МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

**КУРСОВА РОБОТА**

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Бази даних»

на тему:

**«База даних сайту прокату автомобілів»**

студента ІІ курсу групи ІПЗ-22-2

спеціальності 121 «Інженерія програмного забезпечення»

Конещука Олександра Сергійовича .

(прізвище, ім’я та по-батькові)

Керівник: Коротун О.В. .

Дата захисту: "29 " грудня\_ 2023 р.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.В. Коротун 1

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ С.М. Кравченко

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.В. Чижмотря 1

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ І.І. Сугоняк 1

(підпис) (прізвище та ініціали)

Житомир – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп’ютерних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень: бакалавр

Спеціальність 121 «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри ІПЗ

Тетяна Вакалюк

« » 2023 р.

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ СТУДЕНТУ

Конещуку Олександру Сергійовичу

1. Тема роботи: База даних сайту прокату автомобілів,.
2. Керівник роботи: Коротун О.В.
3. Строк подання студентом: « 28 » грудня 2023р.
4. Вихідні дані до роботи: Розробка бази даних сайту прокату автомобілів, та програму для користування БД
5. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)
   * + 1. Аналіз інформаційних потоків та особливостей предметної області дослідження
       2. Проектування бази даних за напрямком курсової роботи
       3. Реалізація підсистеми обробки даних за напрямком курсової роботи

4. Адміністрування баз даних

1. Перелік графічного матеріалу(з точним зазначенням обов’язкових креслень)

1. Презентація до КР\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Посилання на репозиторій: <https://gitlab.com/oleksandrkoneschuk/course_2.git>

Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посади консультанта | Підпис, дата | |
| завдання  видав | завдання прийняв |
| 1-3 | КОРОТУН Ольга | 25.10.2023 | 25.10.2023 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання “ 25 ” жовтня 2023 р.

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів курсового проекту | Строк виконання етапів проекту | Примітки |
| 1 | Постановка задачі | 03.11.2023 | Виконано |
| 2 | Пошук, огляд та аналіз аналогічних розробок | 04.11.2023 | Виконано |
| 3 | Формулювання технічного завдання | 05.11.2023 | Виконано |
| 4 | Опрацювання літературних джерел | 06.11.2023 | Виконано |
| 5 | Проектування структури | 06.11.2023 | Виконано |
| 6 | Написання програмного коду | 06.11.2023 | Виконано |
| 7 | Відлагодження | 18.12.2023 | Виконано |
| 8 | Написання пояснювальної записки | 17.12.2023 | Виконано |
| 9 | Захист | 30.12.2023 |  |

**КАЛЕНДАРНИЙ ПЛАН**

**Студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Олександр КОНЕЩУК

(підпис) (прізвище та ініціали)

**Керівник проекту** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ольга КОРОТУН

(підпис) (прізвище та ініціали)

**РЕФЕРАТ**

Завдання на курсову роботу – створення бази даних сайту прокату автомобілів.

Пояснювальна записка до курсового проекту на тему “База даних сайту прокату автомобілів” складається з переліку умовних скорочень та словника, вступу, чотирьох розділів, висновків, списку використаної літератури та додатків.

Текстова частина викладена на 40 сторінках друкованого тексту.

Пояснювально записка містить 47 сторінки додатків. Список використаних джерел містить 10 джерел. У записці є 27 рисунків. Загальний обсяг роботи – 89 сторінок.

Складові пояснювальної записки: вступ, чотири розділи з власними висновками, головний висновок та список використаної літератури.

**Ключові слова:** БД, СУБД, ДАНІ, ПІДКЛЮЧЕННЯ, ОБРОБКА

**ЗМІСТ**

[ВСТУП 6](#_Toc154737509)

[РОЗДІЛ 1. ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ ПОТОКІВ ТА ОСОБЛИВОСТЕЙ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ 7](#_Toc154737510)

[1.1. Аналіз інформаційних потреб та визначення предметної області дослідження: 7](#_Toc154737511)

[1.2. Обґрунтування вибору засобів реалізації: 9](#_Toc154737512)

[РОЗДІЛ 2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ САЙТУ ПРОКАТІВ АВТОМОБІЛІВ 12](#_Toc154737513)

[2.1. Аналіз інформаційних процесів 12](#_Toc154737514)

[2.2. Проектування структури бази даних сайту прокату автомобілів 13](#_Toc154737515)

[2.3. Розробка математичної моделі та алгоритмів обробки даних в системи 17](#_Toc154737516)

[2.3.1. Математична модель 18](#_Toc154737517)

[2.3.2. Алгоритми обробки даних 19](#_Toc154737518)

[РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ОБРОБКИ ДАНИХ САЙТУ ПРОКАТУ АВТОМОБІЛІВ 21](#_Toc154737519)

[3.1. Проектування інтерфейсу обробки даних 21](#_Toc154737523)

[3.2. Реалізація операцій обробки даних в БД сайту прокату автомобілів 32](#_Toc154737524)

[3.3. Організація звітності системи 35](#_Toc154737525)

[РОЗДІЛ 4. АДМІНІСТРУВАННЯ БАЗ ДАНИХ 37](#_Toc154737526)

[4.1. Розробка заходів захисту інформації в БД 37](#_Toc154737527)

[4.2. Налаштування параметрів роботи SQL-сервера 38](#_Toc154737528)

[ВИСНОВКИ 40](#_Toc154737529)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 41](#_Toc154737530)

[ДОДАТКИ 42](#_Toc154737531)

[Додаток А 43](#_Toc154737532)

[Додаток Б 43](#_Toc154737533)

# ВСТУП

Сучасний розвиток інформаційних технологій визначає актуальність створення та вдосконалення інформаційних систем, зокрема, систем прокату автомобілів. З метою вдосконалення процесів управління автопрокатом та забезпечення зручного та ефективного користувацького інтерфейсу було обрано напрямок дослідження "База даних сайту прокату автомобілів".

**Мета курсової роботи** полягає в дослідженні особливостей проектування та реалізації баз даних для системи прокату автомобілів.

**Завданнями** курсової роботи є:

1. Аналіз теоретичних засад проектування та реалізації систем на основі баз даних.
2. Визначення інформаційних потреб предметної області дослідження.
3. Аналіз напрямку ризиків інформаційних потоків та їх структури.
4. Проектування бази даних за визначеною предметною областю.
5. Розробка математичної та алгоритмічної моделі функціонування системи на основі БД.
6. Реалізація БД та інтерфейсних засобів інформаційної системи.

**Предметом дослідження** є можливості застосування концепції баз даних та систем управління базами даних (СУБД) для забезпечення інформаційних потреб предметної області прокату автомобілів.

**Об'єктом дослідження** є методи та засоби проектування баз даних для оптимальної реалізації системи прокату автомобілів.

**Методи та засоби дослідження**, використані під час написання курсової роботи, включають монографічні, аналітичні, математичні, графічні методи, методи об'єктно-орієнтованого проектування та програмування, а також інші методи, які сприяють розумінню та вирішенню завдань, поставлених перед системою прокату автомобілів.

# ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ ПОТОКІВ ТА ОСОБЛИВОСТЕЙ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ

## Аналіз інформаційних потреб та визначення предметної області дослідження:

В сучасному світі, де мобільність та транспортна доступність мають величезне значення, прокат автомобілів стає популярним і зручним сервісом. Розробка бази даних для системи прокату автомобілів відповідає сучасним тенденціям та відповідає реальним потребам користувачів.

**Опис предметної області:**

1. *Таблиця "Klient" (Клієнт):* Містить дані про клієнтів, їх особисту інформацію.
2. *Таблиця "Avto" (Автомобіль):* Зберігає інформацію про наявні автомобілі, їх характеристики та умови прокату.
3. *Таблиця "Rent" (Прокат):* Містить дані про конкретні прокати, дати початку та закінчення, а також пов'язані з клієнтами та автомобілями.
4. *Таблиця "Discount" (Знижка):* Зберігає інформацію про різні види знижок та їх умови застосування для клієнтів.
5. *Таблиця "Fines" (Штрафи):* Містить інформацію про штрафи, пов'язані з прокатами, такі як затримка повернення автомобіля та інші.

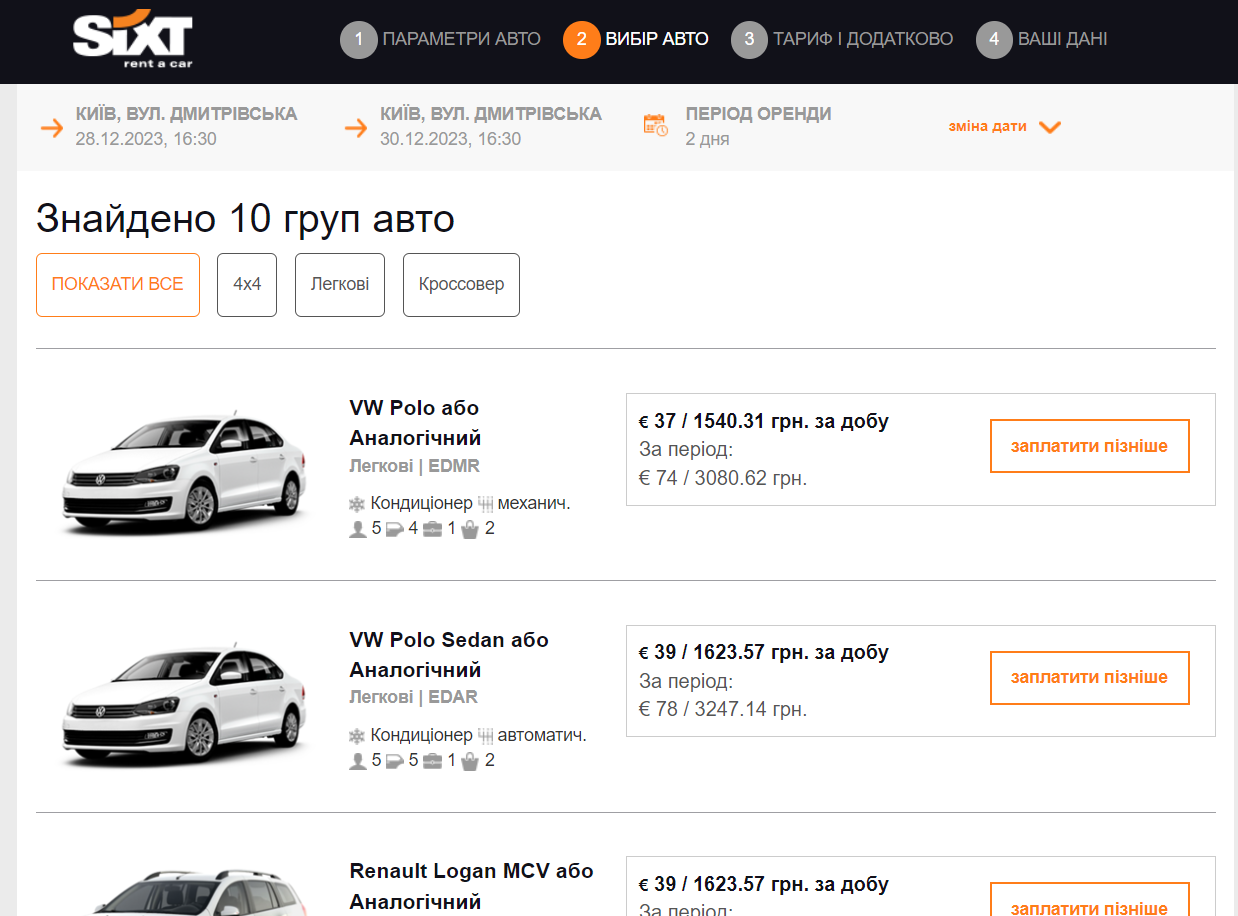
**Інформаційні потреби користувачів:**

* **Адміністратор системи:** Має можливість керувати автомобільним парком, реєстрацією клієнтів, встановленням та контролем знижок.
* **Клієнт:** Може швидко та зручно знаходити інформацію про доступні автомобілі, умови прокату та розрахунків, а також використовувати знижки.

**Очікувані результати та користь проекту:**

Реалізація даної бази даних покликана спростити та оптимізувати управління процесами прокату автомобілів, підвищити зручність взаємодії для користувачів та забезпечити ефективний контроль над фінансовими показниками системи.

Також були переглянуті наступні аналоги, такі як SIXT, Seven Cars:



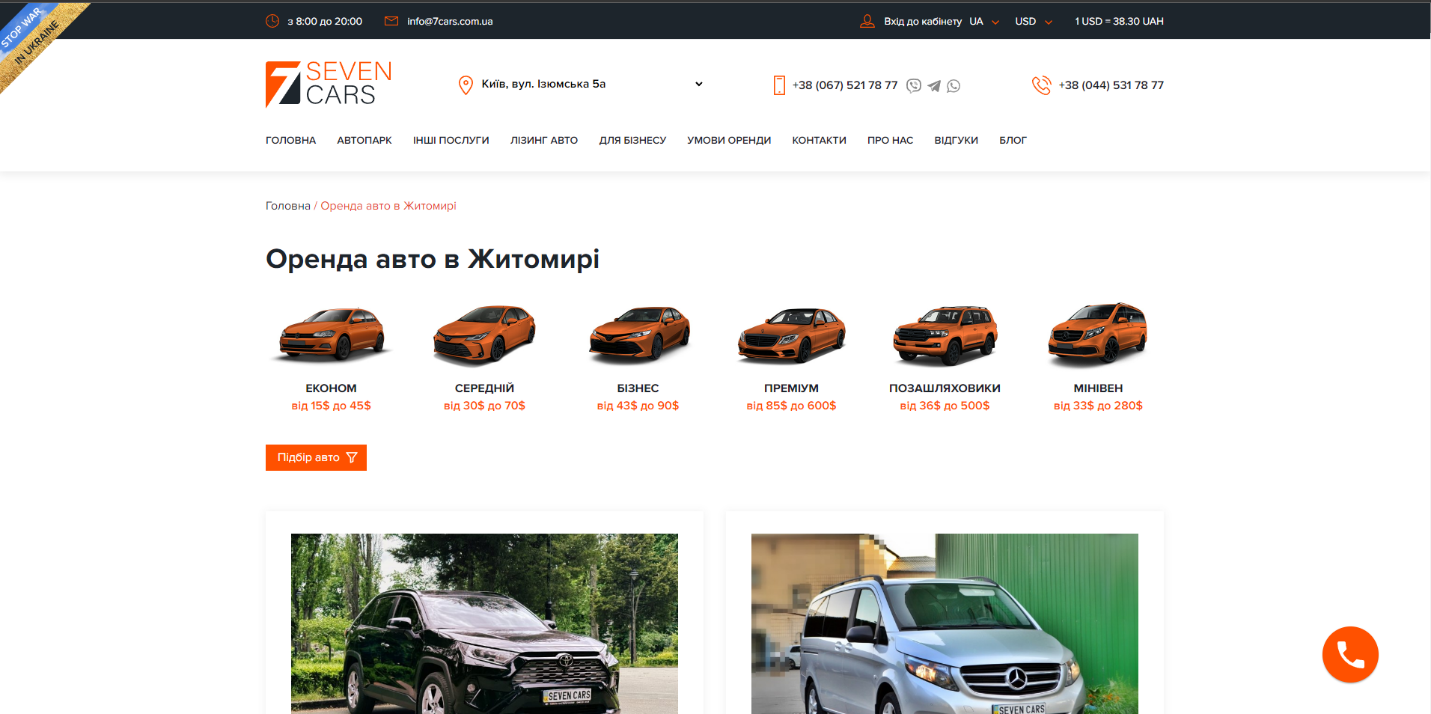
Рис.1.1. Аналіз продуктів з ринку

Рис.1.2. Аналіз продуктів з ринку

Ці аналоги мають плюси та мінуси:

Плюси:

* Доступний у багатьох містах України
* Можна орендувати авто закордоном
* Великий вибір авто
* Наявні додаткові послуги (водій, автокрісло для дитини і тд.)

Мінуси:

* Ціна стабільна, не зменшується від терміну прокату
* Місцями заплутаний інтерфейс

## Обґрунтування вибору засобів реалізації:

1. **Мова програмування C#:** Використання мови C# обґрунтоване високою продуктивністю та зручністю для розробки десктопних додатків у середовищі WPF (Windows Presentation Foundation). C# надає широкі можливості для реалізації функціоналу, обробки баз даних та взаємодії з користувачем.
2. **База даних MS SQL Server:** Використання Microsoft SQL Server обґрунтоване його надійністю, масштабованістю та інтеграцією з технологіями Microsoft. SQL Server забезпечує ефективне зберігання та обробку даних, а також можливість використання транзакцій для забезпечення консистентності даних.
3. **Windows Presentation Foundation (WPF):** Використання WPF обґрунтоване можливістю створення сучасних та естетичних інтерфейсів користувача для десктопних додатків. WPF дозволяє легко реалізувати багатофункціональні вікна, взаємодію з користувачем та відображення даних.
4. **Git для контролю версій:** Використання Git обґрунтоване його популярністю, можливістю колективної роботи та збереження історії змін коду.

Вибір цих засобів дозволить забезпечити високу якість та зручність у розробці та використанні системи прокату автомобілів.

Таблиця 1.1

Порівняння характеристик СУБД

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | MS SQL | MySQL | MongoDB |
| Адміністративне керування | Добре | Відмінно | Добре |
| Графічні інструменти | Відмінно | Добре | Відмінно |
| Простота обслуговування | Відмінно | Відмінно | Відмінно |
| Механізм даних | Добре | Відмінно | Відмінно |
| Робота з декількома ЦП |  | Відмінно | Добре |
| Одночасний доступ декількох користувачів | Добре | Відмінно | Добре |
| Підключення до Web | Задовільно | Відмінно | Відмінно |
| Інтеграція з іншими СУБД | Добре | Добре | Задовільно |
| Мова SQL | Відмінно | Добре | Власна мова запитів |
| Модель даних | Реляційна | Реляційна | Документно-орієнтовна |

В результаті дослідження СУБД було зроблено висновок, що для поставленої задачі краще використати MS SQL. Перевагами цієї СУБД є те, що вона реляційна, використовує мову запитів SQL, що дозволить додатку бути більш гнучким, висока швидкість роботи додатку, стабільність, функціональність, наявність великої кількості інформації про СУБД.

**Висновки до першого розділу**

У першому розділі був проведений аналіз інформаційних потреб та визначена предметна область дослідження в контексті розробки системи прокату автомобілів. Було розглянуто основні таблиці бази даних, які включають інформацію про клієнтів, автомобілі, прокати, знижки та штрафи. Також були визначені інформаційні потреби користувачів, таких як адміністратори та клієнти, з метою оптимізації управління та використання системи.

Також було порівняно

Важливою частиною розділу є обґрунтування вибору засобів реалізації. Визначено, що використання мови програмування C# спільно з технологіями WPF та базою даних SQL Server найбільш оптимальне для розробки десктопної системи прокату автомобілів. Також вказано на використання ADO.NET для забезпечення взаємодії з базою даних та Git для контролю версій, що забезпечить ефективний розвиток та управління проектом.

Було досліджено та проаналізовано СУБД, які можна використати для розробки бази даних.

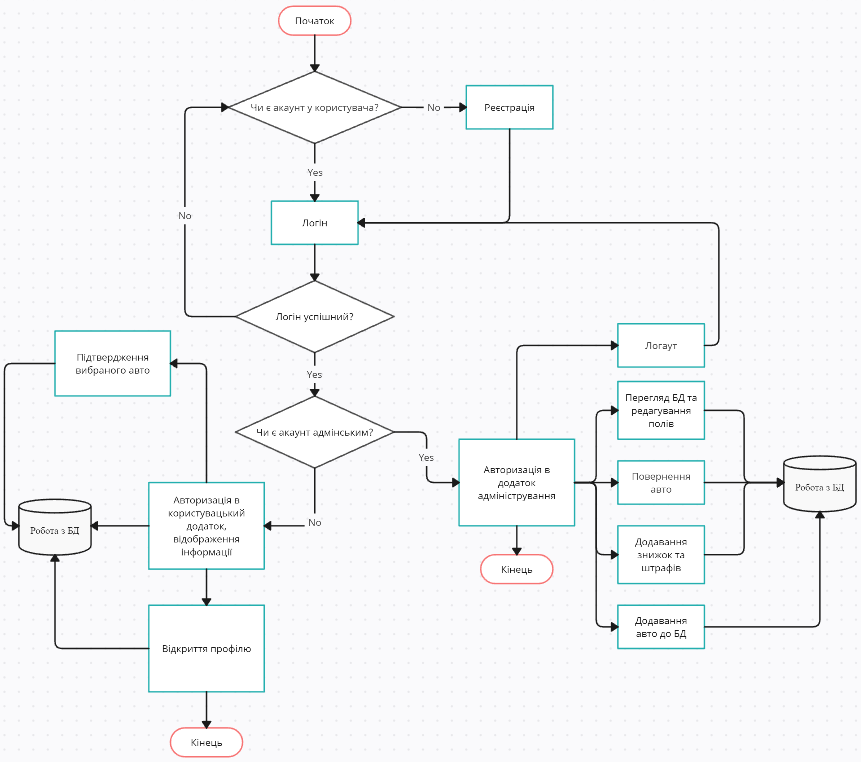
Зазначені висновки формують підґрунтя для подальших етапів розробки системи та визначають ключові аспекти, які будуть узяті до уваги при реалізації проекту.

# ПРОЕКТУВАННЯ БАЗИ ДАНИХ САЙТУ ПРОКАТІВ АВТОМОБІЛІВ

## Аналіз інформаційних процесів

Обгрунтувавши всі ідеї у визначенні функціоналу та потреби задачі, створився алгоритм роботи додатку.

При запуску додатку користувача зустріне вікно авторизаці, та можливість створити акаунт. Після створення акаунту користувача перемістить на вікно авторизації. Після авторизації користувачу буде відображено основна сторінка, в якій міститься інформація про наявні автомобілі, також є поле пошуку, сортування та вибору дати для ораенди авто. Коли користувач обере дату оренди та натисне на кнопку «Обрати авто» йому буде відображено вікно підтвердження замовлення, в якому відобразяться дати оренди та вартість оренди. Також у користувача є можливість переглянути свій профіль де містить коротка інформація про нього, історія замовлень, його штрафи та знижки. Якщо у вікні авторизації було введено данні, які є даними адміністратора, то буде відкрито меню адміністрування, з можливістю обрати різні функції такі як: додавання авто в БД, відкрити вікно в якому відображується інформація про базу данних в елементі dataGrid, є можливість змінювати вміст полів, повернення авто, тобто додавання в БД дату фактичного кінця оренди авто, та вікно для редагування, додавання даних.

Рис. 2.1. Діаграма алгоритму роботи клієньтського додатку

## Проектування структури бази даних сайту прокату автомобілів

Визначившись із приблизним функціоналом додатку та алгоритмами роботи як додатку, так і БД, ми можемо приступити до створення самої БД.

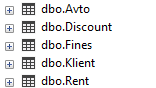




Рис. 2.2-3. Список таблиць та процедур в БД

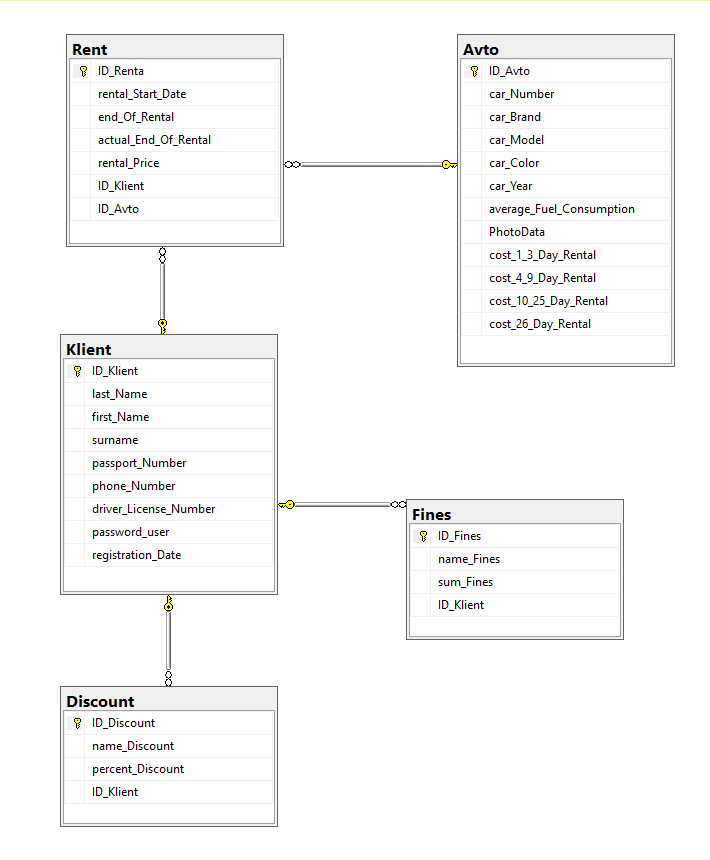
За даними таблицями можна створити діаграму бази данних

Рис.2.4. Діаграма БД

Таблиця 2.1

Таблиця Klient

| **Field Name** | **DataType** | **PK** | **FK** | **Description** |
| --- | --- | --- | --- | --- |
| ID\_Klient | int | + | - | Код клієнта |
| last\_Name | nvarchar(20) | - | - | Прізвище клієнта |
| first\_Name | nvarchar(20) | - | - | Ім’я клієнта |
| surname | nvarchar(20) | - | - | По батькові клієнта |
| passport\_Number | char(9) | - | - | Номер паспорту клієнта |
| phone\_Number | char(10) | - | - | Номер телофона клієнта форматом, також логін для авторизації в додатку |
| driver\_License\_Number | nvarchar(10) | - | - | Номер водійського посвідчення клієнта |
| password\_user | varchar(50) | - | - | Пароль клієнта |
| registration\_Date | date | - | - | Дата реєстрації клієнта |

Продовження табл. 2.1

Таблиця 2.2

Таблиця Avto

| **Field Name** | **DataType** | **PK** | **FK** | **Description** |
| --- | --- | --- | --- | --- |
| ID\_Avto | int | + | - | Код автомобіля |
| car\_Number | nvarchar(50) | - | - | Номер автомобіля вормату АМ0000МА |
| car\_Brand | nvarchar(50) | - | - | Марка автомобіля |
| car\_Model | nvarchar(50) | - | - | Модель автомобіля |
| car\_Color | nvarchar(50) | - | - | Колір автомобіля |
| car\_Year | date | - | - | Рік випуску автомобіля |
| average\_Fuel\_Consumption | decimal(10, 2) | - | - | Середне споживання палива на 100 км |
| PhotoData | VARBINARY (MAX)  Продовження таблиці 2.2 | - | - | Фото автомобіля, яке зберігається масивом байтів |
| cost\_1\_3\_Day\_Rental | decimal(10, 2) | - | - | Вартість 1-3 днів прокату в доларах США |
| cost\_4\_9\_Day\_Rental | decimal(10, 2) | - | - | Вартість 4-9 днів прокату в доларах США |
| cost\_10\_25\_Day\_Rental | decimal(10, 2) | - | - | Вартість 10-25 днів прокату в доларах США |
| cost\_26\_Day\_Rental | decimal(10, 2) | - | - | Вартість 26+ днів прокату в доларах США |

Таблиця 2.3

Таблиця Rent

| **Field Name** | **DataType** | **PK** | **FK** | **Description** |
| --- | --- | --- | --- | --- |
| ID\_Renta | int | + | - | Код прокату |
| rental\_Start\_Date | date | - | - | Дата початку прокату автомобіля |
| end\_Of\_Rental | date | - | - | Кінець прокату автомобіля |
| actual\_End\_Of\_Rental | date | - | - | Фактичний кінець прокату автомобіля (коли повернули авто) |
| rental\_Price | decimal(10, 2) | - | - | Вартість оренди авто в залежності від кількості днів, та наявниї знижок |
| ID\_Klient | int | - | + | Код автомобіля |
| ID\_Avto | int | - | + | Код клієнта |

Продовження таблиці 2.3

Таблиця 2.4

Таблиця Fines

| **Field Name** | **DataType** | **PK** | **FK** | **Description** |
| --- | --- | --- | --- | --- |
| ID\_Fines | int | + | - | Код штрафа |
| name\_Fines | nvarchar(50) | - | - | Назва штрафу |
| sum\_Fines | decimal(10, 2) | - | - | Сума штрафу |
| ID\_Klient | int | - | + | Код клієнта |

Таблиця 2.5

Таблиця Discount

| **Field Name** | **DataType** | **PK** | **FK** | **Description** |
| --- | --- | --- | --- | --- |
| ID\_Discount | int | + | - | Код знижк |
| name\_Discount | nvarchar(50) | - | - | Назва знижки |
| percent\_Discount | int | - | - | Відсоток знижки на поїздки |
| ID\_Klient | int | - | + | Код клієнта |

## Розробка математичної моделі та алгоритмів обробки даних в системи

Для системи управління прокатом автомобілів важливо розробити математичну модель та алгоритми обробки даних для ефективного функціонування та надання користувачам зручного та надійного сервісу. У цьому розділі ми розглянемо основні аспекти розробки математичної моделі та алгоритмів для системи прокату автомобілів.

### **Математична модель**

Математична модель системи прокату автомобілів базується на залученні кількох ключових параметрів та взаємодії між ними. Основні компоненти моделі включають:

* Клієнти (К): Представлені даними про клієнтів, які включають особисту інформацію.
* Автомобілі (А): Визначені характеристики автомобілів, такі як марка, модель, рік випуску та інші технічні параметри.
* Прокати (R): Інформація про окремі прокати, включаючи дату початку та закінчення, актуальну дату повернення, ціну прокату та інші відомості.
* Знижки (D): Різні види знижок, їх назва та відсоток знижки для кожного клієнта.
* Штрафи (F): Інформація про штрафи, які можуть бути застосовані в разі порушень, таких як затримка повернення автомобіля.

Математична модель може визначати залежності між цими компонентами, такі як розрахунок ціни прокату, знижок відповідно до історії клієнта та обчислення штрафів.

### **Алгоритми обробки даних**

Для ефективної роботи системи необхідно розробити алгоритми обробки даних, які забезпечать швидкий та точний доступ до інформації. Деякі ключові алгоритми включають:

* Розрахунок ціни прокату (СostСalculation): Алгоритм, який визначає ціну прокату в залежності від тривалості та інших параметрів.
* Розрахунок знижок (GetDiscount): Визначення величини знижки для кожного клієнта на основі його історії та наявних акцій.
* Перевірка доступності автомобілів (FindAvto): Перевірка наявності автомобілів на певні дати для запобігання конфліктам при бронюванні.
* Налаштування штрафів (trg\_overdue\_rent\_fine): Алгоритм, який визначає, чи є порушення правил прокату та застосовує штрафи.

Ці алгоритми допомагають системі ефективно виконувати свої функції та забезпечують користувачам зручний та надійний сервіс.

**Висновки до другого розділу:**

У цьому розділі було проведено аналіз інформаційних процесів для системи управління прокатом автомобілів. Визначено функціонал додатку та створено алгоритм його роботи. Зокрема, розглянуто процес авторизації користувачів, пошуку, сортування та бронювання автомобілів, а також адміністративні функції.

На основі цього була розроблена структура бази даних, яка включає таблиці для зберігання інформації про клієнтів, автомобілі, оренду, штрафи та знижки. Додатково, було надано опис полів кожної таблиці, визначено ключі та зовнішні ключі для забезпечення цілісності даних.

У розділі також розглянуто математичну модель системи, яка включає основні компоненти та залежності між ними. Алгоритми обробки даних були визначені для ефективного виконання функцій системи, таких як розрахунок ціни прокату, визначення знижок, перевірка наявності автомобілів та налаштування штрафів.

Загальний вигляд бази даних було проілюстровано за допомогою діаграми, що включає в себе зв'язки між таблицями.

Результатом цього розділу є чітке визначення структури бази даних та алгоритмів обробки даних, що є основою для подальшої реалізації та розробки системи управління прокатом автомобілів.

# РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ОБРОБКИ ДАНИХ САЙТУ ПРОКАТУ АВТОМОБІЛІВ



## Проектування інтерфейсу обробки даних

При запуску додатку користувача зустрічає вікно авторизації за існуючими даними аккаунту (логін та пароль) та можливість створити новий акаунт.

Рис.3.1. Авторизація

Якщо користувач не має акаунту в БД, то він може створити його натиснувши на надпис «Ще немає акаунту?», у вікні він зможе ввести свої дані, та відбудеться перевірка на наявність цього користувача в БД, якщо перевірка пройдена, то акаунт буде створено, а користувач перейде на вікно авторизації.

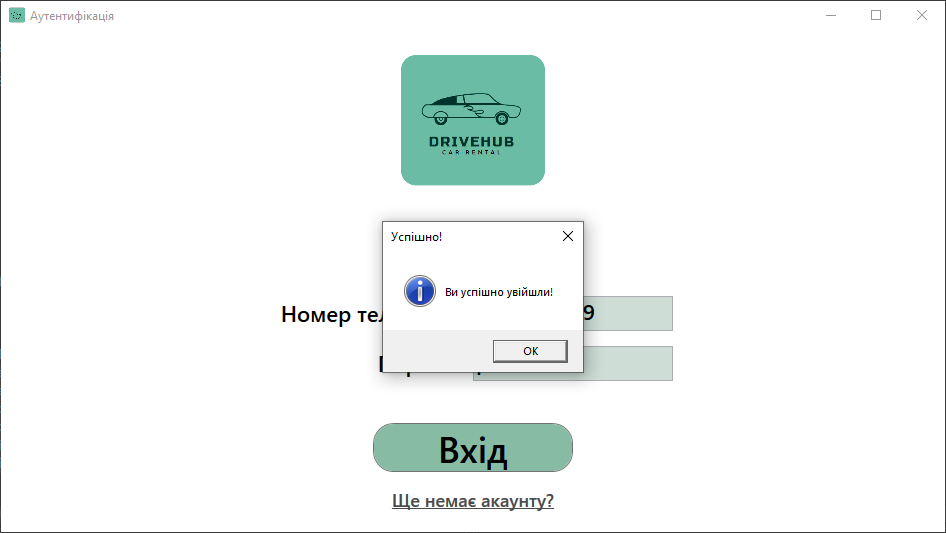
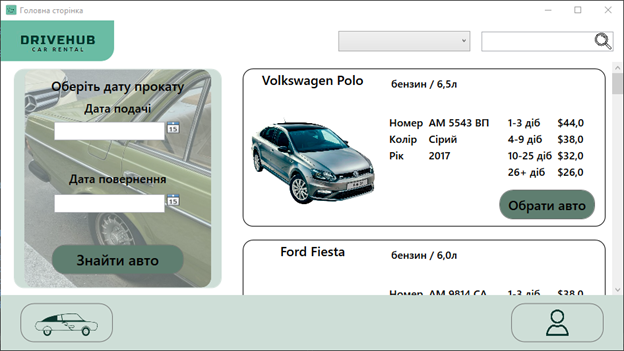
Рис.3.2. Вікно реєстрації

Рис.3.3. Успішна авторизація

Після того як користувач успішно авторизується йому відобразиться основна сторінка з даними про наявні автомобілі в прокаті(є можливість пролистування), також він зможе відшукати авто в пошуку, сортувати автомобілі за певними критеріми таким як:

* Рік випуску, за зростанням;
* Рік випуску, за спаданням;
* Від дешевих до дорогих;
* Від дорогих до дешевих;
* Розхід палива, за зростанням;
* Розхід палива, за спаданням

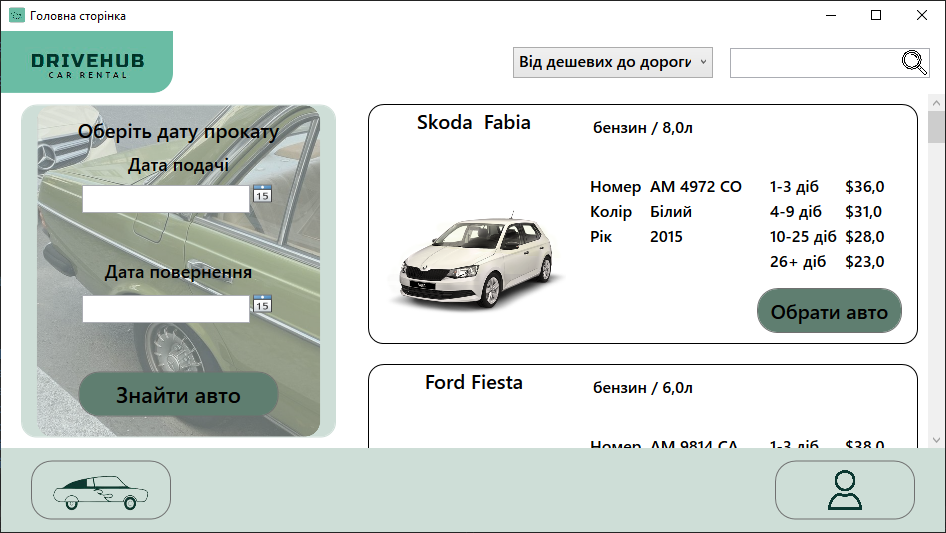
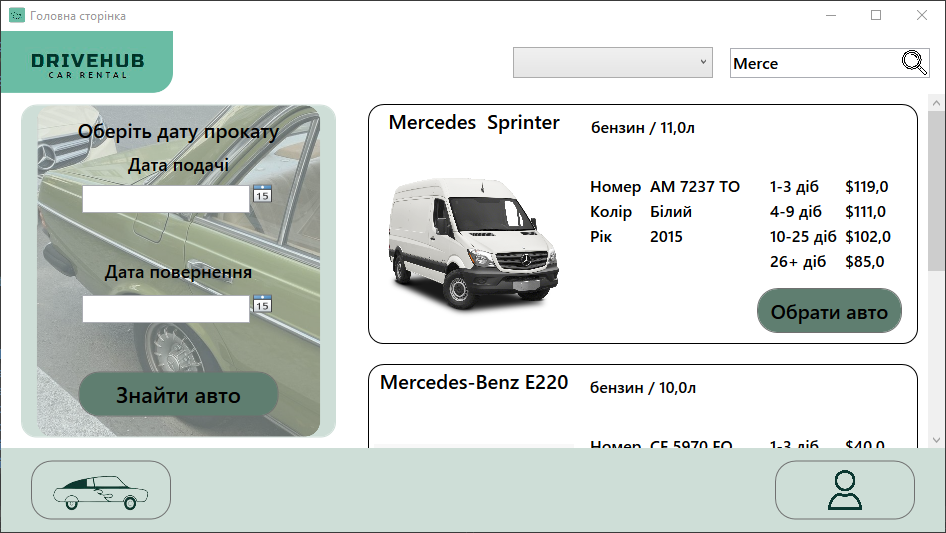
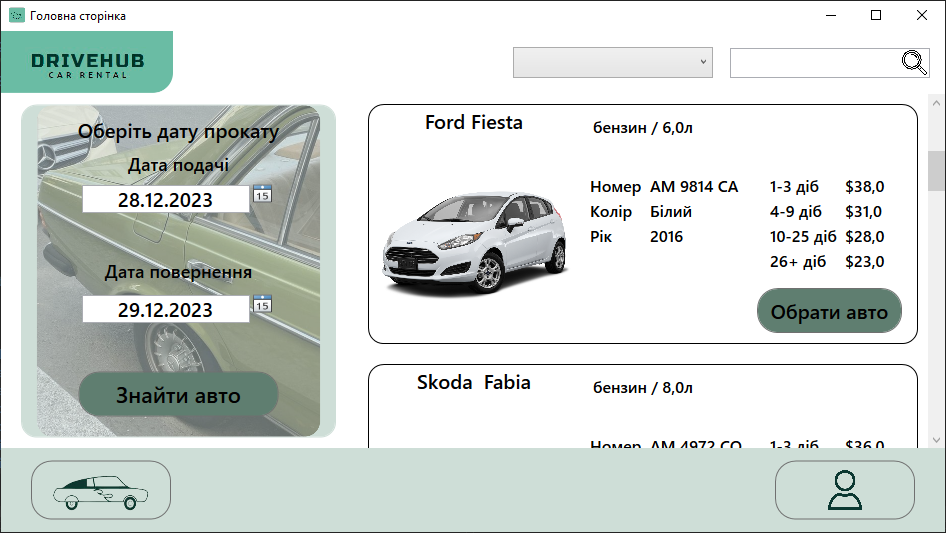
Рис.3.4. Основна сторінка

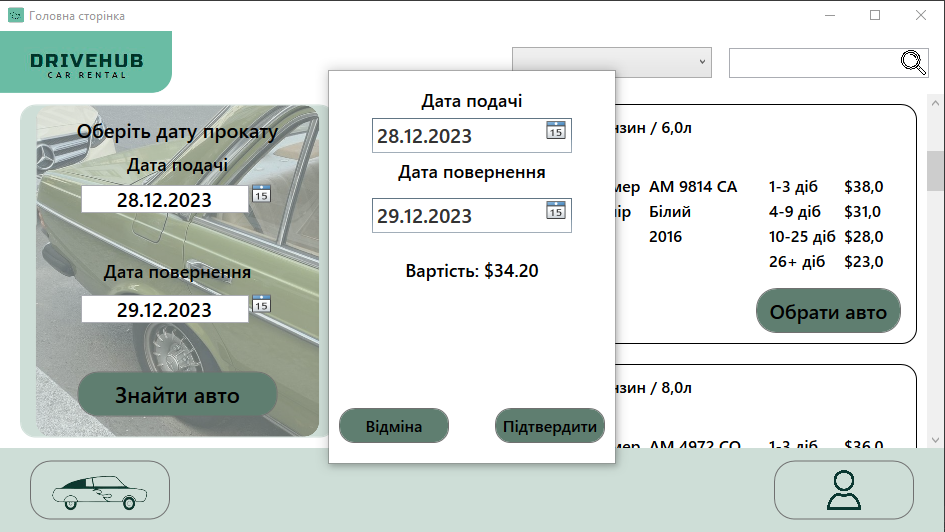
Рис.3.5. Сортування від дешевих авто, до дорогих

Рис.3.6. Пошук авто через пошук

Реалізована можливість шукати авто по даті, спочатку користувач обирає дату подічі авто, а потім дату повернення авто, тобто дати оренди. Після натискання на кнопку «Знайти авто» відбувається пошук вільних авто на ту дату, та вони відображаться в правій частині додатку

Рис.3.7. Відображення вільних авто на дати 28.12-29.12

Після того, як користувач обрав потрібну йому даду, та авто яке йому сподобалось, він натискає на кнопку «Обрати авто». Ціна автоматично підраховується в залежності від кількості днів прокату, та наявних знижок у клієнта, якщо знижок більше ніж одна, то вони не просто додаються а сумарна знижка підраховується наступним чином, до першої знижки додаються всі інші знижки поділені на 3. Якщо в користувача немає знижок, він отримає повідомлення про це. Також користувач, або підтвердить свій вибір після чого данні будуть занесені в БД, або відмовиться і він просто поверенться на минулу сторінку.

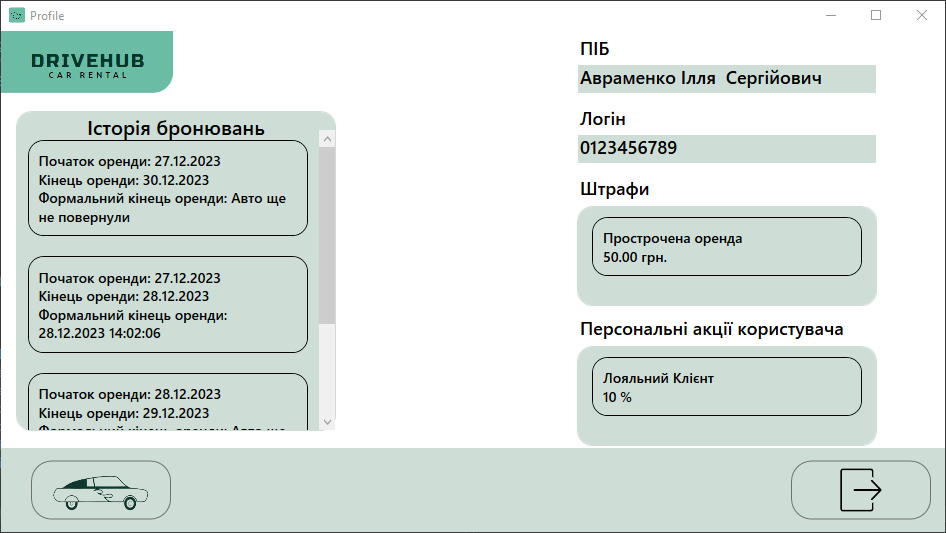
Рис.3.8. Підтвердження замовлення

У корстувача в нижньому меню сорінки є кнопка для переходу в профіль.



Рис.3.9. Кнопка для переходу в профіль

При переході в профіль користувач має змогу переглянути відомості про свій акаунт, а саме «ПІБ», «Логін», «Історію бронювань», «Штрафи», «Персональні акції користувача»

Рис.3.10. Відображення профіля

У нижньому меню користувач має можливість повернутися на головну сторінку та вийти з додатку.



Рис.3.11. Кнопка повернення в головне меню



Рис.3.12. Кнопка виходу з програми

Також реалізована можливість обробки даних та взаємодія з БД через додаток, для цього потрібно знати логін та пароль адміністратора логін – 0979742402, та пароль – admin. Після введення цих даних у вікні авторизації буде відкрито меню адміністрування.

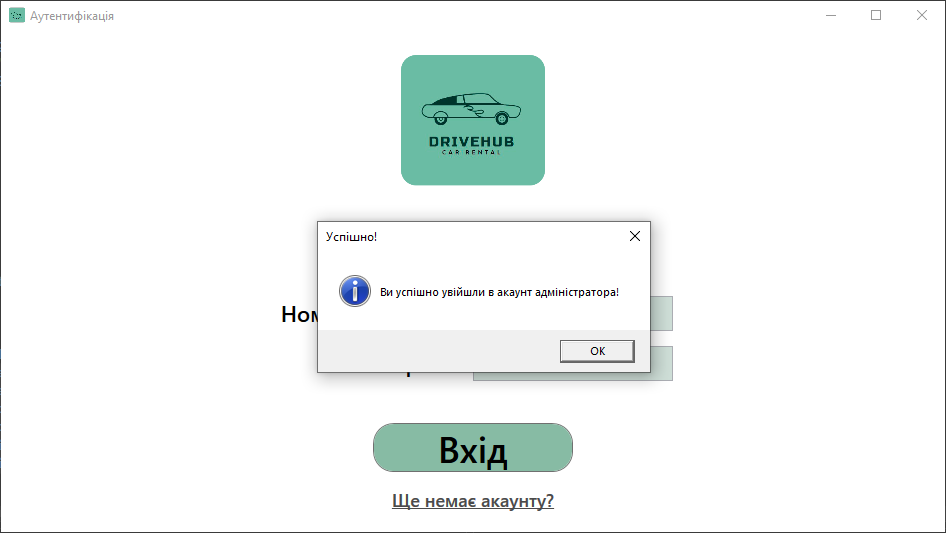
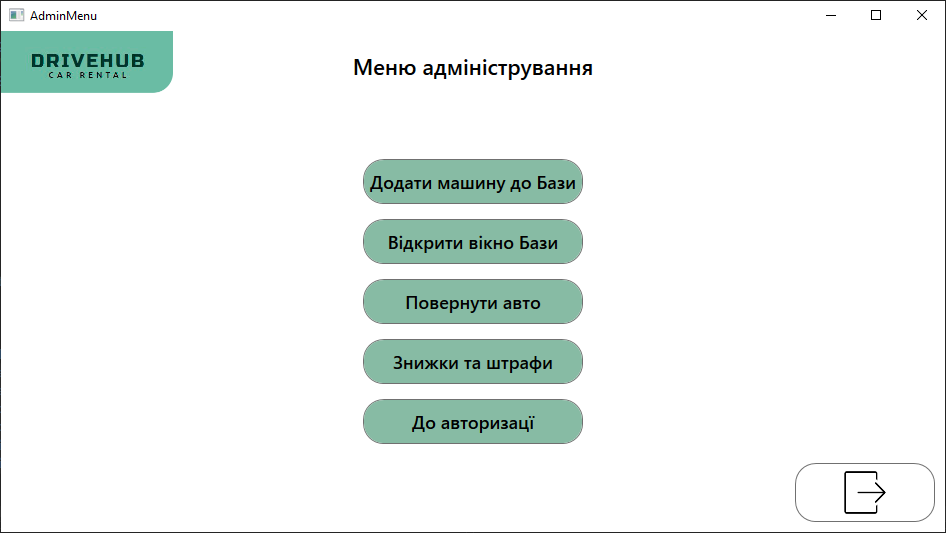
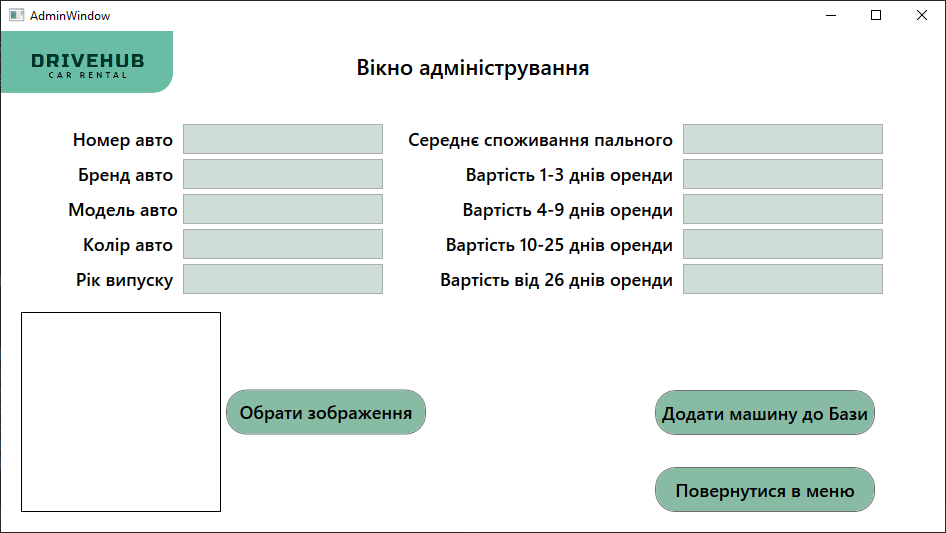
Рис.3.13. Вхід в додаток адміністрування

Рис.3.14. Меню адміністрування

У меню адміністрування адмін має можливість:

* Додати авто до БД
* Відкрити вікно БД
* Повернути авто(тобто додати фактичний кінець аренди)
* Відкрити вікно для додавання або зміни знидоу та штрафів користувачів
* Повернутися до авторизації

Вікно додавання машини до БД, адмін має можлиість додати авто та вказати всі потрібні данні номер авто, бренд авто, модель авто, колір авто, рік випуску авто, середне споживання палива на 100 км, вартість прокату за 1-3, 4-9, 10-25, 26+ днів прокату, також додати фото автомобіля, для якого буде виконано предпоказ. Є можливість повернення в меню.

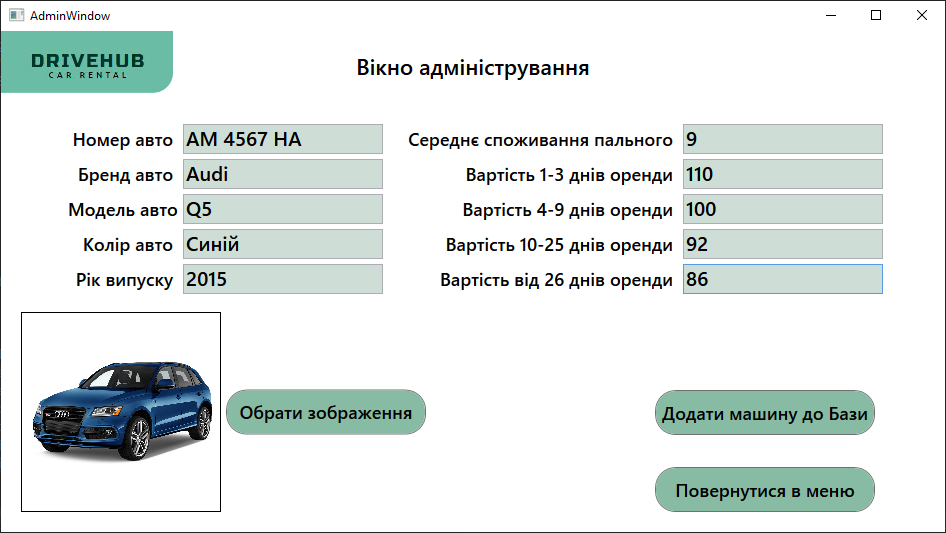
Рис.3.15. Вікно додавання авто

Рис.3.16. Вигляд вікна з заповненою інформацією

Якщо ж адміністратор обере відкрити вікно бази даних, то перед ним з’явиться таблиця dataGrid в якій можна переглянути інформацію про певну таблицю, також таблицю можна змінити за допомогою comboBox, реалізовано пошук інформації за допомогою процедур, та є можливість змінювати інформацію в певній комірці, якщо це дозволяє властивості комірки, наприклад РК комірки змінювати не можна. Інформація з комірки додається в textBox де її можна змінити, та відпрацити змінені данні за допомогою кнопки «Зберегти зміни». Також можна видалити рядок в якому було обано комірку за допомогою кнопки «Видалити обраний рядок».

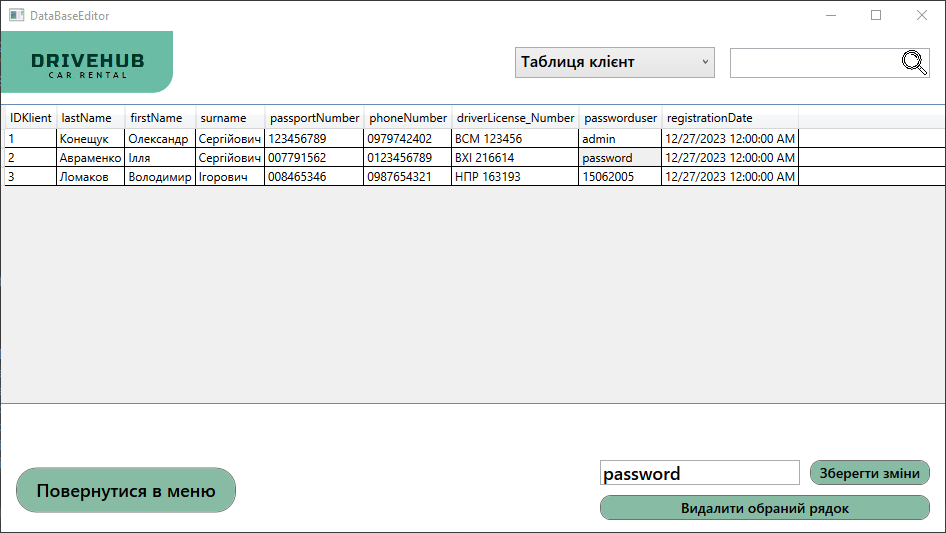


Рис.3.17. Вигляд вікна адміністрування БД в таблиці Klient

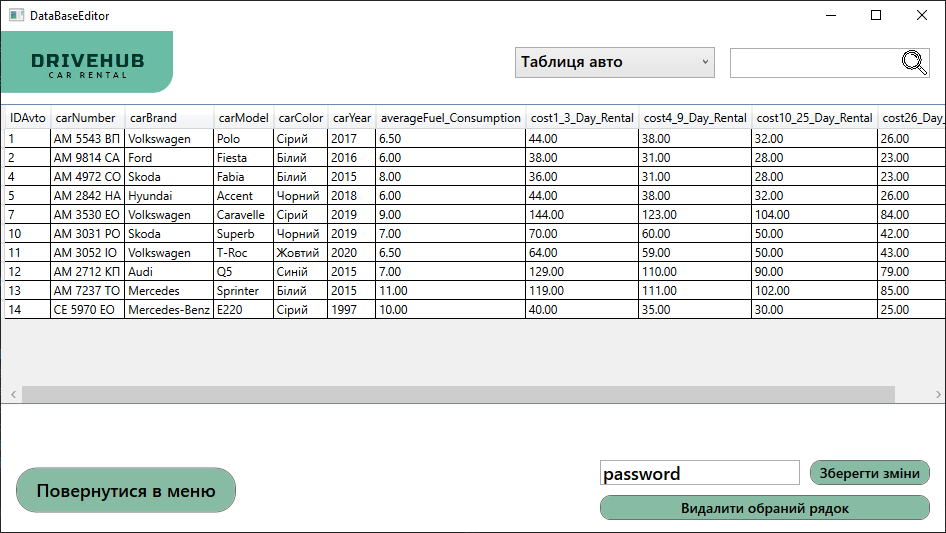


Рис.3.18. Вигляд вікна адміністрування БД в таблиці Avto, не надано можливості переглядати та змінювати вміст поля PhotoData, для запобігання не бажаних наслідків для фото, так як фото заноситься в байтовому вигляді

Коли адміністратор обере можливість повернення авто, то відкриється вікно в якому є така можливість. При виборі комірки з стовпчика actual\_End\_Of\_Rental буде надана можливість отримати поточну дату та час, яка автоматично заноситься до textBox в якому можна коригувати дату, або одразу відправити дані в БД. Користувачеві наданий обов’язок повернути авто в день закінчення оренди до 13.00, якщо ж клієнт проігнорував це правило, то за допомогою тригера, який порівнює фактичну дату закінчення оренди з очікуваною датою, автоматично буде внесено запис в табличку з штрафом за просрочену оренду авто

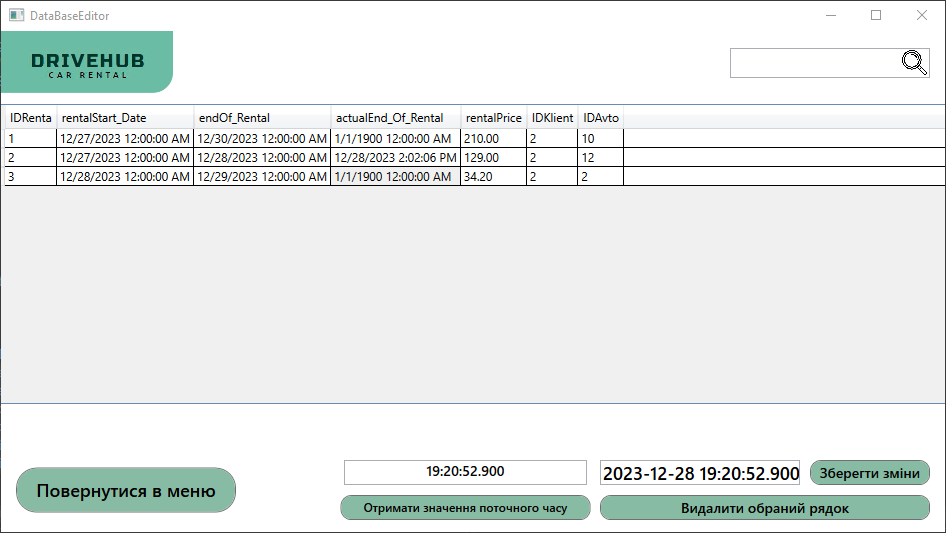
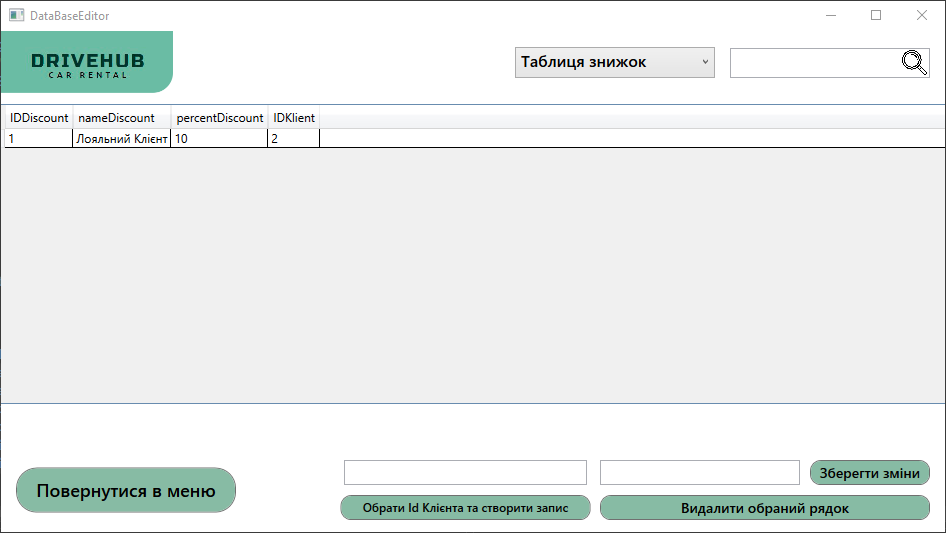


Рис.3.19. Вигляд вікна повернення авто

Якщо адміністратор обере івкно «Знижки та штрафи», він отримає можливість редагувати тільки данні з цих двох табличок з певною можливість додавання запису власноруч, для цього йому потрібно буде обрати ID\_Klient вписати його в textBox та натиснути кнопку «Обрати Id Клієнта та створити запис». Після цього у відкритій таблиці буде створено пустий запис для обраного клієнта, далі адміністратор повинен внести інформацію про знижку або штраф у відповідні комірки.

Рис.3.20. Вікно інформації про знижки та штрафи

## Реалізація операцій обробки даних в БД сайту прокату автомобілів

Підключення до бази даних реалізовано через створений клас. Тут реалізовано методи відкритя зв’язку з БД, закриття зв’язку, також зберігається рядок підключння до БД.

public class DataBase

{

SqlConnection sqlConnection = new SqlConnection(@"Data Source=DESKTOP-FUAH6BA;Initial Catalog=Car\_Rent;Integrated Security=True");

public void openConnection()

{

if(sqlConnection.State == System.Data.ConnectionState.Closed)

{

sqlConnection.Open();

}

}

public void closeConnection()

{

if(sqlConnection.State == System.Data.ConnectionState.Open)

{

sqlConnection.Close();

}

}

public SqlConnection getSqlConnection()

{

return sqlConnection;

}

}

Запити до БД відправлялися приблизно в такому форматі, сам рядок запиту створюється в інших методах, було використано reader для відображення інформації про наявні авто в елементі listBox, дані зчитуються з БД записуються в клас та в список, і виводять в додаток за допомогою textBox та властивості Binding

try

{

dataBase.openConnection();

using (SqlCommand command = new SqlCommand(queryString, dataBase.getSqlConnection()))

{

using (SqlDataReader reader = command.ExecuteReader())

{

List<Car> cars = new List<Car>();

while (reader.Read())

{

string carBrand = reader["car\_Brand"].ToString();

string carModel = reader["car\_Model"].ToString();

Car car = new Car

{

ID\_Avto = (int)reader["ID\_Avto"],

CarNumber = reader["car\_Number"].ToString(),

CarBrandModel = $"{carBrand} {carModel}",

CarColor = reader["car\_Color"].ToString(),

CarYear = (int)reader["car\_Year"],

AverageFuelConsumption = (decimal)reader["average\_Fuel\_Consumption"],

PhotoData = reader["PhotoData"] as byte[],

Cost1\_3DayRental = (decimal)reader["cost\_1\_3\_Day\_Rental"],

Cost4\_9DayRental = (decimal)reader["cost\_4\_9\_Day\_Rental"],

Cost10\_25DayRental = (decimal)reader["cost\_10\_25\_Day\_Rental"],

Cost26DayRental = (decimal)reader["cost\_26\_Day\_Rental"]

};

cars.Add(car);

}

return cars;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при отриманні даних про авто: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

return null;

}

finally

{

dataBase.closeConnection();

}

Приклад рядка запиту з пошуком інформації

private void Search()

{

string queryString = $"SELECT \* FROM Avto WHERE concat (ID\_Avto, car\_Number, car\_Brand, car\_Model, car\_Color, car\_Year, average\_Fuel\_Consumption) like '%" + searchTextBox.Text + "%'";

UpdateCarsCollection(queryString);

}

Для виклику процедур було створено два методи, в методі ProcessTable, отримується поточна таблиця та в рядок запиту додається назва процедури для кожної таблиці окремо, далі запит відправляюється в метод пошуку, там виконується з’єднання з БД відправка запиту, та шукомої інформації, після всіх дій відображається інформація які збіглася з введеним текстом, та закривається зв’язок БД.

private void ProcessTable()

{

string proc;

switch (\_showBy)

{

case "Klient":

proc = "SearchInfoInKlient";

Search(proc);

break;

case "Avto":

proc = "SearchInfoInAvto";

Search(proc);

break;

case "Rent":

proc = "SearchInfoInRent";

Search(proc);

break;

default:

MessageBox.Show($"Не відома таблиця: {\_showBy}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

}

private void Search(string proc)

{

try

{

dataBase.openConnection();

using (SqlCommand command = new SqlCommand(proc, dataBase.getSqlConnection()))

{

command.CommandType = CommandType.StoredProcedure;

command.Parameters.AddWithValue("@searchText", searchTextBox.Text);

using (SqlDataAdapter adapter = new SqlDataAdapter(command))

{

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGrid.ItemsSource = dataTable.DefaultView;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при пошуку даних: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

dataBase.closeConnection();

}

}

Запити на вставку інформації відбуваються наступним чином розберемо на прикладі вікна реєстрації. Після введення інформації у відповідні поля та натиску на кнопку яка викликає подію, виконується перевірка на введення інформації в усі поля, і далі формується запит на основі отриманих даних, та відправляється до БД за вже відомим принципом.

DateTime currentDate = DateTime.Now;

var registration\_Date = currentDate.ToString("yyyy-MM-dd");

string querystring = $"insert into Klient(last\_Name, first\_Name, surname, passport\_Number, phone\_Number, driver\_License\_Number, password\_user, registration\_Date) " +

$"values('{lastName}', '{firstName}', '{surname}', '{passportNumber}', '{phoneNumber}', '{driverLicense}', '{password}', '{registration\_Date}')";

SqlCommand command = new SqlCommand(querystring, dataBase.getSqlConnection());

dataBase.openConnection();

if(command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Ви успішно зареєстровані!", "Успішно!", MessageBoxButton.OK, MessageBoxImage.Information);

log\_in log\_in = new log\_in();

log\_in.Show();

this.Close();

}

else

MessageBox.Show("Не зареєстровано! Перевірте введення!", "Реєстрація не вдалась!", MessageBoxButton.OK, MessageBoxImage.Error);

dataBase.closeConnection();

}

Тригер був доданий в БД, та автоматично викликається при додаванні чи оновлені інформації в таблиці Rent

## Організація звітності системи

Під час розробки курсової роботи не було додано жодної організації звітності системи.

**Висновки до третього розділу**

Курсова робота з питань реалізації підсистеми обробки даних для системи прокату автомобілів містить широкий функціонал і обробляє різні аспекти від авторизації та реєстрації користувачів до взаємодії з базою даних та адмініструванням. Основні висновки та аналіз вашої роботи включають наступне:

- Реалізовано повноцінну систему авторизації та реєстрації для користувачів з можливістю створення нових облікових записів та перевірки їх унікальності.

- Розроблено зручний інтерфейс для користувачів, який включає авторизацію, перегляд доступних автомобілів, пошук, сортування та резервування.

- Додано можливість адміністратору додавати нові автомобілі до бази даних з вказанням всіх необхідних параметрів та фотографій.

- Реалізовано інтерфейс для перегляду та редагування інформації в базі даних, включаючи функції пошуку, зміни та видалення записів.

- Створено функціонал для повернення автомобілів, включаючи можливість редагування дати повернення та застосування штрафів у випадку просрочення.

- Реалізовано інтерфейс для управління знижками та штрафами для користувачів, включаючи можливість додавання нових записів та редагування існуючих.

- Не було виявлено жодних зазначень щодо організації звітності системи у вашій роботі.

Курсова робота включає широкий функціонал для системи прокату автомобілів та відображає глибокі знання розробки програмного забезпечення. Інтерфейс користувача та адміністративні можливості добре продумані. Додавання звітності та додаткових аналітичних можливостей може бути корисним напрямком для подальшого розвитку системи.

# АДМІНІСТРУВАННЯ БАЗ ДАНИХ

## Розробка заходів захисту інформації в БД

У даному розділі проводиться аналіз та розробка заходів захисту інформації в базі даних системи управління прокатом автомобілів. Забезпечення безпеки бази даних включає в себе різноманітні аспекти, що охоплюють автентифікацію, авторизацію, шифрування даних, регулярні резервні копії, моніторинг та журналювання, фізичний захист і безпеку серверів, оновлення та захист від зазначень, а також надійність та стійкість системи.

**Автентифікація та Авторизація:** Запроваджено надійну систему автентифікації для гарантування доступу лише вповноваженим користувачам. Крім того, використовуються різні рівні доступу для ефективного обмеження прав користувачів відповідно до їхніх обов'язків.

**Регулярні резервні копії:** Здійснюється регулярне створення резервних копій бази даних для запобігання втраті даних у випадку аварій, збоїв обладнання чи кібератак. Забезпечується зберігання резервних копій в безпечних місцях, а також періодичне тестування процесу відновлення.

**Фізичний захист і Безпека серверів:** Забезпечено фізичний захист серверів, де розміщена база даних. Фізичний доступ до серверних приміщень обмежений, а обладнання знаходиться в безпечному середовищі.

**Оновлення та Захист від Зазначень:** Регулярно оновлюється базове програмне забезпечення та система управління базою даних. Приділяється увага захисту від зазначень, використовуючи параметризовані запити та інші техніки.

**Надійність та Стійкість:** Забезпечується стійкість системи до перевантажень та великого обсягу запитів. Вживаються заходи для уникнення втрати даних та забезпечення найвищого рівня надійності.

## Налаштування параметрів роботи SQL-сервера

Налаштування параметрів роботи SQL-сервера є важливою частиною адміністрування бази даних. Нижче наведено кілька ключових параметрів, які налаштовувані в SQL-сервері курсової роботи:

**Максимальний обсяг пам'яті (Max Server Memory):** 2147483647 MB

**Опис:** Визначає максимальний обсяг оперативної пам'яті, який може використовувати SQL-сервер.

**Важливість:** Контроль за використанням пам'яті, що дозволяє уникнути витоку ресурсів.

**Максимальна кількість одночасних з'єднань (Max Concurrent Connections):** 0 = unlimited

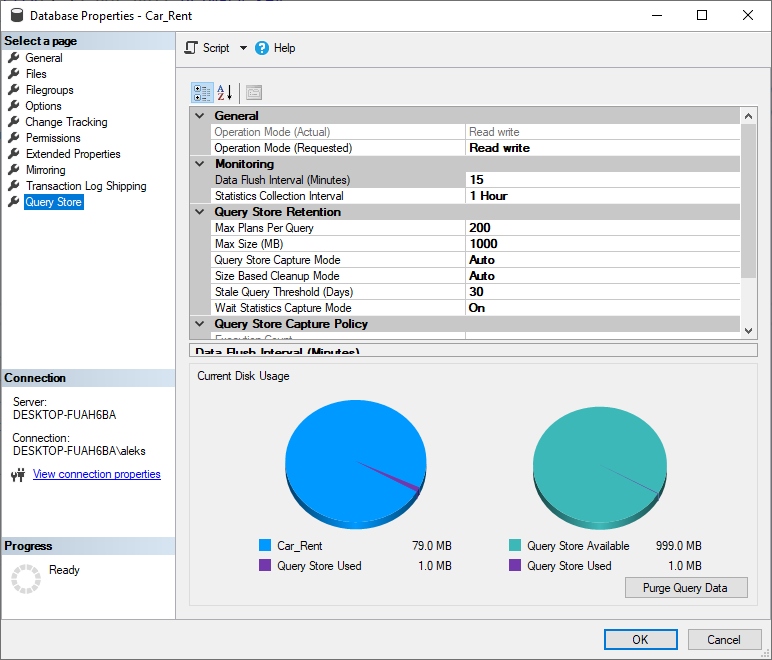
**Опис:** Визначає максимальну кількість одночасних з'єднань, які може обробляти SQL-сервер.

**Важливість:** Забезпечення оптимального розподілу ресурсів та управління одночасними з'єднаннями.

**Автоматичний розподіл пам'яті (Auto Memory Distribution):** yes

**Опис:** Визначає, чи SQL-сервер може автоматично розподіляти пам'ять між різними компонентами.

**Важливість:** Оптимізація використання пам'яті та підтримка продуктивності.

Рис.4.1. Параметри сервера sql

Ці параметри дозволяють забезпечити оптимальну продуктивність, безпеку та надійність роботи SQL-сервера. Налаштування повинні враховувати конкретні вимоги та обсяги роботи системи управління прокатом автомобілів.

# ВИСНОВКИ

Курсова робота на тему «База даних сайту прокату автомобілів» є результатом комплексного дослідження та розробки системи управління прокатом автомобілів. Проект включав в себе створення бази даних за допомогою реляційної СУБД SQL Server, розробку клієнтського та адміністраційного додатку на мові програмування C# з використанням технологій WPF, .NET Framework.

В процесі розробки БД було ретельно вивчено вимоги предметної області, проведено аналіз інформаційних потреб та ризиків. Створено базу даних з урахуванням специфіки прокату автомобілів, включаючи таблиці для автомобілів, клієнтів, бронювань та інші важливі елементи.

Клієнтський додаток був розроблений з функціоналом, спрямованим на зручне та ефективне користування системою прокату. Впроваджено підключення до БД, отримання та відображення даних, а також забезпечено можливість їх зміни з додатку.

Тестування додатку дозволило виявити та виправити помилки, а також здійснити додаткові покращення БД та оптимізацію роботи з нею. Проект успішно відповідає визначеним цілям та завданням, що покладалися перед ним.

У підсумку можна зазначити, що курсова робота успішно вирішує завдання створення та оптимізації бази даних для ефективного управління прокатом автомобілів, а також надає зручний та функціональний інтерфейс для користувачів.

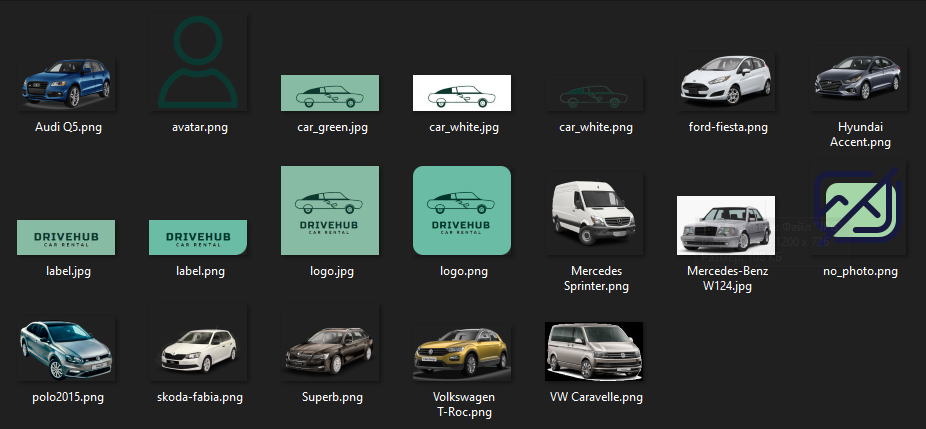
## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jon Skeet – “C# in Depth” 2019. 528 c.
2. Joseph Albahari, Ben Albahari - "C# 9.0 in a Nutshell" 2020. 1064 c.
3. .NET Framework documentation <https://learn.microsoft.com/uk-ua/dotnet/framework/>
4. Sheridan Yuen – “Mastering Windows Presentation Foundation” 2020. 638с.
5. [SQL](https://w3schoolsua.github.io/sql/index.html) Підручник – W3Schools https://w3schoolsua.github.io /sql/index.html#gsc.tab=0
6. Microsoft learn SELECT examples (Transact-SQL) <https://learn.microsoft.com/en-us/sql/t-sql/queries/select-examples-transact-sql?view=sql-server-ver16>
7. Use .NET and C# in Visual Studio to connect to and query a database <https://learn.microsoft.com/en-us/azure/azure-sql/database/connect-query-dotnet-visual-studio?view=azuresql>
8. Create a SQL Server Database programmatically by using ADO.NET and Visual C# .NET <https://learn.microsoft.com/en-us/troubleshoot/developer/visualstudio/csharp/language-compilers/create-sql-server-database-programmatically>
9. Pro C# 8 with .NET Core 3" by Andrew Troelsen
10. SQL Performance Explained" by Markus Winand

## ДОДАТКИ

# Додаток А

Файли зображень



# Додаток Б

Вихідний код:

Клас DataBase:

namespace MyLib

{

public class DataBase

{

SqlConnection sqlConnection = new SqlConnection(@"Data Source=DESKTOP-FUAH6BA;Initial Catalog=Car\_Rent;Integrated Security=True");

public void openConnection()

{

if(sqlConnection.State == System.Data.ConnectionState.Closed)

{

sqlConnection.Open();

}

}

public void closeConnection()

{

if(sqlConnection.State == System.Data.ConnectionState.Open)

{

sqlConnection.Close();

}

}

public SqlConnection getSqlConnection()

{

return sqlConnection;

}

}

}

Клас Car:

namespace MyLib

{

public class Car

{

public int ID\_Avto { get; set; }

public string CarNumber { get; set; }

public string CarBrand { get; set; }

public string CarModel { get; set; }

public string CarColor { get; set; }

public int CarYear { get; set; }

public decimal AverageFuelConsumption { get; set; }

public byte[] PhotoData { get; set; }

public decimal Cost1\_3DayRental { get; set; }

public decimal Cost4\_9DayRental { get; set; }

public decimal Cost10\_25DayRental { get; set; }

public decimal Cost26DayRental { get; set; }

public string CarBrandModel { get; set; }

}

}

Сторінка log\_in

Xaml

<Window x:Class="Course.log\_in"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Course"

mc:Ignorable="d" Icon="Resources/logo.png"

WindowStartupLocation="CenterScreen"

Title="Аутентифікація" Height="540" Width="960">

<Grid>

<Image x:Name="img\_logo" HorizontalAlignment="Center" Width="144" Margin="0,24,0,0" VerticalAlignment="Top" Source="Resources/logo.png"/>

<Label x:Name="label\_authentication" Content="Аутентифікація" HorizontalAlignment="Center" VerticalAlignment="Top" FontFamily="Yu Gothic UI Semibold" FontSize="22" Margin="0,189,0,0" FontWeight="Bold"/>

<Grid Width="400" Height="100" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,257,0,0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="200"/>

<ColumnDefinition Width="200"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

</Grid.RowDefinitions>

<Label x:Name="label\_login" Grid.Row="0" Grid.Column="0" Content="Номер телефону "

FontFamily="Yu Gothic UI Semibold" FontSize="22" VerticalAlignment="Center" HorizontalAlignment="Right" />

<TextBox x:Name="textBox\_login" Grid.Row="0" Grid.Column="1" TextWrapping="Wrap"

FontFamily="Yu Gothic UI Semibold" FontSize="22" Height="35" Background="#CEDED7"/>

<Label x:Name="label\_password" Grid.Row="1" Grid.Column="0" Content="Пароль "

FontFamily="Yu Gothic UI Semibold" FontSize="22" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_password" Grid.Row="1" Grid.Column="1" TextWrapping="Wrap"

FontFamily="Yu Gothic UI Semibold" FontSize="22" Height="35" Background="#CEDED7"/>

</Grid>

<Button x:Name="button\_Log\_in" Content="Вхід" FontFamily="Yu Gothic UI Semibold" FontSize="36" Height="49" Width="200" Background="#87BBA4" HorizontalAlignment="Center"

VerticalAlignment="Bottom" Click="button\_Log\_in\_Click" Cursor="Hand" Margin="0,0,0,60">

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

<TextBlock x:Name="sign\_up\_label" Text="Ще немає акаунту?" FontFamily="Yu Gothic UI Semibold" HorizontalAlignment="Center" VerticalAlignment="Bottom" Foreground="#FF4A4A4A" TextDecorations="Underline" FontSize="18" MouseLeftButtonDown="SignUpLabel\_Click" Cursor="Hand" Margin="0,0,0,20"/>

</Grid>

</Window>

xaml.cs

using MyLib;

using System.Data;

using System.Data.SqlClient;

using System.Windows;

using System.Windows.Input;

namespace Course

{

public partial class log\_in : Window

{

DataBase dataBase = new DataBase();

public log\_in()

{

InitializeComponent();

}

private void button\_Log\_in\_Click(object sender, RoutedEventArgs e)

{

var loginUser = textBox\_login.Text;

var passUser = textBox\_password.Text;

SqlDataAdapter adapter = new SqlDataAdapter();

DataTable table = new DataTable();

dataBase.openConnection();

string querystring = $"select ID\_Klient, phone\_Number, password\_user from Klient where phone\_Number = '{loginUser}' and password\_user = '{passUser}'";

SqlCommand command = new SqlCommand(querystring, dataBase.getSqlConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

object result = command.ExecuteScalar();

if (table.Rows.Count == 1 && loginUser != "0979742402")

{

int idKlient = (int)result;

MessageBox.Show("Ви успішно увійшли!", "Успішно!", MessageBoxButton.OK, MessageBoxImage.Information);

MainWindow MainWindow = new MainWindow(idKlient);

MainWindow.Show();

dataBase.closeConnection();

this.Close();

}

else if(table.Rows.Count == 1 && loginUser == "0979742402")

{

MessageBox.Show("Ви успішно увійшли в акаунт адміністратора!", "Успішно!", MessageBoxButton.OK, MessageBoxImage.Information);

AdminMenu AdminMenu = new AdminMenu();

AdminMenu.Show();

dataBase.closeConnection();

this.Close();

}

else

MessageBox.Show("Такого акаунту не існує або \nне правильний пароль!", "Акаунту не існує!", MessageBoxButton.OK, MessageBoxImage.Error);

}

private void SignUpLabel\_Click(object sender, MouseButtonEventArgs e)

{

sign\_up signUpWindow = new sign\_up();

signUpWindow.Show();

this.Close();

}

}

}

Сторінка sign\_up

xaml

<Window x:Class="Course.sign\_up"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Course"

mc:Ignorable="d"

WindowStartupLocation="CenterScreen"

Title="Реєстрація" Height="540" Width="960">

<Grid>

<Image x:Name="img\_logo" HorizontalAlignment="Center" Width="144" VerticalAlignment="Top" Source="Resources/logo.png" Margin="0,5,0,0">

<Image.OpacityMask>

<ImageBrush ImageSource="Resources/logo.png"/>

</Image.OpacityMask>

</Image>

<Label x:Name="label\_sign\_up" Content="Реєстрація" HorizontalAlignment="Center" VerticalAlignment="Top" FontFamily="Yu Gothic UI Semibold" FontSize="22" Margin="0,140,0,0"/>

<Grid Width="400" Height="245" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,185,0,0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="200"/>

<ColumnDefinition Width="200"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="35"/>

<RowDefinition Height="35"/>

<RowDefinition Height="35"/>

<RowDefinition Height="35"/>

<RowDefinition Height="35"/>

<RowDefinition Height="35"/>

<RowDefinition Height="35"/>

</Grid.RowDefinitions>

<Label x:Name="label\_lastName" Grid.Row="0" Grid.Column="0" Content="Прізвище " FontFamily="Yu Gothic UI Semibold" FontSize="20" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_lastName" Grid.Row="0" Grid.Column="1" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

<Label x:Name="label\_firstName" Grid.Row="1" Grid.Column="0" Content="Ім'я " FontFamily="Yu Gothic UI Semibold" FontSize="20" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_firstName" Grid.Row="1" Grid.Column="1" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

<Label x:Name="label\_surname" Grid.Row="2" Grid.Column="0" Content="По батькові " FontFamily="Yu Gothic UI Semibold" FontSize="20" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_surname" Grid.Row="2" Grid.Column="1" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

<Label x:Name="label\_passportNumber" Grid.Row="3" Grid.Column="0" Content="Номер паспорту " FontFamily="Yu Gothic UI Semibold" FontSize="20" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_passportNumber" Grid.Row="3" Grid.Column="1" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

<Label x:Name="label\_phoneNumber" Grid.Row="4" Grid.Column="0" Content="Номер телефону " FontFamily="Yu Gothic UI Semibold" FontSize="20" VerticalAlignment="Center" HorizontalAlignment="Right" />

<TextBox x:Name="textBox\_phoneNumber" Grid.Row="4" Grid.Column="1" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

<Label x:Name="label\_driverLicense" Grid.Row="5" Grid.Column="0" Content="Номер водійського " FontFamily="Yu Gothic UI Semibold" FontSize="20" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_driverLicense" Grid.Row="5" Grid.Column="1" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

<Label x:Name="label\_password" Grid.Row="6" Grid.Column="0" Content="Пароль " FontFamily="Yu Gothic UI Semibold" FontSize="20" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_password" Grid.Row="6" Grid.Column="1" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

</Grid>

<Button x:Name="button\_Sign\_up" Content="Реєстрація" FontFamily="Yu Gothic UI Semibold" Background="#87BBA4"

FontSize="22" Height="45" Width="200" HorizontalAlignment="Center" Cursor="Hand" Click="button\_Sign\_up\_Click" VerticalAlignment="Bottom" Margin="0,0,0,15" FontWeight="Bold">

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</Window>

xaml.cs

using MyLib;

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows;

using System.Windows.Input;

namespace Course

{

public partial class sign\_up : Window

{

DataBase dataBase = new DataBase();

public sign\_up()

{

InitializeComponent();

}

private void button\_Sign\_up\_Click(object sender, RoutedEventArgs e)

{

var lastName = textBox\_lastName.Text;

var firstName = textBox\_firstName.Text;

var surname = textBox\_surname.Text;

var passportNumber = textBox\_passportNumber.Text;

var phoneNumber = textBox\_phoneNumber.Text;

var driverLicense = textBox\_driverLicense.Text;

var password = textBox\_password.Text;

if (string.IsNullOrWhiteSpace(lastName) ||

string.IsNullOrWhiteSpace(firstName) ||

string.IsNullOrWhiteSpace(surname) ||

string.IsNullOrWhiteSpace(passportNumber) ||

string.IsNullOrWhiteSpace(phoneNumber) ||

string.IsNullOrWhiteSpace(driverLicense) ||

string.IsNullOrWhiteSpace(password))

{

MessageBox.Show("Будь ласка, заповніть всі поля.", "Помилка вводу", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (phoneNumber.Length > 50 || password.Length > 50)

{

MessageBox.Show("Довжина логіну або пароля перевищує 50 символів!", "Помилка вводу", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (checkUser())

{

return;

}

DateTime currentDate = DateTime.Now;

var registration\_Date = currentDate.ToString("yyyy-MM-dd");

string querystring = $"insert into Klient(last\_Name, first\_Name, surname, passport\_Number, phone\_Number, driver\_License\_Number, password\_user, registration\_Date) " +

$"values('{lastName}', '{firstName}', '{surname}', '{passportNumber}', '{phoneNumber}', '{driverLicense}', '{password}', '{registration\_Date}')";

SqlCommand command = new SqlCommand(querystring, dataBase.getSqlConnection());

dataBase.openConnection();

if(command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Ви успішно зареєстровані!", "Успішно!", MessageBoxButton.OK, MessageBoxImage.Information);

log\_in log\_in = new log\_in();

log\_in.Show();

this.Close();

}

else

MessageBox.Show("Не зареєстровано! Перевірте введення!", "Реєстрація не вдалась!", MessageBoxButton.OK, MessageBoxImage.Error);

dataBase.closeConnection();

}

private Boolean checkUser()

{

string loginUser = textBox\_phoneNumber.Text;

string passUser = textBox\_password.Text;

SqlDataAdapter adapter = new SqlDataAdapter();

DataTable table = new DataTable();

string queryString = "SELECT ID\_Klient, phone\_Number, password\_user FROM Klient WHERE phone\_Number = @loginUser AND password\_user = @passUser";

using (SqlCommand command = new SqlCommand(queryString, dataBase.getSqlConnection()))

{

command.Parameters.AddWithValue("@loginUser", loginUser);

command.Parameters.AddWithValue("@passUser", passUser);

adapter.SelectCommand = command;

adapter.Fill(table);

}

if (table.Rows.Count > 0)

{

MessageBox.Show("Користувач вже існує!", "Користувач існує!", MessageBoxButton.OK, MessageBoxImage.Error);

return true;

}

else

{

return false;

}

}

}

}

Сторінка MainWindow

xaml

<Window x:Class="Course.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Course"

Language="uk-ua"

xml:lang="uk-UA"

mc:Ignorable="d" Icon="Resources/logo.png"

WindowStartupLocation="CenterScreen"

Title="Головна сторінка" Height="540" Width="960">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="63"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="84"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Image HorizontalAlignment="Left" Height="63" VerticalAlignment="Top" Width="172" Source="Resources/label.png"/>

<ComboBox x:Name="sortComboBox" VerticalAlignment="Center" HorizontalAlignment="Center"

FontFamily="Yu Gothic UI Semibold" Background="#87BBA4" BorderBrush="#87BBA4"

Width="200" Height="31" SelectionChanged="sortComboBox\_SelectionChanged" Text="За замовчуванням" FontSize="16" Margin="280,0,0,0">

<ComboBoxItem Content="Рік випуску, за зростанням" Tag="car\_Year" />

<ComboBoxItem Content="Рік випуску, за спаданням" Tag="car\_Year DESC" />

<ComboBoxItem Content="Від дешевих до дорогих" Tag="cost\_1\_3\_Day\_Rental" />

<ComboBoxItem Content="Від дорогих до дешевих" Tag="cost\_1\_3\_Day\_Rental DESC" />

<ComboBoxItem Content="Розхід палива, за зростанням" Tag="average\_Fuel\_Consumption" />

<ComboBoxItem Content="Розхід палива, за спаданням" Tag="average\_Fuel\_Consumption DESC" />

</ComboBox>

<Grid Height="30" Width="200" VerticalAlignment="Center" HorizontalAlignment="Right" Margin="0,0,15,0" >

<TextBox x:Name="searchTextBox" TextChanged="searchTextBox\_TextChanged" VerticalContentAlignment="Center"

FontFamily="Yu Gothic UI Semibold" FontSize="16" Height="30" Width="200"/>

<Image Source="Resources/loupe.png" Height="25" Width="25" VerticalAlignment="Center" HorizontalAlignment="Right" Margin="0,0,3,0"></Image>

</Grid>

</Grid>

<Grid Grid.Row="1">

<Grid.Resources>

<Style x:Key="ListBoxItemStyle" TargetType="ListBoxItem">

<Setter Property="Background" Value="Transparent"/>

<Setter Property="BorderBrush" Value="Transparent"/>

<Setter Property="BorderThickness" Value="0"/>

<Setter Property="Padding" Value="0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="ListBoxItem">

<ContentPresenter />

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Grid.Resources>

<Grid Height="333" Width="315" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="20 0 0 0">

<Border CornerRadius="15" Background="#CEDED7" BorderBrush="#CEDED7" BorderThickness="1">

<Grid Background="Transparent">

<Image Source="Resources/findAvto.png"/>

<TextBox Margin="10" Height="30" VerticalAlignment="Top" Background="Transparent" HorizontalAlignment="Center" Text="Оберіть дату прокату" FontSize="20" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<TextBox Margin="10,45,10,10" Background="Transparent" VerticalAlignment="Top" HorizontalAlignment="Center" Text="Дата подачі" FontSize="18" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<DatePicker x:Name="Start\_Date\_DatePicker" Height="35" Width="200" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,76,0,0"

FontFamily="Yu Gothic UI Semibold" FontSize="20" Language="uk-UA" FirstDayOfWeek="Monday" BorderBrush="Transparent">

<DatePicker.Resources>

<Style TargetType="DatePickerTextBox">

<Setter Property="Control.Template">

<Setter.Value>

<ControlTemplate>

<TextBox x:Name="PART\_TextBox"

Text="{Binding Path=SelectedDate, StringFormat='dd.MM.yyyy', RelativeSource={RelativeSource AncestorType={x:Type DatePicker}}}"

Focusable="False"

HorizontalContentAlignment="Center"

VerticalContentAlignment="Center" />

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</DatePicker.Resources>

<DatePicker.Background>

<ImageBrush/>

</DatePicker.Background>

</DatePicker>

<TextBox VerticalAlignment="Center" Background="Transparent" HorizontalAlignment="Center" Text="Дата повернення" FontSize="18" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<DatePicker x:Name="end\_Of\_Rental\_DatePicker" Height="35" Width="200" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="0,76,0,0"

FontFamily="Yu Gothic UI Semibold" FontSize="20" Language="uk-UA" FirstDayOfWeek="Monday" BorderBrush="Transparent">

<DatePicker.Resources>

<Style TargetType="DatePickerTextBox">

<Setter Property="Control.Template">

<Setter.Value>

<ControlTemplate>

<TextBox x:Name="PART\_TextBox"

Text="{Binding Path=SelectedDate, StringFormat='dd.MM.yyyy', RelativeSource={RelativeSource AncestorType={x:Type DatePicker}}}"

Focusable="False"

HorizontalContentAlignment="Center"

VerticalContentAlignment="Center" />

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</DatePicker.Resources>

</DatePicker>

<Button x:Name="button\_FindAvto" Content="Знайти авто" FontFamily="Yu Gothic UI Semibold" Background="#5F7E70"

FontSize="22" Height="45" Width="200" HorizontalAlignment="Center" Cursor="Hand" VerticalAlignment="Bottom" Margin="0,0,0,20" FontWeight="Bold" Click="button\_FindAvto\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</Border>

</Grid>

<ListBox x:Name="carListBox" Width="590" HorizontalContentAlignment="Stretch" HorizontalAlignment="Right" ItemContainerStyle="{StaticResource ListBoxItemStyle}">

<ListBox.ItemTemplate>

<DataTemplate>

<Grid Height="240" Width="550" HorizontalAlignment="Right" Margin="10">

<Border CornerRadius="15" BorderBrush="Black" BorderThickness="1">

<Grid>

<TextBox Text="{Binding Path=AverageFuelConsumption, StringFormat='бензин / {0:N1}л'}" IsReadOnly="True"

Height="22" Width="auto" FontSize="16" FontFamily="Yu Gothic UI Semibold" Margin="10"

BorderBrush="Transparent" SelectionBrush="{x:Null}" HorizontalAlignment="Center" VerticalAlignment="Top" IsHitTestVisible="False"/>

<Grid Height="240" Width="200" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="5">

<Grid.RowDefinitions>

<RowDefinition Height="40"/>

<RowDefinition Height="200"/>

</Grid.RowDefinitions>

<TextBox Grid.Row="0" Text="{Binding CarBrandModel}" IsReadOnly="True" Height="35

" Width="auto" FontSize="20"

FontWeight="Bold" FontFamily="Yu Gothic UI Semibold" HorizontalAlignment="Center" VerticalAlignment="Center"

BorderBrush="Transparent" SelectionBrush="{x:Null}" IsHitTestVisible="False" />

<Image Grid.Row="1" Source="{Binding PhotoData}" HorizontalAlignment="Center" VerticalAlignment="Center"></Image>

</Grid>

<Grid Height="100" Width="330" HorizontalAlignment="Right" VerticalAlignment="Center">

<Grid.RowDefinitions>

<RowDefinition Height="25"></RowDefinition>

<RowDefinition Height="25"></RowDefinition>

<RowDefinition Height="25"></RowDefinition>

<RowDefinition Height="25"></RowDefinition>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="60"></ColumnDefinition>

<ColumnDefinition Width="120"></ColumnDefinition>

<ColumnDefinition Width="75"></ColumnDefinition>

<ColumnDefinition Width="75"></ColumnDefinition>

</Grid.ColumnDefinitions>

<TextBox Grid.Column="0" Grid.Row="0" Text="Номер" FontSize="16" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<TextBox Text="{Binding CarNumber}" Grid.Column="1" Grid.Row="0" IsReadOnly="True" FontSize="16" FontFamily="Yu Gothic UI Semibold" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<TextBox Grid.Column="0" Grid.Row="1" Text="Колір" FontSize="16" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<TextBox Text="{Binding CarColor}" Grid.Column="1" Grid.Row="1" IsReadOnly="True" FontSize="16" FontFamily="Yu Gothic UI Semibold" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<TextBox Grid.Column="0" Grid.Row="2" Text="Рік" FontSize="16" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<TextBox Text="{Binding CarYear}" Grid.Column="1" Grid.Row="2" IsReadOnly="True" FontSize="16" FontFamily="Yu Gothic UI Semibold" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<TextBox Grid.Column="2" Grid.Row="0" Text="1-3 діб" FontSize="16" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<TextBox Text="{Binding Cost1\_3DayRental, StringFormat=${0:N1}}" Grid.Column="3" Grid.Row="0" IsReadOnly="True" FontSize="16" FontFamily="Yu Gothic UI Semibold" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<TextBox Grid.Column="2" Grid.Row="1" Text="4-9 діб" FontSize="16" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<TextBox Text="{Binding Cost4\_9DayRental, StringFormat=${0:N1}}" Grid.Column="3" Grid.Row="1" IsReadOnly="True" FontSize="16" FontFamily="Yu Gothic UI Semibold" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<TextBox Grid.Column="2" Grid.Row="2" Text="10-25 діб" FontSize="16" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<TextBox Text="{Binding Cost10\_25DayRental, StringFormat=${0:N1}}" Grid.Column="3" Grid.Row="2" IsReadOnly="True" FontSize="16" FontFamily="Yu Gothic UI Semibold" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<TextBox Grid.Column="2" Grid.Row="3" Text="26+ діб" FontSize="16" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<TextBox Text="{Binding Cost26DayRental, StringFormat=${0:N1}}" Grid.Column="3" Grid.Row="3" IsReadOnly="True" FontSize="16" FontFamily="Yu Gothic UI Semibold" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

</Grid>

<Button x:Name="button\_ChooseCar" Content="Обрати авто" FontFamily="Yu Gothic UI Semibold" Background="#5F7E70"

FontSize="20" Height="45" Width="145" Cursor="Hand" FontWeight="Bold" Click="button\_ChooseCar\_Click"

VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="0, 0, 15, 10" Tag="{Binding ID\_Avto}">

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</Border>

</Grid>

</DataTemplate>

</ListBox.ItemTemplate>

<ListBox.Template>

<ControlTemplate TargetType="ListBox">

<Border>

<ScrollViewer VerticalScrollBarVisibility="Auto">

<ItemsPresenter />

</ScrollViewer>

</Border>

</ControlTemplate>

</ListBox.Template>

</ListBox>

</Grid>

<Grid x:Name="bottom\_menu" Width="auto" Height="84" VerticalAlignment="Bottom" Background="#CEDED7" Grid.Row="2">

<Button x:Name="MainWindow\_Button" Height="59" Width="140" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="30,0,0,0">

<Button.Background>

<ImageBrush>

<ImageBrush.RelativeTransform>

<TransformGroup>

<ScaleTransform CenterY="0.5" CenterX="0.5"/>

<SkewTransform CenterX="0.5" CenterY="0.5"/>

<RotateTransform CenterX="0.5" CenterY="0.5"/>

<TranslateTransform Y="-0.05"/>

</TransformGroup>

</ImageBrush.RelativeTransform>

<ImageBrush.ImageSource>Resources/car\_white.png</ImageBrush.ImageSource>

</ImageBrush>

</Button.Background>

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="Profile\_Button" Height="59" Width="140" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="0,0,30,0" Click="Profile\_Button\_Click">

<Button.Background >

<ImageBrush ImageSource="Resources/avatar.png">

<ImageBrush.RelativeTransform>

<TransformGroup>

<ScaleTransform CenterY="0.5" CenterX="0.5" ScaleX="0.3" ScaleY="0.75"/>

<SkewTransform CenterX="0.5" CenterY="0.5"/>

<RotateTransform CenterX="0.5" CenterY="0.5"/>

<TranslateTransform/>

</TransformGroup>

</ImageBrush.RelativeTransform>

</ImageBrush>

</Button.Background>

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</Grid>

</Window>

xaml.cs

using MyLib;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Data.SqlClient;

using System.Windows;

using System.Windows.Controls;

namespace Course

{

public partial class MainWindow : Window

{

private ObservableCollection<Car> cars;

DataBase dataBase = new DataBase();

private int \_idKlient;

private DateTime \_startDate;

private DateTime \_endDate;

public MainWindow(int idKlient)

{

InitializeComponent();

\_idKlient = idKlient;

cars = new ObservableCollection<Car>();

carListBox.ItemsSource = cars;

UpdateCarsCollection("SELECT \* FROM Avto");

DateTime currentDate = DateTime.Now;

DateTime nextDay = currentDate.AddDays(1);

Start\_Date\_DatePicker.DisplayDateStart = currentDate;

end\_Of\_Rental\_DatePicker.DisplayDateStart = nextDay;

\_startDate = currentDate;

\_endDate = nextDay;

}

private void sortComboBox\_SelectionChanged(object sender, System.Windows.Controls.SelectionChangedEventArgs e)

{

ComboBoxItem selectedItem = (ComboBoxItem)sortComboBox.SelectedItem;

if (selectedItem != null)

{

string orderBy = selectedItem.Tag.ToString();

string queryString = $"SELECT \* FROM Avto ORDER BY {orderBy}";

UpdateCarsCollection(queryString);

}

}

private void Search()

{

string queryString = $"SELECT \* FROM Avto WHERE concat (ID\_Avto, car\_Number, car\_Brand, car\_Model, car\_Color, car\_Year, average\_Fuel\_Consumption) like '%" + searchTextBox.Text + "%'";

UpdateCarsCollection(queryString);

}

private void searchTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

Search();

}

private void UpdateCarsCollection(string queryString)

{

cars.Clear();

foreach (var car in GetCarData(queryString))

{

cars.Add(car);

}

}

private void button\_FindAvto\_Click(object sender, RoutedEventArgs e)

{

\_startDate = (Start\_Date\_DatePicker.SelectedDate ?? DateTime.MinValue).Date;

var startDate = \_startDate.ToString("yyyy-MM-dd");

\_endDate = (end\_Of\_Rental\_DatePicker.SelectedDate ?? DateTime.MinValue).Date;

var endDate = \_endDate.ToString("yyyy-MM-dd");

if (\_startDate < \_endDate)

{

string queryString = $"SELECT Avto.\* FROM Avto LEFT JOIN Rent ON Avto.ID\_Avto = Rent.ID\_Avto WHERE Rent.ID\_Renta IS NULL OR (Rent.end\_Of\_Rental < '{startDate}' OR Rent.rental\_Start\_Date > '{endDate}');";

UpdateCarsCollection(queryString);

}

else

{

MessageBox.Show("Дата початку повинна бути менше за дату завершення.", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private List<Car> GetCarData(string queryString)

{

try

{

dataBase.openConnection();

using (SqlCommand command = new SqlCommand(queryString, dataBase.getSqlConnection()))

{

using (SqlDataReader reader = command.ExecuteReader())

{

List<Car> cars = new List<Car>();

while (reader.Read())

{

string carBrand = reader["car\_Brand"].ToString();

string carModel = reader["car\_Model"].ToString();

Car car = new Car

{

ID\_Avto = (int)reader["ID\_Avto"],

CarNumber = reader["car\_Number"].ToString(),

CarBrandModel = $"{carBrand} {carModel}",

CarColor = reader["car\_Color"].ToString(),

CarYear = (int)reader["car\_Year"],

AverageFuelConsumption = (decimal)reader["average\_Fuel\_Consumption"],

PhotoData = reader["PhotoData"] as byte[],

Cost1\_3DayRental = (decimal)reader["cost\_1\_3\_Day\_Rental"],

Cost4\_9DayRental = (decimal)reader["cost\_4\_9\_Day\_Rental"],

Cost10\_25DayRental = (decimal)reader["cost\_10\_25\_Day\_Rental"],

Cost26DayRental = (decimal)reader["cost\_26\_Day\_Rental"]

};

cars.Add(car);

}

return cars;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при отриманні даних про авто: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

return null;

}

finally

{

dataBase.closeConnection();

}

}

private void Profile\_Button\_Click(object sender, RoutedEventArgs e)

{

Profile profile = new Profile(\_idKlient);

profile.Show();

this.Close();

}

private void button\_ChooseCar\_Click(object sender, RoutedEventArgs e)

{

Button clickedButton = sender as Button;

if (clickedButton != null)

{

if (\_startDate < \_endDate)

{

int idAvto = (int)clickedButton.Tag;

ChooseCarWindow childWindow = new ChooseCarWindow(\_startDate, \_endDate, idAvto, \_idKlient);

childWindow.Owner = this;

childWindow.ShowDialog();

}

else

{

MessageBox.Show("Дата початку повинна бути менше за дату завершення.", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

}

Сторінка ChooseCarWindow

xaml

<Window x:Class="Course.ChooseCarWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Course" WindowStyle="None"

mc:Ignorable="d" WindowStartupLocation="CenterOwner" FontFamily="Yu Gothic UI Semibold"

Title="Підтвердження вибору авто" MaxHeight="400" MaxWidth="300" MinWidth="300" MinHeight="400" FontSize="18">

<Grid>

<TextBox VerticalAlignment="Top" HorizontalAlignment="Center" Text="Дата подачі" FontSize="18" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False" Margin="0,10,0,0"/>

<DatePicker x:Name="Start\_Date\_DatePicker" Height="35" Width="200" HorizontalAlignment="Center" IsHitTestVisible="False"

VerticalAlignment="Top" FontFamily="Yu Gothic UI Semibold" FontSize="20" Language="uk-UA" Margin="0,41,0,0"/>

<TextBox VerticalAlignment="Top" HorizontalAlignment="Center" Text="Дата повернення" FontSize="18" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False" Margin="0,81,0,0"/>

<DatePicker x:Name="end\_Of\_Rental\_DatePicker" Height="35" Width="200" HorizontalAlignment="Center" IsHitTestVisible="False"

VerticalAlignment="Top" FontFamily="Yu Gothic UI Semibold" FontSize="20" Language="uk-UA" Margin="0,121,0,0"/>

<Button x:Name="button\_Close" Content="Відміна" FontFamily="Yu Gothic UI Semibold" Background="#5F7E70"

FontSize="16" Height="35" Width="110" HorizontalAlignment="Left" Cursor="Hand" VerticalAlignment="Bottom" Margin="10,0,0,20" FontWeight="Bold" Click="button\_Close\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="15">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

<TextBox x:Name="cost\_TextBox" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="18"

FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<Button x:Name="button\_Submit" Content="Підтвердити" FontFamily="Yu Gothic UI Semibold" Background="#5F7E70"

FontSize="16" Height="35" Width="110" HorizontalAlignment="Right" Cursor="Hand" VerticalAlignment="Bottom" Margin="0,0,10,20" FontWeight="Bold" Click="button\_Submit\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="15">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</Window>

xaml.cs

using MyLib;

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows;

namespace Course

{

public partial class ChooseCarWindow : Window

{

DataBase dataBase = new DataBase();

private DateTime \_startDate;

private DateTime \_endDate;

private int \_idKlient;

private int \_idAvto;

private decimal \_cost;

private int \_percentDiscount;

private decimal \_cost1\_3DayRental;

private decimal \_cost4\_9DayRental;

private decimal \_cost10\_25DayRental;

private decimal \_cost26DayRental;

public ChooseCarWindow(DateTime startDate, DateTime endDate, int ID\_Avto, int idKlient)

{

InitializeComponent();

\_idKlient = idKlient;

\_idAvto = ID\_Avto;

\_startDate = startDate;

\_endDate = endDate;

Start\_Date\_DatePicker.SelectedDate = \_startDate;

end\_Of\_Rental\_DatePicker.SelectedDate = \_endDate;

GetDiscount();

СostСalculation();

}

private void СostСalculation()

{

try

{

dataBase.openConnection();

string queryString = @"

SELECT cost\_1\_3\_Day\_Rental, cost\_4\_9\_Day\_Rental, cost\_10\_25\_Day\_Rental, cost\_26\_Day\_Rental

FROM Avto

WHERE ID\_Avto = @\_idAvto

";

using (SqlCommand command = new SqlCommand(queryString, dataBase.getSqlConnection()))

{

command.Parameters.AddWithValue("@\_idAvto", \_idAvto);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

\_cost1\_3DayRental = (decimal)reader["cost\_1\_3\_Day\_Rental"];

\_cost4\_9DayRental = (decimal)reader["cost\_4\_9\_Day\_Rental"];

\_cost10\_25DayRental = (decimal)reader["cost\_10\_25\_Day\_Rental"];

\_cost26DayRental = (decimal)reader["cost\_26\_Day\_Rental"];

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при отриманні даних про ціни авто: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

dataBase.closeConnection();

}

TimeSpan difference = \_endDate - \_startDate;

int numberOfDays = difference.Days;

decimal percent = \_percentDiscount;

decimal priceWithoutDiscount;

if (numberOfDays >= 1 && numberOfDays <= 3)

{

priceWithoutDiscount = \_cost1\_3DayRental \* numberOfDays;

\_cost = priceWithoutDiscount - priceWithoutDiscount \* (percent / 100);

}

else if (numberOfDays >= 4 && numberOfDays <= 9)

{

priceWithoutDiscount = \_cost4\_9DayRental \* numberOfDays;

\_cost = priceWithoutDiscount - priceWithoutDiscount \* (percent / 100);

}

else if (numberOfDays >= 10 && numberOfDays <= 25)

{

priceWithoutDiscount = \_cost10\_25DayRental \* numberOfDays;

\_cost = priceWithoutDiscount - priceWithoutDiscount \* (percent / 100);

}

else

{

priceWithoutDiscount = \_cost26DayRental \* numberOfDays;

\_cost = priceWithoutDiscount - priceWithoutDiscount \* (percent / 100);

}

cost\_TextBox.Text = $"Вартість: ${\_cost.ToString("0.00")}";

}

private void button\_Close\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void GetDiscount()

{

try

{

dataBase.openConnection();

string queryString = "SELECT percent\_Discount FROM Discount WHERE ID\_Klient = @\_idKlient";

using (SqlCommand command = new SqlCommand(queryString, dataBase.getSqlConnection()))

{

command.Parameters.AddWithValue("@\_idKlient", \_idKlient);

using (SqlDataReader reader = command.ExecuteReader())

{

int count = 0;

while (reader.Read())

{

int currentDiscount = reader.GetInt32(0);

if (count == 0)

{

\_percentDiscount = currentDiscount;

}

else

{

\_percentDiscount += currentDiscount / 3;

}

count++;

}

if (count <= 0)

{

MessageBox.Show($"Для клієнта з ID {\_idKlient} не знайдено знижок.");

}

}

}

}

catch (Exception ex)

{

HandleException("Помилка при отриманні даних про ціни авто", ex);

}

finally

{

dataBase.closeConnection();

}

}

private void button\_Submit\_Click(object sender, RoutedEventArgs e)

{

if (checkRent())

{

return;

}

try

{

var startDate = \_startDate.ToString("yyyy-MM-dd");

var endDate = \_endDate.ToString("yyyy-MM-dd");

string queryString = @"

INSERT INTO Rent (rental\_Start\_Date, end\_Of\_Rental, actual\_End\_Of\_Rental, rental\_Price, ID\_Klient, ID\_Avto)

VALUES (@startDate, @endDate, ' ', @cost, @idKlient, @idAvto)

";

SqlCommand command = new SqlCommand(queryString, dataBase.getSqlConnection());

dataBase.openConnection();

command.Parameters.AddWithValue("@startDate", startDate);

command.Parameters.AddWithValue("@endDate", endDate);

command.Parameters.AddWithValue("@cost", \_cost);

command.Parameters.AddWithValue("@idKlient", \_idKlient);

command.Parameters.AddWithValue("@idAvto", \_idAvto);

if (command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Ви успішно забронювали авто!", "Успішно!", MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

MessageBox.Show("Не вдалося забронювати авто! Спробуйте ще раз!", "Бронювання не вдалось!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

catch (Exception ex)

{

HandleException("Помилка при бронюванні авто", ex);

}

finally

{

dataBase.closeConnection();

}

}

private Boolean checkRent()

{

var startDate = \_startDate.ToString("yyyy-MM-dd");

var endDate = \_endDate.ToString("yyyy-MM-dd");

SqlDataAdapter adapter = new SqlDataAdapter();

DataTable table = new DataTable();

string queryString = @"

SELECT 1

FROM Rent

WHERE ID\_Avto = @idAvto AND

NOT (end\_Of\_Rental < @startDate OR rental\_Start\_Date > @endDate);";

SqlCommand command = new SqlCommand(queryString, dataBase.getSqlConnection());

{

command.Parameters.AddWithValue("@startDate", startDate);

command.Parameters.AddWithValue("@endDate", endDate);

command.Parameters.AddWithValue("@idAvto", \_idAvto);

adapter.SelectCommand = command;

adapter.Fill(table);

}

if (table.Rows.Count > 0)

{

MessageBox.Show("Машина зайнята на ці числа!", "Заброньвано!", MessageBoxButton.OK, MessageBoxImage.Error);

return true;

}

else

{

return false;

}

}

private void HandleException(string errorMessage, Exception ex)

{

MessageBox.Show($"{errorMessage}: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

Сторінка Profile

xaml

<Window x:Class="Course.Profile"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Course" WindowStartupLocation="CenterScreen"

mc:Ignorable="d" Icon="Resources/logo.png"

Title="Profile" Height="540" Width="960">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="63"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="84"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0" Background="Transparent">

<Image HorizontalAlignment="Left" Height="63" VerticalAlignment="Top" Width="172" Source="Resources/label.png"/>

</Grid>

<Grid Grid.Row="1">

<Grid Width="320" Height="320" Margin="15" HorizontalAlignment="Left">

<Border Background="#CEDED7" CornerRadius="15" BorderBrush="#CEDED7" BorderThickness="1">

<Grid>

<Grid.Resources>

<Style x:Key="ListBoxItemStyle" TargetType="ListBoxItem">

<Setter Property="Background" Value="#CEDED7"/>

<Setter Property="BorderBrush" Value="Black"/>

<Setter Property="BorderThickness" Value="1"/>

<Setter Property="Padding" Value="0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="ListBoxItem">

<ContentPresenter />

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Grid.Resources>

<TextBox Text="Історія бронювань" Background="#CEDED7" Height="30" HorizontalAlignment="Center" VerticalAlignment="Top" FontSize="20" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False"/>

<ListBox x:Name="orderListBox" Width="320" Height="300" HorizontalContentAlignment="Stretch" HorizontalAlignment="Center" VerticalAlignment="Bottom" ItemContainerStyle="{StaticResource ListBoxItemStyle}">

<ListBox.ItemTemplate>

<DataTemplate>

<Grid Height="auto" Width="280" HorizontalAlignment="Center" Margin="10">

<Border Background="#CEDED7" CornerRadius="15" BorderBrush="Black" BorderThickness="1">

<Grid>

<StackPanel Margin="10">

<TextBlock Text="{Binding Path=StartDate, StringFormat='Початок оренди: {0:dd.MM.yyyy}'}" FontSize="14" FontFamily="Yu Gothic UI Semibold"/>

<TextBlock Text="{Binding Path=EndDate, StringFormat='Кінець оренди: {0:dd.MM.yyyy}'}" FontSize="14" FontFamily="Yu Gothic UI Semibold"/>

<TextBlock Text="{Binding Path=ActualEndDate, StringFormat='Формальний кінець оренди: {0:dd.MM.yyyy HH:mm}'}" TextWrapping="Wrap" FontSize="14" FontFamily="Yu Gothic UI Semibold"/>

</StackPanel>

</Grid>

</Border>

</Grid>

</DataTemplate>

</ListBox.ItemTemplate>

<ListBox.Template>

<ControlTemplate TargetType="ListBox">

<Border>

<ScrollViewer VerticalScrollBarVisibility="Auto">

<ItemsPresenter />

</ScrollViewer>

</Border>

</ControlTemplate>

</ListBox.Template>

</ListBox>

</Grid>

</Border>

</Grid>

</Grid>

<Grid Width="300" Height="420" HorizontalAlignment="Right" VerticalAlignment="Top" Panel.ZIndex="2" Margin="0,-5,68,0" Grid.RowSpan="2">

<Grid.RowDefinitions>

<RowDefinition Height="35"></RowDefinition>

<RowDefinition Height="35"></RowDefinition>

<RowDefinition Height="35"></RowDefinition>

<RowDefinition Height="35"></RowDefinition>

<RowDefinition Height="35"></RowDefinition>

<RowDefinition Height="105"></RowDefinition>

<RowDefinition Height="35"></RowDefinition>

<RowDefinition Height="105"></RowDefinition>

</Grid.RowDefinitions>

<TextBox Grid.Row="0" Text="ПІБ" FontSize="18" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False" VerticalAlignment="Bottom"/>

<TextBox x:Name="full\_nameTextBox" Background="#CEDED7" Grid.Row="1" IsReadOnly="True" FontSize="18" FontFamily="Yu Gothic UI Semibold" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False" Height="30"/>

<TextBox Grid.Row="2" Text="Логін" FontSize="18" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False" VerticalAlignment="Bottom"/>

<TextBox x:Name="loginTextBox" Background="#CEDED7" Grid.Row="3" IsReadOnly="True" FontSize="18" FontFamily="Yu Gothic UI Semibold" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False" Height="30"/>

<TextBox Grid.Row="4" Text="Штрафи" FontSize="18" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False" VerticalAlignment="Bottom"/>

<Grid Width="300" Height="100" Grid.Row="5" VerticalAlignment="Top" Margin=" 0 5 0 0" HorizontalAlignment="Center">

<Border Background="#CEDED7" CornerRadius="15" BorderBrush="#CEDED7" BorderThickness="1">

<Grid>

<Grid.Resources>

<Style x:Key="ListBoxItemStyle" TargetType="ListBoxItem">

<Setter Property="Background" Value="#CEDED7"/>

<Setter Property="BorderBrush" Value="Black"/>

<Setter Property="BorderThickness" Value="1"/>

<Setter Property="Padding" Value="0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="ListBoxItem">

<ContentPresenter />

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Grid.Resources>

<ListBox x:Name="finesListBox" HorizontalContentAlignment="Stretch" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0 10 0 10" ItemContainerStyle="{StaticResource ListBoxItemStyle}">

<ListBox.ItemTemplate>

<DataTemplate>

<Grid Height="auto" Width="270" HorizontalAlignment="Center" Margin="0 0 0 10">

<Border Background="#CEDED7" CornerRadius="15" BorderBrush="Black" BorderThickness="1">

<Grid>

<StackPanel Margin="10">

<TextBlock Text="{Binding Path=NameFines}" FontSize="14" FontFamily="Yu Gothic UI Semibold"/>

<TextBlock FontSize="14" FontFamily="Yu Gothic UI Semibold">

<Run Text="{Binding Path=SumFines }" />

<Run Text="грн."/>

</TextBlock>

</StackPanel>

</Grid>

</Border>

</Grid>

</DataTemplate>

</ListBox.ItemTemplate>

<ListBox.Template>

<ControlTemplate TargetType="ListBox">

<Border>

<ScrollViewer VerticalScrollBarVisibility="Auto">

<ItemsPresenter />

</ScrollViewer>

</Border>

</ControlTemplate>

</ListBox.Template>

</ListBox>

</Grid>

</Border>

</Grid>

<TextBox Grid.Row="6" Text="Персональні акції користувача" FontSize="18" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" BorderBrush="Transparent" SelectionBrush="Transparent" IsHitTestVisible="False" VerticalAlignment="Bottom"/>

<Grid Width="300" Height="100" Grid.Row="7" VerticalAlignment="Top" Margin=" 0 5 0 0" HorizontalAlignment="Center">

<Border Background="#CEDED7" CornerRadius="15" BorderBrush="#CEDED7" BorderThickness="1">

<Grid>

<Grid.Resources>

<Style x:Key="ListBoxItemStyle" TargetType="ListBoxItem">

<Setter Property="Background" Value="#CEDED7"/>

<Setter Property="BorderBrush" Value="Black"/>

<Setter Property="BorderThickness" Value="1"/>

<Setter Property="Padding" Value="0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="ListBoxItem">

<ContentPresenter />

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Grid.Resources>

<ListBox x:Name="discountListBox" HorizontalContentAlignment="Stretch" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0 10 0 10" ItemContainerStyle="{StaticResource ListBoxItemStyle}">

<ListBox.ItemTemplate>

<DataTemplate>

<Grid Height="auto" Width="270" HorizontalAlignment="Center" Margin="0 0 0 10">

<Border Background="#CEDED7" CornerRadius="15" BorderBrush="Black" BorderThickness="1">

<Grid>

<StackPanel Margin="10">

<TextBlock Text="{Binding Path=NameDiscount}" FontSize="14" FontFamily="Yu Gothic UI Semibold"/>

<TextBlock FontSize="14" FontFamily="Yu Gothic UI Semibold">

<Run Text="{Binding Path=PercentDiscount}" />

<Run Text="%"/>

</TextBlock>

</StackPanel>

</Grid>

</Border>

</Grid>

</DataTemplate>

</ListBox.ItemTemplate>

<ListBox.Template>

<ControlTemplate TargetType="ListBox">

<Border>

<ScrollViewer VerticalScrollBarVisibility="Auto">

<ItemsPresenter />

</ScrollViewer>

</Border>

</ControlTemplate>

</ListBox.Template>

</ListBox>

</Grid>

</Border>

</Grid>

</Grid>

<Grid Grid.Row="2" x:Name="bottom\_menu" Width="960" Height="84" VerticalAlignment="Bottom" Background="#CEDED7">

<Button x:Name="MainWindow\_Button" Height="59" Width="140" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="30,0,0,0" Click="MainWindow\_Button\_Click">

<Button.Background>

<ImageBrush>

<ImageBrush.RelativeTransform>

<TransformGroup>

<ScaleTransform CenterY="0.5" CenterX="0.5"/>

<SkewTransform CenterX="0.5" CenterY="0.5"/>

<RotateTransform CenterX="0.5" CenterY="0.5"/>

<TranslateTransform Y="-0.05"/>

</TransformGroup>

</ImageBrush.RelativeTransform>

<ImageBrush.ImageSource>Resources/car\_white.png</ImageBrush.ImageSource>

</ImageBrush>

</Button.Background>

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="button\_exit" Height="59" Width="140" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="0,0,30,0" Click="button\_exit\_Click">

<Button.Background>

<ImageBrush ImageSource="Resources/exit\_icon.png">

<ImageBrush.RelativeTransform>

<TransformGroup>

<ScaleTransform CenterY="0.5" CenterX="0.5" ScaleX="0.3" ScaleY="0.75"/>

<SkewTransform CenterX="0.5" CenterY="0.5"/>

<RotateTransform CenterX="0.5" CenterY="0.5"/>

<TranslateTransform/>

</TransformGroup>

</ImageBrush.RelativeTransform>

</ImageBrush>

</Button.Background>

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</Grid>

</Window>

xaml.cs

using MyLib;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Globalization;

using System.Windows;

namespace Course

{

public partial class Profile : Window

{

private int \_idKlient;

DataBase dataBase = new DataBase();

public class BookingHistoryItem

{

public DateTime StartDate { get; set; }

public DateTime EndDate { get; set; }

public string ActualEndDate { get; set; }

}

public class FinesHistoryItem

{

public string NameFines { get; set; }

public string SumFines { get; set; }

}

public class DiscountHistoryItem

{

public string NameDiscount { get; set; }

public string PercentDiscount { get; set; }

}

public Profile(int idKlient)

{

InitializeComponent();

\_idKlient = idKlient;

GetProfile();

LoadBookingHistory(\_idKlient);

LoadFinesHistory(\_idKlient);

LoadDiscountHistory(\_idKlient);

}

private void GetProfile()

{

try

{

dataBase.openConnection();

string queryString = "SELECT \* FROM Klient WHERE ID\_Klient = @\_idKlient";

using (SqlCommand command = new SqlCommand(queryString, dataBase.getSqlConnection()))

{

command.Parameters.AddWithValue("@\_idKlient", \_idKlient);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

string last\_Name = reader["last\_Name"].ToString();

string first\_Name = reader["first\_Name"].ToString();

string surname = reader["surname"].ToString();

full\_nameTextBox.Text = $"{last\_Name} {first\_Name} {surname}";

loginTextBox.Text = reader["phone\_Number"].ToString();

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при отриманні даних про клієнта: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

dataBase.closeConnection();

}

}

private void LoadBookingHistory(int idKlient)

{

try

{

dataBase.openConnection();

string queryString = @"

SELECT rental\_Start\_Date, end\_Of\_Rental, actual\_End\_Of\_Rental

FROM Rent

WHERE ID\_Klient = @idKlient

";

using (SqlCommand command = new SqlCommand(queryString, dataBase.getSqlConnection()))

{

command.Parameters.AddWithValue("@idKlient", idKlient);

using (SqlDataReader reader = command.ExecuteReader())

{

List<BookingHistoryItem> bookingHistory = new List<BookingHistoryItem>();

while (reader.Read())

{

DateTime rentalStartDate = (DateTime)reader["rental\_Start\_Date"];

DateTime endOfRental = (DateTime)reader["end\_Of\_Rental"];

DateTime actualEndOfRental = (DateTime)reader["actual\_End\_Of\_Rental"];

string actualEndDateText = (actualEndOfRental == DateTime.ParseExact("1900-01-01 00:00:00.000", "yyyy-MM-dd HH:mm:ss.fff", CultureInfo.InvariantCulture))

? "Авто ще не повернули"

: actualEndOfRental.ToString("dd.MM.yyyy HH:mm:ss");

bookingHistory.Add(new BookingHistoryItem

{

StartDate = rentalStartDate,

EndDate = endOfRental,

ActualEndDate = actualEndDateText

});

}

orderListBox.ItemsSource = bookingHistory;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при отриманні історії бронювань: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

dataBase.closeConnection();

}

}

private void LoadFinesHistory(int idKlient)

{

try

{

dataBase.openConnection();

string queryString = @"

SELECT name\_Fines, sum\_Fines

FROM Fines

WHERE ID\_Klient = @idKlient

";

using (SqlCommand command = new SqlCommand(queryString, dataBase.getSqlConnection()))

{

command.Parameters.AddWithValue("@idKlient", idKlient);

using (SqlDataReader reader = command.ExecuteReader())

{

List<FinesHistoryItem> finesHistory = new List<FinesHistoryItem>();

while (reader.Read())

{

string name\_Fines = reader["name\_Fines"].ToString();

string sum\_Fines = reader["sum\_Fines"].ToString();

finesHistory.Add(new FinesHistoryItem

{

NameFines = name\_Fines,

SumFines = sum\_Fines

});

}

finesListBox.ItemsSource = finesHistory;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при отриманні історії штрафів: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

dataBase.closeConnection();

}

}

private void LoadDiscountHistory(int idKlient)

{

try

{

dataBase.openConnection();

string queryString = @"

SELECT name\_Discount, percent\_Discount

FROM Discount

WHERE ID\_Klient = @idKlient

";

using (SqlCommand command = new SqlCommand(queryString, dataBase.getSqlConnection()))

{

command.Parameters.AddWithValue("@idKlient", idKlient);

using (SqlDataReader reader = command.ExecuteReader())

{

List<DiscountHistoryItem> discountHistory = new List<DiscountHistoryItem>();

while (reader.Read())

{

string name\_Discount = reader["name\_Discount"].ToString();

string percent\_Discount = reader["percent\_Discount"].ToString();

discountHistory.Add(new DiscountHistoryItem

{

NameDiscount = name\_Discount,

PercentDiscount = percent\_Discount

});

}

discountListBox.ItemsSource = discountHistory;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при отриманні історії персональних акцій користувача: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

dataBase.closeConnection();

}

}

private void MainWindow\_Button\_Click(object sender, RoutedEventArgs e)

{

MainWindow MainWindow = new MainWindow(\_idKlient);

MainWindow.Show();

this.Close();

}

private void button\_exit\_Click(object sender, RoutedEventArgs e)

{

Application.Current.Shutdown();

}

}

}

Сторінка AdminMenu

xaml

<Window x:Class="Course.AdminMenu"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Course"

mc:Ignorable="d" WindowStartupLocation="CenterScreen"

Title="AdminMenu" Height="540" Width="960">

<Grid>

<Image HorizontalAlignment="Left" Height="63" VerticalAlignment="Top" Width="172" Source="Resources/label.png"/>

<Label x:Name="label\_admin" Content="Меню адміністрування" HorizontalAlignment="Center" VerticalAlignment="Top"

FontFamily="Yu Gothic UI Semibold" FontSize="22" Margin="0,15,0,0"/>

<Grid HorizontalAlignment="Center" VerticalAlignment="Center" Margin="0 40 0 0">

<Grid.RowDefinitions>

<RowDefinition Height="60"></RowDefinition>

<RowDefinition Height="60"></RowDefinition>

<RowDefinition Height="60"></RowDefinition>

<RowDefinition Height="60"></RowDefinition>

<RowDefinition Height="60"></RowDefinition>

</Grid.RowDefinitions>

<Button x:Name="open\_AdminWindow" Grid.Row="0" Content="Додати машину до Бази" FontFamily="Yu Gothic UI Semibold" Background="#87BBA4"

FontSize="18" Height="45" Width="220" Cursor="Hand" FontWeight="Bold" Click="open\_AdminWindow\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="open\_DataBaseEditor" Grid.Row="1" Content="Відкрити вікно Бази" FontFamily="Yu Gothic UI Semibold" Background="#87BBA4"

FontSize="18" Height="45" Width="220" Cursor="Hand" FontWeight="Bold" Click="open\_DataBaseEditor\_Click" >

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="return\_TheCar" Grid.Row="2" Content="Повернути авто" FontFamily="Yu Gothic UI Semibold" Background="#87BBA4"

FontSize="18" Height="45" Width="220" Cursor="Hand" FontWeight="Bold" Click="return\_TheCar\_Click" >

<Button.Template>

<ControlTemplate TargetType="{x:Type Button}">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="finesDiscount" Grid.Row="3" Content="Знижки та штрафи" FontFamily="Yu Gothic UI Semibold" Background="#87BBA4"

FontSize="18" Height="45" Width="220" Cursor="Hand" FontWeight="Bold" Click="finesDiscount\_Click">

<Button.Template>

<ControlTemplate TargetType="{x:Type Button}">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="back\_to\_log\_in" Grid.Row="4" Content="До авторизацї" FontFamily="Yu Gothic UI Semibold" Background="#87BBA4"

FontSize="18" Height="45" Width="220" Cursor="Hand" FontWeight="Bold" Click="back\_to\_log\_in\_Click" >

<Button.Template>

<ControlTemplate TargetType="{x:Type Button}">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

<Button x:Name="button\_exit" Height="59" Width="140" HorizontalAlignment="Right" VerticalAlignment="Bottom" Click="button\_exit\_Click" Margin="0,0,10,10" Cursor="Hand">

<Button.Background>

<ImageBrush ImageSource="/Resources/exit\_icon.png">

<ImageBrush.RelativeTransform>

<TransformGroup>

<ScaleTransform CenterY="0.5" CenterX="0.5" ScaleX="0.3" ScaleY="0.75"/>

<SkewTransform CenterX="0.5" CenterY="0.5"/>

<RotateTransform CenterX="0.5" CenterY="0.5"/>

<TranslateTransform/>

</TransformGroup>

</ImageBrush.RelativeTransform>

</ImageBrush>

</Button.Background>

<Button.Template>

<ControlTemplate TargetType="{x:Type Button}">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</Window>

xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Course

{

public partial class AdminMenu : Window

{

public AdminMenu()

{

InitializeComponent();

}

private void open\_AdminWindow\_Click(object sender, RoutedEventArgs e)

{

AdminWindow AdminWindow = new AdminWindow();

AdminWindow.Show();

this.Close();

}

private void open\_DataBaseEditor\_Click(object sender, RoutedEventArgs e)

{

DataBaseEditor DataBaseEditor = new DataBaseEditor("SELECT \* FROM Klient", 2, "Klient");

DataBaseEditor.Show();

this.Close();

}

private void back\_to\_log\_in\_Click(object sender, RoutedEventArgs e)

{

log\_in log\_in = new log\_in();

log\_in.Show();

this.Close();

}

private void button\_exit\_Click(object sender, RoutedEventArgs e)

{

Application.Current.Shutdown();

}

private void return\_TheCar\_Click(object sender, RoutedEventArgs e)

{

DataBaseEditor DataBaseEditor = new DataBaseEditor("SELECT \* FROM Rent", 1, "Rent");

DataBaseEditor.Show();

this.Close();

}

private void finesDiscount\_Click(object sender, RoutedEventArgs e)

{

DataBaseEditor DataBaseEditor = new DataBaseEditor("SELECT \* FROM Discount", 3, "Discount");

DataBaseEditor.Show();

this.Close();

}

}

}

Сторінка AdminWindow

xaml

<Window x:Class="Course.AdminWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Course" WindowStartupLocation="CenterScreen"

mc:Ignorable="d"

Title="AdminWindow" Height="540" Width="960">

<Grid>

<Image HorizontalAlignment="Left" Height="63" VerticalAlignment="Top" Width="172" Source="Resources/label.png"/>

<Label x:Name="label\_admin" Content="Вікно адміністрування" HorizontalAlignment="Center" VerticalAlignment="Top"

FontFamily="Yu Gothic UI Semibold" FontSize="22" Margin="0,15,0,0"/>

<Grid Width="820" Height="175" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,90,0,0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="120"/>

<ColumnDefinition Width="200"/>

<ColumnDefinition Width="300"/>

<ColumnDefinition Width="200"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="35"/>

<RowDefinition Height="35"/>

<RowDefinition Height="35"/>

<RowDefinition Height="35"/>

<RowDefinition Height="35"/>

</Grid.RowDefinitions>

<Label x:Name="label\_carNumber" Grid.Row="0" Grid.Column="0" Content="Номер авто " FontFamily="Yu Gothic UI Semibold" FontSize="18" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_carNumber" Grid.Row="0" Grid.Column="1" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

<Label x:Name="label\_carBrand" Grid.Row="1" Grid.Column="0" Content="Бренд авто " FontFamily="Yu Gothic UI Semibold" FontSize="18" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_carBrand" Grid.Row="1" Grid.Column="1" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

<Label x:Name="label\_carModel" Grid.Row="2" Grid.Column="0" Content="Модель авто " FontFamily="Yu Gothic UI Semibold" FontSize="18" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_carModel" Grid.Row="2" Grid.Column="1" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

<Label x:Name="label\_carColor" Grid.Row="3" Grid.Column="0" Content="Колір авто " FontFamily="Yu Gothic UI Semibold" FontSize="18" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_carColor" Grid.Row="3" Grid.Column="1" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

<Label x:Name="label\_carYear" Grid.Row="4" Grid.Column="0" Content="Рік випуску " FontFamily="Yu Gothic UI Semibold" FontSize="18" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_carYear" Grid.Row="4" Grid.Column="1" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

<Label x:Name="label\_averageFuelConsumption" Grid.Row="0" Grid.Column="2" Content="Середнє споживання пального " FontFamily="Yu Gothic UI Semibold" FontSize="18" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_averageFuelConsumption" Grid.Row="0" Grid.Column="3" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

<Label x:Name="label\_cost1\_3DayRental" Grid.Row="1" Grid.Column="2" Content="Вартість 1-3 днів оренди " FontFamily="Yu Gothic UI Semibold" FontSize="18" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_cost1\_3DayRental" Grid.Row="1" Grid.Column="3" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

<Label x:Name="label\_cost4\_9DayRental" Grid.Row="2" Grid.Column="2" Content="Вартість 4-9 днів оренди " FontFamily="Yu Gothic UI Semibold" FontSize="18" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_cost4\_9DayRental" Grid.Row="2" Grid.Column="3" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

<Label x:Name="label\_cost10\_25DayRental" Grid.Row="3" Grid.Column="2" Content="Вартість 10-25 днів оренди " FontFamily="Yu Gothic UI Semibold" FontSize="18" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_cost10\_25DayRental" Grid.Row="3" Grid.Column="3" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

<Label x:Name="label\_cost26DayRental" Grid.Row="4" Grid.Column="2" Content="Вартість від 26 днів оренди " FontFamily="Yu Gothic UI Semibold" FontSize="18" VerticalAlignment="Center" HorizontalAlignment="Right"/>

<TextBox x:Name="textBox\_cost26DayRental" Grid.Row="5" Grid.Column="3" TextWrapping="Wrap" FontFamily="Yu Gothic UI Semibold" FontSize="20" Height="30" Background="#CEDED7"/>

</Grid>

<Grid Width="405" Height="200" HorizontalAlignment="Left" VerticalAlignment="Bottom" Margin="20,0,0,20" >

<Border BorderBrush="Black" BorderThickness="1" Width="200" Height="200" VerticalAlignment="Top" HorizontalAlignment="Left">

<Image x:Name="imagePreview" Width="200" Height="200" Stretch="Uniform" />

</Border>

<Button x:Name="button\_OpenImage" Content="Обрати зображення" FontFamily="Yu Gothic UI Semibold" Background="#87BBA4"

FontSize="18" Height="45" Width="200" HorizontalAlignment="Right" Cursor="Hand" Click="OpenImageDialog" VerticalAlignment="Center" FontWeight="Bold">

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

<Button x:Name="button\_AddToBD" Content="Додати машину до Бази" FontFamily="Yu Gothic UI Semibold" Background="#87BBA4"

FontSize="18" Height="45" Width="220" HorizontalAlignment="Right" Cursor="Hand" VerticalAlignment="Bottom" FontWeight="Bold" Margin="0,0,70,97" Click="button\_AddToBD\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="button\_GoToAdminMenu" Content="Повернутися в меню" FontFamily="Yu Gothic UI Semibold" Background="#87BBA4"

FontSize="18" Height="45" Width="220" HorizontalAlignment="Right" Cursor="Hand" VerticalAlignment="Bottom" FontWeight="Bold" Margin="0,0,70,20" Click="button\_GoToAdminMenu\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</Window>

xaml.cs

using Microsoft.Win32;

using MyLib;

using System;

using System.Data.SqlClient;

using System.IO;

using System.Windows;

using System.Windows.Media.Imaging;

namespace Course

{

public partial class AdminWindow : Window

{

DataBase dataBase = new DataBase();

public AdminWindow()

{

InitializeComponent();

}

private byte[] selectedImageData;

private void OpenImageDialog(object sender, RoutedEventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Зображення|\*.jpg;\*.jpeg;\*.png;\*.gif|Всі файли|\*.\*";

if (openFileDialog.ShowDialog() == true)

{

// Отримати вибраний файл та відобразити його в Image

string imagePath = openFileDialog.FileName;

selectedImageData = File.ReadAllBytes(imagePath);

imagePreview.Source = new BitmapImage(new Uri(imagePath));

}

}

private void button\_AddToBD\_Click(object sender, RoutedEventArgs e)

{

try

{

if (selectedImageData == null)

{

MessageBox.Show("Будь ласка, оберіть фото", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

string carNumber = textBox\_carNumber.Text;

string carBrand = textBox\_carBrand.Text;

string carModel = textBox\_carModel.Text;

string carColor = textBox\_carColor.Text;

int carYear = int.Parse(textBox\_carYear.Text);

decimal averageFuelConsumption = decimal.Parse(textBox\_averageFuelConsumption.Text);

byte[] photoData = selectedImageData;

decimal cost1\_3DayRental = decimal.Parse(textBox\_cost1\_3DayRental.Text);

decimal cost4\_9DayRental = decimal.Parse(textBox\_cost4\_9DayRental.Text);

decimal cost10\_25DayRental = decimal.Parse(textBox\_cost10\_25DayRental.Text);

decimal cost26DayRental = decimal.Parse(textBox\_cost26DayRental.Text);

dataBase.openConnection();

string queryString = @"INSERT INTO Avto

(car\_Number, car\_Brand, car\_Model, car\_Color, car\_Year,

average\_Fuel\_Consumption, PhotoData,

cost\_1\_3\_Day\_Rental, cost\_4\_9\_Day\_Rental, cost\_10\_25\_Day\_Rental, cost\_26\_Day\_Rental)

VALUES

(@CarNumber, @CarBrand, @CarModel, @CarColor, @CarYear,

@AverageFuelConsumption, @PhotoData,

@Cost1\_3DayRental, @Cost4\_9DayRental, @Cost10\_25DayRental, @Cost26DayRental)";

using (SqlCommand command = new SqlCommand(queryString, dataBase.getSqlConnection()))

{

command.Parameters.AddWithValue("@CarNumber", carNumber);

command.Parameters.AddWithValue("@CarBrand", carBrand);

command.Parameters.AddWithValue("@CarModel", carModel);

command.Parameters.AddWithValue("@CarColor", carColor);

command.Parameters.AddWithValue("@CarYear", carYear);

command.Parameters.AddWithValue("@AverageFuelConsumption", averageFuelConsumption);

command.Parameters.AddWithValue("@PhotoData", photoData);

command.Parameters.AddWithValue("@Cost1\_3DayRental", cost1\_3DayRental);

command.Parameters.AddWithValue("@Cost4\_9DayRental", cost4\_9DayRental);

command.Parameters.AddWithValue("@Cost10\_25DayRental", cost10\_25DayRental);

command.Parameters.AddWithValue("@Cost26DayRental", cost26DayRental);

command.ExecuteNonQuery();

}

dataBase.closeConnection();

MessageBox.Show("Дані успішно додані до бази даних", "Успіх", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при додаванні даних: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void button\_GoToAdminMenu\_Click(object sender, RoutedEventArgs e)

{

AdminMenu AdminMenu = new AdminMenu();

AdminMenu.Show();

this.Close();

}

}

}

Сторінка DataBaseEditor

xaml

<Window x:Class="Course.DataBaseEditor"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Course"

mc:Ignorable="d" WindowStartupLocation="CenterScreen"

Title="DataBaseEditor" Height="540" Width="960">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="63"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="84"/>

</Grid.RowDefinitions>

<DataGrid Grid.Row="1" x:Name="dataGrid" Width="950" Height="300" HorizontalAlignment="Center" SelectionMode="Single" SelectionUnit="Cell"

VerticalAlignment="Top" Margin="0,10,0,0" AutoGenerateColumns="True" IsReadOnly="True" PreviewMouseLeftButtonUp="DataGrid\_PreviewMouseLeftButtonUp"/>

<Grid Grid.Row="0">

<Image HorizontalAlignment="Left" Height="63" VerticalAlignment="Top" Width="172" Source="Resources/label.png"/>

<ComboBox x:Name="showComboBox" VerticalAlignment="Center" HorizontalAlignment="Right"

FontFamily="Yu Gothic UI Semibold" Background="#87BBA4" BorderBrush="#87BBA4" SelectedIndex="1"

Width="200" Height="31" FontSize="16" SelectionChanged="showComboBox\_SelectionChanged" Margin="0,0,230,0">

</ComboBox>

<Grid Height="30" Width="200" VerticalAlignment="Center" HorizontalAlignment="Right" Margin="0,0,15,0" >

<TextBox x:Name="searchTextBox" TextChanged="searchTextBox\_TextChanged" VerticalContentAlignment="Center"

FontFamily="Yu Gothic UI Semibold" FontSize="16" Height="30" Width="200"/>

<Image Source="Resources/loupe.png" Height="25" Width="25" VerticalAlignment="Center" HorizontalAlignment="Right" Margin="0,0,3,0"></Image>

</Grid>

</Grid>

<Grid Grid.Row="2">

<Button x:Name="button\_GoToAdminMenu" Content="Повернутися в меню" FontFamily="Yu Gothic UI Semibold" Background="#87BBA4"

FontSize="18" Height="45" Width="220" HorizontalAlignment="Left" Cursor="Hand" VerticalAlignment="Center" FontWeight="Bold" Click="button\_GoToAdminMenu\_Click" Margin="15,0,0,0">

<Button.Template>

<ControlTemplate TargetType="{x:Type Button}">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="20">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

<Grid x:Name="gridTime" Width="250" Height="60" VerticalAlignment="Center" Margin="0,0,15,0">

<TextBox x:Name="timeTextBox" Width="243" Height="25" FontSize="14" TextAlignment="Center" FontFamily="Yu Gothic UI Semibold" IsReadOnly="True" Focusable="False" HorizontalAlignment="Center" VerticalAlignment="Top"/>

<Button x:Name="GetDateTime" Width="250" Height="25" Content="Отримати значення поточного часу" Click="GetDateTime\_Click" VerticalAlignment="Bottom" HorizontalAlignment="Center" Background="#87BBA4" FontFamily="Yu Gothic UI Semibold" Cursor="Hand" FontSize="12">

<Button.Template>

<ControlTemplate TargetType="{x:Type Button}">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

<Grid x:Name="gridId" Width="250" Height="60" VerticalAlignment="Center" Margin="0,0,15,0">

<TextBox x:Name="idTextBox" Width="243" Height="25" FontSize="14" TextAlignment="Center" FontFamily="Yu Gothic UI Semibold" HorizontalAlignment="Center" VerticalAlignment="Top"/>

<Button x:Name="GetId" Width="250" Height="25" Content="Обрати Id Клієнта та створити запис" Click="GetIdKlient\_Click" VerticalAlignment="Bottom" HorizontalAlignment="Center" Background="#87BBA4" FontFamily="Yu Gothic UI Semibold" Cursor="Hand" FontSize="12">

<Button.Template>

<ControlTemplate TargetType="{x:Type Button}">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

<Grid Height="60" Width="330" HorizontalAlignment="Right" Margin="0,0,15,0">

<TextBox x:Name="textBox\_forUpdate" Background="Transparent" HorizontalAlignment="Left" VerticalAlignment="top"

FontSize="18" FontFamily="Yu Gothic UI Semibold" Width="200" Height="25"/>

<Button x:Name="SaveChangesButton" Content="Зберегти зміни" FontFamily="Yu Gothic UI Semibold" Background="#87BBA4"

FontSize="14" Height="25" Width="120" Cursor="Hand" FontWeight="Bold" HorizontalAlignment="Right" VerticalAlignment="Top"

Click="SaveChangesButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="button\_Delete" Content="Видалити обраний рядок" FontFamily="Yu Gothic UI Semibold" Background="#87BBA4"

FontSize="14" Height="25" Width="330" Cursor="Hand" FontWeight="Bold" VerticalAlignment="Bottom"

Click="button\_Delete\_Click">

<Button.Template>

<ControlTemplate TargetType="{x:Type Button}">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</Grid>

</Grid>

</Window>

xaml.cs

using MyLib;

using System;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

namespace Course

{

public partial class DataBaseEditor : Window

{

DataBase dataBase = new DataBase();

private string \_showBy;

private string \_columnName;

private object \_Id;

private string \_firstColumnName;

public DataBaseEditor(string queryString, int check, string showBy)

{

InitializeComponent();

LoadDataIntoGrid(queryString);

if (check == 1)

{

showComboBox.Visibility = Visibility.Hidden;

gridId.Visibility = Visibility.Hidden;

\_showBy = showBy;

ComboBoxItem item3 = new ComboBoxItem() { Content = "Таблиця оренди авто", Tag = "Rent" };

showComboBox.Items.Add(item3);

}

else if(check == 2)

{

gridTime.Visibility = Visibility.Hidden;

gridId.Visibility = Visibility.Hidden;

\_showBy = showBy;

ComboBoxItem item1 = new ComboBoxItem() { Content = "Таблиця авто", Tag = "Avto" };

ComboBoxItem item2 = new ComboBoxItem() { Content = "Таблиця клієнт", Tag = "Klient" };

ComboBoxItem item3 = new ComboBoxItem() { Content = "Таблиця оренди авто", Tag = "Rent" };

ComboBoxItem item4 = new ComboBoxItem() { Content = "Таблиця штрафів", Tag = "Fines" };

ComboBoxItem item5 = new ComboBoxItem() { Content = "Таблиця знижок", Tag = "Discount" };

showComboBox.Items.Add(item1);

showComboBox.Items.Add(item2);

showComboBox.Items.Add(item3);

showComboBox.Items.Add(item4);

showComboBox.Items.Add(item5);

}

else

{

gridTime.Visibility = Visibility.Hidden;

ComboBoxItem item4 = new ComboBoxItem() { Content = "Таблиця штрафів", Tag = "Fines" };

ComboBoxItem item5 = new ComboBoxItem() { Content = "Таблиця знижок", Tag = "Discount" };

showComboBox.Items.Add(item4);

showComboBox.Items.Add(item5);

}

}

private void showComboBox\_SelectionChanged(object sender, System.Windows.Controls.SelectionChangedEventArgs e)

{

ComboBoxItem selectedItem = (ComboBoxItem)showComboBox.SelectedItem;

if (selectedItem != null)

{

\_showBy = selectedItem.Tag.ToString();

string queryString = $"SELECT \* FROM {\_showBy}";

LoadDataIntoGrid(queryString);

}

}

private void DataGrid\_PreviewMouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

DependencyObject dep = (DependencyObject)e.OriginalSource;

while ((dep != null) && !(dep is DataGridCell))

{

dep = VisualTreeHelper.GetParent(dep);

}

if (dep == null)

{

return;

}

DataGridCellInfo selectedCell = dataGrid.SelectedCells.FirstOrDefault();

if (selectedCell != null)

{

object cellContent = selectedCell.Column.GetCellContent(selectedCell.Item);

if (cellContent is TextBlock textBlock)

{

string cellValue = textBlock.Text;

textBox\_forUpdate.Text = cellValue;

\_columnName = selectedCell.Column.Header.ToString();

\_firstColumnName = dataGrid.Columns.Count > 0 ? dataGrid.Columns[0].Header.ToString() : string.Empty;

object rowData = selectedCell.Item;

if (rowData is DataRowView dataRowView)

{

\_Id = dataRowView.Row.ItemArray.Length > 0 ? dataRowView.Row.ItemArray[0] : null;

}

}

}

}

private void DataUpdate(string newValue)

{

try

{

dataBase.openConnection();

string updateQuery = $"UPDATE {\_showBy} SET {\_columnName} = @NewValue WHERE {\_firstColumnName} = @ConditionValue";

using (SqlCommand command = new SqlCommand(updateQuery, dataBase.getSqlConnection()))

{

command.Parameters.AddWithValue("@NewValue", newValue);

command.Parameters.AddWithValue("@ConditionValue", \_Id);

command.ExecuteNonQuery();

}

MessageBox.Show("Дані успішно оновлено", "Інформація", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при оновленні даних: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

dataBase.closeConnection();

textBox\_forUpdate.Clear();

}

}

private void button\_Delete\_Click(object sender, RoutedEventArgs e)

{

try

{

dataBase.openConnection();

string deleteQuery = $"DELETE FROM {\_showBy} WHERE {\_firstColumnName} = @IdToDelete";

using (SqlCommand command = new SqlCommand(deleteQuery, dataBase.getSqlConnection()))

{

command.Parameters.AddWithValue("@IdToDelete", \_Id);

command.ExecuteNonQuery();

}

MessageBox.Show("Рядок успішно видалено", "Інформація", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при видаленні рядка: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

dataBase.closeConnection();

LoadDataIntoGrid($"SELECT \* FROM {\_showBy}");

textBox\_forUpdate.Clear();

}

}

private void SaveChangesButton\_Click(object sender, RoutedEventArgs e)

{

DataGridCellInfo selectedCell = dataGrid.SelectedCells[0];

object cellContent = selectedCell.Column.GetCellContent(selectedCell.Item);

string newValue = textBox\_forUpdate.Text;

if (cellContent is TextBlock textBlock)

{

textBlock.Text = newValue;

}

DataUpdate(newValue);

textBox\_forUpdate.Clear();

}

private void LoadDataIntoGrid(string queryString)

{

DataTable dataTable = GetDataFromDatabase(queryString);

dataGrid.ItemsSource = dataTable.DefaultView;

HideColumn("PhotoData");

}

private void HideColumn(string columnName)

{

DataGridColumn column = dataGrid.Columns.FirstOrDefault(c => c.Header.ToString() == columnName);

if (column != null)

{

column.Visibility = Visibility.Collapsed;

}

}

private DataTable GetDataFromDatabase(string queryString)

{

DataTable dataTable = new DataTable();

try

{

dataBase.openConnection();

using (SqlCommand command = new SqlCommand(queryString, dataBase.getSqlConnection()))

{

using (SqlDataAdapter adapter = new SqlDataAdapter(command))

{

adapter.Fill(dataTable);

}

}

return dataTable;

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при отриманні даних з БД: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

return null;

}

finally

{

dataBase.closeConnection();

}

}

private void ProcessTable()

{

string proc;

switch (\_showBy)

{

case "Klient":

proc = "SearchInfoInKlient";

Search(proc);

break;

case "Avto":

proc = "SearchInfoInAvto";

Search(proc);

break;

case "Rent":

proc = "SearchInfoInRent";

Search(proc);

break;

default:

MessageBox.Show($"Не відома таблиця: {\_showBy}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

}

private void Search(string proc)

{

try

{

dataBase.openConnection();

using (SqlCommand command = new SqlCommand(proc, dataBase.getSqlConnection()))

{

command.CommandType = CommandType.StoredProcedure;

command.Parameters.AddWithValue("@searchText", searchTextBox.Text);

using (SqlDataAdapter adapter = new SqlDataAdapter(command))

{

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGrid.ItemsSource = dataTable.DefaultView;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при пошуку даних: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

dataBase.closeConnection();

}

}

private void searchTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

ProcessTable();

}

private void button\_GoToAdminMenu\_Click(object sender, RoutedEventArgs e)

{

AdminMenu AdminMenu = new AdminMenu();

AdminMenu.Show();

this.Close();

}

private void GetDateTime\_Click(object sender, RoutedEventArgs e)

{

if (\_showBy == "Rent" && \_columnName == "actual\_End\_Of\_Rental")

{

DateTime currentDateTime = DateTime.Now;

string formattedDateTime = currentDateTime.ToString("yyyy-MM-dd HH:mm:ss.fff");

timeTextBox.Text = currentDateTime.ToString("HH:mm:ss.fff");

textBox\_forUpdate.Text = formattedDateTime;

MessageBox.Show($"Ви вибрали: {formattedDateTime}");

}

else

MessageBox.Show($"Помилка при виборі комірок, ви можете \nзмінити дату тільки в таблиці Rent \nдля її стовпця actual\_End\_Of\_Rental", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

private void GetIdKlient\_Click(object sender, RoutedEventArgs e)

{

try

{

dataBase.openConnection();

string insertQuery = $"INSERT INTO {\_showBy} (ID\_Klient) Values (@ID\_Klient);";

using (SqlCommand command = new SqlCommand(insertQuery, dataBase.getSqlConnection()))

{

command.Parameters.AddWithValue("@ID\_Klient", idTextBox.Text);

command.ExecuteNonQuery();

MessageBox.Show("Дані успішно додано", "Успіх", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при вставці даних: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

dataBase.closeConnection();

LoadDataIntoGrid($"SELECT \* FROM {\_showBy}");

idTextBox.Clear();

}

}

}

}