



МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«МИРЭА – Российский технологический университет»**  
**РТУ МИРЭА**

---

**Институт информационных технологий (ИИТ)**  
**Кафедра математического обеспечения и стандартизации информационных**  
**технологий (МОСИТ)**

### **КУРСОВАЯ РАБОТА**

по дисциплине «Проектирование и разработка мобильных приложений»

**Тема курсовой работы:** «Информационно-справочная система  
“Автомобили автосалона”»

Студент группы ИКБО-61-23 Соколов Алексей Андреевич

(подпись)

Руководитель  
курсовой работы

Доцент Синицын И.В.

(подпись)

Работа представлена к защите «9» 06 2025 г.

«9» 06 2025 г.

Допущен к защите

Москва 2025 г.



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

---

**Институт информационных технологий (ИИТ)**

**Кафедра математического обеспечения и стандартизации информационных  
технологий (МОСИТ)**

Утверждаю

И.О. Заведующего кафедрой МОСИТ

Член — Головин С.А.

«17» февраля 2025 г.

**ЗАДАНИЕ  
на выполнение курсовой работы**

по дисциплине «Проектирование и разработка мобильных приложений»

Студент Соколов Алексей Андреевич

Группа ИКБО-61-23

**Тема «Информационно-справочная система “Автомобили автосалона”»**

**Исходные данные:** Разрабатываемый прототип мобильного приложения должен предоставлять возможности полностью интерактивной системы с понятным дружественным UI и обеспечивать необходимую функциональность, которая формируется в зависимости от заданной темы и предметной области изучаемых вопросов.

**Перечень вопросов, подлежащих разработке, и обязательного графического материала:**

Установка и настройка мобильной ОС с применением виртуальных сред. Установка и настройка IDE для мобильной разработки. Установка и настройка эмуляторов мобильного девайса.

Анализ предметной области разрабатываемой программной системы. сбор и анализ требований. составление ТЗ согласно ГОСТ 19.201-78. Реализация ролевой модели безопасности.

Изучение жизненного цикла мобильных программ и их компонентов, а также создание мобильного программного комплекса с применением языков программирования высокого уровня согласно теме курсовой работы. Реализация в создаваемом программном комплексе визуальных элементов, сервисов, методов хранения данных.

Тестирование и отладка кода созданного программного продукта, контрольные примеры работы.

Возможно портирование написанной программной системы на внешних хостах в сети Интернет.

Отчет по курсовой работе в виде пояснительной записки.

**Срок представления к защите курсовой работы:**

*Иван Синицын*  
до «30» мая 2025 г.

Синицын И.В.

(ФИО руководителя)

«17» февраля 2025 г.

Соколов А.А.

(ФИО обучающегося)

«17» февраля 2025 г.

**Задание на курсовую работу выдал**

*Алексей Соколов*  
Подпись руководителя

**Задание на курсовую работу получил**

*Алексей Соколов*  
Подпись обучающегося

# **СОДЕРЖАНИЕ**

<b>СОДЕРЖАНИЕ.....</b>	<b>1</b>
<b>ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ.....</b>	<b>2</b>
<b>ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ.....</b>	<b>3</b>
<b>ВВЕДЕНИЕ.....</b>	<b>4</b>
<b>1 ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ.....</b>	<b>6</b>
1.1   Описание предметной области.....	6
1.2   Анализ существующих аналогов .....	7
1.2.1 Auto.ru.....	7
1.2.2 Fresh Auto .....	8
1.2.3 Major Auto .....	10
1.3   Техническое задание .....	12
1.3.1 Определение пользовательских требований .....	12
1.3.2 Определение функциональных требований .....	13
<b>2 ПРОЕКТНАЯ ЧАСТЬ .....</b>	<b>15</b>
2.1   Описание процесса в нотации IDEF0 .....	15
2.2   Описание процесса в нотации DFD .....	23
2.3   Описание и обоснование выбора программного обеспечения.....	25
2.4   Архитектура программной системы .....	25
<b>3 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ .....</b>	<b>28</b>
3.1   Описание моделей и структур данных.....	28
3.2   Тестирование программного продукта .....	35
3.3   Полный листинг .....	41
<b>ЗАКЛЮЧЕНИЕ .....</b>	<b>72</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....</b>	<b>73</b>

## **ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ**

В настоящем отчете применяют следующие термины с соответствующими определениями.

Авторизация — процесс предоставления пользователю или группе пользователей определенных разрешений, прав доступа и привилегий в компьютерной системе

База данных (БД) — совокупность хранимых в памяти компьютера данных, относящихся к определенному объему или кругу деятельности, специально организованных, обновляемых и логически связанных между собой

Приложение — прикладная система или программа, предназначенная для решения задач в конкретной области техники

## **ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ**

В настоящем отчете применяют следующие сокращения и обозначения.

API – Application Programming Interface

DFD (Data flow diagrams) — диаграмма потоков данных

IDEF (Integrated Definition) — комплексное описание

SDK – Software Development Kit

UI – User Interface

XML – eXtensible Markup Language

# ВВЕДЕНИЕ

Целью данной курсовой работы является разработка прототипа мобильного приложения для платформы Android, предназначенного для взаимодействия пользователей с каталогом автомобилей автосалона. Приложение предоставляет удобный интерфейс для просмотра автомобилей, фильтрации по параметрам, добавления в избранное, а также отображения подробной информации и контактных данных офиса.

Актуальность темы обусловлена растущей популярностью цифровых решений в автомобильной индустрии. Большинство автосалонов стремятся к автоматизации процессов взаимодействия с клиентами, включая поиск автомобилей, уточнение технических характеристик и проведение онлайн-консультаций. Пользователям важно получать доступ к информации быстро и удобно, а также иметь возможность работать с интерфейсом в любое время, без необходимости обращения к менеджеру лично.

В рамках разработки приложения реализуются следующие основные задачи: создание структурированного каталога автомобилей, реализация пользовательской авторизации и профиля, управление избранными автомобилями, отображение текущего состояния купленных авто, просмотр похожих моделей, а также предоставление информации об офисе автосалона. Все данные хранятся в локальной базе данных Room, а основная информация о автомобилях загружается из Firebase Realtime Database.

Объектом исследования выступает информационная система мобильного автосалона. Предметом исследования являются методы проектирования клиентской части мобильных приложений, а также взаимодействие с локальными и облачными базами данных.

В исследовательской части курсовой работы рассматриваются существующие решения и подходы, выполняется анализ предметной области и разрабатываются диаграммы в нотациях IDEF0 и DFD, описывающие основные процессы взаимодействия между пользователями, хранилищами данных и

интерфейсом. В проектной части определяются функциональные требования к системе, выбираются архитектурные подходы и стек технологий. В технологической части приводится реализация ключевых компонентов приложения, включая интерфейсы и логическую обработку данных, а также проводится тестирование основных функций.

# **1 ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ**

## **1.1 Описание предметной области**

В предметную область данной курсовой работы входят особенности цифрового представления ассортимента автомобилей, реализуемых в автосалонах [1], а также механизмы взаимодействия пользователей с информацией об автомобилях через мобильное приложение. В рамках данной области рассматриваются процессы отображения, фильтрации, анализа и хранения данных о транспортных средствах, а также возможности персонализации пользовательского опыта.

Предметная область включает:

- каталог автомобилей, содержащий технические характеристики, изображения и другую справочную информацию;
- пользовательские действия по выбору и сравнению автомобилей;
- фильтрацию моделей по заданным параметрам (марка, тип кузова, коробка передач, привод и др.);
- управление списком избранного;
- отображение контактной информации и данных о местоположении автосалона;
- авторизацию и ведение пользовательской сессии;
- хранение информации о купленных автомобилях и их текущем состоянии.

Все вышеуказанные функции охватываются разрабатываемой системой и реализуются с использованием соответствующих программных и информационных компонентов, таких как база данных, визуальные шаблоны интерфейса и API-хранилища.

## **1.2 Анализ существующих аналогов**

При разработке информационно-справочной системы «Автомобили автосалона» важно учитывать существующие решения на рынке [1], чтобы определить их сильные и слабые стороны и на этой основе сформировать конкурентоспособный продукт. Ниже приведен анализ нескольких популярных мобильных приложений, предоставляющих информацию об автомобилях и автосалонах.

### **1.2.1 Auto.ru**

Описание:

Auto.ru — одно из крупнейших российских приложений для покупки и продажи автомобилей, принадлежащее компании «Яндекс». Приложение предоставляет пользователям возможность искать автомобили по различным параметрам, просматривать фотографии, читать отзывы и связываться с продавцами.

Достоинства:

- обширная база данных автомобилей;
- удобный поиск и фильтрация по множеству параметров;
- возможность просмотра истории автомобиля по VIN-коду;
- интеграция с другими сервисами «Яндекса».

Недостатки:

- перегруженность интерфейса из-за большого количества функций.
- наличие рекламы, что может отвлекать пользователя.
- некоторые функции доступны только при регистрации.

Далее покажем интерфейс основных страниц приложения.

На Рисунке 1 представлен интерфейс странички с автомобилем.

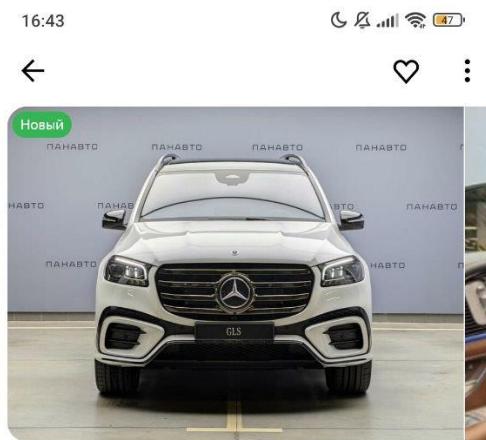


Рисунок 1 – Интерфейс странички с автомобилем

## 1.2.2 Fresh Auto

### Описание:

Fresh Auto — автомобильный маркетплейс, предлагающий пользователям возможность покупки и продажи автомобилей, а также получения одобрения по кредиту онлайн.

### Достоинства:

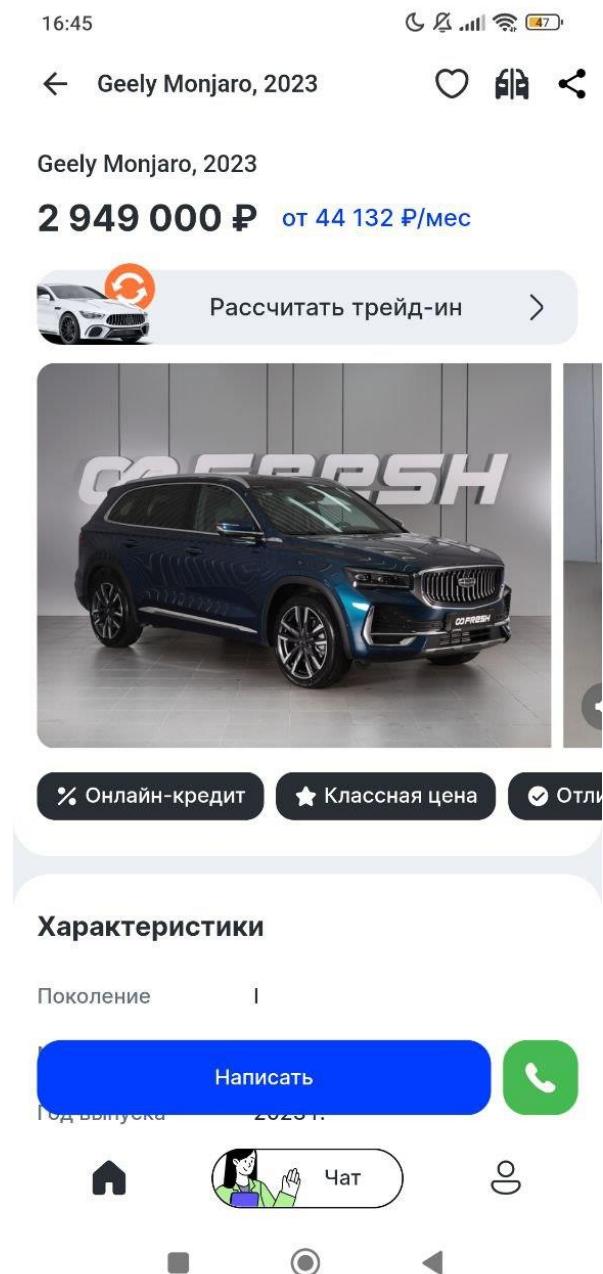
- современный и интуитивно понятный интерфейс;
- возможность онлайн-оформления кредита;
- широкий выбор автомобилей с пробегом и новых моделей.

### Недостатки:

- ограниченная география действия (в основном крупные города России);
- меньшая база данных по сравнению с конкурентами.

Покажем далее на рисунках интерфейс основных страниц приложения.

На Рисунке 2 показан интерфейс странички с автомобилем, в котором содержатся данные об автомобиле, фото автомобиля со всех сторон и прочая полезная информация.



**Рисунок 2 – Интерфейс странички с автомобилем**

### **1.2.3 Major Auto**

Описание:

Приложение от одного из крупнейших автодилеров России — Major Auto. Предоставляет информацию о наличии автомобилей, ценах и акциях.

Достоинства:

- актуальная информация о наличии автомобилей в автосалонах;
- уведомления о специальных предложениях и акциях;
- возможность записи на тест-драйв или сервисное обслуживание.

Недостатки:

- ограниченная функциональность по сравнению с маркетплейсами;
- интерфейс может показаться устаревшим;
- сильно ограниченный функционал без регистрации.

Покажем на рисунках интерфейсы главных страниц приложения.

На Рисунке 3 показан интерфейс страницы приложения с автомобилем.

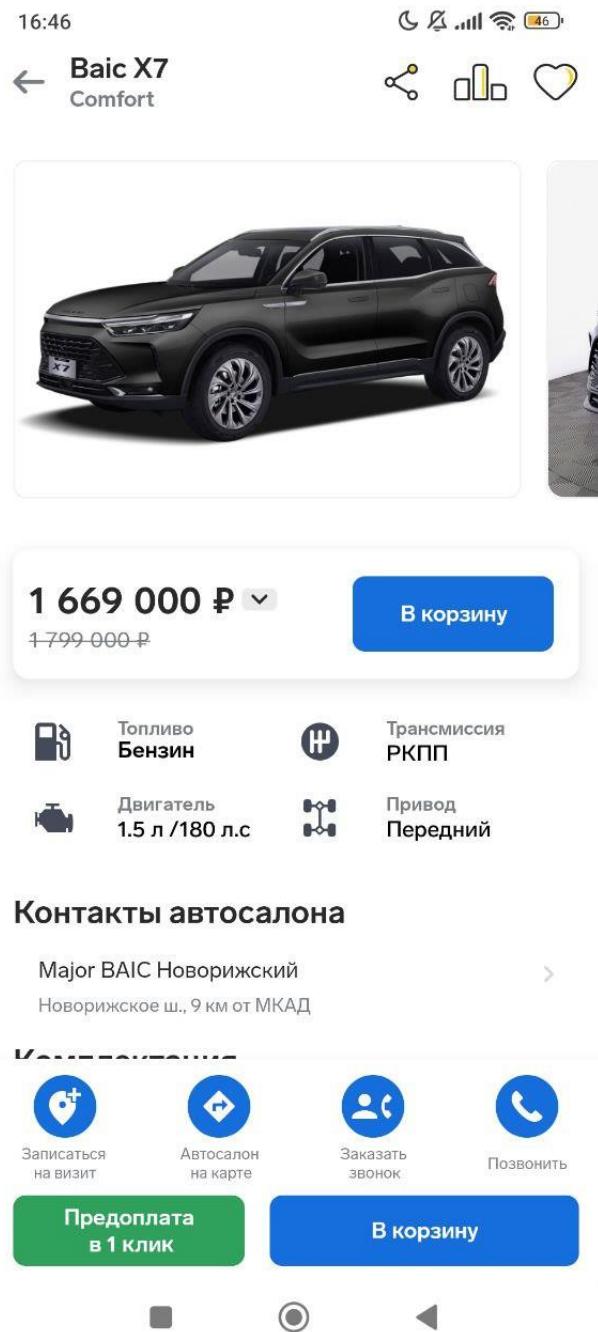


Рисунок 3 – Интерфейс странички с автомобилем

Соединим полученные данные в таблицу сравнения (таблица 1).

*Таблица 1 – Таблица сравнения существующих аналогов*

Приложение	Основные функции	Достоинства	Недостатки
Auto.ru	Покупка и продажа авто, история по VIN номеру	Обширная база, интеграция с Яндекс	Перегруженность интерфейса, реклама
Fresh Auto	Маркетплейс, онлайн-кредиты	Современный интерфейс, онлайн-оформление	Ограниченнная география работы
Major Auto	Информация о наилиции, акции	Актуальные данные, уведомления	Ограниченный функционал

### **1.3 Техническое задание**

#### **1.3.1 Определение пользовательских требований**

На основании анализа существующих решений (Auto.ru, Fresh Auto, Major Auto), можно выделить следующие пользовательские ожидания и предпочтения:

- простота и интуитивность интерфейса;
- возможность быстрого поиска по параметрам;
- персонализация и управление избранным;
- минимум ограничений без регистрации;
- отсутствие лишней рекламы.

В связи с этим, пользователи ожидают от создаваемого приложения:

- удобного интерфейса для подбора автомобилей;
- возможности фильтрации и сравнения моделей;
- отображения актуальной информации без регистрации;
- сохранения списка «избранных» автомобилей;
- отображения контактной информации автосалона и навигации к нему;
- работы без постоянного подключения к интернету.

### **1.3.2 Определение функциональных требований**

На основе пользовательских требований и анализа аналогов, формулируются следующие функциональные требования к системе:

1. Каталог автомобилей:

- загрузка и отображение списка автомобилей;
- фильтрация по основным параметрам (марка, модель, КПП, привод и т.д.);
- просмотр детальной информации по выбранному автомобилю.

2. Система авторизации и регистрации:

- регистрация нового пользователя с сохранением данных;
- авторизация существующего пользователя;
- хранение сессии через SharedPreferences.

3. Избранное:

- добавление и удаление автомобилей в избранное;
- хранение списка избранного в локальной базе данных;
- отображение только избранных моделей в отдельном разделе.

4. Профиль пользователя:

- отображение имени пользователя;
- отображение списка купленных автомобилей;
- отображение текущего состояния автомобиля (пробег, топливо, масло и т.д.).

5. Контактная информация:

- отображение адреса автосалона и способа связи;
- интеграция с картой;
- возможность перехода к звонку или чату.

6. Интерфейс:

- шаблоны отображения карточек, списков, форм регистрации;
- использование ресурсов res/drawable и XML-макетов.

7. Работа с локальной базой данных (Room):

- хранение информации об авторизованном пользователе;
  - хранение избранных и купленных автомобилей;
  - поддержка операций CRUD (создание, чтение, обновление, удаление).
8. Производительность и UX:
- высокая скорость отклика интерфейса;
  - стабильная работа при медленном соединении;
  - отображение placeholder'ов при загрузке изображений.

## 2 ПРОЕКТНАЯ ЧАСТЬ

### 2.1 Описание процесса в нотации IDEF0

Функциональная модель в нотации IDEF0 представляет собой совокупность взаимосвязанных процессов (функций) [2], реализуемых внутри системы, а также взаимодействие этих процессов с внешней средой. Каждый функциональный блок изображается в виде прямоугольника и отражает определённую функцию системы. Стандарт IDEF0 предполагает, что на одной диаграмме должно быть от трёх до шести блоков на уровне декомпозиции, но на контекстном уровне допускается только один блок, описывающий всю систему в целом.

Каждая стрелка на диаграмме имеет своё назначение:

- вход (Input) — данные, преобразуемые функцией для получения результата;
- выход (Output) — результат преобразования, создаваемый функцией;
- управление (Control) — правила, ограничения и условия, определяющие поведение функции;
- механизм (Mechanism) — ресурсы или инструменты, обеспечивающие выполнение функции.

В данном случае рассматривается контекстная диаграмма системы «Автосалон», отображающая весь процесс работы мобильного приложения. Центральный функциональный блок описывает общее поведение приложения — от обработки пользовательских данных до формирования интерфейса.

На входе в систему подаются:

- список автомобилей в формате JSON, полученный из Firebase;
- введённые пользователем данные при регистрации или авторизации;
- данные о состоянии автомобилей (пробег, уровень топлива), необходимые для отображения в профиле.

В качестве управляющих воздействий (Control) выступают:

- бизнес-логика, определяющая сценарии работы с избранным, фильтрацией и просмотром карточек;
- условия фильтрации, устанавливаемые пользователем (марка, КПП, тип привода и т. д.);
- текущая сессия пользователя (SharedPreferences), влияющая на доступ к данным;
- XML-макеты, служащие шаблонами визуального представления информации.

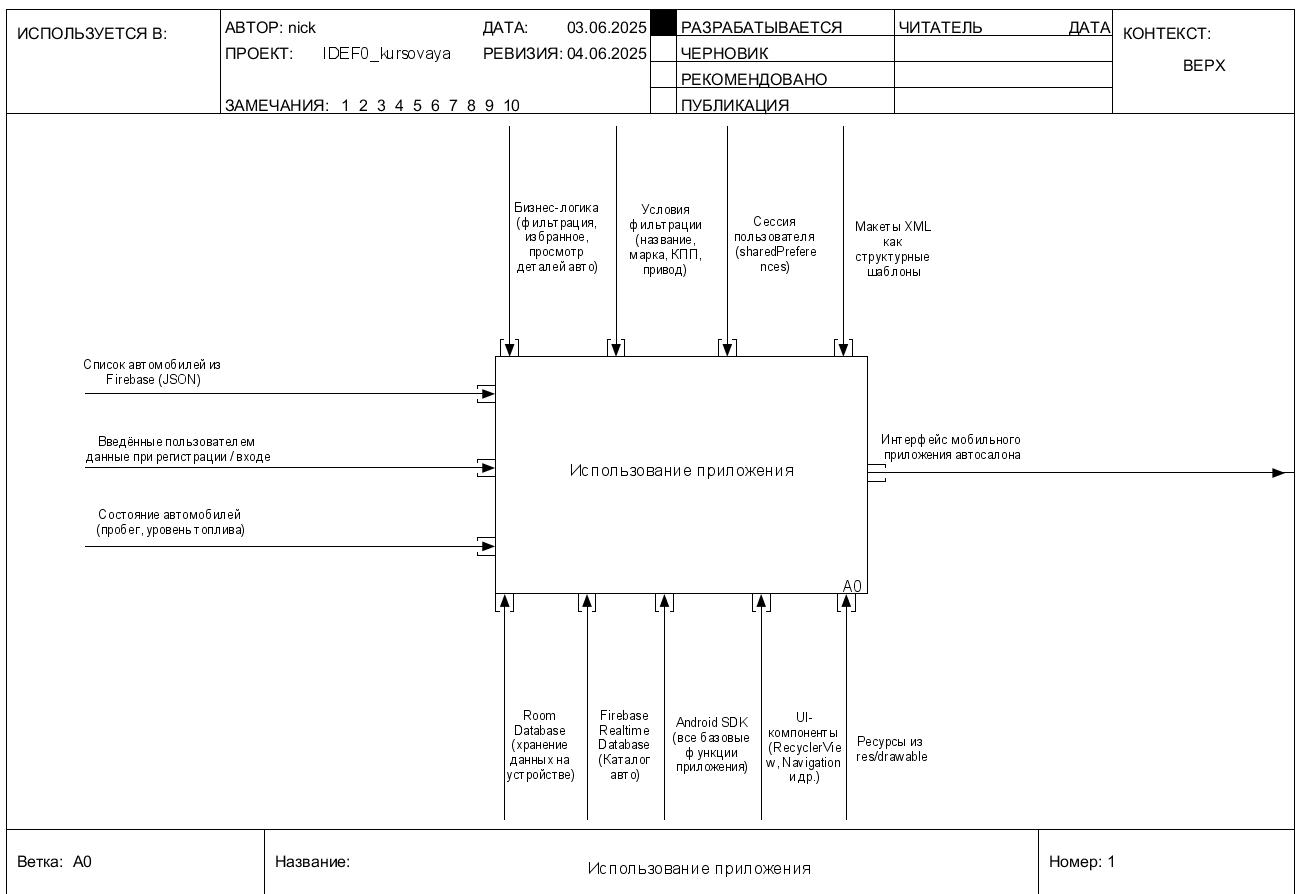
Выполнение функций обеспечивают механизмы (Mechanism):

- Room Database — локальное хранилище данных на устройстве;
- Firebase Realtime Database — удалённое хранилище каталога автомобилей;
- Android SDK — набор системных библиотек и базовых функций платформы Android;
- UI-компоненты — элементы пользовательского интерфейса (RecyclerView, Navigation и др.);
- ресурсы из папки res/drawable — изображения и стили, используемые в интерфейсе.

На выходе система формирует:

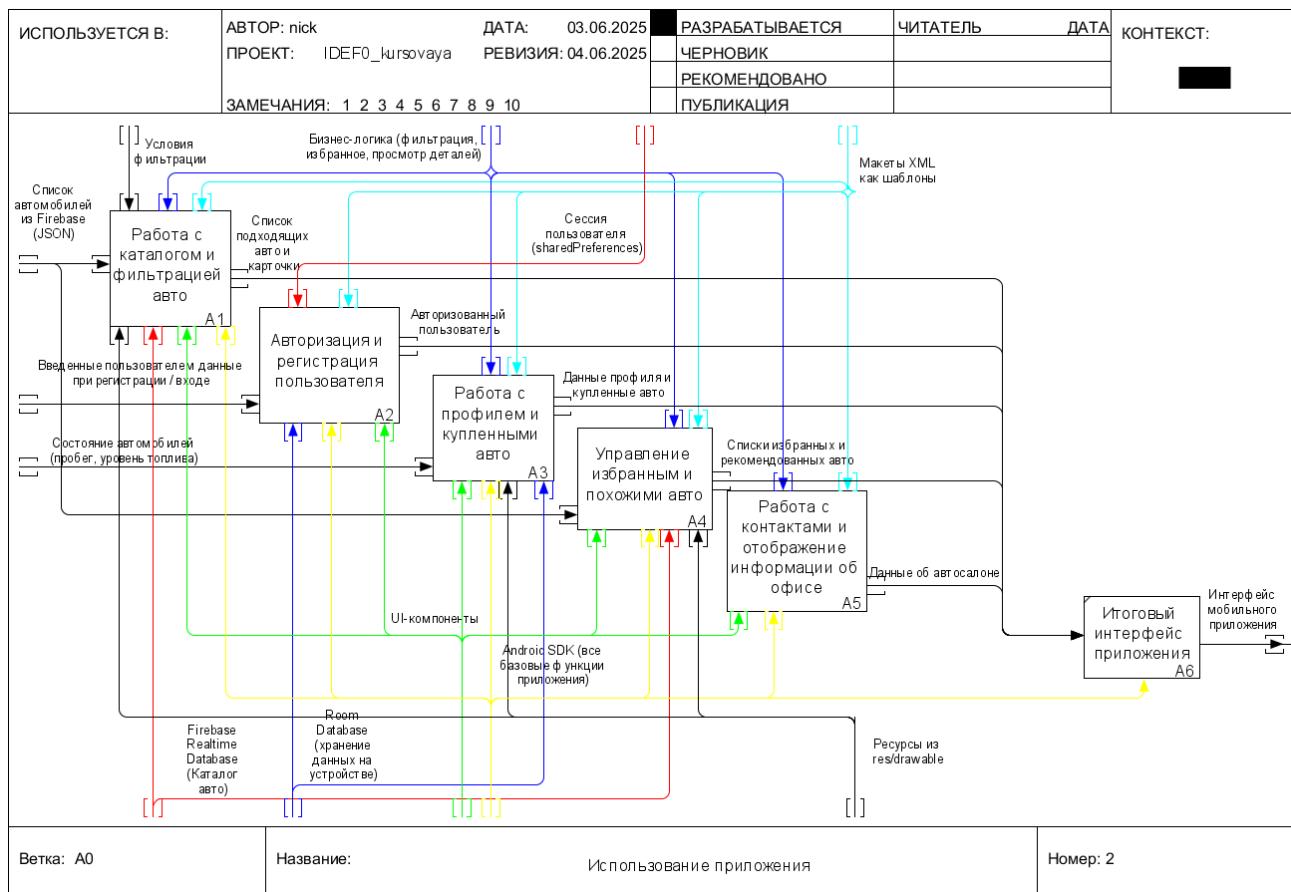
- интерфейс мобильного приложения автосалона, который отображается пользователю и обеспечивает доступ к каталогу, профилю, карте и другим функциям.

Покажем на Рисунке 4 диаграмму процесса в нотации IDEF0.



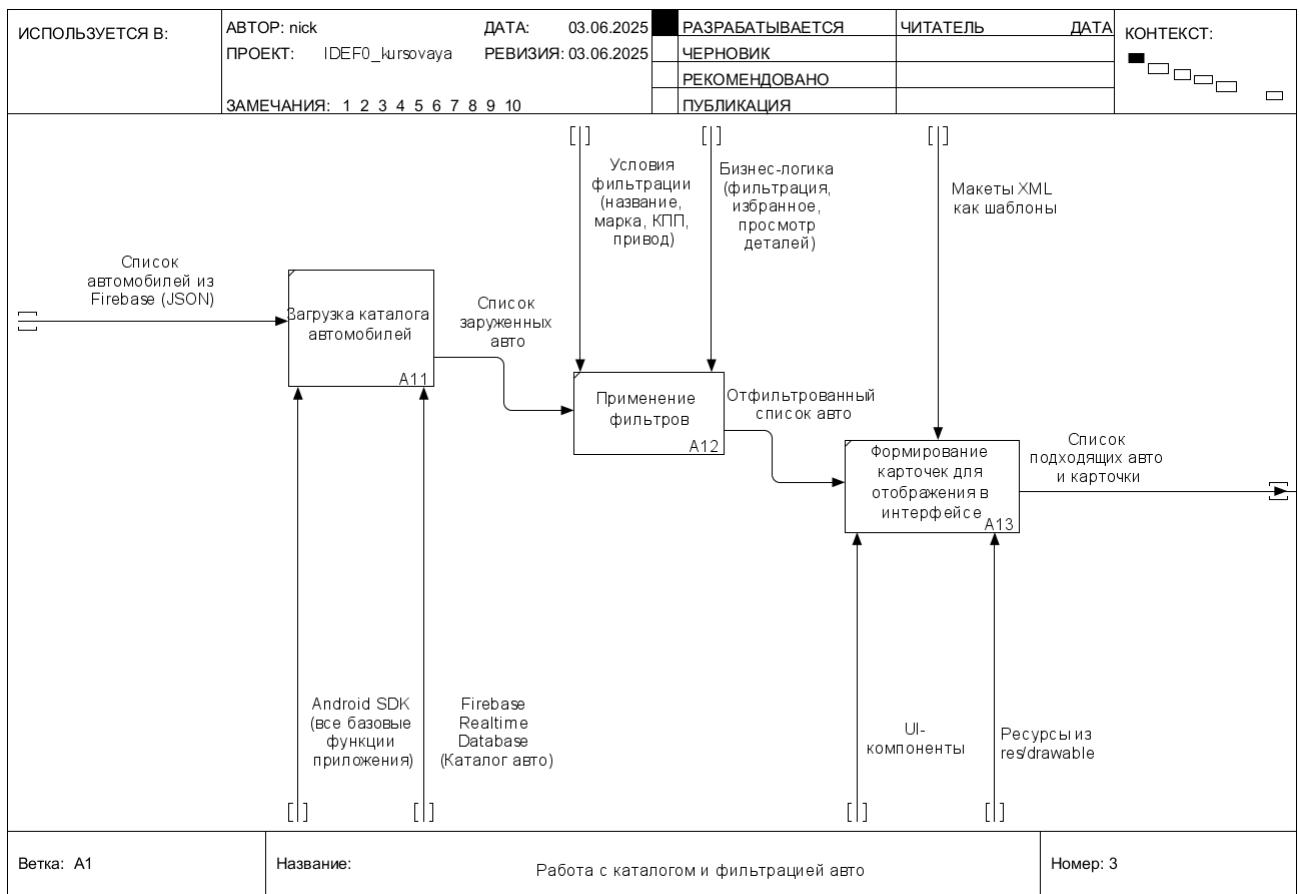
**Рисунок 4 – Описание процесса в нотации IDEF0**

Декомпозиция процесса использования прототипа приложения представлена на Рисунке 5.



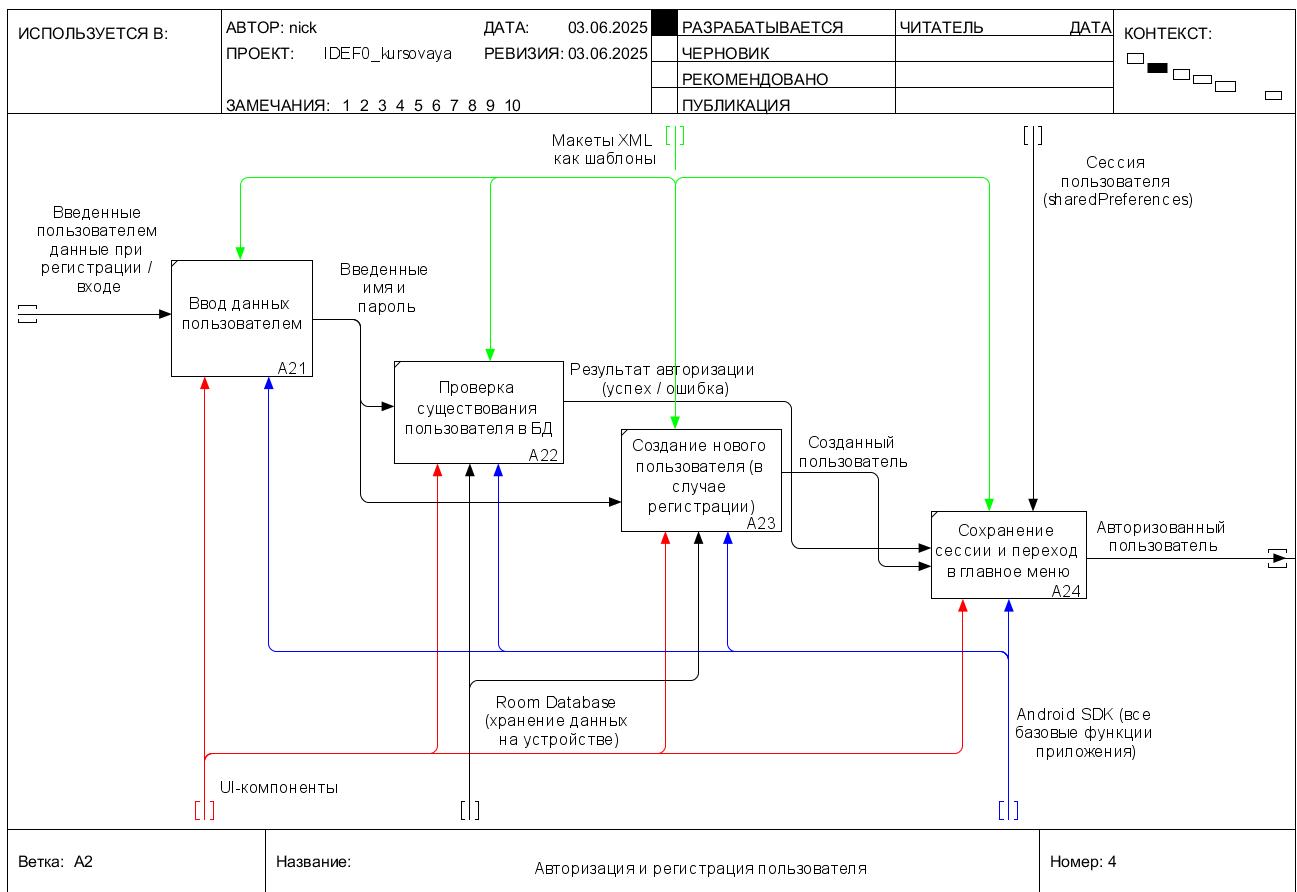
**Рисунок 5 – Декомпозиция процесса использования приложения в нотации IDEF0**

Процесс работы с каталогом и фильтрацией автомобилей (Рисунок 6) включает в себя загрузку данных об автомобилях из удалённой базы Firebase, применение условий фильтрации (по марке, КПП, приводу и др.), а также формирование карточек подходящих авто для отображения в интерфейсе. Процесс ограничивается бизнес-логикой приложения и структурными XML-шаблонами, отвечающими за визуальное представление. На выходе формируется отфильтрованный список автомобилей и соответствующие карточки.



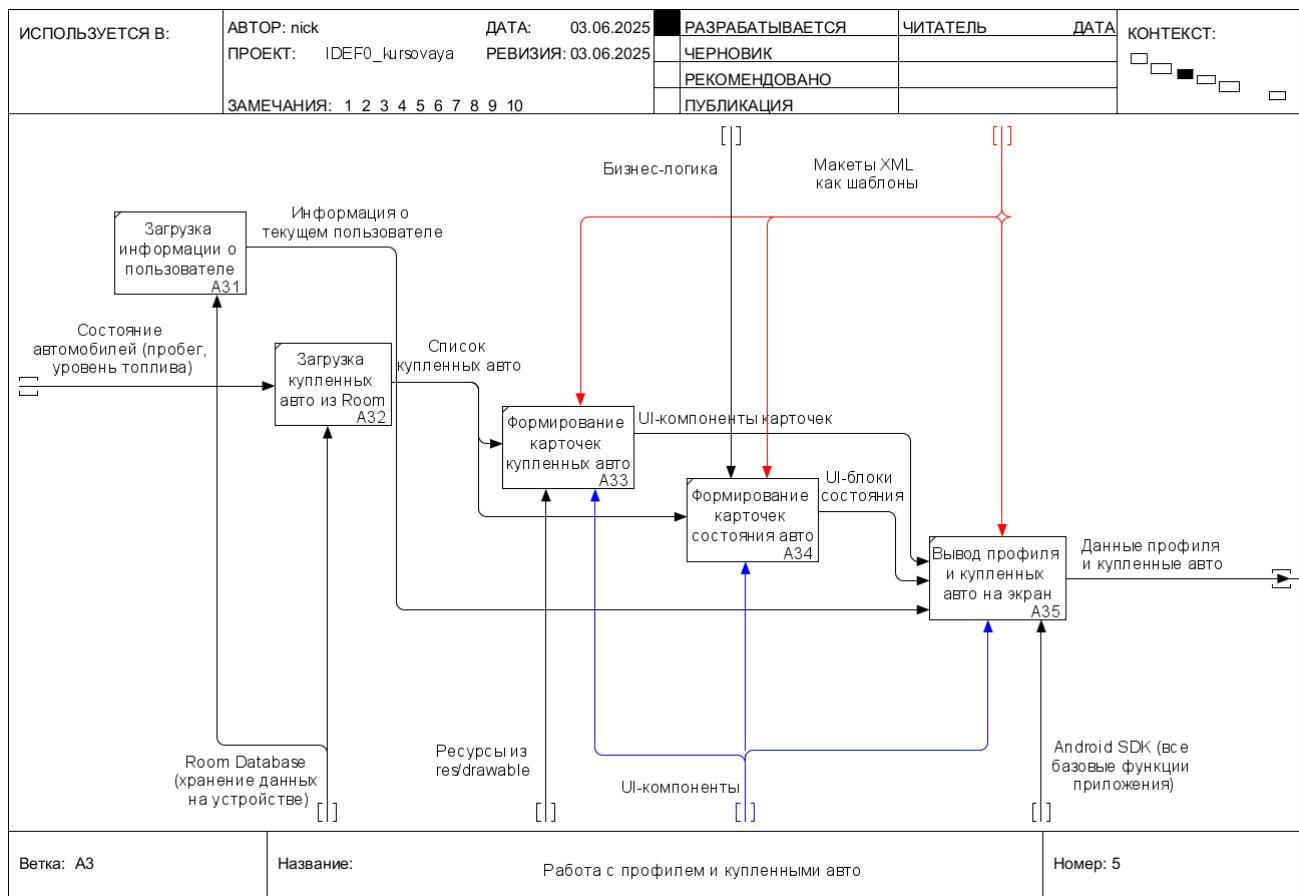
**Рисунок 6 – Декомпозиция процесса “Работа с каталогом и фильтрацией авто”**

Процесс авторизации и регистрации пользователя (Рисунок 7) охватывает все этапы взаимодействия пользователя с системой при входе или создании новой учетной записи. На вход подаются введённые данные (логин и пароль), которые пользователь вводит через графический интерфейс. Эти данные проходят проверку на наличие в локальной базе данных (Room DB). В случае успешной проверки осуществляется авторизация, в противном случае — создаётся новая учетная запись. Далее происходит сохранение сеанса в системе (SharedPreferences) и переход в основное меню приложения. Управление процессом осуществляется с использованием XML-шаблонов интерфейса. На выходе формируется объект авторизованного пользователя, необходимый для персонализированной работы приложения.



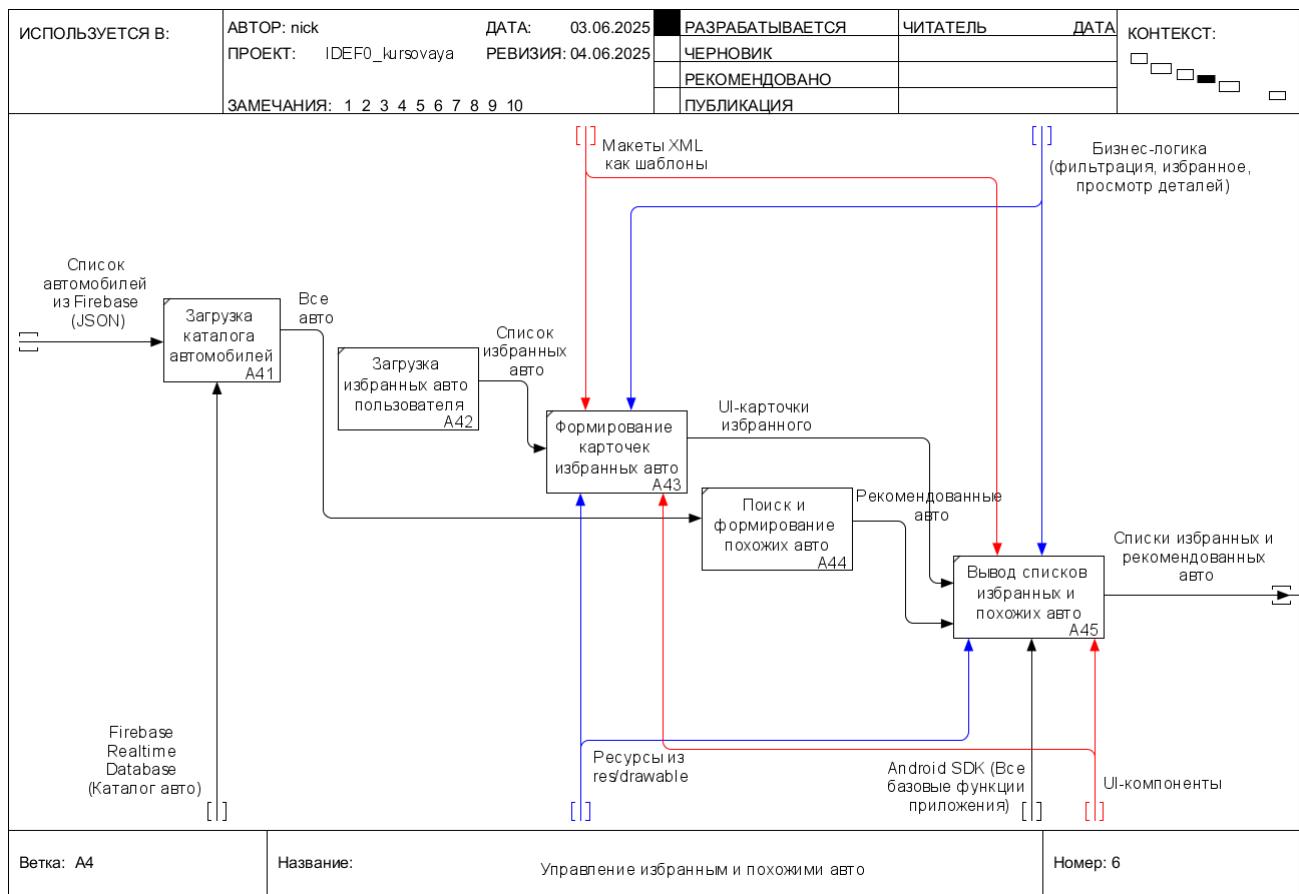
**Рисунок 7 – Декомпозиция процесса “Авторизация и регистрация пользователя”**

Процесс работы с профилем и купленными автомобилями (Рисунок 8) начинается с загрузки информации о текущем пользователе и получения состояния автомобилей. Сведения извлекаются из локальной базы данных Room DB, включая данные о пробеге, уровне топлива и списке купленных авто. Далее формируются карточки профиля и состояния автомобилей с использованием бизнес-логики, UI-компонентов и шаблонов интерфейса (макетов XML). Информация представляется в виде отдельных визуальных блоков и выводится на экран пользователя. На выходе процесса формируются полные данные профиля и купленных автомобилей, отображаемые в соответствующем фрагменте приложения.



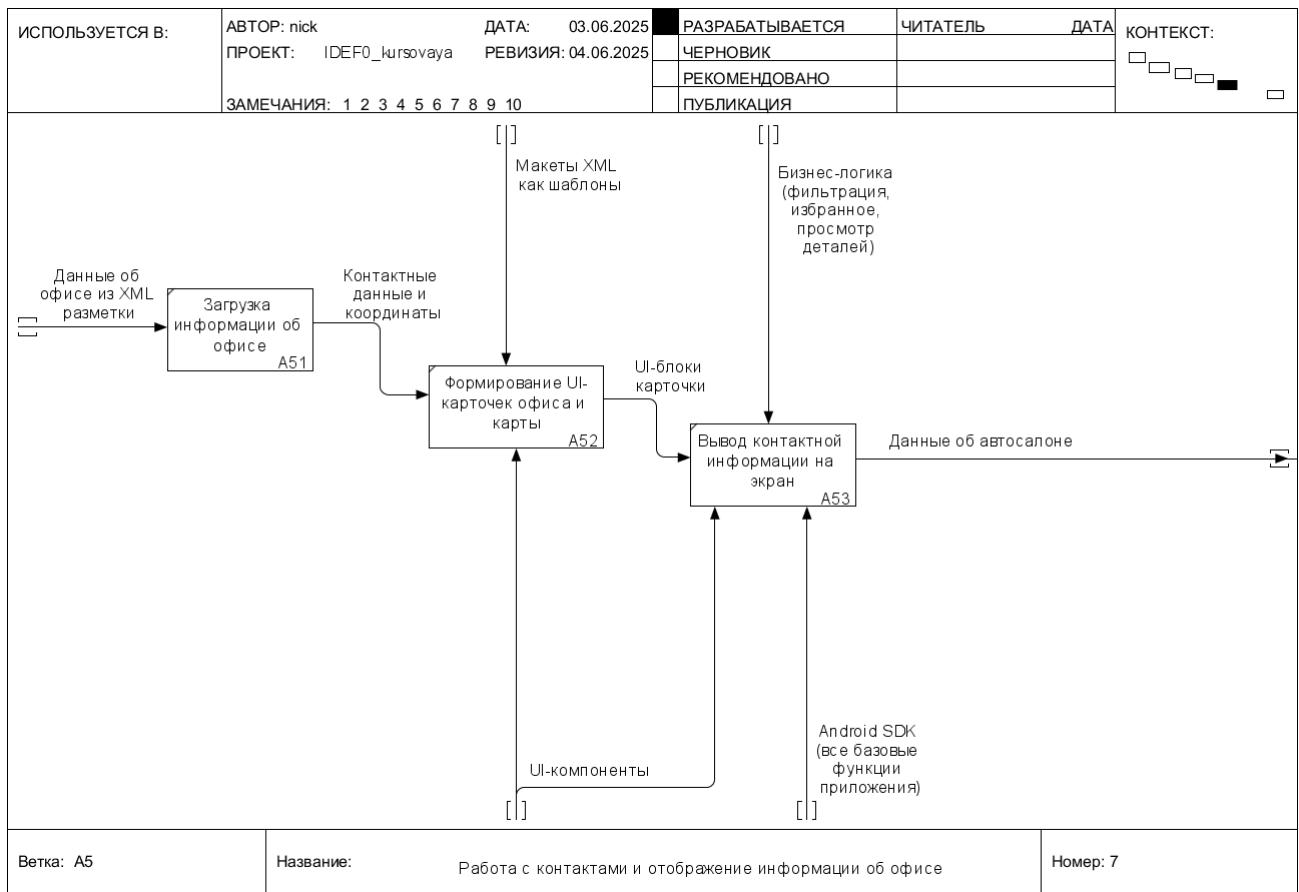
**Рисунок 8 – Декомпозиция процесса “Работа с профилем и купленными авто”**

Процесс управления избранным и похожими автомобилями (Рисунок 9) начинается с загрузки каталога всех автомобилей из Firebase и получения списка избранных авто пользователя из локальной базы данных. Эти данные используются для формирования UI-карточек избранных моделей. Далее по критериям схожести выполняется поиск аналогичных автомобилей. На основе полученных данных формируются рекомендованные карточки, с использованием бизнес-логики, шаблонов интерфейса и UI-компонентов. Финальный результат — формирование и вывод списка избранных и похожих авто, отображаемого в соответствующем разделе приложения.



**Рисунок 9 – Декомпозиция процесса “Управление избранным и похожими авто”**

Процесс работы с контактами и отображение информации об офисе (Рисунок 10) начинается с загрузки информации об офисе из XML-разметки, содержащей координаты и контактные данные. Эти сведения передаются на этап формирования пользовательских карточек, где с учетом шаблонов интерфейса и бизнес-логики создаются UI-блоки с картой и контактами. Используя UI-компоненты и стандартные функции Android SDK, данные выводятся на экран в виде структурированной информации об офисе автосалона. Результатом является готовый для взаимодействия пользователем экран с адресом, картой и средствами связи.



**Рисунок 10 – Декомпозиция процесса “Работа с контактами и отображение информации об офисе”**

## 2.2 Описание процесса в нотации DFD

Диаграмма в нотации DFD [3] отображает течение информации в пределах процесса. Для отображения входных и выходных данных используются стрелки. Процессы отображаются прямоугольниками с скругленными углами. Хранилища — прямоугольники без правой стороны, внешние объекты — прямоугольники с широкими границами [4].

На Рисунке 11 представлена диаграмма в нотации DFD, отображающая взаимодействие пользователя с мобильным приложением автосалона.

Изначально происходит авторизация и регистрация пользователя. На вход поступают логин и пароль, данные проверяются в локальной базе Room DB, при необходимости создается новый пользователь. Сессия сохраняется в SharedPreferences и подтвержденный вход направляется на устройство пользователя.

После входа пользователь может просматривать каталог автомобилей. Приложение получает список авто из Firebase DB и отображает карточки с учетом примененных фильтров. Карточки формируются с помощью XML-шаблонов и UI-компонентов.

Далее пользователь может просматривать профиль и купленные авто. Сведения загружаются из Room DB, карточки формируются на основе состояния авто. Визуализация реализуется с использованием ресурсов приложения.

Также пользователь может управлять избранным: добавлять или удалять авто. Список избранного хранится в локальной базе, формируются отдельные карточки, которые выводятся на экран вместе с рекомендованными моделями.

Наконец, пользователь может перейти к экрану с контактами автосалона. Информация о местоположении и способах связи отображается с использованием шаблонов и данных XML.

Результатом работы всех процессов является полноценный интерфейс мобильного приложения автосалона, отображаемый на устройстве пользователя.

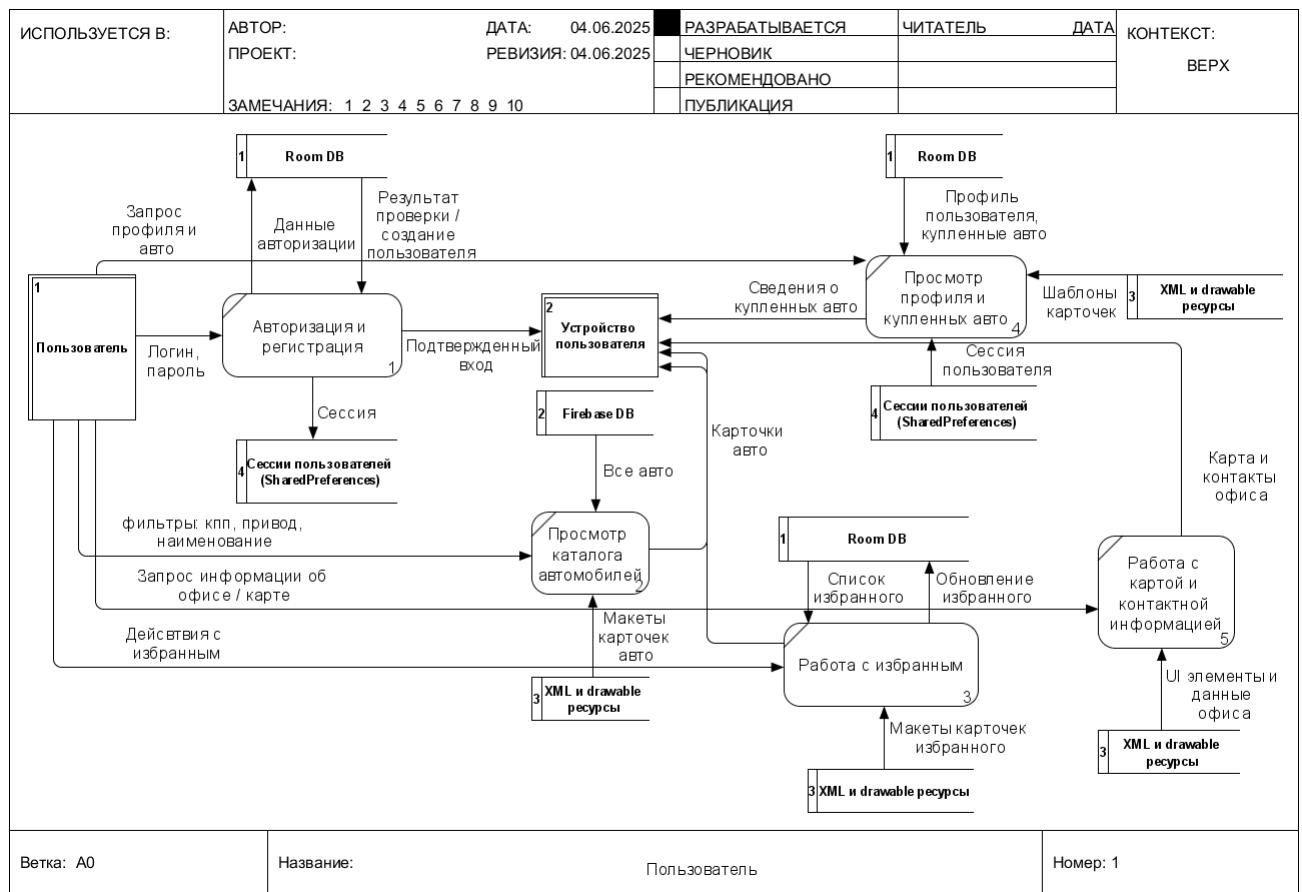


Рисунок 11 – Декомпозиция использования приложения в нотации DFD

## **2.3 Описание и обоснование выбора программного обеспечения**

Для реализации мобильного приложения автосалона в качестве основной интегрированной среды разработки была выбрана Android Studio, созданная и поддерживаемая компанией Google. Этот инструмент является официальной средой разработки для платформы Android и обеспечивает полный набор функций для создания, тестирования и отладки мобильных приложений.

В качестве основного языка программирования использовался Java, так как он является одним из ключевых языков, поддерживаемых платформой Android. Java обладает широкой документацией, стабильной экосистемой и богатой библиотечной поддержкой, что упрощает реализацию бизнес-логики и взаимодействие с системными компонентами Android. Кроме того, использование Java обеспечивает высокую совместимость с компонентами Android SDK и внутренними механизмами фреймворка.

Также в проекте активно применялись встроенные инструменты Android SDK, такие как RecyclerView, Navigation, SharedPreferences и Room. Эти компоненты позволили реализовать ключевые функции приложения: фильтрацию автомобилей, ведение избранного, отображение данных профиля и локальное хранение информации на устройстве.

## **2.4 Архитектура программной системы**

В ходе разработки проекта структура Java-классов была логически организована по пакетам, каждый из которых отвечает за отдельную область функциональности:

1. activities — содержит основные активности приложения:
  - MainActivity — главная точка входа после авторизации;

- LoginActivity, RegisterActivity — экраны входа и регистрации;
  - DetailsActivity — отображение детальной информации об автомобиле.
2. fragments — содержит UI-фрагменты:
- HomeFragment — список доступных автомобилей;
  - FavoritesFragment — отображение избранных авто;
  - ProfileFragment — профиль пользователя с купленными авто;
  - ContactsFragment — экран с контактной информацией и картой.
3. data — отвечает за работу с локальной базой данных Room:
- AppDatabase — конфигурация базы данных;
  - CarDao, UserDao — интерфейсы для доступа к таблицам;
  - CarData — инициализация демонстрационных данных.
4. models — содержит модельные классы:
- Car — модель автомобиля;
  - User — модель пользователя.
5. adapters — хранит адаптеры для отображения списков:
- CarAdapter — адаптер для RecyclerView.
6. utils — вспомогательные классы:
- DatabaseHelper — работа с Room;
  - FirebaseHelper — взаимодействие с Firebase.
- App — класс инициализации глобальных настроек.

Также в директории res используется стандартная структура:

- layout/ — XML-макеты экранов и фрагментов;
- drawable/ — изображения и иконки;
- values/ — ресурсы (строки, стили, цвета);
- menu/ — XML-файл с меню нижней навигации.
- mipmap/ — хранит иконки запуска приложения в разных разрешениях (например, ic\_launcher).

- `navigation/` — содержит граф навигации `nav_graph.xml`, который описывает переходы между экранами.
- `xml/` — используется для хранения дополнительных настроек, таких как `preferences.xml` или файл с описанием фильтров (например, `car_filters.xml`).

## 3 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

### 3.1 Описание моделей и структур данных

Класс HomeFragment [5] отвечает за отображение каталога автомобилей в главном экране приложения. Он реализует функциональность поиска, фильтрации и отображения карточек автомобилей, полученных из базы данных Room.

Основные компоненты:

- RecyclerView используется для отображения списка автомобилей;
- EditText (searchInput) [6] реализует текстовый поиск по названию модели;
- три Spinner отвечают за фильтрацию по производителю, типу коробки передач и приводу;
- CarAdapter связывает данные с отображением в RecyclerView.

При первом запуске, если база пуста, список заполняется из CarData и сохраняется в локальную БД. Пользователь может фильтровать автомобили по нескольким критериям одновременно. При клике на автомобиль открывается DetailsActivity с подробной информацией.

Методы:

- onCreateView — инициализирует все компоненты, настраивает адаптер и обработчики;
- initSpinners — готовит списки значений для выпадающих фильтров;
- filterAll — производит поиск и фильтрацию автомобилей по введённым условиям;
- onResume — при возврате на фрагмент обновляет список автомобилей и пересчитывает фильтрацию.

На Рисунке 12 покажем первую часть кода класса HomeFragment.java.

```
public class HomeFragment extends Fragment {
    private RecyclerView recyclerView;
    3 usages
    private CarAdapter adapter;
    10 usages
    private List<Car> carList;
    2 usages
    private List<Car> filteredList;
    3 usages
    private EditText searchInput;
    4 usages
    private Spinner spinnerManufacturer, spinnerGearbox, spinnerDrive;

    public HomeFragment() {}

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_home, container, attachToRoot: false);

        recyclerView = view.findViewById(R.id.carRecyclerView);
        searchInput = view.findViewById(R.id.search_input);
        spinnerManufacturer = view.findViewById(R.id.spinner_manufacturer);
        spinnerGearbox = view.findViewById(R.id.spinner_gearbox);
        spinnerDrive = view.findViewById(R.id.spinner_drive);

        recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));

        AppDatabase db = DatabaseHelper.getDatabase(getContext());
        carList = db.carDao().getAllCars();

        if (carList.isEmpty()) {
            carList = CarData.getCarList(); //список по умолчанию
            for (Car car : carList) {
                db.carDao().insert(car); //Добавляем в БД
            }
            carList = db.carDao().getAllCars(); //Перезапрашиваем из БД
        }

        filteredList = new ArrayList<>(carList);

        initSpinners();
    }
}
```

**Рисунок 12 – Код HomeFragment ч.1**

На Рисунке 13 покажем вторую часть кода HomeFragment.java.

```

        adapter = new CarAdapter(getContext(), filteredList, car -> {
            Intent intent = new Intent(getContext(), DetailsActivity.class);
            intent.putExtra("name", "car", car);
            startActivity(intent);
        });

        recyclerView.setAdapter(adapter);

        searchInput.addTextChangedListener(new TextWatcher() {
            @Override public void beforeTextChanged(CharSequence s, int start, int count, int after) {}
            @Override public void onTextChanged(CharSequence s, int start, int before, int count) {
                filterAll();
            }
            @Override public void afterTextChanged(Editable s) {}
        });

        return view;
    }

    1 usage
    private void initSpinners() {
        Set<String> manufacturers = new HashSet<>();
        Set<String> gearboxes = new HashSet<>();
        Set<String> drives = new HashSet<>();

        manufacturers.add("Все марки");
        gearboxes.add("Все коробки");
        drives.add("Все приводы");

        for (Car car : carList) {
            manufacturers.add(car.getManufacturer());
            gearboxes.add(car.getGearBox());
            drives.add(car.getDriveGear());
        }

        ArrayAdapter<String> adapterMan = new ArrayAdapter<>(requireContext(), android.R.layout.simple_spinner_dropdown_item, new ArrayList<>(manufacturers));
        ArrayAdapter<String> adapterGear = new ArrayAdapter<>(requireContext(), android.R.layout.simple_spinner_dropdown_item, new ArrayList<>(gearboxes));
        ArrayAdapter<String> adapterDrive = new ArrayAdapter<>(requireContext(), android.R.layout.simple_spinner_dropdown_item, new ArrayList<>(drives));
    }
}

```

**Рисунок 13 – Код HomeFragrment ч.2**

На Рисунке 14 покажем третью часть кода HomeFragment.

```

spinnerManufacturer.setAdapter(adapterMan);
spinnerGearbox.setAdapter(adapterGear);
spinnerDrive.setAdapter(adapterDrive);

AdapterView.OnItemSelectedListener listener = new AdapterView.OnItemSelectedListener() {
    1 usage
    @Override public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        filterAll();
    }

    1 usage
    @Override public void onNothingSelected(AdapterView<?> parent) {}
};

spinnerManufacturer.setOnItemSelectedListener(listener);
spinnerGearbox.setOnItemSelectedListener(listener);
spinnerDrive.setOnItemSelectedListener(listener);
}

3 usages
private void filterAll() {
    String search = searchInput.getText().toString().toLowerCase();
    String selectedManufacturer = spinnerManufacturer.getSelectedItem().toString();
    String selectedGearbox = spinnerGearbox.getSelectedItem().toString();
    String selectedDrive = spinnerDrive.getSelectedItem().toString();

    List<Car> filtered = new ArrayList<>();

    for (Car car : carList) {
        boolean matchesSearch = car.getModelName().toLowerCase().contains(search);
        boolean matchesManufacturer = selectedManufacturer.equals("Все марки") || car.getManufacturer().equals(selectedManufacturer);
        boolean matchesGearbox = selectedGearbox.equals("Все коробки") || car.getGearBox().equals(selectedGearbox);
        boolean matchesDrive = selectedDrive.equals("Все приводы") || car.getDriveGear().equals(selectedDrive);

        if (matchesSearch && matchesManufacturer && matchesGearbox && matchesDrive) {
            filtered.add(car);
        }
    }
}

```

**Рисунок 14 – Код HomeFragment ч.3**

На Рисунке 15 покажем финальную часть кода HomeFragment.

```

        adapter.updateList(filtered);
    }

    @Override
    public void onResume() {
        super.onResume();
        carList.clear();
        carList.addAll(DatabaseHelper.getDatabase(getContext()).carDao().getAllCars());
        filterAll();
    }
}

```

**Рисунок 15 – Код HomeFragment ч.4**

Класс Car представляет модель автомобиля и используется для хранения и отображения информации о конкретных транспортных средствах в приложении.

Это сущность Room, аннотированная `@Entity(tableName = "car")`, благодаря чему данные хранятся в локальной базе данных. Класс реализует интерфейс `Serializable`, что позволяет передавать экземпляры между экранами через Intent.

Основные поля включают:

- `id` — уникальный идентификатор (автоматически генерируется);
- `modelName`, `manufacturer`, `year`, `imageUrl` — основные характеристики автомобиля;
- `engineType`, `engineVolume`, `gearBox`, `driveGear` — параметры двигателя и трансмиссии;
- `price` — цена автомобиля;
- `mileage`, `fuelLevel`, `approxKmLeft` — показатели состояния авто;
- `isFavorite`, `isBought` — флаги, обозначающие добавление в избранное и покупку.

Метод `needsOilChange()` используется для определения необходимости замены масла по пробегу.

Класс снабжен полным набором геттеров и сеттеров, необходимых для отображения и обновления данных в пользовательском интерфейсе.

Класс `User` представляет учетную запись пользователя и используется для авторизации и аутентификации в системе. Он аннотирован `@Entity`, что позволяет сохранять данные пользователей в локальной базе Room.

Содержит следующие поля:

- `id` — первичный ключ, генерируемый автоматически;
- `username` — имя пользователя;
- `password` — строка пароля (в текущей реализации хранится в открытом виде, что допустимо только на этапе прототипа).

Конструктор принимает имя пользователя и пароль. Класс также содержит методы доступа к полю `username`.

Модель используется в связке с `UserDao` для создания, поиска и хранения учетных записей в локальной базе данных приложения.

Интерфейс CarDao (Рисунок 16) представляет собой компонент доступа к данным (DAO) [7] для работы с таблицей автомобилей в локальной базе данных Room. Он содержит методы для вставки, обновления и получения данных:

- `insert(Car car)` — добавляет новый автомобиль в базу данных;
- `updateCar(Car car)` — обновляет данные существующего автомобиля;
- `getAllCars()` — возвращает список всех автомобилей;
- `getFavoriteCars()` — возвращает список автомобилей, отмеченные как избранные;
- `getCarById(int id)` — получает автомобиль по его идентификатору;
- `getBoughtCars()` — возвращает список приобретённых пользователем автомобилей.

Этот интерфейс используется во всех фрагментах, где необходимо взаимодействие с автокаталогом, избранными или купленными авто, обеспечивая эффективную работу с локальной базой данных.

```

@Dao
public interface CarDao {

    2 usages 1 implementation
    @Insert
    void insert(Car car);

    3 usages 1 implementation
    @Update
    void updateCar(Car car);

    5 usages 1 implementation
    @Query("SELECT * FROM car")
    List<Car> getAllCars();

    2 usages 1 implementation
    @Query("SELECT * FROM car WHERE is_favorite = 1")
    List<Car> getFavoriteCars();

    1 usage 1 implementation
    @Query("SELECT * FROM car WHERE id = :id LIMIT 1")
    Car getCarById(int id);

    1 usage 1 implementation
    @Query("SELECT * FROM car WHERE bought = 1")
    List<Car> getBoughtCars();
}

```

**Рисунок 16 – Интерфейс CarDao**

Интерфейс UserDao (Рисунок 17) реализует методы доступа к данным пользователей в базе данных Room [7]. Он обеспечивает основные операции, связанные с регистрацией и авторизацией:

- `insert(User user)` — добавляет нового пользователя в базу данных;
- `login(String username, String password)` — проверяет наличие пользователя с указанными данными для авторизации;
- `getUserByUsername(String username)` — позволяет убедиться, что логин не занят при регистрации;
- `findByLogin(String username)` — возвращает пользователя по логину, используется для дополнительных проверок.

Этот DAO играет ключевую роль в модуле регистрации и входа в систему, обеспечивая базовую проверку данных и хранение аккаунтов.

```
@Dao
public interface UserDao {

    1 usage 1 implementation
    @Insert
    void insert(User user);      //Метод для добавления нового пользователя

    1 usage 1 implementation
    @Query("SELECT * FROM User WHERE username = :username AND password = :password")
    User login(String username, String password);   //Метод для проверки данных пользователя (авторизация)

    1 usage 1 implementation
    @Query("SELECT * FROM User WHERE username = :username")
    User getUserByUsername(String username);        //Метод для проверки уникальности логина при регистрации

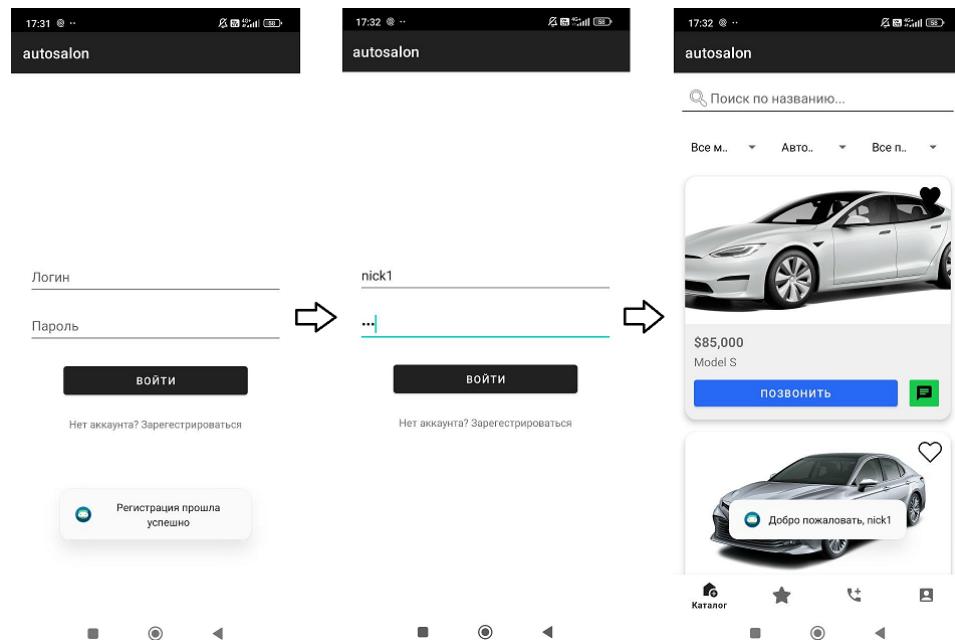
    1 usage 1 implementation
    @Query("SELECT * FROM user WHERE username = :username LIMIT 1")
    User findByLogin(String username);
}
```

Рисунок 17 – Интерфейс UserDao

### 3.2 Тестирование программного продукта

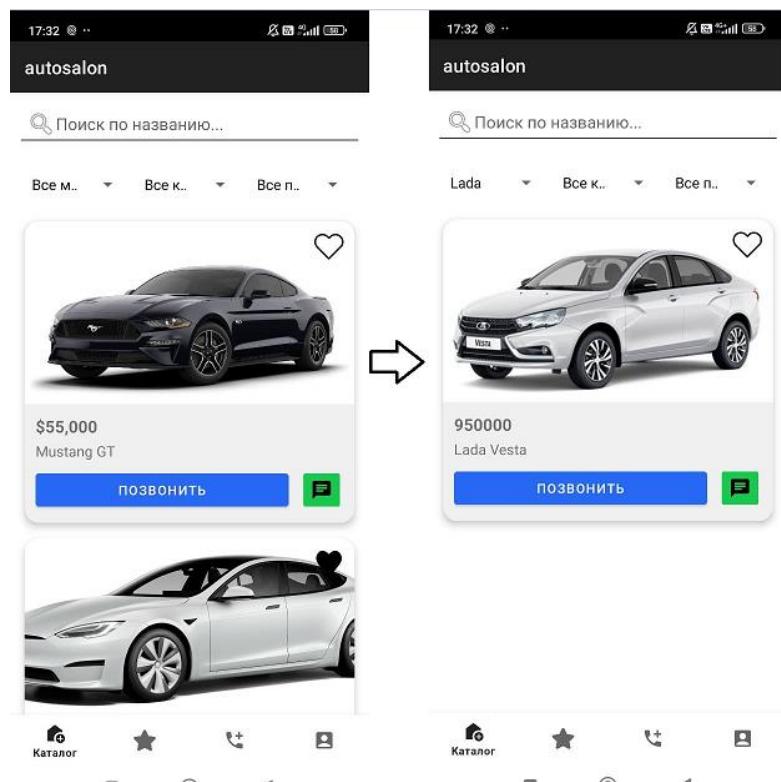
Для проведения тестирования мобильного приложения был выбран метод черного ящика, то есть метод, когда тестировщику неизвестно внутреннее устройство тестируемой системы. Для этого некоторым пользователям было выдано приложение, в ходе проверки которого были протестированы все функциональные возможности системы, работа при различной скорости интернета.

На Рисунке 18 результат тестирования экранов регистрации и авторизации.



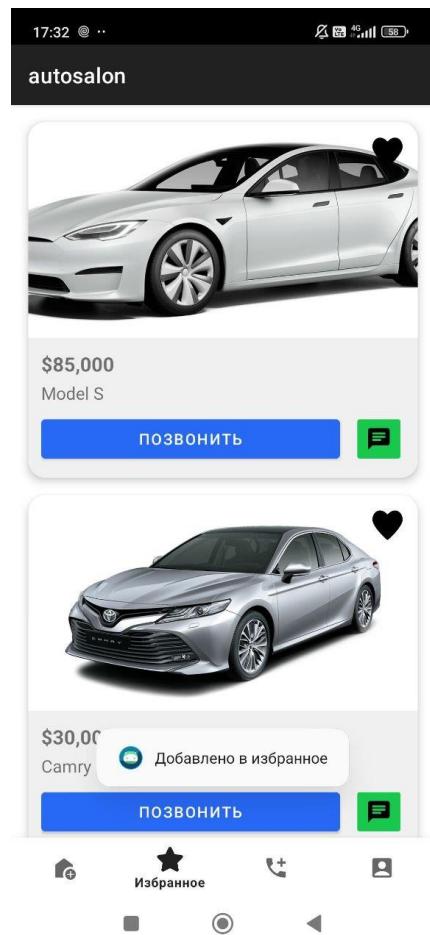
**Рисунок 18 – Экраны регистрации и авторизации**

Далее на Рисунке 19 показаны экраны каталога.



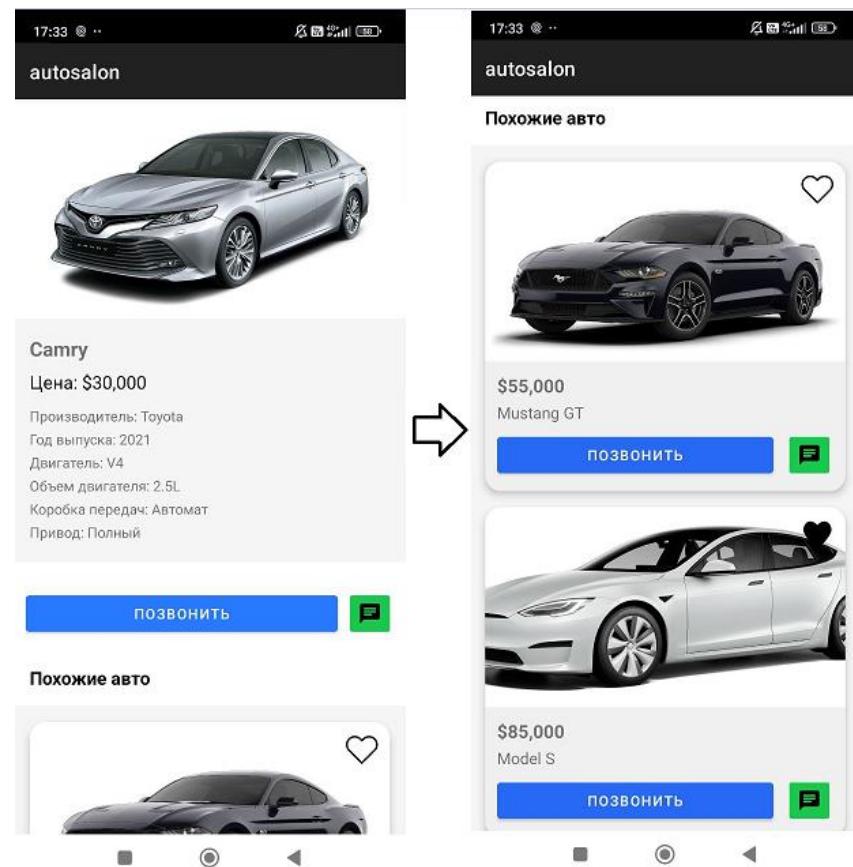
**Рисунок 19 – Экраны каталога**

На Рисунке 20 показан экран избранного.



**Рисунок 20 – Экран избранного**

На Рисунке 21 показаны экраны информации о конкретном авто.



**Рисунок 21 – Экраны конкретного авто**

На Рисунке 22 показан экран с контактными данными автосалона.

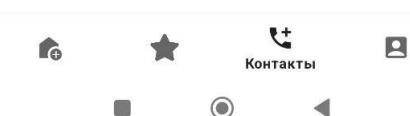
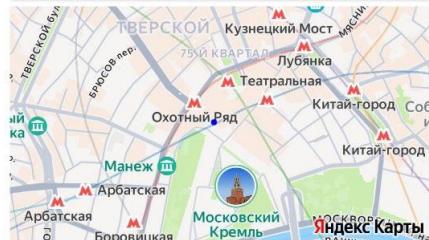


## Наш автосалон

ул. Примерная, д. 123

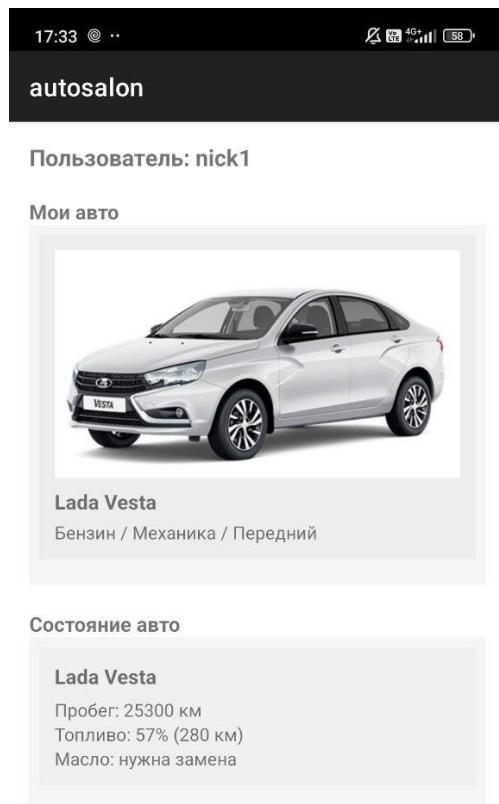
[ПОЗВОНИТЬ НАМ](#)

[НАПИСАТЬ НА EMAIL](#)



**Рисунок 22 – Экран с контактными данными**

На Рисунке 23 показан экран профиля пользователя.



**Рисунок 23 – Экран профиля пользователя**

### 3.3 Полный листинг

#### Листинг 1 – DetailsActivity

```
package com.example.autosalon.activities;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import com.example.autosalon.R;
import com.example.autosalon.data.CarDao;
import com.example.autosalon.utils.DatabaseHelper;
import com.example.autosalon.models.Car;

import java.util.List;

public class DetailsActivity extends AppCompatActivity {
    private ImageView detailImage;
    private TextView model, manufacturer, year, engine, volume, gearbox,
drive, price;

    private CarDao carDao;
    private Car car;

    private LinearLayout similarCarsContainer;
    private LayoutInflater inflater;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_details);

        detailImage = findViewById(R.id.detail_image);
        model = findViewById(R.id.detail_model);
        manufacturer = findViewById(R.id.detail_manufacturer);
        year = findViewById(R.id.detail_year);
        engine = findViewById(R.id.detail_engine);
        volume = findViewById(R.id.detail_volume);
        gearbox = findViewById(R.id.detail_gearbox);
        drive = findViewById(R.id.detail_drive);
        price = findViewById(R.id.detail_price);
        similarCarsContainer = findViewById(R.id.container_similar_cars);

        carDao = DatabaseHelper.getDatabase(this).carDao();
        inflater = LayoutInflater.from(this);

        car = (Car) getIntent().getSerializableExtra("car");

        if (car != null) {
            //Картинка
            int imageRes = getLocalImageResource(car.getModelName());
            detailImage.setImageResource(imageRes);
        }
    }
}
```

```

        model.setText(car.getModelName());
        manufacturer.setText("Производитель: " + car.getManufacturer());
        year.setText("Год выпуска: " + car.getYear());
        engine.setText("Двигатель: " + car.getEngineType());
        volume.setText("Объем двигателя: " + car.getEngineVolume());
        gearbox.setText("Коробка передач: " + car.getGearBox());
        drive.setText("Привод: " + car.getDriveGear());
        price.setText("Цена: " + car.getPrice());

        loadSimilarCars();
    }
}

private void loadSimilarCars() {
    List<Car> allCars = carDao.getAllCars();
    for (Car otherCar : allCars) {
        if (otherCar.getId() == car.getId()) continue; // не показываем
текущую машину

        View card = inflater.inflate(R.layout.item_car,
similarCarsContainer, false);

        ImageView image = card.findViewById(R.id.car_image);
        TextView modelText = card.findViewById(R.id.car_model);
        TextView priceText = card.findViewById(R.id.car_price);
        ImageView favIcon = card.findViewById(R.id.fav_icon);
        Button callBtn = card.findViewById(R.id.button_call);
        ImageButton msgBtn = card.findViewById(R.id.button_message);

        image.setImageResource(getLocalImageResource(otherCar.getModelName()));
        modelText.setText(otherCar.getModelName());
        priceText.setText(otherCar.getPrice());
        favIcon.setImageResource(otherCar.isFavorite() ?
R.drawable.ic_favorite : R.drawable.ic_favorite_border);

        favIcon.setOnClickListener(v -> {
            boolean fav = !otherCar.isFavorite();
            otherCar.setFavorite(fav);
            carDao.updateCar(otherCar);
            favIcon.setImageResource(fav ? R.drawable.ic_favorite :
R.drawable.ic_favorite_border);
        });

        similarCarsContainer.addView(card);
    }
}

private int getLocalImageResource(String modelName) {
    switch (modelName.toLowerCase()) {
        case "camry":
            return R.drawable.camry;
        case "bmw 3 series":
            return R.drawable.bmw3;
        case "kia rio":
            return R.drawable.rio;
        case "hyundai solaris":
            return R.drawable.solaris;
        case "audi a4":
            return R.drawable.audi_a4;
        case "tesla model 3":
            return R.drawable.tesla3;
        case "lada vesta":
            return R.drawable.lada;
        case "mustang gt":
            return R.drawable.mustang_gt;
    }
}

```

```
        return R.drawable.mustang;
    case "model s":
        return R.drawable.teslas;
    default:
        return R.drawable.placeholder;
    }
}
```

## Листинг 2 – LoginActivity

```
package com.example.autosalon.activities;

import android.os.Bundle;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.example.autosalon.activities.MainActivity;

import com.example.autosalon.data.UserDao;
import com.example.autosalon.models.User;
import com.example.autosalon.utils.DatabaseHelper;

import com.example.autosalon.R;

public class LoginActivity extends AppCompatActivity {

    private EditText etUsername, etPassword;
    private Button btnLogin;
    private UserDao userDao;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        etUsername = findViewById(R.id.et_login_username);
        etPassword = findViewById(R.id.et_login_password);
        btnLogin = findViewById(R.id.btn_login);

        userDao = DatabaseHelper.getDatabase(this).userDao();

        btnLogin.setOnClickListener(v-> {
            String username = etUsername.getText().toString().trim();
            String password = etPassword.getText().toString().trim();

            if (username.isEmpty() || password.isEmpty()) {
                Toast.makeText(this, "Введите логин и пароль",
Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

```
        return;
    }

    User user = userDao.login(username, password);
    if (user == null) {
        Toast.makeText(this, "Неверные данные",
Toast.LENGTH_SHORT).show();
        return;
    }

    //Сохранение текущей сессии локально
    SharedPreferences prefs = getSharedPreferences("UserSession",
Context.MODE_PRIVATE);
    prefs.edit().putString("loggedInUser", username).apply();

    Toast.makeText(this, "Добро пожаловать, " + username,
Toast.LENGTH_SHORT).show();

    //Переход в основную активность
    startActivity(new Intent(this, MainActivity.class));
    finish();
});

TextView tvRegister = findViewById(R.id.tv_register_link);
tvRegister.setOnClickListener(v -> {
    startActivity(new Intent(this, RegisterActivity.class));
    finish();
});
}
```

### Листинг 3 – MainActivity

```
package com.example.autosalon.activities;

import android.os.Bundle;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.example.autosalon.activities.MainActivity;

import com.example.autosalon.data.UserDao;
import com.example.autosalon.models.User;
import com.example.autosalon.utils.DatabaseHelper;

import com.example.autosalon.R;

public class LoginActivity extends AppCompatActivity
```

```

private EditText etUsername, etPassword;
private Button btnLogin;
private UserDao userDao;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);

    etUsername = findViewById(R.id.et_login_username);
    etPassword = findViewById(R.id.et_login_password);
    btnLogin = findViewById(R.id.btn_login);

    userDao = DatabaseHelper.getDatabase(this).userDao();

    btnLogin.setOnClickListener(v-> {
        String username = etUsername.getText().toString().trim();
        String password = etPassword.getText().toString().trim();

        if (username.isEmpty() || password.isEmpty()) {
            Toast.makeText(this, "Введите логин и пароль",
Toast.LENGTH_SHORT).show();
            return;
        }

        User user = userDao.login(username, password);
        if (user == null) {
            Toast.makeText(this, "Неверные данные",
Toast.LENGTH_SHORT).show();
            return;
        }

        //Сохранение текущей сессии локально
        SharedPreferences prefs = getSharedPreferences("UserSession",
Context.MODE_PRIVATE);
        prefs.edit().putString("loggedInUser", username).apply();

        Toast.makeText(this, "Добро пожаловать, " + username,
Toast.LENGTH_SHORT).show();

        //Переход в основную активность
        startActivity(new Intent(this, MainActivity.class));
        finish();
    });
}

TextView tvRegister = findViewById(R.id.tv_register_link);
tvRegister.setOnClickListener(v -> {
    startActivity(new Intent(this, RegisterActivity.class));
    finish();
});
}
}

```

#### Листинг 4 – RegisterActivity

```

package com.example.autosalon.activities;

import android.os.Bundle;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;

```

```
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import android.content.Intent;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.example.autosalon.data.UserDao;
import com.example.autosalon.models.User;
import com.example.autosalon.utils.DatabaseHelper;

import com.example.autosalon.R;

public class RegisterActivity extends AppCompatActivity {

    private EditText etUsername, etPassword;
    private Button btnRegister;
    private UserDao userDao;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        TextView tvLoginLink = findViewById(R.id.tv_login_link);
        tvLoginLink.setOnClickListener(v -> {
            startActivity(new Intent(this, LoginActivity.class));
            finish();
        });

        etUsername = findViewById(R.id.et_username);
        etPassword = findViewById(R.id.et_password);
        btnRegister = findViewById(R.id.btn_register);

        userDao = DatabaseHelper.getDatabase(this).userDao();

        btnRegister.setOnClickListener(v -> {
            String username = etUsername.getText().toString().trim();
            String password = etPassword.getText().toString().trim();

            if (username.isEmpty() || password.isEmpty()) {
                Toast.makeText(this, "Введите логин и пароль",
                Toast.LENGTH_SHORT).show();
                return;
            }

            if (userDao.getUserByUsername(username) != null) {
                Toast.makeText(this, "Пользователь с таким логином уже
существует", Toast.LENGTH_SHORT).show();
                return;
            }

            userDao.insert(new User(username, password));
            Toast.makeText(this, "Регистрация прошла успешно",
            Toast.LENGTH_SHORT).show();

            startActivity(new Intent(this, LoginActivity.class));
            finish();
        });
    }
}
```

## Листинг 5 – CarAdapter

```
package com.example.autosalon.adapters;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.example.autosalon.R;
import com.example.autosalon.data.CarDao;
import com.example.autosalon.models.Car;
import com.example.autosalon.utils.DatabaseHelper;

import java.util.List;

public class CarAdapter extends RecyclerView.Adapter<CarAdapter.CarViewHolder> {

    private Context context;
    private List<Car> carList;
    private OnItemClickListener listener;
    private CarDao carDao;

    public interface OnItemClickListener {
        void onItemClick(Car car);
    }

    public CarAdapter(Context context, List<Car> carList, OnItemClickListener listener) {
        this.context = context;
        this.carList = carList;
        this.listener = listener;
        this.carDao = DatabaseHelper.getDatabase(context).carDao();
    }

    public void updateList(List<Car> newList) {
        carList.clear();
        carList.addAll(newList);
        notifyDataSetChanged();
    }

    @NonNull
    @Override
    public CarViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.item_car,
parent, false);
        return new CarViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull CarViewHolder holder, int position)
{
```

```

        Car car = carList.get(position);
        holder.modelText.setText(car.getModelName());
        holder.priceText.setText(car.getPrice());

        // Подгружаем изображение

holder.imageView.setImageResource(getLocalImageResource(car.getModelName()));

        // Слушатель нажатия на карточку
holder.itemView.setOnClickListener(v -> listener.onItemClick(car));

        // Установка иконки избранного
updateFavoriteIcon(holder.favIcon, car.isFavorite());

        // Обработка клика на иконку избранного
holder.favIcon.setOnClickListener(v -> {
    boolean newState = !car.isFavorite();
    car.setFavorite(newState);
    carDao.updateCar(car); // Обновляем в базе
    updateFavoriteIcon(holder.favIcon, newState);
    Toast.makeText(context,
        newState ? "Добавлено в избранное" : "Удалено из
избранного",
        Toast.LENGTH_SHORT).show();
});
}

private int getLocalImageResource(String modelName) {
    switch (modelName.toLowerCase()) {
        case "camry":
            return R.drawable.camry;
        case "bmw 3 series":
            return R.drawable.bmw3;
        case "kia rio":
            return R.drawable.rio;
        case "hyundai solaris":
            return R.drawable.solaris;
        case "audi a4":
            return R.drawable.audi_a4;
        case "tesla model 3":
            return R.drawable.tesla3;
        case "lada vesta":
            return R.drawable.lada;
        case "mustang gt":
            return R.drawable.mustang;
        case "model s":
            return R.drawable.teslas;
        default:
            return R.drawable.placeholder;
    }
}

private void updateFavoriteIcon(ImageView icon, boolean isFavorite) {
    icon.setImageResource(isFavorite ? R.drawable.ic_favorite :
R.drawable.ic_favorite_border);
}

@Override
public int getItemCount() {
    return carList.size();
}

public static class CarViewHolder extends RecyclerView.ViewHolder {
    ImageView imageView;

```

```

        TextView modelText, priceText;
        ImageView favIcon;

        public CarViewHolder(@NonNull View itemView) {
            super(itemView);
            imageView = itemView.findViewById(R.id.car_image);
            modelText = itemView.findViewById(R.id.car_model);
            priceText = itemView.findViewById(R.id.car_price);
            favIcon = itemView.findViewById(R.id.fav_icon);
        }
    }
}

```

## Листинг 6 – AppDatabase

```

package com.example.autosalon.data;

import androidx.room.Database;
import androidx.room.RoomDatabase;

import com.example.autosalon.models.Car;
import com.example.autosalon.models.User;

@Database(entities = {User.class, Car.class}, version = 4)      //Database –
аннотация, указывающая список таблиц (entities) и версию БД (version)
public abstract class AppDatabase extends RoomDatabase {      //RoomDatabase –
абстрактный базовый класс, предоставляющий доступ к DAO
    public abstract UserDao userDao();
    public abstract CarDao carDao();
}

```

## Листинг 7 – CarDao

```

package com.example.autosalon.data;

import androidx.room.Dao;
import androidx.room.Insert;
import androidx.room.Query;
import androidx.room.Delete;
import androidx.room.Update;

import com.example.autosalon.models.Car;

import java.util.List;

@Dao
public interface CarDao {

    @Insert
    void insert(Car car);

    @Update
    void updateCar(Car car);

    @Query("SELECT * FROM car")
    List<Car> getAllCars();

    @Query("SELECT * FROM car WHERE is_favorite = 1")
    List<Car> getFavoriteCars();
}

```

```

    @Query("SELECT * FROM car WHERE id = :id LIMIT 1")
    Car getCarById(int id);

    @Query("SELECT * FROM car WHERE bought = 1")
    List<Car> getBoughtCars();
}

```

## Листинг 8 – CarData

```

package com.example.autosalon.data;

import com.example.autosalon.models.Car;

import java.util.ArrayList;
import java.util.List;

public class CarData {
    public static List<Car> getCarList() {
        List<Car> cars = new ArrayList<>();

        cars.add(new Car("Mustang GT", "Ford", 2022,
"https://link.to/mustang.jpg", "V8",
"5.0L", "Механика", "Передний", "$55,000"));

        cars.add(new Car("Model S", "Tesla", 2023,
"https://link.to/models.jpg", "Electric",
"-", "Автомат", "Полный", "$85,000"));

        cars.add(new Car("Camry", "Toyota", 2021, "https://link.to/camry.jpg",
"V4",
"2.5L", "Автомат", "Полный", "$30,000"));

        return cars;
    }
}

```

## Листинг 9 – UserDao

```

package com.example.autosalon.data;

import androidx.room.Dao;
import androidx.room.Insert;
import androidx.room.Query;

import com.example.autosalon.models.User;

//Room автоматически генерирует реализацию при сборке проекта
@Dao
public interface UserDao {

    @Insert
    void insert(User user);      // Метод для добавления нового пользователя

    @Query("SELECT * FROM User WHERE username = :username AND password = :password")
    User login(String username, String password);    //Метод для проверки
    данных пользователя (авторизация)

    @Query("SELECT * FROM User WHERE username = :username")
    User getUserByUsername(String username);          //Метод для проверки
    уникальности логина при регистрации
}

```

```
    @Query("SELECT * FROM user WHERE username = :username LIMIT 1")
    User findByLogin(String username);
}
```

## Листинг 10 – ContactsFragment

```
package com.example.autosalon.fragments;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import com.example.autosalon.R;
import com.yandex.mapkit.MapKitFactory;
import com.yandex.mapkit.geometry.Point;
import com.yandex.mapkit.mapview.MapView;

public class ContactsFragment extends Fragment{

    private MapView mapView;
    private static final String MAPVIEW_BUNDLE_KEY="MapViewBundleKey";

    public ContactsFragment() {
        super(R.layout.fragment_contacts);
    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        MapKitFactory.initialize(requireContext());
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_contacts, container,
false);

        Button callButton = view.findViewById(R.id.button_call);
        Button emailButton = view.findViewById(R.id.button_email);

        callButton.setOnClickListener(v -> {
            Intent intent = new Intent(Intent.ACTION_DIAL);
            intent.setData(Uri.parse("tel:+79778718057"));
            startActivity(intent);
        });

        emailButton.setOnClickListener(v -> {
            Intent intent = new Intent(Intent.ACTION_SENDTO);
            intent.setData(Uri.parse("mailto:info@autosalon.ru"));
            intent.putExtra(Intent.EXTRA_SUBJECT, "Вопрос по автомобилям");
            startActivity(intent);
        });
    }
}
```

```

        });

        mapView = view.findViewById(R.id.mapView);

        mapView.getMapWindow().getMap().getMapObjects().addPlacemark(
            new Point(55.7558, 37.6173)
        );

        return view;
    }

    @Override
    public void onStart() {
        super.onStart();
        if (mapView != null) mapView.onStart();
        MapKitFactory.getInstance().onStart();
    }

    @Override
    public void onStop() {
        if (mapView != null) mapView.onStop();
        MapKitFactory.getInstance().onStop();
        super.onStop();
    }
}

```

### Листинг 11 – FavoritesFragment

```

package com.example.autosalon.fragments;

import android.content.Intent;
import android.os.Bundle;

import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.example.autosalon.R;
import com.example.autosalon.activities.DetailsActivity;
import com.example.autosalon.adapters.CarAdapter;
import com.example.autosalon.utils.DatabaseHelper;
import com.example.autosalon.models.Car;

import java.util.List;

public class FavoritesFragment extends Fragment {
    private RecyclerView recyclerView;
    private CarAdapter adapter;
    private List<Car> favorites;

    public FavoritesFragment() {
        super(R.layout.fragment_favorites);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_favorites, container,

```

```

    false);

    recyclerView = view.findViewById(R.id.favoritesRecyclerView);
    recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));

    favorites =
DatabaseHelper.getDatabase(getContext()).carDao().getFavoriteCars();

    adapter = new CarAdapter(getContext(), favorites, car -> {
        Intent intent = new Intent(getContext(), DetailsActivity.class);
        intent.putExtra("car", car);
        startActivity(intent);
    });

    recyclerView.setAdapter(adapter);

    return view;
}

@Override
public void onResume() {
    super.onResume();
    favorites.clear();

favorites.addAll(DatabaseHelper.getDatabase(getContext()).carDao().getFavorite
Cars());
    adapter.notifyDataSetChanged();
}
}

```

## Листинг 12 – HomeFragment

```

package com.example.autosalon.fragments;

import android.content.Intent;
import android.os.Bundle;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.text.Editable;
import android.text.TextWatcher;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;

import com.example.autosalon.R;
import com.example.autosalon.activities.DetailsActivity;
import com.example.autosalon.adapters.CarAdapter;
import com.example.autosalon.data.AppDatabase;
import com.example.autosalon.data.CarData;
import com.example.autosalon.models.Car;
import com.example.autosalon.utils.DatabaseHelper;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;

```

```

import java.util.Set;

public class HomeFragment extends Fragment {
    private RecyclerView recyclerView;
    private CarAdapter adapter;
    private List<Car> carList;
    private List<Car> filteredList;
    private EditText searchInput;
    private Spinner spinnerManufacturer, spinnerGearbox, spinnerDrive;

    public HomeFragment() {}

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_home, container,
false);

        recyclerView = view.findViewById(R.id.carRecyclerView);
        searchInput = view.findViewById(R.id.search_input);
        spinnerManufacturer = view.findViewById(R.id.spinner_manufacturer);
        spinnerGearbox = view.findViewById(R.id.spinner_gearbox);
        spinnerDrive = view.findViewById(R.id.spinner_drive);

        recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));

        AppDatabase db = DatabaseHelper.getDatabase(getContext());
        carList = db.carDao().getAllCars();

        if (carList.isEmpty()) {
            carList = CarData.getCarList();
            for (Car car : carList) {
                db.carDao().insert(car);
            }
            carList = db.carDao().getAllCars();
        }

        filteredList = new ArrayList<>(carList);

        initSpinners();

        adapter = new CarAdapter(getContext(), filteredList, car -> {
            Intent intent = new Intent(getContext(), DetailsActivity.class);
            intent.putExtra("car", car);
            startActivity(intent);
        });
        recyclerView.setAdapter(adapter);

        searchInput.addTextChangedListener(new TextWatcher() {
            @Override public void beforeTextChanged(CharSequence s, int start,
int count, int after) {}
            @Override public void onTextChanged(CharSequence s, int start, int
before, int count) {
                filterAll();
            }
            @Override public void afterTextChanged(Editable s) {}
        });

        return view;
    }

    private void initSpinners() {
        Set<String> manufacturers = new HashSet<>();

```

```

Set<String> gearboxes = new HashSet<>();
Set<String> drives = new HashSet<>();

manufacturers.add("Все марки");
gearboxes.add("Все коробки");
drives.add("Все приводы");

for (Car car : carList) {
    manufacturers.add(car.getManufacturer());
    gearboxes.add(car.getGearBox());
    drives.add(car.getDriveGear());
}

ArrayAdapter<String> adapterMan = new ArrayAdapter<>(requireContext(),
android.R.layout.simple_spinner_dropdown_item, new
ArrayList<>(manufacturers));
ArrayAdapter<String> adapterGear = new
ArrayAdapter<>(requireContext(),
android.R.layout.simple_spinner_dropdown_item, new ArrayList<>(gearboxes));
ArrayAdapter<String> adapterDrive = new
ArrayAdapter<>(requireContext(),
android.R.layout.simple_spinner_dropdown_item, new ArrayList<>(drives));

spinnerManufacturer.setAdapter(adapterMan);
spinnerGearbox.setAdapter(adapterGear);
spinnerDrive.setAdapter(adapterDrive);

AdapterView.OnItemSelectedListener listener = new
AdapterView.OnItemSelectedListener() {
    @Override public void onItemSelected(AdapterView<?> parent, View
view, int position, long id) {
        filterAll();
    }

    @Override public void onNothingSelected(AdapterView<?> parent) {}
};

spinnerManufacturer.setOnItemSelectedListener(listener);
spinnerGearbox.setOnItemSelectedListener(listener);
spinnerDrive.setOnItemSelectedListener(listener);
}

private void filterAll() {
    String search = searchInput.getText().toString().toLowerCase();
    String selectedManufacturer =
spinnerManufacturer.getSelectedItem().toString();
    String selectedGearbox = spinnerGearbox.getSelectedItem().toString();
    String selectedDrive = spinnerDrive.getSelectedItem().toString();

    List<Car> filtered = new ArrayList<>();

    for (Car car : carList) {
        boolean matchesSearch =
car.getModelName().toLowerCase().contains(search);
        boolean matchesManufacturer = selectedManufacturer.equals("Все
марки") || car.getManufacturer().equals(selectedManufacturer);
        boolean matchesGearbox = selectedGearbox.equals("Все коробки") ||
car.getGearBox().equals(selectedGearbox);
        boolean matchesDrive = selectedDrive.equals("Все приводы") ||
car.getDriveGear().equals(selectedDrive);

        if (matchesSearch && matchesManufacturer && matchesGearbox &&
matchesDrive) {
            filtered.add(car);
        }
    }
}

```

```

        }

        adapter.updateList(filtered);
    }

    @Override
    public void onResume() {
        super.onResume();
        carList.clear();

carList.addAll(DatabaseHelper.getDatabase(getContext()).carDao().getAllCars());
;
        filterAll();
    }
}

```

### Листинг 13 – ProfileFragment

```

package com.example.autosalon.fragments;

import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.bumptech.glide.Glide;
import com.example.autosalon.R;
import com.example.autosalon.models.Car;
import com.example.autosalon.models.User;
import com.example.autosalon.utils.DatabaseHelper;

import java.util.List;

public class ProfileFragment extends Fragment {
    private TextView usernameText;

    public ProfileFragment() {
        super(R.layout.fragment_profile);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        TextView usernameText = view.findViewById(R.id.text_username);
        LinearLayout myCarsContainer =
view.findViewById(R.id.container_my_cars);
        LinearLayout statusContainer =
view.findViewById(R.id.container_car_status);
    }
}

```

```

    // Получаем имя пользователя через Room (по текущей сессии)
    SharedPreferences prefs =
requireContext().getSharedPreferences("UserSession", Context.MODE_PRIVATE);
    String login = prefs.getString("loggedInUser", "Гость");
    User currentUser =
DatabaseHelper.getDatabase(requireContext()).userDao().findByLogin(login);
    usernameText.setText("Пользователь: " + (currentUser != null ?
currentUser.getUsername() : "Гость"));

    // Получаем купленные авто через Room
    List<Car> boughtCars =
DatabaseHelper.getDatabase(getContext()).carDao().getBoughtCars();
    LayoutInflater inflater = LayoutInflater.from(getContext());

    for (Car car : boughtCars) {
        View carCard = inflater.inflate(R.layout.item_car_mini,
myCarsContainer, false);

        ImageView image = carCard.findViewById(R.id.img_car);
        TextView title = carCard.findViewById(R.id.txt_model);
        TextView specs = carCard.findViewById(R.id.txt_specs);

        image.setImageResource(getLocalImageResource(car.getModelName()));
        title.setText(car.getModelName());
        specs.setText(car.getEngineType() + " / " + car.getGearBox() + " /
" + car.getDriveGear());

        myCarsContainer.addView(carCard);

        // Состояние авто
        View statusCard = inflater.inflate(R.layout.item_car_status,
statusContainer, false);
        ((TextView)
statusCard.findViewById(R.id.txt_model)).setText(car.getModelName());
        ((TextView)
statusCard.findViewById(R.id.txt_mileage)).setText("Пробег: " +
car.getMileage() + " км");
        ((TextView)
statusCard.findViewById(R.id.txt_fuel)).setText("Топливо: " +
car.getFuelLevel() + "% (" + car.getApproxKmLeft() + " км)");
        ((TextView) statusCard.findViewById(R.id.txt_oil)).setText("Масло:
" + (car.needsOilChange() ? "нужна замена" : "в норме"));

        statusContainer.addView(statusCard);

        Log.d("PROFILE", "Купленное авто: " + car.getModelName());
    }
}

private int getLocalImageResource(String modelName) {
    switch (modelName.toLowerCase()) {
        case "camry":
            return R.drawable.camry;
        case "bmw 3 series":
            return R.drawable.bmw3;
        case "kia rio":
            return R.drawable.rio;
        case "hyundai solaris":
            return R.drawable.solaris;
        case "audi a4":
            return R.drawable.audi_a4;
        case "tesla model 3":
            return R.drawable.tesla3;
        case "lada vesta":
            return R.drawable.lada;
    }
}

```

```

        case "mustang gt":
            return R.drawable.mustang;
        case "model s":
            return R.drawable.teslas;
        default:
            return R.drawable.placeholder;
    }
}

```

### Листинг 14 – Car (базовый класс)

```

package com.example.autosalon.models;

import androidx.room.ColumnInfo;
import androidx.room.Entity;
import androidx.room.PrimaryKey;

import com.google.firebaseio.database.PropertyName;
import com.google.gson.annotations.SerializedName;

import java.io.Serializable;

@Entity(tableName = "car")
public class Car implements Serializable {
    @PrimaryKey(autoGenerate = true)
    private int id;

    @ColumnInfo(name = "is_favorite")
    private boolean isFavorite;

    private String modelName;
    private String manufacturer;
    private int year;
    private String imageUrl;
    private String engineType;
    private String engineVolume;
    private String gearBox;
    private String driveGear;
    private String price;
    private int mileage;
    private int fuellLevel;
    private int approxKmLeft;
    @ColumnInfo(name = "bought")
    @PropertyName("bought")
    private boolean isBought;

    public boolean needsOilChange() {
        return mileage > 10000;
    }

    // конструкторы
    public Car() {}

    public Car(String modelName, String manufacturer, int year, String
imageUrl, String engineType,
            String engineVolume, String gearBox, String driveGear, String
price) {
        this.modelName = modelName;
        this.manufacturer = manufacturer;
        this.year = year;
        this.imageUrl = imageUrl;
    }
}

```

```

        this.engineType = engineType;
        this.engineVolume = engineVolume;
        this.gearBox = gearBox;
        this.driveGear = driveGear;
        this.price = price;
    }

    // геттеры
    public String getModelName() {
        return this.modelName;
    }
    public String getManufacturer() {
        return this.manufacturer;
    }
    public int getYear() {
        return this.year;
    }
    public String getImageUrl() {
        return this.imageUrl;
    }
    public String getEngineType() {
        return this.engineType;
    }
    public String getEngineVolume() {
        return this.engineVolume;
    }
    public String getGearBox() {
        return this.gearBox;
    }
    public String getDriveGear() {
        return this.driveGear;
    }
    public String getPrice() {
        return this.price;
    }
    public boolean isFavorite() {return isFavorite;}
    public int getId() {return this.id;}
    public int getMileage() {return this.mileage;}
    public int getFuelLevel() {return this.fuelLevel;}
    public int getApproxKmLeft() {return this.approxKmLeft;}
    @PropertyName("bought")
    public boolean isBought() {return this.isBought;}

    // сеттеры

    public void setModelName(String modelName) {
        this.modelName = modelName;
    }

    public void setManufacturer(String manufacturer) {
        this.manufacturer = manufacturer;
    }

    public void setYear(int year) {
        this.year = year;
    }

    public void setImageUrl(String imageUrl) {
        this.imageUrl = imageUrl;
    }

    public void setEngineType(String engineType) {
        this.engineType = engineType;
    }
}

```

```

public void setEngineVolume(String engineVolume) {
    this.engineVolume = engineVolume;
}

public void setGearBox(String gearBox) {
    this.gearBox = gearBox;
}

public void setDriveGear(String driveGear) {
    this.driveGear = driveGear;
}

public void setPrice(String price) {
    this.price = price;
}

public void setFavorite(boolean favorite) {isFavorite = favorite; }

public void setId(int id) {this.id = id;}
public void setMileage(int mileage) {this.mileage = mileage;}
public void setFuelLevel(int fuelLevel) {this.fuelLevel = fuelLevel;}
public void setApproxKmLeft(int approxKmLeft) {this.approxKmLeft =
approxKmLeft;}
@PropertyName("bought")
public void setBought(boolean bought) {this.isBought = bought;}
}

```

### Листинг 15 – User (базовый класс)

```

package com.example.autosalon.models;

import androidx.room.Entity;
import androidx.room.PrimaryKey;

@Entity
public class User {

    @PrimaryKey(autoGenerate = true)
    public int id;

    public String username;
    public String password;

    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public String getUsername() {return this.username;}
    public void setUsername(String username) {this.username = username;}
}

```

### Листинг 16 – activity\_details.xml

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"

```

```
    android:layout_height="match_parent"
    android:fillViewport="true">
<LinearLayout
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:background="#FFFFFF">

    <ImageView
        android:id="@+id/detail_image"
        android:layout_width="match_parent"
        android:layout_height="220dp"
        android:scaleType="centerCrop"
        android:background="#EEEEEE" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:background="#F5F5F5"
        android:padding="16dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/detail_model"
            android:text="Модель"
            android:textSize="20sp"
            android:textStyle="bold"
            android:layout_marginBottom="8dp" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/detail_price"
            android:text="Цена"
            android:textSize="18sp"
            android:textColor="#000000"
            android:layout_marginBottom="12dp" />

        <TextView android:id="@+id/detail_manufacturer"
            android:text="Производитель"
            android:layout_marginBottom="4dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>

        <TextView android:id="@+id/detail_year"
            android:text="Год выпуска"
            android:layout_marginBottom="4dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>

        <TextView android:id="@+id/detail_engine"
            android:text="Тип двигателя"
            android:layout_marginBottom="4dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>

        <TextView android:id="@+id/detail_volume"
            android:text="Объём двигателя"
            android:layout_marginBottom="4dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
```

```
<TextView android:id="@+id/detail_gearbox"
    android:text="Коробка"
    android:layout_marginBottom="4dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>

<TextView android:id="@+id/detail_drive"
    android:text="Привод"
    android:layout_marginBottom="4dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
</LinearLayout>

<!-- Нижний блок с кнопками -->
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="12dp"
    android:gravity="center"
    android:layout_marginTop="16dp">

    <Button
        android:id="@+id/button_call"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="48dp"
        android:text="Позвонить"
        android:backgroundTint="#2979FF"
        android:textColor="#FFFFFF"
        android:layout_marginEnd="8dp"/>

    <ImageButton
        android:id="@+id/button_message"
        android:layout_width="48dp"
        android:layout_height="48dp"
        android:src="@drawable/ic_message"
        android:contentDescription="Написать"
        android:backgroundTint="#19C94E"
        android:padding="8dp"/>
</LinearLayout>

<TextView
    android:id="@+id/text_similar_title"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Похожие авто"
    android:textSize="18sp"
    android:textStyle="bold"
    android:padding="16dp"
    android:textColor="#000000" />

<LinearLayout
    android:id="@+id/container_similar_cars"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="8dp"
    android:background="#F5F5F5" />
</LinearLayout>
</ScrollView>
```

## Листинг 17 – activity\_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:padding="24dp"
    android:gravity="center"
    android:orientation="vertical">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:id="@+id/et_login_username"
        android:hint="Логин"
        android:inputType="textPersonName"/>

    <EditText
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:id="@+id/et_login_password"
        android:hint="Пароль"
        android:inputType="textPassword"
        android:layout_marginTop="16dp"/>

    <Button
        android:layout_width="250dp"
        android:layout_height="50dp"
        android:id="@+id/btn_login"
        android:text="Войти"
        android:layout_marginTop="24dp"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/tv_register_link"
        android:text="Нет аккаунта? Зарегистрироваться"
        android:textSize="14sp"
        android:clickable="true"
        android:focusable="true"
        android:layout_marginTop="24dp"/>
</LinearLayout>
```

## Листинг 18 – activity\_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <androidx.fragment.app.FragmentContainerView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/nav_host_fragment"
        android:name="androidx.fragment.app.Fragment"/>

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottom_navigation"
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom"
    app:menu="@menu/bottom_nav_menu"/>

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

### Листинг 19 – activity\_register.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="24dp"
    android:gravity="center">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:id="@+id/et_username"
        android:hint="Логин"
        android:inputType="textPersonName"/>

    <EditText
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:id="@+id/et_password"
        android:hint="Пароль"
        android:inputType="textPassword"
        android:layout_marginTop="16dp"/>

    <Button
        android:layout_width="250dp"
        android:layout_height="50dp"
        android:text="Зарегистрироваться"
        android:layout_marginTop="24dp"
        android:id="@+id/btn_register"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/tv_login_link"
        android:text="Уже зарегистрированы? Войти"
        android:layout_marginTop="16dp"
        android:textSize="14sp"
        android:clickable="true"
        android:focusable="true"/>
</LinearLayout>
```

### Листинг 20 – activity\_details.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior">
```

```
<fragment
    android:id="@+id/nav_host_fragment_content_details"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:defaultNavHost="true"
    app:navGraph="@navigation/nav_graph" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

## Листинг 21 – fragment\_contacts.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/contact_title"
        android:text="Наш автосалон"
        android:textSize="24sp"
        android:textStyle="bold"
        android:gravity="center"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/contact_adress"
        android:text="ул. Примерная, д. 123"
        android:textSize="16sp"
        android:layout_marginTop="12dp"
        android:gravity="center"/>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/button_call"
        android:text="Позвонить нам"
        android:layout_marginTop="8dp"/>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/button_email"
        android:text="Написать на email"
        android:layout_marginTop="8dp"/>

    <com.yandex.mapkit.mapview.MapView
        android:id="@+id/mapView"
        android:layout_width="match_parent"
        android:layout_height="200dp" />
</LinearLayout>
```

## Листинг 22 – fragments\_favorites.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.recyclerview.widget.RecyclerView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/favoritesRecyclerView"
        android:padding="8dp"
        android:clipToPadding="false"/>

</FrameLayout>
```

## Листинг 23 – fragments\_home.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="8dp">

    <EditText
        android:id="@+id/search_input"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Поиск по названию..."
        android:inputType="text"
        android:drawableStart="@android:drawable/ic_menu_search"
        android:backgroundTint="@color/black"
        android:padding="10dp"
        android:layout_marginBottom="8dp"/>

    <!-- Фильтры -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="8dp">

    <!-- Спиннеры -->
    <Spinner
        android:id="@+id/spinner_manufacturer"
        android:layout_width="0dp"
        android:layout_height="48dp"
        android:layout_weight="1"/>

    <Spinner
        android:id="@+id/spinner_gearbox"
        android:layout_width="0dp"
        android:layout_height="48dp"
        android:layout_weight="1"
        android:layout_marginStart="8dp"/>

    <Spinner
        android:id="@+id/spinner_drive"
```

```

        android:layout_width="0dp"
        android:layout_height="48dp"
        android:layout_weight="1"
        android:layout_marginStart="8dp"/>/>
    
```

```

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/carRecyclerView"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:clipToPadding="false"
    android:contentDescription="Список автомобилей"/>

```

```

</LinearLayout>

```

## Листинг 24 –fragments\_profile.xml

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:padding="16dp"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:id="@+id/text_username"
            android:text="Пользователь: login123"
            android:textStyle="bold"
            android:textSize="18sp"
            android:layout_marginBottom="12dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <!-- Блок: Мои авто -->
        <TextView
            android:text="Мои авто"
            android:textSize="16sp"
            android:textStyle="bold"
            android:layout_marginTop="8dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

        <LinearLayout
            android:id="@+id/container_my_cars"
            android:orientation="vertical"
            android:background="#F5F5F5"
            android:padding="8dp"
            android:layout_marginBottom="12dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

        <!-- Блок: Состояние авто -->
        <TextView
            android:text="Состояние авто"
            android:textSize="16sp"

```

```

        android:textStyle="bold"
        android:layout_marginTop="8dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <LinearLayout
        android:id="@+id/container_car_status"
        android:orientation="vertical"
        android:background="#F5F5F5"
        android:padding="8dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    </LinearLayout>
</ScrollView>

```

## Листинг 25 – item\_car.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    card_view:cardCornerRadius="16dp"
    card_view:cardElevation="6dp">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#f0f0f0">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="200dp">

            <ImageView
                android:id="@+id/car_image"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:scaleType="centerCrop" />

            <ImageView
                android:id="@+id/fav_icon"
                android:layout_width="32dp"
                android:layout_height="32dp"
                android:layout_alignParentEnd="true"
                android:layout_margin="12dp"
                android:src="@drawable/ic_favorite_border"
                android:contentDescription="Favorite icon" />

        </RelativeLayout>

        <LinearLayout
            android:orientation="vertical"
            android:padding="12dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

```

```

<TextView
    android:id="@+id/car_price"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="4 500 000 ₽"
    android:textSize="18sp"
    android:textStyle="bold" />

<TextView
    android:id="@+id/car_model"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Mercedes E-class"
    android:textSize="16sp"
    android:textStyle="normal"
    android:layout_marginTop="4dp" />

<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:layout_marginTop="8dp">

    <Button
        android:id="@+id/button_call"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="48dp"
        android:text="Позвонить"
        android:backgroundTint="#2768F5"
        android:textColor="@android:color/white" />

    <ImageButton
        android:id="@+id/button_message"
        android:layout_width="48dp"
        android:layout_height="48dp"
        android:layout_marginStart="12dp"
        android:backgroundTint="#19C94E"
        android:foregroundTint="#19C94E"
        android:src="@drawable/ic_message"
        android:contentDescription="Написать сообщение" />

```

</LinearLayout>

</LinearLayout>

</LinearLayout>

</androidx.cardview.widget.CardView>

## Листинг 26 – item\_car\_mini.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#EEEEEE"
    android:orientation="vertical"
    android:padding="12dp"
    android:layout_marginBottom="12dp">

    <ImageView
        android:id="@+id/img_car"
        android:layout_width="match_parent"

```

```

        android:layout_height="180dp"
        android:scaleType="centerCrop"
        android:contentDescription="Изображение автомобиля" />

    <TextView
        android:id="@+id/txt_model"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Модель"
        android:textStyle="bold"
        android:textSize="16sp"
        android:layout_marginTop="8dp" />

    <TextView
        android:id="@+id/txt_specs"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Характеристики"
        android:textSize="14sp"
        android:layout_marginTop="4dp" />

</LinearLayout>

```

### Листинг 27 – item\_car\_status.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:background="#F0F0F0"
    android:padding="12dp"
    android:layout_marginBottom="12dp">

    <TextView
        android:id="@+id/txt_model"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Модель"
        android:textStyle="bold"
        android:textSize="16sp" />

    <TextView
        android:id="@+id/txt_mileage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Пробег: 0 км"
        android:layout_marginTop="6dp" />

    <TextView
        android:id="@+id/txt_fuel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Топливо: 0% (0 км)" />

    <TextView
        android:id="@+id/txt_oil"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Масло: в норме" />

</LinearLayout>

```

## Листинг 28 – bottom\_nav\_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/nav_home"
        android:title="Каталог"
        android:icon="@drawable/ic_home"/>

    <item
        android:id="@+id/navFavorites"
        android:title="Избранное"
        android:icon="@drawable/ic_favorites" />

    <item
        android:id="@+id/nav_contacts"
        android:title="Контакты"
        android:icon="@drawable/ic_info" />

    <item
        android:id="@+id/nav_profile"
        android:title="Профиль"
        android:icon="@drawable/ic_profile"/>
</menu>
```

## ЗАКЛЮЧЕНИЕ

В рамках курсовой работы было разработано и протестировано мобильное Android-приложение справочного назначения для автосалона. Оно предоставляет пользователю удобный доступ к каталогу автомобилей с возможностью фильтрации, просмотра характеристик, добавления моделей в избранное и получения контактной информации об офисе.

Реализованы регистрация и авторизация пользователей, сохранение сессий, хранение и отображение купленных автомобилей, а также фильтрация и сортировка моделей по заданным параметрам. Архитектура построена по принципу "Single Activity — Multiple Fragments", что обеспечивает модульность и удобство сопровождения проекта.

Для разработки использованы Android Studio, язык Java, библиотека Room (локальная база данных), Firebase Realtime Database, SharedPreferences (хранение сессий), а также UI-компоненты Android SDK. Визуальное оформление реализовано с использованием XML и drawable-ресурсов.

Проведённое тестирование подтвердило стабильную работу приложения. Продукт может быть использован в учебных целях или доработан под задачи реального автосалона. Работа позволила закрепить навыки мобильной разработки и проектирования пользовательских интерфейсов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Спинов А. Р., Бокарев Д. Р. Бизнес-процесс продажи автомобилей: учебно-методическое пособие [Электронный ресурс]. – МАДИ. – URL: <https://lib.madi.ru/fel/fel1/fel24E671.pdf> (дата обращения: 01.06.2025).
2. IDEF0: знакомство с нотацией и пример использования [Электронный ресурс]. – URL: <https://www.trinion.org/blog/idef0-znakomstvo-s-notaciey-i-primer-ispolzovaniya> (дата обращения: 02.06.2025).
3. DFD (Data Flow Diagram) Диаграммы – зачем они нужны и какие бывают [Электронный ресурс] / Хабр – URL: <https://habr.com/ru/articles/668684/> (дата обращения: 02.06.2025).
4. Что такое DFD (диаграммы потоков данных) [Электронный ресурс]. – URL: <https://www.trinion.org/blog/chto-takoe-dfd-diagrammy-potokov-danniykh> (дата обращения: 02.06.2025).
5. Фрагменты [Электронный ресурс] – URL: <https://metanit.com/java/android/8.1.php> (дата обращения 04.06.2025)
6. Основы создания интерфейса Android Studio [Электронный ресурс] – URL: <https://metanit.com/java/android/3.1.php> (дата обращения: 04.06.2025).
7. Сохраните данные в локальной базе данных с помощью Room [Электронный ресурс] – URL: <https://developer.android.com/training/data-storage/room?hl=ru> (дата обращения 04.06.2025).

