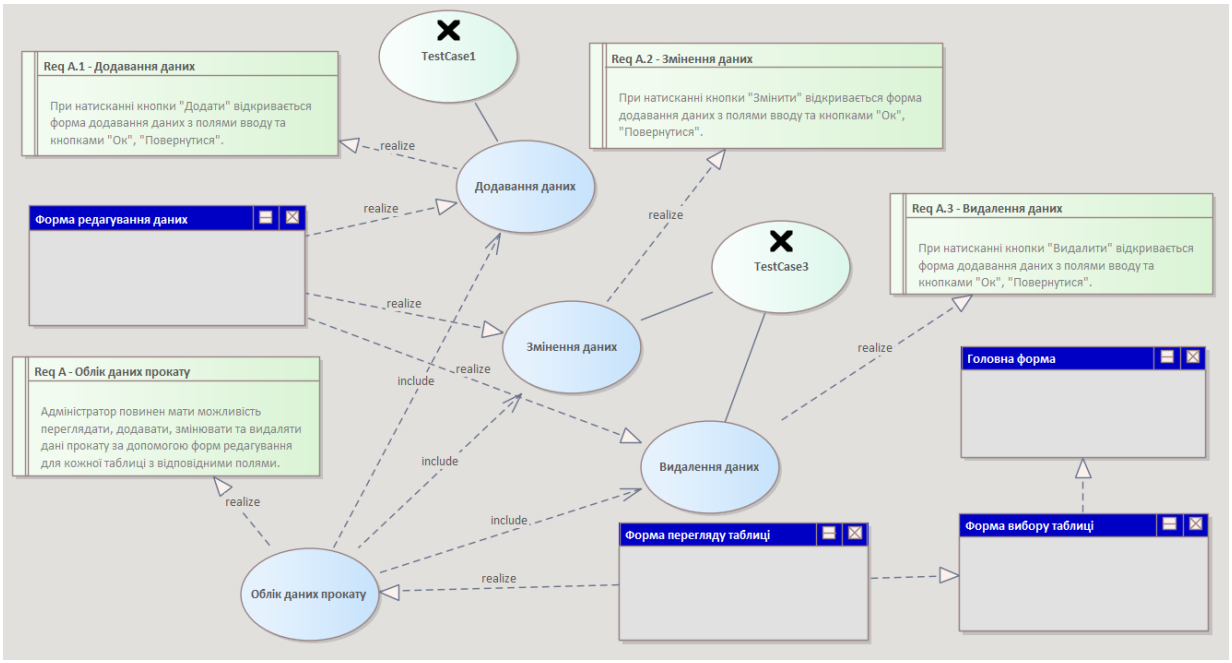

Рисунок 2.13. – Діаграма трасування вимог до системи, прецедентів, тест-кейсів та елементів інтерфейсу. Облік даних прокату.

На рисунку 2.14 представлена діаграма трасування вимог до системи, прецедентів, тест-кейсів та елементів інтерфейсу для аналізу даних прокату.

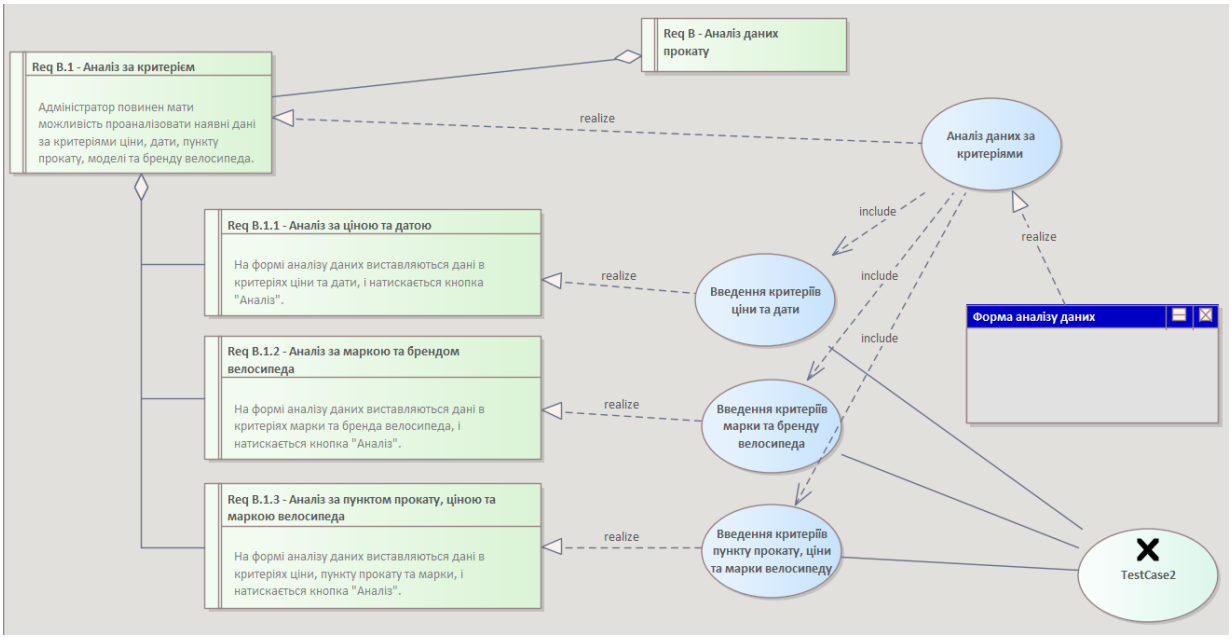

Рисунок 2.14. – Діаграма трасування вимог до системи, прецедентів, тест-кейсів та елементів інтерфейсу. Аналіз даних прокату.

На рисунку 2.15 представлена діаграма трасування вимог до системи, прецедентів, тест-кейсів та елементів інтерфейсу для збереження аналізу у файл.

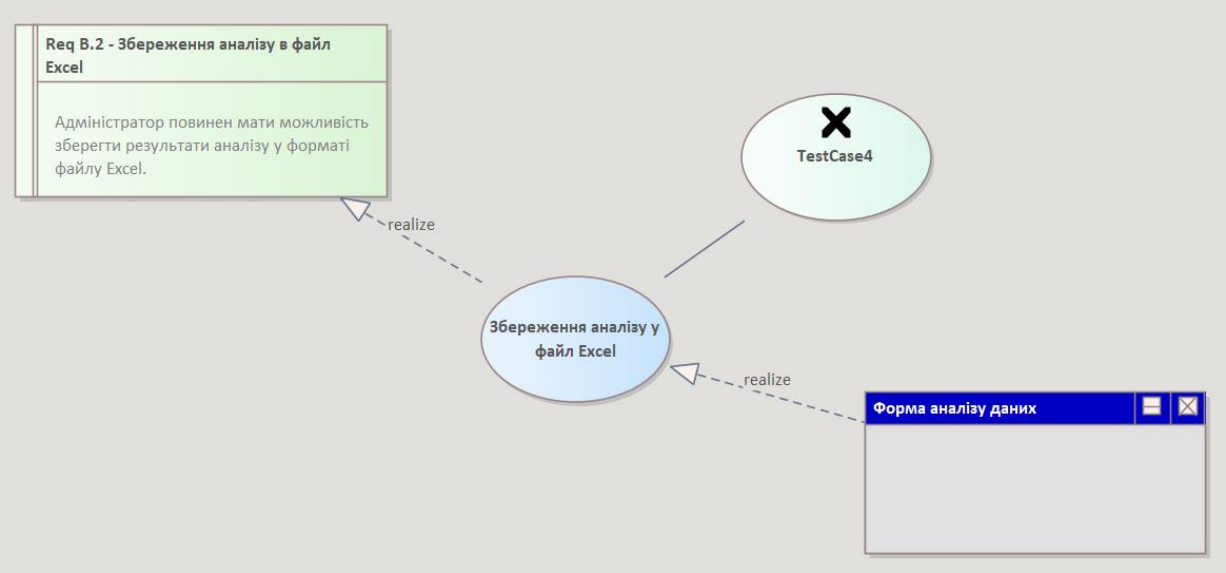

Рисунок 2.15. – Діаграма трасування вимог до системи, прецедентів, тест-кейсів та елементів інтерфейсу. Збереження аналізу у файл.

В таблиці 2.6 представлена матриця взаємозв’язків.

Таблиця 2.6

МАТРИЦЯ ВЗАЄМОЗВ’ЯЗКІВ

Тест-кейс / Вимоги	TestCase1	TestCase2	TestCase3	TestCase4
Req A – Облік даних прокату				
Req A.1 – Додавання даних	X			
Req A.2 – Змінення даних			X	
Req A.3 – Видалення даних			X	
Req B – Аналіз даних прокату				
Req B.1 – Аналіз за критерієм				
Req B.1.1 – Аналіз за ціною та датою		X		
Req B.1.2 – Аналіз за маркою та брендом велосипеда		X		
Req B.1.3 – Аналіз за пунктом прокату, ціною та маркою велосипеда		X		
Req B.2 – Збереження аналізу в файл Excel				X

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ ПІДСИСТЕМИ

3.1. Інформаційне забезпечення

Одним з головних елементів більшості інформаційної системи є системи управління базами даних.

Обираючи СУБД важливо обрати ту базу даних, яка в найбільшій мірі відповідає вимогам до інформаційної системи. У випадку вибору СУБД для системи з обліку та аналізу прокату велосипедів – був проведений порівняльний аналіз серед найвідоміших систем управління базами даних: Oracle, Microsoft SQL Server, MySQL, Microsoft Access.

Oracle - це найбільша фірма-розробник баз даних для Windows NT і UNIX. СУБД Oracle має більше можливостей ніж інші продукти, якщо говорити про масштабність та потужність. Але це породжує непотрібну надлишковість та дороговизну. Тому вона підходить лише для підприємства, які працюють з великими масивами інформації.

MySQL – система управління базами даних, яку часто використовують веб-програмісти, при розробці своїх сайтів. Проте, за своїми можливостям вона значно поступається SQL Server та Oracle.

Microsoft Access – невелика, проста в оволодінні система управління базами даних. Можливостей цієї системи цілком достатньо для розробки, але вона значно ненадійніша за MS SQL Server.

Microsoft SQL Server - це СУБД, яка розроблена компанією Microsoft. В якості основної мови запитів використовується Transact-SQL. Сьогодні вона широко використовується у роботі з базами даних різних розмірів.

Найважливішими характеристиками СУБД Microsoft SQL Server є наступні:

– простота адміністрування;
– можливість підключення до Web;
– швидкодія й функціональні можливості механізму сервера СУБД;
– наявність засобів віддаленого доступу.

Для проєктування БД для системи з обліку та аналізу прокату велосипедів слід зупинитися на Microsoft SQL Server 2019 – як компроміс між функціональністю та доступністю.

Інфологічна модель. Враховуючи поставлену задачу, на основі описаної предметної області було спроєктовано інформаційно-логічну модель (див. рис. 3.8). Вона містить шість об’єктів: Оренда, Клієнт, Велосипед, Бренд, Пункт прокату та Місто.

Опис атрибутів інформаційних об’єктів наведені у таблицях 3.1, 3.2, 3.3, 3.4, 3.5 та 3.6 цього розділу.

Таблиця 3.1

ОПИС АТРИБУТІВ ІНФОРМАЦІЙНОГО ОБ’ЄКТУ «ОРЕНДА».

Назва атрибута	Формат	Іденти-фікатор	Обов’яз-ковий (необов’яз-ковий)	Обмеження на право звертання	Первинний (вторин-ний) ключ	Виво-димість значень	Дублю-вання значень	Індексне поле
Код оренди	9(8)	RentalId	так	Має право адміністратор мережі та відділ статистики	ПК	—	Ні	ІНД
Назва оренди	X(50)	RentalName	так			Так	Так	
Код клієнта	9(8)	ClientId	так		ВК	—	Так	ІДД
Код велосипеда	9(8)	BicyclId	так		ВК	—	Так	ІДД
Дата початку	X(20)	DateFrom	так			Так	Так	
Дата закінчення	X(20)	DateTo	так			Так	Так	
Код пункту прокату	9(8)	DepartId	так		ВК	—	Так	ІДД
Ціна	9(8).	Price	так			Так	Так	

Таблиця 3.2

ОПИС АТРИБУТІВ ІНФОРМАЦІЙНОГО ОБ'ЄКТУ «КЛІЄНТ».

Назва атрибута	Формат	Іденти-фікатор	Обов'яз-ковий (необов'яз-ковий)	Обмеження на право звертання	Первинний (вторин-ний) ключ	Виво-димість значень	Дублю-вання знач-чень	Індексне поле
Код клієнта	9(8)	ClientId	так	Має право адміністратор мережі та відділ статистики	ПК	—	Ні	ІНД
ПІБ клієнта	X(200)	ClientName	так			Так	Так	
Паспорт клієнта	X(50)	Passport	так			—	Ні	
Кредитна карта	X(16)	CreditCard	так			—	Ні	
Код міста	9(8)	CityId	так		ВК	Так	Так	ІДД
Адреса клієнта	X(100)	ClientAddress	так			Так	Так	

Таблиця 3.3

ОПИС АТРИБУТІВ ІНФОРМАЦІЙНОГО ОБ'ЄКТА «ВЕЛОСИПЕД».

Назва атрибута	Формат	Іденти-фікатор	Обов'яз-ковий (необов'яз-ковий)	Обмеження на право звертання	Первинний (вторин-ний) ключ	Виво-димість значень	Дублю-вання значень, %	Індексне поле
Код велосипеда	9(8)	BicycleId	так	Має право адміністратор мережі відділ статистики	ПК	—	Ні	ІНД
Назва велосипеда	X(200)	BicycleName	так			Так	Ні	
Тип велосипеда	X(50)	BicycleType	так			Так	Так	
Код бренда	9(8)	BrandId	так		ВК	—	Так	ІДД
Рік	X(4)	BicycleYear	так			Так	Так	

Таблиця 3.4

ОПИС АТРИБУТІВ ІНФОРМАЦІЙНОГО ОБ'ЄКТА «БРЕНД».

Назва атрибута	Формат	Іденти-фікатор	Обов'яз-ковий (необов'яз-ковий)	Обмеження на право звертання	Первинний (вторин-ний) ключ	Вивo-димість значень	Дублю-вання зна-чень, %	Індексне поле
Код бренда	9(8)	BrandId	так	Має право адміністратор мережі та відділ статистики	ПК	—	Ні	ІНД
Назва бренда	X(50)	BrandName	так			Так	Ні	
Ліцензія	9(8)	License	так			Так	Так	
Код міста	9(8)	CityId	так		ВК	—	Так	ІДД
Адреса бренда	X(20)	BrandAddress	так			Так	Так	

Таблиця 3.5

ОПИС АТРИБУТІВ ІНФОРМАЦІЙНОГО ОБ'ЄКТА «ПУНКТ ПРОКАТУ».

Назва атрибута	Формат	Ідентифікатор	Обов'яз-ковий (необов'яз-ковий)	Обмеження на право звертання	Первинний (вторин-ний) ключ	Вивo-димість значень	Дублю-вання значень, %	Індексне поле
Код пункту прокату	9(8)	DepartId	так	Має право адміністратор мережі та відділ статистики	ПК	—	Ні	ІНД
Назва пункту прокату	X(100)	DepartName	так			Так	Так	
Адреса пункту прокату	X(100)	DepartAddress	так			Так	Так	

Таблиця 3.6

ОПИС АТРИБУТІВ ІНФОРМАЦІЙНОГО ОБ'ЄКТА «МІСТО».

Назва атрибута	Формат	Іденти-фікатор	Обов'яз-ковий (необов'яз-ковий)	Обмеження на право звертання	Первинний (вторин-ний) ключ	Виво-димість значень	Дублю-вання значень, %	Індексне поле
Код міста	9(8)	CityId	так	Має право адміністратор мережі та відділ статистики	ПК	—	Ні	ІНД
Назва міста	X(100)	CityName	так			Так	Так	
Область	X(100)	District	так			Так	Так	
Країна	X(50)	Country	так			Так	Так	

Щоб описати структури об'єктів та їх зв'язки слід розробити ER-діаграму для кожного з них. Вона дає можливість описати ці об'єкти в загальних рисах.

Виходячи з аналізу предметної області можна виділити наступні структурні зв'язки:

– «Клієнт» та «Оренда» зі зв'язком «замовляє»;
– «Оренда» та «Пункт прокату» зі зв'язком «здійснюється у»;
– «Велосипед» та «Оренда» зі зв'язком «входить до»;
– «Бренд» та «Велосипед» зі зв'язком «має»;
– «Бренд» та «Місто» зі зв'язком «знаходиться в»;
– «Клієнт» та «Місто» зі зв'язком «живе в».

Визначимо тип зв'язку між об'єктами «Клієнт» та «Оренда». Один клієнт може орендувати декілька велосипедів, але один орендований велосипед не може мати декількох клієнтів. Тому, тип зв'язку між об'єктами однин до багатьох (1: Б).

Складемо ER-діаграму для об'єктів «Клієнт» та «Оренда» зі зв'язком «замовляє» (рис. 3.1).

Рисунок 3.1. – ER-діаграма для об’єктів «Клієнт» та «Оренда» зі зв’язком «замовляє»

Визначимо тип зв'язку між об’єктами «Оренда» та «Пункт прокату». В одному пункті прокату можна здійснювати декілька оренд велосипеду, але одна и таж сама оренда не може здійснюватися у декількох пунктах прокату. Тому, тип зв'язку між об’єктами багато до одного (Б: 1).

Складемо ER-діаграму для об’єктів «Оренда» та «Пункт прокату» зі зв’язком «здійснюється у» (рис. 3.2).

Рисунок 3.2. – ER-діаграма для об’єктів «Оренда» та «Пункт прокату» зі зв’язком «здійснюється у»

Визначимо тип зв'язку між об’єктами «Велосипед» та «Оренда». До оренди входить лише один велосипед, але можна здійснити декілька оренд цього ж велосипеду. Тому, тип зв'язку між об’єктами однин до багатьох (1: Б).

Складемо ER-діаграму для об’єктів «Велосипед» та «Оренда» зі зв’язком «входить до» (рис. 3.3).

Рисунок 3.3. – ER-діаграма для об’єктів «Велосипед» та «Оренда» зі зв’язком «входить до»

Визначимо тип зв'язку між об’єктами «Бренд» та «Велосипед». Бренд моє декілька велосипедів, а велосипед може відноситися тільки до одного бренду. Тому, тип зв'язку між об’єктами однин до багатьох (1: Б).

Складемо ER-діаграму для об’єктів «Бренд» та «Велосипед» зі зв’язком «має» (рис. 3.4).

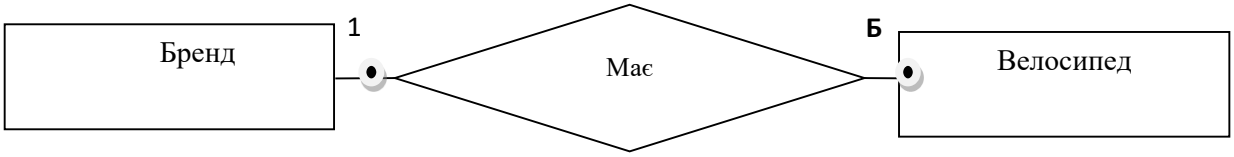

Рисунок 3.4. – ER-діаграма для об’єктів «Велосипед» та «Оренда» зі зв’язком «входить до»

Визначимо тип зв'язку між об’єктами «Бренд» та «Місто». В одному місті можуть знаходитися декілька брендів. Тому, тип зв'язку між об’єктами однин до багатьох (1: Б).

Складемо ER-діаграму для об’єктів «Бренд» та «Місто» зі зв’язком «знаходиться в» (рис. 3.5).

Рисунок 3.5. – ER-діаграма для об’єктів «Бренд» та «Місто» зі зв’язком «знаходиться в»

Визначимо тип зв'язку між об’єктами «Клієнт» та «Місто». В одному місті можуть жити декілька клієнтів. Тому, тип зв'язку між об’єктами однин до багатьох (Б: 1).

Складемо ER-діаграму для об’єктів «Клієнт» та «Місто» зі зв’язком «живе в» (рис. 3.6).

Рисунок 3.6. – ER-діаграма для об’єктів «Клієнт» та «Місто» зі зв’язком «живе в»

Для предметної області «Проєктування інформаційної системи з обліку та аналізу прокату велосипедів» було розроблено загальну ER-діаграму предметної області (див. рис. 3.7).

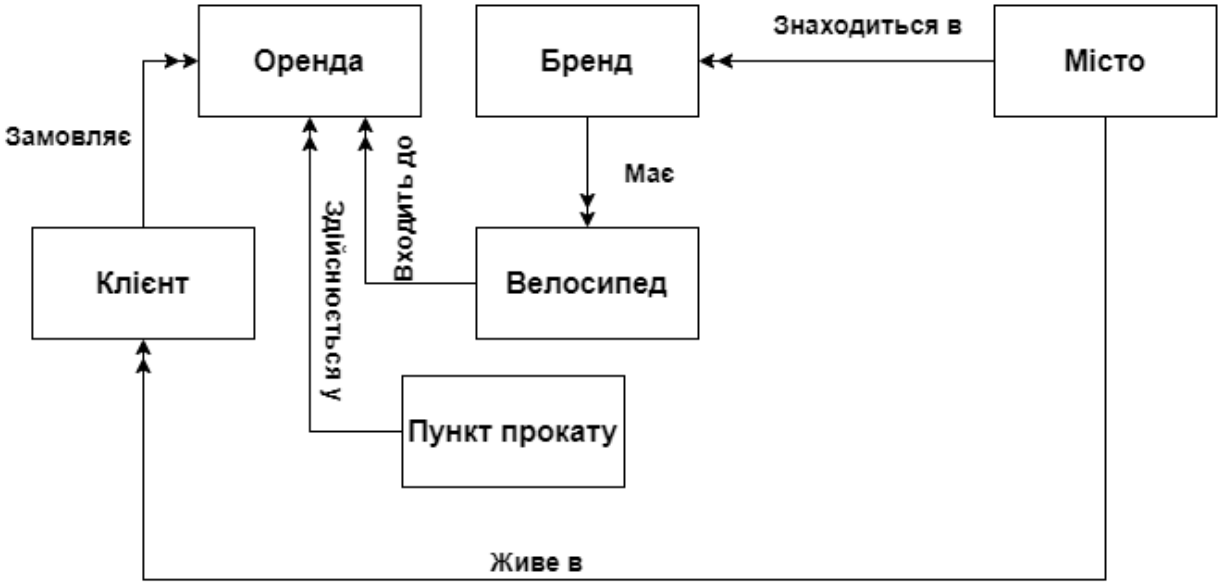

Рисунок 3.7 – Загальна ER-діаграма предметної області.

Після визначення структурних зав’язків та їх графічного представлення на ER-діаграмах, можна представити вигляд інформаційно-логічної моделі (див. рис. 3.8).

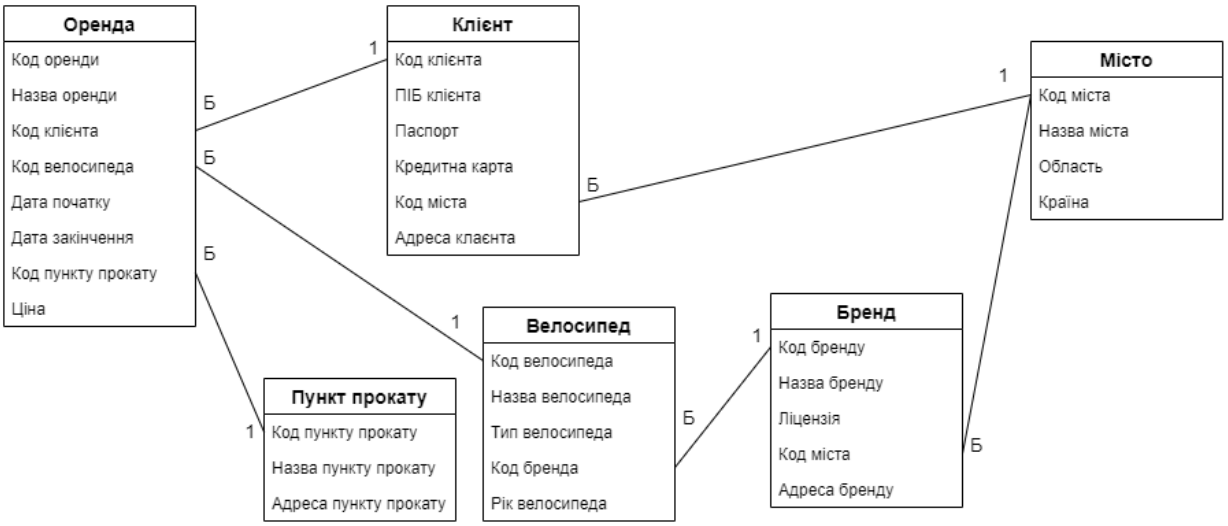

Рисунок 3.8. – Інформаційно-логічна модель предметної області

Даталогічна модель. Даталогічне проєктування зводиться до подання інфологічної моделі в термінах обраної системи управління базами даних.

Реляційна модель представлення даних у БД зображена на рис. 3.9.

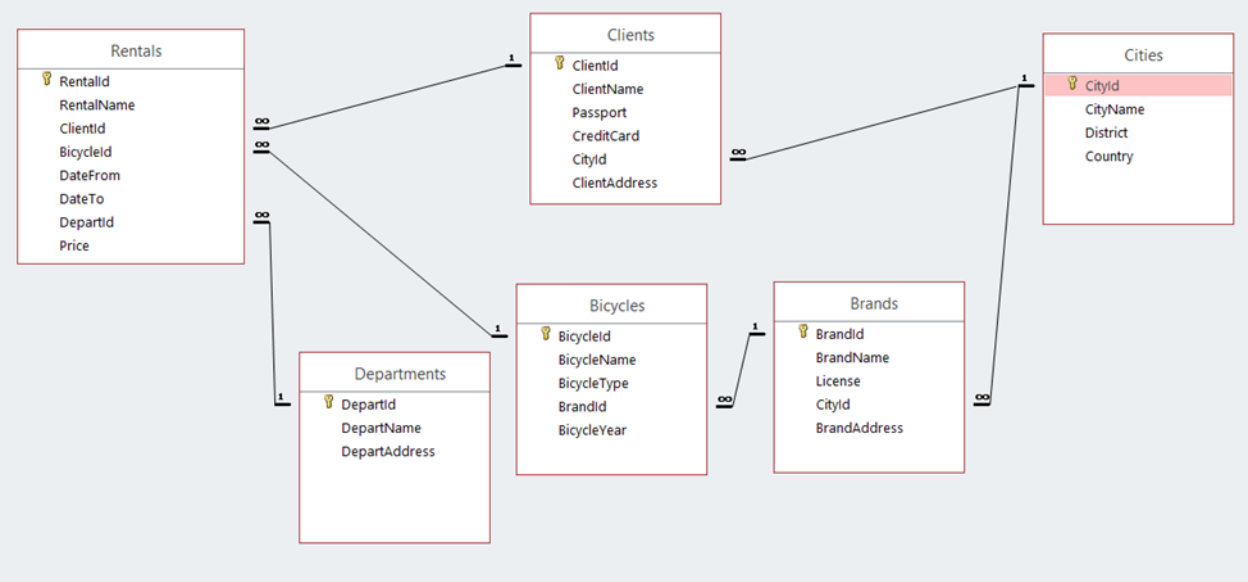

Рисунок 3.9. – Реляційна модель бази даних ІС з обліку та аналізу прокату велосипедів

На рисунку 3.10 представлено загальну схему інформаційного забезпечення. Класифікаторами і кодами, які використовуються у системі є коди оренди, клієнта, велосипеда, бренда, пункту прокату та міста.

У додатку А представлено тестовий приклад таблиць БД.

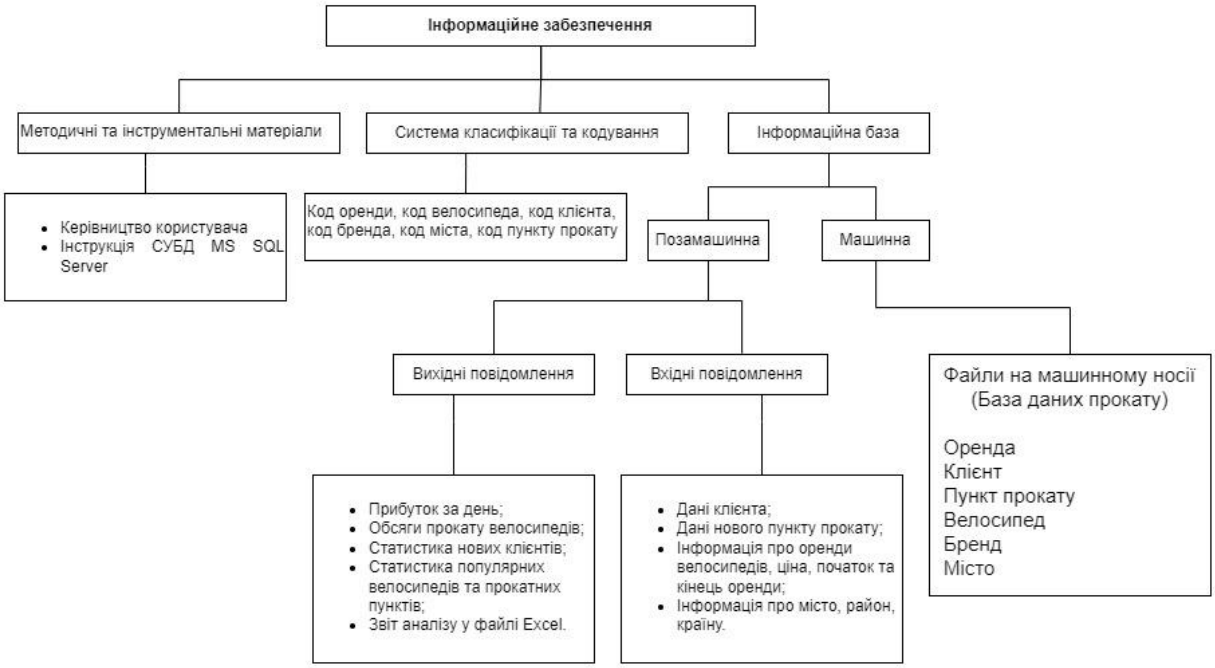

Рисунок 3.10. – Схема інформаційного забезпечення

3.2. Технічне забезпечення

Щоб отримати повне фізичне представлення інформаційної системи потрібна інформація про те, на якій платформі і на яких обчислювальних

засобах, призначених для збору, зберігання, нагромадження, обробки, передачі та виведення інформації вона реалізована. У цьому може допомогти розробка схема автоматизації (див. рис. 3.11) та визначення комплексу технічних засобів (діаграма розгортання).

До технічних засобів введення інформації для системи використовуються: пристрій введення – клавіатура та маніпулятор – миша. До засобів обробки, які використовуються системою відносяться: персональні ЕОМ та сервер. Інформація виводиться за допомогою технічних засобів виведення інформації – монітори.

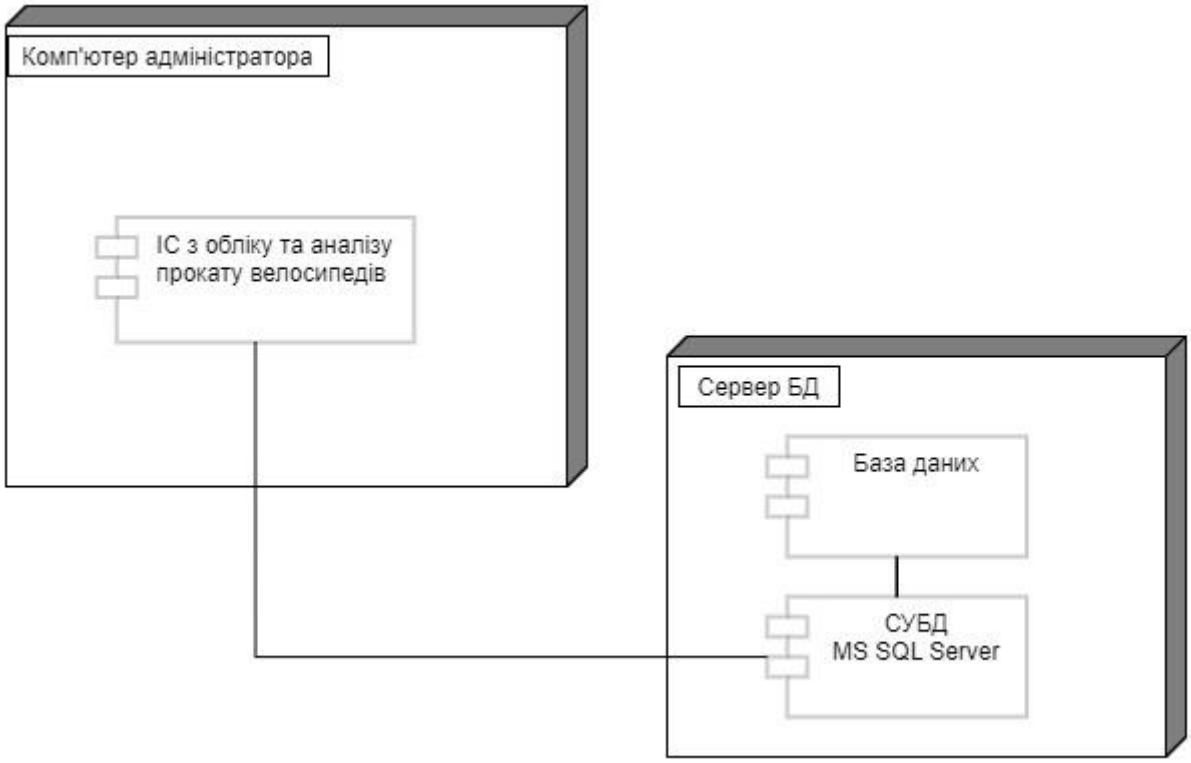

Рисунок 3.11. – Загальна схема автоматизації (Діаграма розгортання КТЗ)

3.3. Програмне забезпечення

Інформаційна система повинна запускатися на 64-розрядній операційній системі Microsoft Windows 10. Для оптимальної роботи системи повинна бути встановлена остання версія Microsoft Visual C++ Redistributable Package Hybrid, а також остання версія .NET.

Щоб відобразити залежності між програмними компонентами, що виникають на етапі компіляції або в процесі виконання програми, зокрема

зв'язок файлів вихідного коду з динамічними бібліотеками, формами було створено діаграму компонентів (див. рис. 3.12).

У Додатку В представлено тексти програми.

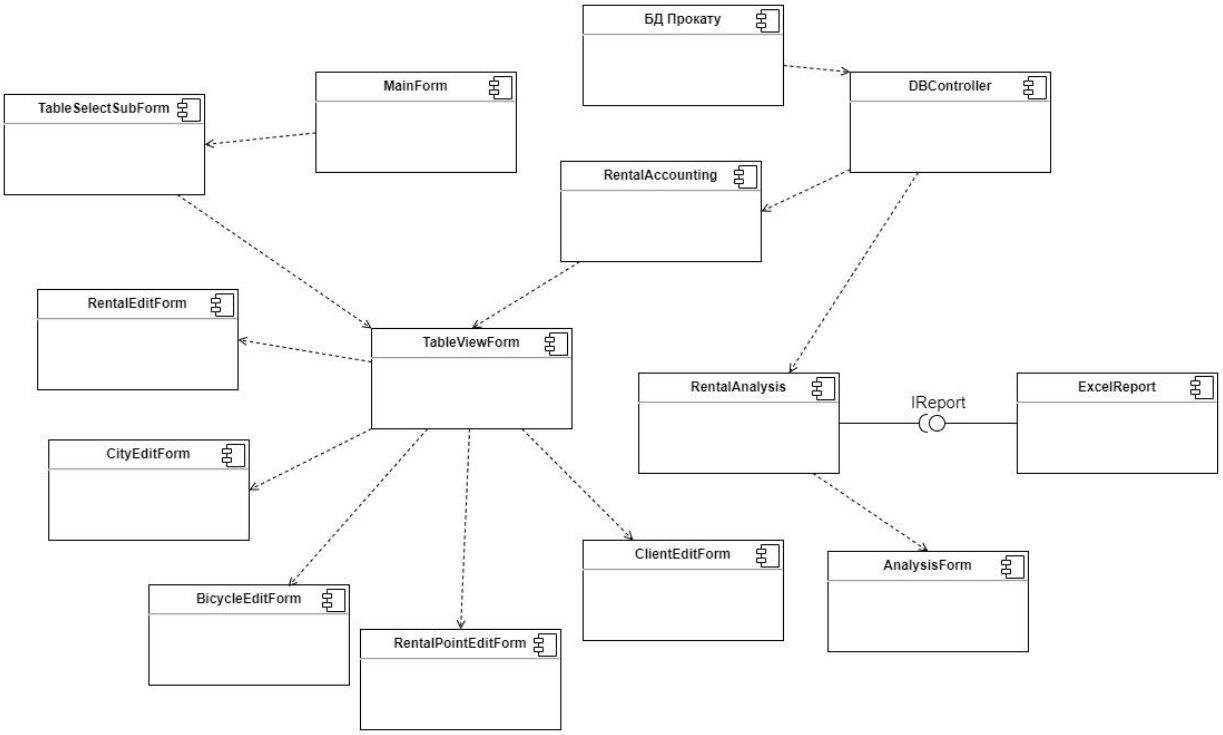

Рисунок 3.12. Діаграма компонентів

3.4. Результати реалізації інформаційної підсистеми

Після реалізації системи при використанні обраних методології та технологій слід викласти результати роботи системи так, як якби вона була впроваджена. Усі екранні форми, які ілюструють результати роботи системи представлено в додатку Б.

Екранна форма головної сторінки з можливістю перейти до довідників, де можна переглянути, додати, змінити та видалити інформацію, та аналізу, де можна провести аналіз даних за критеріями представлено на рисунку Б.1 додатку Б. На рисунку Б.2 додатку Б представлена екранна форма вибору потрібної таблиці. На рисунку Б.3 додатку Б представлена екранна форма, на якій можна переглянути таблиці та перейти до виконання операцій додавання, змінення та видалення. На рисунках Б.4, Б.5, Б.6 додатку Б представлені екранні форми редагування даних для функцій додавання, змінення та видалення даних.

На рисунку Б.7 додатку Б представлена екранна форма аналізу даних. Тут є різні категорії аналізу: за датою початку та кінця оренди, за ціною оренди, за маркою велосипеду, за брендом велосипеду, за пунктом прокату, також тут можна розрахувати дохід за певну дати та переглянути, які велосипеди є найпопулярнішими.

Якщо записати ціну тільки в одне з полів, то вибірка буде відповідною: від зазначеної ціни або до зазначеної ціни. Вибірка виконується за декількома таблицями у відповідності від заповнених полів. Після проведення аналізу, результати можна зберегти у Excel файл. На рисунку Б.8 та Б.9 представлено виконання аналізу та збереження у файл.

ВИСНОВОК

Метою кваліфікаційного бакалаврського проєкту було проєктування та реалізація інформаційної системи з обліку та аналізу прокату велосипедів за допомогою програмних засобів Microsoft Visual Studio та технології Windows Forms .Net Framework.

Було проведено аналіз предметної області. Досліджено процеси обліку та аналізу роботи мережі прокату велосипедів. Проведено аналіз існуючих підходів до автоматизації задачі. В процесі проведення було виявлено: особливості та специфіку предметної області, визначені основні терміни і поняття, суб'єкти та об'єкти, способи взаємодії суб'єктів, способи використання об'єктів, закономірності; визначені бізнес-вимоги, функціональні та нефункціональні вимоги до розроблюваної системи; розглянуто історію створення та впровадження систем спільного використання велосипедів, їх будову, принцип роботи; розглянуто існуючі аналоги систем обліку та аналізу, визначено їх переваги та недоліки.

Розроблена концепція інформаційної системи, обґрунтовані програмні засоби, технології та методології. Розглянуто варіанти для побудови та управління базами даних. Обрано найоптимальніший варіант, який забезпечує компроміс між функціональністю та доступністю. Розроблено прототип системи. Проведено моделювання системи з використанням інструментарію побудови UML-схем. Для визначення та опису функціональності інформаційної системи було створено діаграму варіантів використання. До кожного прецеденту була представлена своя діаграма послідовності.

На етапі проєктування враховувались не тільки особливості предметної області, а й такі принципи об'єктно-орієнтованого підходу як абстракція, інкапсуляція, поліморфізм. Визначено призначення, техніко-економічна сутність задачі. Наведено перелік об’єктів, за управління якими розв’язується задача. Проведено верифікацію проєктних рішень шляхом розроблення діаграми трасування для вимог, прецедентів, варіантів тестування та відповідних зв’язків мж ними. Описано призначення й

використання вхідної інформації, періодичність розв’язання і термін видачі вихідної інформації. Побудована інформаційна модель задачі. Були проаналізовані можливості різних компонентів графічних форм, реалізація яких можлива завдяки використанню технології WindowsForms .Net Framework. Результати проєктування були оформлені у вигляді діаграми класів, станів, дій, компонентів та розгортання.

Розроблено програмне забезпечення для реалізації системи. Запроєктовано базу даних, визначено всі вимоги до технічного забезпечення та його конфігурації. Результати реалізації проілюстровано у додатках.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Recommended Reading and Links on Public Bicycle Schemes / Kodukula, Santhosh // gtz, 2010., 13 ст.
2. A guide to hire bikes and public bike share schemes – [Електронний ресурс] / We are cycling UK // https://www.cyclinguk.org/article/guide-hire-bikes-and-public-bike-share-schemes
3. Bike-sharing: History, Impacts, Models of Provision, and Future / Paul DeMaio // Journal of Public Transportation, 2009 р., 16 ст.
4. EasyPro – [Електронний ресурс] / EasySoftware // https://easysoftware.pro/projects/programma-ucheta-prokata/
5. РемОнлайн – [Електронний ресурс] / РемОнлайн // https://remonline.ua/hire-and-rental/
6. Торгсофт – [Електронний ресурс] / Торгсофт // https://torgsoft.ua/about/aboutus/
7. Private Bike-share Services Gain Traction – [Електронний ресурс] / naiop // https://www.naiop.org/en/Research-and-Publications/Magazine/2015/Winter-2015-2016/Business-Trends/Private-Bike-share-Services-Gain-Traction
8. Уніфікована мова моделювання UML – [Електронний ресурс] / Портал знань // http://www.znannya.org/?view=uml
9. Мова програмування C# і .NET – [Електронний ресурс] / METANIT.COM // https://metanit.com/sharp/general.php
10. Повний посібник з мови програмування C# 8.0 і .NET Core 3 – [Електронний ресурс] / METANIT.COM // https://metanit.com/sharp/tutorial/
11. Керівництво по програмуванню в Windows Forms – [Електронний ресурс] / METANIT.COM // https://metanit.com/sharp/windowsforms/
12. Введення в Windows Forms – пишемо першу програму – [Електронний ресурс] / Code-Live.ru // https://code-live.ru/post/first-windows-form/
13. Обзор Windows Forms – [Електронний ресурс] / Microsoft Build // https://docs.microsoft.com/ru-ru/dotnet/framework/winforms/windows-forms-overview
14. Документація по C# – [Електронний ресурс] / Microsoft Build // https://docs.microsoft.com/ru-ru/dotnet/csharp/

15. Підручники по C# – [Електронний ресурс] / Microsoft Build // https://docs.microsoft.com/ru-ru/dotnet/csharp/tutorials/
16. C# для професіоналів: тонкощі програмування, 4 видання / Джон Скіт // Видавництво Діалектика, 2020 р., 600 ст.
17. C# 8.0. Кишеньковий довідник / Джозеф Албахари, Бен Албахари // Видавництво Диалектика - Киев, 2020 р., 240 ст.
18. C# 4.0 Повне керівництво / Герберт Шилдт // И.Д. Вильямс,2011р.,1056 ст.
19. Мова програмування C # 7 і платформи .NET і .NET Core / Ендрю Троелсен // Диалектика-Вільямс, 2019р.,1328ст.
20. Проєктування баз і сховищ даних: Навч. посібник./ Н.В. Ситник // — К.:КНЕУ, 2004. —348 с.
21. .NET Desktop Guide for Windows Forms / Microsoft // https://docs.microsoft.com/en-us/dotnet/desktop/winforms/?view=netframeworkdesktop-4.8&preserve-view=true
22. Additions to Windows Forms for the .NET Framework 2.0 / Microsoft // https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2010/35f2fe4h(v=vs.100)
23. C# Windows Forms Application Tutorial with Example / Guru99 // https://www.guru99.com/c-sharp-windows-forms-application.html
24. Чому SOLID — важлива складова мислення програміста. Розбираємося на прикладах з кодом / Dou // https://dou.ua/lenta/articles/solid-principles/
25. Проєктування інформаційних систем. Навчальний посібник / Литвин В.В. , Пасічник В.В. , Шаховська Н.Б. // Магнолія, 2006 - 380 с.
26. C# для професіоналів: тонкощі програмування, 3 видання / Джон Скіт // Діалектика-Вільямс, 2019 – 608 с.
27. C# 7.0. Довідник. Повний опис мови / Джозеф Албахарі, Бен Албахарі // Діалектика-Вільямс, 2018 – 1024 с.
28. Принципи, патерни та методики гнучкої розробки мовою C# / Роберт Мартін, Міка Мартін // Символ-Плюс, 2017 – 757 с.

ДОДАТКИ

Додаток А

Тестовий приклад таблиць БД

	BicycleId	BicycleName	BicycleType	BrandID	BicycleYear
1	1	Tempus Mountcycle	Mountain	1	2021
2	2	Tempus Citycycle	City	1	2021
3	3	Velocraft Runda	City	2	2021
4	4	New Light	City	3	2021
5	5	Night	City	1	2021
6	6	Neuro	Mountain	2	2021

Рисунок А.1 – Таблиця «Велосипед»

	RentalId	RentalName	ClientId	BicycleId	DateFrom	DateTo	DepartId	Price
1	1	rent #32423	1	1	2021-04-06 13:35:00	2021-04-06 14:00:00	1	30
2	2	rent #24432	2	3	2021-04-06 17:13:00	2021-04-06 17:47:00	3	39
3	3	rent #83243	3	1	2021-04-07 12:45:00	2021-04-07 13:10:00	5	30
4	4	rent #87765	4	2	2021-04-08 11:00:00	2021-04-08 11:30:00	2	35
5	5	rent #32423	1	1	2021-04-09 09:36:00	2021-04-09 10:00:00	5	29
6	6	rent #24432	2	3	2021-04-08 15:23:00	2021-04-08 15:55:00	2	37
7	7	rent #83243	3	1	2021-04-07 20:05:00	2021-04-07 20:32:00	4	32
8	8	rent #87765	4	2	2021-04-09 14:10:00	2021-04-09 14:31:00	3	26
9	9	rent #32423	1	1	2021-04-05 07:30:00	2021-04-05 08:35:00	1	70
10	10	rent #24432	2	3	2021-04-05 10:13:00	2021-04-05 10:46:00	4	38
11	11	rent #83243	3	1	2021-04-10 07:34:00	2021-04-10 08:03:00	5	34
12	12	rent #87765	4	2	2021-04-09 17:23:00	2021-04-09 17:59:00	3	41

Рисунок А.2 – Таблиця «Оренда»

	ClientId	ClientName	Passport	CreditCard	CityID	ClientAddress
1	1	Antonov M.V.	#2343245	2345123498766789	1	st. Boryspilska, 10
2	2	Hlebov B.N.	#9324253	2345123498766789	1	st. Trostianetska, 20
3	3	Baranenko A.O.	#923425	2345123498766789	2	st. Torska, 9
4	4	Ulkenko T.A.	#3404523	2345123498766789	3	st. Morska, 5
5	9	Haharin M.V.	#352342	6345526497768789	1	st. Bavarska, 10
6	10	Dronov B.N.	#5463343	3453124987236784	1	st. Alpinska, 20
7	11	Marmalenko A.O.	#645234	4434512349876678	2	st. Karlushevska, 9
8	12	Murko T.A.	#0238423	8345123498766789	3	st. Myrhorodska, 5

Рисунок А.3 – Таблиця «Клієнт»

	DepartId	DepartName	DepartAddress
1	1	Depart1	st. Boryspilska, 10
2	2	Depart2	st. Nemska, 30
3	3	Depart3	st. Debrova, 20
4	4	Depart4	st. Tumenska, 10
5	5	Depart5	st. Alchevska, 5

Рисунок А.4 – Таблиця «Пункт прокату»

	BrandId	BrandName	License	CityID	BrandAddress
1	1	Tempus	#43545665	1	st. Vadyma Hetmana, 35
2	2	Velocraft	#93724254	2	st. Tretia, 8
3	3	Bicyclight	#63245224	3	st. Gutskoho, 12

Рисунок А.5 – Таблиця «Бренд»

	CityId	CityName	District	Country
1	1	Kyiv	Kyivska	Ukraine
2	2	Lviv	Lvivska	Ukraine
3	3	Bila Tserkva	Kyivska	Ukraine
4	4	Kharkiv	Kharkivska	Ukraine
5	5	Dnipro	Dnipropetrovska	Ukraine

Рисунок А.6 – Таблиця «Місто»

Додаток Б

Ілюстрації результатів реалізації

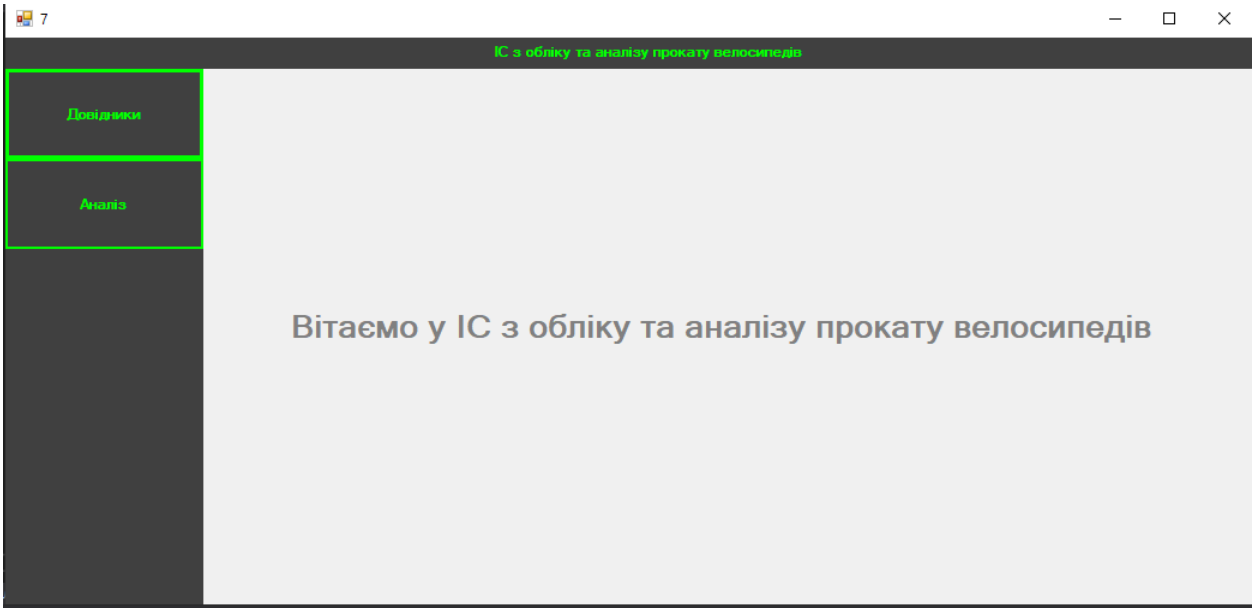

Рисунок Б.1 – Екранна форма головної сторінки

Рисунок Б.2 – Екранна форма вибору таблиці

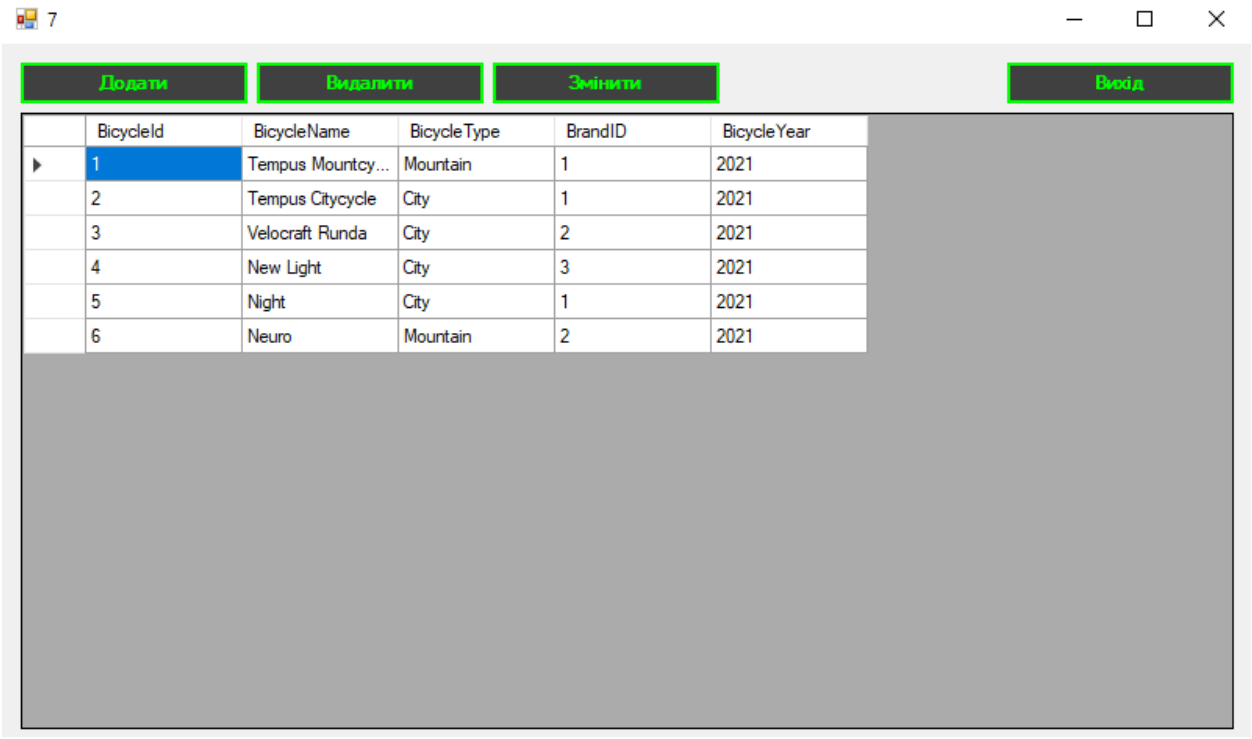

Рисунок Б.3 – Екранна форма перегляду обраної таблиці (Велосипед)

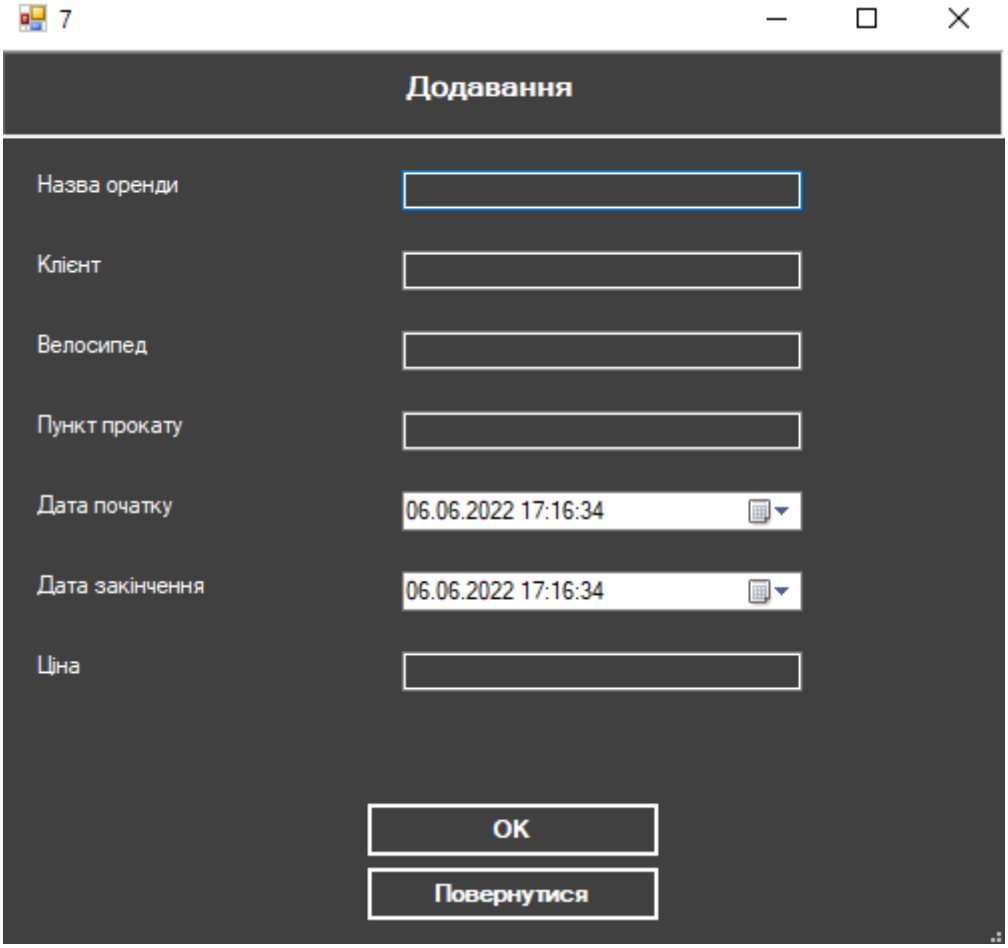

Рисунок Б.4 – Екранна форма редагування таблиці. Додавання даних.

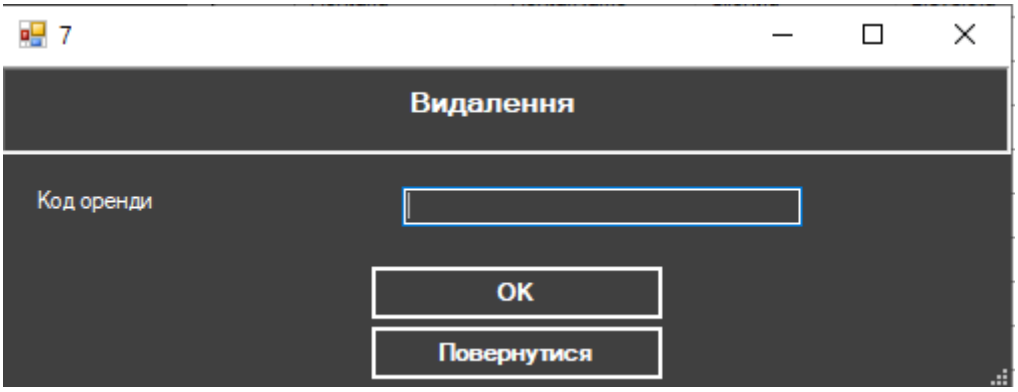

Рисунок Б.5 – Екранна форма редагування таблиці. Видалення даних.

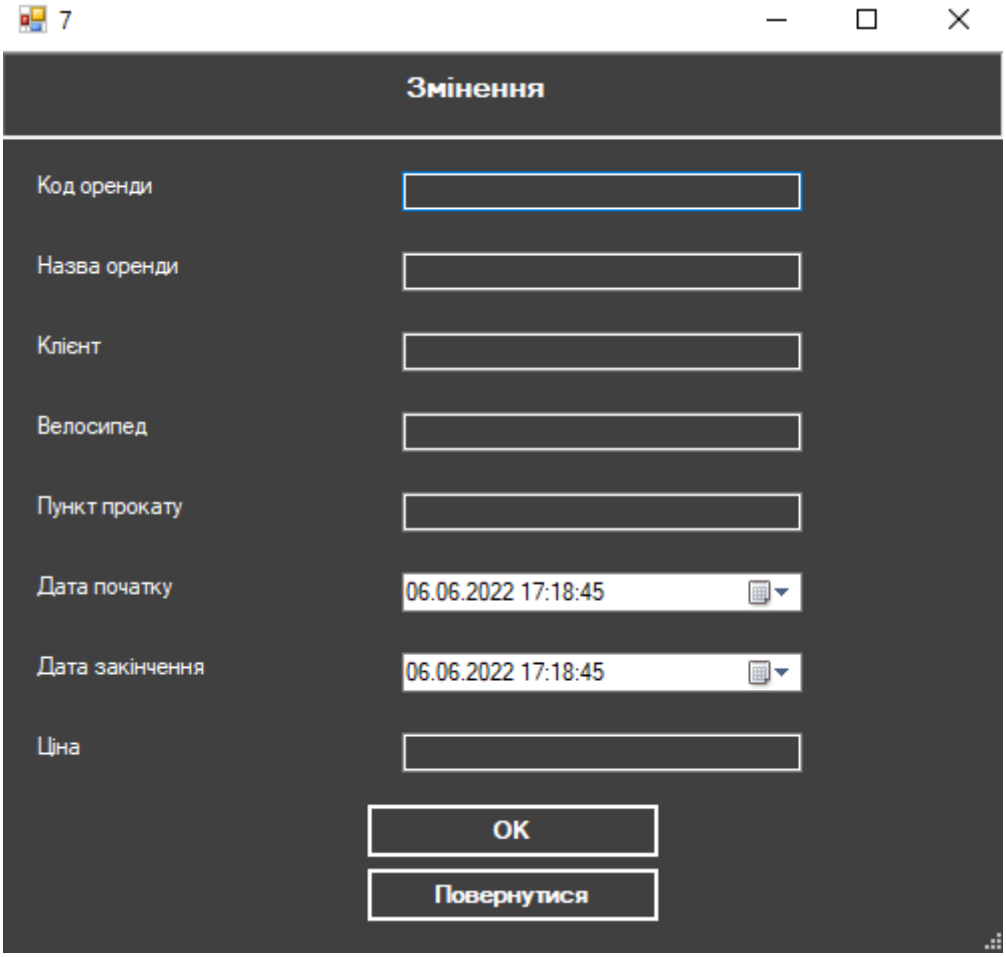

Рисунок Б.6 – Екранна форма редагування таблиці. Змінення даних.

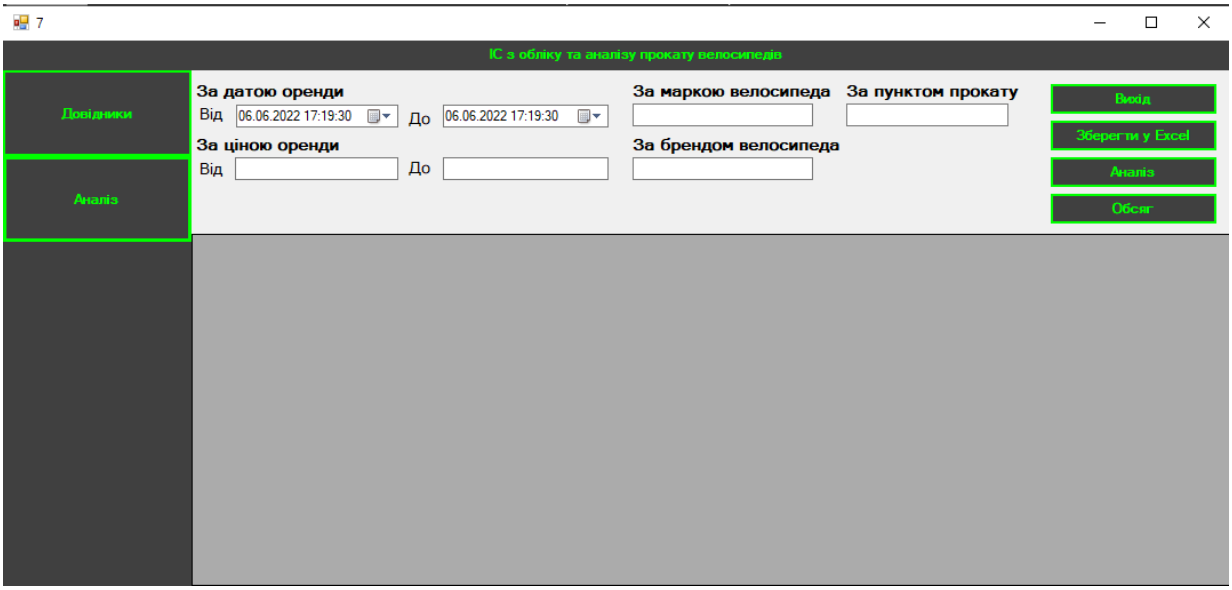

Рисунок Б.7 – Екранна форма аналізу даних

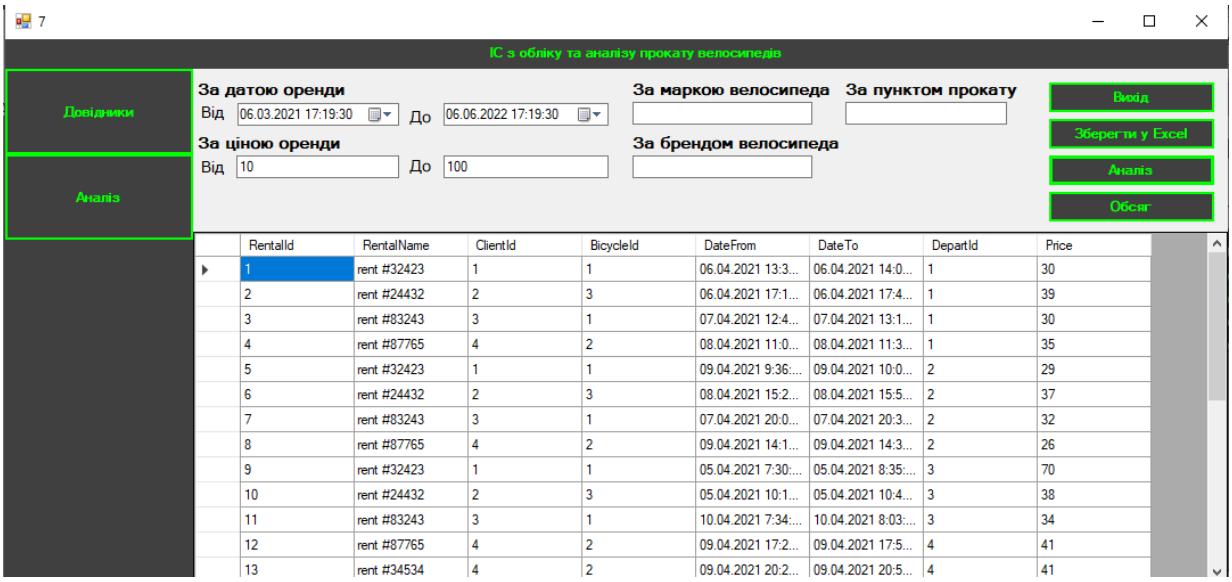

Рисунок Б.8 – Аналіз за діапазоном ціни та дати

	A	B	C	D	E	F	G	H
1	1	rent #3242	1	1	06.04.2021 13:35	06.04.2021 14:00	1	30
2	2	rent #2443	2	3	06.04.2021 17:13	06.04.2021 17:47	1	39
3	3	rent #8324	3	1	07.04.2021 12:45	07.04.2021 13:10	1	30
4	4	rent #8776	4	2	08.04.2021 11:00	08.04.2021 11:30	1	35
5	5	rent #3242	1	1	09.04.2021 9:36	09.04.2021 10:00	2	29
6	6	rent #2443	2	3	08.04.2021 15:23	08.04.2021 15:55	2	37
7	7	rent #8324	3	1	07.04.2021 20:05	07.04.2021 20:32	2	32
8	8	rent #8776	4	2	09.04.2021 14:10	09.04.2021 14:31	2	26
9	9	rent #3242	1	1	05.04.2021 7:30	05.04.2021 8:35	3	70
10	10	rent #2443	2	3	05.04.2021 10:13	05.04.2021 10:46	3	38
11	11	rent #8324	3	1	10.04.2021 7:34	10.04.2021 8:03	3	34
12	12	rent #8776	4	2	09.04.2021 17:23	09.04.2021 17:59	4	41
13	13	rent #3453	4	2	09.04.2021 20:23	09.04.2021 20:59	4	41

Рисунок Б.9 – Збереження результатів у файл Excel

Додаток В

Тексти програми

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;

namespace BicycleRentalAnalysis
{
    public class DBController
    {
        string connectionString;
        string sqlExpression;
        SqlDataAdapter adapter;
        DataSet dataSet;

        public DBController()
        {
            connectionString = "Server=DESKTOP-
GPEV3D5;Database=Bicycle_Rental;Trusted_Connection=True;";

        }

        public void DataExtraction(string tableSelect)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                sqlExpression = $"SELECT * FROM {tableSelect}";
                adapter = new SqlDataAdapter(sqlExpression, connection);
                dataSet = new DataSet();
                adapter.Fill(dataSet);
            }
        }

        public void Extract(string sqlExpression)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {

                adapter = new SqlDataAdapter(sqlExpression, connection);
                dataSet = new DataSet();
```

```
                adapter.Fill(dataSet);
            }
        }
        public DataSet GetDataSet()
        {
            return dataSet;
        }

        public void AddData(string sqlExpression)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                this.sqlExpression = sqlExpression;
                SqlCommand command = new SqlCommand(sqlExpression, connection);
                command.ExecuteNonQuery();
            }

        }

        public void DeleteData(string sqlExpression)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                this.sqlExpression = sqlExpression;
                SqlCommand command = new SqlCommand(sqlExpression, connection);
                command.ExecuteNonQuery();
            }
        }

        public void Updatedata(string tableSelect, string fieldSet, string fieldWhere)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                sqlExpression = $"UPDATE {tableSelect} SET {fieldSet} WHERE
{fieldWhere}";
                SqlCommand command = new SqlCommand(sqlExpression, connection);
                command.ExecuteNonQuery();
            }
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data;

namespace BicycleRentalAnalysis
{
    public class RentalAccounting
    {
        DBController dbController;
        string tableSelect;
        //Form activeForm;

        public RentalAccounting()
        {
            dbController = new DBController();
        }

        public void View(DataGridView dataGridView, string tableSelect)
```

```
        {
            this.tableSelect = tableSelect;
            dbController.DataExtraction(tableSelect);
            foreach (DataTable dataTable in dbController.GetDataSet().Tables)
            {
                foreach (DataColumn column in dataTable.Columns)
                    dataGridView.Columns.Add(column.ColumnName, column.ColumnName);
                foreach (DataRow row in dataTable.Rows)
                {
                    dataGridView.Rows.Add(row.ItemArray);

                }
            }
        }

        public string GetTableSelect()
        {
            return tableSelect;
        }

        public void Add(string Name, string Client, string Bicycle, string dateF,
string dateT, string Point, float price)
        {
            var stringValue = $"INSERT INTO rentals (rentalname, clientid, bicycleid,
datefrom, dateto, departid, price) " +
                $"VALUES ('{Name}', {Client}, {Bicycle}, '{dateF}', '{dateT}', {Point},
{price})";
            dbController.AddData(stringValue);
        }

        public void Add(string Name, string Type, string Brand, string Year)
        {
            var stringValue = $"INSERT INTO bicycles (bicyclename, bicycletype,
brandid, bicycleyear) " +
                $"VALUES ('{Name}', '{Type}', {Brand}, {Year})";
            dbController.AddData(stringValue);
        }

        public void Add(string Name, string Passport, string CreditCard, string City,
string Address)
        {
            var stringValue = $"INSERT INTO clients (clientname, passport, creditcard,
cityid, clientaddress) " +
                $"VALUES ('{Name}', '{Passport}', '{CreditCard}', {City},
'{Address}')";
            dbController.AddData(stringValue);
        }

        public void Add(string Name, string Address)
        {
            var stringValue = $"INSERT INTO departments (departname, departaddress) " +
                $"VALUES ('{Name}', '{Address}')";
            dbController.AddData(stringValue);
        }


        public void Delete(string Id)
        {
            var stringValue = $"DELETE FROM {tableSelect} WHERE {Id}";
            dbController.DeleteData(stringValue);
        }


        public void Update(string Id, string Name, string Client, string Bicycle,
string Point, string dateF, string dateT)
        {
```

```
            var stringValue = $"{Id},'{Name}', {Client}, {Bicycle}, '{dateF}',
'{dateF}'";
           // dbController.AddData(tableSelect, stringValue);
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data;

namespace BicycleRentalAnalysis
{
    public class RentalAnalysis
    {
        DBController dbController;
        string stringExpression;

        public RentalAnalysis()
        {
            dbController = new DBController();
        }

        public void Analyze(float price, string date)
        {
            stringExpression = $"SELECT * FROM rentals WHERE price = {price} and
dateFrom = '{date}'";
        }
        public void Analyze(float priceFrom, float PriceTo, string dateFrom, string
dateTo)
        {
            stringExpression = $"SELECT * FROM rentals " +
                $"WHERE price BETWEEN {priceFrom} AND {PriceTo} " +
                $"AND DateFrom BETWEEN '{dateFrom}' AND '{dateTo}'" +
                $"AND DateTo BETWEEN '{dateFrom}' AND '{dateTo}'";
            dbController.Extract(stringExpression);
        }

        /*
        public void Analyze(string bicycleName, string bicycleBrand)
        {
            stringExpression = $"SELECT bicycles.BicycleName, bicycles.BicycleType,
brands.BrandName " +
                $"FROM bicycles, brands " +
                $"WHERE bicycles.BrandID = brands.BrandId " +
                $"and bicycles.BicycleName = '{bicycleName}' and brands.BrandName =
'{bicycleBrand}' ";
            dbController.Extract(stringExpression);
        }
        */


        public void Analyze(string departmentName, float price, string bicycleName)
        {
            stringExpression = $"SELECT departments.DepartName, rentals.Price,
bicycles.BicycleName " +
                 $"FROM bicycles, departments, rentals " +
                 $"WHERE rentals.DepartID = departments.DepartId " +
                 $"and rentals.BicycleID = bicycles.BicycleId " +
                 $"and departments.DepartName = '{departmentName}' " +
                 $"and rentals.price = {price} " +
                 $"and bicycles.BicycleName = '{bicycleName}'";
            dbController.Extract(stringExpression);
```

```
        }

        /// <summary>
        /// Сума доходу велосипеда {bicycleName} пункту прокату {DepartId} (грн)
        /// </summary>
        /// <param name="bicycleName"></param>
        /// <param name="DepartId"></param>
        /// <param name="DateFrom"></param>
        /// <param name="DateTo"></param>
        public void Analyze(string bicycleName, int DepartId, string DateFrom, string
DateTo)
        {
            stringExpression = $"select SUM(price) AS 'Сума доходу велосипеда
{bicycleName} пункту прокату {DepartId} (грн)' from rentals, bicycles " +
                $"where rentals.BicycleId = bicycles.BicycleId " +
                $"AND BicycleName = '{bicycleName}' AND DepartId = {DepartId} " +
                $"AND DateFrom BETWEEN '{DateFrom}' AND '{DateTo}'" +
                $"AND DateTo BETWEEN '{DateFrom}' AND '{DateTo}'";
            dbController.Extract(stringExpression);
        }

        /// <summary>
        /// Сума доходу велосипеда {bicycleName} (грн)
        /// </summary>
        /// <param name="bicycleName"></param>
        /// <param name="DateFrom"></param>
        /// <param name="DateTo"></param>
        public void Analyze(string bicycleName, string DateFrom, string DateTo)
        {
            stringExpression = $"select SUM(price) AS 'Сума доходу велосипеда
{bicycleName} (грн)' from rentals, bicycles " +
                $"where rentals.BicycleId = bicycles.BicycleId " +
                $"AND BicycleName = '{bicycleName}' " +
                $"AND DateFrom BETWEEN '{DateFrom}' AND '{DateTo}'" +
                $"AND DateTo BETWEEN '{DateFrom}' AND '{DateTo}'";
            dbController.Extract(stringExpression);
        }

        /// <summary>
        /// Обсяг прокату велосипеда {bicycleName} пункту прокату {DepartId}
        /// </summary>
        /// <param name="bicycleName"></param>
        /// <param name="DepartId"></param>
        public void Analyze(string bicycleName, int DepartId)
        {
            stringExpression = $"select COUNT(*) AS 'Обсяг прокату велосипеда
{bicycleName} пункту прокату {DepartId}' from rentals, bicycles " +
                $"where rentals.BicycleId = bicycles.BicycleId " +
                $"AND BicycleName = '{bicycleName}' AND DepartId = {DepartId}";
            dbController.Extract(stringExpression);
        }

        /// <summary>
        /// 'Обсяг прокату всіх велосипедів пункту прокату {DepartId} за період
        /// </summary>
        /// <param name="DepartId"></param>
        /// <param name="DateFrom"></param>
        /// <param name="DateTo"></param>
        public void Analyze(int DepartId, string DateFrom, string DateTo)
        {
            stringExpression = $"DECLARE @dateF smalldatetime, @dateT smalldatetime " +
                $"SET @dateF = '{DateFrom}'; SET @dateT = '{DateTo}'; " +
                $"select COUNT(*) AS 'Обсяг прокату всіх велосипедів пункту прокату
{DepartId} за період', " +
                $"@dateF AS 'ВІД', @dateT AS 'ДО' from rentals where DepartId =
{DepartId} " +
                $"AND DateFrom BETWEEN '{DateFrom}' AND '{DateTo}' " +
```

```
                $"AND DateTo BETWEEN '{DateFrom}' AND '{DateTo}'";
            dbController.Extract(stringExpression);
        }

        public void Analyze(string dateF, string dateT, int BicycleId)
        {
            stringExpression = $"DECLARE @sum_bicycle float, @sum_all float, @sum float
" +
                $"SET @sum_bicycle = (select SUM(DATEDIFF(MINUTE, DateFrom, DateTo))
from rentals " +
                $"where BicycleId = {BicycleId} AND DateFrom BETWEEN '{dateF}' AND
'{dateT}' AND " +
                $"DateTo BETWEEN '{dateF}' AND '{dateT}'); " +
                $"SET @sum_all = (select SUM(DATEDIFF(MINUTE, DateFrom, DateTo)) from
rentals " +
                $"where DateFrom BETWEEN '{dateF}' AND '{dateT}' AND  " +
                $"DateTo BETWEEN '{dateF}' AND '{dateT}'); " +
                $"SET @sum = @sum_bicycle / @sum_all;" +
                $"SELECT @sum AS 'Коефіцієнт популряності'";
            dbController.Extract(stringExpression);
        }

        public void Report(ExcelReport excelReport, DataGridView dataGridView)
        {
            excelReport.Report(dataGridView);
        }

        public void ViewResult(DataGridView dataGridView)
        {
            foreach (DataTable dataTable in dbController.GetDataSet().Tables)
            {
                foreach (DataColumn column in dataTable.Columns)
                    dataGridView.Columns.Add(column.ColumnName, column.ColumnName);
                foreach (DataRow row in dataTable.Rows)
                {
                    dataGridView.Rows.Add(row.ItemArray);

                }
            }
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BicycleRentalAnalysis
{
    interface IReport
    {
        void Report(DataGridView dataGridView);
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BicycleRentalAnalysis
{
    static class Program
```

```
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Reflection;
using Excel = Microsoft.Office.Interop.Excel;
using System.Data;


namespace BicycleRentalAnalysis
{
    public class ExcelReport : IReport
    {
        public void Report(DataGridView dataGridView)
        {
            SaveFileDialog sfd = new SaveFileDialog();
            DialogResult dr = sfd.ShowDialog();
            if (dr == DialogResult.OK)
            {
                Excel.Application ExcelApp = new Excel.Application();
                Excel.Workbook workbook = ExcelApp.Workbooks.Add();
                Excel.Worksheet worksheet = workbook.ActiveSheet;

                for(int i = 1; i < dataGridView.RowCount + 1; i++)
                {
                    for(int j = 1; j < dataGridView.ColumnCount + 1; j++)
                    {
                        worksheet.Rows[i].Columns[j] = dataGridView.Rows[i - 1].Cells[j
- 1].Value;
                    }
                }
                ExcelApp.AlertBeforeOverwriting = false;
                workbook.SaveAs(sfd.FileName + ".xls");
                ExcelApp.Quit();
                MessageBox.Show("Records Successfully Exported");
            }
            else if (dr != DialogResult.OK)
            {
                MessageBox.Show("Not Exported");
            }
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BicycleRentalAnalysis
{
    public partial class AnalysisForm : Form
    {
        RentalAnalysis rentalAnalysis;
        RentalAccounting rentalAccounting;
        ExcelReport excelReport;
        public AnalysisForm()
        {
            InitializeComponent();
            rentalAnalysis = new RentalAnalysis();
            rentalAccounting = new RentalAccounting();
            excelReport = new ExcelReport();
            dataGridView.Rows.Clear();
            dataGridView.Columns.Clear();
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void btnAnalysis_Click(object sender, EventArgs e)
        {
            dataGridView.Rows.Clear();
            dataGridView.Columns.Clear();
            if(String.IsNullOrEmpty(textBicycle.Text) &&
String.IsNullOrEmpty(textDepart.Text) && String.IsNullOrEmpty(textBrand.Text)
                && String.IsNullOrEmpty(textPriceFrom.Text) &&
String.IsNullOrEmpty(textPriceTo.Text))
                MessageBox.Show("Заповність необхідні поля");
            else
            {
                if (!String.IsNullOrEmpty(textBicycle.Text) &&
!String.IsNullOrEmpty(textDepart.Text) && !String.IsNullOrEmpty(textBrand.Text)
                && !String.IsNullOrEmpty(textPriceFrom.Text) &&
!String.IsNullOrEmpty(textPriceTo.Text))
                    MessageBox.Show("Заповність необхідні поля. Всі поля не можуть бути
заповненими!");
                else if (!String.IsNullOrEmpty(textPriceFrom.Text) &&
!String.IsNullOrEmpty(textPriceTo.Text))
                {
                   rentalAnalysis.Analyze(float.Parse(textPriceFrom.Text),
float.Parse(textPriceTo.Text), dateF.Value.ToString(), dateT.Value.ToString());
                   rentalAnalysis.ViewResult(dataGridView);
                }
                else if (!String.IsNullOrEmpty(textBicycle.Text) &&
!String.IsNullOrEmpty(textDepart.Text)
                    && String.IsNullOrEmpty(textPriceFrom.Text) &&
String.IsNullOrEmpty(textPriceTo.Text))
                {
                    /// Сума доходу велосипеда {bicycleName} пункту прокату {DepartId}
(грн)
                    rentalAnalysis.Analyze(textBicycle.Text,
int.Parse(textDepart.Text), dateF.Value.ToString(), dateT.Value.ToString());
                    rentalAnalysis.ViewResult(dataGridView);
                }
                else if (!String.IsNullOrEmpty(textBicycle.Text) &&
String.IsNullOrEmpty(textDepart.Text)
                    && String.IsNullOrEmpty(textPriceFrom.Text) &&
String.IsNullOrEmpty(textPriceTo.Text))
                {
                    /// Сума доходу велосипеда {bicycleName} (грн)
```

```
                    rentalAnalysis.Analyze(textBicycle.Text, dateF.Value.ToString(),
dateT.Value.ToString());
                    rentalAnalysis.ViewResult(dataGridView);
                }
                else
                    MessageBox.Show("Заповність необхідні поля. Аналізувати за обраними
ознаками неможна!");


            }


        }

        private void btnSaveFile_Click(object sender, EventArgs e)
        {
            rentalAnalysis.Report(excelReport, dataGridView);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            dataGridView.Rows.Clear();
            dataGridView.Columns.Clear();
            if (!String.IsNullOrEmpty(textBicycle.Text) &&
!String.IsNullOrEmpty(textDepart.Text) && String.IsNullOrEmpty(textBrand.Text)
                && String.IsNullOrEmpty(textPriceFrom.Text) &&
String.IsNullOrEmpty(textPriceTo.Text))
            {
                /// Обсяг прокату велосипеда {bicycleName} пункту прокату {DepartId}
                rentalAnalysis.Analyze(textBicycle.Text, int.Parse(textDepart.Text));
                rentalAnalysis.ViewResult(dataGridView);
            }
            else if (!String.IsNullOrEmpty(textDepart.Text) &&
String.IsNullOrEmpty(textBicycle.Text) && String.IsNullOrEmpty(textBrand.Text)
                && String.IsNullOrEmpty(textPriceFrom.Text) &&
String.IsNullOrEmpty(textPriceTo.Text))
            {
                ///Обсяг прокату всіх велосипедів пункту прокату {DepartId} за період
                rentalAnalysis.Analyze(int.Parse(textDepart.Text),
dateF.Value.ToString(), dateT.Value.ToString());
                rentalAnalysis.ViewResult(dataGridView);
            }
            else if (!String.IsNullOrEmpty(textBicycle.Text) &&
String.IsNullOrEmpty(textBrand.Text)
                && String.IsNullOrEmpty(textPriceFrom.Text) &&
String.IsNullOrEmpty(textPriceTo.Text) && String.IsNullOrEmpty(textDepart.Text))
            {
                ///Коефіціфєнт популярності
                rentalAnalysis.Analyze(dateF.Value.ToString(), dateT.Value.ToString(),
int.Parse(textBicycle.Text));
                rentalAnalysis.ViewResult(dataGridView);
            }
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BicycleRentalAnalysis
```

```
{
    public partial class BicycleEditForm : Form
    {
        RentalAccounting rentalAccounting;
        string mode;
        public BicycleEditForm(RentalAccounting rentalAccounting, string mode)
        {
            InitializeComponent();
            this.rentalAccounting = rentalAccounting;
            this.mode = mode;
            if (mode == "Add")
                AddMode();
            else if (mode == "Update")
                UpdateMode();
            else if (mode == "Delete")
                DeleteMode();


        }

        private void AddMode()
        {
            this.labelTitel.Text = "Додавання";

            labelName.Location = new Point(15, 60);
            labelType.Location = new Point(15, 100);
            labelBrand.Location = new Point(35, 140);
            labelYear.Location = new Point(45, 180);


            textBicycleName.Location = new Point(120, 60);
            comboBicycleType.Location = new Point(120, 100);
            textBicycleBrand.Location = new Point(120, 140);
            textBicycleYear.Location = new Point(120, 180);

            this.Controls.Add(this.labelYear);
            this.Controls.Add(this.labelName);
            this.Controls.Add(this.labelBrand);
            this.Controls.Add(this.labelType);


            this.Controls.Add(this.comboBicycleType);
            this.Controls.Add(this.textBicycleYear);
            this.Controls.Add(this.textBicycleBrand);
            this.Controls.Add(this.textBicycleName);

        }

        private void UpdateMode()
        {
            this.labelTitel.Text = "Зміненння";
            this.Controls.Add(this.labelId);
            this.Controls.Add(this.labelYear);
            this.Controls.Add(this.labelName);
            this.Controls.Add(this.labelBrand);
            this.Controls.Add(this.labelType);


            this.Controls.Add(this.comboBicycleType);
            this.Controls.Add(this.textBicycleYear);
            this.Controls.Add(this.textBicycleBrand);
            this.Controls.Add(this.textBicycleName);
            this.Controls.Add(this.textBicycleId);
        }

        private void DeleteMode()
        {
```

```
            this.labelTitel.Text = "Видалення";

            this.Controls.Add(this.labelId);
            this.Controls.Add(this.textBicycleId);

            this.Size = new Size(520, 250);

            this.btnOK.Location = new Point(185, 130);
            this.btnBack.Location = new Point(185, 160);


        }

        private void btnOK_Click(object sender, EventArgs e)
        {
            if (mode == "Add")
            {
                if (textBicycleName.Text == "" || comboBicycleType.Text == "" ||
textBicycleBrand.Text == "" || textBicycleYear.Text == "")
                    MessageBox.Show("Всі поля повинні будти заповнені");
                else if (textBicycleName.Text == "" && comboBicycleType.Text == "" &&
textBicycleBrand.Text == "" && textBicycleYear.Text == "")
                    MessageBox.Show("Всі поля повинні будти заповнені");
                else
                    rentalAccounting.Add(textBicycleName.Text, comboBicycleType.Text,
textBicycleBrand.Text, textBicycleYear.Text);
            }
            else if (mode == "Delete")
            {

                if (textBicycleId.Text == "")
                    MessageBox.Show("Всі поля повинні будти заповнені");
                else
                    rentalAccounting.Delete("BicycleId = " + textBicycleId.Text);
            }
            //else if (mode == "Update")

            this.Close();
        }

        private void btnBack_Click(object sender, EventArgs e)
        {
            this.Close();
        }

    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BicycleRentalAnalysis
{

    public partial class MainForm : Form
    {
        private Form activeForm;
        public MainForm()
        {
```

```
            InitializeComponent();
        }
        private void OpenChildForm(Form childForm)
        {
            if(activeForm != null)
            {
                activeForm.Close();
            }
            activeForm = childForm;
            childForm.TopLevel = false;
            childForm.FormBorderStyle = FormBorderStyle.None;
            childForm.Dock = DockStyle.Fill;
            this.panelDesktop.Controls.Add(childForm);
            this.panelDesktop.Tag = childForm;
            childForm.BringToFront();
            childForm.Show();
            //labelTopTitle.Text = childForm.Text + " - " + labelTopTitle.Text;


        }

        private void btnDirectory_Click(object sender, EventArgs e)
        {
            OpenChildForm(new TableSelectSubForm());
        }

        private void btnAnalysis_Click(object sender, EventArgs e)
        {
            OpenChildForm(new AnalysisForm());
        }
    }


}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BicycleRentalAnalysis
{
    public partial class RentalEditForm : Form
    {
        RentalAccounting rentalAccounting;
        string mode;

        public RentalEditForm(RentalAccounting rentalAccounting, string mode)
        {
            InitializeComponent();
            this.rentalAccounting = rentalAccounting;
            this.mode = mode;
            if (mode == "Add")
                AddMode();
            else if (mode == "Update")
                UpdateMode();
            else if (mode == "Delete")
                DeleteMode();

        }

        private void AddMode()
```

```
{
    this.labelTitel.Text = "Додавання";

    labelName.Location = new Point(15, 60);
    labelClient.Location = new Point(15, 100);
    labelBicycle.Location = new Point(15, 140);
    labelPoint.Location = new Point(15, 180);
    labelDateF.Location = new Point(15, 220);
    labelDateT.Location = new Point(15, 260);
    labelPrice.Location = new Point(15, 300);



    tbName.Location = new Point(200, 60);
    tbClient.Location = new Point(200, 100);
    tbBicycle.Location = new Point(200, 140);
    tbPoint.Location = new Point(200, 180);
    dateF.Location = new Point(200, 220);
    dateT.Location = new Point(200, 260);
    tbPrice.Location = new Point(200, 300);


    this.Controls.Add(labelName);
    this.Controls.Add(labelClient);
    this.Controls.Add(labelBicycle);
    this.Controls.Add(labelPoint);
    this.Controls.Add(labelDateF);
    this.Controls.Add(labelDateT);
    this.Controls.Add(labelPrice);


    this.Controls.Add(tbName);
    this.Controls.Add(tbClient);
    this.Controls.Add(tbBicycle);
    this.Controls.Add(tbPoint);
    this.Controls.Add(dateF);
    this.Controls.Add(dateT);
    this.Controls.Add(tbPrice);
}

private void UpdateMode()
{
    this.labelTitel.Text = "Зміненння";

    labelId.Location = new Point(15, 60);

    labelName.Location = new Point(15, 100);
    labelClient.Location = new Point(15, 140);
    labelBicycle.Location = new Point(15, 180);
    labelPoint.Location = new Point(15, 220);
    labelDateF.Location = new Point(15, 260);
    labelDateT.Location = new Point(15, 300);
    labelPrice.Location = new Point(15, 340);

    tbId.Location = new Point(200, 60);

    tbName.Location = new Point(200, 100);
    tbClient.Location = new Point(200, 140);
    tbBicycle.Location = new Point(200, 180);
    tbPoint.Location = new Point(200, 220);
    dateF.Location = new Point(200, 260);
    dateT.Location = new Point(200, 300);
    tbPrice.Location = new Point(200, 340);

    this.Controls.Add(labelId);
    this.Controls.Add(labelName);
    this.Controls.Add(labelClient);
```

```
            this.Controls.Add(labelBicycle);
            this.Controls.Add(labelPoint);
            this.Controls.Add(labelDateF);
            this.Controls.Add(labelDateT);
            this.Controls.Add(labelPrice);

            this.Controls.Add(tbId);
            this.Controls.Add(tbName);
            this.Controls.Add(tbClient);
            this.Controls.Add(tbBicycle);
            this.Controls.Add(tbPoint);
            this.Controls.Add(dateF);
            this.Controls.Add(dateT);
            this.Controls.Add(tbPrice);
        }

        private void DeleteMode()
        {

            this.labelTitel.Text = "Видалення";

            labelId.Location = new Point(15, 60);


            tbId.Location = new Point(200, 60);


            this.Controls.Add(labelId);


            this.Controls.Add(tbId);

            this.Size = new Size(520, 200);

            this.btnOK.Location = new Point(185, 100);
            this.btnBack.Location = new Point(185, 130);


        }

        private void btnOK_Click(object sender, EventArgs e)
        {

            if (mode == "Add")
            {
                if (tbName.Text == "" || tbClient.Text == "" || tbBicycle.Text == "" ||
tbPoint.Text == "")
                    MessageBox.Show("Всі поля повинні будти заповнені");
                else if (tbName.Text == "" && tbClient.Text == "" && tbBicycle.Text ==
"" && tbPoint.Text == "")
                    MessageBox.Show("Всі поля повинні будти заповнені");
                else
                    rentalAccounting.Add(tbName.Text, tbClient.Text, tbBicycle.Text,
dateF.Value.ToString(),
                        dateT.Value.ToString(), tbPoint.Text,
float.Parse(tbPrice.Text));
            }
            else if (mode == "Delete")
            {

                if (tbId.Text == "")
                    MessageBox.Show("Всі поля повинні будти заповнені");
                else
                    rentalAccounting.Delete("RentalId = " + tbId.Text);
            }
            //else if (mode == "Update")
            this.Close();
```

```
        }

        private void btnBack_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BicycleRentalAnalysis
{
    public partial class RentalPointEditForm : Form
    {
        RentalAccounting rentalAccounting;
        string mode;
        public RentalPointEditForm(RentalAccounting rentalAccounting, string mode)
        {
            InitializeComponent();
            this.rentalAccounting = rentalAccounting;
            this.mode = mode;
            if (mode == "Add")
                AddMode();
            else if (mode == "Update")
                UpdateMode();
            else if (mode == "Delete")
                DeleteMode();
        }

        private void DeleteMode()
        {
            this.labelTitel.Text = "Видалення";


            this.Size = new Size(520, 250);

            this.btnOK.Location = new Point(185, 130);
            this.btnBack.Location = new Point(185, 160);
            this.Controls.Add(this.textDepartId);
            this.Controls.Add(this.labelDepartId);
        }

        private void UpdateMode()
        {
            this.labelTitel.Text = "Зміненння";
            this.Controls.Add(this.textDepartAddress);
            this.Controls.Add(this.textDepartName);
            this.Controls.Add(this.textDepartId);
            this.Controls.Add(this.labelDepartAddress);
            this.Controls.Add(this.labelDepartName);
            this.Controls.Add(this.labelDepartId);
        }

        private void AddMode()
        {
            this.labelTitel.Text = "Додавання";

            labelDepartName.Location = new Point(15, 60);
```

```
            labelDepartAddress.Location = new Point(15, 100);



            textDepartName.Location = new Point(140, 60);
            textDepartAddress.Location = new Point(140, 100);


            this.Controls.Add(this.textDepartAddress);
            this.Controls.Add(this.textDepartName);
            this.Controls.Add(this.labelDepartAddress);
            this.Controls.Add(this.labelDepartName);
        }

        private void btnOK_Click(object sender, EventArgs e)
        {
            if (mode == "Add")
            {
                if (textDepartName.Text == "" || textDepartAddress.Text == "")
                    MessageBox.Show("Всі поля повинні будти заповнені");
                else if (textDepartName.Text == "" && textDepartAddress.Text == "")
                    MessageBox.Show("Всі поля повинні будти заповнені");
                else
                    rentalAccounting.Add(textDepartName.Text, textDepartAddress.Text);
            }
            else if (mode == "Delete")
            {

                if (textDepartId.Text == "")
                    MessageBox.Show("Всі поля повинні будти заповнені");
                else
                    rentalAccounting.Delete("DepartId = " + textDepartId.Text);
            }
            //else if (mode == "Update")
            this.Close();
        }

        private void btnBack_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BicycleRentalAnalysis
{

    public partial class TableSelectSubForm : Form
    {
        public TableSelectSubForm()
        {
            InitializeComponent();
        }

        private void btnRental_Click(object sender, EventArgs e)
        {
```

```
            TableViewForm tableForm = new TableViewForm("rentals");
            tableForm.Show();
        }

        private void btnBicycle_Click(object sender, EventArgs e)
        {
            TableViewForm tableForm = new TableViewForm("bicycles");
            tableForm.Show();
        }

        private void btnClient_Click(object sender, EventArgs e)
        {
            TableViewForm tableForm = new TableViewForm("clients");
            tableForm.Show();
        }

        private void btnDepart_Click(object sender, EventArgs e)
        {
            TableViewForm tableForm = new TableViewForm("departments");
            tableForm.Show();
        }

        private void btnBrand_Click(object sender, EventArgs e)
        {
            TableViewForm tableForm = new TableViewForm("brands");
            tableForm.Show();
        }

        private void btnCity_Click(object sender, EventArgs e)
        {
            TableViewForm tableForm = new TableViewForm("cities");
            tableForm.Show();
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BicycleRentalAnalysis
{
    public partial class TableViewForm : Form
    {
        RentalAccounting rentalAccounting;
        Form activeForm;
        public TableViewForm(string tableSelect)
        {
            InitializeComponent();
            rentalAccounting = new RentalAccounting();
            rentalAccounting.View(dataGridView, tableSelect);
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            OpenEditForm("Add");
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            OpenEditForm("Delete");
```

```
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            OpenEditForm("Update");
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void OpenEditForm(string mode)
        {
            if (rentalAccounting.GetTableSelect() == "bicycles")
                activeForm = new BicycleEditForm(rentalAccounting, mode);
            else if (rentalAccounting.GetTableSelect() == "rentals")
                activeForm = new RentalEditForm(rentalAccounting, mode);
            else if (rentalAccounting.GetTableSelect() == "departments")
                activeForm = new RentalPointEditForm(rentalAccounting, mode);
            else if (rentalAccounting.GetTableSelect() == "clients")
                activeForm = new ClientEditForm(rentalAccounting);

            try
            {
                activeForm.ShowDialog();

                dataGridView.Rows.Clear();
                dataGridView.Columns.Clear();
                rentalAccounting.View(dataGridView, rentalAccounting.GetTableSelect());
            }
            catch
            {
                MessageBox.Show("Оберіть іншу таблицю");
            }
        }
    }
}
```