# Вступ

У наш час, в світі безперервного потоку інформації, все більшої актуальності набувають процеси автоматизації робіт рутинного характеру і робіт, які важко піддаються формалізації. У зв'язку з цим щорічно розробляються програми, покликані вирішити виникаючі проблеми.

На реалізацію однієї з таких проблем і направлений даний програмний продукт. Цей проект покликаний максимально спростити і автоматизувати роботу з великими обсягами інформації.

Створення інформаційної системи передбачає, що основні операції по накопиченню, зберіганню і переробці інформації покладається на обчислювальну техніку, фахівець ж виконує лише певну частину ручних операцій і операцій. Обчислювальна техніка при цьому працює в тісній взаємодії з користувачем, який контролює її дії, змінюючи значення окремих параметрів в ході рішення задачі, а також вводить вихідні дані для вирішення завдань і функцій управління.

Темою даного курсового проекту є інформаційний веб-орієнтований сервіс для оренди коптерів.

Робота працівника досить великого агентства з оренди дронів пов'язана з накопиченням величезної кількості інформації про коптери, клієнтів і чеки виконаних оренд. Зберігання такого об’єму інформації на паперових носіях просто неможливо. При необхідності отримання необхідної інформації доведеться витратити велику кількість часу, що при сучасному об’ємі пасажиропотоку приведе до повної паралізації роботи агентства. Існує два шляхи вирішення цієї проблеми: значне розширення штату працівників агентств і надання більшої площі приміщення для їх роботи і написання інформаційної системи, котра перенесе велику частину роботи агента на обчислювальну техніку. Також основою для розробки послужив ряд причин, що виникають через вплив людського чинника, які негативно позначаються на якості і інтенсивності роботи агентства:

а) можливість випадкової помилки в ручному заповненні бланків;

б) неможливість швидко впоратися з великим обсягом інформації;

в) складність пошуку необхідної інформації.

Метою розробки є максимально можлива автоматизація процесу, усунення перерахованих недоліків існуючої системи роботи продажу ваучерів, і привнесення в неї факторів, що позитивно впливають на якість і терміни виконання реалізованих в ній функцій:

а) зменшення часу виконання кожної функції;

б) автоматичне створення чеку;

в) простий і швидкий пошук;

г) автоматичне відправлення електронного листа клієнту при закінченні оренди.

Все вищезазначене свідчить про актуальність проблеми, варіант вирішення якої представлений в даному курсовому проекті.

**1 АНАЛІЗ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ**

* 1. Аналіз предметної області

У сучасному процесі глобалізації все більшу роль грають орендні агентства. У зв'язку з цим усе більш актуальним стає питання автоматизації обслуговування клієнтів таких фірм.

Абсолютних аналогів даної системи немає. Одними з часткових аналогів є «Getmancar», «Mobilecar», але це сервіси оренди автомобілей за допомогою мобільного пристрою (див. рис. 1.1).

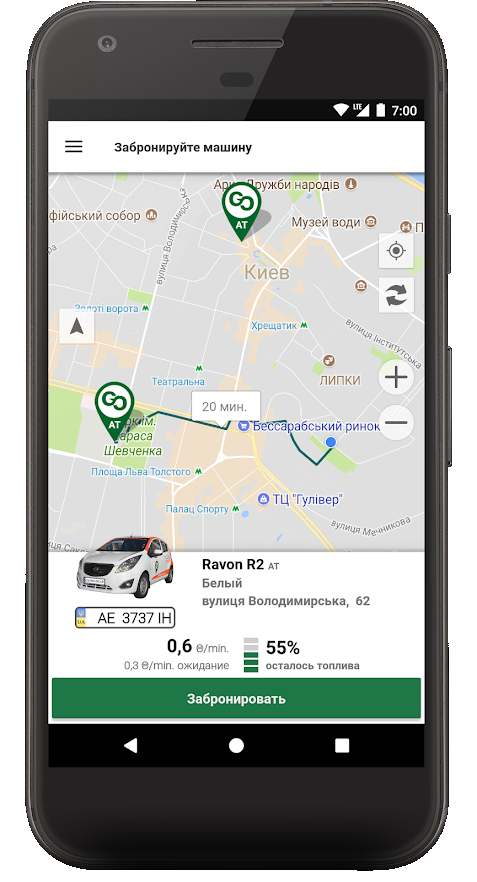


Рисунок 1.1 – Додаток «Getmancar» на мобільному пристрої

Основна проблема цієї предметної області полягає у величезному об'ємі інформації, що зберігається, і необхідності швидкого пошуку необхідної її частини. Зберігання її в паперовому виді вимагає великого простору і повністю позбавляє можливості швидкого доступу. Також проблемою є те, що інформація ділиться на окремі модулі: на інформацію про коптери, клієнтів та платіжні чеки. При необхідності отримання інформації про копт ери і про клієнтів які орендували цей коптер довелося б працювати з двома масивами різнотипної інформації [1]. Вочевидь, що для зберігання тисяч коптерів, клієнтів, чеків і т.д. необхідна інформаційна система, яка буде зберігати велику кількість даних та представляти швидкий та зручний доступ до них. Щоб спростити роботу користувача і надати дійсно потрібні йому функції, система повинна буде використовувати базу даних і надавати зручне відображення даних про коптери; швидкий пошук, фільтрацію і сортування потрібної інформації; отримання найбільш потрібної статистики і можливість формування чеків для друку.

Вимоги до програми:

а) можливість додавання нових коптерів, редагування існуючих (зміна технічних характеристик, фірми) і видалення;

б) фільтрація і пошук коптерів по назві, вартості;

в) можливість отримання інформації про коптер;

г) оформлення оренди при узгодженням з клієнтом;

д) оформлення чека до виконаної оренди;

е) отримання електронного листа з чеком.

Виходячи з вимог, можна скласти список об’єктів, інформацію по яких необхідно зберігати в базі даних інформаційної системи. Об'єктами в цій системі будуть: тур, чек, туристичний ваучер.

Фірми: ID номер у базі; назва; код країни;

Чеки: ID номер у базі; номер оренди, дата створення, загальна сума.

Знижки: ID номер у базі; тип знижки; кількість у процентах;

Коптери: ID номер у базі, статус, широта, довгота, вартість за хвилину, ID фірми, максимальна швидкість, максимальна висота польоту, тип управління, тип коптеру.

Клієнти: ID номер у базі; Прізвище; Ім’я; Пароль; Стать; Дата народження; Паспортні данні; Номер телефону; Email; Знижка.

Чеки: ID номер у базі; ID путівки; ID клієнту; кількість замовлених путівок; загальна вартість; дата продажу;

Країни: код країни; назва.

Оренди: ID номер у базі; ID номер клієнта; ID номер коптера; дата початку.

* 1. Опис функціональної структури системи

В умовах великої конкуренції в області оренд транспортних засобів чи електронних пристроїв одну з головних ролей грає швидкість реєстрації, пошуку й оформлення оренди. Для вирішення даного завдання програмний продукт надасть користувачу можливість швидкої оренди коптеру за допомогою зручного інтерфейсу користувача. Для клієнтів, які постійно користуються послугами агентств з питань оренди електронних пристроїв, гостро стоїть питання про фінансові витрати на оренди пристроїв. Для цього система буде автоматично відстежувати постійних клієнтів і вносити їх до списку клієнтів, яким надається знижка. Щоб прискорити і автоматизувати роботу працівника агентства з оренди, даний програмний продукт буде автоматично робити розсилку повідомлень щодо завершення оренди і отримання чеку.

Таким чином, дана інформаційна система буде значно збільшувати швидкість роботи агентства за рахунок значної автоматизації рутинної роботи.

* 1. Опис інформаційних потреб користувачів

На основі аналізу предметної області можна виділити наступні інформаційні потреби користувачів в даній системі.

Для клієнтів агентств:

а) вхід в програму;

б) перегляд інформації про коптери;

в) перегляд статистики, щодо зайнятих та вільних коптерів;

г) оренда коптерів;

д) оформлення чеку.

Для робітників агентств:

а) всі потреби клієнтів;

б) зміна інформації у коптерах;

в) додавання даних в систему;

г) видалення даних з системи;

д) перегляд інформації про клієнтів.

* 1. Опис існуючого документообігу в ПЗ

Документообіг предметної області «Агентство з оренди коптерів»

складається з наступних документів:

а) чек, який містить інформації про коптер та клієнта.

1.5 Опис обмежень цілісності

У даній галузі присутні такі обмеження щодо ідентифікації об'єктів:

а) кожна коптер має свою унікальну назву;

б) кожен клієнт має унікальний ID;

в) кожний чек має унікальний ID;

г) кожна країна має унікальний код;

д) кожна знижка має унікальний ID;

е) кожний фірма має унікальний ID;

ж) кожна оренда має унікальний ID.

Також даної предметної області притаманні такі обмеження щодо відносин між об'єктами:

а) кожен клієнт має одну чи не однієї оренди;

б) напрям в фірма може мати одну, багато, чи не одного коптерів;

в) кожен клієнт має одну, чи не однієї знижки;

г) одна оренда має один чек;

д) одна оренда має один коптер;

ж) одна фірма має одну країну.

2 ПОСТАНОВКА ЗАДАЧИ

Метою курсової роботи є реалізація робочого сервісу для оренди дронів різного типу і спрощення рутинної праці шляхом використання автоматизації процесу. Сервіс потрібен реалізовувати основні функції, які задані аналізом предметної області. Крім того, він повинен мати багаторівневу архітектуру, бути багатошаровим і мати можливість масштабуватися; на клієнтському рівні повинен бути веб-клієнт і мобільний додаток.

2.1 Основний функціонал системи

а) введення, зберігання, видалення і редагування інформації про коптери;

б) пошук, сортування (за ім’ям) та фільтрація коптерів, користувачів та фірм серед загальної кількості;

в) введення, зберігання, видалення і редагування інформації про фірми;

г) відображення на загальній мапі обраного коптера;

д) автоматичне знаходження найближчої зарядної станції для коптера;

е) відсилання телеметрії з коптера (координати);

ж) автоматичне оформлення чеку, який містить основну інформацію про коптер і клієнта і який можна роздрукувати;

и) автоматичне відправлення чеку на електрону адресу клієнта при завершенні оренди;

2.2 Допущення та залежності

Для коректної роботи усіх частин проекту наведений список допущень:

* користувачі системи мають пристрій з GPS системою;
* користувачі системи мають пристрій з доступом до Інтернету;
* користувачі зацікавлені в короткотривалій оренді дронів.

Також існують наступні залежності:

* додаток буде орієнтуватися на версії Android 4.0 і вище;
* додаток буде орієнтуватися на новітні версії браузерів (IE 9+, Chrome 45+, Safari 4.0+).

2.3 Релізи

Розробка даного програмного продукту була поділена на декілька релізів.

Необхідний функціонал для першого релізу:

а) реєстрація користувача ( з веб-сайту);

б) авторизація в системі (з веб-сайту);

в) можливість додавання нових коптерів, брендів (з веб-сайту);

г) можливість пошуку коптера або бренда за назвою (з веб-сайту);

д) можливість перегляду дронів на зашальній мапі (з мобільного додатку та веб-сайту);

ж) можливість перегляду основної інформації про коптер або бренд (з мобільного додатку та веб-сайту);

з) можливість замовити оренду коптера (з мобільного додатку та веб-сайту).

к) можливість отримувати електронний лист при завершенні оренди (з мобільного додатку та веб-сайту);

Реалізація наступних релізів:

л) можливість зареєструватися (з мобільного додатку);

м) можливість авторизуватися (з мобільного додатку);

п) можливість додавати нові коптери (з мобільного додатку);

р) можливість аналізувати телеметрію з коптуру (веб-сайт).

2.4 Користувацькі обмеження

Для коректного використання даного продукту та викреслення нерозуміння з боку користувачів, є деякі користувацькі обмеження:

а) користувач повинен пам’ятати логін та пароль;

б) користувач повинен авторизуватися для користування системою.

2.5 Бізнес - потреби та пріоритетність

Даний продукт може залучити користувачів та нових розробників своєю унікальністю та актуальною функціональністю (таблиця 2.1).

Таблиця 2.1 – Особи та їх інтереси

|  |  |  |
| --- | --- | --- |
| Інтереси | Обмеження | Зацікавлена ​​особа |
| Управління персональним орендованим дроном. | Бюджет | Користувач |
| Продовження таблиці 2.1 |  |  |
| Розробка масштабується. Система, що відповідає встановленим критеріям якості. | Терміни реалізації та бюджет | Розробник |
| Своєчасне отримання готового продукту з документацією. | Нема | Власник продукту |

Час і бюджет - головні обмеження розробника, таким чином потрібно відокремити пріоритетні показники для того, щоб акцентувати свою увагу на них в процесі розробки і бути більш стійким до стресових ситуацій, пов'язаних з нестачею ресурсів.

2.6. Середа оточення

Так як предметною областю даного веб-орієнтованого сервісу не передбачено використання певної СУБД для зберігання і роботи з інформацією системи була обрана СУБД Microsoft SQL Server 2017.

Даний веб-орієнтований сервіс буде розподілений на наступні части:

а)  серверна частина – платформа Microsoft .Net Core 2.2, мова програмування C#;

б) клієнтська частина – платформа Angular 7, мова програмування TypeScript, мова розмітки HTML 5, мова стилю CSS 3;

в) мобільна частина – платформа Android, мова програмування Kotlin;

г) емулятор IoT пристрою – платформа Microsoft .Net Core 2.2, мова програмування C#, розміщення Microsoft Azure Hub.

Сервіс буде розроблена в середовищі Microsoft Visual Studio 2017 [2].

У зв'язку з вибором засобів розробки, виникають обмеження з вибором операційної системи для роботи з системою. Користувачі повинні мати комп'ютер під управлінням ОС Microsoft Windows XP і вище, браузер (IE 9+, Chrome 45+, Safari 4.0+), мобільний додаток із встановленою версією ОС Android не менш ніж 4.0.0. Для функціонування емулятору або серверною частиною користувач не повинен встановлювати додаткове програмне забезпечення, тому що ці елементи розгорнуті у платформі хмарних обчислень Microsoft Azure.

1. АРХІТЕКТУРА ПРОЕКТУ

3.1 Архітектура проекту

Архітектура клієнт-сервер (див. рис. 3.1) є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

* набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
* набір клієнтів, які використовують сервіси, що надаються серверами;
* мережа, яка забезпечує взаємодію між клієнтами та серверами

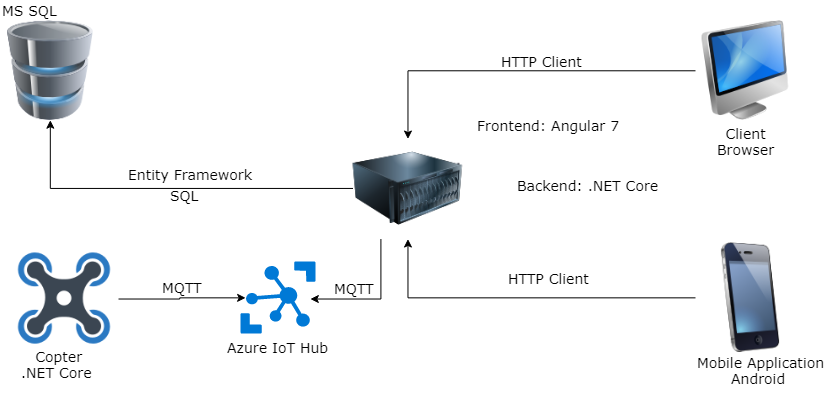


Рисунок 3.1 – Архітектура проекта

На рисунку 3.1 можна побачити взаємодію між різними компонентами системи. Так, для отримання результату з веб-браузеру користувача, front-end посилає запит на back-end сервер. Back-end сервер в свою чергу валідує дані та звертається до бази даних, далі в зворотньому напрямку компоненти передають результат до користувача.

Для більш детального аналізу архітектури, побудуємо діаграму розсортування (див. рис. 3.2)

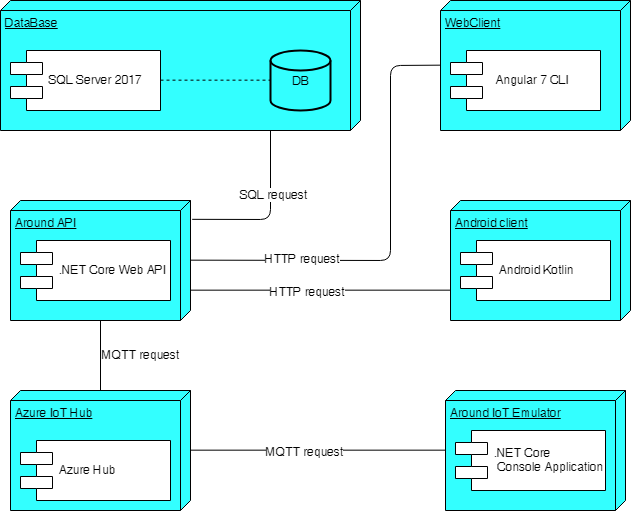


Рисунок 3.2 – Діграма розсортування

При роботі з базою даних було використано реляційну базу даних MSSQL та SQL Server 2017.

В якості клієнта може використовувати як web-сайт, так і мобільний додаток (Android) для виконання оренди коптеру, що включає запит на замовлення. Користувач для роботи з системою повинен авторизуватися. Цей процес включає хешування, за допомогою інтегрованою функції мови C#, паролю клієнта, направлення запиту на пошук користувача у базі. У тому разі якщо користувач існує в базі, йому видається унікальний JWT токен, який має час життя (5 хвилин) і який прикріплюється до кожного запиту, щоб сервер був впевнений, що запит робить авторизований користувач.

При роботі з коптером використовується спеціальна захищена черга у Azure IoT Hub та спеціальний протокол спілкування MQTT.

3.2 Будування UML діаграми

У даній системі існує три виду акторів – «Guest» – неавторизований користувач, авторизований користувач – «Authorized client» і адміністратор – «Administrator». Неавторизований користувач має можливість авторизуватись або зареєструватися у системі. Авторизований користувач має всі можливості ролі «Неавторизований користувач». Крім того він може продивлятися інформацію про коптери; отримувати статистику про зайняті коптери; знаходити коптери за допомогою фільтрів; отримувати знижку; продивлятися місцезнаходження коптеру на загальній мапі; замовляти оренду коптеру; отримувати чек. Останній актор – адміністратор має можливості ролі «Авторизований користувач» крім того, може редагувати коптери та інші сутності,

На основі цього складемо USE CASE діаграму запропонованої системи (див рис. 3.2).

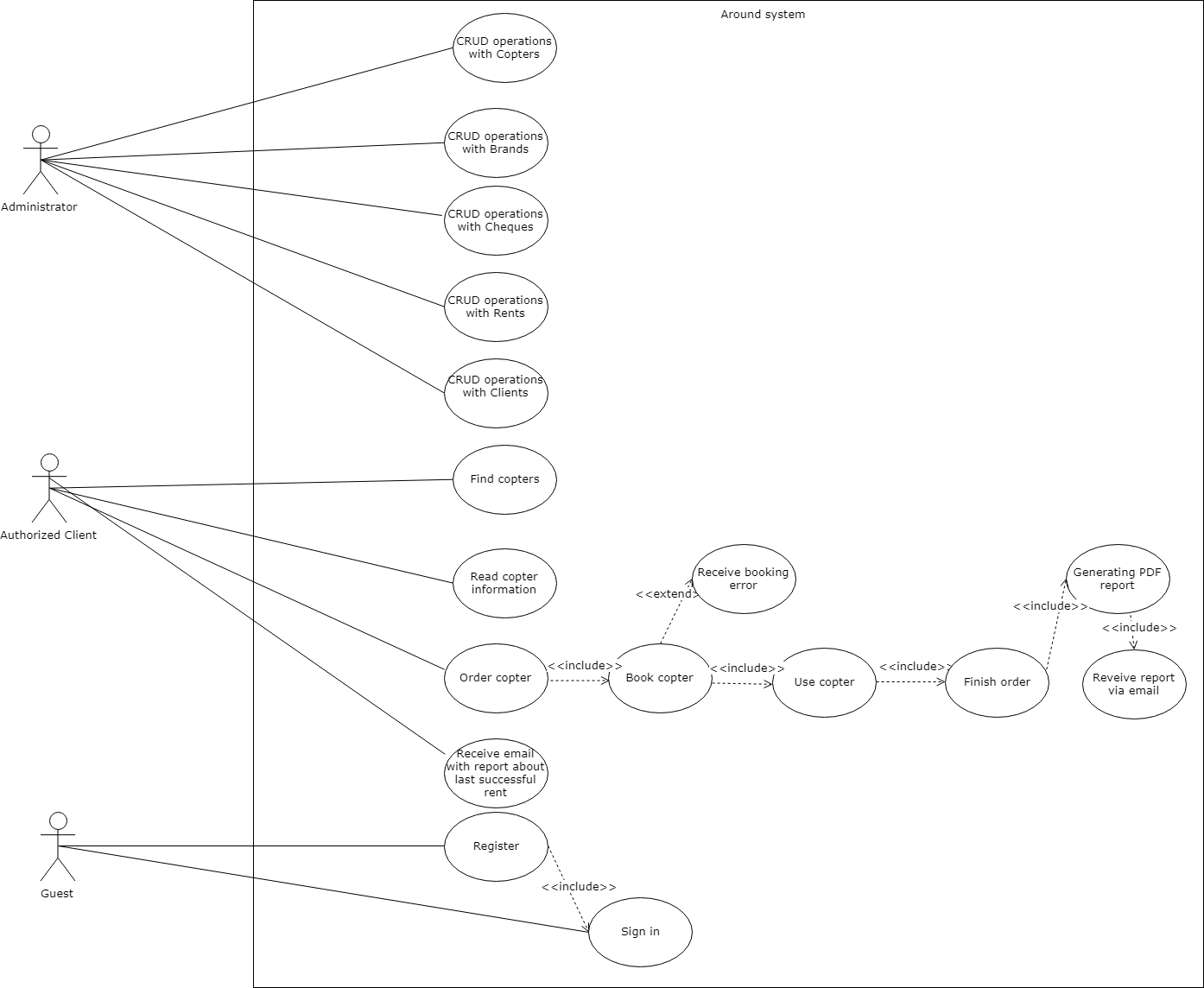


Рисунок 3.2 – UseCase діаграма

3.2 Побудова ER-діаграми

Побудова ER-діаграми є одним з важливих етапів проектування бази даних. Вона дозволяє схематично побачити модель майбутньої бази даних, яка буде побудована на основі цієї діаграми. На основі розробленої концептуальної моделі можна визначити таблиці, з котрих складатиметься база даних. Сутностями є: клієнт, коптер, чек і оренда. Виходячи з функціонала, який має надавати інформаційна система на етапі аналізу предметної області було отримано список даних, які необхідно зберігати в базі даних. Об'єднавши суті з даними можна побудувати модель бази даних у вигляді ER-діаграми, яка демонструє ставлення і зв'язку між таблицями в реляційної базі даних [3].

Етапи побудови ER-діаграми:

а) на основі концептуальної моделі визначені сутності, які будуть присутні у базі даних;

б) визначено відносини між отриманими сутностями (1:1; 1: М; М: М);

в) на основі інформації про об'єкти даної інформаційної системи, отриманих на етапі аналізу предметної області, визначаємо атрибути для даних про сутності;

г) проаналізувавши дані, що зберігаються визначено, що деякі поля таблиць можуть мати значення NULL. У таблиці «Clients» зберігаються дані про клієнтів агентства з оренди техніки. Кожен клієнт має право не повідомляти, контактний номер телефону, але тоді клієнт може не отримати інформацію щодо знижок.

В результаті ER-моделювання отримана діаграма, що демонструє схему бази даних і відносини між таблицями в базі даних (див. рис 3.3).

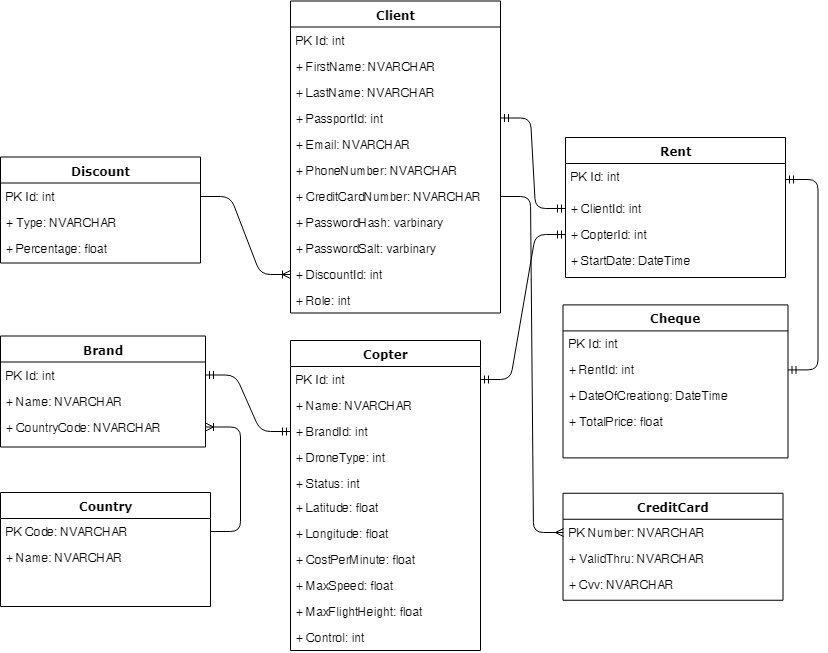


Рисунок 3.2 – ER діаграма бази даних

На основі цієї інформації можна приступати до створення файлу бази даних [4].

3.3. Побудова схеми реляційної бази даних в третій нормальної формі

На основі побудованої ER-діаграм побудуємо схема реляційної бази даних. Всі сутності, які були в концептуальній моделі та ER-діаграмі, становлять таблиці бази даних.. У результаті отримаємо схему бази даних (рис 3.3).

Необхідно довести, що база даних знаходиться у 3НФ. Перш ніж приступити до отримання таблиць в 3НФ наведемо визначення [5].

Відношення знаходиться у третій нормальній формі тоді і тільки тоді, коли воно знаходиться в другій нормальній формі і жоден неключовий атрибут не є транзитивно залежним від її первинного ключа. Транзитивна залежність — це залежність між неключовими ат­рибутами.

Отже переглянемо таблицю Clients(Клієнти). Ключом у неї є поле Id. При дальнішому розгляді можна зрозуміти, що всі наступні поля, а саме: FirstName (Ім’я), LastName (Прізвище), Sex (Стать), DateOfBorn (Дата народження), Passport(Паспортні дані), PhoneNumber (Номер телефону), Email (Електронная адресса), DiscountId(Код знижки), UserLevel (Рівень туриста) залежать лише від поля Id, тобто не мають транзитивну залежну. Як результат, виконується основна умова приведення до третьої нормальної форми.

Наступна таблиця Discount (Знижки). Ключом у неї є поле Id. Поля Type (Тип знижки), Percantages(Кількість у процентах) є незалежними між собою. Тобто всі поля таблиці Знижки залежні тількі від ключа. Отже, виконується основна умова 3НФ.

Таблиця Cheque (Платіжний чек) має ключ Id. Розглянемо інші поля: RentId (Код оренди), DateOfCreation (Дата створення), TotalPrice (Загальная вартість). При аналізі зрозуміло, що жодне з полів цієї таблиці не має залежності від не ключових атрибутів, тобто таблиця знаходиться у 3НФ.

Проаналізуємо таблицю Countries (Країни). Кючом цієї таблиці є поле Code. Єдине поле , окрім ключа, є поле Country отже таблиця автоматично знаходиться у третій нормальній формі.

Розглянемо таблицю Copters (Коптери). Ключом є поле Id. Розглянемо наступні поля Name (Ім’я), Status (Статус), Latitude (Широта), Longitude(Довгота), CostPerMinute (Вартість за хвилину), BrandId (Номер фірми), MaxSpeed (Максимальна швидкість), MaxHeight (Максимальна висота), Control (Тип управління), DroneType (Тип коптера). При розгляданні можна сказати, що всі поля залежать лише від ключа таблиці, тобто виконується основна умова 3НФ.

Роглянемо таблицю Rent (Оренди). Ключем цієї таблиці є поле Id. Проаналізуємо поля: ClientId (Номер клієнту), CopterId (номер коптеру), StartTime(Дата початку). Можна сказати, що жоден не ключовий атрибут не має транзитивної залежності. Отже Таблиця знаходиться у 3НФ.

Проаналізуємо таблицю Brands. Ключом таблиці є поле Id. Проаналізуємо інші поля Name (Назва) та CountryCode (Код країни). Можна сказати, що жоден не ключовий атрибут не має транзитивної залежності. Отже таблиця автоматично знаходиться у 3НФ.

Отже, якщо кожна з таблиць бази даних знаходиться у третій нормальній формі, то і вся таблиця знаходиться у третій нормальній формі. В результаті приведення таблиць бази даних до 3НФ була доведена правильність побудованої моделі бази даних на етапі побудови ER-діаграми. Таким чином, отримана точна модель бази даних, і можна приступати до створення файлу бази даних.

# 4 Кодування

## 4.1 Back-end сервер

У якості платформи була обрана платформа .NET Core 2.2 мови розробки бекенду було обрано C#. Завдяки даному вибору розробка API була швидкою, а код вийшов компактним і відмінно читаним. Крім того, має можливість дуже шкидко й гнучко масштабуватись. Серверна частина побудована за патерном Factory. У якості IDE був обраний Visual Studio, оскільки він повністю задовольняє мої потреби як розробника і є найкращим в своїй сфері.

Сервер системи створювався на платформі .NET Core 2.2.. Була створена база даних на основі підходу Code First, а також були написані контролери. Для доступу до даних використовувався Entity Framework Core.

Крім того весь код побудований з використанням патерну Dependency Injection (паттерн в якому використовується принцип низької залежності компонентів один від одного).

Для прикладу наведемо код програмної реалізації моделі «Коптер»:

public class Copter

{

public int Id { get; set; }

public string Name { get; set; }

public Status Status { get; set; }

public double Latitude { get; set; }

public double Longitude { get; set; }

public double CostPerMinute { get; set; }

public int BrandId { get; set; }

public double MaxSpeed { get; set; }

public double MaxFlightHeight { get; set; }

public Control Control { get; set; }

public DroneType DroneType { get; set; }

public virtual Brand Brand { get; set; }

public virtual List<Rent> Rents { get; set; }

}

Даний код буде автоматично оброблений і на основі нього буде створено таблицю в базі даних. Аби це сталося необхідно спочатку створити та реалізувати клас DronesharingContext (додати усі сутності, які ми хочемо бачити у базі даних). Далі потрібно створити клас DesignTimeDbContextFactory і реалізувати інтерфейс IDesignTimeDbContextFactory<DronesharingContext>:

public class DesignTimeDbContextFactory : IDesignTimeDbContextFactory<DronesharingContext>

{

public DronesharingContext CreateDbContext(string[] args)

{

IConfigurationRoot configuration = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json")

.Build();

var builder = new DbContextOptionsBuilder<DronesharingContext>();

var connectionString = configuration.GetConnectionString("DefaultConnection");

builder.UseSqlServer(connectionString);

return new DronesharingContext(builder.Options);

}

}

Також наведемо приклад коду RentController,а саме код замовлення оренди коптеру:

[HttpPost]

[Route("create")]

public async Task<IActionResult> CreateRent([FromBody] RentAggregate rentDto)

{

var copter = \_copterRepository.Get(rentDto.CopterId).Result;

if (copter.Status != Status.Ordered)

{

var rent = Rent.CreateFromDto(rentDto);

rent = await \_rentRepository.Create(rent);

await \_hub.StartUsingCopter(rent);

\_copterRepository.UpdateStatus(rentDto.CopterId);

Продовження лістингу коду:

return Ok("Success");

}

return BadRequest();

}

В даному прикладі користувач відправляє на сервер свій ідентифікатор та ідентифікатор коптеру, який він хоче орендувати. Після цього система перевіряє, чи не зайнятий коптер, якщо дрон вільний, то йому відсилається запит, щоб заблокувати його від спроб інших користувачів забронювати цей коптер.Теперь користувач може використовувати дрон за своїм поглядом. Для тестування різних видів запитів я використовував дві утілити – Swagger UI та Postman.

4.2 Front-end клієнт

Веб-додаток було написано за допомогою технології Angular 7 та мови програмування TypeScript. Я обрал платформу Angular через те, що це проект двох прогресивних корпорацій – Google та Microsoft. У останній час зв’язка .NET Core і Angular стала дуже міцною. Це можна пояснити сильною активністю двох великих ком’юніті – розробників .NET та Angular. Кількість issue(пропозицій) на офіційних GitHub репозиторіях обох фірм зростається кожного дня. Це говорить про те, що йде активна модифікація обох платформ. Саме це сприяло моєму вибору цієї міцної зв’язки. В якості середи розробки виступала Visual Studio Code.

Наведемо приклад коду – сервіс, що відповідає за лотримання даних з серверної частини та розпакування ії у потрібні елементи для роботи на стороні клієнту:

@Injectable()

export class CopterService {

constructor(private http: HttpClient) { }

baseUrl = 'http://localhost:55555/api/Copter/';

getCopters() {

return this.http.get<CopterDto[]>(this.baseUrl);

}

getCopterById(id: number) {

return this.http.get<CopterDto>(this.baseUrl + '/' + id);

}

createCopter(copter: CopterAggregate) {

return this.http.post(this.baseUrl, copter,

{

headers: new HttpHeaders({ 'Content-Type': 'application/json' }),

responseType: 'text'

})

.subscribe(data => console.log('Works!'));

}

updateCopter(copter: Copter) {

return this.http.put(this.baseUrl + '/' + copter.id, copter);

}

deleteCopter(id: number) {

return this.http.delete(this.baseUrl + '/' + id);

}

}

Цей код демонструє CRUD методи для сутності Copter. Як можна побачити при виконанні POST запиту ми обов’язково повинні виконати функцію subscribe, якщо вона не буде викликана, то сервер не отримає необхідну модель і не зможе обробити запит.

Для методів, що потребують від користувача бути авторизованим, створений спеціальний клас-провайдер – TokenInterceptor, що перехоплює вихідні HTTP запити, та прописує їм у заголовку Authorization значення JWT токену авторизації користувача.

Одна з найважливіших особливостей цього проекту – це інтеграція із Google API. Завдяки Google Maps клієнт може подивитися, де зараз знаходиться коптер , який він хоче орендувати (див. рис. 4.1)

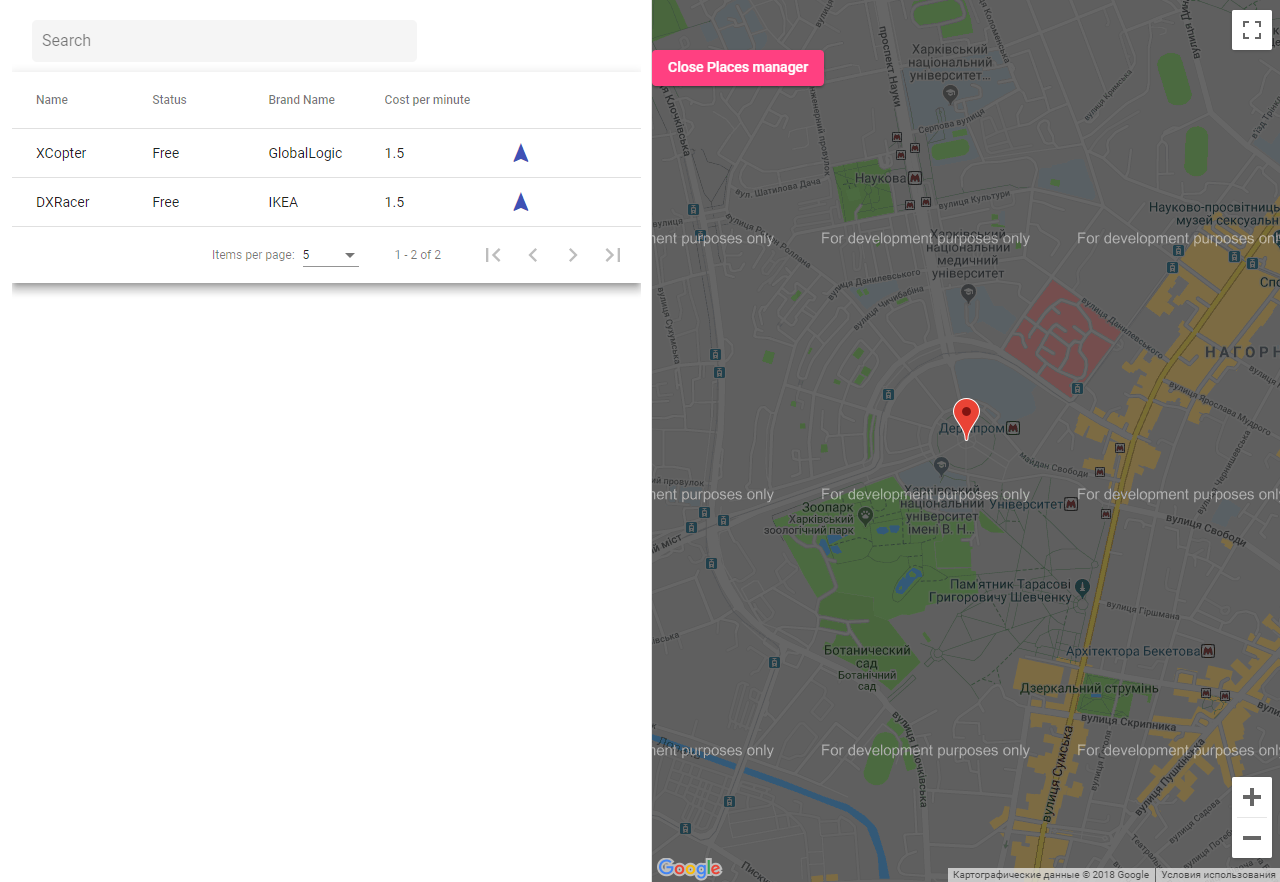


Рисунок 4.1 – Відібраження коптерів на загальній мапі

Таблиця у лівій частині екрана дозволяє користувачеві шукати потрібний коптер за ім’ям, дивитися, де він на мапі, а поті натискати кнопку оренди.

4.3 Мобільний клієнт

Мобільний додаток біло створено за допомогою мови Kotlin, Android API та IDE Android Studio, тобто, мобільний додаток є нативним додатком для платформи Android, і, таким чином, використовує всі відповідні переваги нативної розробки.

Для HTTP запитів до сервера було використано бібліотеку Retrofit2. Увесь код звернення до методів REST API сервера було інкапсульовано у сінглтон об’єкти. На рисунку наведен приклад списку доступних коптерів (див. рис. 4.2).

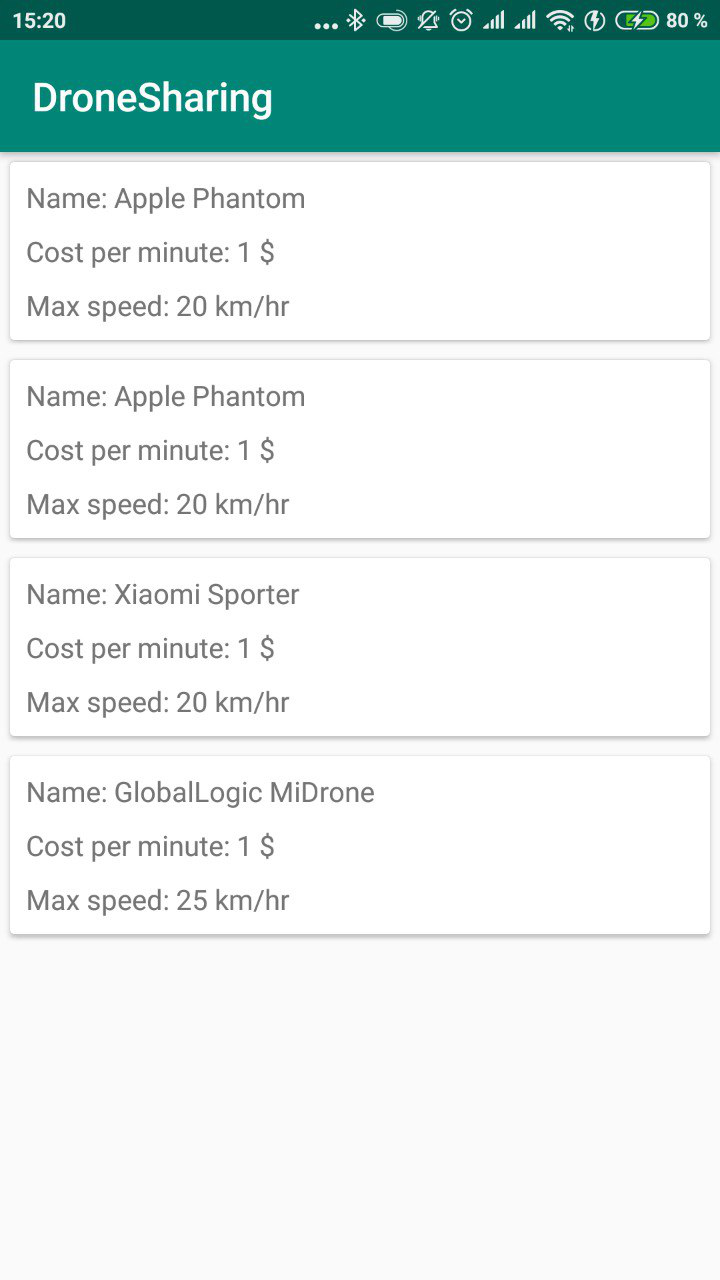
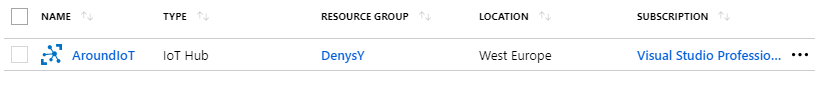


Рисунок 4.2 – Мобільний додаток

4.5 Розумний пристрій

В даному курсовому проекті розумний пристрій реалізований як емулятор IoT пристрою. Спочатку я створит .NET Core Console Application для отримання запросу з серверу. Потім я створив і зареєстрував новий пристрій в Azure IoT Hub. Тепер цей консольний додаток є повним IoT пристроєм, який спілкується із сервером через спеціальний протокол MQTT.(див. рис. 4.3)



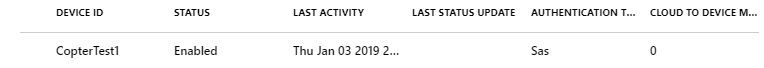


Рисунок 4.3 – Доска Azure Portal

На рисунку ми бачимо, що пристрій CopterTest1 є пристроєм ресурсу AroundIoT. Завдяки спеціальній властивості IoTHubSecret серверна частина може відправляти повідомлення у чергу на Azure. У той сами час ми знаємо, що якщо повідомлення будло відправлено в чергу, воно там буде зберігатися 30 днів (настройка за замовченням). Тепер коптер може вичитати повідомлення із абу як тільки буде онлайн.