МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ГНАТЮК МАКСИМ АНДРІЙОВИЧ

Допускається до захисту:
завідувач кафедри
інформаційних технологій,
д-р техн. наук, доцент
_____ Тетяна НЕСКОРОДЄВА
«_____» _____ 2022р.

РОЗРОБКА СИСТЕМИ РОЗПІЗНАВАННЯ, ЗБОРУ ТА ОБРОБКИ ДАНИХ ПРО ТРАНСПОРТІ ЗАСОБИ

Спеціальність 122 «Комп'ютерні науки»

Кваліфікаційна (магістерська) робота

Науковий керівник:
Римар П. В., старший викладач
кафедри інформаційних технологій
Науковий консультант:
Штовба С.Д.,
д-р техн. наук, професор

_____
(підпис)

Оцінка: _____/_____/_____
(бали за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця – 2022

Гнатюк М.А. Розробка системи розпізнавання, збору та обробки даних про транспорті засоби. Спеціальність 122 «Комп'ютерні науки», Донецький національний університет імені Василя Стуса, Вінниця, 2022.

Метою виконання даної кваліфікаційної роботи є створення системи розпізнавання номерних знаків транспортних засобів та збору даних на основі отриманої інформації.

У вступі наведено актуальність розробки даної системи, об’єкт та предмет дослідження, а також мету роботи.

У першому розділі розглянуто постановку задачі, проблематику та огляд існуючих аналогів, визначення переваг та недоліків кожного.

Другий розділ освітлює технології та методи, які були використані у створенні даної системи.

Третій розділ описує вирішення проблеми обраними технологіями.

Ключові слова: розпізнавання авто, Android, Kotlin, мобільний додаток, ML Kit.

Hnatyuk M.A. Development of a vehicle recognition and data collection system. Specialty 122 "Computer Science". Vasyl Stus Donetsk National University, Vinnytsia, 2022.

The main purpose of the qualification work is the development of vehicle license plate recognition system and collect data based on the information received.

The introduction presents the relevance of the development of this system, the object and subject, the purpose of the work of the research as well.

In the first chapter, the statement of the problem, problems and an overview of existing analogues, determination of advantages and disadvantages are considered.

The second section illuminates the technologies and methods that were used in the creation of this application.

The third section describes the solution of the problem with selected technologies.

Keywords: car recognition, Android, Kotlin, mobile application, ML Kit.

ЗМІСТ

ВСТУП

Обрана тема є актуальною, оскільки станом на сьогодні в Україні згідно статистики у кожного 4-го жителя є транспортний засіб. Це значить те, що країна має близько 15 млн. транспортних засобів у своєму автопарку. І нерідко виникає потреба мати змогу швидко отримати інформацію про авто (марку, модель, рік випуску, тощо), наприклад, для людини яка знаходиться на авторинку та хоче придбати собі автомобіль або ж просто люди, які зацікавлені в автомобільній тематиці та побачивши цікавий транспортний засіб хочуть отримати про нього більше характеристик. Здається, немає іншого виходу, як просто знати, як люди можуть тебе обдурити, і як від цього захиститися. Окрім зовнішнього вигляду та технічного стану автомобіля, покупець повинен перевірити документи покупець повинен перевірити документи, реєстраційний номер, VIN-код. Це публічні дані, які використовуються для ідентифікації та перевірки транспортного засобу, тому чесний продавець не буде їх безпідставно приховувати чи надто акцентувати на цьому увагу. Продавець повинен бути власником транспортного засобу. Документи, реєстраційний номер повинні бути оригінальними. VIN-код не може бути ні пошкодженим, ні свіжо-пофарбованим, ні заклеєним. Щоправда, легальна процедура зміни номерних знаків коштує недорого, але VIN-код є незмінним, тому всі дані про автомобіль будуть прив'язані до нього. Є багато даних, які допоможуть Вам чітко зрозуміти історію автомобіля і його проблемні деталі.

- Кількість людей, як довго вони володіли (що буде характеризувати тип використання)
- ДТП, штрафи
- Дані дилерського центру
- Дані про країну імпорту
- Попередні продажі, тощо

Враховуючи швидкий розвиток та популярність операційної системи Android, її великої кількості бібліотек для розпізнавання об’єктів - це робить її ідеальним варіантом для вирішення поставленої задачі.

Метою роботи є дослідження існуючих методів розпізнавання номерних знаків та їх покращення. А також знаходження та поєднання інформації про розпізнаний транспортний засіб. Реалізація поставленої задачі на мобільній платформі Android.

Об’єктом дослідження в рамках магістерської роботи є процес розпізнавання транспортних засобів та отримання максимально повної інформації по ним.

Предметом дослідження є методи розпізнавання транспортних засобів, а саме їх номерних знаків та знаходження інформації по даним транспортним засобам у державних реєстрах та різноманітним джерелам даних.

Теоретична цінність кваліфікаційної роботи полягає в аналізі технологій розпізнавання рухомих об’єктів, а саме номерних знаків транспортних засобів, та наявних відкритих та закритих джерел даних для збору інформації про ТЗ.

Практична цінність даної роботи пов’язана із використанням розробленого додатку при потребах отримання максимально-доступної інформації про транспортний засіб на основі його номерних знаків та/або серійного номеру.

Задачами даної роботи є наступне:

1. Аналіз додатків у схожій тематиці, розгляд їх недоліків та переваг.
2. Поглиблене вивчення розробки під ОС Android, методів машинного навчання для розпізнавання об’єктів.
3. Написання серверної частини для зберігання інформації про користувачів та транспортні засоби.
4. Безпосереднє написання системи розпізнавання, збору та обробки інформації про транспорті засоби.

РОЗДІЛ 1

ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Концентрація мобільних пристроїв на ринку дуже стрімко зростає, пропорційно зростає і кількість мобільних додатків. Ця величезна галузь розширюється з кожним днем і не збирається зупинятись. На ринку праці мобільних додатків стрімко збільшилась кількість розробників, кількість самих додатків зростає у геометричній прогресії. Також досяг позахмарних показників - дохід, котрий створюється індустрією мобільних програм. Із розвитком мобільних пристроїв збільшується і їхня потужність, що в свою чергу дає змогу обробляти більше інформації, наприклад, аналізувати об’єкти за допомогою камери в режимі реального часу. Це в сукупності із розвитком інформаційних технологій, збільшенням державних реєстраційних баз даних, різноманітних джерел по отриманню інформації про транспортні засоби робить мобільний девайс ідеальним пристроєм для створення системи яка розглядається в даній роботі [1].

На базі операційної системи Android [2] випускається все більше і більше смартфонів, планшетних ПК та інших видів пристроїв, зручних для використання як в повсякденному житті, так і у вузькоспрямованому плані. Чому ж ця операційна система така поширена? Є кілька причин:

1. Підтримка великої кількості пристроїв різних виробників.
2. Характеризується маленьким порогом входу у розробку Android додатків. Google надає безкоштовну IDE для програмування, у той же час Apple, для їх системи iOS – потребує немалих фінансових вкладень для початку.

Окрім перераховного вище, одна з найважливіших переваг для роботи зі сторонніми ресурсами операційної системи Andoid є наявність безкоштовних бібліотек (Google Map API, та, що немало важливо бібліотекою ML Kit, яка знадобиться для розпізнавання номерних знаків), в той час як у свою чергу для Windows Phone Mobile таких бібліотек немає. [4]

Нижче приведений графік відсоткового співвідношення мобільних пристроїв на різних оперативних системах. Можемо бачити, що мобільні пристрої на ОС Android займають більшість ринку. Зазначимо, що даний графік відображає співвідношення в загальному у світі, але в Україні ситуація майже не відрізняється – частка пристроїв на ОС Android становить близько 70%. Так як система розпізнавання яку ми розглядаємо на даному етапі реалізації буде працювати лише із транспортними зареєстрованими в Україні очевидним є вибір на користь ОС Android. [5]

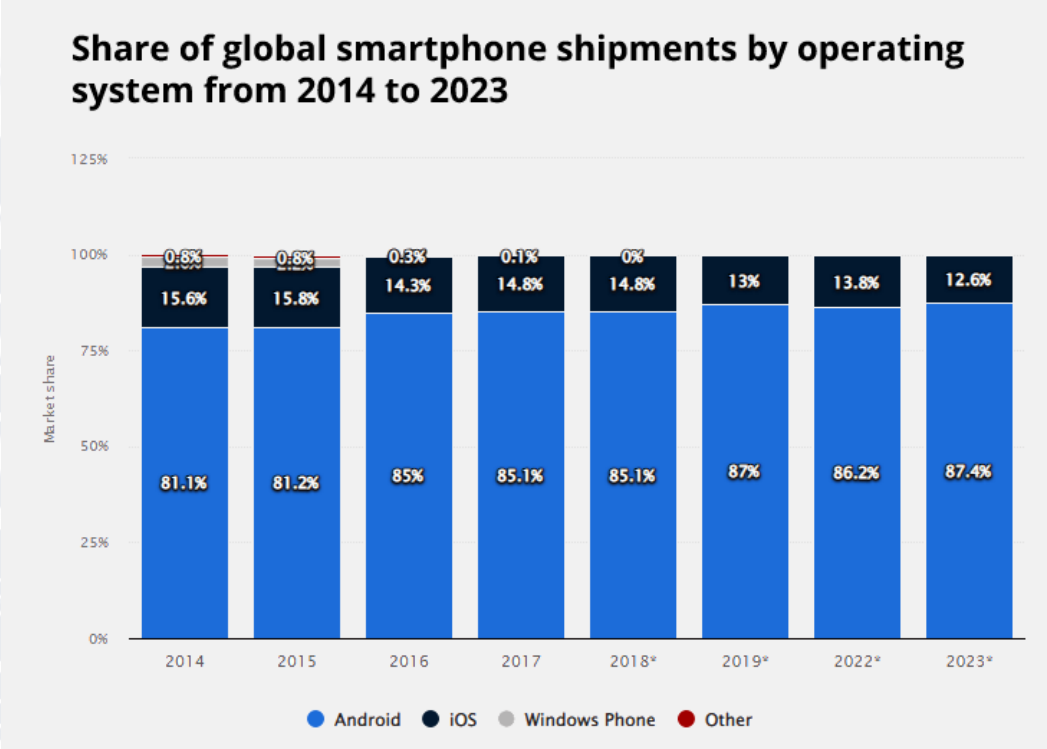

Рисунок 1.1 – Графік співвідношення мобільних пристроїв на різних ОС [1], [4]

1.2 Постановка задачі

Необхідно реалізувати систему розпізнавання номерних знаків транспортних засобів під управлінням системи Android. Функції, які повинні бути доступні у додатку:

1. Створення нового користувача із використанням електронної пошти та паролю. Має бути можливість зареєструватись через сервіси Google та Facebook.

2. Авторизація. Відбувається за допомогою пошти та паролю або сервісів Google та Facebook.
3. Розпізнавання номерного реєстраційного знаку транспортного засобу. У разі розпізнання одразу декількох транспортних засобів має бути вибір одного із них.
4. Додатково має бути можливість розпізнавання VIN номера автомобіля як альтернативний спосіб.
5. Ввід автомобільного номера вручну. Не завжди є змога навести камеру на автомобіль, тому має бути можливість ввести реєстраційний номер власноруч.
6. Отримання інформації по вказаному транспортному засобу, а саме:
    a. Марка, модель та рік виготовлення
    b. Колір
    c. Тип палива, об’єм двигуна
    d. Середня ціна на ринку автомобілів (береться із відкритого Autoria API)
    e. Інформація про реєстрацію (область та дата реєстрації)
    f. Інформація про викрадення
    g. Інформація про участь на аукціонах, інформація про пошкодження
7. Можливість перегляду раніше оглянутих транспортних засобів
8. Можливість додавання розпізнаного транспортного засобі до списку улюблених. Даний список буде прив’язаний до профіля користувача та зберігатись після виходу.

1.3 Аналіз існуючих рішень

Системи розпізнавання номерних знаків транспортних засобів та збору інформації уже присутні на ринку мобільних додатків. Розглянемо найбільш популярні із них, визначимо переваги та недоліки, складемо порівняльну таблицю.

1. CarPlates

CarPlates – один із найбільш популярних додатків у даній тематиці, є лідером у кількості скачувань у Google Play Market. Має можливість розпізнавання номерних знаків та VIN коду автомобіля. Видає досить багато інформації включно із даними про участь в аукціонах, але вони не є безкоштовними. Є змога переглядати історію пошуків та додавати знайдені авто до улюблених.

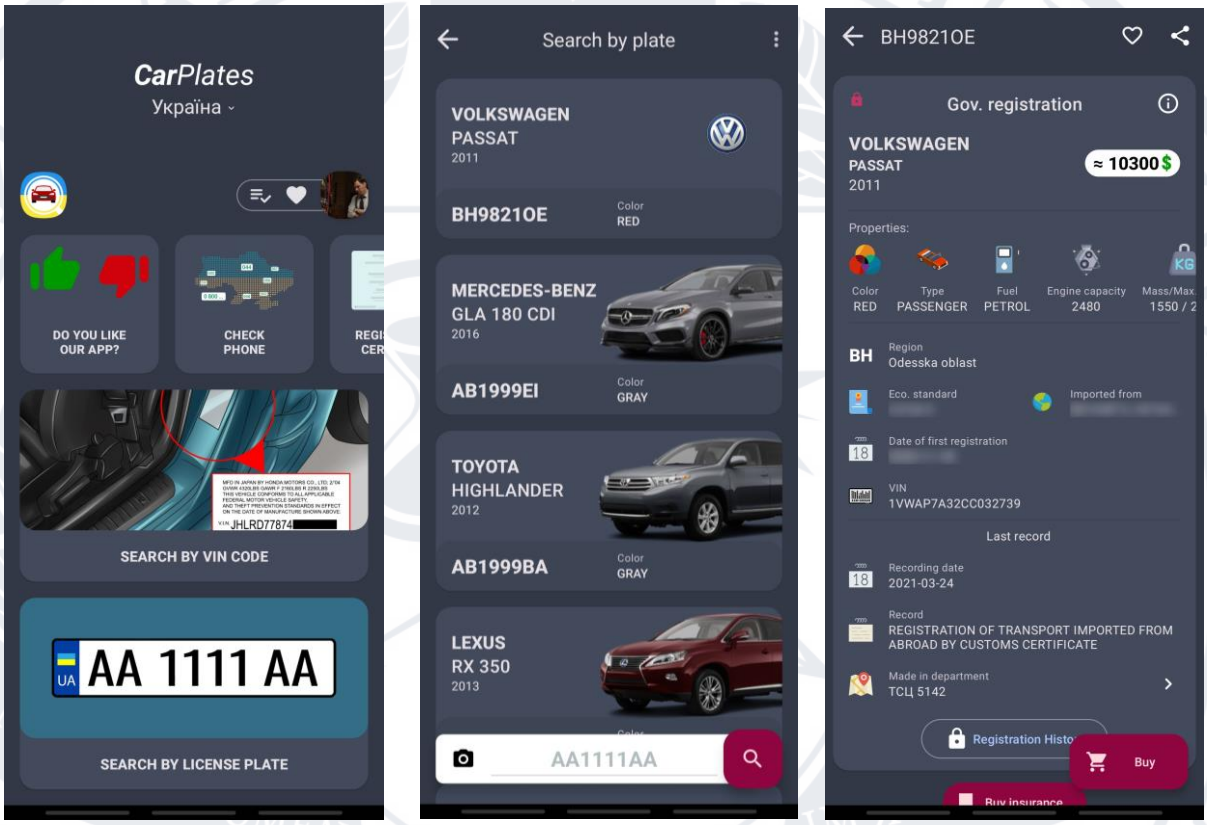

Рисунок 1.2 – Мобільний додаток CarPlates

2. Автобот Україна

Автобот Україна – другий за популярністю мобільний додаток із збором даних про транспортні засоби. Даний додаток не має можливості розпізнавання номерних знаків або VIN коду авто. Також немає історії переглянутих авто та додавання в збережені. Інформація отримується лише з одного джерела – реєстру МВС України.

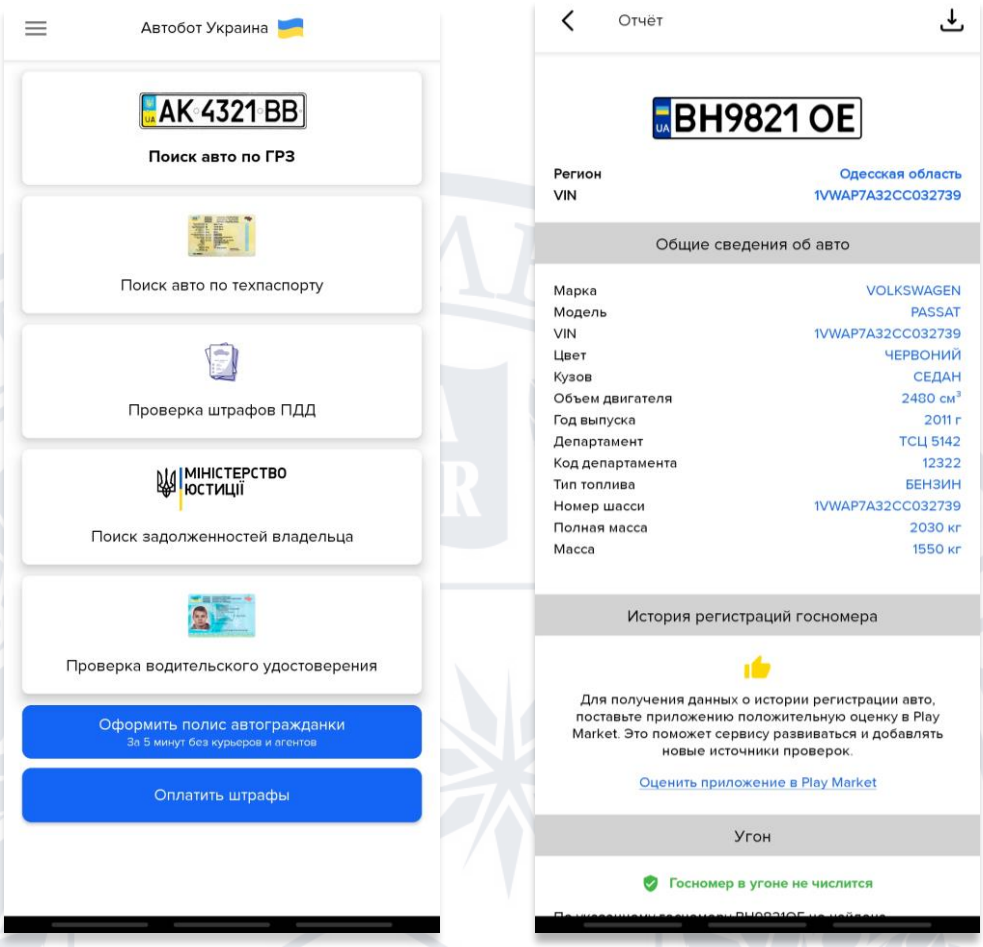

Рисунок 1.3 – Мобільний додаток Автобот Україна

3. Перевірка Авто та Бази МВС Україна

Дані два додатки фактично є копіями та відрізняються лише дизайном. Вони не мають можливості розпізнавати номерні знаки. Вся інформація береться із одного джерела. Перегляду історії та вподобаних авто немає. Має зрозумілий та сучасний дизайн. Повністю безкоштовні, але присутня реклама.

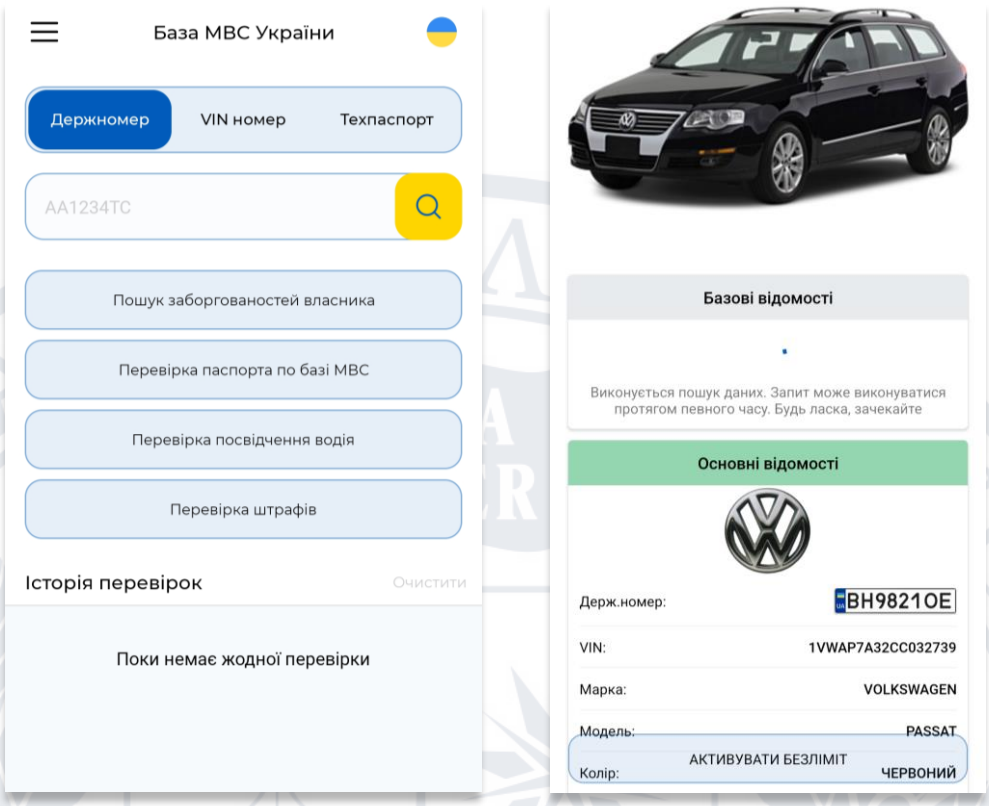

Рисунок 1.4 – Мобільний додаток Перевірка Авто

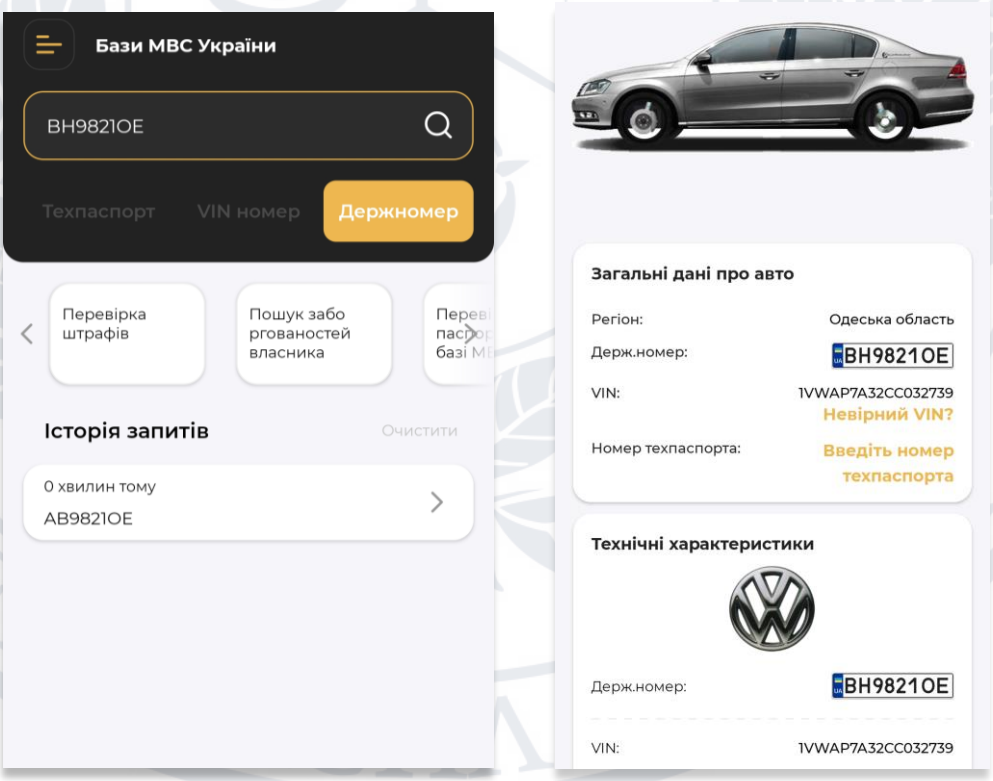

Рисунок 1.5 – Мобільний додаток Бази МВС Україна

4. Перевірка номера

Перевірка номера – найменш популярний додаток із першої п’ятірки в Google Play Market. Має можливість сканувати номерний знак, але лише в режимі фотографії. До недоліків можна також віднести застарілий дизайн.

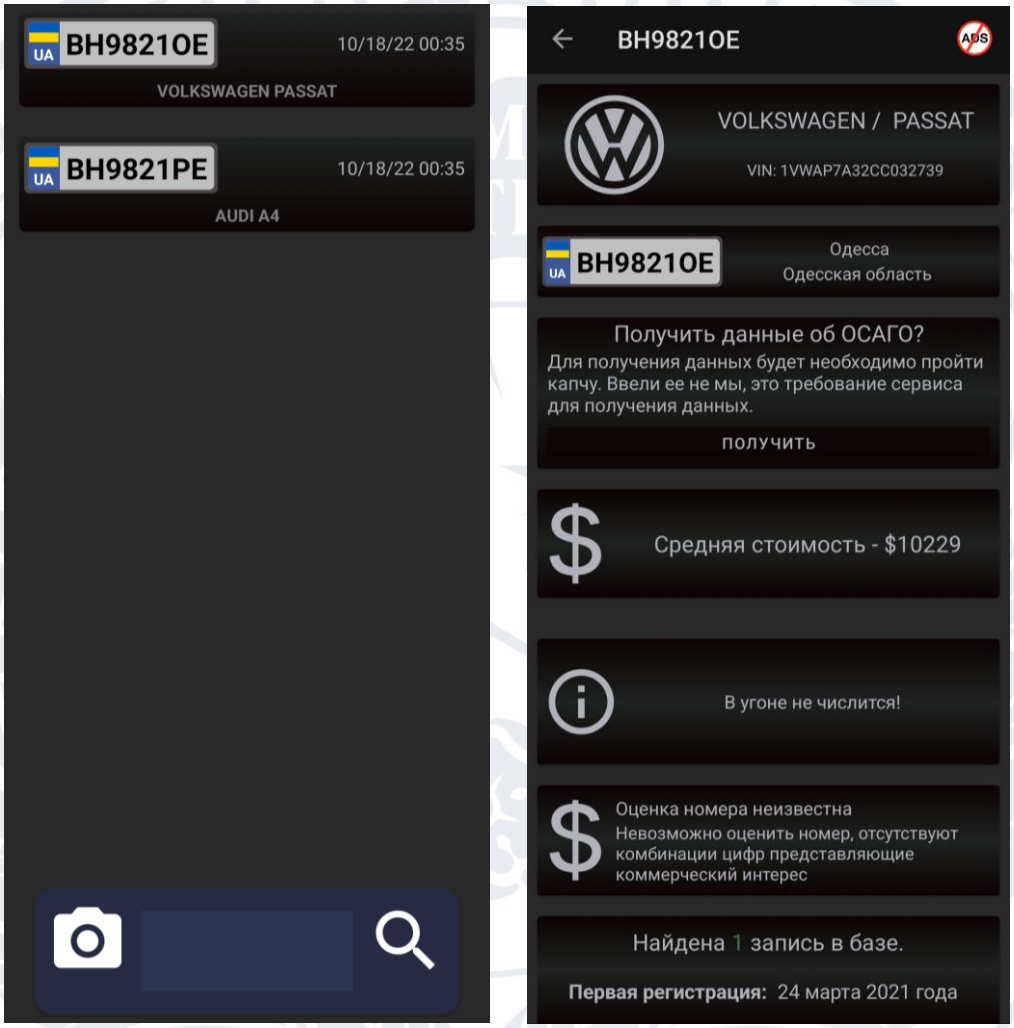

Рисунок 1.6 – Мобільний додаток Перевірка номера

Після розгляду альтернатив складемо порівняльну таблицю для виявлення переваг та недоліків зазначених вище додатків:

	Car Plates	Автобот Україна	Перевірка авто	Бази МВС України	Перевірка номера
Можливість сканування номерних знаків транспортних засобів	ТАК	НІ	НІ	НІ	ЧАСТКОВО

Можливість сканування VIN коду транспортних засобів	ТАК	НІ	НІ	НІ	НІ
Безкоштовність	ЧАСТКОВО	ТАК	ТАК	ТАК	ТАК
Сучасний зрозумілий дизайн	ТАК	ТАК	ТАК	ТАК	НІ
Збереження історії пошуку	ТАК	НІ	НІ	НІ	НІ
Додавання до улюблених	ТАК	НІ	НІ	НІ	НІ
Декілька джерел збору даних	ТАК	НІ	ТАК	ТАК	НІ
Дешифрування VIN коду	ТАК	ТАК	НІ	НІ	НІ
Можливість переглянути участь в аукціонах	ТАК	НІ	НІ	НІ	НІ
Перегляд середньої ціни	ТАК	НІ	НІ	ТАК	ТАК

Таблиця 1.1 – Порівняння існуючих аналогів

Аналізуючи результати, можемо дійти висновку, що не всі із додатків є безкоштовними, лише один має повноцінний функціонал розпізнавання номерних знаків та VIN коду транспортних засобів. Лише деякі із додатків мають декілька джерел збору інформації. Історію зберігання та додавання до улюблених є лише в першому додатку.

1.4 Інструменти для написання клієнтської частини

На сьогодні екосистема Android пропонує багато можливостей для розробників: це універсальна відкрита платформа, якою користуються мільйони користувачів у всьому світі. За десятиліття існування системи з’явились безліч інструментів для Android розробників, які допомагають

вирішувати складні задачі, швидко знаходити рішення проблем. Їх ефективність постійно підвищується.

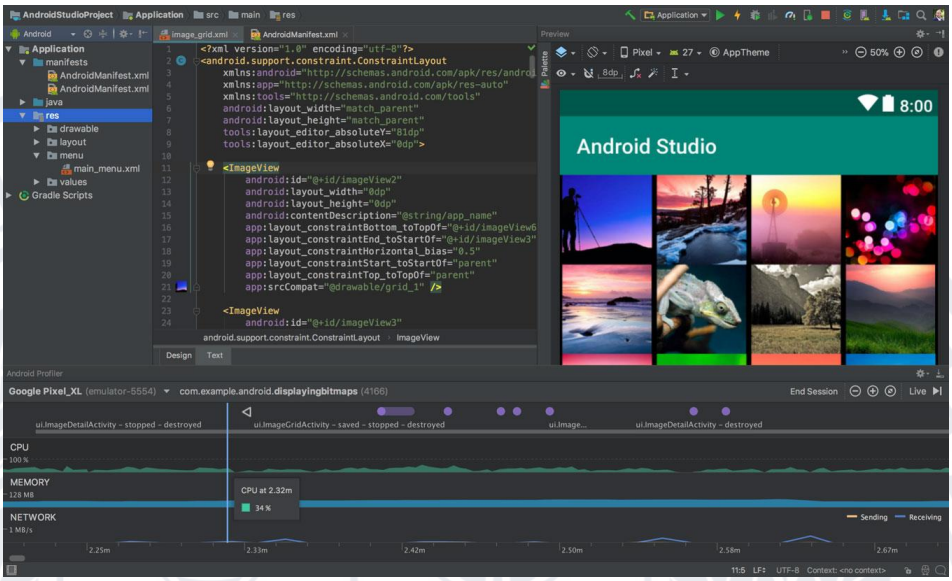

Рисунок 1.6 – IDE Android Studio

Android Studio [6]

Це офіційне інтегроване середовище розробки (IDE) для розробки додатків Android. Він заснований на IntelliJ IDEA, інтегрованому середовищі розробки Java для програмного забезпечення, і містить його інструменти для редагування коду та розробника. Щоб підтримувати розробку додатків в операційній системі Android, Android Studio використовує систему збірки на основі Gradle, емулятор, шаблони коду та інтеграцію з Github. Кожен проект в Android Studio має одну або кілька модальностей із вихідним кодом і файлами ресурсів. Ці модальності включають модулі програми Android, модулі бібліотеки та модулі Google App Engine. Основними відмінностями від інших середовищ розробки є:

- Розробка інтерфейсу користувача: це одна із найбільших переваг. Android Studio дає можливість масштабовані макети із різноманітними стилями та безліччю функцій.
- Чітка структура проекту та навігація по файловій системі.
- Багато влаштованих плагінів для розробки: термінал для виводу системних записів, інспектор мережевих запитів та локальної бази даних, тощо.

Android Device Monitor

Один з вбудованих інструментів в Android Studio - Android Device Monitor, він проводить моніторинг вашого віртуального або фізичного пристрою під час виконання програми і отримує інформацію про процеси, які виконуються по потоку, мережеву статистику, LogCat, тощо. Даний інструмент відмінно підходить для тестування продуктивності ваших додатків. Саме цей інструмент нам допоможе виявити використання ресурсів під час розпізнавання номерних знаків. Так як це процес який потребує багато ресурсів, може бути необхідність в оптимізації для коректної роботи на не потужних пристроях.

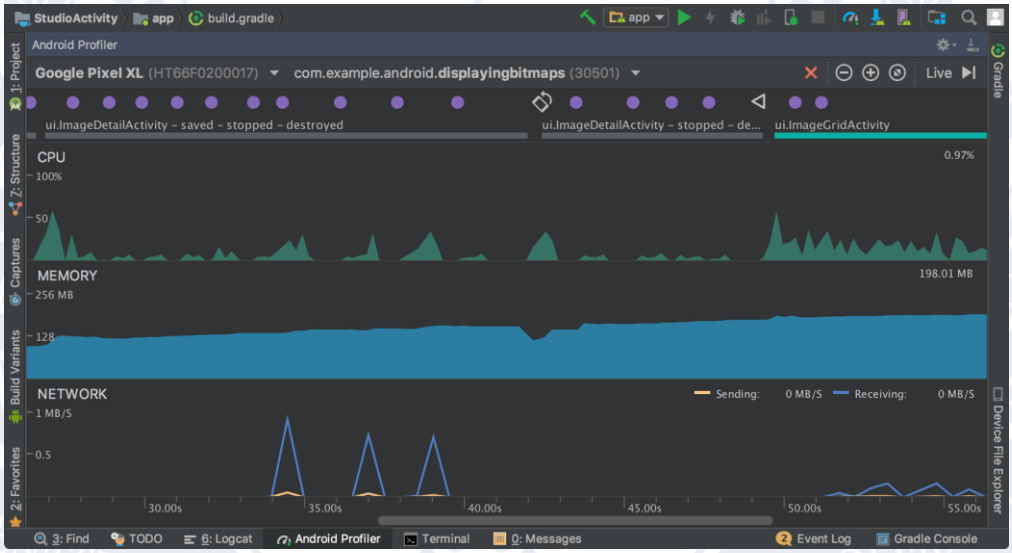

Рисунок 1.8 – моніторинг ресурсів запущеного додатку

Android Debug Bridge

За допомогою ADB (консольний додаток для ПК) - проводиться налагодження Android пристроїв і емуляторів. Він працює за принципом клієнт-сервер. Як тільки ви запустили перший раз ADB з будь-якою командою - створюється сервер у вигляді системної служби (демона), котра прослуховує всі команди, які посилаються на порт 5037. ADB входить в комплект з Android Studio і вам навряд доведеться встановлювати його самостійно.

Figma

Figma - це хмарний кросплатформовий сервіс для дизайнерів інтерфейсів та розробників, з яким можна працювати безпосередньо у браузері. Це

платформа, де є багато чого необхідного для роботи з векторними об'єктами, шрифтами, графікою, ефектами, тощо. Саме за допомогою цієї програми буде створено користувацький дизайн нашої системи. Цей інструмент є безкоштовним для власного використання.

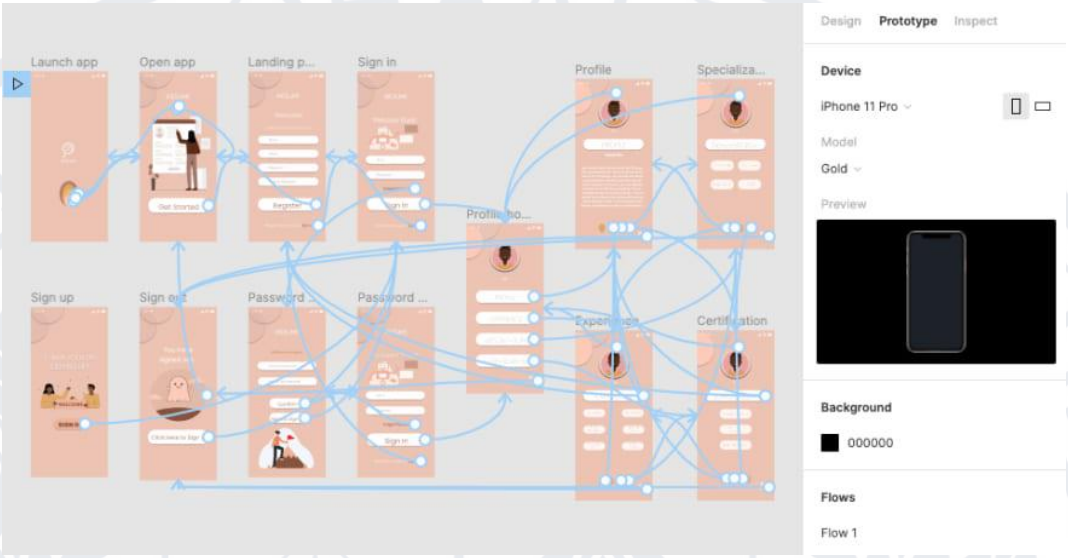

Рисунок 1.9 – платформа для створення прототипів Figma

1.5 Інструменти для написання серверної частини

Для того щоб профілі користувачів зберігались у хмарному сховищі, там знадобиться серверна частина. Для цього було обрано Kotlin, аби працювати з однією мовою програмування водночас. Для того щоб працювати із фреймворком Kotlin Server Side потрібна IntelliJ Idea як середовище розробки.

IntelliJ IDEA – зручне інтегроване середовище швидкої розробки для Java, JavaScript, Python, що включає підтримку останніх технологій та фреймворків, розроблена компанією JetBrains. IntelliJ IDEA, після індексування вихідного коду - надає багато можливостей для ефективної і швидкої розробки програмного забезпечення: розумне автодоповнення, аналіз коду в реальному часі і провідна система рефакторингу. Розумне автодоповнення пропонує тільки ті елементи коду, котрі є актуальними в даному контексті, в той час як просте автодоповнення рекомендує тільки імена ключових слів, полів, методів і класів в області видимості.

Heroku

Так як нам необхідно мати доступ до даних нашого профілю, а відповідно історії переглядів та вподобаних записів – то наш віддалений сервер повинен бути завжди доступний. Для цього скористаємось хмарним сервісом для розгортування веб-сервісів. Heroku - це сервіс, який надає розробнику настроєне середовище розгортання веб-проектів. Тобто, це свого роду спеціалізований хостинг, де вже налаштовані всі необхідні сервіси - мови програмування, бази даних тощо. Користувачеві лише залишається за допомогою команди відправити код своєї програми на платформу і вона відразу стає доступною в мережі.

Висновок до розділу 1

Даний розділ присвячено актуальності мобільних пристроїв з різними ОС та додатків, потребу системи розпізнавання транспортних засобів та збору інформації по ним. Окрім того, були оглянуті необхідні функції програми, її альтернативи на сьогодні, їх переваги та недоліки.

РОЗДІЛ 2

ОПИС ТЕХНОЛОГІЙ ВИКОРИСТАНИХ У РОЗРОБЦІ

2.1 Технології використані у розробці клієнтської частини

На початку свого розвитку та ще не так давно основною мовою розробки під платформу Android була Java, але інформаційні технології розвиваються в результаті чого з’являються нові мови програмування. [8]. Однією із таких стала програмування мова Kotlin, розроблена компанією **JetBrains**. У 2017 році, на конференції **Google I/O**, Kotlin був офіційно проголошений офіційною та рекомендованою мовою програмування під платформу Android [9]. Розглянемо основні нововведення та переваги над Java:

1. Kotlin Corounites – робота із асинхроними задачами;
2. Null Safety – інструменти для безпечної взаємодії із nullable типами даних задля уникнення NullPointerException у Java
3. Data Class – створені для легкої роботи із класами як із моделями даних
4. Extensions – можливість розширювати закриті компоненти бібліотек, тощо.
5. Короткий синтаксис – Kotlin має багато конструкцій для того, щоб робити код більш простим та зрозумілим

Безпосередньо для розробки мобільного додатку було використано Android SDK [9]. Android SDK — це набір бібліотек і інструментів розробки програмного забезпечення, необхідних для розробки програм Android. Має підтримку широкої кількості різноманітних мобільних пристроїв, таких як мобільні телефони, планшетні комп'ютери, розумні годинники, тощо.

Kotlin Coroutines [11]

Співпрограми були представлені в Kotlin 1.1. Фактично вони є новим способом написання асинхронного неблокувального коду. Асинхронний код (від asynchronous programming) — це код, який виконується паралельно іншим. Його також називають неблокуючим, оскільки він не блокує основний потік. Асинхронне програмування допомагає швидше виконувати кілька непов’язаних

завдань. У синхронному програмуванні код виконується рядок за рядком. Їх характерними особливостями є:

- Легкість: ви можете запускати багато співпрограм в одному потоці завдяки підтримці призупинення, яке не блокує потік, у якому запущена співпрограма.
- Менше витоків пам'яті.
- Вбудована підтримка скасування.
- Інтеграція з великою кількістю інших бібліотек.

Navigation Components [12]

Одним із найважливіших аспектів вдалої взаємодії користувача із додатком є внутрішня навігація. Jetpack Navigation Components – набір інструментів та бібліотек, що спрощують роботу із навігацією та роблять додаток спроможним до масштабування. Бібліотека пропонує ряд переваг, зокрема:

- Анімація та поведінка переходів за замовчуванням.
- Реалізація навігаційних шаблонів інтерфейсу користувача.
- Інструмент Android Studio для візуалізації та редагування потоку навігації програми

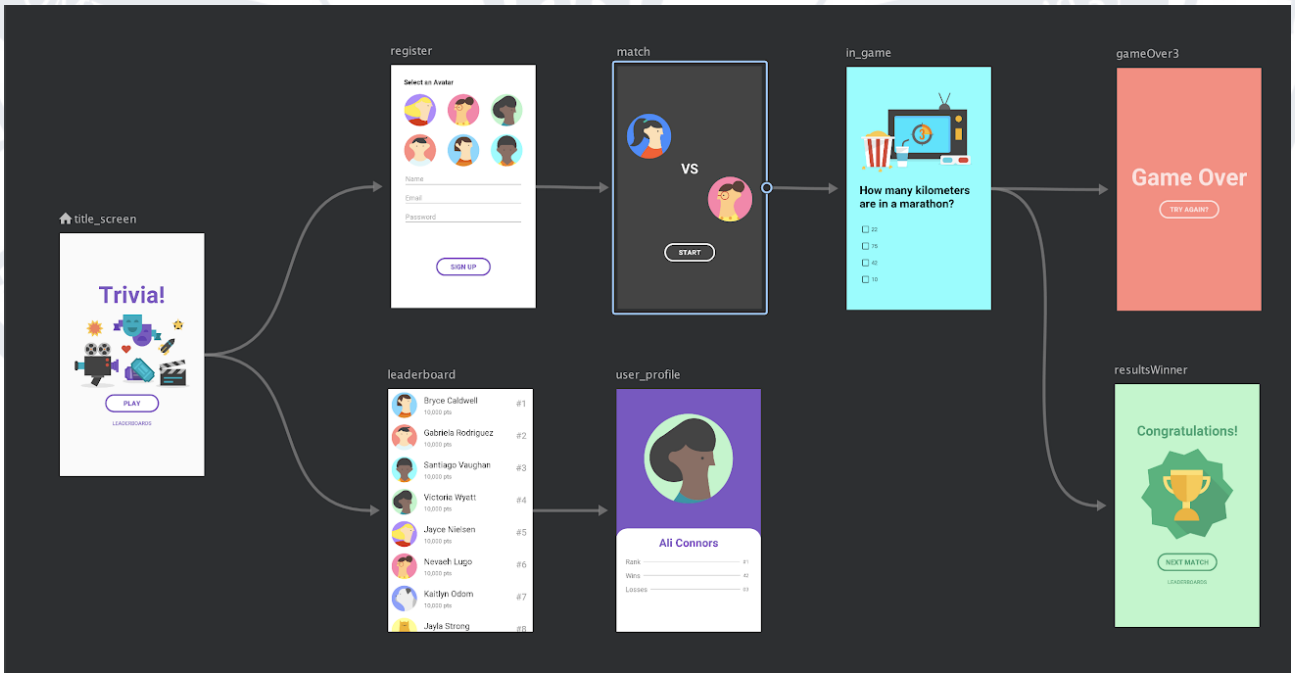

Рисунок 2.1 – Перевага у візуалізація списку екранів та переходів між ними

Hilt [13]

Впровадження залежностей (Depency Injection) - це шаблон проектування, який використовується для реалізації IoC (Inverion of Control). Це дозволяє створювати залежні об’єкти поза класом і надавати ці об’єкти класу різними способами. Використовуючи DI, ми переносимо створення та прив’язування залежних об’єктів за межі класу, який залежить від них. [14]. Шаблон включає 3 типи класів:

- Клас клієнта: клас клієнта (залежний клас) — це клас, який залежить від класу обслуговування
- Клас обслуговування: клас обслуговування (залежність) — це клас, який надає обслуговування класу клієнта.
- Сервіс: вводить об’єкт класу обслуговування в клас клієнта.

Протягом тривалого часу в Android для використання шаблону впровадження залежностей використовувалась бібліотека Dagger. На одній із конференцій Google I/O у 2021 представили новий набір інструментів та бібліотек Android Hilt, яка використовує Dagger у своїй внутрішній реалізації. Переваги Hilt, на які слід звернути увагу:

1. Тестування стає простішим у реалізації
2. Зменшення кількості коду
3. Спрощення конфігурації

Рисунок 2.2 – Hilt – Основні компоненти шаблону DI

Retrofit [15]

Реалізація нашого додатку будується згідно принципів клієнт-серверної архітектури. Як результат цього, нам буде необхідно робити HTTP запити. Для таких задач у більшості випадків в Android використовують бібліотеку мережевої взаємодії Retrofit.

Retrofit — це надійний клієнт REST для Android, Java і Kotlin. Бібліотека надає потужну структуру для автентифікації та взаємодії з API та надсилання мережевих запитів за допомогою OkHttp. Ця бібліотека робить завантаження даних JSON або XML із веб-API досить простим. Після завантаження даних вони перетворюються на звичайний об'єкт даних Java (POJO). До переваг даної бібліотеки можна віднести таке:

1. Можливість зміни принципу серіалізації
2. Інтеграції із бібліотеками багатопоточних операцій
3. Використання OkHTTP у внутрішній реалізації
4. Можливість додавання власних перехоплювачів

```
interface AuthService {

   @POST("$PUB/auth/login")
   suspend fun login(@Body login: LoginRequestBody): AuthResponse

   @POST("$PUB/auth/register")
   suspend fun register(@Body register: RegisterRequestBody): AuthResponse
}
```

Приклад використання API Retrofit

Android KTX [16]

Android KTX - це набір інструментів та бібліотек з розширеннями Kotlin для спрощення роботи з іншими Jetpack та Android бібліотеками. KTX використовує інструменти Kotlin, щоб надати розробникам найоптимальніший підхід при роботі з Jetpack і іншими API. Кожна бібліотека відокремлена в окрему бібліотеку Kotlin KTX з розширеннями Kotlin, що відносяться до цього розділу.

Приклад: робота із посиланнями

Часто існує необхідність перетворити тип даних string до об’єкту посилання, аби використати його в подальшому для завантаження чогось, тощо. Для цього ми можемо просто скористатись методами *Uri.parse(string)* та *string.toUri()*:

```
// Kotlin
val uri = Uri.parse(uriString)
// android KTX
val uri = uriString.toUri()
```

Приклад використання Android KTX в роботі із посиланнями

RecyclerView [17]

Це елемент користувацького інтерфейсу, який дає змогу створювати динамічні списки зі змогою прокрутки. Він з’явився у версії Android Lollipop та став невід’ємною частиною майже кожного додатку. Він є більш ефективною реалізацією списку і розділяє відповідальність між класами. До основних його переваг над звичайними списками у Android є:

1. Вбудована анімація
2. Покращена продуктивність, швидкість роботи
3. Реалізація шаблону ViewHolder
4. Декорування елементів, широкі можливості для створення унікальних елементів списку

Firebase ML Kit [18]

ML Kit - це мобільний SDK, який привносить досвід Google в області машинного навчання в додатки для Android і iOS у вигляді потужної, але простого у використанні бібліотеки. ML Kit SDK тепер повністю орієнтований на машинне навчання на пристрої, він пропонує:

- швидке розблокування сценаріїв використання в режимі реального часу, оскільки обробка відбувається на пристрої і немає мережевих затримок

- працює в автономному режимі, тому навіть коли мережа нестабільна або кінцевий користувач вашого додатку знаходиться в зоні, де немає зв'язку, він працює [19]
- зберігається конфіденційність: оскільки вся обробка виконується локально, немає необхідності відправляти конфіденційні дані користувача через мережу на сервер.

Рисунок 2.3 - Основні можливості ML Kit

2.2 Технології використані у розробці серверної частини

Для того щоб розробляти серверну частину швидше було прийнято рішення використати для реалізації мову програмування Kotlin. У ролі базової бібліотеки для сервера буде виступати Ktor.

Ktor [20] - це платформа для легкого створення серверних додатків — веб-додатків, HTTP-сервісів, браузерних додатків. Сучасні серверні програми мають бути асинхронними, щоб надавати користувачам найкращу взаємодію, а співпрограми Kotlin надають чудові засоби, щоб зробити це простим і зрозумілим способом. Перевагами даної бібліотеки є:

- Ktor створено з нуля за допомогою Kotlin і Coroutines.

- Ktor дозволяє використовувати лише те, що потрібно, і структурувати програму так, як це потрібно. Крім того, є змога легко розширити Ktor за допомогою власного плагіна.
- Має безліч інтеграційних бібліотек для роботи із функціями авторизації, зберігання даних в базах даних, управлення запитами, тощо.
- Постійно підтримка від розробників

PostgreSql [23]

Функцію зберігання даних у нашому серверному додатку виконує бібліотека PostgreSQL. Вона є СУБД, тобто об'єктно-реляційною системою управлення базою даних, базується на мові SQL і підтримує численні можливості. Вона є альтернативою комерційним СУБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так и СУБД з відкритим кодом (MySQL, Firebird, SQLite). На відміну іншим проектам з відкритим кодом, наприклад Apache, FreeBSD або MySQL, PostgreSQL не є підконтрольною одній компанії, вона розробляється завдяки співпраці багатьох людей та компаній, які хочуть використовувати її та впроваджувати у неї найновіші підходи. До переваг вище розглянутої СУБД віднесемо [24]:

1. Механізми транзакцій та реплікацій – потужні та надійні
2. Інтегрована система вбудованих мов програмування.
3. Підтримка баз даних необмежених розмірів
4. підтримка завантаження С-сумісних модулів;
5. Наслідування.
6. Легке масштабування.

Netty [25]

Netty — це платформа клієнт-сервера, яка дозволяє швидко та легко розробляти мережеві програми, такі як сервери протоколів і клієнти. Це значно спрощує та оптимізує мережеве програмування, наприклад сервер сокетів TCP та UDP.

Netty було ретельно розроблено з урахуванням досвіду, отриманого від реалізації багатьох протоколів, таких як FTP, SMTP, HTTP, а також різноманітних двійкових і текстових застарілих протоколів. У результаті Netty вдалося знайти спосіб досягти простоти розробки, продуктивності, стабільності та гнучкості без компромісів.

Ktor JWT [26]

JWT (Json Web Token) - це відкритий галузевий стандарт, який використовується для обміну інформацією між двома об’єктами, як правило, клієнтом (наприклад, зовнішнім інтерфейсом програми) і сервером (сервером програми). Вони містять об’єкти JSON, які містять інформацію, якою потрібно поділитися. Кожен JWT також підписується за допомогою криптографії (хешування), щоб гарантувати, що вміст JSON (також відомий як претензії JWT) не може бути змінений клієнтом або зловмисною стороною. Ktor, який був описаний раніше, має вбудовану підтримку JWT, який є механізмом для автентифікації корисних навантажень, кодованих JSON.

Ktor Locations [27]

Дана бібліотека допомагає керувати вхідними маршрутами та обробляти їх. Вона значно спрощує з ними роботу, наприклад, із аргументами, які йдуть разом із запитом.

Приклад: побудова та обробка маршруту. Даний код буде співпадати буде перенаправляти обробку запита */list/movies/page/10.*

```
@Location("/list/{name}/page/{page}")
data class Listing(val name: String, val page: Int)
```

Приклад обробки посилання із параметрами

Exposed [28]

Для того щоб зручно та легко керувати даними через PostgresSQL скористаємось Exposed. Це полегшена бібліотека SQL на основі драйвера JDBC для мови Kotlin. Exposed має два різновиди доступу до бази даних: DSL із

захищеним типом обгортки SQL і легкі об’єкти доступу до даних (DAO). Дана бібліотека розроблена компанією JetBrains та має відкритий код (ліцензія Apache). Вона забезпечує ідіоматичний API Kotlin для деяких реалізацій реляційних баз даних, водночас згладжуючи відмінності між різними типами баз даних. Exposed дає можливість використовувати її як DSL високого рівня над SQL, а також як полегшений ORM (об'єктно-реляційне відображення). До переваг бібліотеки можна віднести:

1. Змістовна документація
2. Легка у використанні
3. Підтримка розширень Kotlin
4. Велика кількість вбудованих функцій

Firebase ML Kit [18]

Завдяки TensorFlow Mobile і TensorFlow Lite впровадження та використання багатьох моделей у додатках для Android стало дуже простим. Однак розробка і навчання моделей по-прежнему вимагає великого майстерності, часу і зусиль, не кажучи вже про вичислильну потужність. З цієї причини більшість розробників неохотно використовують можливості машинного навчання для своїх додатків. З Firebase ML Kit Google хоче змінити це. Firebase ML Kit - це бібліотека, яка дозволяє вам легко і з мінімальним кодом використовувати безліч високоточних, попередньо навчених глибоких моделей у ваших додатках для Android. Більшість запропонованих моделей доступні як локально, так і в Google Cloud. [29]

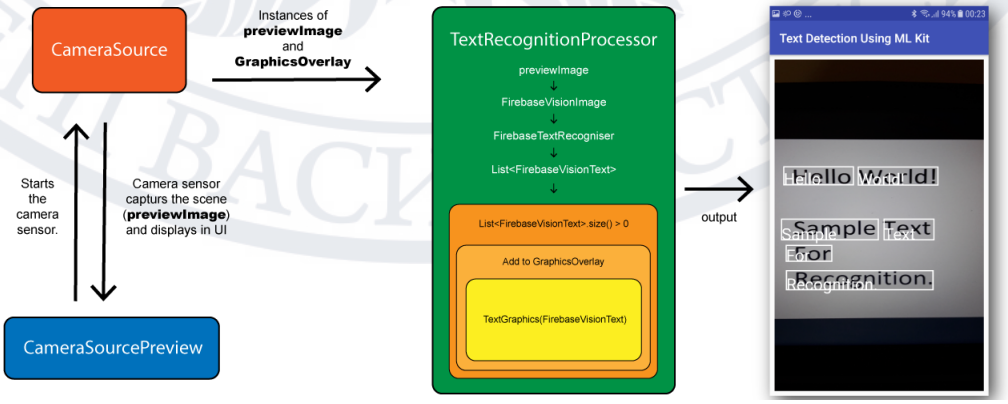

Рисунок 2.4 – Схема взаємодії мобільного додатку та серверної частини ML Kit

Висновок до розділу 2

Даний розділ було посвячено основним інструментам та технологіям для швидкої та комфортної розробки клієнтської та серверної частини. Проаналізовано основні бібліотеки для побудови додатку, визначено їх переваги. Приділена увага бібліотеці розпізнавання об’єктів та принципам її роботи.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКА

3.1 Збір даних

Реєстраційні дані

З 28.08.2018 року відкриті та доступні дані про реєстрацію автомобілів, які були поставлені на облік з 2013 року. HSC.gov.ua (Відкрито сервіс на основі відкритих даних про першу реєстрацію автомобілів в Україні). Даний набір даних надає реєстраційні дані та технічну інформацію про автомобіль за його номерним знаком номер:

- Марка
- Модель
- Рік випуску
- Двигун
- Вага

На жаль, VIN-код там відсутній, тому ці дані доводиться комбінувати іншими джерелами. Data.gov.ua (Інформація про транспортні засоби та їх власників).

Українське автострахування – МТСБУ [30]

МТСБУ (офіційна інформація) - Моторне транспортне страхове бюро України, це єдине об'єднання страхових компаній, яке здійснює обов'язкове страхування цивільно-правової відповідальності власників наземних транспортних засобів, яка може бути заподіяна третім особам. Членство в МТСБУ є обов'язковим для здійснення діяльності страховика у обов'язкового страхування цивільно-правової відповідальності власників наземних транспортних засобів.

Даний сервіс надає послуга перевірки страхового полісу для перевірки страхових і технічних даних, разом із тим надаючи інформацію про VIN код транспортного засобу. Але, API МТСБУ доступний лише для компаній, тому він не може бути інтегрувати в систему неявно.

Рисунок 3.1 - інтерфейс сервісу МТСБУ

Opendatabot

З самого початку Opendatabot - це і компанія-розробник, і платформа, призначена для роботи з державними відкритими даними. 28.08.2018 вони випустили додатковий функціонал до платформи: використання бота для даних про авто.

В сукупності із інформацією що надає hsc.gov.ua ми можемо отримати всі дані для подальших дій. Більше того, іноді це єдине місце, де можливо отримати інформацію про авто під час тестування. Для комерції у них є Opendatabot API. Цей API буде використовуватися для ідентифікації автомобіля та збору даних.

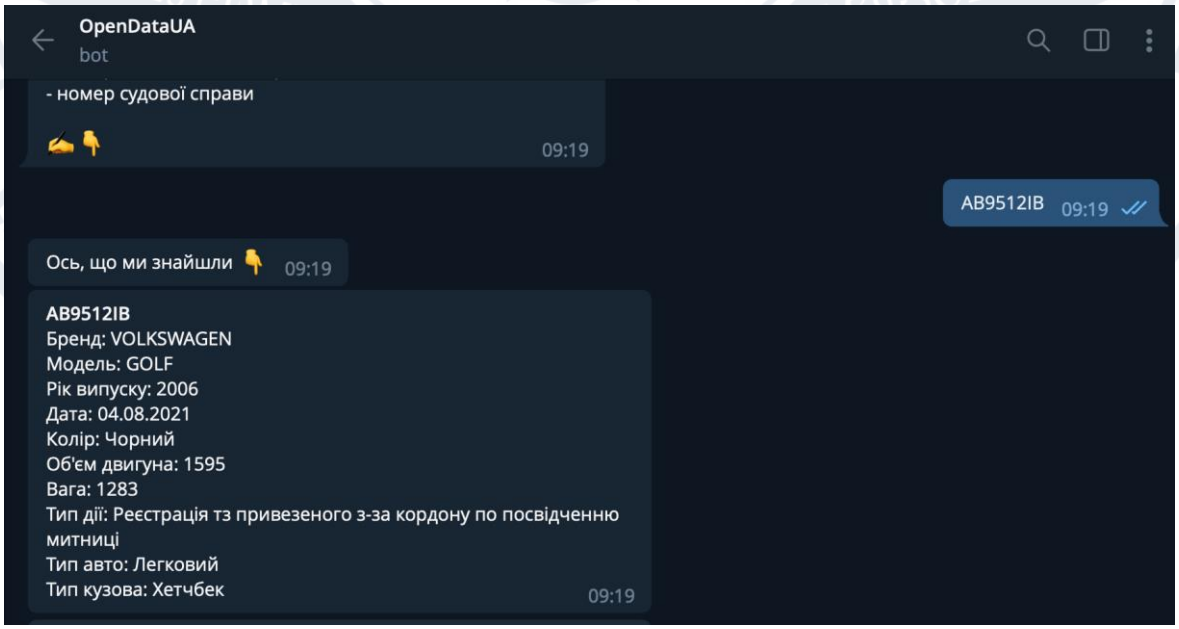

Рисунок 3.2 - інтерфейс сервісу OpenDataBot

```
"status": "ok",
 "data": {
  "count": 1,
  "items": [
   {
    "nRegNew": "AX5870HX",
    "brand": "VOLKSWAGEN",
    "capacity": "1984",
    "color": "СИНІЙ",
    "fuel": "БЕНЗИН",
    "kind": "ЛЕГКОВИЙ УНІВЕРСАЛ-В",
    "model": "TIGUAN",
    "nSeating": "5",
    "ownWeight": "1629",
    "makeYear": "2011",
    "totalWeight": "2180",
    "dFirstReg": "2020-03-03",
    "dReg": "2020-03-03",
    "vin": "WVGAV7AXXCW521718"
   }
 ]
}
```


Приклад OpenDataBot API

AutoRia Car Market

Ця платформа має свою структуровану систему для опису лістингу транспортних засобів та відкрита для інших розробників через портал API Ria.com. Я отримав збільшення ліміту на запити до API прямо в той день, коли я попросив про це. Величезна кількість даних доступний за допомогою API.

У прикладах результатів запитів деякі поля були видалені з метою оптимізації розміру. Актуальні приклади API доступні у великій документації по API AutoRia.

https://developers.ria.com/auto/search?api_key=<API_KEY>&category_id=1&countpage=3&page=1

```
"result": {
 "search_result": {
  "ids": [
   "26897548",
   "20327838",
   "26753282"
  ],
  "count": 180970
 }
}
```


Приклад API запиту

https://developers.ria.com/auto/search?api_key=<API_KEY>&category_id=1&plateNumber.length.gte=1&countpage=3&page=1

За допомогою API ми можемо отримати список актуальних оголошень, які мають номерний знак, розпізнаний або заявлений власником з параметром plateNumber.length.gte=1 ~92000 результатів.

https://developers.ria.com/auto/info?api_key=<API_KEY>&auto_id=100002

А також отримати детальну інформацію про автомобіль з його лістинговим ідентифікатором.

https://developers.ria.com/auto/fotos/198101?api_key=<API_KEY>'

Фотографії для них будуть доступні за допомогою Photos API.


```
"status": 1,
"data": {
 "198101": {
  "194991": {
   "photo_id": 194991,
   "auto_id": 198101,
   "status": 0,
   "checked": 1,
   "sortingIndex": 0,
   "date_add": "0000-00-00:00:00",
   "description": null,
   "url": "auto/photo/19/1949/194991/194991.jpg"
  }
```

```
  }
}
```


Приклад відповіді API для отримання фотографії

Використовуючи ці API, було створено, так і хмарний скрипти для збору даних. Після обробки, за номером автомобіля ми можемо отримати посилання та дані.


```
{
  "userId": 26782,
  "locationCityName": "Полтава",
  "cityLocative": "Полтаве",
  "exchangeType": "Любой",
  "addDate": "2006-09-18 16:34:24",
  "soldDate": "2008-02-20 17:16:04",
  "userPhoneData": {
    "phoneId": "",
    "phone": ""
  },
  "USD": 5800,
  "autoData": {
    "active": false,
    "description": "Сигнализация, центр. замок",
    "year": 2001,
    "autoId": 100002,
    "bodyId": 3,
    "statusId": 1,
    "withVideo": false,
    "race": "69 тыс. км",
    "raceInt": 69,
    "fuelId": 1,
    "fuelName": "Бензин, 1.5 л.",
    "fuelNameEng": "benzin",
    "gearBoxId": 1,
    "gearboxName": "Ручная / Механика",
    "driveName": "Не указано",
    "isSold": true,
    "mainCurrency": "USD",
    "fromArchive": true,
    "categoryId": 0,
    "categoryNameEng": "legkovie",
    "subCategoryNameEng": "sedan",
```

```
   "custom": 0
  },
  "markName": "ВАЗ",
  "markNameEng": "vaz",
  "markId": 88,
  "modelName": "21099",
  "modelNameEng": "21099",
  "modelId": 855,
  "photoData": {
   "count": 1,
   "seoLinkM":
"https://cdn2.riastatic.com/photosnew/auto/photo/vaz_21099__71712m.jpg"
  },
  "linkToView": "/auto_vaz_21099_100002.html",
  "title": "ВАЗ 21099",
  "VIN": "",
  "haveInfotechReport": false
}
```


Повноцінний приклад відповіді AutoRia API

RST UA

Даний ресурс є третім за кількістю автомобілів, не має API, не має номерного знаку / VIN коду в структурі. Незважаючи на те, що це маркетплейс-конкурент, тому аудиторія перетинається дуже багато, на ньому є різні оголошення, які більше ніде не публікуються. Більше того, він зберігає всі зображення за <ID>-<Номер фото>.

http://i1.rst.ua/oldcars/volkswagen/tiguan/big/11110952-1.jpg

Було написано скрипт, який збирає ці зображення. Так як немає реєстраційних номерів, отримувати зображення з цього ресурсу, розпізнаючи номерні знаки і зберігати отримані дані. Це, безумовно, буде основною відмінністю від інших рішень.

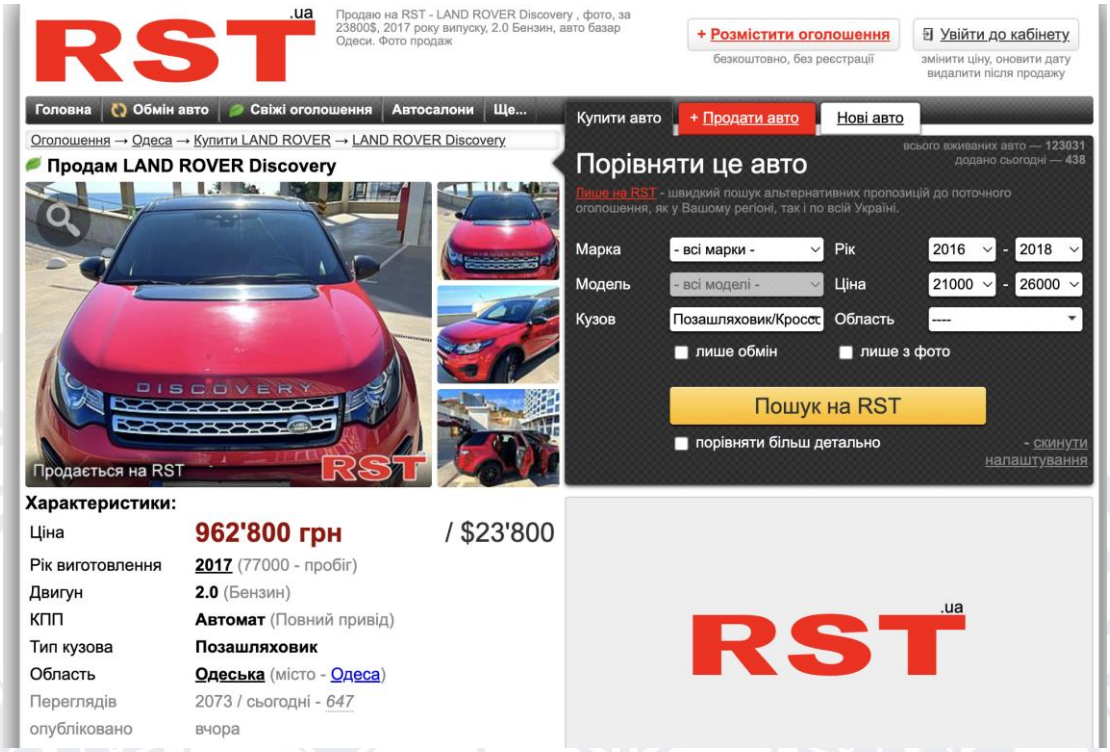

Рисунок 3.3 - користувацький інтерфейс RST.ua

AutoDev Automotive APIs

Прості, потужні та зручні у використанні автомобільні API для розробки хмарних веб- та мобільних додатків для розробників та компаній будь-якого розміру (із сайту розробника). Даний сервіс можна використати додатково до перелічених вище. Як параметр запит приймає VIN код транспортного засобу https://auto.dev/api/vin/ZPBUA1ZL9KLA00848


```
{
 "make": {
  "name": "Lamborghini",
  "niceName": "lamborghini"
 },
 "model": {
  "id": "Lamborghini_Urus",
  "name": "Urus",
  "niceName": "urus"
 },
 "engine": {
  "name": "Gas",
  "equipmentType": "ENGINE",
  "availability": "STANDARD",
  "compressionRatio": 9.7,
  "cylinder": 8,
```

```
  "size": 4,
  "displacement": 3996,
  "configuration": "V",
  "fuelType": "premium unleaded (required)",
  "horsepower": 641,
  "torque": 627,
  "totalValves": 32,
  "type": "gas",
  "code": "8VTTG4.0",
  "compressorType": "twin turbocharger",
  "rpm": {
   "horsepower": 6000,
   "torque": 2250
  },
  "valve": {
   "timing": "variable valve timing",
   "gear": "double overhead camshaft"
  }
 },
 "transmission": {
  "name": "8A",
  "equipmentType": "TRANSMISSION",
  "availability": "STANDARD",
  "automaticType": "Shiftable automatic",
  "transmissionType": "AUTOMATIC",
  "numberOfSpeeds": "8"
 },
 "drivenWheels": "all wheel drive",
 "numOfDoors": "4",
 "options": [
  {
   "category": "Interior",
   "options": [
    {
     "id": "401785427",
     "name": "Bicolor Elegante Leather",
     "description": "Two-tone leather interior",
     "equipmentType": "OPTION",
     "availability": "All"
    },
```


Приклад відповіді AutoDev API

До переваг даного сервісу можна віднести те, що у відповіді ми можемо отримати інформацію про конкретну комплектацію авто, чого не дають інші сайти.

3.2 Розпізнавання транспортних засобів [31]

Технологія розпізнавання номерного знаку набула популярності в останні роки завдяки великій кількості переваг та можливості застосування у різноманітних сферах. Управління дорожнім рухом, розумне паркування, автоматизація стягнення плати, інтелектуальні транспортні системи в розумних містах і аналіз часу в дорозі, відстеження руху транспортних засобів, ідентифікації конкретних автомобілів – це лише деякі з переваг, які ми можемо отримати в результаті. Використання таких систем стає все більш популярним, оскільки технологія швидко розвивається з появою машинного та глибокого навчання, вартість обчислень зменшується, а точність застосованих методів обробки зображень зростає.

АРНЗ – Автоматичне розпізнавання номерних знаків, методика, що використовує оптичне розпізнавання символів у зображеннях номерних знаків транспортних засобів для зчитування реєстраційного номера автомобіля. Автоматична система розпізнавання номерних знаків застосовує різні методи обробки зображень, щоб швидко й автоматично ідентифікувати транспортні засоби на відео чи фотознімках з камер.

Існує два основних типи систем АРНЗ:

- стаціонарні системи – це ті, які встановлено в певному місці та в певний момент часу; наприклад, ці системи можуть бути встановлені на пунктах оплати, прикордонних переходах або в інших стратегічних місцях;
- мобільні системи – це системи, які перевозяться транспортним засобом і можуть використовуватися для сканування номерних знаків інших транспортних засобів; наприклад, ці системи можуть використовуватися поліцейськими автомобілями для відстеження розшукуваних транспортних засобів.

Системи автоматичного розпізнавання номерних знаків (АРНЗ) застосовують оптичне розпізнавання символів у поєднанні з іншими методами

обробки зображень для зчитування номерних знаків транспортних засобів. АРНЗ є однією з найточніших і широко застосовуваних систем комп’ютерного зору. Методи, які використовуються, постійно вдосконалюються, щоб підвищити продуктивність, точність, економічність, надійність і масштабованість програмного забезпечення автоматичного розпізнавання номерних знаків.

Існує кілька важливих методів, що працюють в сукупності в системі АРНЗ. Найважливішими методами системи АРНЗ є:

- виявлення об'єктів у реальному час: виявлення об’єктів використовує глибоке навчання для розпізнавання транспортних засобів і різних класів транспортних засобів (автобус, вантажівка, автомобіль, мікроавтобус, мотоцикл тощо) на зображеннях відеопотоків; сучасні алгоритми виявлення об’єктів, такі як YOLOv3 або YOLOv7, використовують нейронні мережі, навчені на наборі даних зображень;
- обробка зображень: обробка зображень включає традиційні методи комп’ютерного зору, які використовуються для нормалізації та підготовки зображень до обробки за допомогою алгоритму оптичного розпізнавання символів. Оскільки додатки АРНЗ зазвичай використовуються в складних умовах реального світу з різним освітленням, оклюзією, погодою та непослідовними налаштуваннями, функції обробки зображень використовуються для підвищення різкості, корекції кольорів або обрізання зображень, щоб значно покращити результати та вихід наступних алгоритми. OpenCV є однією з найбільш широко використовуваних бібліотек для завдань обробки зображень у системах АРНЗ;
- оптичне розпізнавання символів: оптичне розпізнавання символів (OCR) [32] є важливою технікою комп’ютерного зору для читання тексту із зображень. Це дозволяє системі АРНЗ ідентифікувати номерні знаки. Методи обробки зображень, такі як алгоритми Tesseract OCR або MaskOCR, використовуються для виявлення окремих символів,

перевірки послідовності цих символів і перетворення зображення номерного знака в текст;

- зіставлення шаблонів - це метод порівняння зображення номерного знака з бібліотекою зображень номерних знаків, щоб знайти збіг. Зіставлення за шаблоном — це техніка пошуку певного шаблону в текстовому рядку. У випадку АРНЗ набором символів є номерні знаки автомобіля.

Типові системи АРНЗ включають блок захоплення цифрового зображення (камеру), блок обробки та різні алгоритми для відео аналітики. Процес АРНЗ включає наступні кроки:

- Відео введення та отримання зображень: по-перше, камера АРНЗ захоплює зображення або відео, які містять один або кілька номерних знаків (відео або фото). Часто використовується інфрачервоне освітлення, що дозволяє камерам фіксувати номерні знаки транспортних засобів у нічний час, що дозволяє використовувати АРНЗ у будь-який час дня.
- Виявлення та обрізання номерного знаку: на зображенні номерний знак розпізнається із використанням технологій машинного навчання та комп’ютерного зору. Існують різні методи, які суттєво відрізняються за необхідними обчислювальними ресурсами, складністю, швидкістю та точністю. Загальний метод включає спочатку виявлення транспортних засобів за допомогою виявлення об’єктів, а потім локалізацію номерного знаку в цих обмежувальних прямокутниках. Зазвичай це досягається шляхом пошуку зон контрасту між фоном і номерним знаком. Коли номерний знак виявлено, він обрізається та нормалізується (збільшується різкість, спотворюється та покращується).
- Витяг та читання номерного знаку: до області виявленого номерного знаку застосовується програмне забезпечення оптичного розпізнавання

символів, щоб повернути номерний знак у текстовому форматі. Програмне забезпечення оптичного розпізнавання символів можна оптимізувати для різних наборів символів, що дозволяє використовувати ту саму систему АРНЗ у різних країнах. Результатом системи АРНЗ є номерний знак, часто з ідентифікатором регіону або країни.

- Витяг інформації про номерний знак: нарешті, після перетворення у формат звичайного тексту номерний знак транспортного засобу зберігається в базі даних для інтеграції з іншими IT-системами. Його можна використовувати для порівняння номерного знаку з базою даних зареєстрованих номерних знаків або базами даних білого та чорного списків. У разі виявлення в базі даних програмне забезпечення повертає збережену інформацію про транспортний засіб, наприклад, ім’я та адресу зареєстрованого власника [33].

АРНЗ пропонує численні переваги, які є основою для реальних додатків. Більшість переваг АРНЗ пов’язано з автоматизацією ручних завдань, високоефективним управлінням простором, управлінням і підвищенням досвіду клієнтів.

- Автоматизація: автоматичне розпізнавання номерних знаків дозволяє автоматично подавати сповіщення та контролювати об’єкти. Таким чином, АРНЗ є ключовою технологією для розумних міст або платних станцій.
- Точність: системи АРНЗ можуть досягти дуже високої точності та можуть швидко та легко ідентифікувати транспортні засоби за їхніми номерними знаками.
- Аналітика: згенеровані дані можна використовувати для аналізу потоків трафіку. Це особливо важливо для експлуатації інтелектуальних транспортних систем (ITC), де технології обробки даних використовуються для покращення мобільності людей і товарів,

управління попитом, підвищення безпеки, зменшення заторів на дорогах та ефективного управління інцидентами.

- Ідентифікація: Швидке розпізнавання номерного знаку автомобіля є основою для швидкої та безперебійної ідентифікації автомобіля. Ідентифікацію можна використовувати для надання транспортним засобам доступу або пошуку та відстеження певних транспортних засобів.
- Економічність: точне та швидке розпізнавання номерних знаків не покладається на вхід людини. Велику кількість номерних знаків можна аналізувати дуже швидко, що робить його дуже ефективним методом ідентифікації транспортних засобів. Таким чином, це забезпечує економічно ефективне управління та скорочує час очікування.
- Невелика площа: автоматизована система розпізнавання номерних знаків є відносно економічною для встановлення та експлуатації. Для АРНЗ підходить широкий спектр камер.
- Масштабованість: використання нових технологій, таких як IoT (або AIoT) і машинне навчання на межі (Edge AI), дозволяє розробляти розподілені системи. Граничні обчислення дають змогу реалізувати офлайн-можливості, приватну та економічно ефективну відео-аналітику, необхідну для АРНЗ.
- Зручність: АРНЗ зазвичай інтегрується з іншими IT-системами та працює в екосистемі, щоб забезпечити кінцевим користувачам зручну та безпроблемну роботу. Таким чином, технологія використовується для покращення досвіду клієнтів і пропонування нових послуг і продуктів, таких як автоматизована оплата паркування.
- Універсальність: автоматичну ідентифікацію транспортних засобів можна застосовувати для різноманітних завдань, від керування паркуванням до безпеки, контролю руху, оптимізації логістики у виробництві тощо.
- Безпека: рішення АРНЗ мають велике значення для різноманітних додатків комп’ютерного зору для безпеки та спостереження. Такі системи

допомагають підвищити безпеку, надаючи метод автономної ідентифікації та відстеження кількох транспортних засобів.

- Переваги для навколишнього середовища: використання системи АРНЗ може допомогти зменшити затори та забруднення навколишнього середовища, перешкоджаючи непотрібній їзді. Зменшення часу очікування або часу на пошук паркувальних місць у містах допомагає зменшити забруднення навколишнього середовища.
- Існує низка можливих труднощів, з якими програмне забезпечення повинно впоратися. До них належать:
- погана роздільна здатність зображення, як правило, тому, що пластина розташована на завеликій відстані, але також це буває через використання камери низької якості;
- розмиті зображення, особливо розмиття під час руху;
- погане освітлення та низький контраст через рефлексію, відбиття або тіні;
- об’єкт, що закриває (частину) пластину, часто фаркоп або бруд на пластині;
- інший шрифт (деякі країни не дозволяють такі таблички, усуваючи проблему).

Хоча деякі з цих проблем можуть бути виправлені за допомогою програмного забезпечення, вирішення цих проблем в основному надається апаратній стороні системи. Збільшення висоти камери може уникнути проблем з об'єктами (наприклад, іншими транспортними засобами), що закривають номерний знак, але це створює та посилює інші проблеми, такі як регулювання збільшеного перекосу номерного знаку.

У багатьох країнах зараз використовуються світлоповертаючі номерні знаки. Це повертає світло назад до джерела і, таким чином, покращує контрастність зображення. У деяких країнах символи на табличці не відображають, що забезпечує високий рівень контрасту з відбиваючим фоном за будь-яких умов освітлення. Камера, що використовує інфрачервоне зображення

(зі звичайним кольоровим фільтром над об'єктивом і джерелом інфрачервоного світла поряд з ним), має вигоду з цього, оскільки інфрачервоні хвилі відбиваються назад від пластини. Однак це можливо лише на виділених камерах АРНЗ, тому камери, які використовуються для інших цілей, повинні більшою мірою покладатися на можливості програмного забезпечення. Крім того, коли потрібне повно кольорове зображення, а також використання деталей, отриманих за допомогою АРНЗ, необхідно, щоб одна інфрачервона камера та одна звичайна (кольорова) камера працювали разом.

Розмиті зображення ускладнюють розпізнавання тексту — системи АРНЗ повинні мати коротку витримку, щоб уникнути розмиття зображення під час руху. Щоб уникнути розмиття, витримка камери повинна бути встановлена на 1/1000 секунди. Оскільки автомобіль рухається, нижчі швидкості можуть призвести до того, що зображення буде занадто розмитим, щоб його можна було прочитати за допомогою програмного забезпечення оптичного розпізнавання символів, особливо якщо камера розташована набагато вище за автомобіль. У транспорті, що повільно рухається, або коли камера знаходиться на нижчому рівні, а транспортний засіб наближається до камери під кутом, швидкість затвора не повинна бути такою короткою.

Номерний знак [34]

Для того щоб отримати інформацію по транспортному засобу нам потрібно розпізнати або номерний знак автомобіля, або його VIN код, який є унікальним та незмінним протягом всього життя автомобіля. Для початку розглянемо різні типи номерних знаків транспортних засобів в Україні.
Спеціальні номерні знаки:

Рисунок 3.4 - Номерний знак Поліції зразка 2004

Рисунок 3.5 - Номерний знак Військовий зразка 2004

Рисунок 3.6 - Номерний знак Громадянський зразка 2004

Рисунок 3.7 - Номерний знак Тимчасовий зразка 2004

Реєстраційний знак транспортного засобу в Україні - це металева пластина, що закріплюється на транспортному засобі або причепа для офіційної ідентифікації.

Розмір фактичного однорядкового номерного знаку становить 520 мм х 112 мм.

Дворядковий номерний знак для легкових автомобілів з квадратним місцем кріплення - 300 мм х 150 мм.

Номерні знаки старого стандарту:

Рисунок 3.8 - Номерний знак зразка 1995

Рисунок 3.9 - Нестандартний номерний знак зразка 1995

Рисунок 3.10 - Номерний знак зразка 2004

Поки номерні знаки старого зразка є законними для використання, під час будь-якої операції з автомобілем:

- зміні власника
- перереєстрації
- технічному огляді;

На нього встановлюються нові номерні знаки зразка 2015 року з кодом країни і прапором на синьому фоні. Номерні знаки нового зразку:

Рисунок 3.11 - Нестандартний номерний знак нового зразка 2015

Рисунок 3.12 - Номерний знак нового зразка 2015

Слідуючи Wikipedia:

З метою надання можливості водіям, які користуються транспортними засобами за кордоном, та на виконання Віденської конвенції про дорожній рух, в українських державних номерних знаках використовуються тільки ті символи кирилиці, де гліф нагадує літеру латинського алфавіту; всього 12 символів: A, B, E, I, K, M, H, O, P, C, T,

X).

Отже, формат звичайного номерного знака автомобіля – AA0000BB, де AA – регіональний номер префікс, 0 - цифра (0-9), потім BB, послідовний суфікс. Воно має бути однаковим на номерному знаку та в документах Рисунок 3.13 Але, як було сказано раніше, найточнішою ідентифікацією є VIN автомобіля.

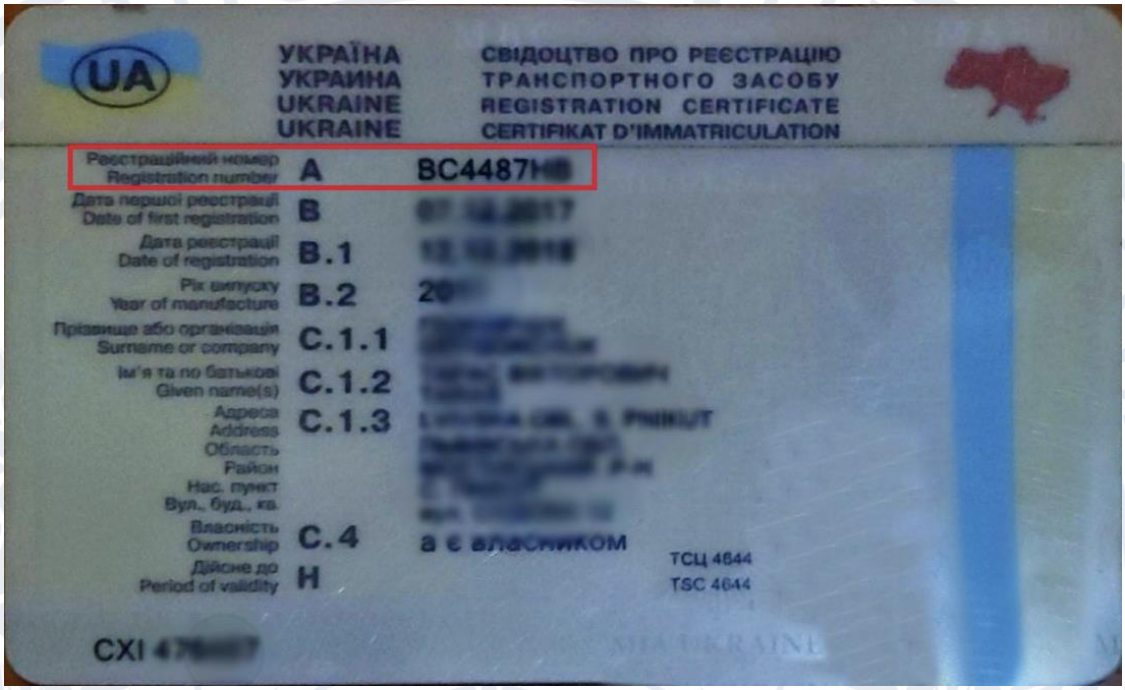

Рисунок 3.13 - Приклад номерного знаку авто

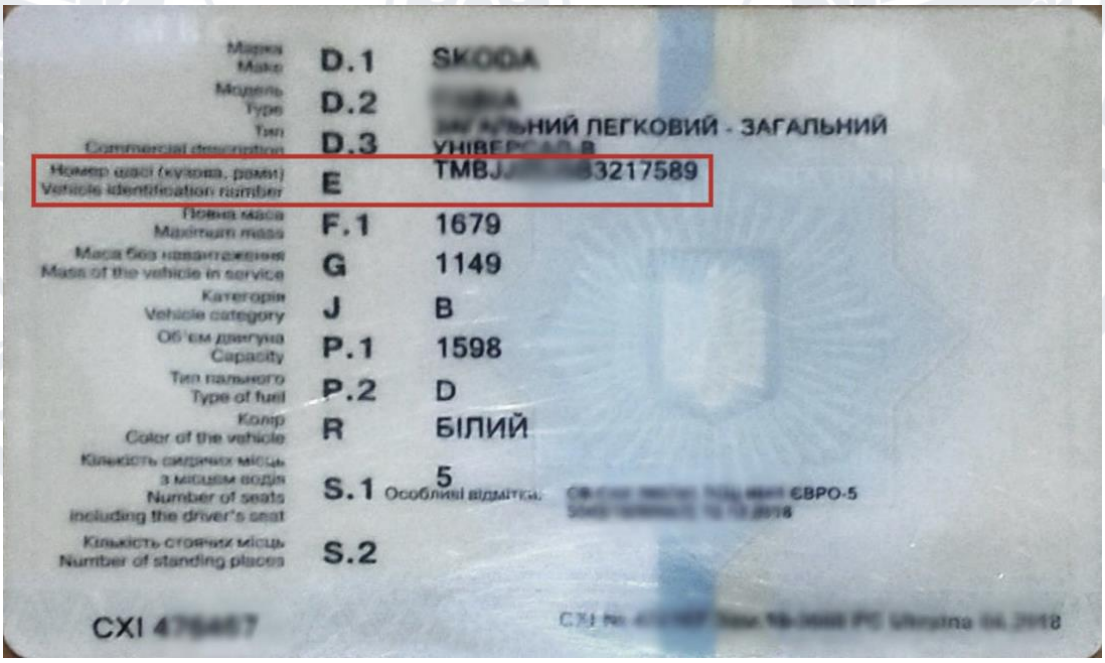

Рисунок 3.14 - Приклад VIN коду авто

VIN код

VIN - ідентифікаційний номер транспортного засобу складається з 17 символів (цифри та великі літери), які виступають в якості унікального ідентифікатора транспортного засобу. VIN відображає унікальні особливості автомобіля, його технічні характеристики та виробника. Він може бути використаний для відстеження відкликань, реєстрацій, гарантійних претензій, крадіжок, та страхового покриття. Комбінація цифр і букви VIN-коду захищена від перебивки номерів шляхом застосування алгоритму підрахунку контрольного числа, який підтримується всіма виробниками автомобілів у США та експортерів у цю країну, в Європі стандарт ISO 3779 дотримується не завжди. Відповідно до стандарту ISO 3779 VIN-код із сімнадцяти символів. Для позначення VIN-коду дозволені наступні символи: *0 1 2 3 4 5 6 7 8 9 A B C D E F G H J K L M N P R S T U V W X Y Z* Символи *I, O, Q* заборонені у вигляді схожих на 0 і 1 і між собою. Значення символів та приклад розшифрування автомобіля Skoda Octavia:

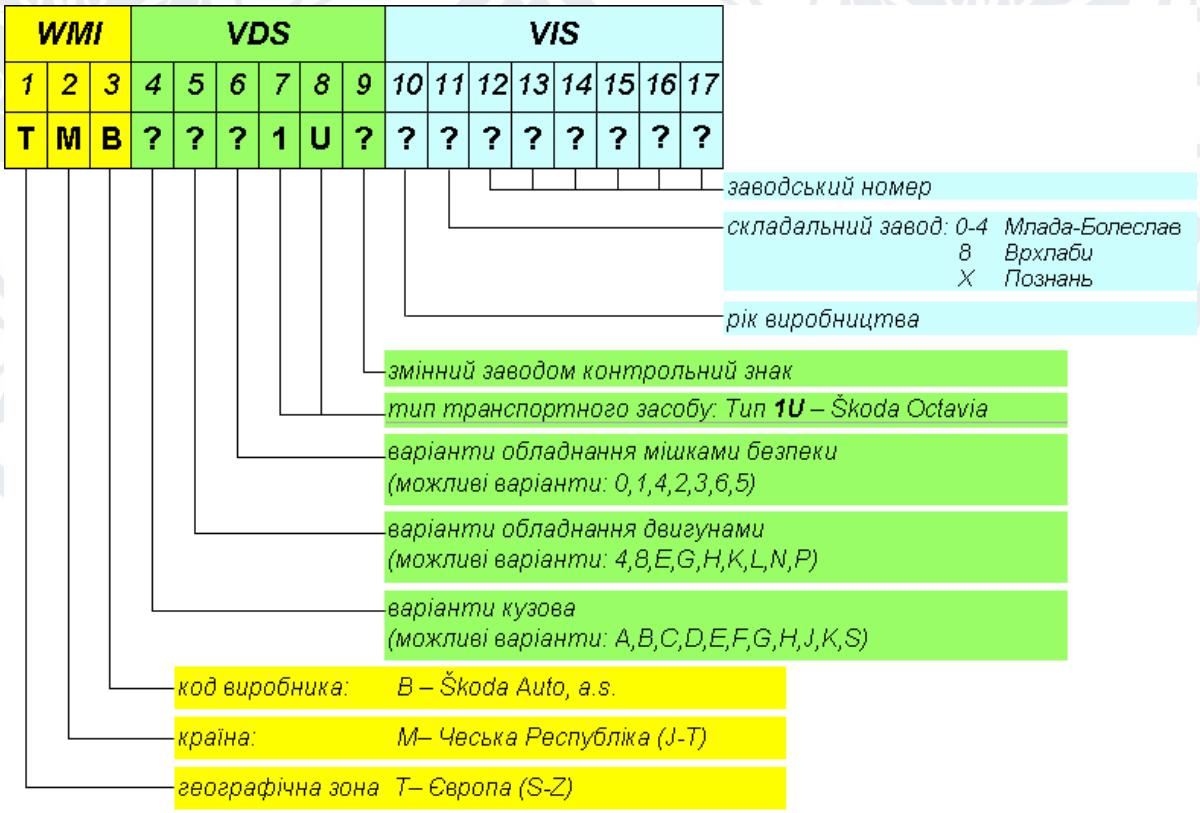

Рисунок 3.15 - Значення VIN коду автомобіля Skoda Octavia

Це лише загальні місця розташування VIN автомобіля, зараз виробники наносять наклейки та ідентифікатори на набагато більшу кількість. традиційними місцями нанесення є блок VIN і головка блоку циліндрів, стійки кузова, дверні пороги, перегородка між двигуном і салоном, а в машині з рамною конструкцією (в основному це позашляховики). Це індивідуально для самого виробника, але всі вони мають хоча б одну з ділянок. На американських автомобілях VIN-код, як правило, розташований з лівої сторони водія в тому місці, де лобове скло з'єднується з капотом і дублюється на наклейці, що знаходиться в проміжку водійської двері. Крім того, він може дублюватися і в інших місцях. Реальні фотографії найбільш поширених місць розташування:

- На приладовій панелі, під лобовим склом, видно ззовні (Рисунок 3.16)
- Зліва вгорі, під капотом, на верхньому кріпленні підвіски (Рисунок 3.17)
- Нижня частина косяка дверей водія (Рисунок 3.18)

Рисунок 3.16 - VIN код під лобовим склом

Рисунок 3.17 - VIN код під капотом авто

Рисунок 3.18 - VIN код біля водійських дверей

3.3 Реалізація клієнтської частини

Базова архітектура

Є багато способів побудови складних систем із правильною архітектурою. Незважаючи на невеликі відмінності між цими методами, вони мають багато спільного. Звичайно, вони обидва визначають способи поділу програми на окремі модулі. При цьому кожна система має принаймні модуль, який містить програмну бізнес-логіку, і модуль, який відображає дані. Кожен підхід дозволяє створити систему, яка відповідає наступним принципам:

1. Архітектура має бути незалежною від різних фреймворків. Звичайно, в сучасному світі не обійтися без бібліотек, які дозволяють вирішувати роботу простіше і ефективніше, але тут важливо розуміти, що бібліотека має бути інтегрована у вашу архітектуру, а не адаптувати архітектуру до обраної бібліотеки. Використовуючи бібліотеку, яка змушує перебудувати програму, будуть виникати обмеженнями цієї бібліотеки та неспромога перебудувати так, як потрібно. Бібліотеки призначені для використання лише як допоміжні засоби.
2. Система повинна бути протестована, водночас має бути можливість протестувати ці два модулі системи окремо, а також перевірити взаємодію між цими модулями та їх інтеграцію в систему. Крім того, систему потрібно протестувати без інтерфейсу користувача, реального сервера та з використанням бази даних, тобто архітектура має бути незалежною від середовища.
3. Аналізуючи попередні пункти, можна зрозуміти такі принципи, які говорять про те, що програма має бути незалежною від усього: від інтерфейсу користувача, від використання бази даних, від сервера та інших елементів середовища. Незалежність архітектури від середовища важлива, оскільки вона дозволяє змінювати різні компоненти середовища без зміни самої архітектури. Що означає змінити компоненти архітектури? Це може бути зміна вибору бази даних (або взагалі відхилення) або зміна частини інтерфейсу користувача програми (наприклад, вам потрібно змінити вигляд екрана).

Clean Architecture Android [35]

З точки зору тестування програми, найбільш критично є бізнес-логіка або бізнес-правила, які визначають характер роботи додатка. Для початку їх необхідно протестувати незалежно від інших компонентів. Для досягнення незалежності та можливості тестування рекомендується розділити додаток на 3 ключові рівні.

1. Рівень даних (Data Layer)
2. Рівень бізнес-логіки (Domain Layer)
3. Рівень представлення (Presentation Layer)

У той же час, щоб гарантувати максимальну незалежність цих рівнів, кожен рівень використовує власну модель даних, яка трансформується при взаємодії з шарами. Схема цих шарів виглядає наступним чином [36]:

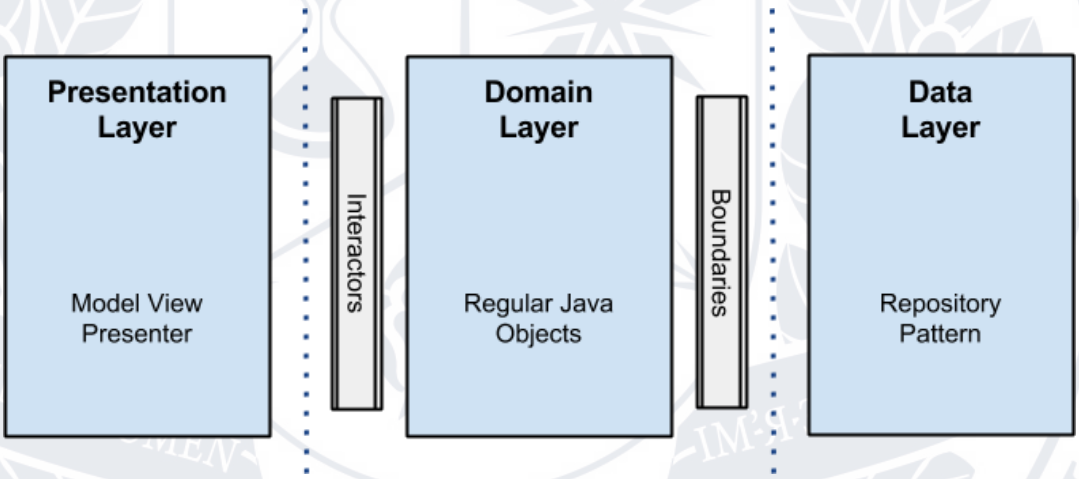

Рисунок 3.19 – Clean Architecture. Шари архітектури

Також, окрім підходу до організації коду, призначення класів за завданнями, слід звернути увагу на архітектурні підходи до реалізації та взаємодії об’єктів. У цій задачі використовується шаблон проектування систем - MVVM.

Model-View-ViewModel (тобто MVVM) [50] - це архітектурний шаблон клієнтської програми, запропонований Джоном Госсманом (John Gossman) як альтернатива шаблонам MVC і MVP при використанні технології зв'язки даних (Data Binding). Його концепція полягає у тому, що відокремити логіку

представлення даних від бізнес-логіки, помістивши її в окремий клас для більш чіткого поділу. Розглянемо значення кожної із трьох частин назви паттерну:

- Model - це логіка, пов'язана з даними програми. Іншими словами - це POJO, класи API, бази даних тощо.
- View - фактично, це layout екрану, з усіма віджетами, необхідними для відображення інформації.
- ViewModel - об'єкт, який описує логіку поведінки View в результаті роботи Model. Його можна назвати моделлю поведінки View. Він може бути текстовому форматі, так і у форматі логіки керування видимістю або станом відображення компонентів (наприклад - завантаження, помилка, порожній екран тощо). Він також описує дії, які були ініційовано користувачем (введення тексту, натискання кнопок, жести, тощо)

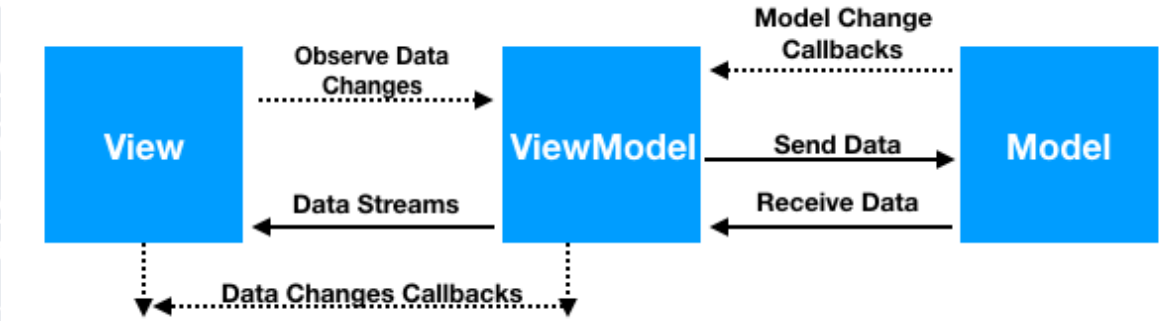

Рисунок 3.20 - Стандартна схема взаємодії компоненті паттерну MVVM

Які переваги в даного підходу [37]:

- Гнучкість розробки. Цей метод підвищує зручність командної роботи, так як коли один член команди займається компонуванням і оформленням екрана – інший описує логіку отримання даних і обробки даних у цей момент;
- Тестування. Така побудова структури спрощує процес написання тестів і процес створення mock-об'єктів. Крім того, найчастіше немає потреби в автоматизованому тестуванні інтерфейсу користувача, оскільки можна обернути саму ViewModel unit-тестами;

- Логічне розділення. Більше розділення робить код більш гнучким і зручним для обслуговування, не кажучи вже про те, що він стає більш читабельним. Кожен модуль відповідає лише за свою певну функцію.

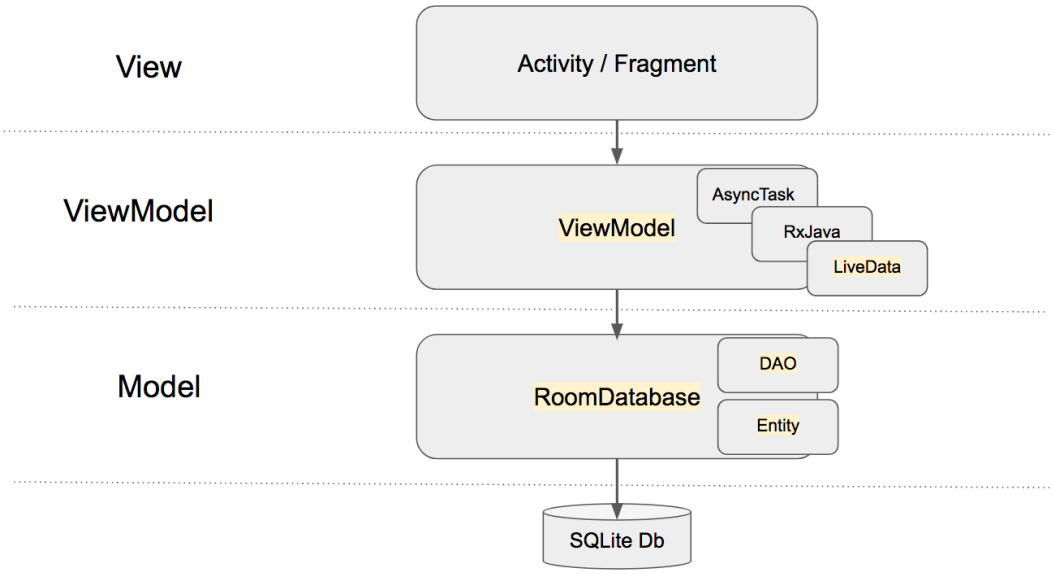

Рисунок 3.21 – Відношення компонентів у архітектурі MVVM

3.4 Реалізація серверної частини

Структура серверної частини розділена на модулі за призначенням.

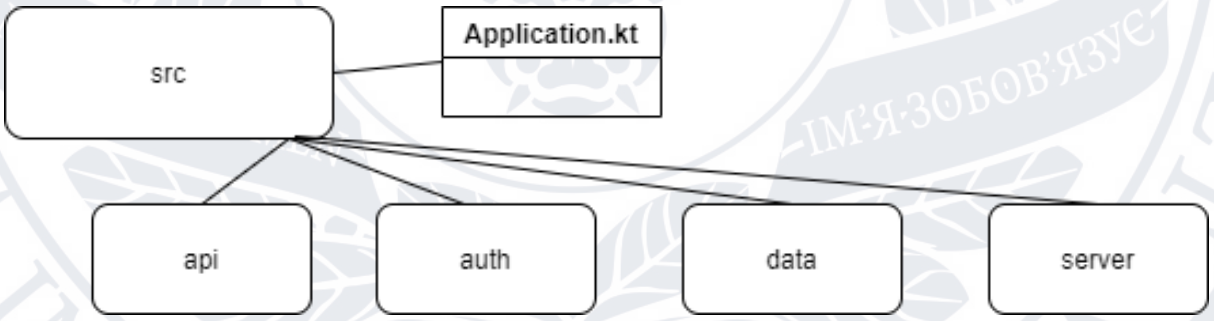

Рисунок 3.22 – Структура програми серверної частини

Модуль побудови запитів

Цей модуль є відповідальним за взаємодію користувача з сервером REST [38].

- Request – об’єкт, який ми приймаємо в тілі запитів.
- Response – об’єкт відповіді, який ми надсилаємо, коли запит буде виконано успішно

- Routing – функціональність, розроблена для обробки конкретних запитів
- Kit – містить константи, необхідні для обробки помилок, функції для перетворення рядків у Json потрібного формату

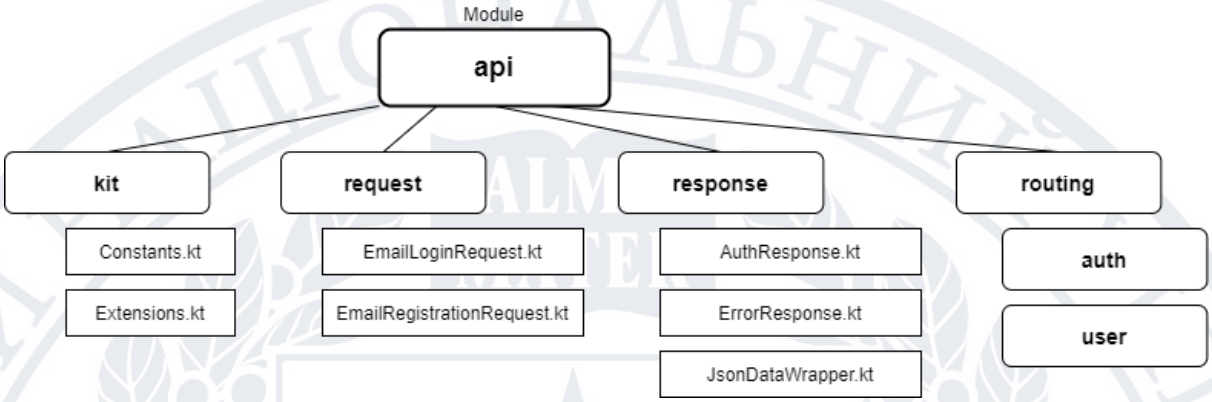

Рисунок 3.23 – API модуль серверної частини

Модуль авторизації користувача

Містить реалізацію криптографічного алгоритму хешування і додаткові функції, які використовуються в класі JWTService для генерації токенів JWT [26].

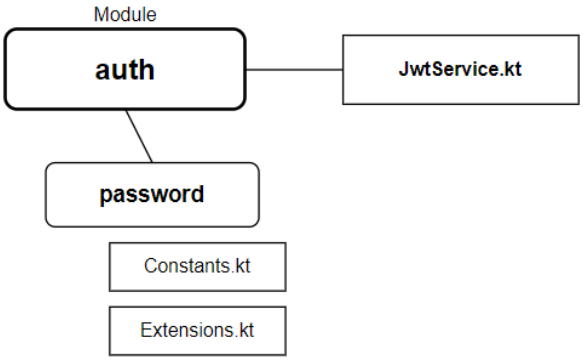

Рисунок 3.24 – Authentication модуль серверної частини

Server модуль (Конфігураційний модуль)

Містить назви змінних конфігурації для бази даних, ключів JWT, тощо. Він також містить допоміжні функції для роботи зі змінними конфігурації на тестових і віддалених серверах [39].

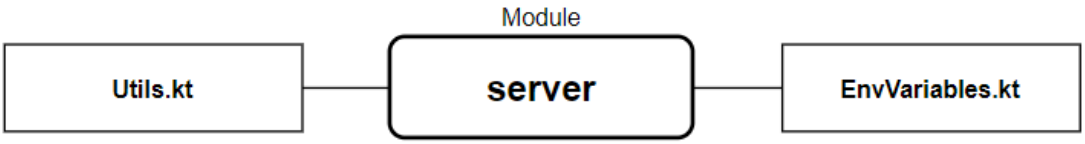

Рисунок 3.25 – Конфігураційний модуль

Модуль зберігання даних

Містить реалізацію сховища, необхідного для обробки запитів користувачів, а також налаштування PostgreSQL бази даних і опис таблиць до неї [40].

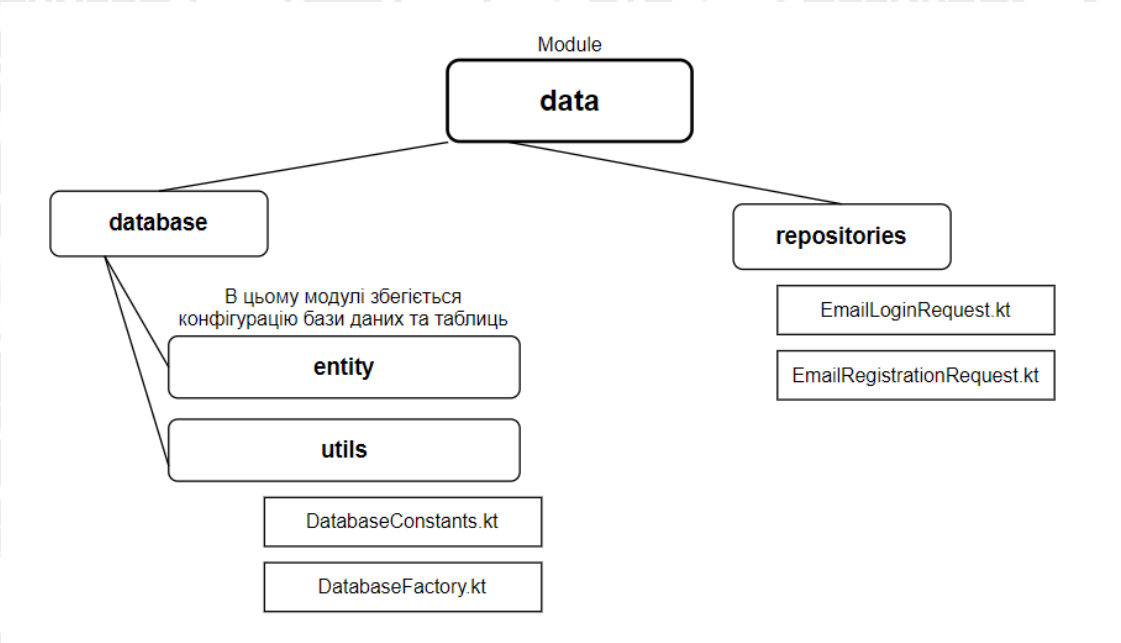

Рисунок 3.26 – Модуль зберігання даних

3.5 Огляд користувацького інтерфейсу

Початковий екран та екрани авторизації

Даний екран користувацького інтерфейсу містить наступний функціонал:

- Можливість авторизуватись за допомогою сервісів Google або Facebook

- Перейти далі на етап авторизації за допомогою електронної пошти та пароля
- Пропустити етап авторизації (у даному випадку користувач не буде мати змогу переглядати історію пошуку та вподобаних транспортних засобів)
- При натисканні системної кнопки «Назад» додаток закриється та втратить свій актуальний стан.

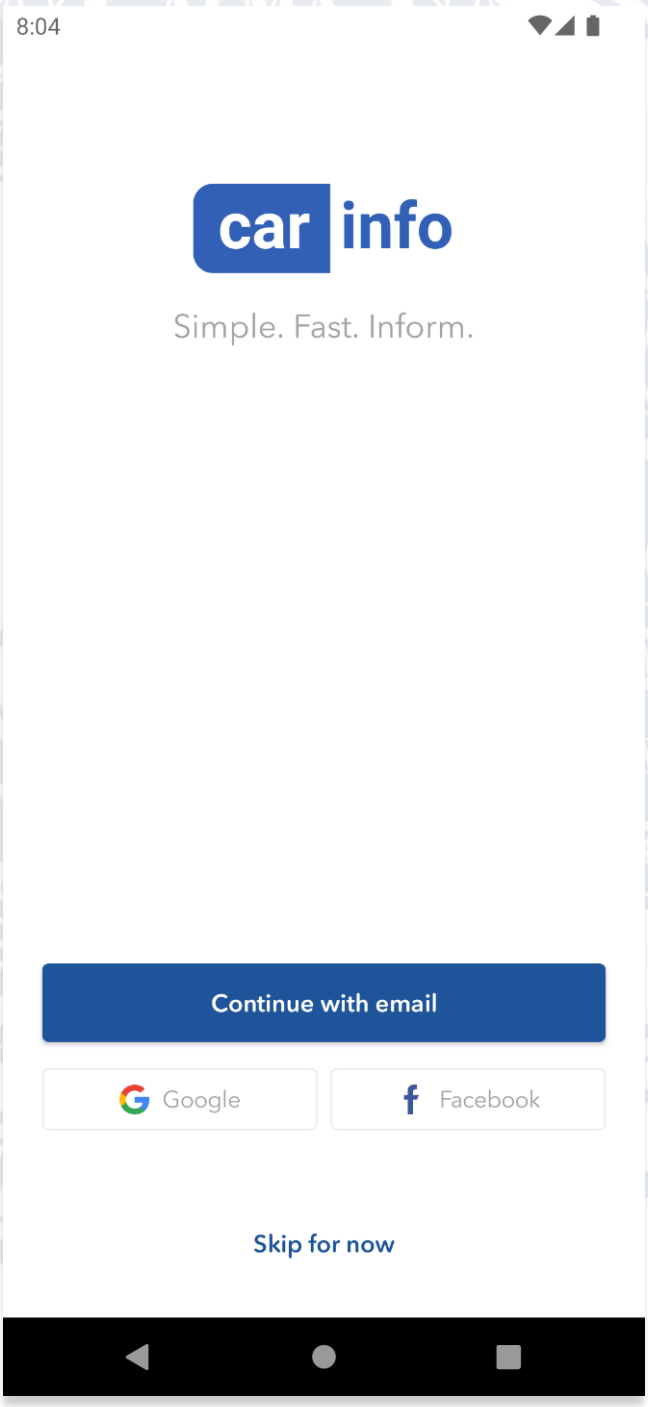

Рисунок 3.27 – Початковий екран

Екран входу та реєстрації через електронну пошту

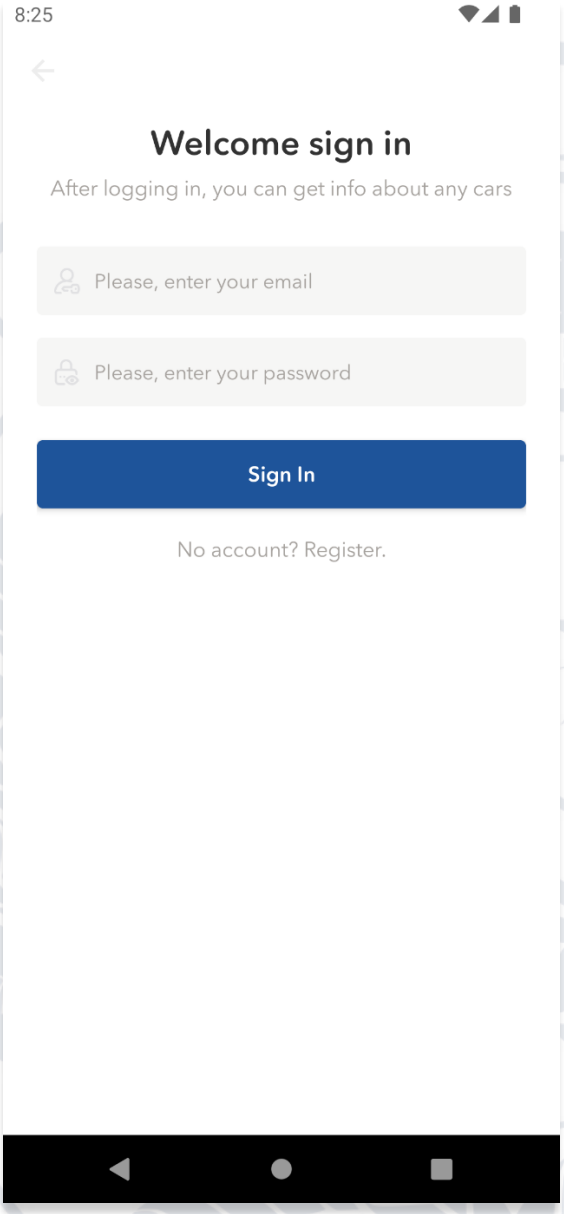

Рисунок 3.28 – Екран входу

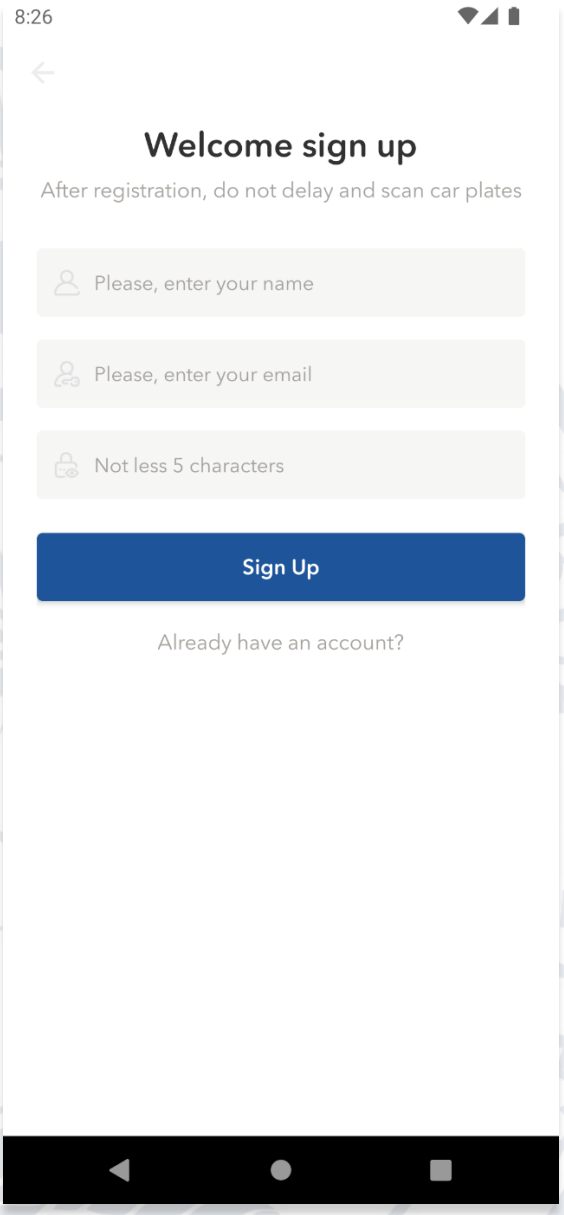

Рисунок 3.29 – Екран реєстрації

Дані екрани надають можливість авторизуватись за допомогою електронної пошти шляхом входу в існуючий акаунт, або за допомогою створення нового. Після входу користувач буде мати можливість переглядати історію пошуку та вподобаних транспортних засобів. Перехід на екран реєстрації відбувається із екрану входу.

Головна сторінка

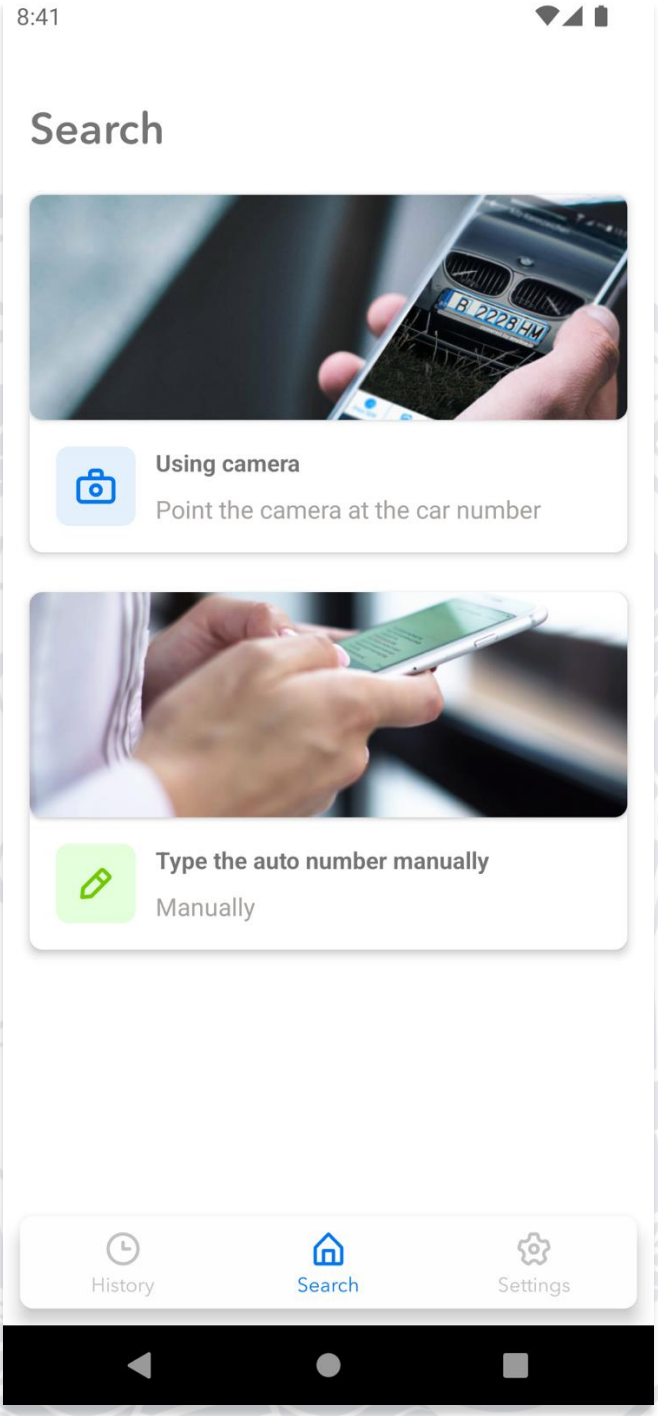

Рисунок 3.30 – Головний екран

Даний екран має наступні можливості:

- Перехід до сторінки сканування номерного знаку
- Перехід до сторінки вводу номерного знаку власноруч
- Перехід до налаштувань
- Перегляд історії пошуку (у випадку, якщо користувач авторизований)

Екран сканування номерних знаків транспортного засобу

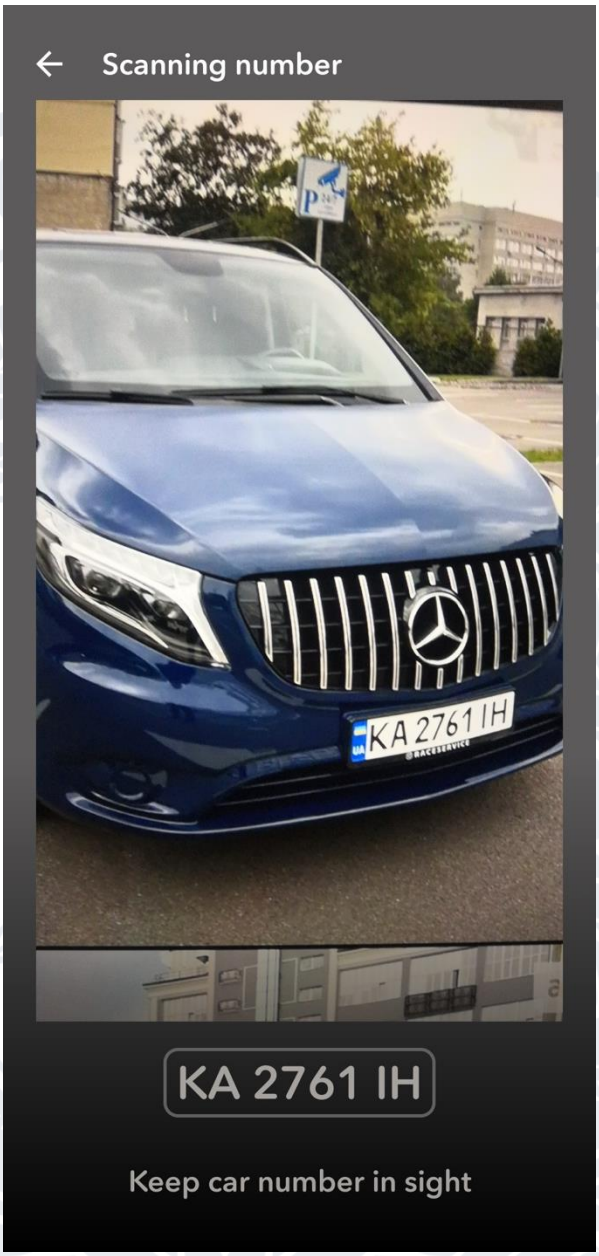

Рисунок 3.31 – Екран сканування номерного знаку

Цей екран надає нам можливість сканувати номерні знаки потрібного нам транспортного засобу. Якщо система розпізнає номерний знак – він з’явиться у нижній частині екрану, після натискання на який ми перейдем на сторінку інформації по даному транспортному засобу. У випадку, якщо у кадрі буде одразу два або більше автомобіля, система дасть змогу обрати один із них для отримання даних.

Екран зібраної інформації про транспортний засіб

Так як даний екран містить досить багато елементів, розглянемо їх окремо.

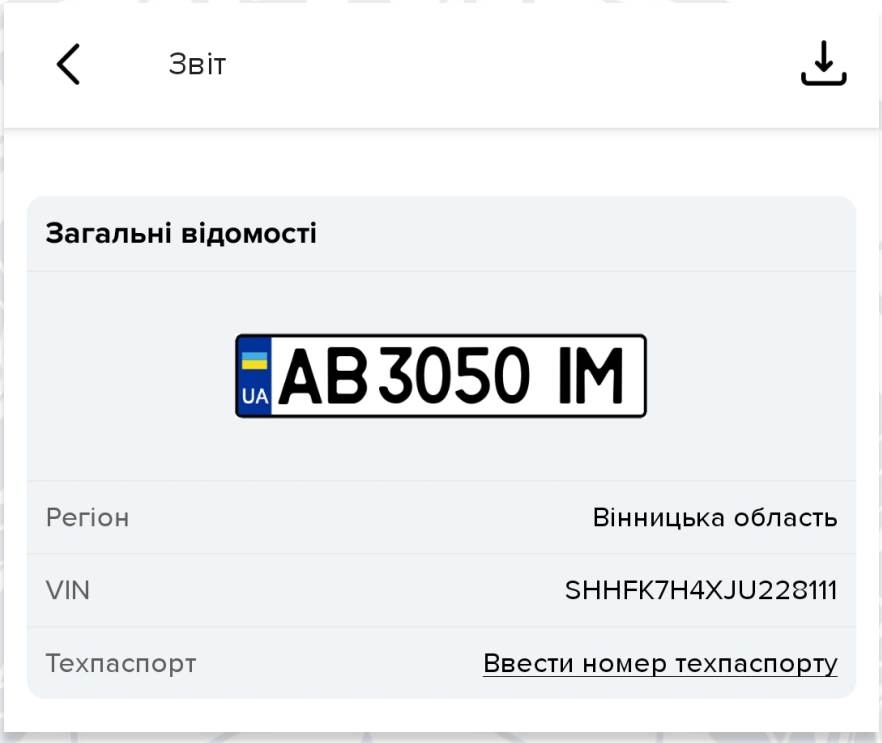

Рисунок 3.32 – Загальні відомості про транспортний засіб

Даний блок відображає загальні відомості про транспортний засіб, а саме:

- *Номерний знак*
- *Регіон, де зареєстрований ТЗ*
- *VIN код (за допомогою нього отримується більшість інформації)*

Технічні характеристики	
Марка	HONDA
Модель	CIVIC
VIN	SHHFK7H4XJU228111
Колір	БІЛИЙ
Кузов	ХЕТЧБЕК
Об'єм двигуна	1498 см^3
Рік випуску	2018 г
Департамент	ТСЦ 0541
Код департаменту	12229
Тип палива	БЕНЗИН
Номер шассі	SHHFK7H4XJU228111
Повна маса	1795 кг

Рисунок 3.33 – Технічні характеристики

Даний блок дає змогу дізнатись наступну інформацію про шуканий ТЗ:

- *Марка*
- *Модель*
- *Колір*
- *Тип кузова*
- *Об’єм двигуна*
- *Тип палива*
- *Повна маса*
- *Номер шасі*
- *Код департаменту (місця, де були видані реєстраційні номера)*
- *Департамент*
- *Рік випуску*
- *VIN код*

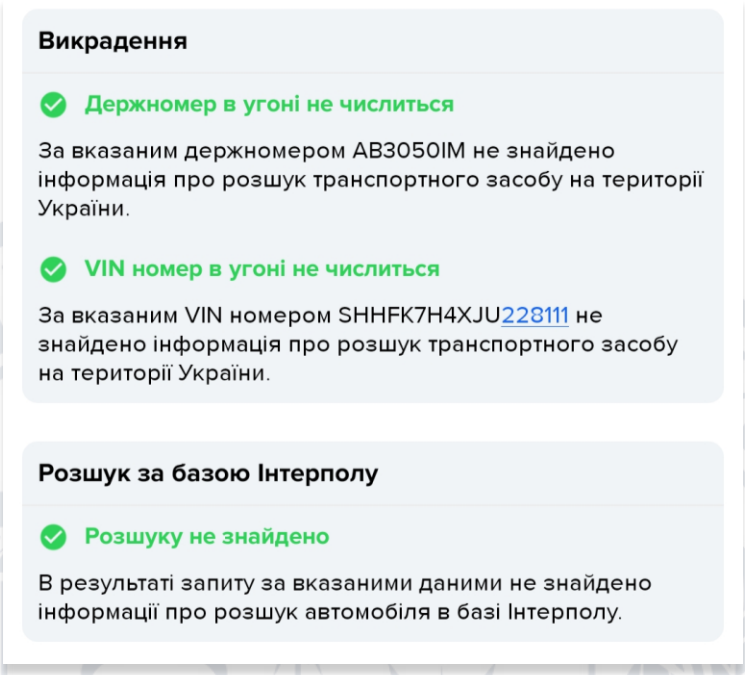

Рисунок 3.34 – Законність ТЗ

Даний блок відображає інформацію про те, чи є транспортний засіб у базах даних про викрадення та базах даних Інтерполу. Дана функція є особливо корисною при покупці автомобіля.

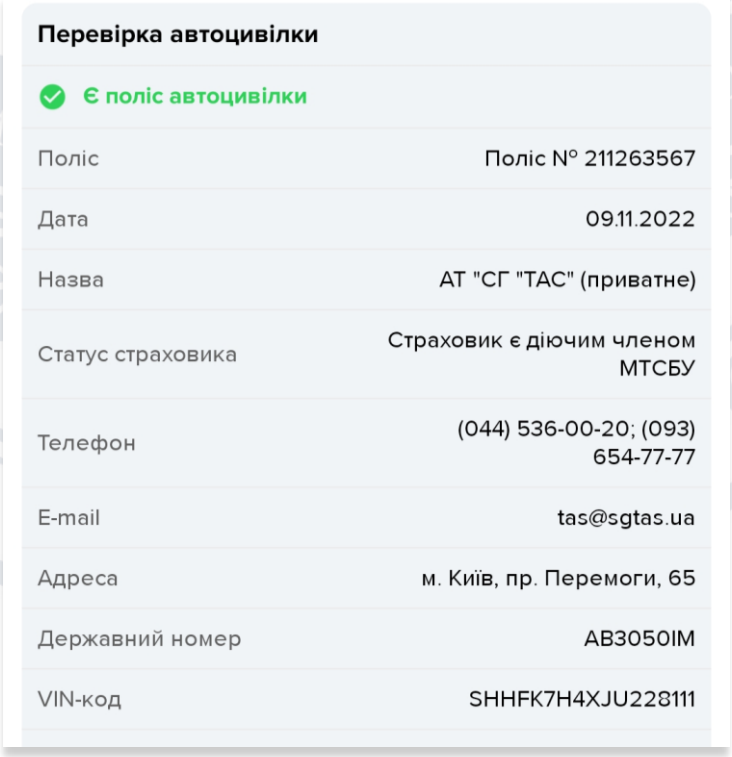

Рисунок 3.34 – Інформація про страхування

У даному блоці можемо бачити інформацію про страхування даного транспортного засобу, а саме наступне:

- *Номер полісу*
- *Назву страхової компанії*
- *Телефон страхової компанії*

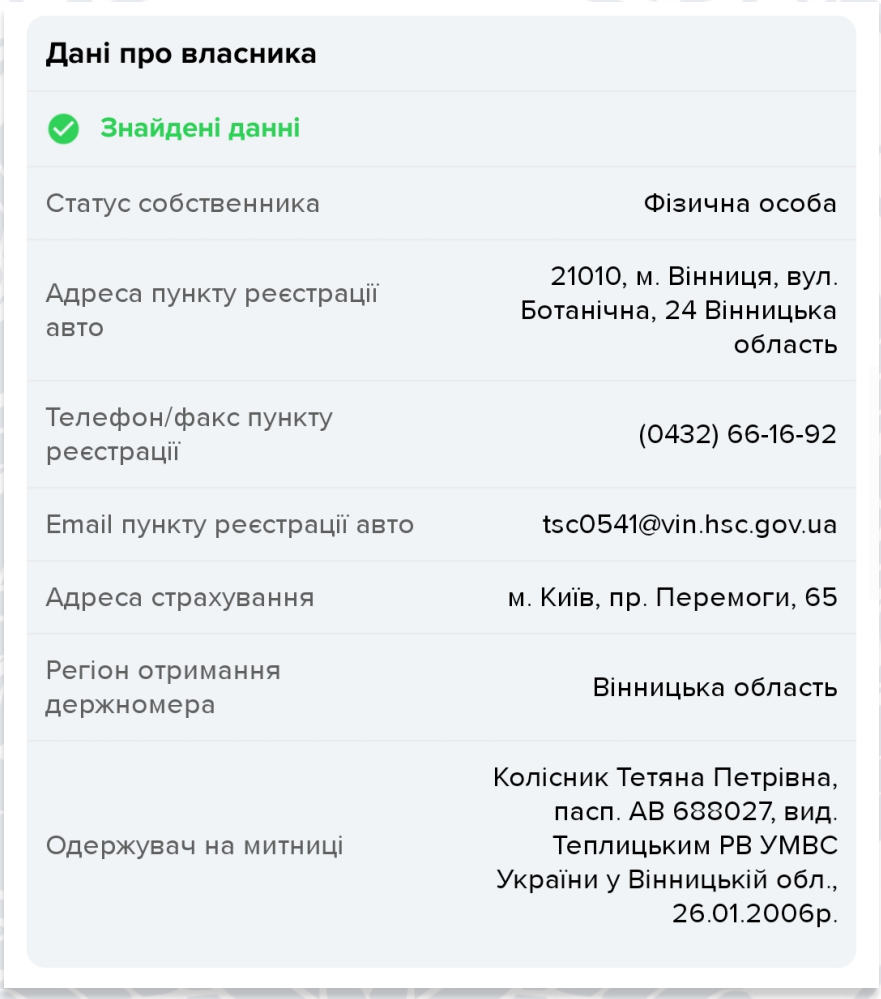

Рисунок 3.35 – Інформація про власника

Даний розділ містить інформацію про власника. А саме:

- *Статус власника*
- *Адресу реєстрації ТЗ*
- *Адресу страхування*
- *Телефон пункту реєстрації ТЗ*
- *Регіон отримання номерного знаку*

Характеристики транспортного засобу

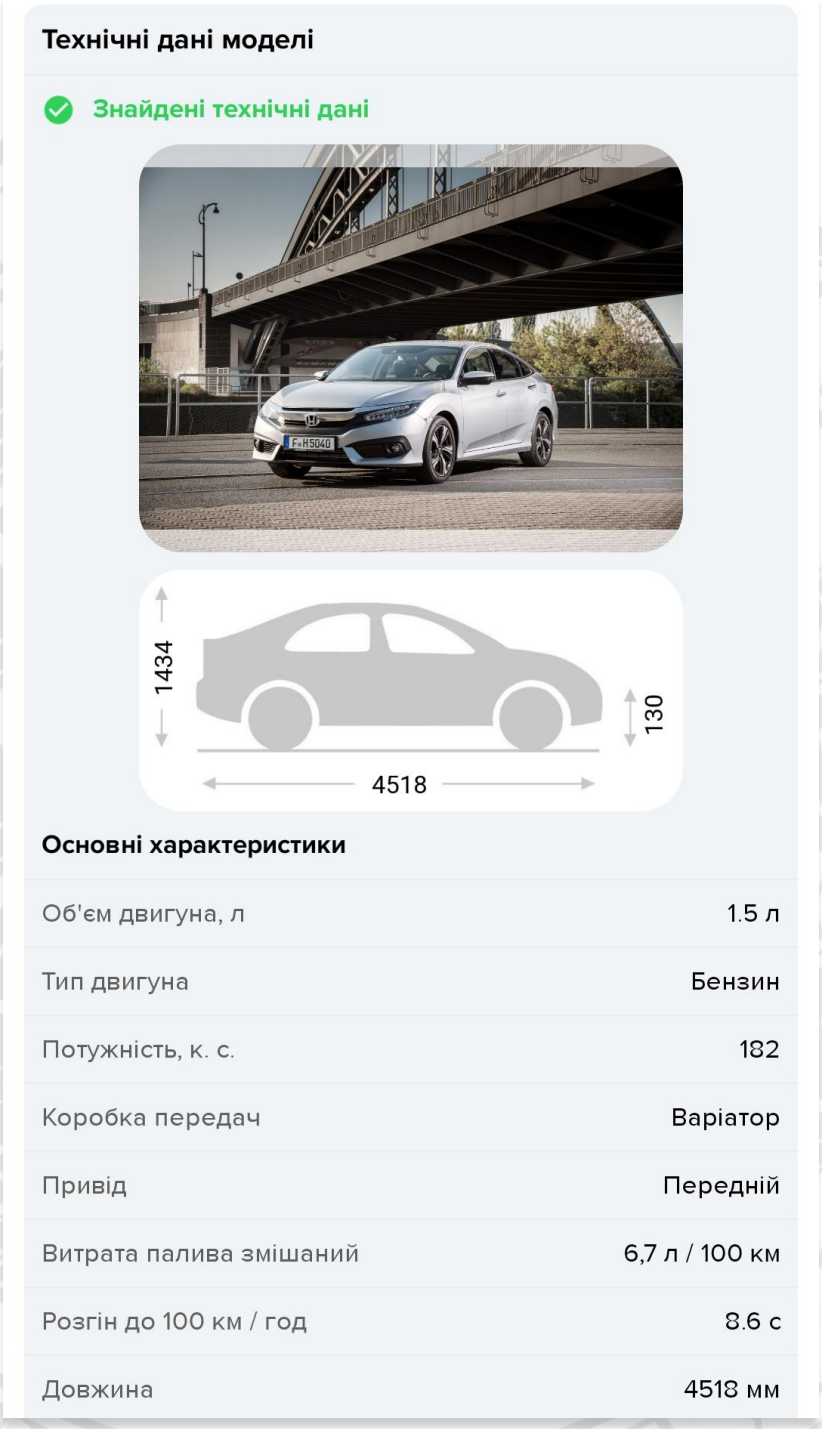

Рисунок 3.36 – Характеристики ТЗ

Даний блок відображає максимальну кількість технічних характеристик, які можливо було отримати за допомогою VIN коду.

Інформація про участь в аукціоні

Участь в аукціоні

2 записи

Дата:	2021-02-15
Марка, модель:	HONDA CIVIC HATCHBACK SPORT
Актуальна вартість:	н.д.
Вартість ремонту:	16670
Стан:	Front Wheel Drive
Розташування:	Grand Rapids (MI)49315
Пробіг:	36380
Основне пошкодження:	Front End
Інші ушкодження:	Left & Right Side

Рисунок 3.371 – Інформація про участь в аукціоні

Фотографії автомобіля 11

Рисунок 3.37.2 – Фото участі в аукціоні

Даний блок відображає інформацію та фото про участь ТЗ в аукціонах різних країн. Це особливо важливо при купівлі авто, коли потрібно побачити стан авто та його пошкодження до ремонту.

Висновок до розділу 3

Даний розділ присвячено програмній реалізації додатку, а саме: державні та приватні джерела даних про транспортні засоби, методи та технологію розпізнавання номерних знаків ТЗ. Також було освітлено реалізацію серверної та клієнтської частини, огляд користувацького інтерфейсу, розгляд функціоналу кожного екрану.

ВИСНОВКИ

У рамках даної кваліфікаційної роботи було розроблено додаток для розпізнавання транспортних засобів та збору інформації під платформу Android. Для авторизації та агрегації зібраної інформації було розроблено серверну частину згідно REST концепції за допомогою платформи Ktor під керуванням мови програмування Kotlin. Даний додаток дає змогу за допомогою номерних знаків транспортних засобів або серійного номеру отримати максимально-доступну по ним інформації із державних та приватних джерел.

У першому розділі було досліджено:

- Аналіз предметної області, актуальність обраної платформи
- Постановка задачі, визначений необхідний функціонал
- Проаналізовано уже існуючі рішення, визначено їх переваги та недоліки
- Визначено інструменти для написання клієнтської та серверної частини

У другому розділі було освітлено технології для реалізації додатку. Виявлено їх переваги, сучасність. Наведено приклади використання даних підходів. Розглянуто технологію для використання технік машинного навчання на платформі Android.

У третьому розділі описано методи розпізнавання, їх відмінності. Розглянуто державні та приватні джерела даних, де буде відбуватись агрегація інформації для представлення користувачу. Освітлено реалізацію клієнтської частини, принцип організації архітектури та реалізацію серверної частини, внутрішню взаємодію її модулів. Також було розглянуто користувацький інтерфейс та функції доступні користувачу.

В результаті було розроблено додаток із наступними функціями:

- Авторизація за допомогою пошти або через сервіси Google та Facebook.
- Розпізнавання номерного реєстраційного знаку транспортного засобу або ввід вручну. У разі розпізнання одразу декількох транспортних засобів має бути вибір одного із них.

- Введення серійного номеру (VIN коду) транспортного засобу замість реєстраційного знаку.
- Отримання інформації по вказаному транспортному засобу, а саме:
    - Марка, модель та рік виготовлення
    - Колір
    - Тип палива, об’єм двигуна
    - Середня ціна на ринку автомобілів (береться із відкритого Autoria API)
    - Інформація про реєстрацію (область та дата реєстрації)
    - Інформація про викрадення
    - Інформація про участь на аукціонах, інформація про пошкодження
- Можливість перегляду раніше оглянутих транспортних засобів та вподобаних ТЗ.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Еволюція мобільних телефонів URL: https://tech.informator.ua/2018/11/25/evolyutsiya-mobilnyh-telefonov-chem-udivlyali-razrabotchiki/ (дата звернення: 31.05.2021)
2. Android URL: https://en.wikipedia.org/wiki/Android_(operating_system) (дата звернення: 31.05.2021)
3. Apple URL: https://en.wikipedia.org/wiki/Apple_Inc. (дата звернення: 31.05.2021)
4. Android vs Apple URL: https://www.businessnewsdaily.com/9906-ios-vs-android-business.html (дата звернення: 31.05.2021)
5. Mobile vs Web URL: https://www.uzhnu.edu.ua/uk/news/-yak-pratsyuye-distantsijne-navchannya-v-uzhnu.htm (дата звернення: 31.05.2021)
6. Android Studio URL: https://developer.android.com/studio (дата звернення: 31.05.2021)
7. Android Studio Features URL: https://developer.android.com/studio/features (дата звернення: 31.05.2021)
8. Programming Android with Kotlin: Achieving Structured Concurrency with Coroutines (Early Release), O'Reilly Media, Pierre-Olivier Laurence, Amanda Hinchman-Dominguez, Mike Dunn, 2021, 524p.
9. On Java 8, Leanpub, Bruce Eckel, 1241p
10. Kotlin at a Glance: Use of Lambdas and higher-order functions to write more concise, clean, reusable, and simple code, BPB Publication, Swati Saxena, 97p
11. Kotlin Coroutines URL: https://kotlinlang.org/docs/coroutines-overview.html (дата звернення: 31.05.2021)
12. Navigation Components URL: https://developer.android.com/guide/navigation (дата звернення: 31.05.2021)
13. Dependency Injection URL: https://www.programmer-books.com/wp-content/uploads/2019/01/Dependency-Injection (дата звернення: 31.05.2021)

14. Dependency Injection - Hilt URL: https://developer.android.com/training/dependency-injection/hilt-android (дата звернення: 31.05.2021)
15. Retrofit URL: https://square.github.io/retrofit/ (дата звернення: 31.05.2021)
16. Android KTX URL: https://developer.android.com/kotlin/ktx (дата звернення: 31.05.2021)
17. Recycler View URL: https://developer.android.com/jetpack/androidx/releases/recyclerview (дата звернення: 31.05.2021)
18. ML Kit. Object Detection URL: https://developers.google.com/ml-kit/vision/object-detection (дата звернення: 30.11.2022)
19. Android Studio Game Development: Concepts and Design, Apress, Jerome DiMarzio, 23p
20. Ktor URL: https://ktor.io/ (дата звернення: 31.05.2021)
21. Elements Of Kotlin, CommonsWare, Mark L. Murphy, 2021, 174p.
22. REST URL: https://docs.marklogic.com/guide/rest-dev.pdf (дата звернення: 31.05.2021)
23. PostgresSql URL: https://en.wikipedia.org/wiki/PostgreSQL (дата звернення: 31.05.2021)
24. PostgresSql URL: http://sd.blackball.lv/library/PostgreSQL_Server_Programming_Second_Edition_(2015).pdf (дата звернення: 31.05.2021)
25. Kotlin Apprentice (3rd Edition), Razeware LLC, Irina Galata, Joe Howard, Ellen Shapiro, 235
26. JWT Handbook URL: https://assets.ctfassets.net/2ntc334xpx65/o5J4X472PQUI4ai6cAcqg/13a2611de03b2c8edbd09c3ca14ae86b/jwt-handbook-v0_14_1 (дата звернення: 31.05.2021)
27. Programming Kotlin Applications: Building Mobile and Server-Side Applications with Kotlin, Wiley, Brett McLaughli, 2021, 145p.

28. Java to Kotlin (Early Release), O'Reilly Media, Duncan McGregor, Nat Pryce, 2021, 313.
29. Df Android Studio 4.2 Development Essentials - Kotlin Edition: Developing Android Apps Using Android Studio 4.2, Kotlin and Android Jetpack, Payload Media, Neil Smyth, 2021, 125p.
30. Українське автострахування URL: https://mtsbu.ua/ua/statistics/ (дата звернення: 30.11.2022)
31. Car License Plate Detection URL: https://www.kaggle.com/datasets/andrewmvd/car-plate-detection (дата звернення: 30.11.2022)
32. OCR URL: https://www.ibm.com/cloud/blog/optical-character-recognition (дата звернення: 30.11.2022)
33. Custom Models With ML Kit URL: https://developers.google.com/ml-kit/custom-models (дата звернення: 30.11.2022)
34. Номерні знаки України URL: https://uk.wikipedia.org/ Номерні_знаки_України (дата звернення: 30.11.2022)
35. Clean Architecture URL: http://prof.mau.ac.ir/images/Uploaded_files/Clean%20Architecture_%20A%20Craftsman%E2%80%99s%20Guide%20to%20Software%20Structure%20and%20Design-Pearson%20Education%20(2018)[7615523] (дата звернення: 31.05.2021)
36. Architecting Android.The clean way URL: https://github.com/android10/Android-CleanArchitecture (дата звернення: 31.05.2021)
37. Learn Android App Development: A step-by-step guide for non coders, Amazon.com Services LLC, Soyombo Soyinka, 124p
38. Exploring Android 2.0, CommonsWare, Mark L. Murphy, 523p
39. Programming DSLs in Kotlin: Design Expressive and Robust Special Purpose Code, Pragmatic Programmers, LLC, Venkat Subramaniam, 2020, 130p.
40. MVVM, MVP, MVC URL: https://pure.tue.nl/ws/files/48628529/Lou_2016 (дата звернення: 31.05.2021)