Національний університет «Одеська політехніка»
Інститут комп'ютерних систем
Кафедра інформаційних систем

КУРСОВА РОБОТА

з дисципліни “Програмування мобільних пристроїв”

Тема “Розробка додатку для продажу авто”

Студента __3__ курсу _AI -182__ групи
спеціальності 122 – «Комп’ютерні науки»
Суржикова Ю.А.
(прізвище та ініціали)
Керівник _____доцент.Годовіченко М.А.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
_____доцент.Смик С.Ю.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____
Кількість балів: _____
Оцінка: ECTS _____

Члени комісії _____ _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали

м. Одеса – 2021 рік

Одеський національний політехнічний університет
Інститут комп'ютерних систем
Кафедра інформаційних систем

ЗАВДАННЯ НА КУРСОВУ РОБОТУ

Суржиков Юрій Анатолійович АІ-182
(прізвище, ім’я, по батькові) (група)

1. Тема роботи: Розробка додатку для продажу авто

2. Термін здачі студентом закінченої роботи

3. Початкові дані до проекту(роботи): Автомобілі, які користувач хоче продати, та спостерігати статистку щодо продаж
тощо)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити) Вимоги до ПП, планування ПП, проектування ПП, програмна реалізація ПП, інструкція користувачеві,

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

Завдання видано

Завдання прийнято до виконання

АННОТАЦІЯ

У даній роботі представлено мобільний додаток на платформу Android, яке допомагає створювати і додавати оголошення на рахунок продажу автомобілів. На даний момент в ньому реалізовано додавання автомобіля, перегляд всіх доданих автомобілів, також в ньому є реєстрація і синхронізація на різних пристроях. Надалі планується розвивати додаток, щоб у ньому була можливість ділитися відгуками, доповнювати авто і отримувати статистику з продажу.

АННОТАЦИЯ

В данной работе представлено мобильное приложение на платформу Android, которое помогает создавать и добавлять объявления на счет продажи автомобилей. На данный момент в нем реализовано добавление автомобиля, просмотр всех добавленных автомобилей, также в нем есть регистрация и синхронизация на различных устройствах. В дальнейшем планируется развивать приложение, чтобы в нем была возможность делиться отзывами, дополнять авто и получать статистику по продажам.

ABSTRACT

This paper presents a mobile application for the Android platform, which helps to create and add ads to the car sales account. At the moment, it implements adding a car, viewing all added cars, it also has registration and synchronization on various devices. In the future, it is planned to develop the application so that it would be possible to share reviews, supplement cars and receive sales statistic

Перелік скорочень

ОС – операційна система

ІС – інформаційна система

БД – база даних

СКБД – система керування базами даних

ПЗ – програмне забезпечення

ПП– програмний продукт

UML – уніфікована мова моделювання

Room – бібліотека для роботи с базою даних

Activity – компонент андроид

ЗМІСТ

ВСТУП

Актуальність теми: Людині завжди характерно бажання досягти найбільшого комфортного існування, а в наш час його атрибутом вважається вихід в Інтернет, причому залишаючись постійно в онлайн. У зв'язку з цим, стала найбільш актуальною розробка різних додатків для мобільних пристроїв під IOS з розширенням можливостей мобільного Інтернету. Скажімо, сьогодні вже і в поїздці людям за наявності з'єднання з мережі з планшета, телефону або інших пристроїв, а спеціально створені додатки підвищують його ефективність і, навіть, вирішують завдання архітектурної 3D візуалізації.

Додатки для мобільних пристроїв призначені на робочі і розважальні цілі. Одні успішно допомагають бізнесменам і офісним працівникам контролювати бізнес і вести по ньому звіти, розробляти дизайн в оригінальному і фірмовому стилі. Інші забезпечують якісне прослуховування музики і перегляд фільмів, підтримують засоби спілкування і виконують ряд інших функцій. Хоча всі програми знаходять свого споживача, але фахівцями відзначається найбільш популярні, бажані компаніями, чия робота ведеться в різних напрямках. Розробка саме таких програм і їх реалізація веде до зростання прибутку, оскільки компанії наполегливо інвестують в напрацювання, що спрощують найскладніший процес ведення бізнесу.

Попит на подібні додатки для мобільних пристроїв стабільно зростає вже кілька останніх років. Напрошується висновок, що на сьогоднішній день актуальність розробки таких програм цілком доцільна і обов'язково отримає відповідне визнання користувачів.

Ідея продажу автомобілів через мобільний додаток є особливо актуальною в еру інтернету, коли практично будь-який підприємець хоче продати авто, та без зайвих зусиль знайти покупця.

Високу популярність серед продавців отримали сервіси, в яких є хмарне сховище данних, та коли можна користуватися декількома девайсами одразу, тобто мають синхронізацію. Як правило такі додатки платні, то ми хочемо зробити додаток, яких вмістить в собі всі потрібні якості

Об'єкт дослідження – продаж авто через мобільний додаток

Предмет дослідження – взаємодія користувача та додатку для продажу автомобілів

Метою роботи є створення додатку, та налаштування адмін консолі Firebase для нього, для створення заявок авто та продажу авто.

В зв’язку з поставленою метою потрібно розв’язати наступні задачі:

— Створити обліковий запис у Firebase, та налаштувати необхідні компоненти.

— Створити мобільний додаток, за допомогою якого можна буде здійснювати продаж авто.

1 ВИЗНАЧЕННЯ ВИМОГ ДО ПРОЕКТУ

1.1 Визначення потреби користувача

Для визначення потреб споживача нам необхідно подивитися, що користувач хоче побачити в додатку який фнукціонал повинен в ньому бути присутнім, щоб він залишився в цьому додатку і продовжував користуватися ним. В першу чергу варто відзначити, що користувачеві головне рабоать якісно додаток, в якому немає недоліків у вигляді «багів», і інших помилок.

Якщо говорити за функціонал додатка, то в ньому обов'язково має бути присутня можливість додавати дані, синхронізувати їх з різними пристроями, і управляти цими даними.

В даному випадку під даними розуміються записи про продажі і автомобілях.

1.2 Проблемний аналіз існуючих рішень

Для того, що оцінити степере того, що необхідно зробити в додатку необхідно подивитися які вже є альтернативи схожих рішень, які є у відкритому доступі.

Складемо певну таблицю для цього, в якій вкажемо продукти і функціонал, який вже доступний в конкретних додатках.

Таблица 1 – Сравнение продуктов

	Можливість створювати оголошення	Можливість переглядати оголошення інших	Можливість вести облік продажів	можливість реєстрації
rst.ua	+	-	-	+
auto.ria.com	+	+	-	+
Наше решение	+	+	+	+

1.3 Що потребується для роботи

Для роботи необхідні наступні бібліотеки і API

- Room версії 2.2.4 (для кешування даних)
- Google Maps API (для запиту інформації про місця, і побудові маршрутів)
- Retrofit версії 2.9.0 (для HTTP запитів на сервер ПП)
- Glide версії 3 (для кешування зображень)
- Gson (для сериализации / десеріалізациі об'єктів в / з JSON)
- Material версії 4.11.0 (для створення view компонентів в ПП)
- Maven версії 3.6.3 (для контролю структури проекту та автоматизації збирання (серверна частина))
- Gradle версії 6.1.1 (для контролю структури проекту та автоматизації збирання (програмна частина))
- Admin console Firebase(для налаштування бази даних)

2 ТЕОРЕТИЧНІ ВІДОМОСТИ

2.1 Firestore сховище

Firestore - це орієнтована на документи база даних NoSQL. На відміну від бази даних SQL, немає таблиць або рядків. Натомість ви зберігаєте дані в документах, які впорядковуються у колекції.

Кожен документ містить набір пар ключ-значення. Firestore оптимізований для зберігання великих колекцій невеликих документів.

Усі документи повинні зберігатися в колекціях. Документи можуть містити підколекції та вкладені об'єкти, обидва з яких можуть включати примітивні поля, такі як рядки, або складні об'єкти, такі як списки.

Колекції та документи створюються неявно у Firestore. Просто призначте дані документу в колекції. Якщо колекції або документа не існує, Firestore їх створює. У Firestore одиницею зберігання є документ. Документ - це полегшений запис, який містить поля, які відображають значення. Кожен документ позначається іменем.

Рисунок 1 – Приклад як зберігається запис

Документи живуть у колекціях, які є просто контейнерами для документів. Наприклад, у вас може бути колекція користувачів, яка містить різних користувачів, кожен із яких представлений документом:

Рисунок 2 – Зберігання колекцій

2.2 Бібліотека Glide

Glide це популярна бібліотека Android з відкритим вихідним кодом для завантаження зображень, відео та анімаційних GIF. З Glide можна завантажувати і відображати медіа з різних джерел, наприклад віддалених серверів або з локальної файлової системи.

За замовчуванням Glide використовує призначену для користувача реалізацію HttpURLConnection для завантаження зображень через Інтернет. Однак Glide також надає плагіни для інших популярних мережевих бібліотек, таких як Volley або OkHttp.

Розробка свого власного функціоналу для відображення і завантаження медіа на Java може виявитися справжнім головним болем: вам потрібно буде подбати про кешування, декодування, управлінні мережевих з'єднань, потоків, обробці виключень і про багато іншого. Glide - це проста у використанні, добре спланована, добре документована і ретельно протестована бібліотека, яка допоможе зберегти вам багато часу - і позбавити вас від головного болю.

У цьому уроці, ми будемо вивчати Glide 3, створюючи на ньому просту галерею зображень. Зображення будуть завантажуватися з інтернету і

відображатися слайдами в RecyclerView; користувач натискає на зображення і відкривається детальна активують із зображенням побільше.

2.3 Kotlin Coroutines

Корутіна - це паралельний шаблон дизайну, який ви можете використовувати на Android для спрощення коду, який виконується асинхронно. Спільні програми були додані до Kotlin у версії 1.3 і базуються на усталених концепціях з інших мов.

На Android спільні програми допомагають керувати тривалими завданнями, які в іншому випадку можуть заблокувати основний потік і спричинити відповідь програми. Понад 50% професійних розробників, які використовують спільні програми, повідомили про збільшення продуктивності. У цій темі описується, як ви можете використовувати програми Kotlin для вирішення цих проблем, дозволяючи писати більш чистий та стислий код програми.

Особливості:

— Coroutines - наше рекомендоване рішення для асинхронного програмування на Android. Варто відзначити такі особливості:

— Легкий: Ви можете запускати безліч програм на одній нитці завдяки підтримці підвіски, яка не блокує нитку, де працює програма. Призупинення економить пам'ять над блокуванням, підтримуючи багато одночасних операцій.

— Менше витоків пам'яті: використовуйте структуровану паралельність для запуску операцій у межах області.

— Вбудована підтримка скасування: Скасування розповсюджується автоматично через діючу ієрархію корутин.

— Інтеграція Jetpack: Багато бібліотек Jetpack включають розширення, що забезпечують повну підтримку супровідних програм. Деякі бібліотеки також надають власну спільну програму, яку ви можете використовувати для структурованої паралельності.

2.4 LiveData

LiveData - це спостережуваний клас власника даних. На відміну від звичайного спостережуваного, LiveData відповідає життєвому циклу, тобто означає, що він поважає життєвий цикл інших компонентів програми, таких як дії, фрагменти чи послуги. Ця обізнаність гарантує, що LiveData оновлює лише спостерігачі компонентів програми, які перебувають у активному стані життєвого циклу.

LiveData вважає спостерігача, який представлений класом Observer, у активному стані, якщо його життєвий цикл знаходиться в старті ЗАПУСКАНОГО або ВІДНОВЛЕНОГО. LiveData лише повідомляє активних спостерігачів про оновлення. Неактивні спостерігачі, зареєстровані для перегляду об’єктів LiveData, не повідомляються про зміни.

Ви можете зареєструвати спостерігача в парі з об’єктом, який реалізує інтерфейс LifecycleOwner. Це співвідношення дозволяє видалити спостерігача, коли стан відповідного об'єкта Життєвого циклу змінюється на ЗНИЩЕНО. Це особливо корисно для дій та фрагментів, оскільки вони можуть безпечно спостерігати за об’єктами LiveData і не турбуватися про витоки - дії та фрагменти миттєво скасовують підписку, коли їх життєві цикли руйнуються.

2.5 Методологія Scrum

Будь-яка розмова про успішне управлінні проектами за допомогою скрам варто починати з визначення скрам.

«Скрам - це фреймворк управління, згідно з яким одна чи кілька кроссфункціональних самоорганізованих команд створюють продукт Інкремент, тобто поетапно. У команді може бути близько семи чоловік ».

«У скрам є система ролей, зустрічей, правил і артефактів. У цій моделі за створення і адаптацію робочих процесів відповідають команди ».

«У скрам використовуються ітерації фіксованої довжини, звані спринті. Вони зазвичай займають 1-2 тижні (до 1 місяця). Скрам команди прагнуть

створювати готовий до постачання (якісно протестований) Інкремент продукту в кожній ітерації ».

Розберемо ці твердження, щоб краще зрозуміти методологію, фреймворки і процеси скрам.

НАВІЩО ПОТРІБЕН скрам?

У Скрам-фреймворка Аджайл-розробки чотири головні переваги:

Відгук на потреби клієнта. Компаніям-розробникам програм занадто добре знайоме вимога «розробити на вчора». Традиційні організації, які працюють за методом водоспаду, вбудовують важливі фичи і функції в розклад з двома релізами в рік - і в процесі нерідко втрачають клієнтів. Якщо ж клієнти не йдуть, вони все одно можуть залишитися незадоволеними і піти з часом, коли зустрінуть більш відповідального конкурента. Працюючи короткими і частими циклами, ви можете надавати клієнтам продукти майже на вимогу і швидко адаптуватися до нових вимог.

Зниження вартості розробки. Аджайл і Скрам довели свою економічність і ефективність. У цих моделях ролі розробників різноманітніші, адже невеликі одиниці можна ефективно тестувати тієї самої командою, яка розробила їх. Спеціалізація зникає або скорочується, в кінцевому рахунку скорочуючи витрати.

Задоволення від роботи. Поставляючи продукти швидко, команда переживає додаткову радість кожен раз, коли робота зроблена і відправляється в світ. Кожна команда розробки знає, як приємний реліз. З скрам команда радіє йому не два, а 12 раз на рік, мінімум.

Більше швидких доходів. Засоби від клієнтів надходять не двічі на рік, а набагато частіше. Крім того, нові фічі теж можна додавати не двічі на рік, а набагато частіше, таким чином приводячи більше клієнтів і швидше обробляючи їх особливі запити.

3 ПРОЕКТУВАННЯ

3.1 Історії користувача до ПП

Тепер перед тим, як починати проектувати, необхідно визначити, що повинно бути присутнім в проекті. Для цього напишемо сценарії використання.

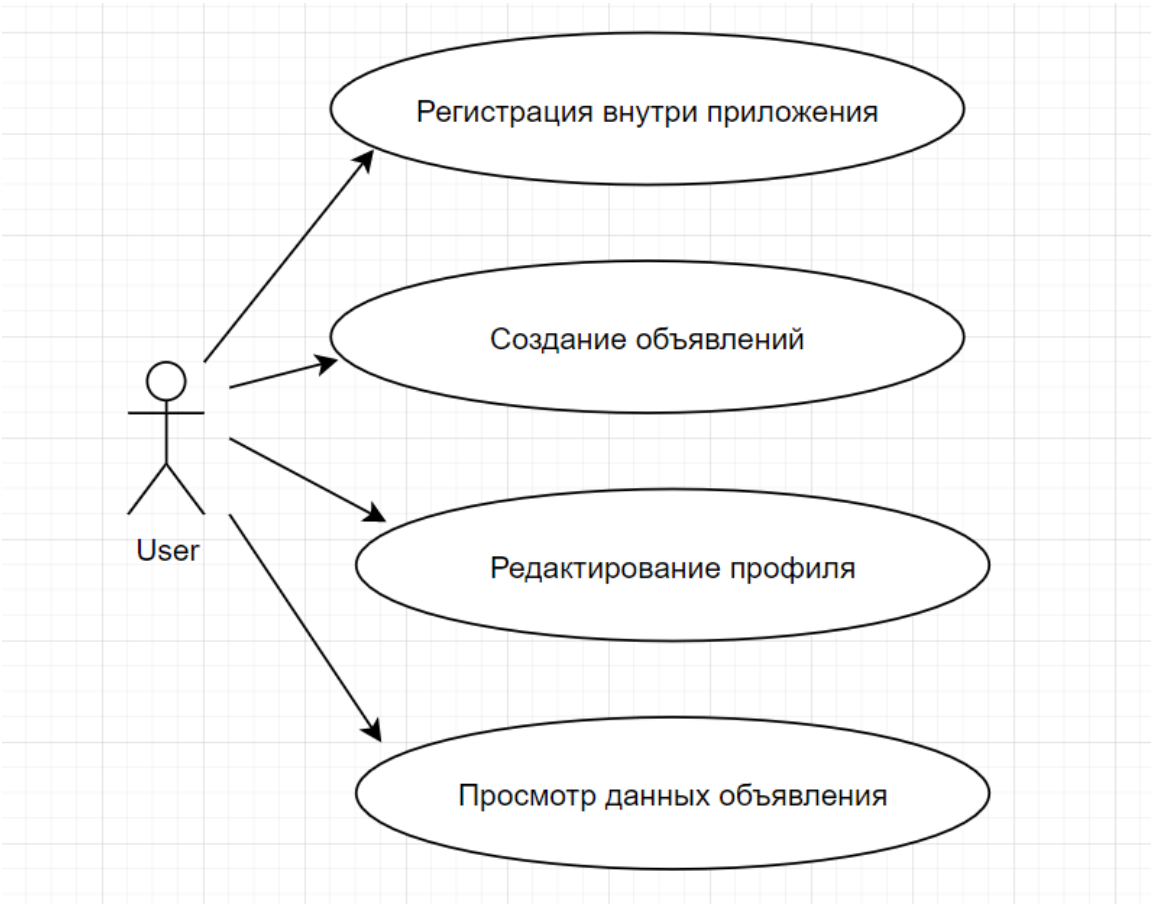


Рисунок 3 - Сценарії використання

3.2 Концептуальний опис архітектури ПП

В якості концептуального опис програмного продукту, буде використовуватися діаграма розгортання, в якій буде чітко показано, які основні компоненти будуть використовуватися для успішного розгортання ПП.

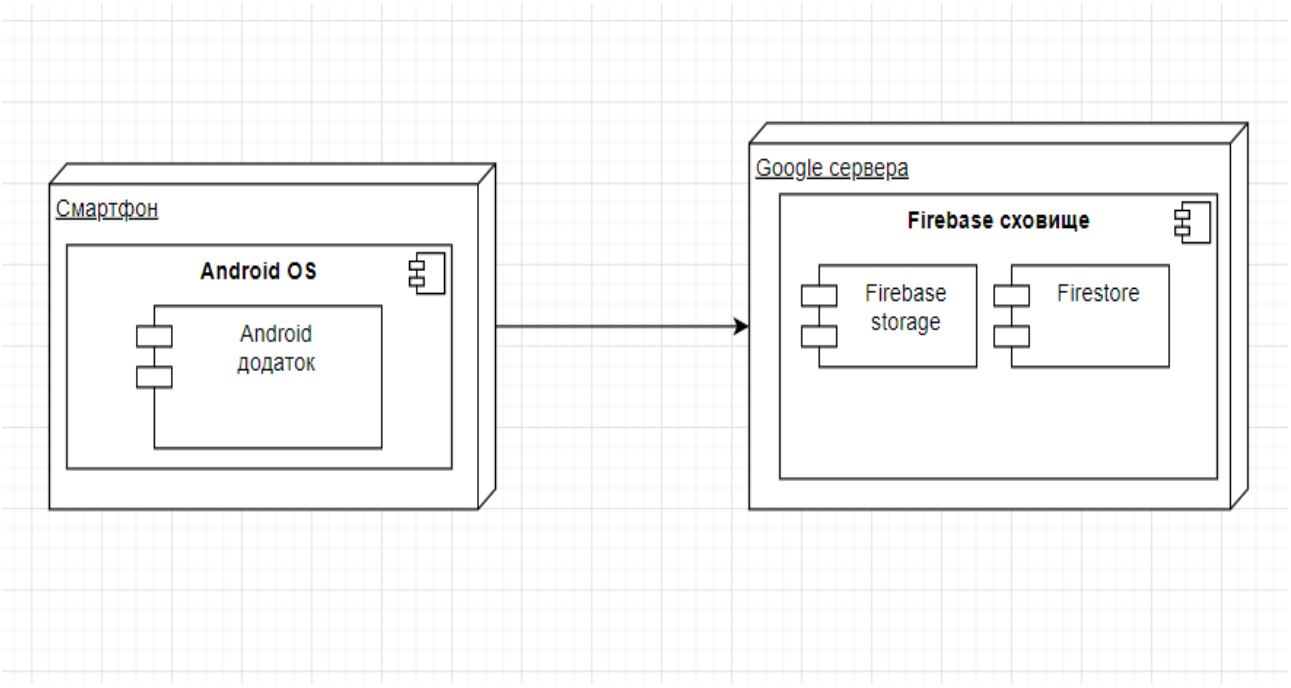


Рисунок 4 - Діаграма розгортання

Уніфікована мова моделювання (UML) був розроблений з метою забезпечити єдиний візуальну мову з багатою семантикою і розгорнутим синтаксисом для проектування і впровадження програмних систем зі складною структурою і комплексним поведінкою. Варто відзначити, що UML застосовується не тільки в розробці програм, але і в інших сферах, наприклад, в схематизації потоків виробничих процесів.

Мова UML отримав широке поширення в середовищі програмістів, однак в цілому мало використовується в розробці баз даних. Одна з причин цього полягає в тому, що творці UML не ставили бази даних в центр уваги. Проте, UML ефективно застосовується в концептуальному моделюванні даних високих рівнів і підходить для створення різних видів діаграм UML.

Знаючи це тепер створимо концептуальну схему на основі uml-діаграми.

Логічне проектування бази даних - це процес створення моделі використовуваної на підприємстві інформації на основі обраної моделі організації даних, але без урахування типу цільової СУБД і інших фізичних аспектів реалізації. Другий етап проектування бази даних називається логічним проектуванням бази даних.

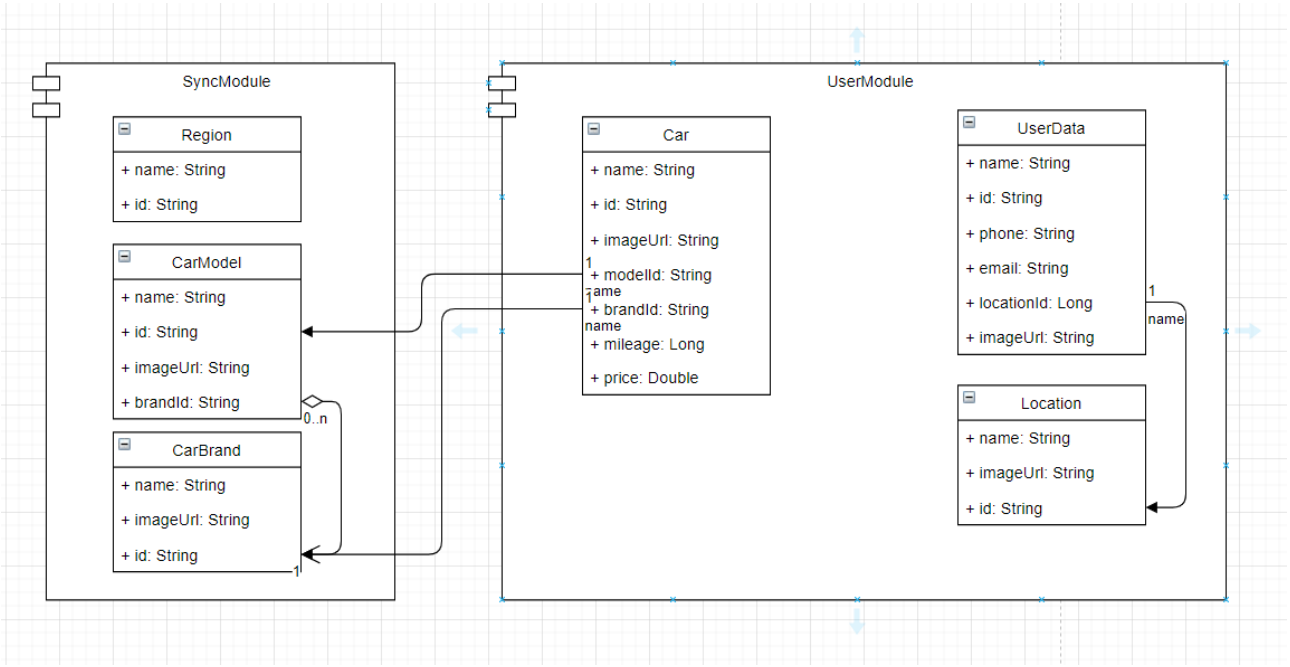


Рисунок 3 - Концептуальная схема данных

Логіки з боку БД в нашому програмному продукті не буде в принципі. Вся логіка буде міститися в програмних класах, які називаються сервісами.

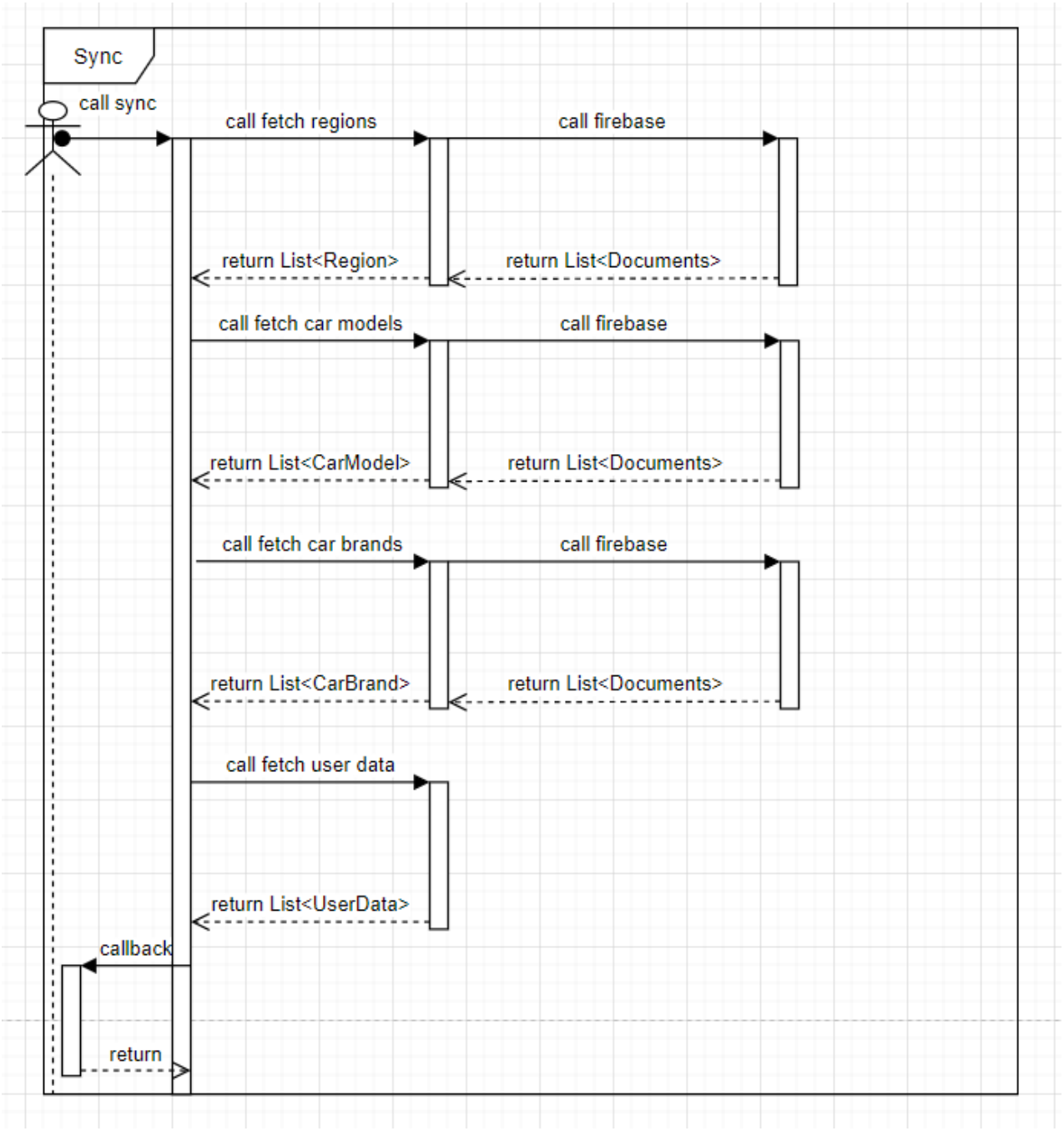


Рисунок 4 – Діаграмма активності

4 РЕАЛІЗАЦІЯ

4.1 Особливості створення мобільного додатку

Структура проекту мобільного додатка буде виглядати наступним чином.

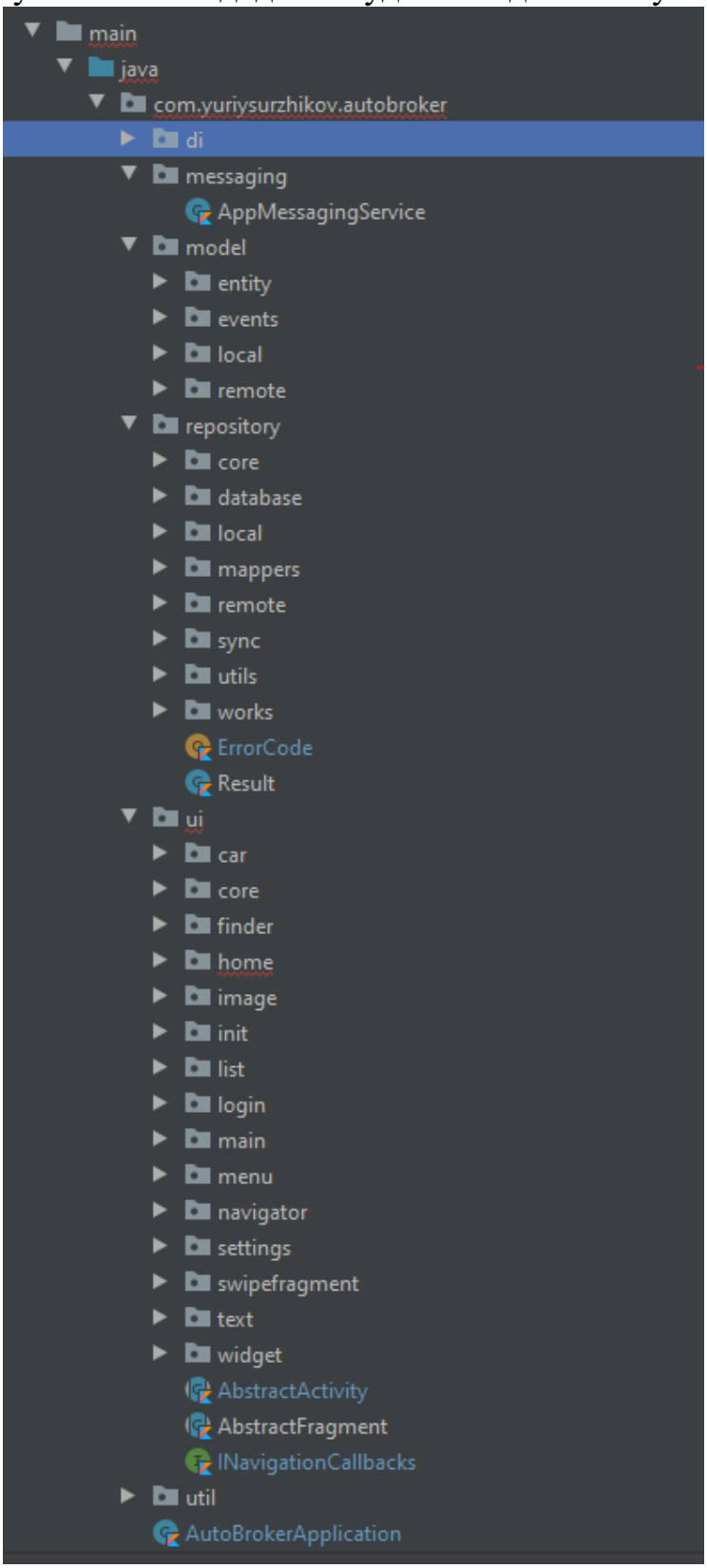


Рисунок 6 – Структура проекту

У створенні мобільного застосування набагато більше класів, які необхідно створити, для того, щоб і структуру не порушувати, і для того, щоб додаток працювало.

В Android є кілька елементів проекту, які відповідають за візуальну частину, і логіку роботи візуальної частини. Для старту програми обов'язково повинен бути хоча б один клас активності. У нашому проекті буде використовуватися один клас активності, а всі інші екрани будуть відкриватися через фрагменти (теж складові одиниці вікон програми).

```
@AndroidEntryPoint
class MainActivity :
    AppCompatActivity(),
    INavigation,
    NavigationListener,
    IRouteOpener {

    private lateinit var binding: ActivityMainBinding
    private lateinit var navigationAdapter: AbstractNavigationAdapter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = DataBindingUtil.setContentView( activity: this, R.layout.activity_main)
        if(CommonUtils.isLocationPermissionGranted( context: this)) {
            launchMainApplication()
        } else {
            binding.navigation.visibility = View.GONE
            navigationAdapter = NavigationPagerAdapter(supportFragmentManager)
            navigationAdapter.addScreen(PermissionsRequestFragment(), TAG: "permissions_provider")
            binding.mainContainer.adapter = navigationAdapter
        }
    }
```

Рисунок 7 – Создание класса активности

Клас обов'язково повинен успадковуватися від класу AppComaptActivity, і реалізовувати метод onCreate (), в якому йде створення елементів view, як раз через DataBinding. Оскільки все в'ю елементи в Android прописуються через xml-файл, то діставати всі ці елементи звідти було дуже проблемно, і тому був придуманий механізм DataBinding, який дозволяє поміщати дані в xml файлі, без необхідності постійного пошуку в'ю елементів. Відбувається це автоматичною генерацією Java класів.

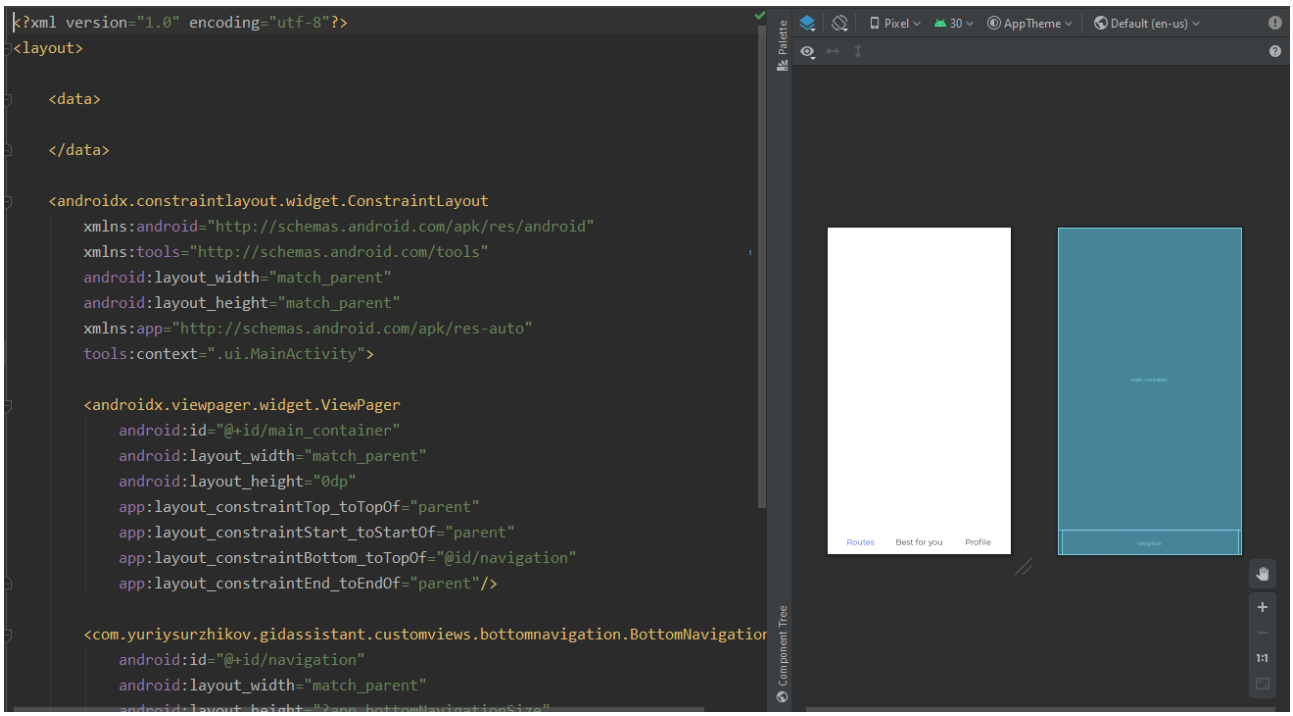


Рисунок 8 – Создание layout с DataBinding

Для правильної архітектури сам Google рекомендує використовувати наступну структуру

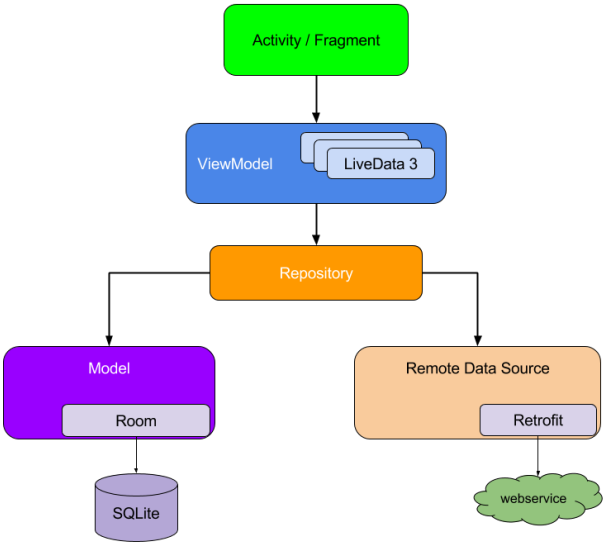


Рисунок 9 – Пример правильной архитектуры приложения от Google

Нижче представлена структура сховища проекту. У ньому є пакет для кожної сутності, для того щоб забезпечити поділ відповідальності, і забезпечити більш гнучку структуру проекту.

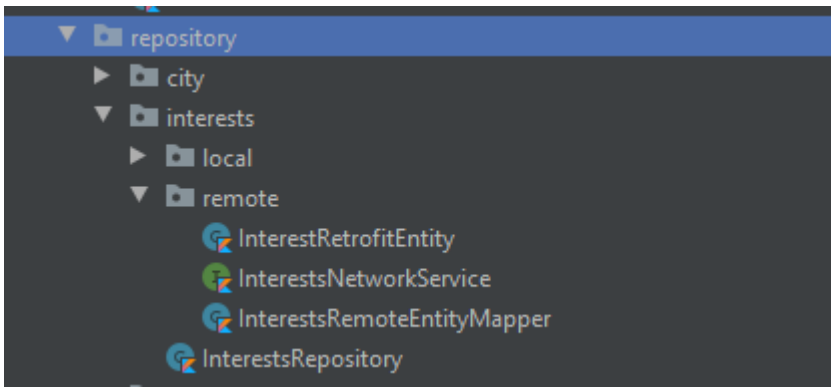


Рисунок 10 – Структура репозитория

Оскільки у нас йде поділ сутностей на ті, які зберігаються в кеші, на ті, які виходять з сервера і на ті, які використовуються для відображення у view, то в пакеті буде йти поділ на remote і local роботу з сутностями.

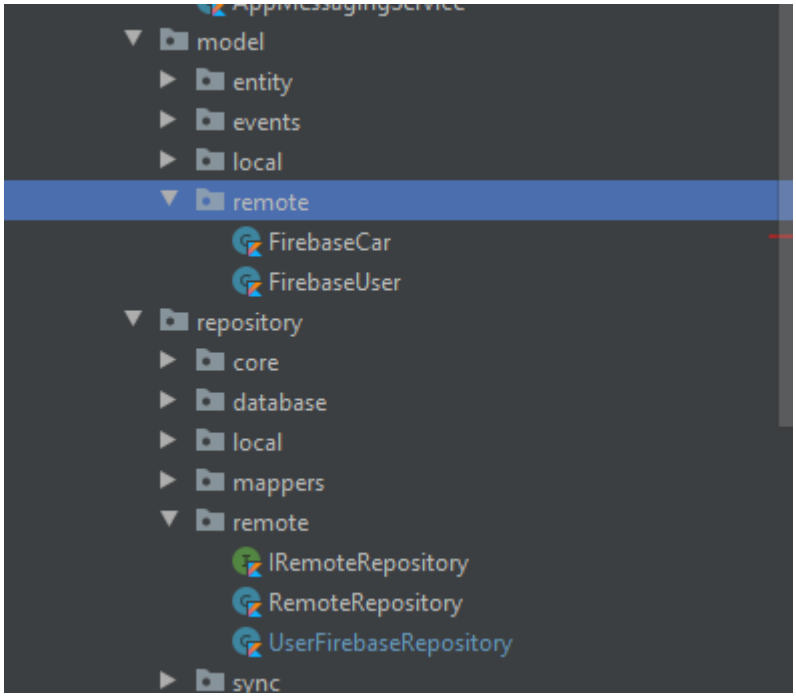


Рисунок 10 – Схема репозитрія

І є також класи для роботи з локальними сутностями. Для того, щоб інтерфейс став репозиторієм (за аналогією з проектом на Java Spring), необхідно цей інтерфейс помітити анотацією @Dao. І кожен метод помітити анотацією або @Insert, @Delete, @Update або @Query. Для тих методів, в яких може бути згенеровано кілька запитів в рамках одного, такі методи необхідно позначити додатково анотацією @Transaction.

```
interface ICrudRepository<T> {
    @Delete
    suspend fun delete(item: T)

    @Update
    suspend fun update(item: T)

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun addAll(items: List<T>)

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun add(item: T)
}
```

Рисунок 11 – Пример создания класса для работы с локальной базой данных

Також в проекті для Android програми буде використовуватися бібліотека для впровадження залежностей Hilt.

4.2 Особливості створення програмних класів

Як уже писалося вище, класи на яких йде відображення якихось view елементів повинні успадковуватися від AppCompatActivity або від Fragment. У нашому проекті буде використовуватися архітектура, побудована на фрагментах. Тобто буде базова активність, і далі будуть створитися фрагменти для відображення будь-яких даних.

```kotlin
@AndroidEntryPoint
abstract class AbstractFragment : Fragment() {

    override fun onStart() {
        super.onStart()
        EventBus.getDefault().register( subscriber: this)
    }

    override fun onStop() {
        super.onStop()
        EventBus.getDefault().unregister( subscriber: this)
    }

    @Subscribe(threadMode = ThreadMode.MAIN)
    open fun onSyncComplete(event: SyncSuccessEvent) {
    }
}
```

Рисунок 12 – Приклад абстрактного класу

Кожен клас зробимо refreshable, для того, щоб можна було оновлювати дані, на кожному з фрагментів, якщо з'явитися така необхідність.

4.3 Особливості розробки алгоритмів методів програмних класів або процедур/функцій

Особливості розробки алгоритмів швидше пов'язані з використовуваними мовами програмування. На Java буде використовуватися StreamAPI для забезпечення максимальної продуктивності додатка (в серверній частині). Мовою Kotlin це по суті той же StreamAPI, але він не буде запускатися в окремому потоці, все буде обчислюватися в головному потоці. Оскільки в серверній частині ми можемо не турбуватися про навантаженості сервера, тому що Java Spring вже використовує багато поточностью в своєму ядрі, і зазначена спочатку навантаження в 300 користувачів одночасно не змусить сервер навантажувати.

Однак на Android не все так гладко з потоками, там немає спеціального механізму обробки даних, який сам би розумів, коли запускати один потік, а коли інший. Для того, щоб забезпечити максимальну продуктивність додатка, ми буде використовувати корутіни від Kotlin.

Використання цих технологій досить просте, якщо не говорити про щось складному, наприклад, потоків (flow) даних.

Для того, щоб запустити нову Кортіна в Kotlin досить викликати наступний метод.

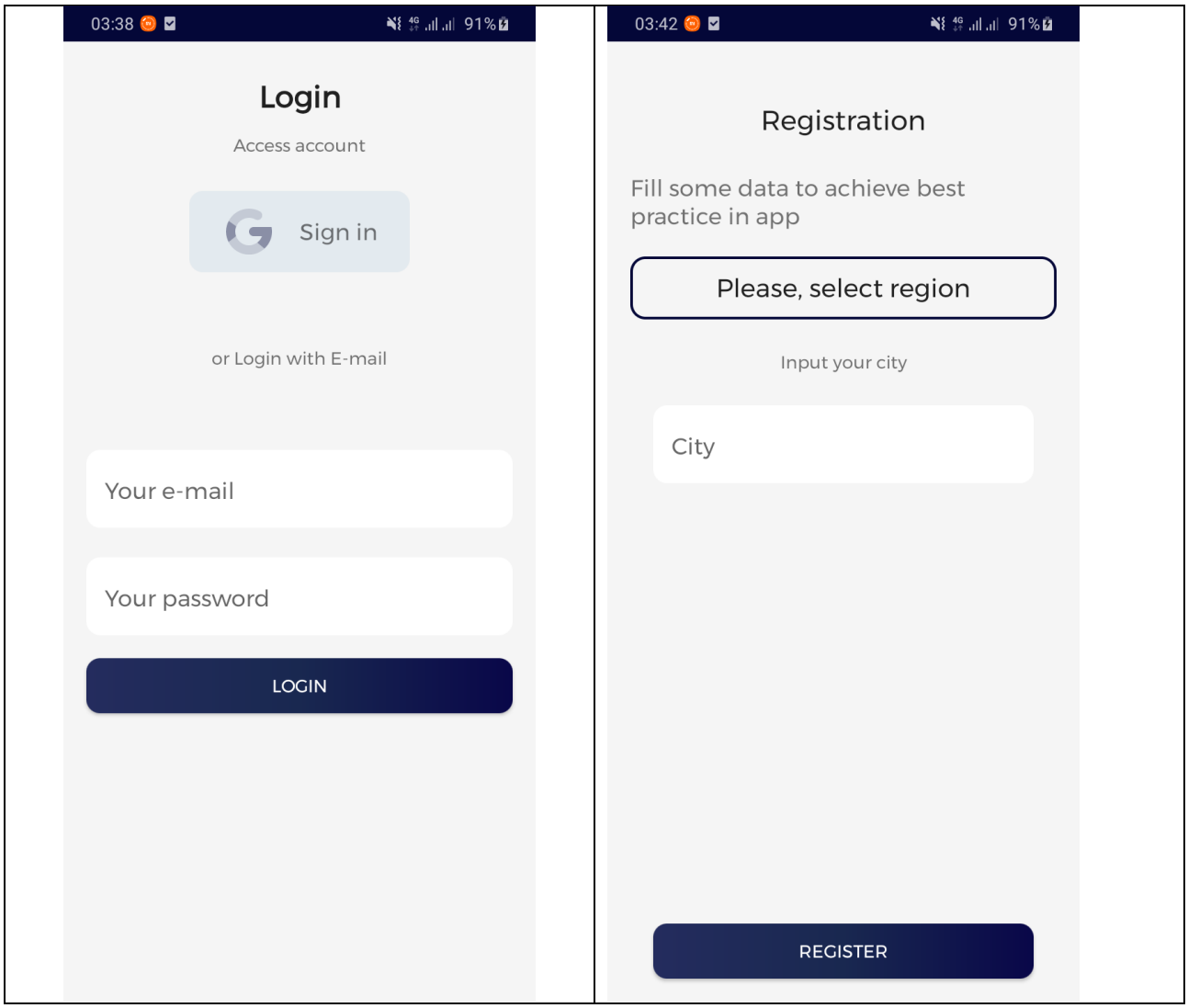
```
private fun load() {
    if (isLoading.get()) {
        return
    }
    isLoading.set(true)
    CoroutineScope(Dispatchers.IO).launch { this: CoroutineScope
        val userCars = userDatabase.getUserCarRepository().getAll().map { it: CarRoom
            carMapper.mapToEntity(it)
        }
        mutableList.postValue(userCars)
        isEmpty.set(userCars.isEmpty())
        isLoading.set(false)
    }
}
```

Рисунок 13 – Пример создания новой корутины

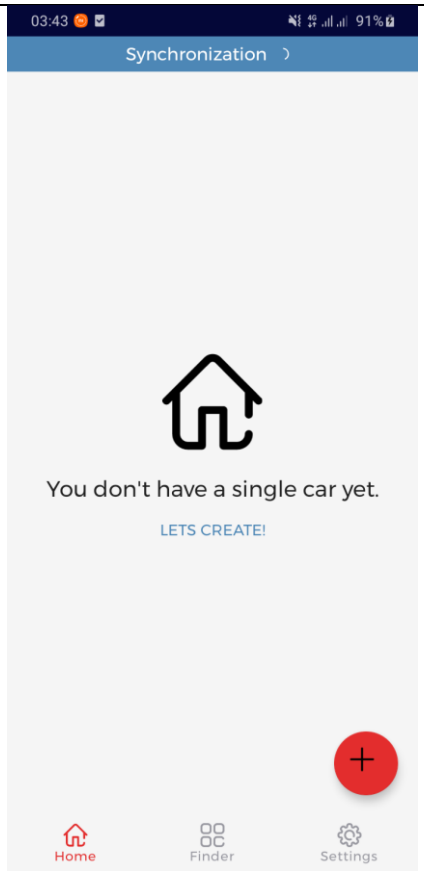
Корутіни можна запускати в декількох стандартних потоках: це Main потік, в якому отрісовиваємих view елементи, IO потік, в якому відбувається обробка I / O даних. I можна запускати в своєму кастомними потоці, вказавши його ID.

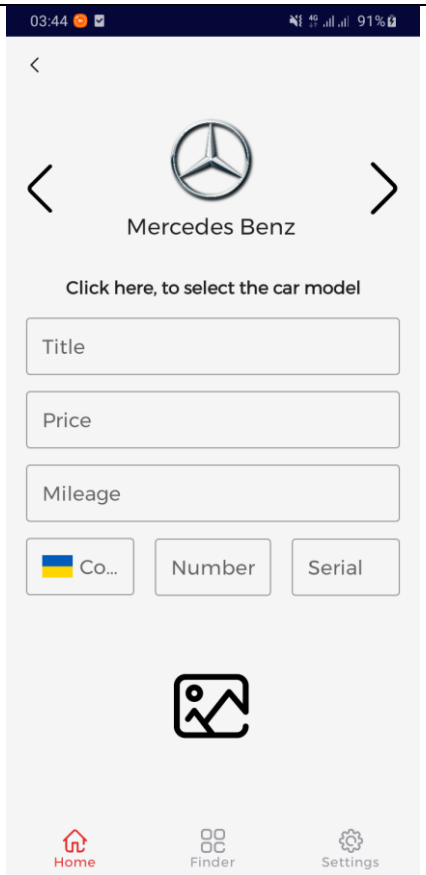
5 ІНСТРУКЦІЯ КОРИСТУВАЧЕВІ

Перш за все, потрібно відзначити, що додаток працью тільки, якщо є підключення до інтернету. Одразу після того, як додаток запуститься, відкриється екран логіну. В якому ми можемо або ввести логін, або увійти за допомогою Google. Якщо профіль ще не створено, відкриється вікно заповнення анкети.

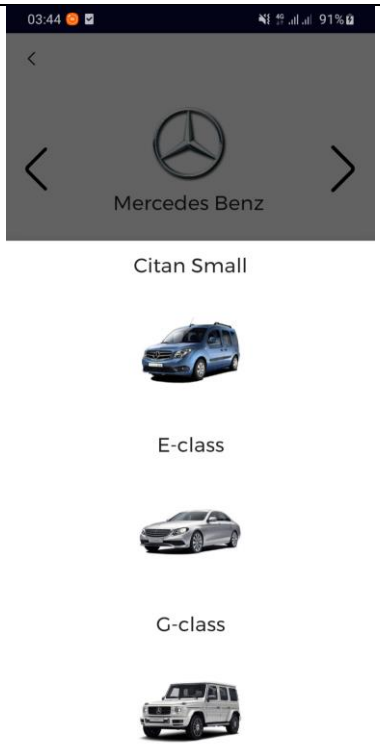


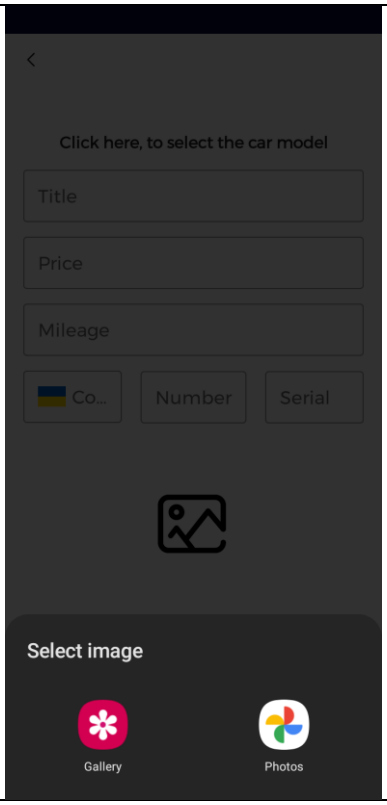
Після введення даних, вас буде перекинуто на головну сторінку На головному екрані буде така іконка, бо у вас ще немає записів.

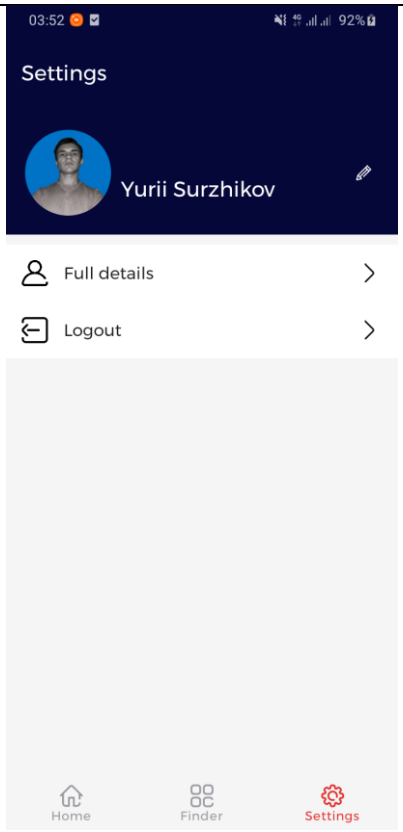


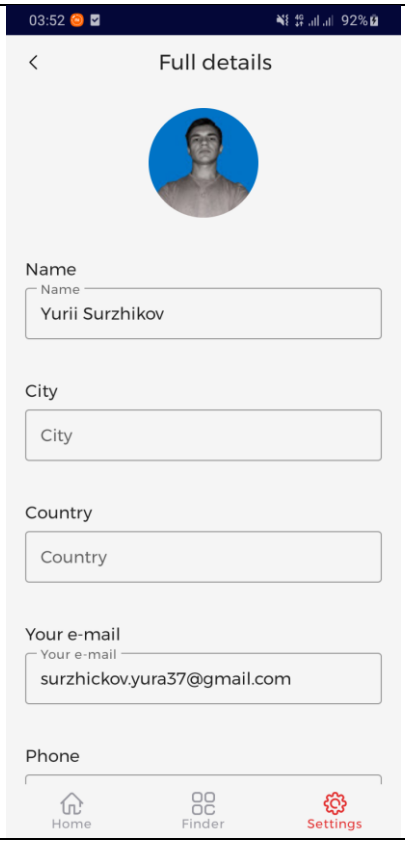


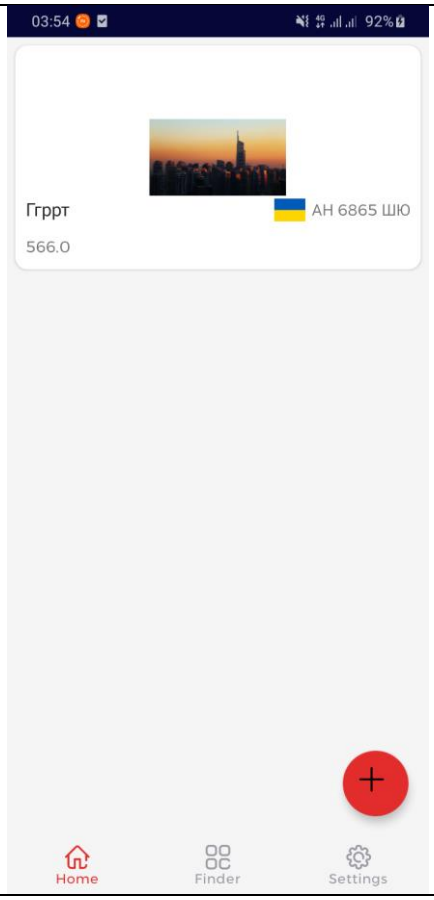
Тепер якщо натисуто буде на «Click here to select car model» - відкриєьбся окно вибору моделі авто.






03:52
92%
Settings
Yurii Surzhikov
Full details
Logout
Home
Finder
Settings


03:52
92%
Full details
Name
Name
Yurii Surzhikov
City
City
Country
Country
Your e-mail
Your e-mail
surzhickov.yura37@gmail.com
Phone
Home
Finder
Settings


03:54
92%
Ггррт
АН 6865 ШЮ
566.0
Home
Finder
Settings

ВИСНОВКИ

В ході виконання роботи був розроблений програмний продукт, в якому є можливість додати новий запис про авто, то поділитися їм.

Відразу варто відзначити, що не всі цілі були досягнуті, деякі з функцій не були реалізовані, через процеси, на які розробник вплинути ніяк не мог. Тут грав роль фактор того, що всі учасники команди працюють, і час, який міг би приділятися розробці програмного продукту просто приділялась виконанню робочих обов'язків в тій компанії, де працює кожен з учасників.

Також, однією з проблем в реалізації проекту, стало те, що з деякими технологіями деякі учасники команди не були знайомі, і реалізацію цих функцій робили інший учасник, у якого були і свої завдання.

У підсумковій версії продукту не було реалізовано глобального пошуку по мережі всіх автомобілів, також є проблема з навігацією по екрам, і повна робота з особистим кабінетом, для, і прогулянках.

Зазначені недоробки планується доробити в наступних курсових роботах, з учетеом тм наступних семестрі

СПИСОК ДЖЕРЕЛ

1. Android Architecture Components – Дмитрий Виноградов // StartAndroid.ru [2017] – https://startandroid.ru/ru/courses/architecture-components.html

2. Изучаем Android – Александ Климов // developer.alexanderklimov.ru [2017] – http://developer.alexanderklimov.ru/android/

3. Android documentation – Google Inc. [2021] – https://developer.android.com/docs

4. Introduction to Activities – Google Inc. [2021] – https://developer.android.com/guide/components/activities/intro-activities

5. Connectivity - https://developer.android.com/guide/topics/connectivity

6. LiveData Overview. How to use in your app – Google Inc. [2021] – https://developer.android.com/topic/libraries/architecture/livedata

7. ViewModel Overview. How to implement in your app – Google Inc. [2021] – https://developer.android.com/topic/libraries/architecture/viewmodel

8. DataBinding: Overview; Get Started – Google Inc [2021] – https://developer.android.com/topic/libraries/data-binding/start