МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Кваліфікаційна робота бакалавра

на тему: Розробка мобільного сервісу для допомоги ведення бізнесу з управління мережею електрозаправних станцій

Виконав студент групи К19і
спеціальності 122 Комп'ютерні науки
Мілєєв Владислав Юрійович

Керівник Штефан
Наталія Зінов'ївна

Консультант д.т.н. доцент
Великодний Станіслав Сергійович

Рецензент к. ф.-м. н., доцент
Ткач Тетяна Борісівна

Одеса 2021

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

СКБД – Система Керування Базою Даних

API – Application Programming Interface

CLR – Common Language Runtime

IDE – Integrated Development Environment

OCPP – Open Charge Point Protocol

UI – User Interface

UML – Unified Modeling Language

UX – User Experience

WPF – Windows Presentation Foundation

XML – Extensible Markup Language

Android – операційна система

BCL – бібліотека базових класів

C# – мова програмування

IntelliSense – технологія автозавершення коду під час введення

Just-in-time компіляція – компіляція на льоту

Microsoft Visual Studio – повнофункціональна інтегроване середовище розробки

Sql Azure – база даних

SQLite – реляційна база даних

Sql Server – база даних

Use-Case – варіант використання

Xamarin – платформа з відкритим кодом для створення додатків для iOS, Android та Windows за допомогою .NET.

XAML – мова декларативною розмітки інтерфейсу

ВСТУП

Ринок мобільних додатків розвивається швидкими темпами і за прогнозами провідних компаній зростання буде збільшуватися. З'являється все більше розробників, компаній і самих додатків. Зростає конкуренція і серед сервісів з однаковими функціями – хтось бере дизайном, хтось додає нові фічі, а хтось розкручується рекламою.

IOS і андроїд розробка додатків робить їх інтуїтивно зрозумілими, простими і неперевантаженість. Користувачів підкуповує можливість швидко і зручно отримати всю інформацію в одному місці [1].

Електрокари – це крок до екологічного і розумного майбутнього. Попит на автомобілі з електричним двигуном зростає по всьому світу. Це пов'язано зі збереженням екології, економічністю і комфортом. Ви їдете на машині, але не забруднюєте навколишнє середовище, не витрачаєте гроші на бензин чи парковку.

Щоб еко-автомобілі поширювалися в Україні, необхідно створити відповідну інфраструктуру. Перш за все, потрібно встановлювати зарядні станції, відводити окремі місця для паркування. Зараз кількість зарядних станцій для електрокарів у всьому світі перевищило 1 млн. Майже 80% глобального автомобільного ринку переходить на електромобілі та гібриди.

Зараз за українським законодавством розміщувати електрозарядкі можна тільки на дорогах державного значення, яких 46,6 тисячі кілометрів. А вимоги до кількості заправок на дорогах місцевого значення, яких 117 тисяч кілометрів, не визначені. Це обмежує можливість пересування на такому виді транспорту по країні. Норма тільки для електрозарядок на парковках – кількість місць на стоянках, обладнаних зарядками, має бути не менше 5% від усієї кількості паркомісць. Також є проблеми з отриманням технічних умов на підключення, виходячи з невизначення напруги і потужності [2].

У міжнародному рейтингу за темпами розвитку електромобілів Україна посіла п'яте місце, випередивши США, Швецію, Норвегію, Ісландію та Китай.

Цей вид транспорту обходиться дешевше, він простіше в обслуговуванні і екологічніше. Однак виникає питання наявності точок зарядки і зручного сервісу для власників електричних автомобілів [3].

Мета дипломної роботи – розробка мобільного додатку для мережи заправок електрокарів, який дозволяє власнику такого авто в лічені секунди знайти зарядну станцію, забронювати місце і час щоб заповнити енергію батареї.

Загальні характеристики кваліфікаційної роботи:

− повний обсяг сторінок пояснювальної записки – 49;

− кількість рисунків – 30;

− кількість посилань – 19.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Опис предметної області

З кожним днем інтерес до альтернативної енергетики не просто зростає, він трансформується в розробки, інновації та виробництво. Так, електромобілі на сьогоднішніх дорогах вже не викликають тієї бурі захвату і інтересу, як це було всього пару-трійку років тому. Зараз електрокар – це норма, як для широких проспектів мегаполісу, так і для тихих вуличок провінційних містечок.

Попит на екологічно чистий транспорт народжує пропозицію на облаштування заправних станцій і пунктів техобслуговування таких авто. Екологічні автомобілі мінімізують вплив на навколишнє середовище: електромобілі не працюють на дизелі і бензині, а гібридні транспортні засоби використовують двигун внутрішнього згоряння в поєднанні з електричним двигуном.

Постійне зростання попиту на електромобілі в різних містах України – це сигнал до того, що зовсім скоро заправки для такого транспорту стануть також поширені, як і газові і бензинові станції [4].

Веб-додаток – це клієнт-серверний додаток, (клієнтом є браузер, а в якості сервера виступає веб-сервер), при якому зберігання даних здійснюється головним чином на сервері, а обмін даними відбувається по мережі. З цього випливає, що для роботи з веб додатком користувачеві необхідний доступ до інтернету.

Відмінною особливістю веб-додатки є масштаб: одночасно їм може користуватися велика кількість людей. Одним з критеріїв вибору розробки клієнт-серверного додатка є той факт, що користувачі не залежать від операційної системи, тому веб-додатки кроссплатформенную. Одне з безперечних плюсів веб-додатків полягає в тому, що вони не вимагають установки на смартфоні. Тобто пам'яті на девайсе вони не займають (тільки трохи кеша), на відміну від мобільних додатків. Проте, необхідно відзначити, що в плані функціоналу, розробка мобільного застосування дає більше, ніж клієнт-

серверна розробка, але зате витрат і часу на неї теж потрібно більше. Недоліком клієнт-серверного рішення є його залежність від мережі. Якщо користувач встановлює мобільний додаток на телефон безпосередньо з магазину додатків (Google Play, App Store), він може користуватися ним в офлайн-режимі, то з веб-додатком так не вийде, – інтернет для нього потрібен завжди.

Також перевагою розробки клієнт-серверних додатків є те, що оновлювати їх можна завжди, в будь-який час, і це не викликає ніяких складнощів: після цього нова версія додатка буде доступна всім користувачам [5]. До переваг розробки мобільних додатків можна віднести наступне:

- великі можливості в розробці функціоналу, і в підсумку результат виходить більш якісним і зручним для користувача (при схожому функціонал);
- можна працювати в режимі офлайн – немає залежності від мережі (за винятком деяких випадків, коли, наприклад, для користування деякими можливостями програми необхідний інтернет);
- у магазини мобільних додатків заходять вже цільові користувачі, – не потрібно особливо витрачатися на seo-просування, як при клієнт-серверної розробці та мобільні додатки набагато краще оптимізовані під мобільні пристрої.
- весь вміст знаходиться вже всередині мобільного застосування, тобто вам не потрібно кожен раз завантажувати всі дані заново, як при роботі з веб-додатком;

Користувач зможе постійно надсилати їм інформацію про знижки, спеціальні пропозиції.

1.2 Характеристика об’єкту розробки

Метою дипломної роботи є розробка мобільного додатку для мережі заправок електрокарів. Це буде в допомогу водіям таких авто для швидкого пошуку вільної станції заправки з можливістю бронювання місця та часу.

Користувач за допомогою інструменту Google Map зможе обрати локацію для заправки у своєму місті. Крім цього, для мобільного додатку буде додано функцію безконтактної сплати за послуги з урахуванням можливого бонусу як для постійного клієнта.

1.3 Огляд аналогів

Щоб реалізувати проект на першому етапі розробки слід оглянути декілька аналогів для виявлення головних від'ємностей, які є суттєвими на даний час для користувача.

Так, першим аналогом було розглянуто додаток «UGV_chargers». Запорізька компанія «ІНФОКОМ ЛТД», вже відома, як перший розробник українського безпілотного автомобіля, підготувала готове рішення для мережі електрозаправних станцій. Мобільний додаток «UGV_chargers» дозволяє водієві електрокара в лічені секунди знайти зарядну станцію і заповнити енергію батареї.

За допомогою програми (рис. 1) користувач «зеленого авто» самостійно будує маршрут до обраної електрозаправки, починає і зупиняє зарядку за допомогою одного натискання, оплачує заправку онлайн і отримує допомогу від диспетчерської служби в режимі 24/7.

Головна перевага «UGV_chargers» – це робота з використанням білінгової системи. Після завершення зарядки, платіж здійснюється через мобільний онлайн-додаток за допомогою банківської карти, через термінал або RFID-карту, пов'язану з профілем користувача.

Крім того, дана система включає в себе моніторинг і диспетчеризацію на базі SCADA WinCC OA фірми Siemens. Це дозволяє проводити віддалене управління станціями, відстежувати їх статус, отримувати всі необхідні параметри і складати фінансову звітність [5].

Рисунок 1 – Функції мобільного додатку «UGV_chargers»

Другий аналог – мобільний додаток «OnCharge» (рис. 2). Він працює через протокол OCPP (Open Charge Point Protocol – європейський протокол, який використовується для організації зв'язку між зарядними станціями електротранспорту та центральною системою управління).

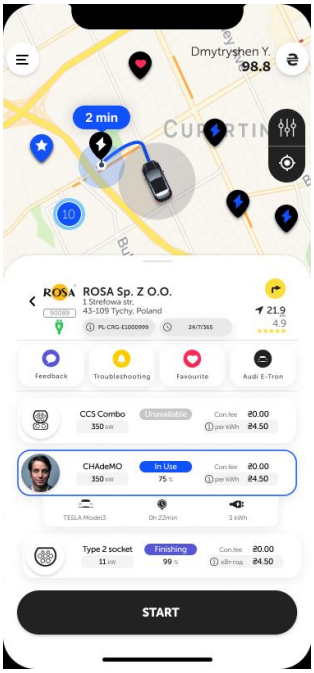


Рисунок 2 – Інтерфейс мобільного додатку «OnCharge»

«OnCharge» – це хмарний сервіс, який був розроблений як сполучна ланка між споживачем і оператором. За допомогою програми власник електрокара може знайти найближчі вільні зарядні станції, побудувати оптимальний маршрут до них, знайти інформацію про тарифи та програм лояльності. У додатку можна подивитися історію транзакцій і статистику по заправкам.

Відсутність логіну і паролю при вході в додаток, тільки прив'язка до номера телефону – дуже зручно для користувача. Також є доступ до зарядці по QR-коду і розрахунок щомиті, а не в кінці транзакції.

Користувач завжди в режимі реального часу бачить, скільки з його рахунку списується коштів, і яка кількість електроенергії він споживає.

Мобільний додаток «To-U» (рис. 3) займає топові місця у рейтингу власників електрокарів.

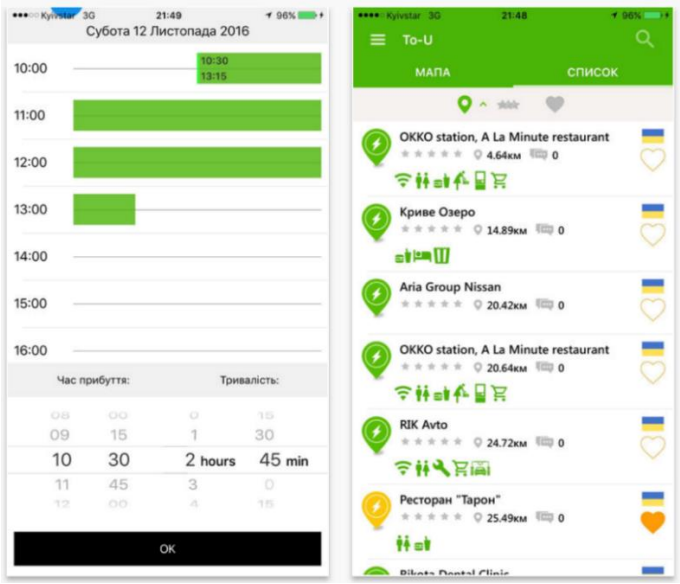


Рисунок 3 – Інтерфейс мобільного додатку «To-U»

У цьому додатку водій знайде перевірені і сертифіковані зарядні станції на кращих локаціях. За допомогою преміум-фільтру користувач не тільки зможе зарезервувати зарядну станцію на зручний час, а й отримати гарантію, що вона включиться саме для його типу авто.

Крім цього, є можливість забронювати сервіс, біля якого знаходиться зарядна станція для ЕМ: номер в готелі, столик в ресторані або, може бути, СПА. Коли зарядка закінчитися – користувачу прийде повідомлення. Використовуючи To-U, ви зможете найкращим чином спланувати свій маршрут, а наш алгоритм підкаже, на який локації вам буде найзручніше зробити зупинку для підзарядки. «To-U» – абсолютно безкоштовний додаток, яке дозволяє:

- ознайомитися з актуальною інформацією про публічну зарядці, сервісами на її локації і відгуками інших користувачів;
- знайти безкоштовну або платну, але найвигіднішу публічну зарядну станцію;
- забронювати зарядку на зручний час, зарезервувати сервіс на локації біля зарядної станції;
- прокласти і спланувати маршрут;
- зробити навігацію за допомогою Google Maps, Waze, Apple Maps;
- фільтрувати за різними категоріями та критеріям локації із зарядними станціями;
- отримувати бонуси і подарунки;
- оплачувати платні зарядні станції прямо в додатку (на партнерських станціях);
- отримувати оповіщення про сеанс зарядки, кількості отриманих кВт і іншу потрібну інформацію [6].

Мобільний додаток «Bosch» для швидкого пошуку зарядних станцій для електромобілів (рис. 4). Пошук зарядної станції, зарядка і оплата здаються простою справою, але в реальності для власників електромобілів все виявляється не так просто.

Сьогодні їм доводиться проводити багато часу, розбираючись з особливостями тарифів і способів оплати у різних операторів, а також в пошуках станції з потрібним типом роз'єму.

Щоб допомогти водіям оперативно знайти найближчу станцію і оплатити спожиту енергію, компанія Bosch об'єднала зусилля з одним з

найпопулярніших в Німеччині сервісів clever-tanken.de з пошуку найбільш вигідних цін на німецьких АЗС.

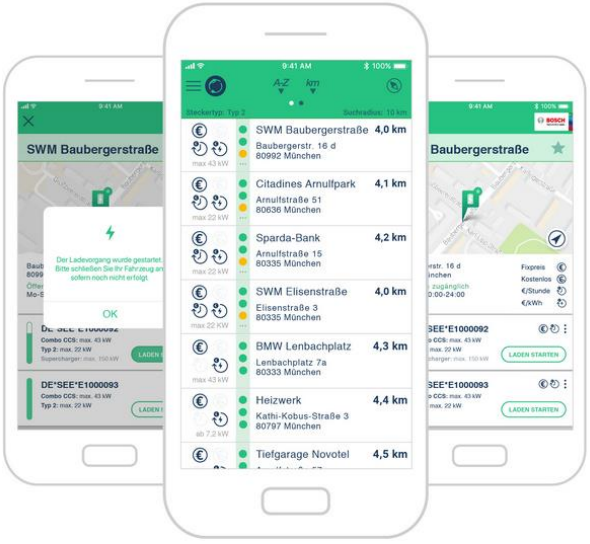


Рисунок 4 – Інтерфейс мобільного додатку Bosch «Clever Laden»

Функція «Розумна зарядка», в рамах якої авторами веб-проекту створено новий розділ сайту, де порівнюється вартість підзарядки електромобілів в різних регіонах і у різних операторів. Цей додаток інформує водіїв не тільки про ціни на бензин, дизельне паливо і скраплений газ, а й про розташування і цінах на енергію на зарядних станціях для електромобілів по всій Німеччині.

Вбудовані фільтри в додатку сортують місця зарядки по доступності, зарядної ємності та типам роз'ємів для підключення електромобілів різних виробників. Крім того, додаток оптимізує процес зарядки за допомогою єдиної системи доступу і оплати.

За «Clever Laden» стоїть потужна мережа IT-систем та розробників. Це дозволяє постійно оновлювати дані в режимі реального часу, щоб водії в будь-який момент часу могли отримати інформацію про кількість вільних місць на будь-який з вхідних в систему станцій. Крім цього, цей додаток постійно оновлюється. Функція реалізована за допомогою технології Bosch IoT Suite – пакетного хмарного рішення для Інтернету речей.

2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ РОЗРОБКИ

2.1 Visual Studio

Microsoft Visual Studio – повнофункціональна інтегроване середовище розробки (IDE) з підтримкою популярних мов програмування. Функціональність Visual Studio охоплює всі етапи розробки програмного забезпечення, надаючи сучасні інструменти для написання коду, проектування графічних інтерфейсів, збірки, налагодження і тестування програм. Можливості Visual Studio можуть бути доповнені шляхом підключення необхідних розширень.

Редактор коду Visual Studio підтримує підсвічування синтаксису, вставку фрагментів коду, відображення структури і пов'язаних функцій. Істотно прискорити роботу допомагає технологія IntelliSense – автозавершення коду під час введення.

Вбудований відладчик Visual Studio використовується для пошуку і виправлення помилок у вихідному коді, в тому числі на низькому апаратному рівні. Інструменти діагностики дозволяють оцінити якість коду з точки зору продуктивності і використання пам'яті.

Дизайнер форм Visual Studio незамінний при розробці програм з графічним інтерфейсом, допомагаючи спроектувати зовнішній вигляд майбутнього програми і роботу кожного елемента інтерфейсу. Крім цього, студія надає комплекс інструментів для автоматизації тестування додатків в частині перевірки роботи інтерфейсів, модульного і навантажувального тестування.

Для командних проектів Visual Studio пропонує підтримку групової роботи, дозволяючи виконувати спільне редагування і налагодження будь-якій частині коду в реальному часі, а в якості системи управління версіями використовувати Team Foundation або Git [7].

Основним розширенням файлу, асоційованим з Microsoft Visual Studio, є SLN – Visual Studio Solution File (Файл рішення Visual Studio), при відкритті якого в програму завантажуються всі дані і проекти, пов'язані із технічною характеристикою програмним рішенням.

За допомогою Visual Studio (рис. 5) можна розробляти:

– класичні додатки для комп'ютера під керуванням операційної системи windows;
– мобільні додатки (windows, ios, android);
– web-додатки;
– хмарні додатки;
– різні розширення для Office, Sharepoint, а також створення власних розширень для Visual Studio;
– ігри;
– бази даних Sql Server і Sql Azure [7].

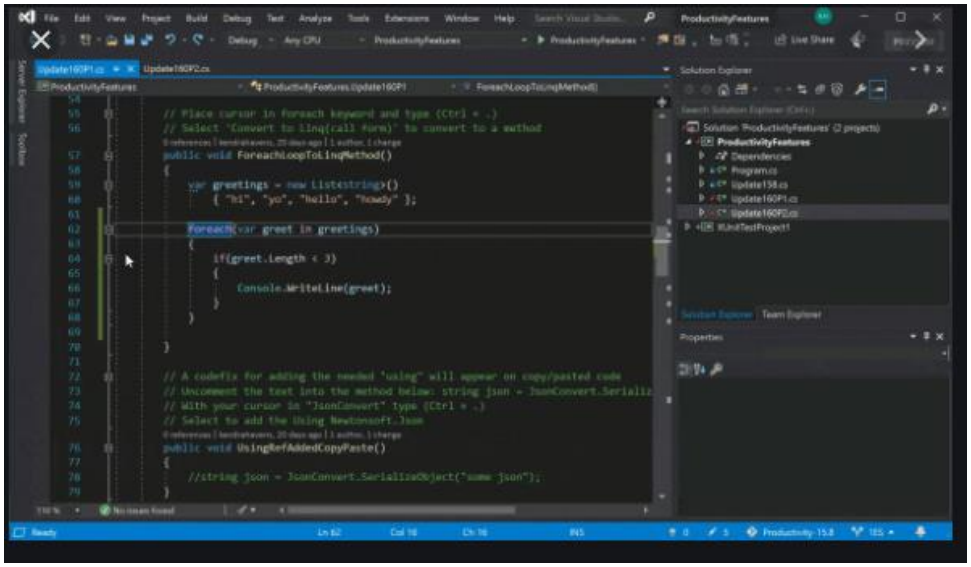


Рисунок 5 – Середа розробки Visual Studio

2.2 Xamarin і крос-платформна розробка

Розробники використовують багатоплатформовий фреймворк Xamarin в основному для створення невеликих проектів або прототипів. Американська компанія Xamarin заснована в 2011 році. Її розробники адаптували платформу .NET Framework, спочатку розраховану на роботу з Microsoft Windows, під інші мобільні платформи.

Переваги фреймворка:

1. Це інструмент .NET, тому він підходить тим, хто спеціалізується саме на .NET розробці. Наприклад, DD Planet: ми працюємо на цій платформі вже 17 років і стали використовувати Xamarin одними з перших на російському ринку.
2. Дозволяє використовувати велику кількість .NET бібліотек. У .NET величезний репозиторій бібліотек. Розробнику там важко чогось не знайти.
3. У розробці використовується мова C#, він дозволяє будувати об'єктно-орієнтовану модель і ізолювати рівні. Це означає, що можна вибудовувати архітектуру з упором на перевикористання коду.
4. UI налаштовується для кожної платформи окремо нативними інструментами.
5. В цілому на базі Xamarin можна вибудувати будь-яке архітектурне рішення. Ігри на ньому особливо не вдієш, але деякі роблять і це.

Основна перевага Xamarin для бізнесу полягає в тому, що він дозволяє істотно (майже в 2 рази) скоротити терміни і витрати на розробку при збереженні всього необхідного функціоналу та якості. Тобто на виході виходить таке ж додаток, як за допомогою розробки на нативних мовах програмування, тільки дешевше і швидше [8].

Головна особливість масштабних програм: складність дизайну і функціональності сприяє збільшенню кількості коду. В результаті зростають час на розробку і бюджети. Адже якщо говорити про нативной розробці, компанія стикається з необхідністю підключити до роботи дві команди розробників: окремо на iOS і на Android, і фактично платить за два додатки замість одного. В цьому випадку Кросплатформені фреймворки, що дозволяють писати код одного разу і перевикористати його на різних платформах, стають важливою альтернативою.

Так, можливість перевикористати загальний код – головний аргумент на користь Xamarin. При розробці великих додатків ми часто стикаємося з тим,

що великі шматки бізнес-логіки доводиться перетягувати між різними частинами програми. Xamarin хороший тим, що дозволяє користуватися загальною кодовою базою Core проекту і перетягувати великі пласти логіки з екрану на екран.

В середньому з його допомогою до 90% загального коду програми без змін програмісти можуть перевикористати на різних платформах. Розробку можна вести під Windows або MacOS і компілювати в нативні пакети додатків: в .apk для Android або .ipa для iOS.

Xamarin заснований на open-source реалізації платформи .NET – Mono. Ця реалізація включає в себе власний компілятор C#, середу виконання, а так само основні .NET бібліотеки. Мета проекту – дозволити запускати програми, написані на C#, на операційних системах, відмінних від Windows – Unix-системах, Mac OS і інших. [9].

З точки зору виконання додатків між iOS і Android є одне ключове відмінність – спосіб їх попередньої компіляції. Як показано на рисунку 6, для виконання додатків в Android використовується віртуальна Java-машина Dalvik.

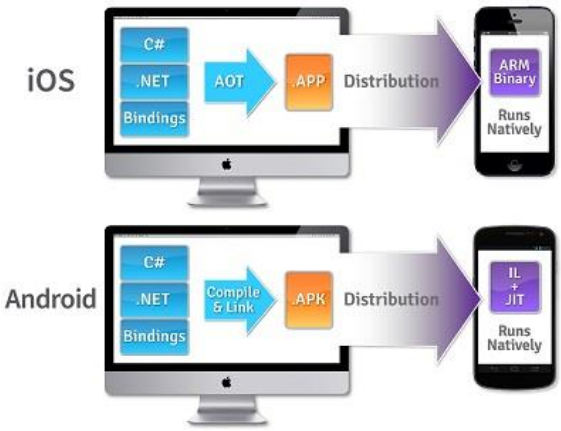


Рисунок 6 – Компіляція у iOS и Android

Нативні додатки, які пишуться на Java, компілюються в якийсь проміжний байт-код, який інтерпретується Dalvik`ом в команди процесора в момент виконання програми (тобто. Аналогічно тому, як працює CLR в .NET). Це Just-

in-time компіляція (компіляція на льоту). В iOS використовується інша модель компіляції – Ahead-of-Time (компіляція перед виконанням). Xamarin враховує ця різниця, надаючи окремі компілятори для кожної з цих платформ, які дозволяють на виході отримувати справжні, нативні додатки, які виконуються поза контекстом браузера і можуть використовувати всі апаратні і програмні ресурси платформи [10].

Для iOS ніякої віртуальної машини немає і програмний код повинен бути просто заздалегідь скомпільований в машинний. Для цієї мети використовується AOT компілятор Mono.

Xamarin поєднує всі можливості вбудованих платформ (рис. 7) і додає кілька потужних особливостей: повна прив'язка до базових SDK, сучасні мовні конструкції, бібліотека базових класів (BCL), сучасна IDE, підтримка мобільного платформ.

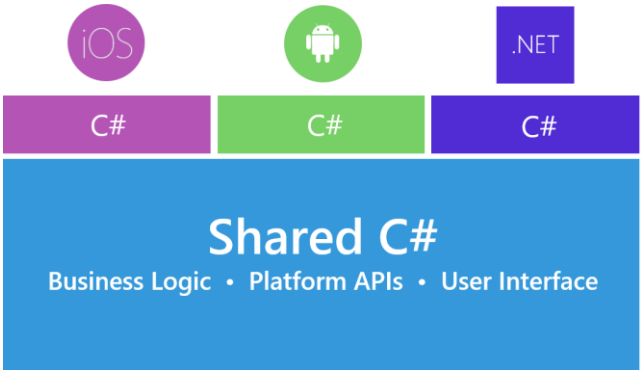


Рисунок 7 – Вбудовані платформи Xamarin

2.3 Мова програмування C# і платформа .NET

На сьогоднішній момент мова програмування C# один з найпотужніших, що швидко розвиваються і затребуваних мов в IT-галузі. На даний момент на ньому пишуться найрізноманітніші програми: від невеликих десктопних програмок до великих веб-порталів і веб-сервісів, які обслуговують щодня мільйони користувачів.

C# є об'єктно-орієнтованим і в цьому плані багато перейняв у Java і C++. Наприклад, C# підтримує поліморфізм, успадкування, перевантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і розширюваних додатків. Коли говорять C#, нерідко мають на увазі технології платформи .NET (Windows Forms, WPF, ASP.NET, Xamarin). Фреймворк .NET представляє потужну платформу для створення додатків [11].

Підтримка декількох мов. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), завдяки чому .NET підтримує кілька мов. При компіляції код компілюється в збірку спільною мовою CIL (Common Intermediate Language). Кросплатформеність. .NET є яку переносять платформою (з деякими обмеженнями). Наприклад, остання версія платформи на даний момент – .NET 5 підтримується на більшості сучасних ОС Windows, MacOS, Linux. Використовуючи різні технології на платформі .NET (рис. 8), можна розробляти програми на мові C# для самих різних платформ – Windows, MacOS, Linux, Android, iOS, Tizen.

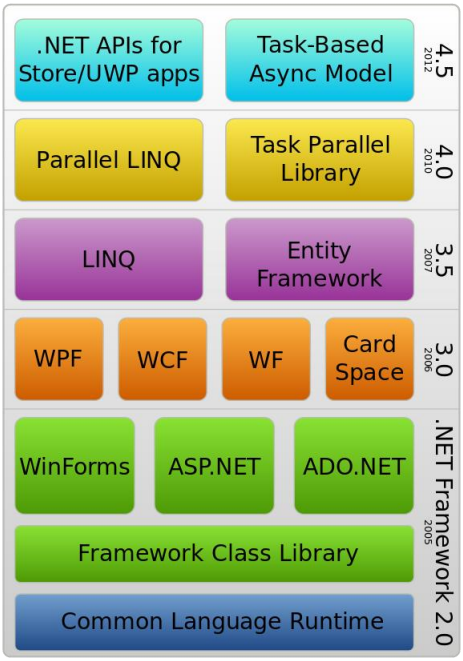


Рисунок 8 – Стек технологій .NET Framework

Потужна бібліотека класів. .NET представляє єдину для всіх підтримуваних мов бібліотеку класів. І яке б додаток ми не збиралися писати на C# – текстовий редактор, чат або складний веб-сайт – так чи інакше ми задіємо бібліотеку класів .NET.

Різноманітність технологій. Загальномовне середовище виконання CLR і базова бібліотека класів є основою для цілого стека технологій, які розробники можуть задіяти при побудові тих чи інших додатків. Наприклад, для роботи з базами даних в цьому стеку технологій призначена технологія ADO.NET і Entity Framework Core.

Для побудови графічних додатків з багатим насиченим інтерфейсом – технологія WPF і UWP, для створення більш простих графічних додатків – Windows Forms. Для розробки мобільних додатків – Xamarin. Для створення веб-сайтів і веб-додатків – ASP.NET і інше [11].

2.4 БД SQLite

SQLite – це вбудована бібліотека в якій реалізовано багато з стандарту SQL 92. Її домаганням на популярність є як власне сам движок бази, так і її інтерфейс (точніше його движок) в межах однієї бібліотеки, а також можливість зберігати всі дані в одному файлі (рис. 9).

Однак, на практиці, SQLite нерідко виявляється в 2-3 рази (і навіть більше) швидше. Таке можливо завдяки високо спорядкованій внутрішньої архітектурі і усунення необхідності в з'єднаннях типу «сервер-клієнт» і «клієнт-сервер».

Все це, зібране в один пакет, лише небагато чим більше за розміром клієнтської частини бібліотеки MySQL, є вражаючим досягненням для повноцінної бази даних.

Використовуючи високо ефективну інфраструктуру, SQLite може працювати в крихітному обсязі виділеної для неї пам'яті, набагато меншому, ніж в будь-яких інших системах баз даних (БД).

Це робить SQLite дуже зручним інструментом з можливістю використання практично в будь-яких завданнях покладених на базу даних.

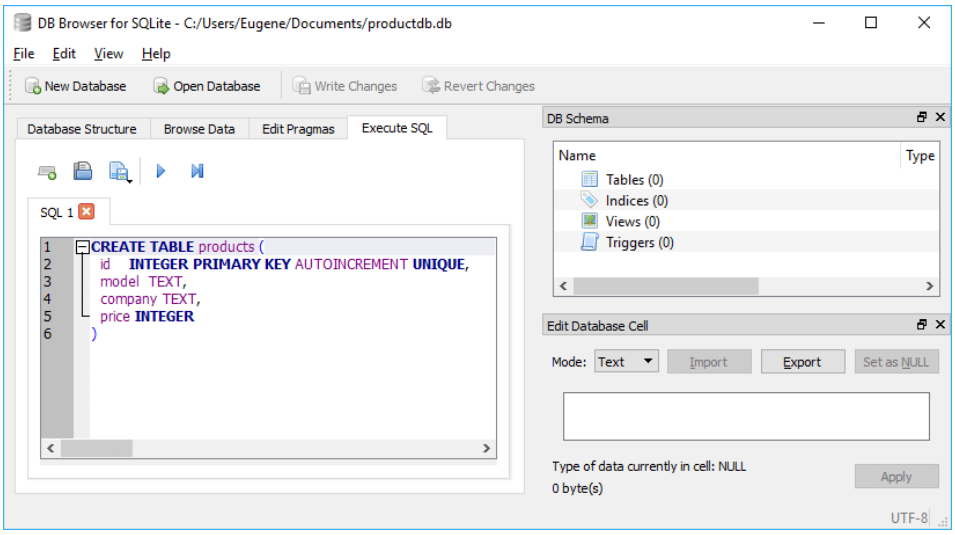


Рисунок 9 – Приклад роботи з SQLite

Переваги:

– популярна;

– надійна;

– консольна утиліта для роботи з базами;

– відкриті вихідні коди.

Недоліки:

– немає збережених процедур;

– немає вбудованої підтримки unicode;

– не підходить для додатків, які часто звертаються до бази;

– sqlite підтримує динамічне тіпізірованія даних [12].

В якості протоколу обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виповнюється програма [13].

SQLite – це система управління базами даних, відмінною рисою якої є її вбудовуваність в додатки. Це означає, що більшість СКБД є самостійними додатками, взаємодія з якими організовано за принципом клієнт-сервер. Програма-клієнт надсилає запит на мові SQL, СКБД, яка в тому числі може перебувати на віддаленому комп'ютері, повертає результат запиту.

SQLite використовує динамічне тіпізірованія даних. Це означає, що тип стовпчика не визначає тип значення, що зберігається в цьому полі записи. Тобто в будь-який стовпець можна занести будь-яке значення. Тип стовпця визначає як порівнювати значення (треба ж їх привести до єдиного типу при порівнянні, скажімо, всередині індексу). Але не зобов'язує заносити значення саме такого типу в стовпець [14].

2.5 Бібліотека QRCoder

QR-код – це дуже ефективна двовимірна (2D) символіка штрих-коду, яка використовує невелику площу квадратних модулів з унікальним малюнком периметра, що допомагає сканера штрих-коду визначати місце розташування осередків і декодувати символ QR-коду. Можуть бути закодовані символи, числа, текст і фактичні байти даних, включаючи символи Юнікоду і зображення. QR означає "Швидкий відгук". Він був створений японською корпорацією Denso-Wave в 1994 році і спрямований на декодування вмісту на високій швидкості. На сьогодні QR-код використовується в мобільних телефонах для полегшення введення даних.

QR-код може зберігати до 7089 цифрових, 4296 буквено-цифрових, 2953 довічних, 1817 ієрогліфів або китайських символів в одному символі штрих-коду і включає код корекції помилок (ECC), який дозволяє безпомилкове читання, навіть якщо символ був частково втрачений або знищений [15].

QRCoder – це проста бібліотека, написана на C# .NET, яка дозволяє створювати QR-коди. Він не має залежностей від інших бібліотек і доступний у форматі .NET Framework та .NET Core PCL на NuGet.

QRCode також можна роздрукувати на візитній картці або показати на будь-якому дисплеї, який потім можна зафіксувати мобільним телефоном за умови, що мобільний телефон має програмне забезпечення для зчитування QRCode.

Бібліотека QRCode забезпечує функції:

- кодування вміст у зображення QR-коду, яке можна зберегти у форматах JPEG, GIF, PNG або Bitmap;
- розшифовка зображення QR-коду [16].

3 ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ

3.1 Проектування UI

Щоб створити ефективний мобільний призначений для користувача інтерфейс, дизайнери повинні враховувати такі аспекти, як визначення користувачів і їх спосіб мислення, розуміння шаблонів проектування призначеного для користувача інтерфейсу, вміти працювати з різними інструментами дизайну і придумувати унікальну естетику, що відповідає вимогам проекту.

Ринок додатків рясніє додатками, які залишаються непоміченими або швидко забуваються, тому, починаючи процес проектування або, що ще краще, під час обговорення можливої стратегії з клієнтом, важливо намітити, як додаток буде грати в їх користь [17].

UX (англ. User experience) дословно означає «досвід користувача». У більш широкому змісті це поняття про весь досвід, який отримує користувач при взаємодії з сайтом або додатком. UX-дизайн відповідає за функції, адаптивність продукту і то, які емоції він виказує у користувачів. Чем поняття інтерфейсу, тим легше користувач отримує результат і робить цільову дію.

UX-дизайн – це проектування інтерфейсу на основі досліджень користувацького досвіду та введень.

Основна роль у програмі – це UI/UX design. Дизайн інтерфейсу мобільного додатка відповідає за те, як виглядає програма, як вона реагує та як вона функціонує. Необхідно розробити додаток, який буде мати багато операцій та швидку реакцію. Фокус інженера-програміста повинен бути зосереджений на детальному аналізі інтерфейсу користувача [18].

UI-дизайн – процес візуалізації прототипу, який розроблений на основі користувацького досвіду та дослідження цільової аудиторії. UI-дизайн включає в себе роботу над графічною частиною інтерфейсу: анімація, ілюстрація, кнопки, меню, слайдери, фотографії та шрифти.

«Mobile App Wireframe» – призначений для прототипування графічних інтерфейсів. У ній можна створити інтерактивні прототипи веб-сайтів та додатки як для настільних комп'ютерів так і для смартфонів, а також інших мобільних пристроїв.

Автори позиціонують «Mobile App Wireframe» як інструмент для професіоналів і у них на цьому є всі основи. Для зберігання даних проекту, використовуються текстові файли . Також для «Mobile App Wireframe» є достатньо обширна бібліотека макетів, шаблонів, виджетів та іконок (рис. 10).

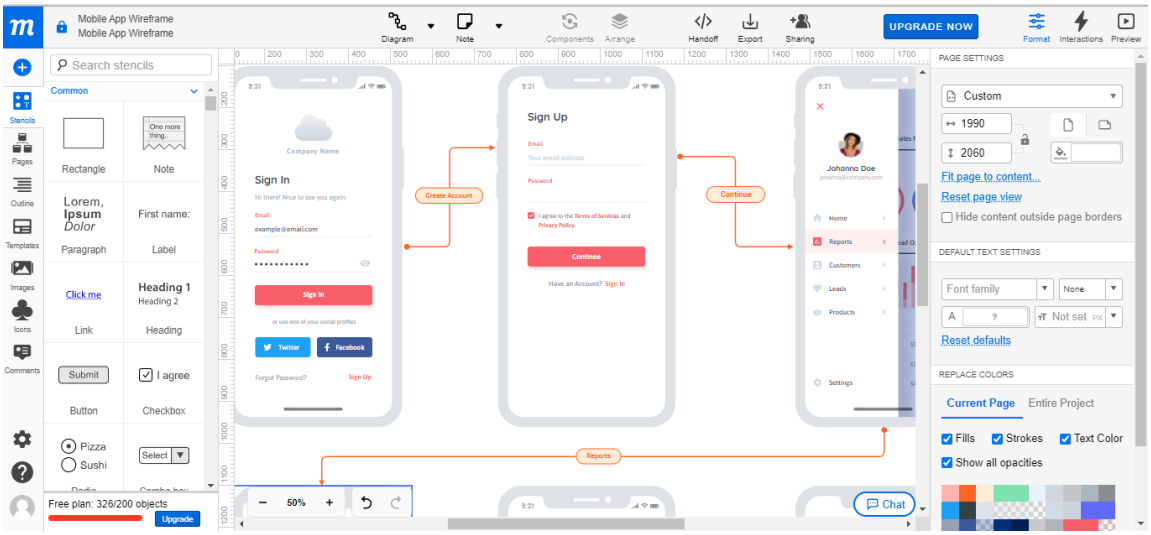


Рисунок 10 – Приклад робочого вікна «Mobile App Wireframe»

Мобільний додаток буде реалізований у вигляді сукупності декількох сторінок, між якими обов’язково повинна бути навігація. Перші три сторінки представляють собою:

– стартова сторінка авторизації (або регістрації) користувача – необхідно заповнити два поля (ім’я та номер телефону);
– сторінка з картою міста відповідно до геолокації користувача на даний час;
– сторінка обраної локації з можливістю бронювання ліміту часу для заправки.

Так, макет для першої сторінки додатку представлено на рисунку 11:

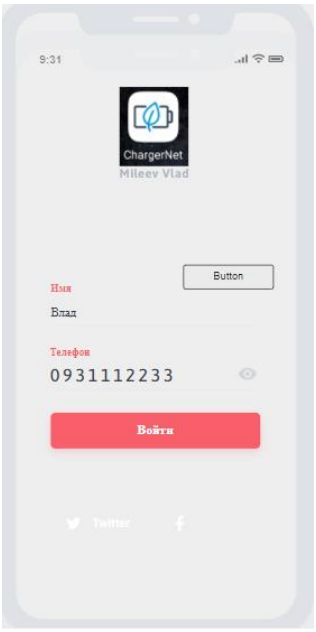


Рисунок 11 – Макет сторінки авторизації

Макети наступних двох сторінок та послідовний перехід між ними зображено на рисунку 12:

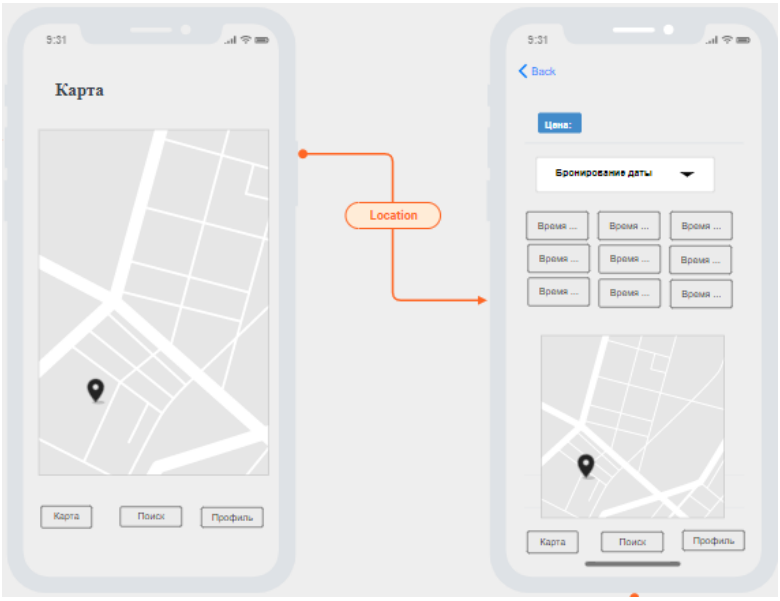


Рисунок 12 – Макет сторінок локації заправок

Наступний макет демонструє сторінки з викликом функцій пошуку локацій та після їх підстредження перехід до сторінки персонального кабінету користувача. В ньому буде розміщено інформації з QR-Code, сумою послуг та перелік локацій, якими користувався водій (рис. 13):

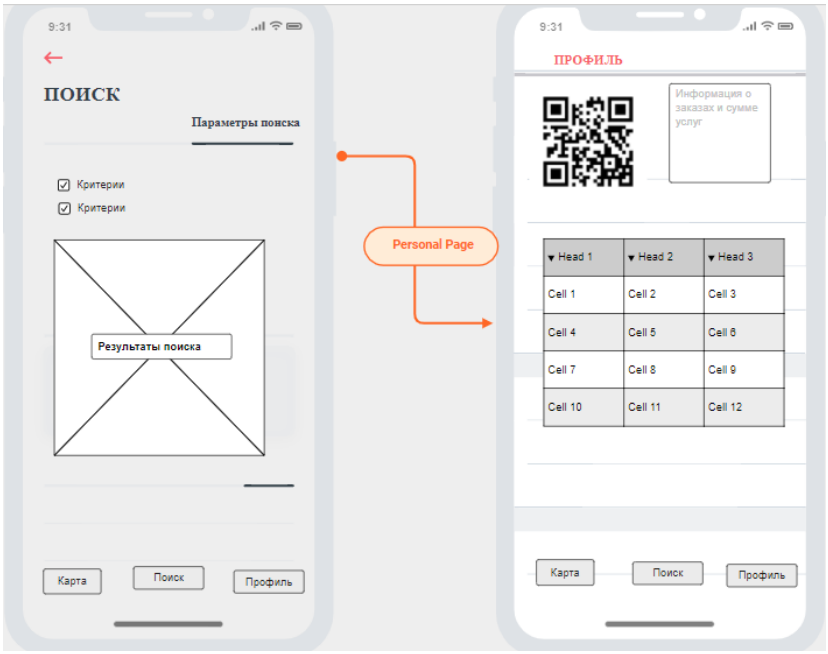


Рисунок 13 – Макет пошукової сторінки та персонального кабінету

3.2 Діаграма Use-Case

Візуальне моделювання в UML можна представити як деякий процес поуровневого спуску від найбільш загальної і абстрактної концептуальної моделі вихідної системи до логічної, а потім і до фізичної моделі відповідної програмної системи.

Для досягнення цих цілей спочатку будується модель у формі так званої діаграми варіантів використання (use case diagram), яка описує функціональне призначення системи або, іншими словами, те, що система буде робити в процесі свого функціонування. Діаграма варіантів використання є вихідним концептуальним уявленням або концептуальною моделлю системи в процесі її

проектування і розробки. Розробка діаграми варіантів використання переслідує мети: визначити загальні межі і контекст модельованої предметної області на початкових етапах проектування системи та сформулювати загальні вимоги до функціонального поведінки проектованої системи.

Діаграма варіантів використання являє собою найбільш загальну концептуальну модель складної системи, що є вихідною для побудови всіх інших діаграм.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою так званих варіантів використання. При цьому актором (actor) або дійовою особою називається будь сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на моделюючу систему так, як визначить сам розробник [18].

У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає акторові. Іншими словами, кожен варіант використання визначає деякий набір дій, що чиниться системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів з системою.

У самому загальному випадку, діаграма варіантів використання є граф спеціального виду, який є графічної нотацією для представлення конкретних варіантів використання, акторів, можливо деяких інтерфейсів, і відносин між цими елементами.

При цьому окремі компоненти діаграми можуть бути укладені в прямокутник, який позначає проектовану систему в цілому. Слід зазначити, що відносинами даного графа можуть бути тільки деякі фіксовані типи взаємозв'язків між акторами і варіантами використання, які в сукупності описують сервіси або функціональні вимоги до моделюється системі [19].

Приклад діаграми використання для мобільного додатку представлено на рисунку 14.

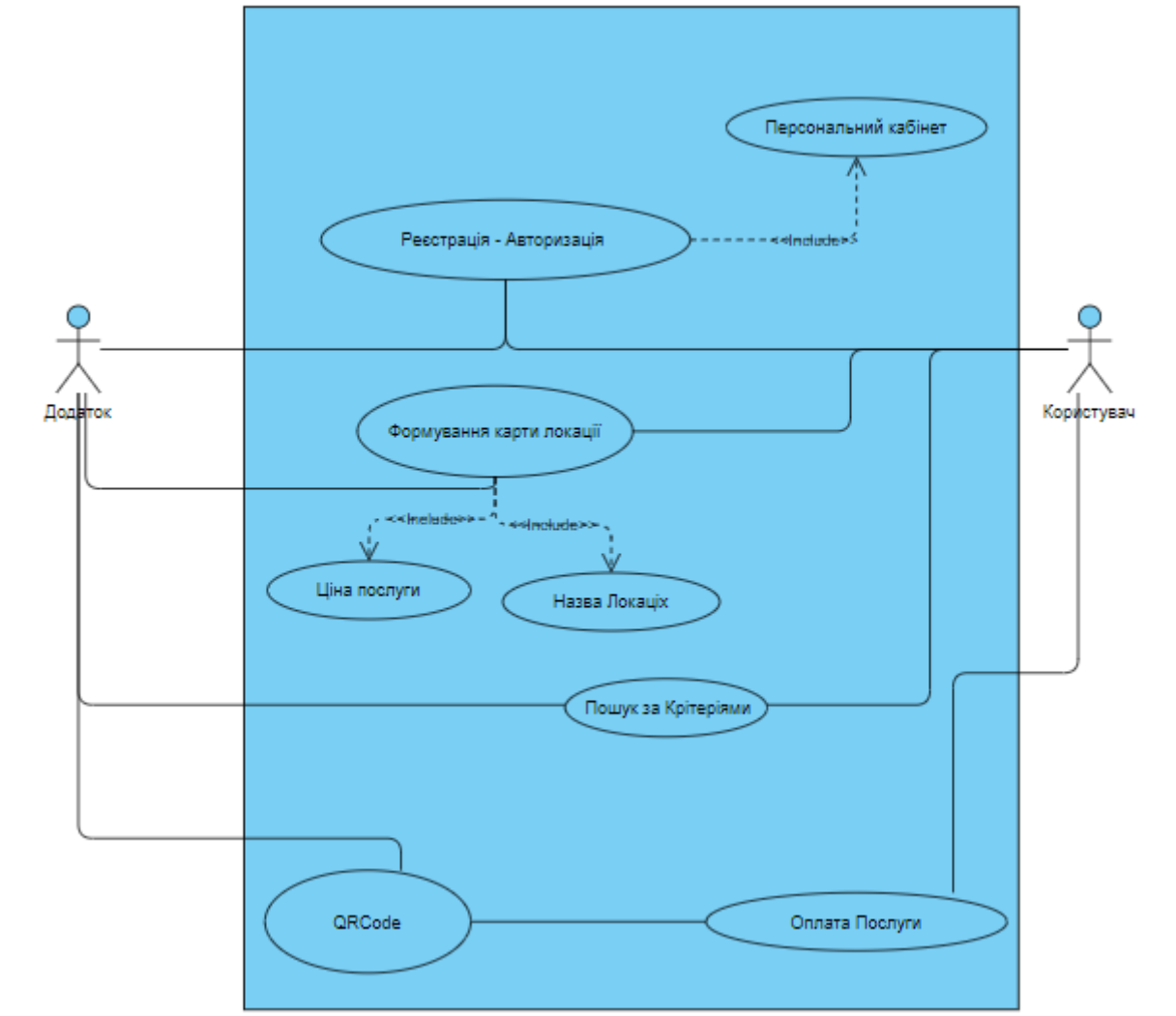


Рисунок 14 – Діаграма варіантів використання

4 ОПИС РОЗРОБКИ ПРОЕКТУ

4.1 Структура проекту

Solution Items – включає до себе два компонента: локальну базу даних типу SQLite та скрипт для первинної ініціалізації системи баз даних (БД).

SQLite – полегшена реляційна система керування базами даних. Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми.

Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок (рис. 15).

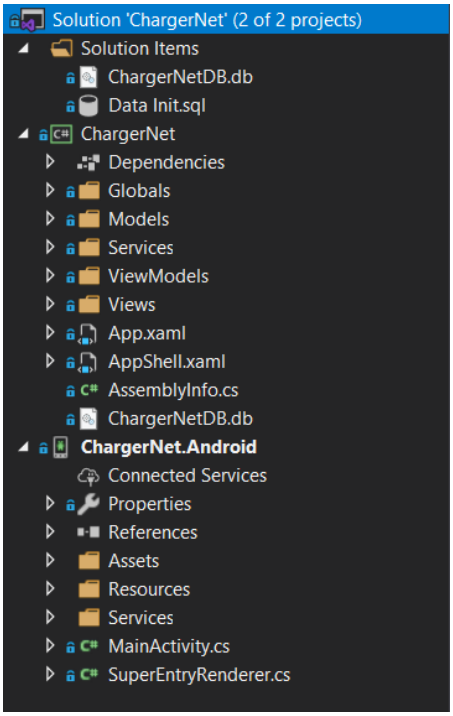


Рисунок 15 – Структура проекту

В випадку цього проекту файл БД називається «ChargerNetDB.db» та складається с таких таблиць, як:

– «Charger»;
– «User»;
– «UserCharger».

Кожна з таблиць має одноймені назви моделей в проекті написаному на платформі Xamarin використовуючи мову програмування C#.

Таблиця «Charger» презентує сутність зарядної станції, яку користувач може забронювати використовуючи свої параметри. Параметри користувачів в свою чергу зберігаються в таблиці «User».

Для зв’язку двох попередніх таблиць в сутність бронювання конкретної зарядної станції на конкретного користувача будуть використовуватися таблиця «UserCharger», в котрій будуть зберігатися зовнішні ключи на таблицю користувача та зарядного пристрою.

Для розуміння даних які зберігаються в кожній з таблиць необхідно розібрати базову модель даних BaseModel. Цей клас презентує базовий шаблон який може бути розширений в майбутньому для додавання службової інформації, але на початку розробки в ньому розміщено лише поле ідентифікатора Id.

Розглянемо таблицю зарядної станції (ЗС) або «Charger», що представлено на рисунку 16.

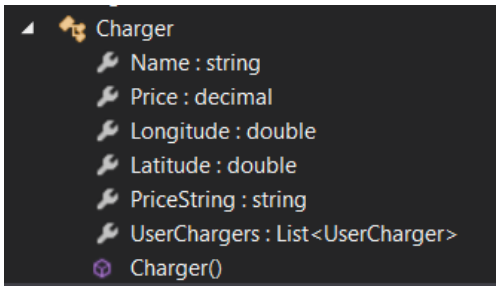


Рисунок 16 – Структура таблиці «Charger»

Дана таблиця складається з 6 основних параметрів: ім’я, ціна, широта та довгота, ціна у виразі рядка та перелік запитів на бронювання.

Слід зазначити, що ціна зберігається в спеціальному форматі створеному для роботи з грошима. А інтерпретований рядок для ціни слугує для передачі даних на інтерфейс мобільного пристрою. Таблиця User», яка містить в собі дані користувача, представлено на рисунку 17.

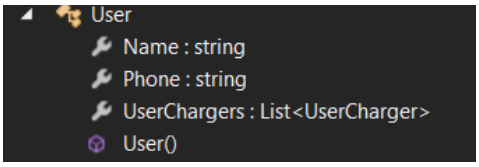


Рисунок 17 – Структура таблиці «User»

В цьому випадку бачимо лише необхідні параметри для ідентифікації користувача, а саме: його телефон та ім’я, а також перелік заброньованих на його ім’я станцій. Таблиця «UserCharger» (рис. 18) зберігає інформацію щодо бронювання станції на конкретний час.

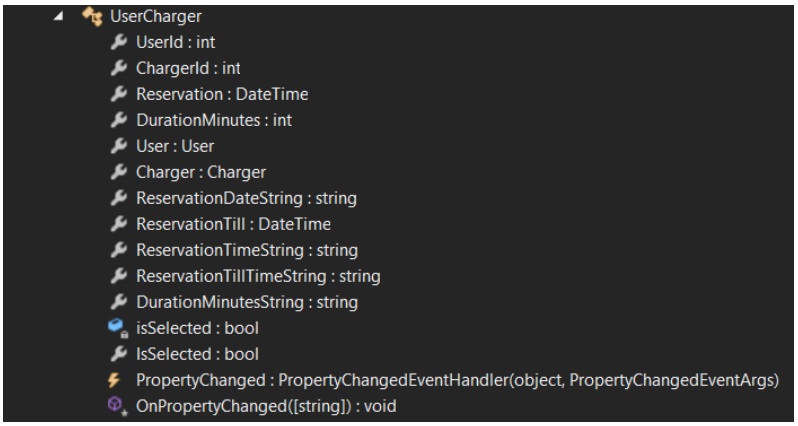


Рисунок 18 – Структура таблиці «UserCharger»

Як видно з переліку характеристик – ця сутність включає в себе два зовнішніх ключи, які були розглянуті в попередніх обґрунтуваннях. Крім цього в

цьому класі вказано довжина у хвилинах (на скільки заброньовано) та дату та час з якого можна буде користуватися використовуючи QR-код.

Усі характеристики мають інтерпретацію у вигляді рядків, для відображення в коректному вигляді на пристроях. Допоміжними даними також будуть виступати isSelected та метод OnPropertyChanged. Перший з них вказує інтерфейсу чи було обрано станцію для зарядки, а другий сигналізує коли будь-яка характеристика була змінена ззовні.

Data Init.sql – проводить ініціалізацію даними, для того щоб проводити перевірку проекту. ChargerNet – назва базового проекту в рішенні, в котрому зберігається платформо-незалежний код. Складається даний проект з таких директорій:

- Globals;
- Models;
- Services;
- ViewModels;
- Views;
- App.xaml/AppShell.xaml.

Перша директорія Globals використовується для зберігання глобальної інформації та статичних класів які дозволяють зберігати її та використовувати з будь-якого шара програми.

Наступна (Models) зберігає дані, як було зазначено вище для роботи з базою даних, фактично це прямий мапінг моделі БД в модель об’єктів C#. Опис сервісів проекту: сховище Services становить значну частину – воно зберігає сервіси для роботи з даними та їх метаданими. Складається програма з такого переліку сервісів (рис. 19).

Слід зазначити, що кожен файл, котрий починається з великої літери «I», є інтерфейсом для використання шаблона «ін’єкція залежності» (Dependency Injection), котрий необхідний для більш тонкого налаштування процесів розробки та споживання сервісів.

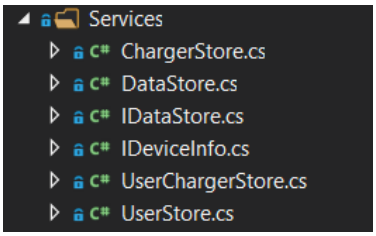


Рисунок 19 – Сервіси проекту

Як бачимо маємо сервіси для роботи з сховищами даних (прямий мапінг з однойменими таблицями розташованими в локальній базі даних):

- DataStore – базовий об’єкт сервісу, від нього має наслідуватися кожен з сервісів;
- ChargerStore;
- UserChargerStore;
- UserStore.

Нижче зазначений інтерфейс кожного з сервісів, котрі призначені для роботи з конкретною таблицею. Для приклада слід оглянути декілька з методів:

```
public interface IDataStore<T>
{
    SQLiteAsyncConnection AsyncConnection { get; }
    SQLiteConnection Connection { get; }
    string ConnectionString { get; }
    bool AddItem(T item);
    Task<bool> AddItemAsync(T item);
    bool DeleteItem(T item);
    Task<bool> DeleteItemAsync(T item);
    T GetItem(int id);
    Task<T> GetItemAsync(int id);
    bool Exists(int id);
    Task<bool> ExistsAsync(int id);
    IEnumerable<T> GetItems();
    Task<IEnumerable<T>> GetItemsAsync();
    bool UpdateItem(T item);
    Task<bool> UpdateItemAsync(T item);
}
```

Перші три параметри: AsyncConnection, Connection, ConnectionString, призначені для роботи з базою даних, та використовуються для передачі можливості з'єднання з провайдером даних, в нашому випадку провайдером даних є SQLite Net Extensions, написана бібліотека для роботи з локальними БД.

Наступні методи такі як:

- AddItem – додавання компонента в таблицю;
- AddItemAsync – асінхроне виконання попереднього методу;
- DeleteItem – видалення рядка даних зі сховища;
- DeleteItemAsync – асінхрона операція видалення;
- GetItem – отримання компонента за його ідентифікатором;
- GetItemAsync – асінхроний запит на отримання;
- Exists – перевірка наявності запису;
- ExistsAsync – асінхроний метод перевірки наявності;
- GetItems – отримання усіх можливих даних;
- GetItemsAsync – варіант отримання асінхроних даних;
- UpdateItem – оновлення елементу;
- UpdateItemAsync – оновлення елементу асінхроного типу.

Кожен з методів націлений на виконання принципу CRUD (Create-Read-Update-Delete).

Розглянемо усі залежності, який проект включає в себе (рис. 20):

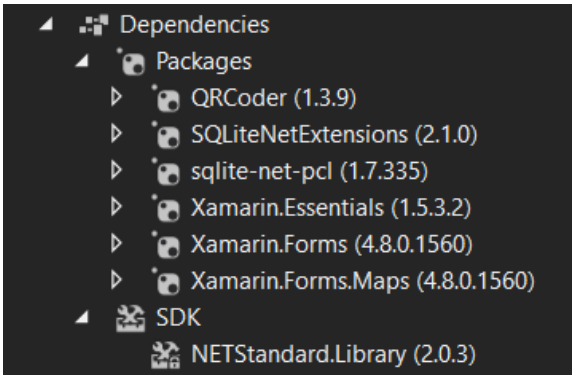


Рисунок 20 – Задіяні пакети для проекту

Xamarin – це платформа з відкритим кодом для створення сучасних та продуктивних додатків для iOS, Android та Windows за допомогою .NET. Xamarin – це абстракційний рівень, який управляє зв'язком спільного коду з базовим кодом платформи. Xamarin працює в керованому середовищі, яке забезпечує такі зручності, як виділення пам'яті та збір сміття.

Xamarin дозволяє розробникам розподіляти в середньому 90% своїх програм між платформами. Цей шаблон дозволяє розробникам писати всю свою бізнес-логіку однією мовою (або повторно використовувати існуючий код програми), але досягати природних характеристик, вигляду та відчуття на кожній платформі.

Елемент керування Map (рис. 21) – це міжплатформний вигляд для відображення та анотування карт.

Рисунок 21 – Приклад додатку з елементом керування Map

Він використовує власний контроль карт для кожної платформи, забезпечуючи швидкий та звичний досвід роботи з картами для користувачів.

Карту можна відобразити, додавши її до макета або сторінки:

```
<ContentPage ...
             xmlns:maps="clr-
```

```
 namespace:Xamarin.Forms.Maps;assembly=Xamarin.Forms.Maps">
     <maps:Map x:Name="map" />
 </ContentPage>
```

Еквівалентний код C# представлено нижче:

```
using Xamarin.Forms;
using Xamarin.Forms.Maps;
namespace WorkingWithMaps
{
    public class MapTypesPageCode: ContentPage
    {
        public MapTypesPageCode)
        {
            Map map = new Map);
            Content = map;
        }  }}
```

Також розглянемо одну з найважливіших частин – це ViewModel, шар який забезпечує роботу програми:

```
public class BaseViewModel: INotifyPropertyChanged
{
    bool isBusy = false;
    public bool IsBusy
    {
        get {return isBusy;}
        set {SetProperty(ref isBusy, value);}
    }
    string title = string.Empty;
    public string Title
    {
        get {return title;}
        set {SetProperty(ref title, value);}
    }
```

Метод зчитування номеру телефона користувача (обов’язкове поле при авторизації):

```
public User User => DependencyService.Get<IDataStore<User>>)
                        .GetItems()
                        .FirstOrDefault(x => x.Phone == Var-
iables.CurrentUser.Current.Phone);
    protected  bool  ResetProperty([CallerMemberName]  string
```

```
propertyName = "", Action onChanged = null)
    {
        onChanged?.Invoke();
        OnPropertyChanged(propertyName);
        return true;
    }
    protected bool SetProperty<T>(ref T backingStore, T
value,
        [CallerMemberName] string propertyName = "",
        Action onChanged = null)
    {          if (EqualityComparer<T>.Default.Equals(back-
ingStore, value))
            return false;
        bakingStore = value;
        onChanged?.Invoke();
        OnPropertyChanged(propertyName);
        return true;  }
```

Для автоматичного оновлення даних на формі:

```
#region INotifyPropertyChanged
    public event PropertyChangedEventHandler Property-
Changed;
    protected void OnPropertyChanged([CallerMemberName]
string propertyName = "")
    {
        var changed = PropertyChanged;
        if (changed == null)
            return;
        changed.Invoke(this, new PropertyChangedEven-
tArgs(propertyName));
    }
    #endregion
 }
```

Ця базова модель представляє методи для оновлення даних які були змінені ззовні користувачем, та дозволяє сповіщувати інші компоненти системи. Наприклад, при натискані на мапу, повинно відобразитись значки на ній, саме завдяки цьому підходу програма може показувати інформацію та сповіщати інші компоненти про цю подію.

В даному випадку використовується патерн MVVM. А для роботи з інтерфейсом користувача використовується технологія XAML.

4.2 Керівництво користувача

Для запуску додатку використовується іконка з логотипом (рис. 22):

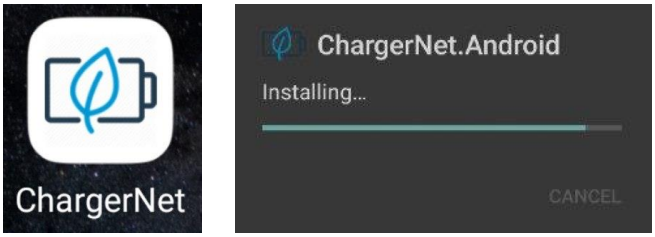


Рисунок 22 – Запуск мобільного додатку «ChargerNet»

На першому етапі використання додатку користувачу необхідно пройти авторизацію або зареєструвати свої дані: ім’я та телефон, як показано на рисунку 23:

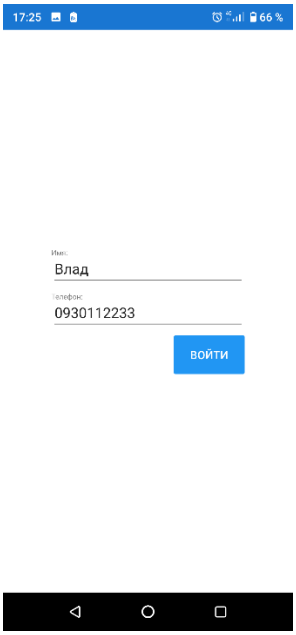


Рисунок 23 – Форма авторизації/реєстрації користувача

Після авторизації на екрані користувача відображається карта місця з локаціями зарядних станцій, як показано на рисунку 24.

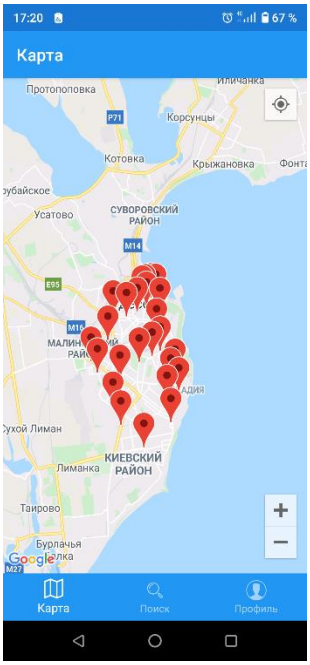
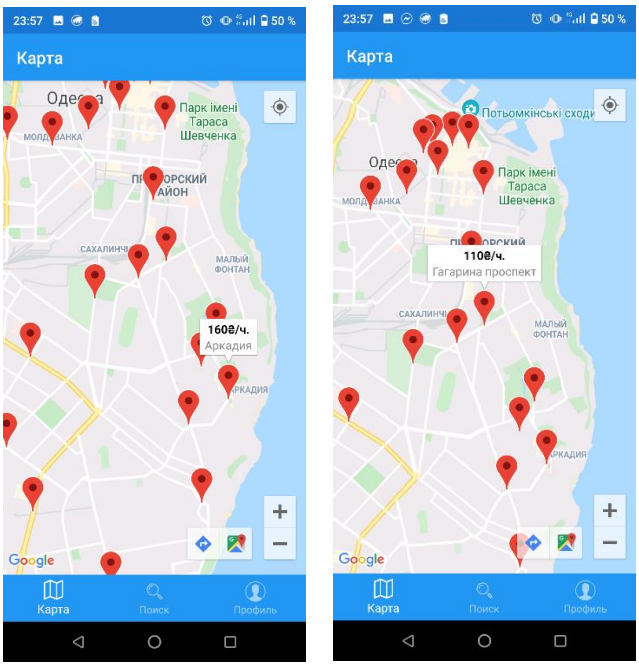


Рисунок 24 – Карта доступних зарядних станцій для обраного місця

На наступному кроці слід обрати одну з локацій, які будуть представлені як окремі кнопки на екрані додатку. Для кожної локації прописується назва вулиці та віддаленість від клієнта (рис. 25).



Римунок 25 – Приклади «кнопок» для переміщення до локації

Щоб ознайомитись у додатку з детальною інформацією щодо обраної локації слід перейти через посилання. Далі користувачу слід обрати ліміт часу, який йому буде необхідно для зарядки електрокару. За замовчуванням весь робочий час поділено на періоди у тридцять хвилин – це мінімальний ліміт, який потрібно для таких авто (рис. 26).

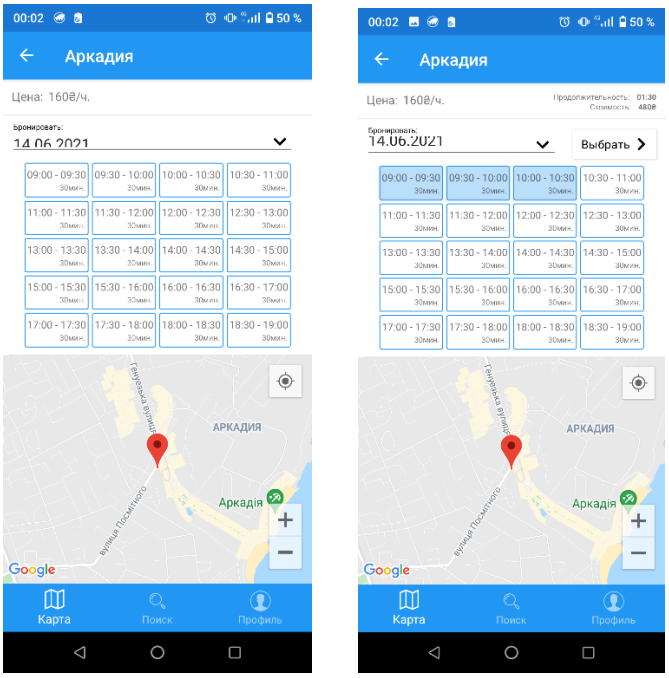


Рисунок 26 – Приклад роботи додатку у режимі «Забронировать»

Якщо деякий інтервал часу вже є зареєстрованим іншим клієнтом, то він буде відображатися як «неактивний».

За результатами вибору користувача додаток відразу прораховує загальний час заправки та ціну щодо оплати послуги. Інформація щодо загального періоду заправки та очікуваної сплати за послуги буде відображено у верхньому правому куті додатку (рис. 27).

Для зручності користувача у мобільному додатку у футері розміщено кнопку переходу до функції пошуку за декількома критеріями: дата обслуговування, ціновий діапазон за послуги та за назвою вулиці міста.

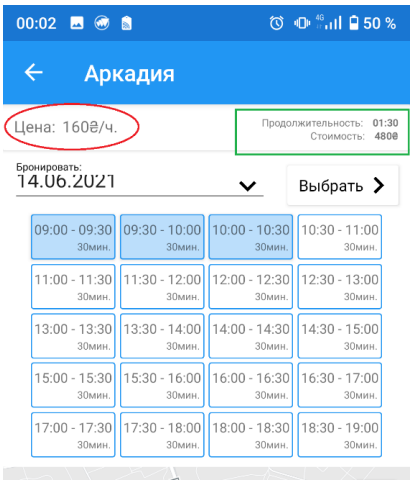


Рисунок 27 – Відображення детальної інформації

Приклад вікна з пошуковими варіантними представлено на рисунку 28:

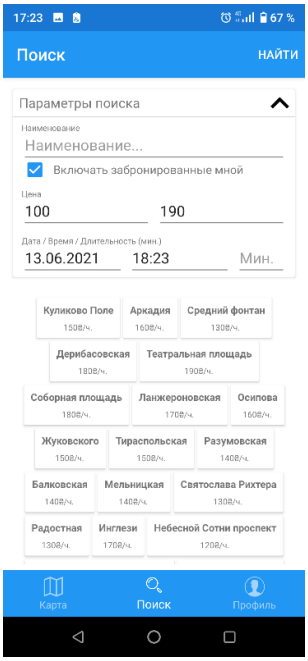


Рисунок 28 – Вікно з пошуковими функціями

Після обслуговування для користувача є можливість за бажанням сплатити рахунок за допомогою QR-Code, тобто система безконтактної оплати (рис. 29). Кожен раз додаток буде формувати так званий QR-Code з

параметрами, тобто з інформацією щодо дати заправки, час та суми послуги для користувача.

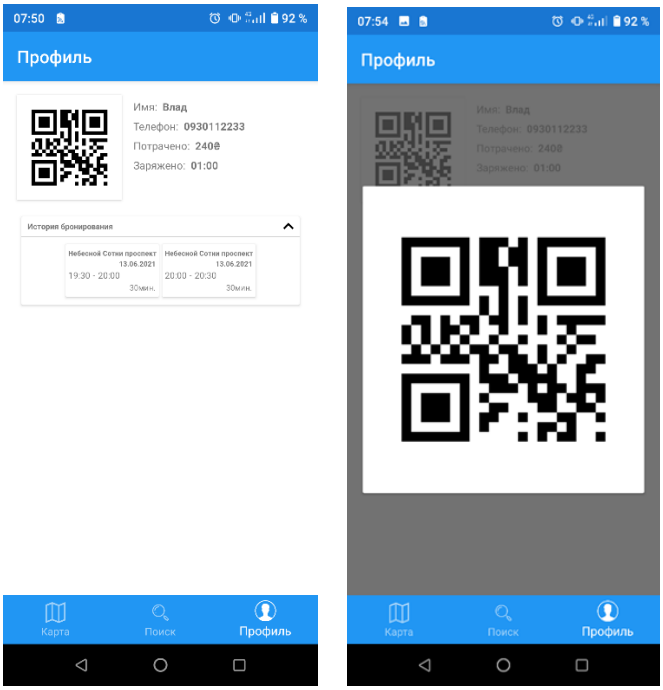


Рисунок 29 – Приклад QR-CODE у особистому кабінеті

Користувач сканує QR-код сканером платіжного додатка смартфона або планшета, автоматично отримує платіжні реквізити продавця (а не заповнює всі реквізити вручну), вибирає платіжну карту, вводить суму, підтверджує платіж, і з його карти списуються грошові кошти на користь продавця послуги.

Усі резерви електрозарядних станцій та історія попередніх заправок завжди зберігається у кабінеті користувача. Це дуже зручно, особливо якщо водій часто використовує «улюблені» локації, таким чином скорочується алгоритм пошуку заправки.

Крім цього, за бажанням користувача є можливість видалити записи (рис. 30).

На даний час у проекті розміщені локації тільки для Одеси. Крім цього, у налаштуваннях додатку є обмеження щодо зберігання даних щодо попередніх локацій обслуговування (до 30).

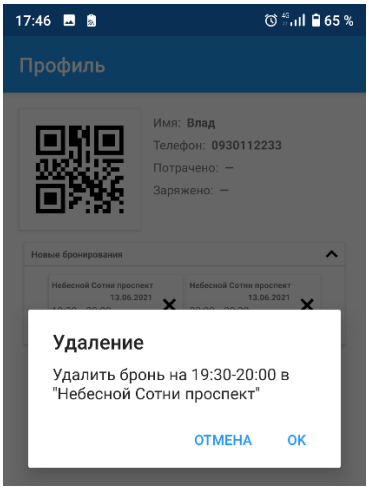


Рисунок 30 – Видалення історії обслуговування

ВИСНОВКИ

Автозаправні станції, що представляють собою повноцінних учасників ринку надання послуг, потребують постійного оновлення та введення різних оптимізуючих ресурсів. Розробка мобільних додатків – це один із способів залучення уваги до вашої мережі, поліпшення позицій вашого бренду на ринку, збільшення довіри з боку старих клієнтів та приведення нових.

Смартфони стали незамінними учасниками загальної життя кожного людини, включаючи власників транспортних засобів. Створення мобільних додатків – гарна можливість використовувати інформаційну середу для вдосконалення контактів з відповідною аудиторією

Об’єкт розробки – це мобільний додаток для бронювання заправок електрокарів. Робота над виконанням диплома проходила у декілька етапів. Так, на першому етапі було виконано:

- аналітичний огляд предметної області, а саме аналіз існуючих мобільних додатків з даної предметної області, виявлення їх недоліків;
- сформовані вимоги і задачі до об’єкту розробки;
- обрані програмні засоби для реалізації програмного додатку.

Під час розробки були реалізовані такі вимоги до проекту, як:

- спроектовано інтерфейс користувача;
- побудовані UML-діаграми;
- реалізована функція пошуку заправок за картою міста;
- можливість огляду вільних заправок та бронювання на конкретний час;
- вивод суми чека та можливість для користувача безконтактної оплати за послуги на заправці.

Як подальший розвиток проекту слід розглянути впровадження мобільного додатку у роботу конкретної мережі заправок з метою ведення бази даних користувачів та можливості отримання для них системи бонусів та скидок на

послуги. Тим самим постачальники послуг отримають розширення клієнтської бази та збільшення прибутків.

Крім цього, в такому разі слід розробити кабінет адміністратору та урахувати методи захисту персональних даних для користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Перспективы рынка мобильных приложений. URL: https://artjoker.ua/ru/blog/perspektivy-rynka-mobilnykh-prilozheniy/ (дата звернення 03.03.2021)
2. Почему электрокар это выгодно? URL: https://news.infocar.ua/infra-struktura_dlya_elektrokarov_chto_est_i_chto_v_planah_138076.html (дата звернення 03.03.2021)
3. Быстрые зарядные станции. URL: https://smart-chargers.ugv.ua/ru/mobilnoe-prilozhenie/ (дата звернення 05.03.2021)
4. Toka.energy. URL: https://toka.energy/chargemap/ (дата звернення 06.03.2021)
5. Мобильное приложение UGV_chargers для электрозаправок от ИНФО-КОМ ЛТД. URL: http://ua.automation.com/content/ugv-chargers-infocom (дата звернення 10.03.2021)
6. 10 міфів про електрокари. URL: garazhhttps://www.garazh.com.ua (дата звернення 12.03.2021)
7. Microsoft Visual Studio. URL: https://open-file.ru/programs/microsoft-visual-studio (дата звернення 15.03.2021)
8. Разработка крупных приложений на Xamarin: в чем выгода. URL: https://vc.ru/dev/181174-razrabotka-krupnyh-prilozheniy-na-xamarin-v-chem-vygoda (дата звернення 16.03.2021)
9. Что такое Mono? URL: http://dir.by/developer/monogame/mono/ (дата звернення 24.03.2021)
10. Подробно о Xamarin. URL: https://habr.com/ru/post/188130/ (дата звернення 25.03.2021)
11. Введение в C#. Язык C# и платформа .NET Core. URL: https://metanit.com/sharp/tutorial/1.1.php (дата звернення 10.04.2021)

12. SQLITE. URL: https://lecturesdb.readthedocs.io/databases/sqlite.html/ (дата звернення 12.04.2021)
13. SQLITE. URL: https://younglinux.info/sqlite/sqlite. (дата звернення 12.04.2021)
14. Национальная библиотека им. Н. Э. Баумана. Bauman National Library. SQLite. URL: https://ru.bmstu.wiki/SQLite (дата звернення 15.04.2021)
15. QRCode Encoder SDK/DLL2.5. URL: https://ru.onmouseenter.com/software/932302 (дата звернення 20.04.2021)
16. Open Source QRCode Library. URL: https://www.codeproject.com/Articles/20574/Open-Source-QRCode-Library (дата звернення 24.04.2021)
17. 10 Application Interface Design Examples: How Top Companies Design Their App's UIs. URL: https://www.eleken.co/blog-posts/10-application-interface-design-examples-how-top-companies-design-their-apps-uis (дата звернення 24.04.2021)
18. UML для бизнес-моделирования: зачем нужны диаграммы процессов. URL: https://evergreens.com.ua/ru/articles/uml-diagrams.html (дата звернення 29.04.2021)
19. Основы бизнес-моделирования: 5 популярных нотаций с примерами. URL: https://flexberry.github.io/ru/gpg_practical-guides-uml.html (дата звернення 29.04.2021)