МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

## федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет им. Г.В. Плеханова» Московский приборостроительный техникум

Курсовой проект

ПМ 01 Разработка модулей программного обеспечения для компьютерных систем

МДК 01.01 Разработка программных модулей

## Специальность 09.02.07 «Информационные системы и программирование» Квалификация: Программист

Тема: «Разработка мобильного приложения для продажи и покупки автомобилей».

Пояснительная записка

Листов: 230

Руководитель

/ М.А. Горбунова

« » 2025 год

Исполнитель

/И.А. Пушкин

« » 2025 год 2025

Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет имени Г.В. Плеханова»

**Московский приборостроительный техникум**

«Утверждаю»

Заместитель директора по учебной работе

Д.А. Клопов

« » 2025 г.

**Задание**

на выполнение курсового проекта (курсовой работы)

## Пушкиным Ильей Александровичем

(фамилия, имя, отчество студента — полностью)

студенту группы П50-3-22 специальности 09.02.07 «Информационные системы и программирование» по МДК 01.01 «Разработка программных модулей»

1. Исходные данные к проекту (работе):
   1. Тема: «Разработка мобильного приложения для продажи и покупки автомобилей».
   2. Состав курсового проекта:
      1. Задание КП
      2. Пояснительная записка
      3. Программа (исходные данные) на электронном носителе
      4. Презентация и инсталляционный пакет программы на электронном носителе
   3. Содержание пояснительной записки: ВВЕДЕНИЕ
2. ОБЩАЯ ЧАСТЬ
   1. Цель разработки
   2. Средства разработки
3. СПЕЦИАЛЬНАЯ ЧАСТЬ
   1. Постановка задачи
      1. Входные данные
      2. Выходные данные
      3. Подробные требования к проекту
   2. Внешняя спецификация
      1. Описание задачи
      2. Входные и выходные данные
      3. Методы
      4. Тесты
      5. Контроль целостности данных
   3. Проектирование
      1. Схема архитектуры приложения
      2. Логическая схема данных
      3. Физическая схема данных
      4. Структурная схема
      5. Функциональная схема
      6. Диаграмма классов
      7. Схема тестирования
      8. Схема пользовательского интерфейса
   4. Результат работы программы
4. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ
   1. Инструментальные средства
   2. Отладка программы
   3. Защитное программирование
   4. Характеристики программы ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ

ПРИЛОЖЕНИЕ А. Техническое задание ПРИЛОЖЕНИЕ Б. Текст программы

ПРИЛОЖЕНИЕ В. Сценарий и результаты тестовых испытаний ПРИЛОЖЕНИЕ Г. Руководство пользователя

1. Содержание задания по проекту (работе) ⁠— перечень вопросов, подлежащих разработке

|  |  |  |  |
| --- | --- | --- | --- |
|  | Разрабатываемый вопрос | Объем от всего задания,  % | Срок выполнения |
| А | Описательная часть проекта (введение, общее описание и т. д.) | 5 | 10.02.2025 |
| 1. | Введение | - | 04.02.2025 |
| 2. | Цель разработки | - | 07.02.2025 |
| 3. | Средства разработки | - | 10.02.2025 |
| Б | Анализ задачи и её постановка | 15 | 15.02.2025 |
| 1. | Определение требований к программе | - | 14.02.2025 |
| 2. | Спецификация программы (описание задачи, описание входных и выходных данных, методы) | - | 14.02.2025 |
| 3. | Тесты, контроль целостности данных | - | 15.02.2025 |
| В | Проектирование и реализация | 55 | 16.03.2025 |
| 1. | Схемы проекта (схема архитектуры, функциональная и структурная схемы, диаграмма классов, схема тестирования, схема пользовательского интерфейса) | - | 15.02.2025 |
| 2. | Реализация в инструментальной среде | - | 16.03.2025 |
| Г | Технологическая часть проекта | 5 | 28.03.2025 |
| 1. | Инструментальные средства разработки | - | 20.03.2025 |
| 2. | Отладка программа | - | 26.03.2025 |
| 3. | Защитное программирование | - | 26.03.2025 |
| 4. | Характеристика программы | - | 28.03.2025 |
| Д | Программная документация | 10 | 14.04.2025 |
| 1. | Приложение А. Техническое задание | - | 11.04.2025 |
| 2. | Приложение Б. Текст программы | - | 12.04.2025 |
| 3. | Приложение В. Сценарий и результаты тестовых испытаний | - | 12.04.2025 |
|  | Приложение Г. Руководство пользователя |  | 14.04.2025 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Разрабатываемый вопрос | Объем от всего задания,  % | Срок выполнения |
| Е | Экспериментальная часть проекта | 10 | 15.04.2025 |
| 1. | Программа на машинном носителе.  Информация на носителе разбита на разделы: эксплуатационный пакет, тексты программы, документация. | - | 15.04.2025 |

Руководитель курсового проекта (работы) Горбунова Мария Александровна, преподаватель

«22» февраля 2025 года / М.А. Горбунова/

Дата выдачи курсового задания «04» февраля 2025 года Срок сдачи законченного проекта (работы) «22» апреля 2025 года

Задание принял к исполнению

«04» февраля 2025 года / И.А. Пушкин /

СОДЕРЖАНИЕ

[Введение 4](#_bookmark0)

1. [Общая часть 6](#_bookmark1)
   1. [Цель разработки 6](#_bookmark2)
   2. [Средства разработки 6](#_bookmark3)
2. [Средства разработки 8](#_bookmark4)
   1. [Постановка задачи 8](#_bookmark5)
      1. [Входные данные 8](#_bookmark6)
      2. [Выходные данные предметной области 8](#_bookmark7)
      3. [Подробные требования к проекту 9](#_bookmark8)
   2. [Внешняя спецификация 10](#_bookmark9)
      1. [Описание задачи 10](#_bookmark10)
      2. [Входные и выходные данные 21](#_bookmark11)
      3. [Методы 25](#_bookmark12)
      4. [Тесты 25](#_bookmark13)
      5. [Контроль целостности данных 26](#_bookmark14)
   3. [Проектирование 27](#_bookmark15)
      1. [Схема архитектуры приложения 27](#_bookmark16)
      2. [Логическая схема данных 27](#_bookmark17)
      3. [Физическая схема данных 27](#_bookmark18)
      4. [Структурная схема 27](#_bookmark19)
      5. [Функциональная схема 28](#_bookmark20)
      6. [Диаграмма классов 29](#_bookmark21)
      7. [Схема тестирования 32](#_bookmark22)
      8. [Схема пользовательского интерфейса 33](#_bookmark23)
   4. [Результат работы программы 34](#_bookmark24)
3. [Технологическая часть 41](#_bookmark25)
   1. [Инструментальные средства 41](#_bookmark26)
   2. [Отладка программы 41](#_bookmark27)
   3. [Защитное программирование 42](#_bookmark28)
   4. [Характеристики программы 44](#_bookmark29)

[Заключение 58](#_bookmark30)

[Список использованных материалов 60](#_bookmark31)

## ВВЕДЕНИЕ

В условиях активного развития информационных технологий и роста популярности мобильных сервисов пользователи всё чаще отдают предпочтение мобильным приложениям для решения повседневных задач, включая покупку, продажу и управление личной информацией. Создание удобных и многофункциональных мобильных платформ стало важной задачей для улучшения пользовательского опыта и расширения возможностей цифровой коммерции. Мобильные приложения объединяют продавцов, покупателей и администраторов в едином пространстве, обеспечивая быстрое и интуитивно понятное взаимодействие. Разработка подобного приложения позволяет значительно повысить доступность сервисов и упростить бизнес- процессы.

Мобильное приложение «Drivee» представляет собой современную платформу для Android-устройств, предназначенную для продажи, покупки автомобилей, а также управления личными объявлениями и записями в бортжурнале. Система объединяет частных продавцов, покупателей и администраторов, создавая удобную среду для поиска транспортных средств, подачи объявлений и проведения модерации. В отличие от традиционных интернет-площадок, приложение «Drivee» оптимизировано для быстрого взаимодействия, что делает процесс выбора автомобиля простым, быстрым и удобным для пользователя.

Приложение ориентировано на пользователей, активно использующих мобильные устройства для повседневных задач. Частные лица могут легко находить интересующие их автомобили, применять расширенные фильтры поиска, добавлять объявления в избранное, а продавцы — публиковать и редактировать свои предложения, управлять личным профилем и следить за состоянием своих объявлений. Администраторы получают возможность модерировать контент, управлять марками и моделями автомобилей, а также контролировать записи бортжурнала, что делает платформу эффективной и безопасной для всех участников.

В рамках курсового проекта будет разработано мобильное приложение

«Drivee» в соответствии с техническим заданием (Приложение А. Техническое задание). Целью разработки является создание комплексной системы, включающей основные модули: регистрация и авторизация пользователей, управление личным профилем, поиск и фильтрацию объявлений, управление избранным, публикацию и редактирование объявлений, работу с бортжурналом и администрирование контента. Эти компоненты обеспечат полный цикл взаимодействия пользователя с платформой.

Приложение «Drivee» значительно упростит процесс поиска, продажи и покупки автомобилей, предоставив пользователям удобный, интуитивный интерфейс и надежные инструменты для взаимодействия. Платформа оптимизирует рабочие процессы продавцов, облегчает покупку автомобилей для частных лиц и обеспечивает эффективную модерацию контента администраторами, повышая общее качество предоставляемых услуг и удовлетворенность пользователей.

## ОБЩАЯ ЧАСТЬ

## Цель разработки

Создание функционального мобильного приложения «Drivee», которое обеспечит удобное взаимодействие между покупателями и продавцами в сфере торговли автомобилями. Проект направлен на повышение доступности автомобильной продукции, улучшение пользовательского опыта и оптимизацию процессов купли-продажи для всех участников платформы.

## Средства разработки

В качестве программных средств разработки использовалось ПО, представленное в Таблице 1 (см. Таблица 1 – Программные средства разработки ПО)

Таблица 1 – Программные средства разработки ПО

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тип средства | Наименование средства | Назначение |
| 1 | 2 | 3 | 4 |
| 1 | Операционная система | MacOS 15.3.2 (24D81) | Организация взаимодействия программ  и пользователя |
| 2 | Инструментальные  средства разработки программного решения | Android Studio | Разработка мобильного приложения |
| 3 | Облачная база данных | Google Firestore Database | Хранение и управление данными мобильного  приложения в облаке |
| 5 | Сервис электронной почты | Firebase | Отправка писем для восстановления пароля и регистрации пользователей, а также для  получения чека заказа |
| 6 | Текстовый редактор | Microsoft® Word для Mac | Написание документации |
| 7 | Графический построитель | Pixso | Создание эскиза визуальной части  мобильного приложения |

## В качестве средств вычислительной техники для проектирования, разработки и тестирования программного комплекса использовались ноутбук MacBook air 15 M3, а также мобильное устройство Xiaomi 4 note. Характеристики технических средств представлены в Таблице 2 (см. Таблица 2 – Характеристики вычислительной техники).

Таблица 2 - Характеристики вычислительной техники

|  |  |  |
| --- | --- | --- |
| № | Характеристика | Значение |
| 1 | 2 | 3 |
| MacBook Air 15 M3 | | |
| 1 | Разрешение экрана | 2880 × 1864 |
| 2 | Видеокарта | Apple M3 |
| 3 | Размер экрана | 15.3 " дюйма |
| 4 | Оперативная память | 16 ГБ |
| 5 | Графический процессор | Apple M3 |
| Xiaomi 4 note | | |
| 1 | Дисплей | 7.5" (2430x1080), Full HD,  Super AMOLED |
| 2 | Встроенная память | 128 ГБ |
| 3 | Оперативная память | 4 ГБ |
| 4 | Процессор | Snapdragon 8g |
| 5 | Количество ядер процессора | 6 |

## СРЕДСТВА РАЗРАБОТКИ

## Постановка задачи

Целью данного проекта является разработка мобильного приложения

«Drivee» для платформы Android, предназначенного для автоматизации процессов публикации, поиска и продажи автомобилей. Приложение должно обеспечивать удобное взаимодействие между покупателями, продавцами и администраторами, предоставляя функционал для частной и коммерческой продажи автомобилей. Оно ориентировано на частных пользователей, автосалоны и сервисные компании, объединяя их в единой экосистеме.

## Входные данные

Мобильное приложение «Drivee» содержит следующие входные данные: информация о пользователях (имя, дата рождения, номер телефона, электронная почта, пароль, фото профиля), автомобилях (марка, модель, год выпуска, цвет, тип кузова, объем двигателя, мощность, тип привода, тип топлива, трансмиссия, пробег, цена, описание, фотографии автомобиля), объявлениях (статус модерации, дата публикации), данных бортжурнала (марка и модель автомобиля, заголовок записи, текст записи, пробег) и параметрах фильтрации поиска (бренд, модель, цена, пробег, город, год выпуска и прочее).

## Выходные данные предметной области

Мобильное приложение «Drivee» содержит следующие выходные данные: о пользователях (информация о профиле: имя, дата рождения, номер телефона, электронная почта, пароль, фотография профиля), об автомобилях (марка, модель, год выпуска, цвет, тип кузова, объем двигателя, мощность, пробег, тип привода, тип топлива, трансмиссия, цена, описание, фотографии автомобилей), об объявлениях (статус модерации: на проверке, подтверждено; дата публикации, идентификатор объявления), о записах в бортжурнале (марка и модель автомобиля, заголовок записи, текст записи, пробег автомобиля на момент записи), о фильтрации объявлений (список

отфильтрованных автомобилей по марке, модели, цене, пробегу, городу, цвету, типу кузова, приводу, типу топлива, трансмиссии).

Также приложение формирует информацию о действиях пользователя, таких как добавление/удаление машины в избранное, удаление или изменение объявлений, изменения профиля пользователя.

## Подробные требования к проекту

Разработать мобильное приложение «Drivee» для платформы Android с использованием облачной СУБД Firebase (Firestore) для хранения и обработки данных. Приложение должно обеспечивать пользователям возможность просматривать объявления о продаже автомобилей, использовать систему фильтрации по различным параметрам и управлять личными данными и объявлениями.

Для доступа ко всем функциям приложения требуется авторизация. Регистрация новых пользователей производится с указанием имени, даты рождения, номера телефона, email и пароля (пароль должен содержать не менее 8 символов, включая строчные, заглавные буквы и цифры). В случае утери пароля предусмотрена возможность его восстановления через email с использованием SMTP-сервера.

Интерфейс приложения должен быть интуитивно понятным, с удобной навигацией через нижнюю панель (главная, избранное, создать объявление, бортжурнал, профиль), обеспечивая пользователю быстрый доступ к основным функциям. Приложение «Drivee» должно поддерживать следующие функциональные возможности:

* + - Возможность авторизации и регистрации пользователей. Возможность восстановления пароля через SMTP.
    - Возможность изменения данных профиля (фото профиля, имя, телефон, email).
    - Возможность создания объявления о продаже автомобиля. Возможность загрузки фотографий автомобиля (до 5 штук).
    - Возможность выбора параметров автомобиля (марка, модель, год, объем двигателя, цвет, мощность, пробег, тип привода, топливо, трансмиссия, цена, описание).
    - Возможность редактирования созданного объявления. Возможность удаления своего объявления.
    - Возможность просмотра всех объявлений.
    - Возможность применения фильтров поиска по марке, модели, году, цене, объему двигателя, мощности, количеству владельцев, цвету, кузову, городу, приводу, типу топлива, типу трансмиссии.
    - Возможность добавления автомобиля в избранное. Возможность удаления автомобиля из избранного.
    - Возможность создания записи в бортжурнале по автомобилю (марка, модель, заголовок записи, текст записи, пробег).
    - Возможность редактирования и удаления записей бортжурнала. Возможность просмотра записей бортжурнала в ленте.
    - Возможность выхода из аккаунта пользователя.
    - Возможность работы администратора для модерирования объявлений и записей бортжурнала.
    - Возможность подтверждения или удаления объявлений администратором.
    - Возможность управления списком марок и моделей автомобилей администратором (добавление новых марок и моделей).

## Внешняя спецификация

## Описание задачи

Разработать мобильное приложение для цифровизации процесса покупки, продажи и администрирования объявлений в сфере автомобильной торговли. Решение включает использование облачной базы данных, управляемой СУБД Firebase (Firestore), и разработку мобильного приложения на платформе Android через Android Studio.

Приложение должно обеспечивать не только отображение данных, но и предоставлять пользователям возможности регистрации, авторизации, восстановления пароля, создания, редактирования и удаления объявлений, управления личными данными, а также работы с избранными объявлениями и бортжурналом.

На этапе анализа были определены основные пользовательские роли: неавторизованный пользователь, покупатель и администратор, а также описан их функционал в системе: регистрация и вход в систему, сброс пароля, просмотр объявлений, фильтрация по параметрам, добавление автомобилей в избранное, создание и управление объявлениями, публикация записей в бортжурнале, модерирование контента (объявлений и записей) администраторами, управление марками и моделями автомобилей.

На Рисунке 1 показана диаграмма прецендентов.

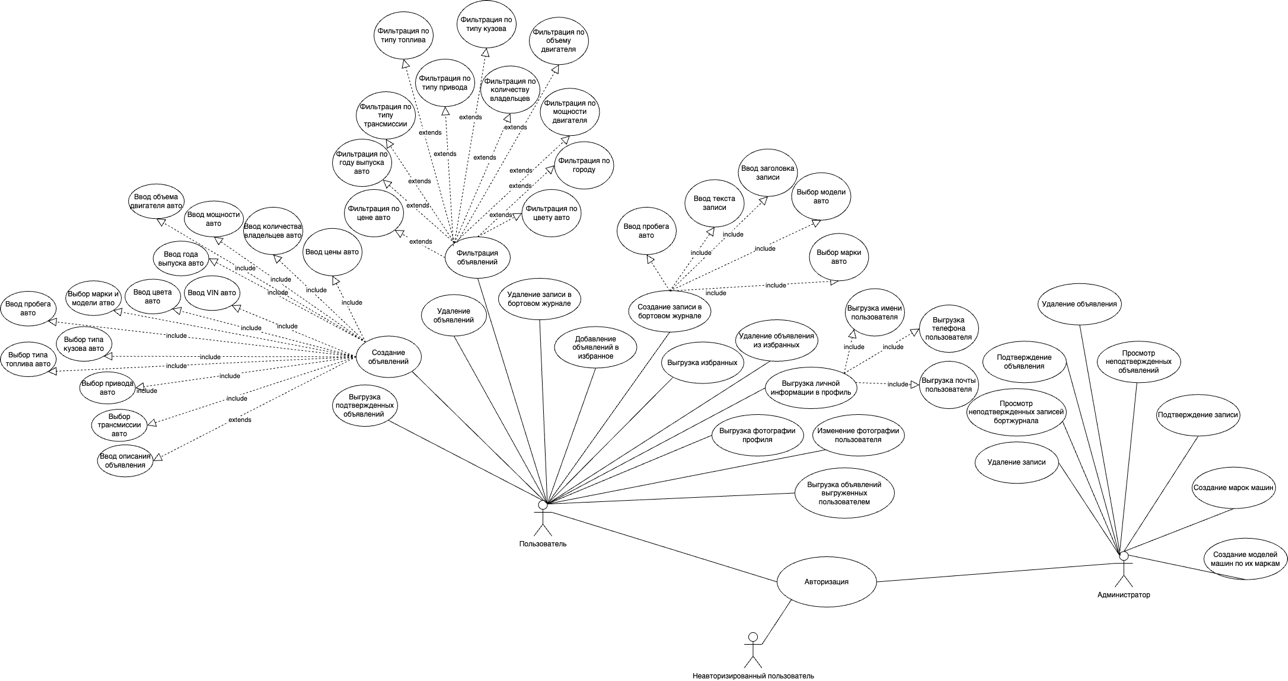


Рисунок 1 - Диаграмма прецедентов

На основе созданной диаграммы прецедентов можно рассмотреть возможности каждого пользователя:

Пользователь с ролью «Неавторизованный пользователь» имеет следующие возможности:

Регистрация в качестве пользователя с заполнением следующих полей: почта, имя, дата рождения, номер телефона, пароль, подтверждение пароля.

Авторизация в системе с заполнением полей: почта и пароль.

Восстановление пароля через отправку ссылки для сброса пароля на почту через SMTP.

Пользователь с ролью «Обычный пользователь» имеет следующие возможности:

Авторизация через заполнение полей: почта и пароль.

Восстановление пароля через отправку ссылки для сброса пароля на почту через SMTP.

Просмотр списка всех доступных объявлений о продаже автомобилей в виде карточек (фото, марка, модель, цена, пробег).

Применение фильтрации объявлений по марке, модели, цене, пробегу и дополнительным параметрам.

Сортировка объявлений по цене (по возрастанию/убыванию) и новизне. Добавление и удаление автомобилей в избранное.

Просмотр избранных автомобилей.

Создание новых объявлений о продаже автомобиля с заполнением данных: VIN, марка, модель, год выпуска, цвет, объем двигателя, мощность, пробег, количество владельцев, цена, описание, город, тип кузова, трансмиссия, тип топлива, тип привода, а также загрузка до 5 фотографий.

Просмотр своих объявлений в профиле. Удаление своих объявлений.

Просмотр профиля пользователя (имя, телефон, email). Изменение фотографии профиля.

Выход из аккаунта.

Работа с бортжурналом: создание записей о техническом состоянии автомобиля с заполнением полей: марка, модель, заголовок, текст записи, пробег.

Просмотр записей бортжурнала.

Пользователь с ролью «Администратор» имеет следующие возможности:

Авторизация через заполнение полей: почта и пароль.

Восстановление пароля через отправку ссылки на почту через SMTP. Просмотр всех неподтвержденных объявлений.

Принятие или удаление объявлений. Просмотр всех записей бортжурнала. Принятие или удаление записей бортжурнала.

Управление списком марок и моделей автомобилей: добавление новых марок и моделей.

Удаление записей или объявлений по необходимости. Выход из аккаунта администратора.

Перед созданием мобильного приложения «Drivee» был выполнен анализ предметной области для определения ключевых процессов.

В результате были выделены процессы, подлежащие автоматизации.

На Рисунках 2–8 представлена диаграмма бизнес-процессов в нотации IDEF0, описывающая состояние перед внедрением приложения покупки и продаж автомобилей «Drivee».

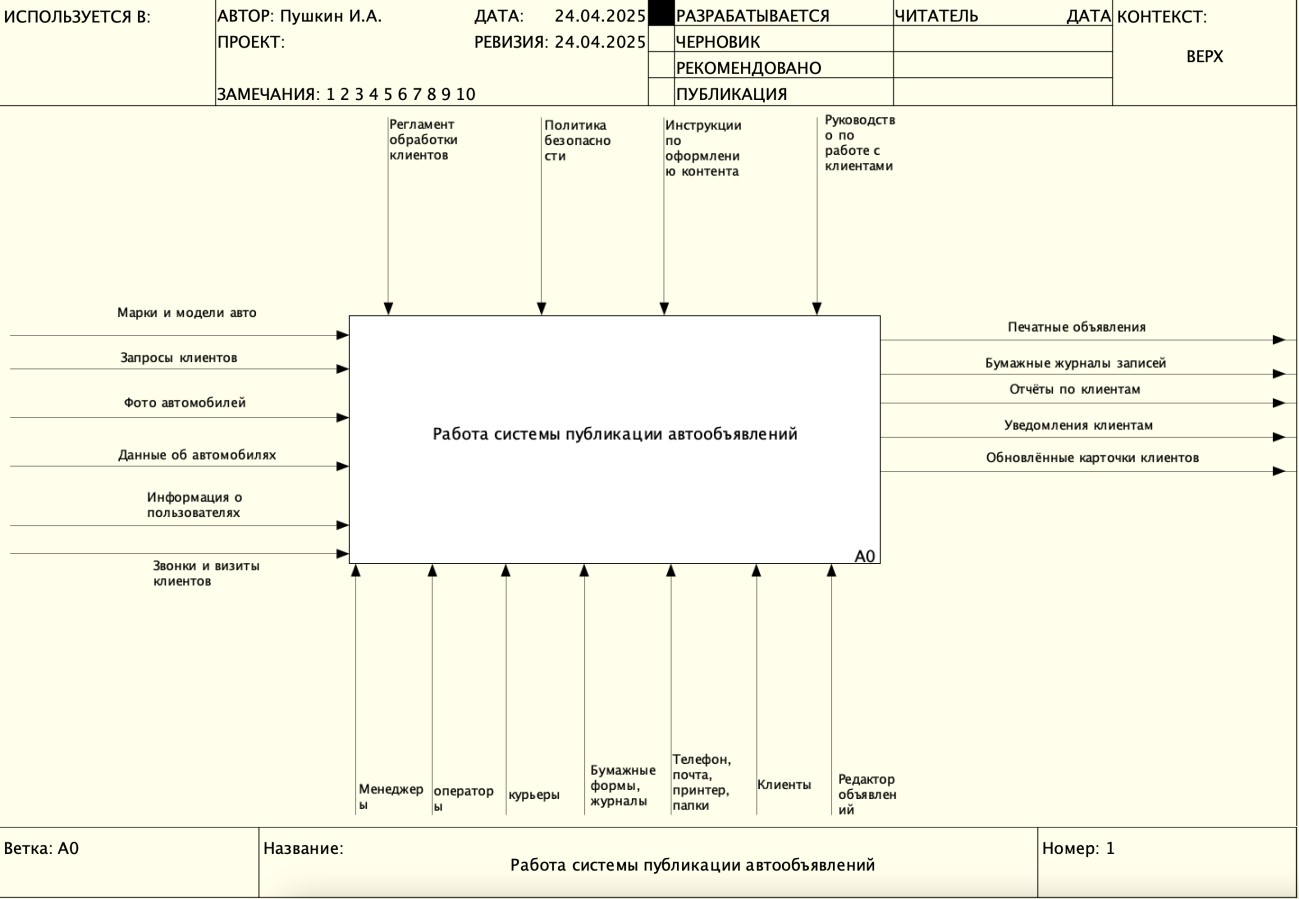


Рисунок 2 - Подпроцесс работы приложения ДО автоматизации

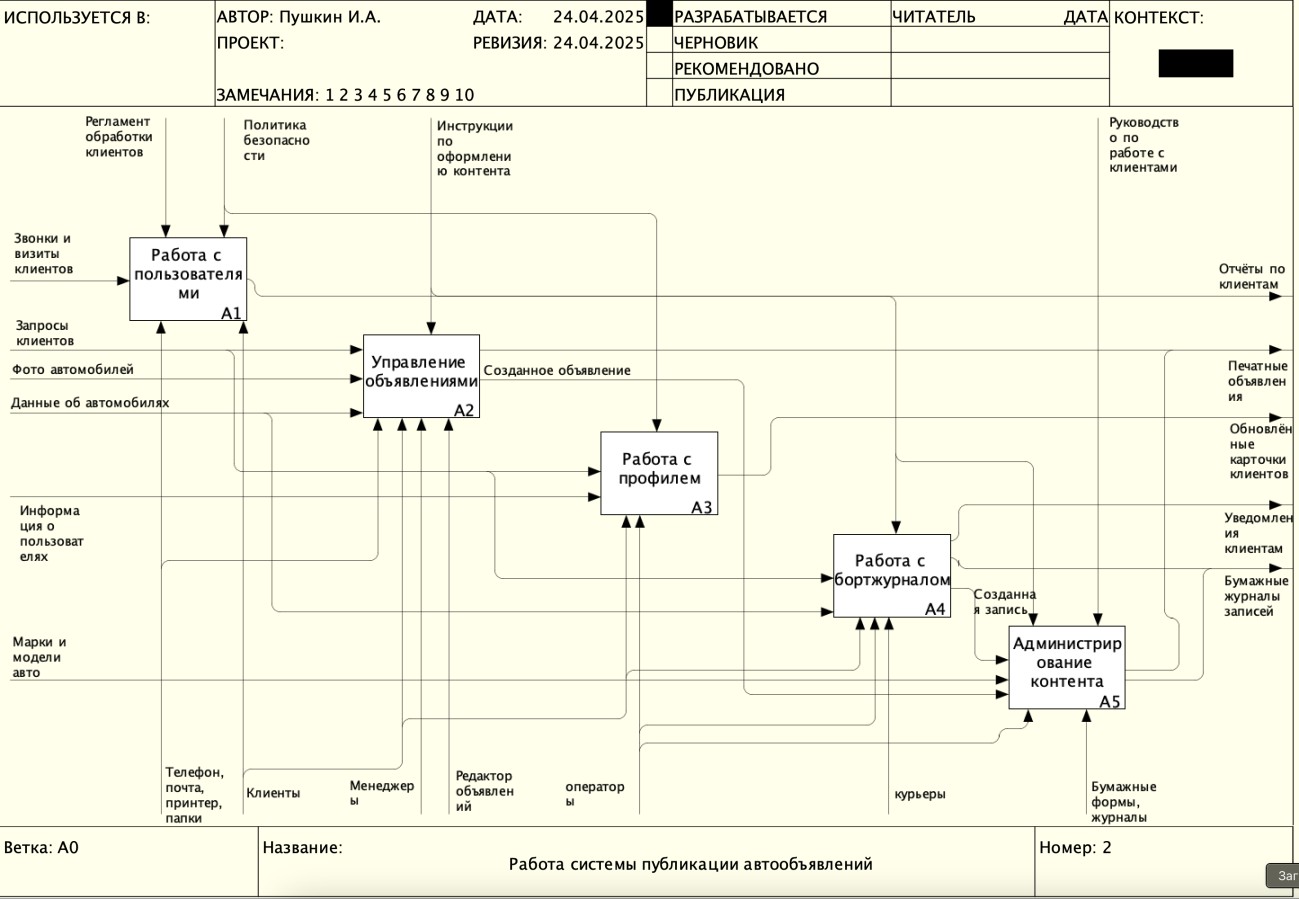


Рисунок 3 - Декомпозиция основного уровня

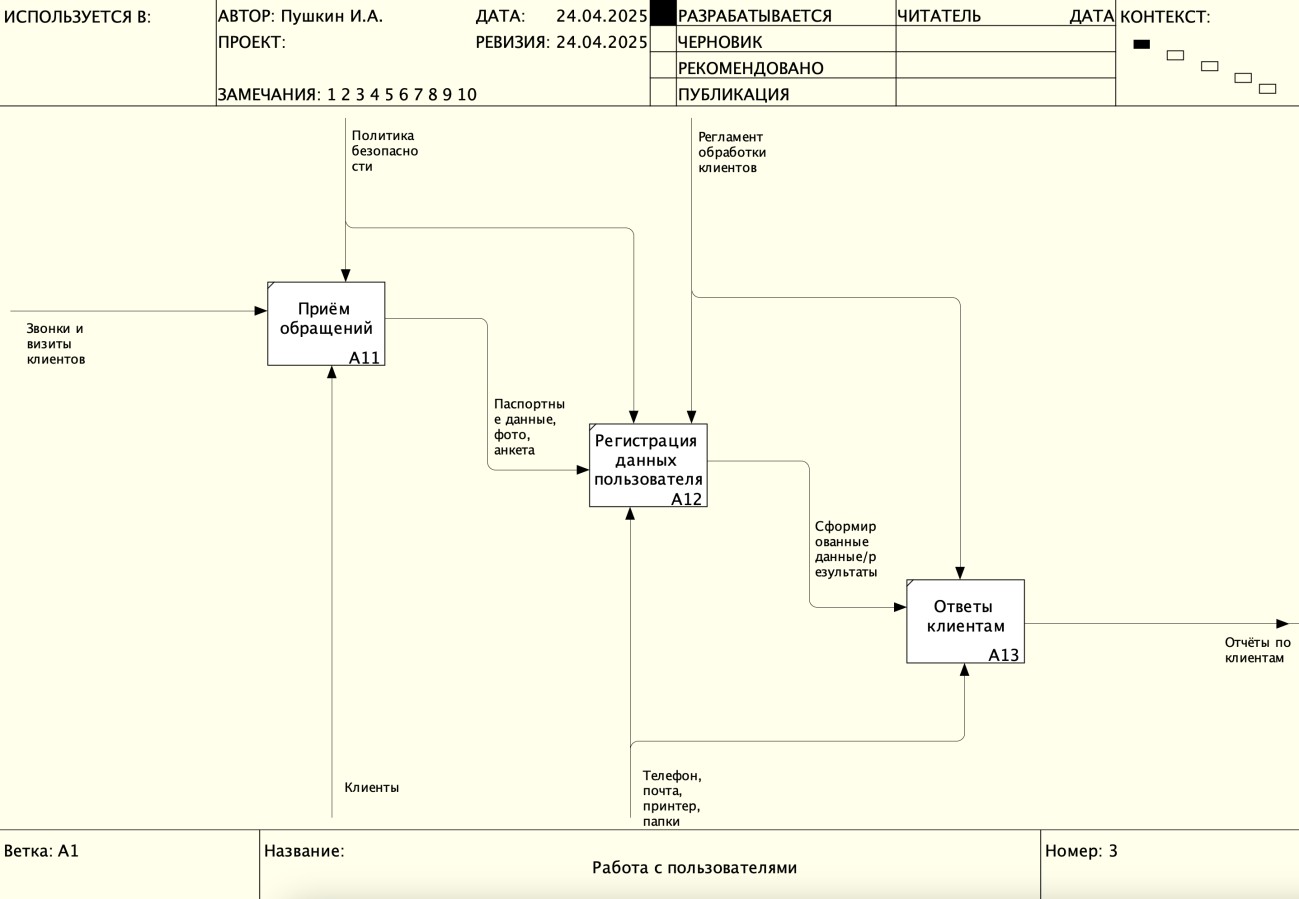


Рисунок 4 – Декомпозиция подпроцесса работы с пользователями

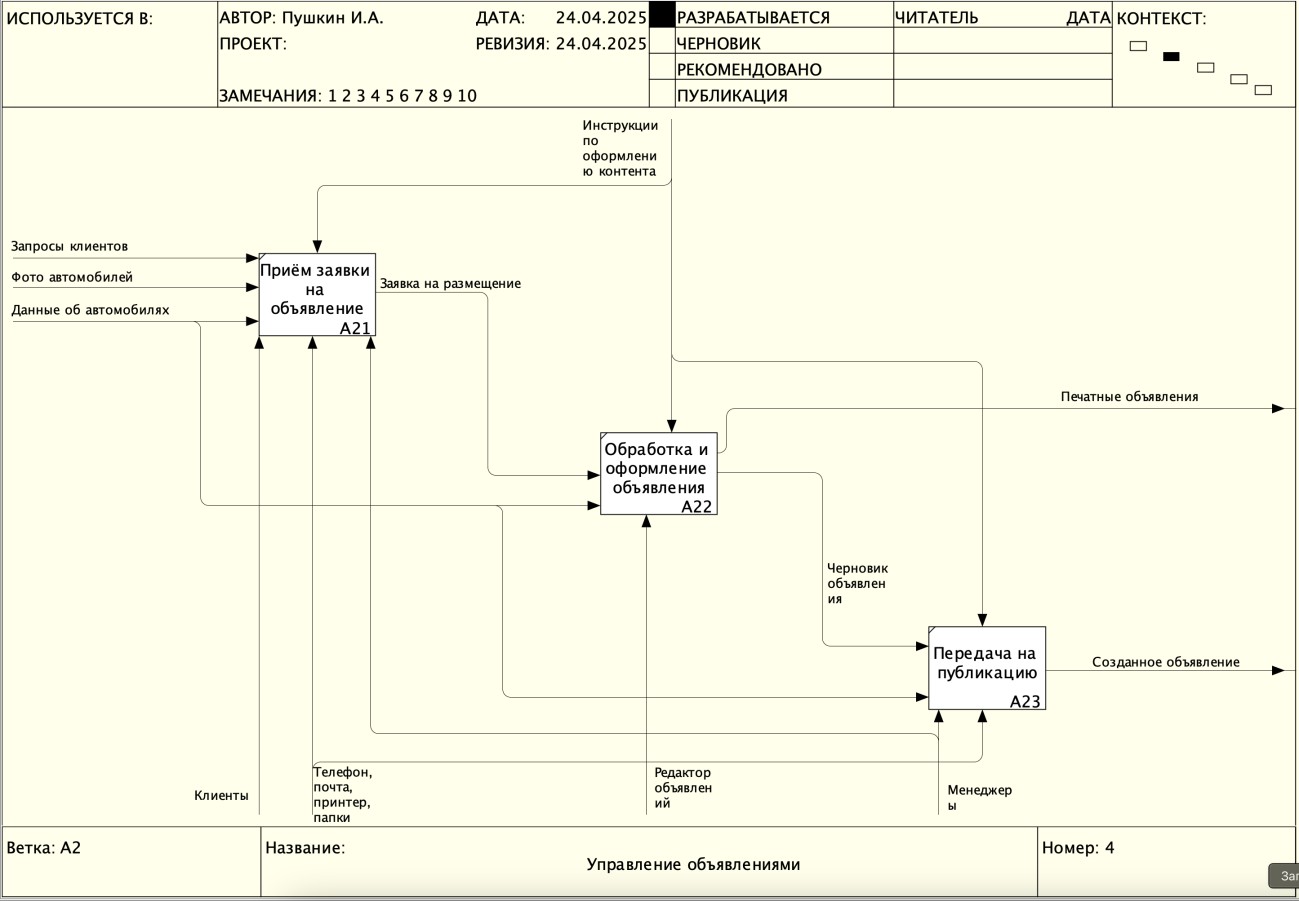


Рисунок 5 - Декомпозиция подпроцесса управления объявления

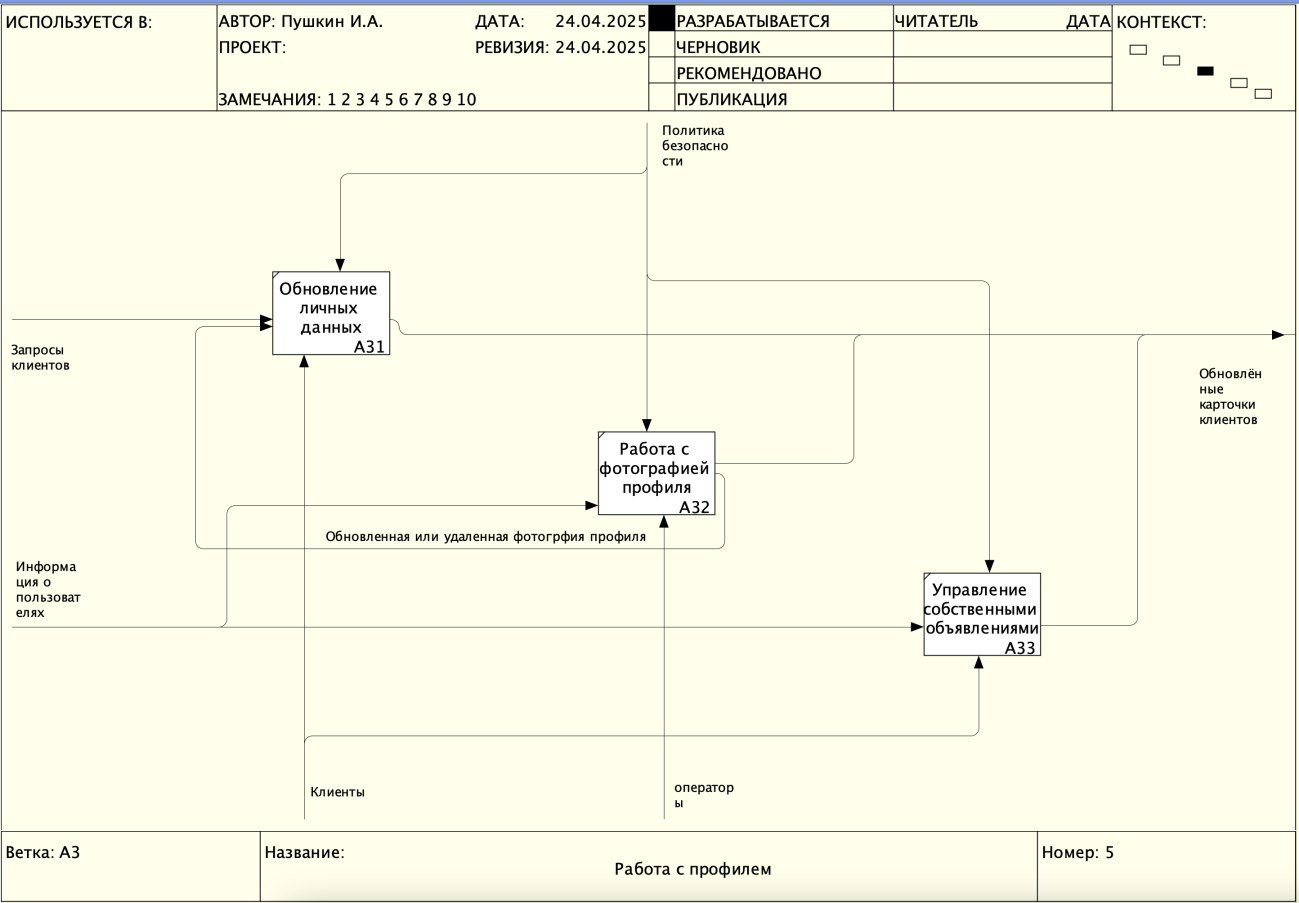


Рисунок 6 - Декомпозиция подпроцесса работы с профилем

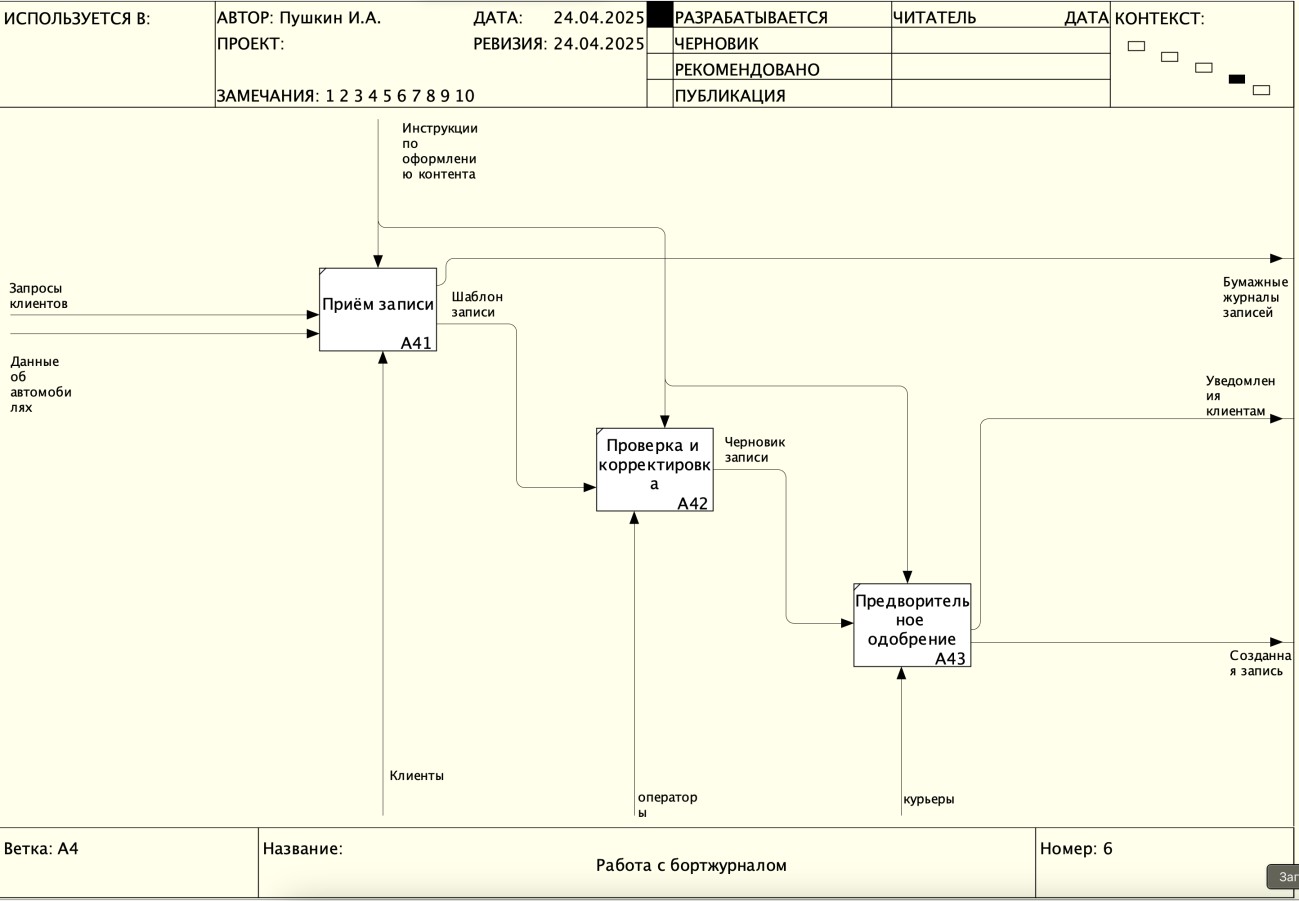


Рисунок 7 - Декомпозиция подпроцесса работы с бортжурналом

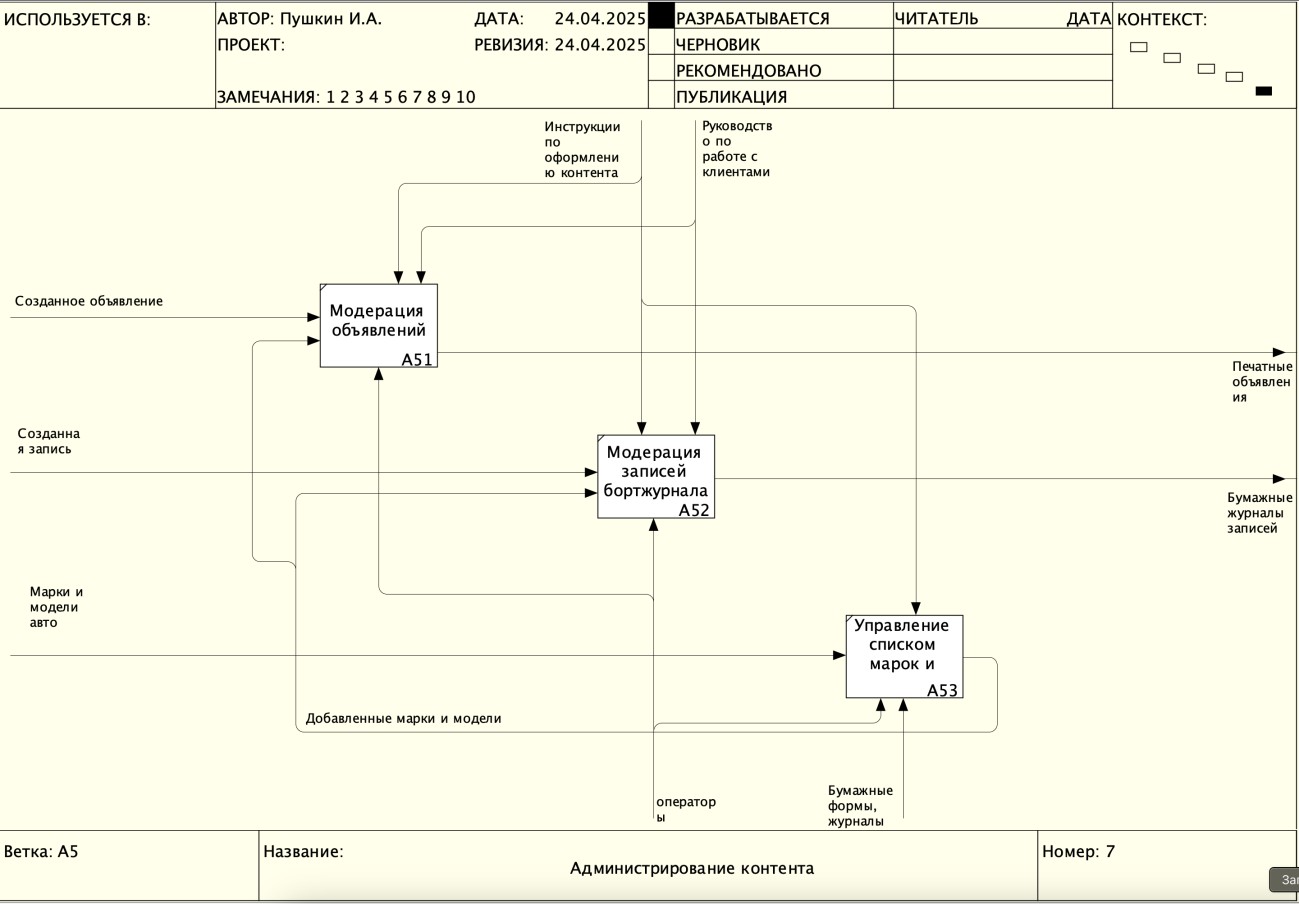


Рисунок 8 - Декомпозиция подпроцесса администрирования контента

## На Рисунках 9–15 представлена диаграмма бизнес-процессов в нотации IDEF0, описывающая состояние после внедрения приложения покупки и продаж автомобилей «Drivee».

Рисунок 9 – Подпроцесс работы приложения ПОСЛЕ автоматизации

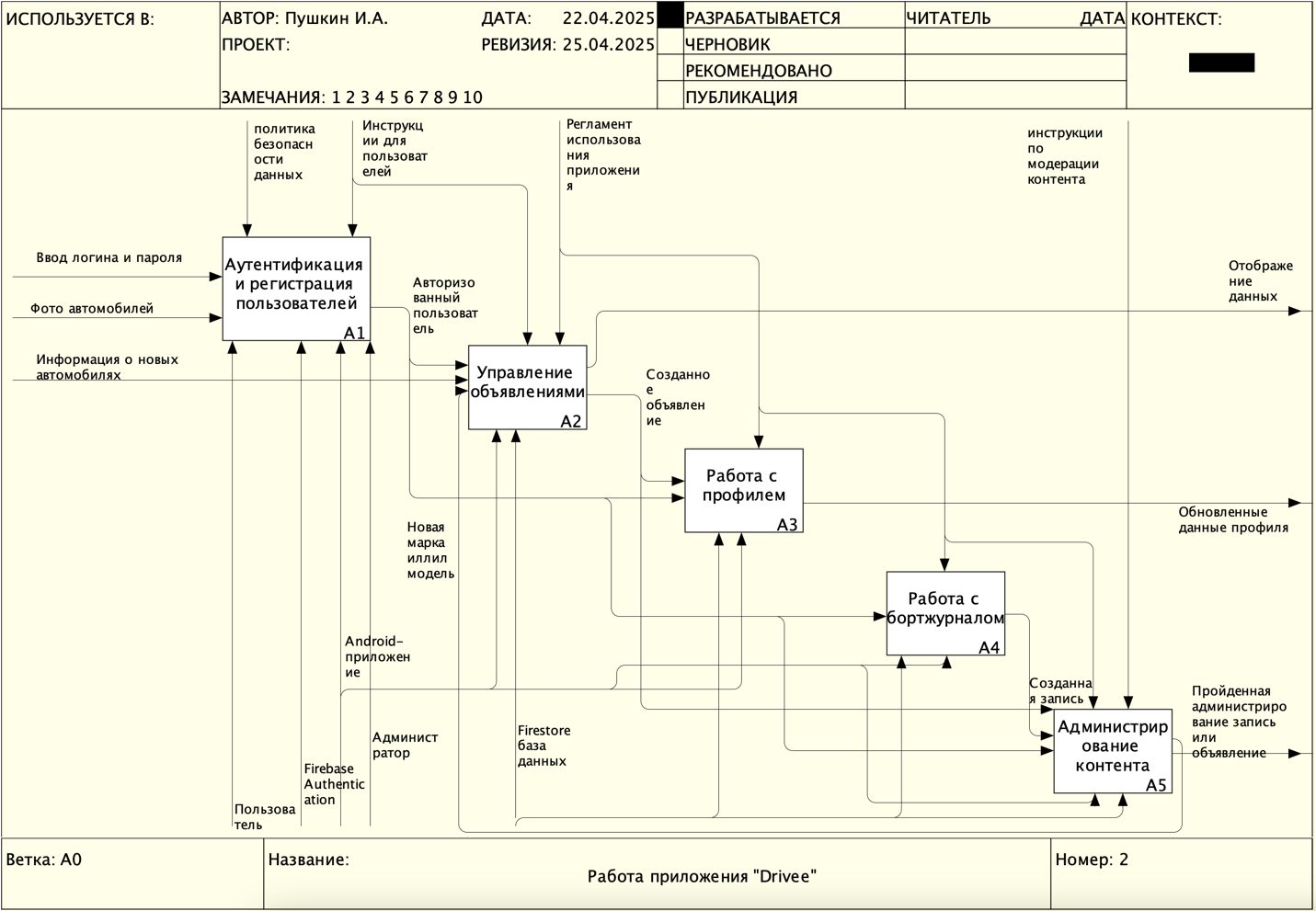


Рисунок 10 – Декомпозиция основного уровня

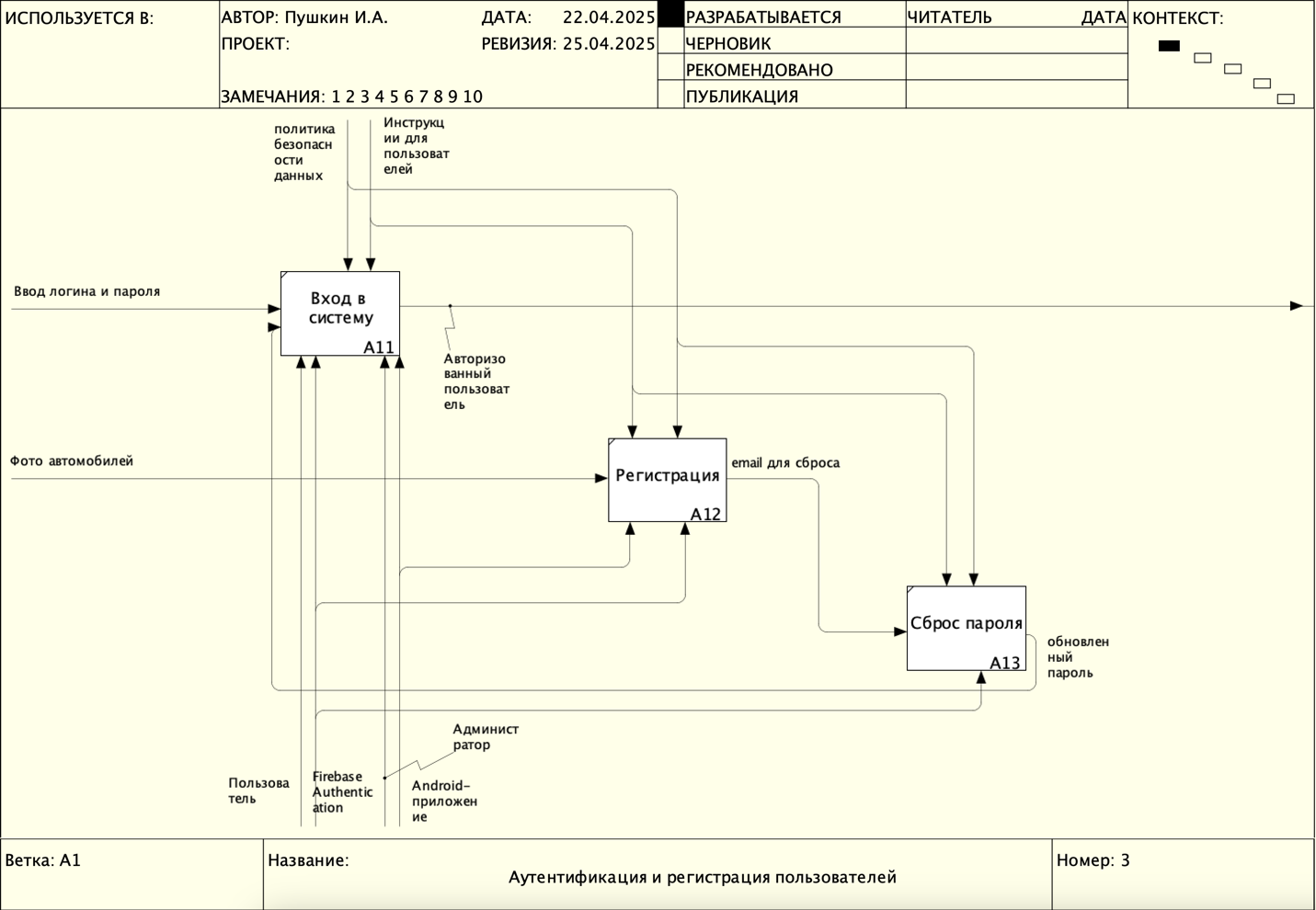


Рисунок 11 – Декомпозиция аутентификации и регистрации пользователей

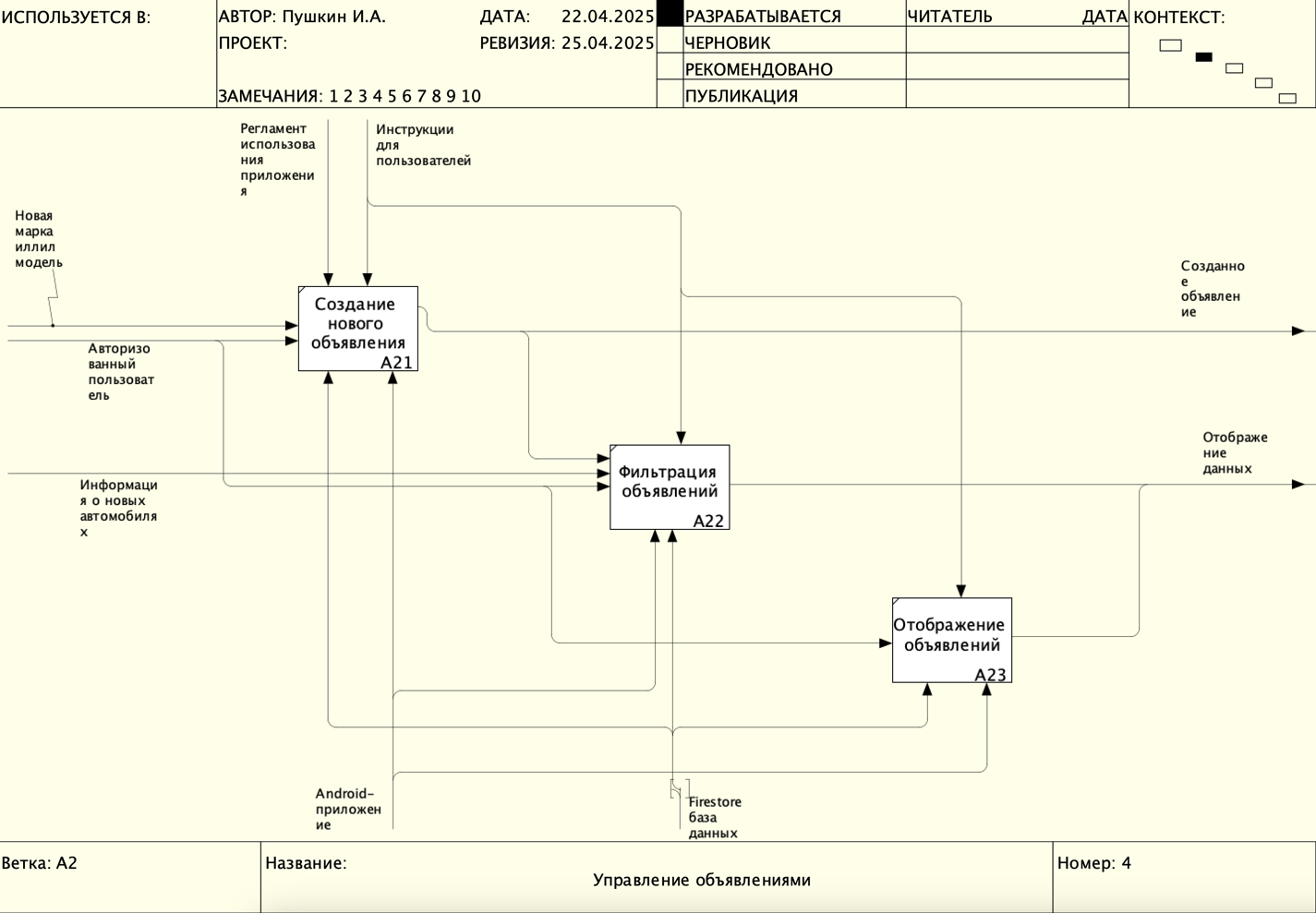


Рисунок 12 – Декомпозиция управления объявлениями

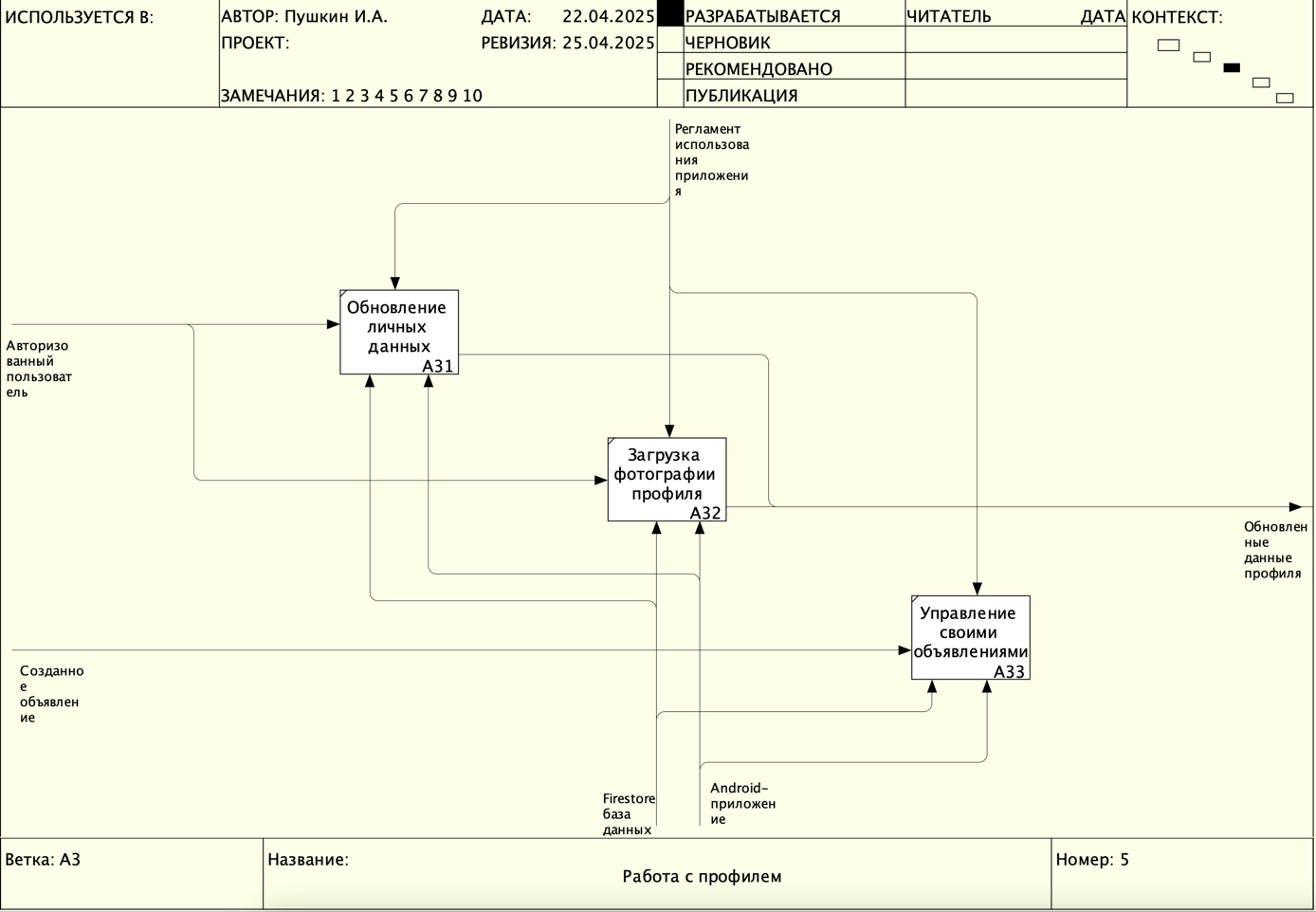


Рисунок 13 – Декомпозиция работы с профилем

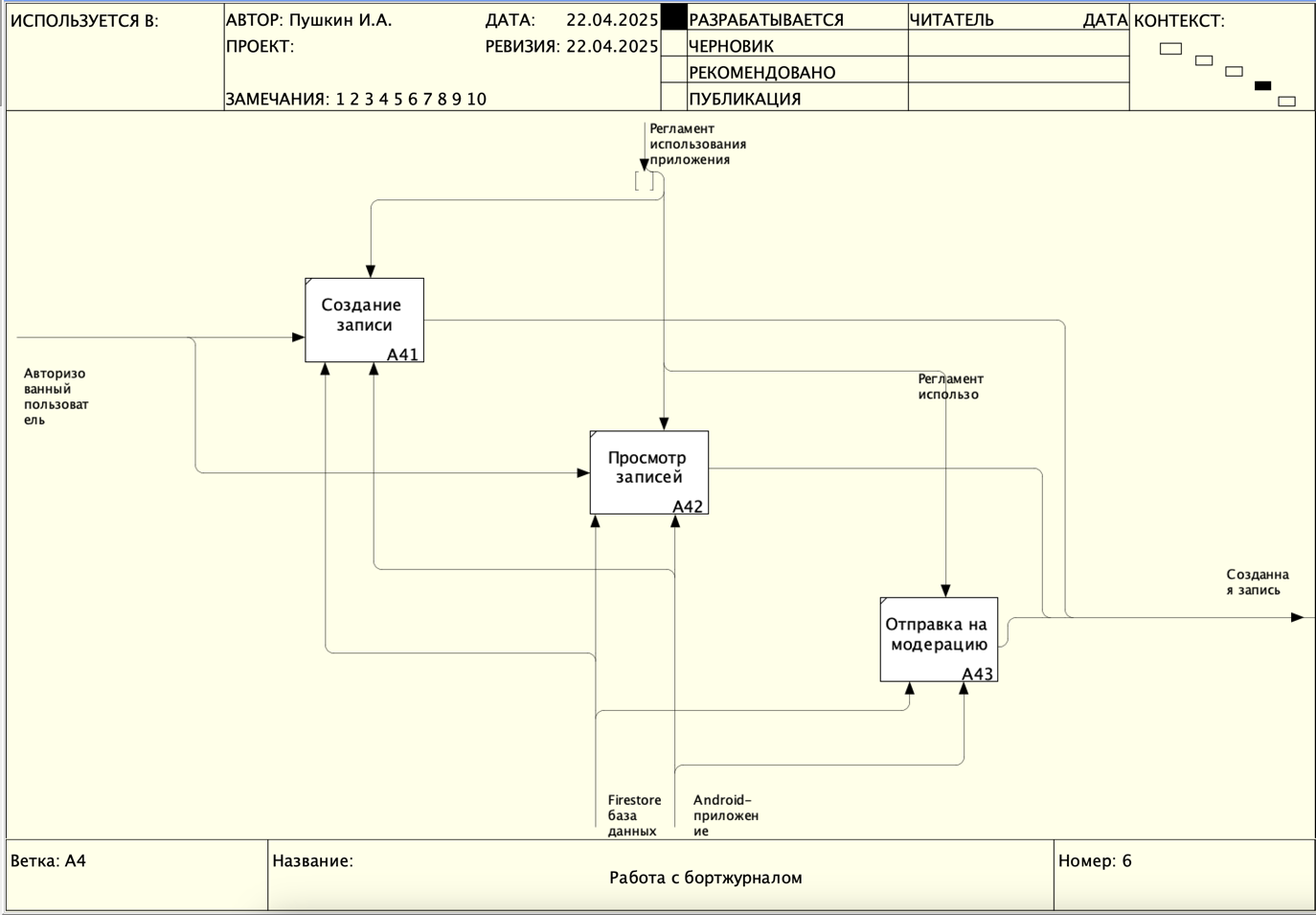


Рисунок 14 – Декомпозиция работы с бортжурналом

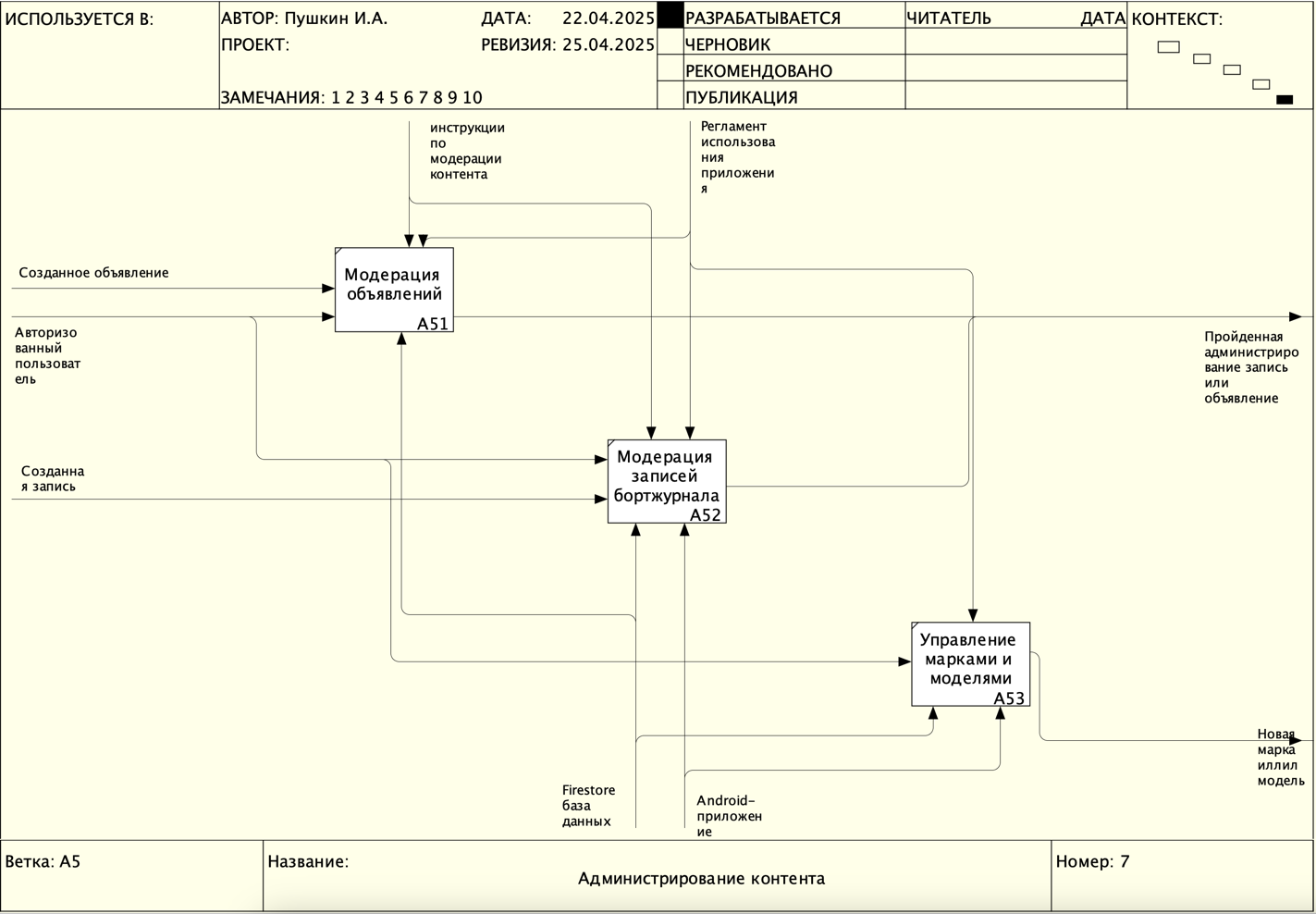


Рисунок 15 – Декомпозиция администрирования контента

## Входные и выходные данные

В Таблице 3 приведены входные данные программы с подробным описанием типа данных, его размера, диапазона изменений, точности представления и формы ввода.

Таблица 3 – Входные данные

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Имя (физический  смысл) | Тип | Размер в  байтах | Диапазон изменений | Точность представления | Форма ввода |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | Email  пользователя | String | 100 | От A до Z, от 0 до  9, символы @ . | Текст | Ввод с  клавиатуры |
| 2 | Пароль  пользователя | String | 100 | От A до Z, от 0 до  9, спецсимволы | Текст | Ввод с  клавиатуры |
| 3 | Имя  пользователя | String | 50 | От A до Z,  кириллица | Текст | Ввод с  клавиатуры |
| 4 | Дата рождения | Date | 8 | Формат  ДД.ММ.ГГГГ | Дата | Выбор даты |
| 5 | Номер  телефона | String | 20 | Формат  +7XXXXXXXXXX | Текст | Ввод с  клавиатуры |
| 6 | Повтор пароля | String | 100 | От A до Z, от 0 до 9, спецсимволы | Текст | Ввод с  клавиатуры |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Имя  (физический смысл) | Тип | Размер  в байтах | Диапазон изменений | Точность представления | Форма ввода |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 7 | VIN  автомобиля | String | 30 | 17 символов  (латиница + цифры) | Текст | Ввод с  клавиатуры |
| 8 | Марка автомобиля | String | 50 | Название марки | Текст | Выбор из  выпадающего списка |
| 9 | Модель автомобиля | String | 50 | Название модели | Текст | Выбор из выпадающего  списка |
| 10 | Год выпуска  автомобиля | Int | 4 | От 1900 до  текущего года | Целое число | Ввод с  клавиатуры |
| 11 | Цвет автомобиля | String | 30 | Название цвета | Текст | Выбор из  выпадающего списка |
| 12 | Объём  двигателя | Double | 8 | От 0.5 до 8.0  литров | С плавающей  точкой | Ввод с  клавиатуры |
| 13 | Мощность  двигателя | Int | 4 | От 30 до 1000 л.с. | Целое число | Ввод с  клавиатуры |
| 14 | Пробег  автомобиля | Int | 4 | От 0 до 500000 км | Целое число | Ввод с  клавиатуры |
| 15 | Количество  владельцев | Int | 4 | От 1 до 10 | Целое число | Ввод с  клавиатуры |
| 16 | Цена  автомобиля | Int | 4 | От 10000 до  50000000 ₽ | Целое число | Ввод с  клавиатуры |
| 17 | Описание  объявления | String | 1000 | Свободный текст  до 1000 символов | Текст | Ввод с  клавиатуры |
| 18 | Город продажи | String | 50 | Название города | Текст | Выбор из  выпадающего списка |
| 19 | Тип кузова | String | 30 | Седан, хэтчбек, кроссовер и т.д. | Текст | Выбор из выпадающего  списка |
| 20 | Тип трансмиссии | String | 30 | АКПП, МКПП и  др. | Текст | Выбор из  выпадающего списка |
| 21 | Тип топлива | String | 30 | Бензин, дизель, электро и т.д. | Текст | Выбор из выпадающего  списка |
| 22 | Тип привода | String | 30 | Передний, задний, полный | Текст | Выбор из выпадающего  списка |
| 23 | Изображение автомобиля | Bitmap | 500  000+ | До 5 файлов | Битовая карта | Выбор из  галереи устройства |
| 24 | Заголовок записи в  бортжурнале | String | 100 | Свободный текст | Текст | Ввод с  клавиатуры |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Имя  (физический смысл) | Тип | Размер  в байтах | Диапазон изменений | Точность представления | Форма ввода |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 25 | Текст записи в  бортжурнале | String | 1000 | Свободный текст | Текст | Ввод с  клавиатуры |
| 26 | Пробег записи бортжурнала | Int | 4 | От 0 до 500000 км | Целое число | Ввод с  клавиатуры |
| 27 | Новая марка  авто (для админа) | String | 50 | Название марки | Текст | Ввод с  клавиатуры |
| 28 | Новая модель  авто (для админа) | String | 50 | Название модели | Текст | Ввод с  клавиатуры |

## В Таблице 5 приведены выходные данные программы.

Таблица 4 – Выходные данные

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Имя (физический смысл) | Тип | Размер  в байтах | Диапазон изменений | Точность представления | Форма вывода |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Таблица пользователей (User) | | | | | | |
| 1 | Name | String | 40 | Текстовое поле | от A до z, 0-9, символы | Вывод в  текстовое поле профиля |
| 2 | Email | String | 40 | Текстовое поле | от A до z, 0-9, символы | Вывод в  текстовое поле профиля |
| 3 | Phone | String | 20 | Текстовое поле | только цифры | Вывод в  текстовое поле профиля |
| 4 | Password  (зашифрованный) | String | 64 | Хэширование | Строка хэш-  значений | Внутреннее  хранение |
| 5 | ProfileImage | String (Base64) | 50000 | Текстовое поле (Base64) | Двоичные данные | Вывод изображения в ImageView  профиля |
| Таблица объявлений (Car) | | | | | | |
| 6 | Brand | String | 40 | Текстовое поле | от A до z, 0-9, символы | Вывод на  карточке и в деталях |
| 7 | Model | String | 40 | Текстовое поле | от A до z, 0-9, символы | Вывод на  карточке и в деталях |
| 8 | Price | Long | 8 | Числовое  поле | Целое число | Вывод на  карточке |
| 9 | Mileage | Long | 8 | Числовое поле | Целое число | Вывод на карточке |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Имя (физический смысл) | Тип | Размер  в байтах | Диапазон изменений | Точность представления | Форма вывода |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 10 | Year | Int | 4 | Числовое поле | Целое число | Вывод в  характеристиках |
| 11 | Volume | Double | 8 | Числовое  поле | Дробное  число | Вывод в  характеристиках |
| 12 | Horsepower | Int | 4 | Числовое  поле | Целое число | Вывод в  характеристиках |
| 13 | Drive | String | 20 | Текстовое  поле | от A до z | Вывод в  характеристиках |
| 14 | Transmission | String | 20 | Текстовое  поле | от A до z | Вывод в  характеристиках |
| 15 | Fuel | String | 20 | Текстовое  поле | от A до z | Вывод в  характеристиках |
| 16 | Color | String | 20 | Текстовое  поле | от A до z | Вывод в  характеристиках |
| 17 | BodyType | String | 20 | Текстовое  поле | от A до z | Вывод в  характеристиках |
| 18 | City | String | 30 | Текстовое поле | от A до z | Вывод в  карточке объявления |
| 19 | Owners | Int | 4 | Числовое  поле | Целое число | Вывод в  характеристиках |
| 20 | Description | String | 500 | Текстовое поле | Символьное поле | Вывод в  описании объявления |
| 21 | PhotoOne | String  (Base64) | 50000 | Текстовое  поле | Base64 код  изображения | Вывод  изображения |
| 22 | PhotoTwo | String  (Base64) | 50000 | Текстовое  поле | Base64 код  изображения | Вывод  изображения |
| 23 | PhotoThree | String  (Base64) | 50000 | Текстовое  поле | Base64 код  изображения | Вывод  изображения |
| 24 | PhotoFour | String  (Base64) | 50000 | Текстовое  поле | Base64 код  изображения | Вывод  изображения |
| 25 | PhotoFive | String  (Base64) | 50000 | Текстовое  поле | Base64 код  изображения | Вывод  изображения |
| Таблица избранного (Favorites) | | | | | | |
| 26 | FavoriteCarIds | Array of String | 500 | Массив строк | ID объявлений | Использование при отображении  избранного |
| Таблица бортжурнала (Bortjournal) | | | | | | |
| 27 | Title | String | 100 | Текстовое  поле | от A до z, 0-9,  символы | Вывод в  карточке записи |
| 28 | Text | String | 1000 | Текстовое  поле | Символьное  поле | Вывод в деталях  записи |
| 29 | Mileage | Long | 8 | Числовое поле | Целое число | Вывод в деталях записи |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Имя (физический смысл) | Тип | Размер  в байтах | Диапазон изменений | Точность представления | Форма вывода |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 30 | Brand | String | 40 | Текстовое  поле | от A до z, 0-9,  символы | Вывод в  карточке записи |
| 31 | Model | String | 40 | Текстовое поле | от A до z, 0-9, символы | Вывод в  карточке записи |

## Методы

В ходе разработки мобильного приложения "Drivee" были использованы следующие методы программирования:

Объектно-Ориентированное Программирование (ООП):

Программа построена на принципах ООП, где каждый элемент системы (например, пользователь, автомобиль, объявление) представлен в виде объектов с собственными полями и методами. Это обеспечило модульность кода, упрощение его сопровождения и расширения.

Структурное программирование:

При реализации отдельных модулей (например, регистрация пользователя, фильтрация объявлений, работа с профилем) применялись принципы структурного программирования. Логика работы приложения была разделена на последовательные, понятные блоки, что повысило читаемость и устойчивость кода.

Реактивное программирование:

При взаимодействии с базой данных Firebase (Firestore) и обработки событий пользовательского интерфейса применялись элементы реактивного подхода. Изменения данных мгновенно отражаются в интерфейсе без необходимости ручного обновления — например, при добавлении новой машины в список объявлений или изменении профиля пользователя.

## Тесты

Сценарий тестирования и результаты тестовых испытаний приведен в Приложении В. Сценарий тестов и результаты тестовых испытаний.

## Контроль целостности данных

В Таблице 5 представлен контроль целостности данных в приложении "Drivee", описывающий типичные ситуации и реакцию системы при сохранении, выводе, изменении или удалении данных.

Таблица 5 – Контроль целостности данных

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ситуация | Аномалия | Реакция программы |
| 1 | 2 | 3 | 4 |
| 1 | Попытка регистрации с уже существующим email | Пользователь уже существует в базе данных | Отображение сообщения "Аккаунт с таким email уже существует" и запрет на  регистрацию |
| 2 | Попытка создать объявление без  заполнения всех  обязательных полей | Некорректный объект в базе данных | Появление предупреждения "Заполните все обязательные поля" и запрет на отправку  формы |
| 3 | Изменение информации в  объявлении (например, цены) | Мгновенное  обновление данных без подтверждения | Изменения сохраняются  только после явного нажатия кнопки "Сохранить" |
| 4 | Попытка удалить объявление без подтверждения | Ошибочное удаление объявления | Появляется диалог подтверждения "Вы уверены,  что хотите удалить объявление?" |
| 5 | Добавление машины в "Избранное", потом её удаление из базы администратором | Ссылка в "Избранном" на несуществующий объект | При попытке открытия карточки - отображение сообщения "Объявление больше недоступно" и  удаление его из списка |
| 6 | Изменение профиля пользователя без подключения к интернету | Данные не  сохраняются в базе данных | Выводится сообщение об ошибке "Нет подключения к интернету. Повторите попытку  позже." |
| 7 | Оформление заказа без выбора машины | Некорректный заказ в базе | Блокировка кнопки "Оформить заказ", пока не будет добавлено хотя бы одно  объявление |

## Проектирование

## Схема архитектуры приложения

Клиент отправляет запрос в облачное хранилище через Web-API Firebase SDK. Этот SDK, находясь внутри облачного хранилища, обрабатывает запрос и взаимодействует с базой данных Firebase Firestore. После того как база данных возвращает нужную информацию, SDK формирует ответ и отправляет его обратно клиенту.

На Рисунке 16 представлена клиент-серверная архитектура программы.

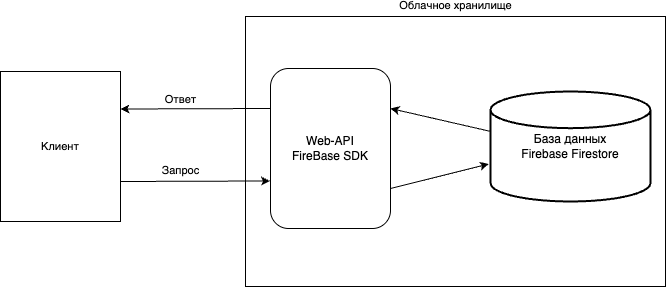


Рисунок 16 - Архитектурная схема приложения

## Логическая схема данных

Разработка данной схемы не предусмотрена в связи с отсутствием необходимости в рамках поставленных задач.

## Физическая схема данных

Разработка данной схемы не предусмотрена в связи с отсутствием необходимости в рамках поставленных задач.

## Структурная схема

На Рисунке 17-19 представлена структурная схема программы. Большинство классов в программе являются глобальными и выступают как вспомогательные элементы, поэтому на схеме приведены зависимые файлы проекта.

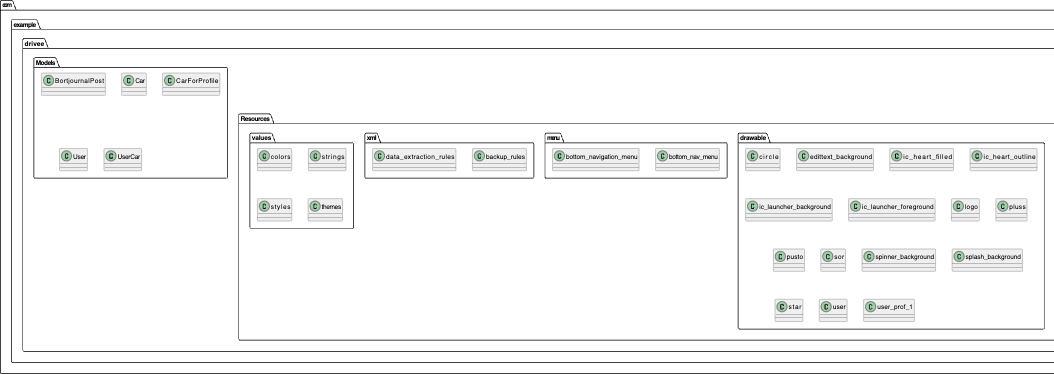


Рисунок 17 – Структурная схема приложения часть 1

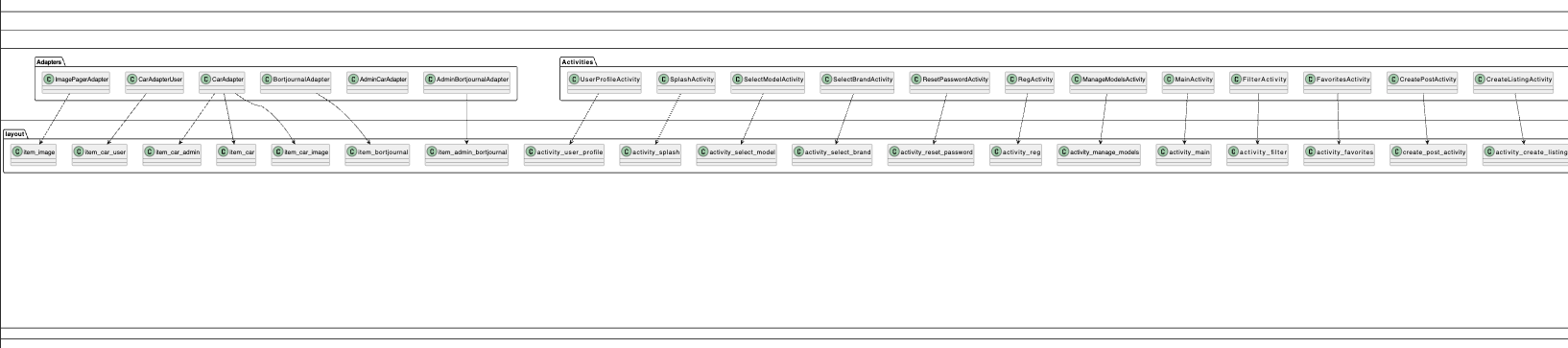


Рисунок 18 - Структурная схема приложения часть 2



Рисунок 19 - Структурная схема приложения часть 1

## Функциональная схема

На диаграмме изображена структура мобильного приложения для продажи и покупки автомобилей. Она состоит из различных страниц, каждая из которых выполняет определённую функцию и взаимодействует с другими. Вот простое описание:

Приложение начинается с экранов входа, регистрации и сброса пароля, которые позволяют пользователю авторизоваться или создать новый аккаунт.

После успешного входа пользователь попадает на главную страницу, где можно перейти к фильтрации объявлений, просмотру профиля, созданию записи, или добавлению объявлений в избранное.

Есть отдельные страницы для создания объявлений и создания записей, где вводятся данные об автомобиле (тип кузова, топливо, привод, марка, модель, VIN, год, цена и пр.). Также присутствует автоматическое определение города по геолокации.

Избранное и профиль пользователя позволяют управлять сохранёнными объявлениями и личной информацией. Страницы подробной информации об объявлениях и записях предоставляют подробности и возможность перейти к другим частям приложения.

Отдельно выделены страницы администрирования, где можно управлять объявлениями, записями, а также добавлять марки и модели автомобилей.

Все страницы связаны между собой переходами, что обеспечивает логичную и полную навигацию по приложению.

На Рисунке 20 представлена функциональная схема.

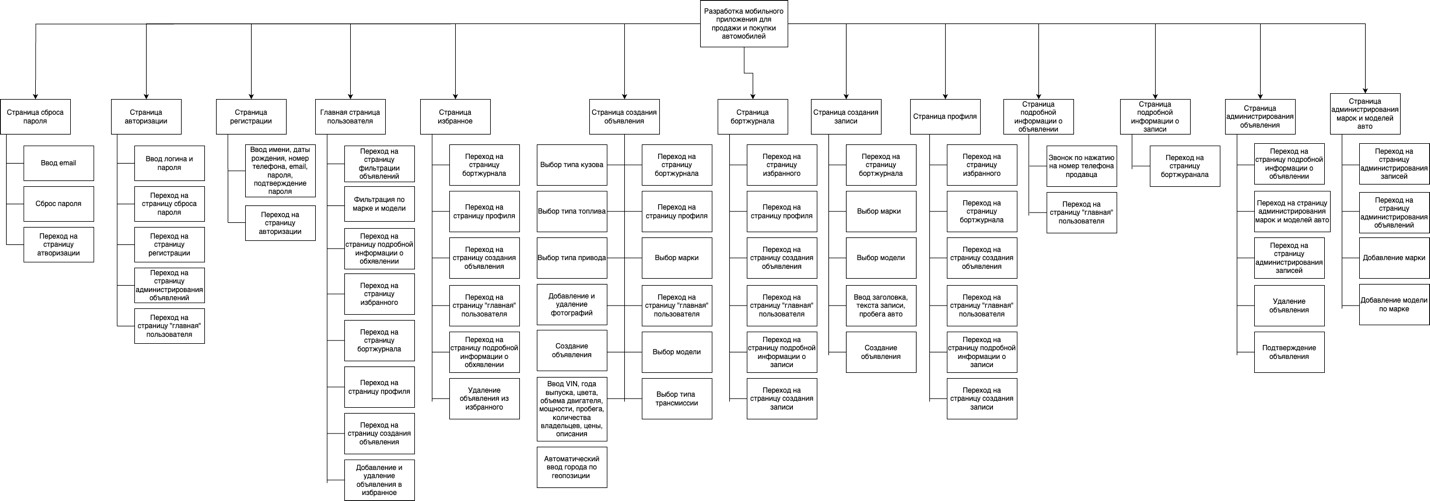


Рисунок 20 - Функциональная схема

## Диаграмма классов

На Рисунках 21-23 представлены части диаграммы классов программы. В классах указаны внутренние атрибуты и методы, используемые программой.

Связь между классами показана стрелками. В глобальных элементах и интерфейсах стрелки не присутствуют, однако также описаны внутренние члены классов.

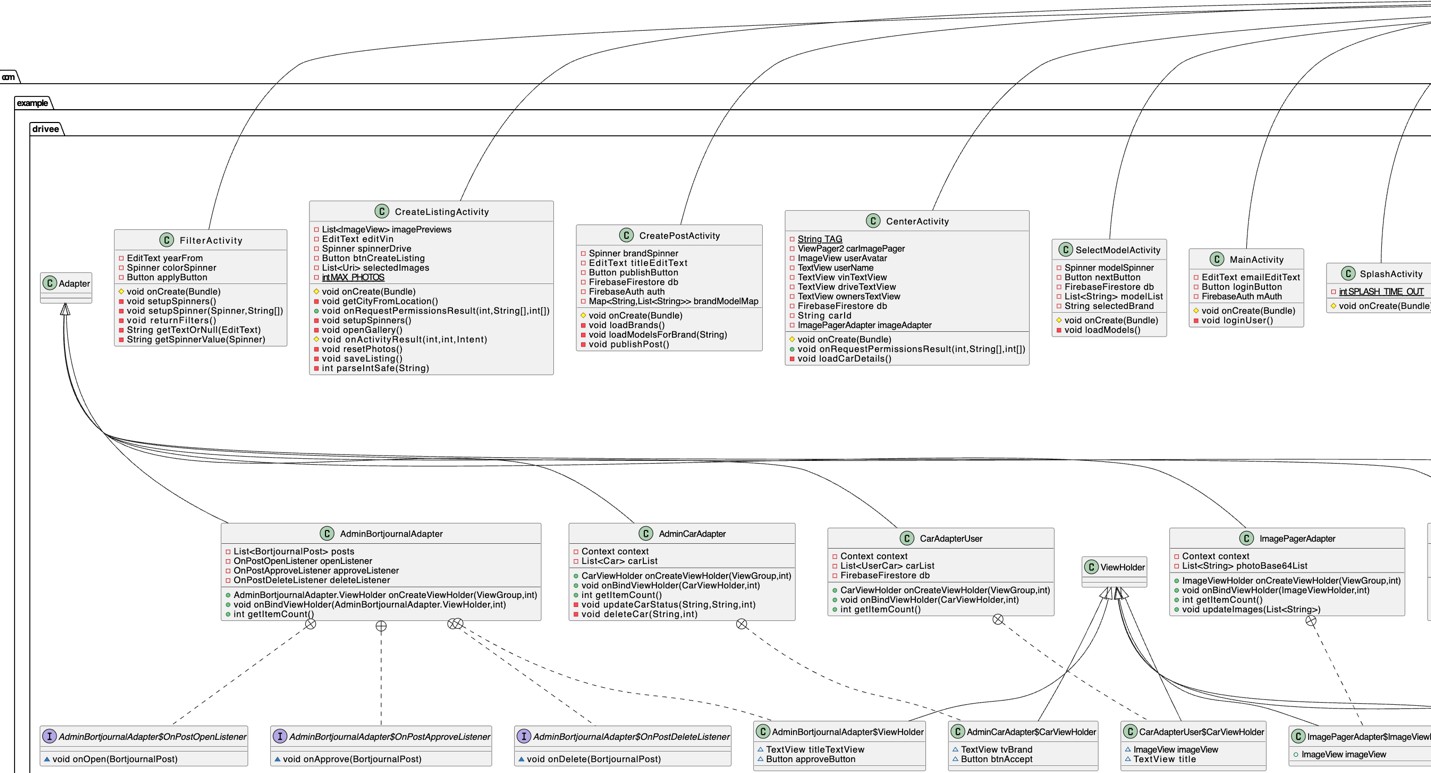


Рисунок 21 - Диаграмма классов часть 1

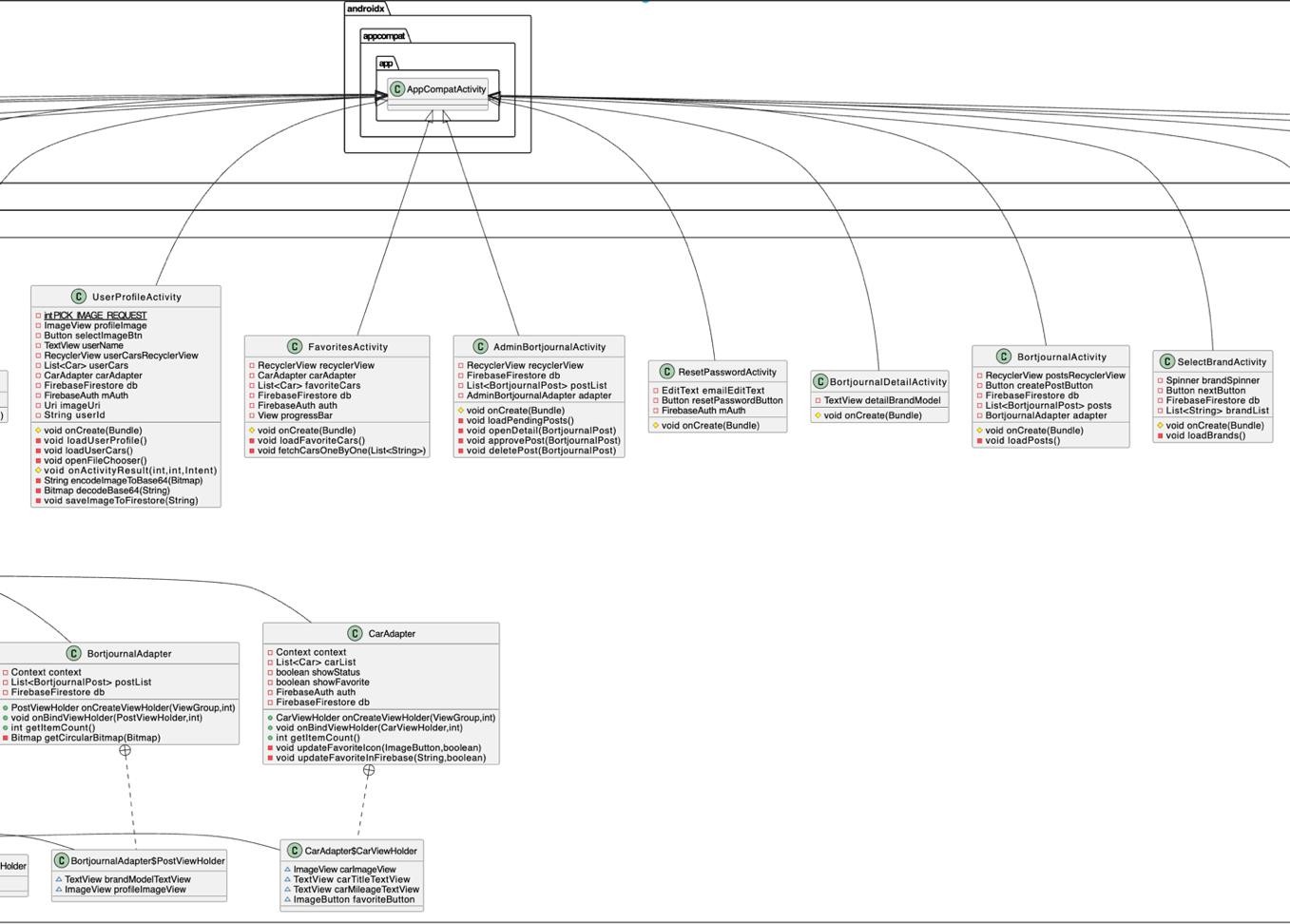


Рисунок 22 - Диаграмма классов часть 2

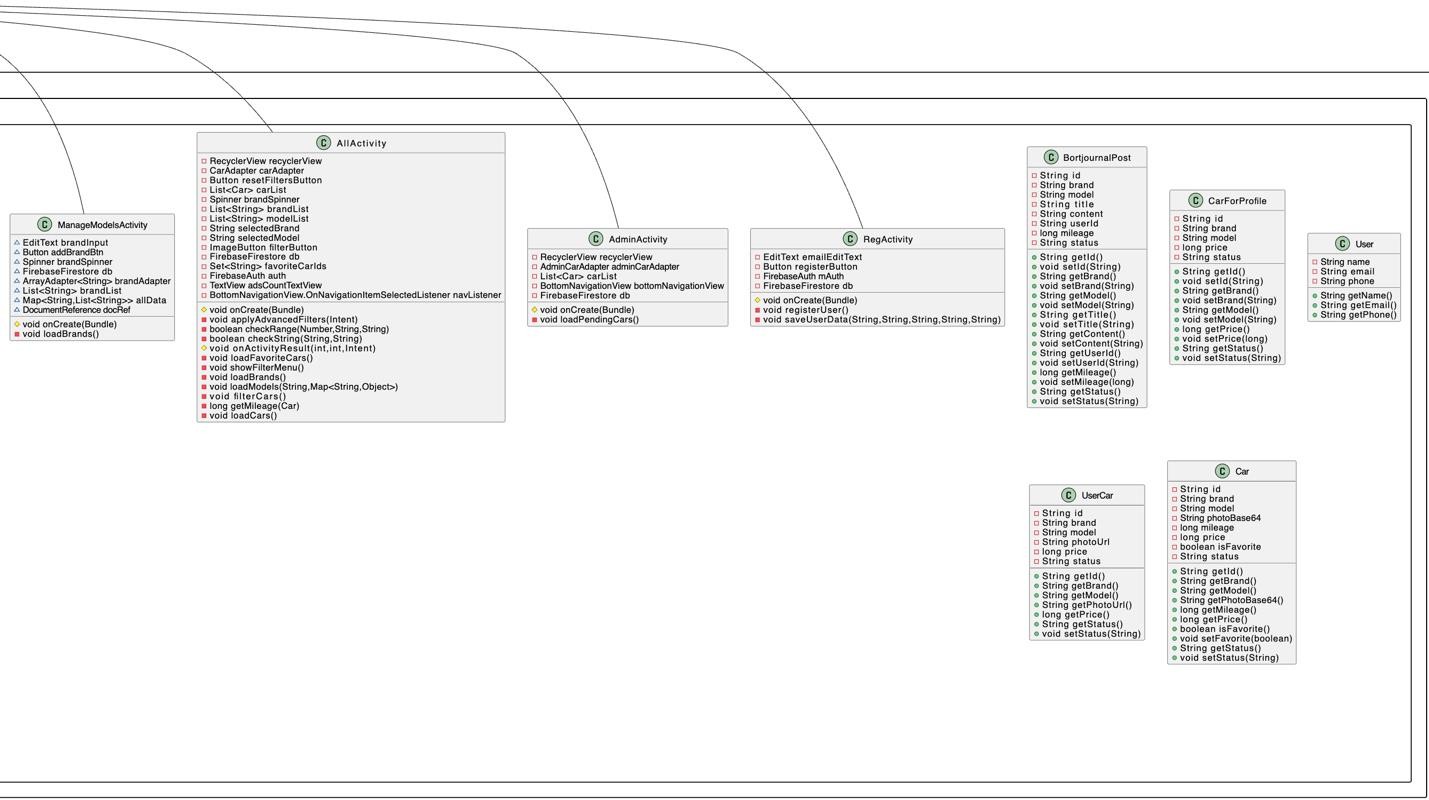


Рисунок 23 - Диаграмма классов часть 3

Таблица 6 – Описание диаграммы классов

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | 2 | 3 |
| 1 | AndroidManifest.xml | конфигурация приложения: разрешения, иконки,  список активити |
| 2 | AdminActivity.java | панель администратора, просмотр необработанных  объявлений |
| 3 | AdminBortjournalActivity. java | панель администратора для управления бортжурналами |
| 4 | AdminBortjournalAdapter.  java | адаптер для отображения бортжурналов в админке |
| 5 | AdminCarAdapter.java | адаптер для списка машин в админке. |
| 6 | AllActivity.java | главный экран со всеми объявлениями +  фильтры и навигация. |
| 7 | BortjournalActivity.java | экран бортжурнала пользователя |
| 8 | BortjournalAdapter.java | адаптер для списка записей в бортжурнале |
| 9 | BortjournalDetailActivity.j  ava | экран подробностей по записи в бортжурнале |
| 10 | BortjournalPost.java | модель (POJO) для записи в бортжурнал |
| 11 | Car.java | модель машины |
| 12 | CarAdapter.java | адаптер для отображения карточек машин |
| 13 | CarAdapterUser.java | адаптер для отображения машин текущего  пользователя |
| 14 | CarForProfile.java | отдельная модель машины для профиля |
| 15 | CenterActivity.java | экран карточки машины |
| 16 | CreateListingActivity.java | создание нового объявления |
| 17 | CreatePostActivity.java | создание записи в бортжурнал |

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | 2 | 3 |
| 18 | FavoritesActivity.java | избранные объявления пользователя |
| № | Название | Описание |
| 19 | FilterActivity.java | расширенные фильтры поиска |
| 20 | ImagePagerAdapter.java | адаптер для ViewPager с фото машины |
| 21 | MainActivity.java | экран авторизации |
| 22 | ManageModelsActivity.ja  va | админка: управление марками и моделями |
| 23 | RegActivity.java | экран регистрации |
| 24 | ResetPasswordActivity.jav  a | сброс пароля |
| 25 | SelectBrandActivity.java | выбор марки авто. |
| 26 | SelectModelActivity.java | выбор модели авто. |
| 27 | SplashActivity.java | splash-экран при запуске |
| 28 | User.java | модель пользователя |
| 29 | UserCar.java | модель машины, привязанной к пользователю |
| 30 | UserProfileActivity.java | профиль пользователя |

## Схема тестирования

Функциональное тестирование охватывает базовые функции приложения: авторизация, регистрация, сброс пароля, загрузка профиля пользователя, выход из аккаунта, просмотр и администрирование объявлений и записей, создание объявлений и записей, применение фильтров поиска, работа с избранным, а также управление марками и моделями авто.

Тестирование производительности проверяет скорость загрузки различных страниц (авторизация, главная, профиль, бордюркула, создание объявления), а также работу приложения при одновременной активности нескольких пользователей.

Тестирование пользовательского интерфейса включает проверку отображения карточек машин, фото профиля, выпадающих списков, форм создания объявлений, навигации и адаптации интерфейса под разные размеры экранов.

Тестирование кроссплатформенности фокусируется на работоспособности приложения на разных версиях Android (от 8 до 12) и на различных устройствах.

Вся схема систематизирует процесс тестирования, чтобы обеспечить надёжность и удобство использования приложения.

На Рисунке 24 представлена схема тестирования настольного приложения «Drivee».

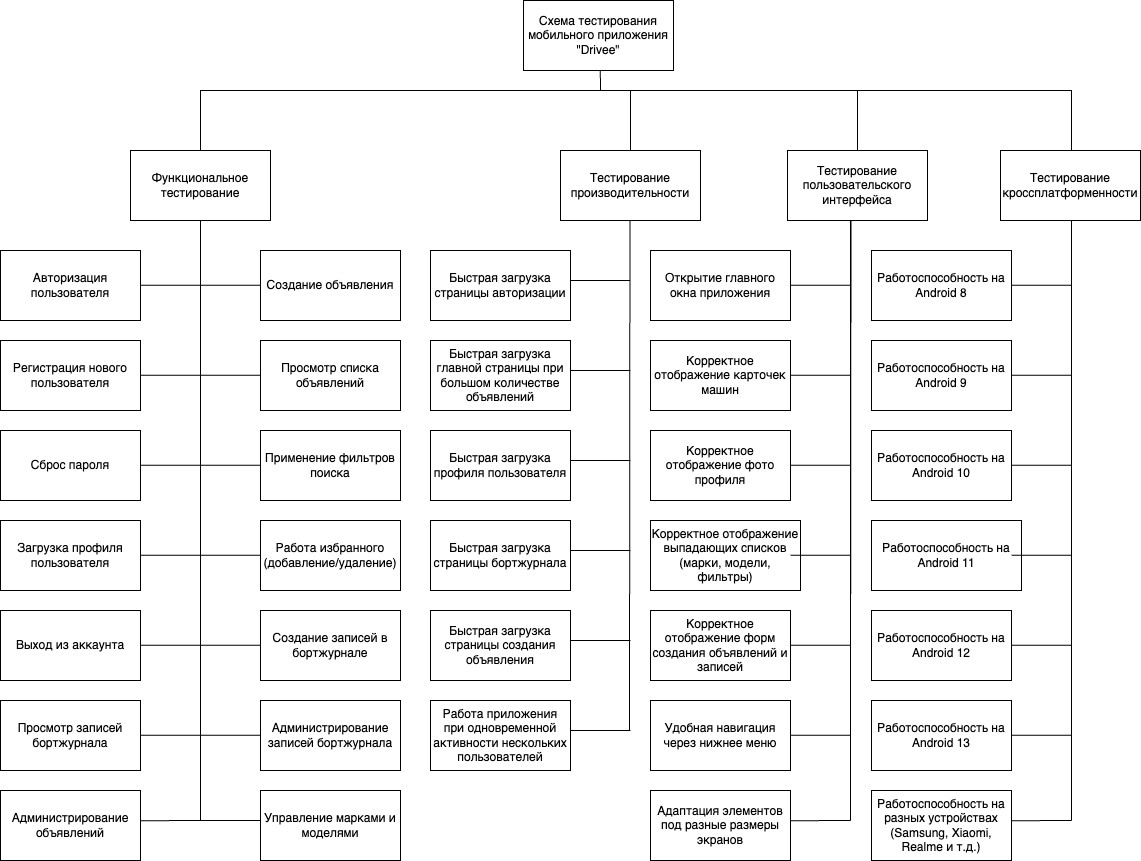


Рисунок 24 – Схема тестирования

## Схема пользовательского интерфейса

Схема показывает, как пользователь перемещается между основными экранами мобильного приложения. Всё начинается со страницы авторизации, откуда можно попасть на регистрацию или сброс пароля. После входа открывается главная страница пользователя с доступом к профилю, поиску, бортжурналу, избранному и созданию объявлений. Из этих разделов можно перейти на страницы выбора марки и модели автомобиля, а также на подробную информацию о машине. Для администратора предусмотрена отдельная ветка с доступом к модерации, управлению бортжурналом и редактированию марок и моделей. Всё соединено так, чтобы переходы между функциями были логичными и быстрыми.

На Рисунке 25 представлена схема пользовательского интерфейса.

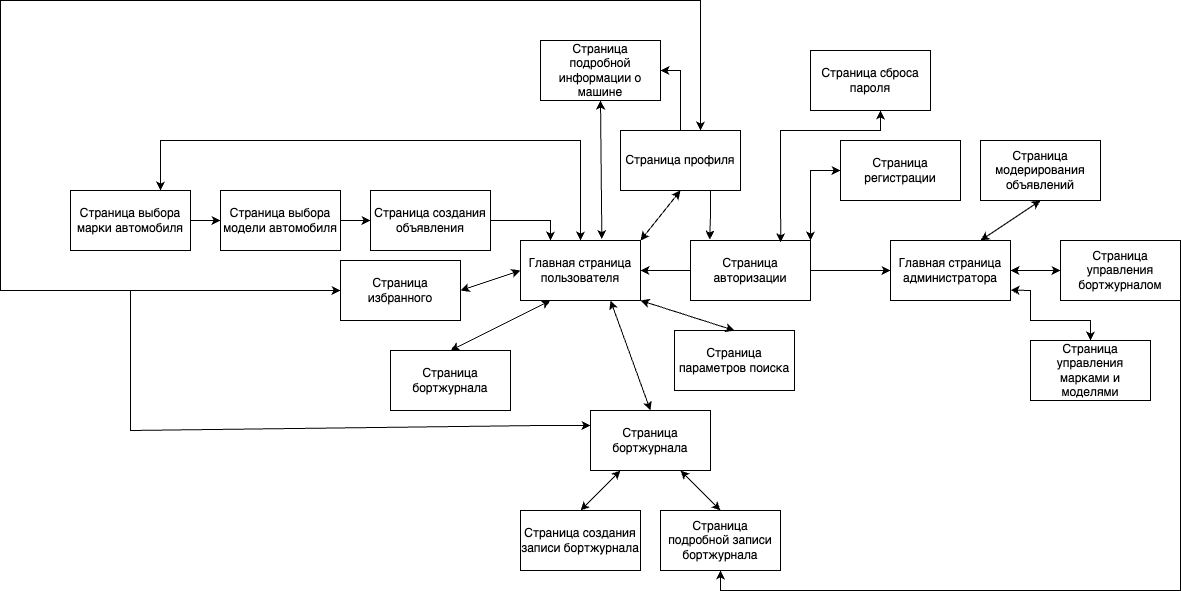


Рисунок 25 – Схема пользовательского интерфейса

## Результат работы программы

На Рисунках 26-37 кратко продемонстрирована работы программы.

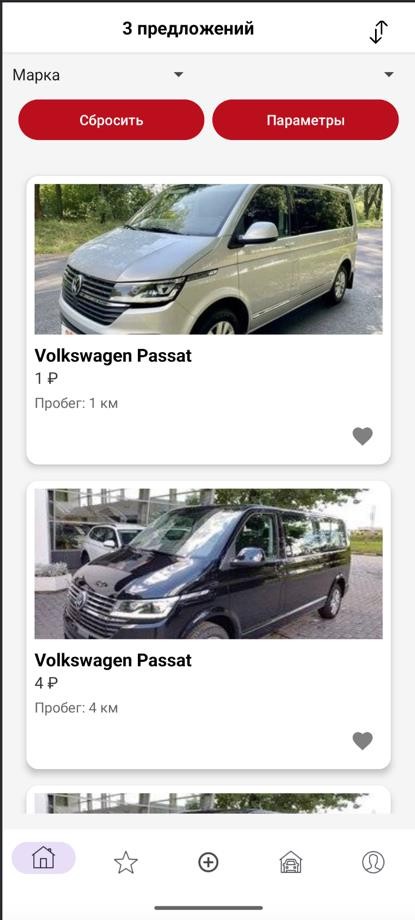


Рисунок 26 – Главная страница пользователя

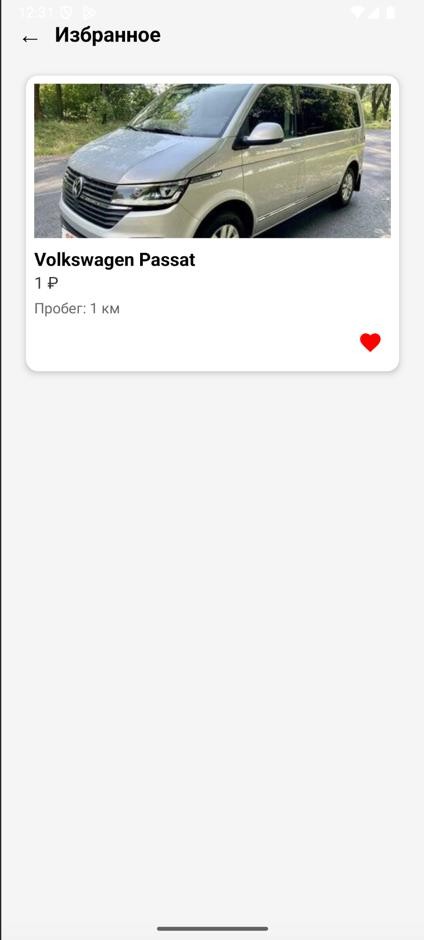


Рисунок 27 – Страница избранное

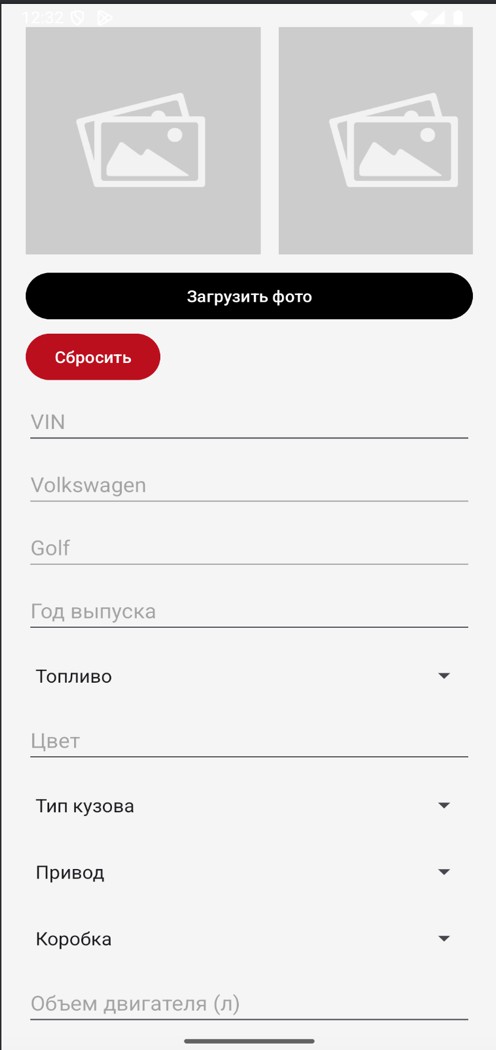


Рисунок 28 – Страница создания объявления

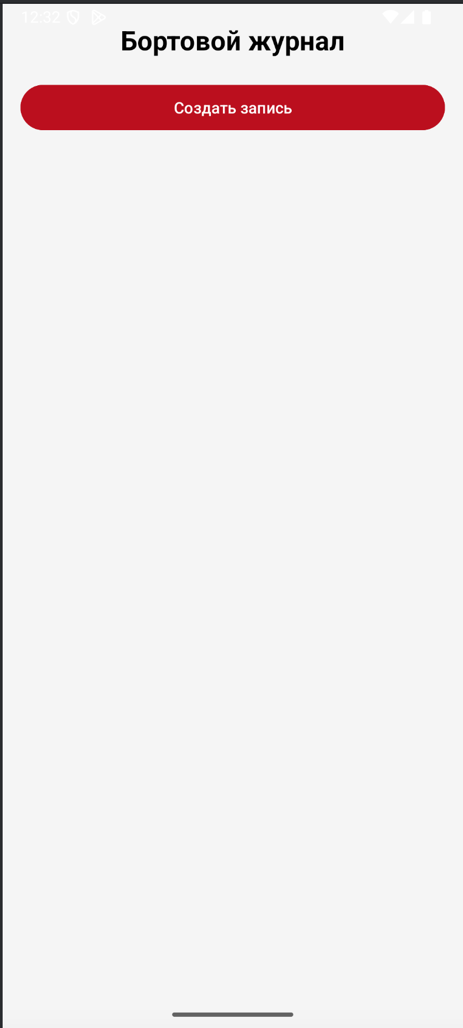


Рисунок 29 – Страница бортового журанала



Рисунок 30 – Страница пользователя

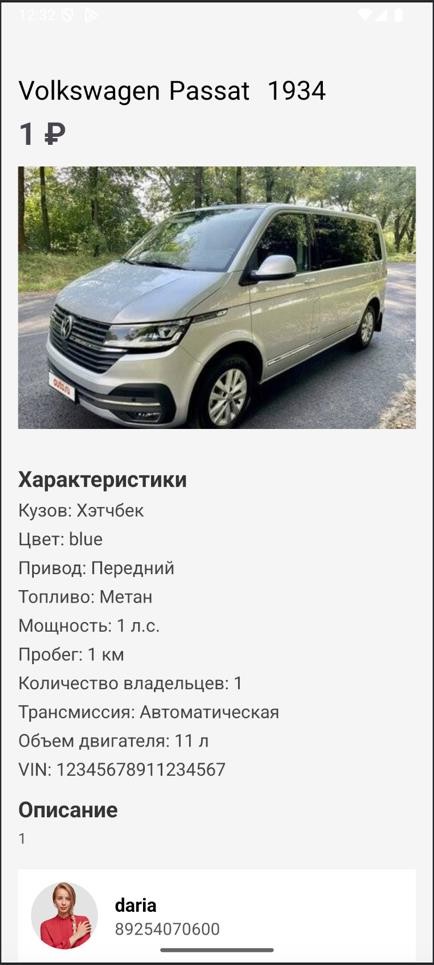


Рисунок 31 – Подробная информации об объявлении

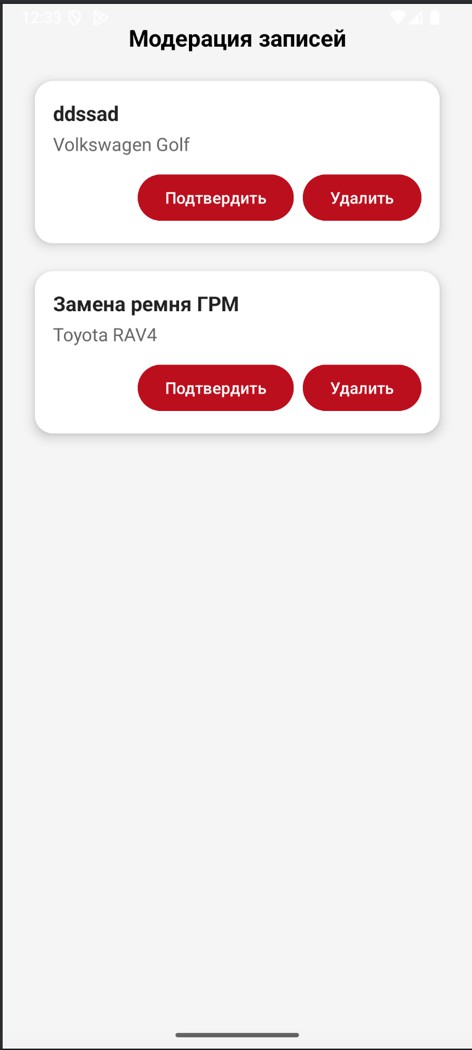


Рисунок 32 – Модерирование записей



Рисунок 33 – Страница редактирования марок и моделей авто

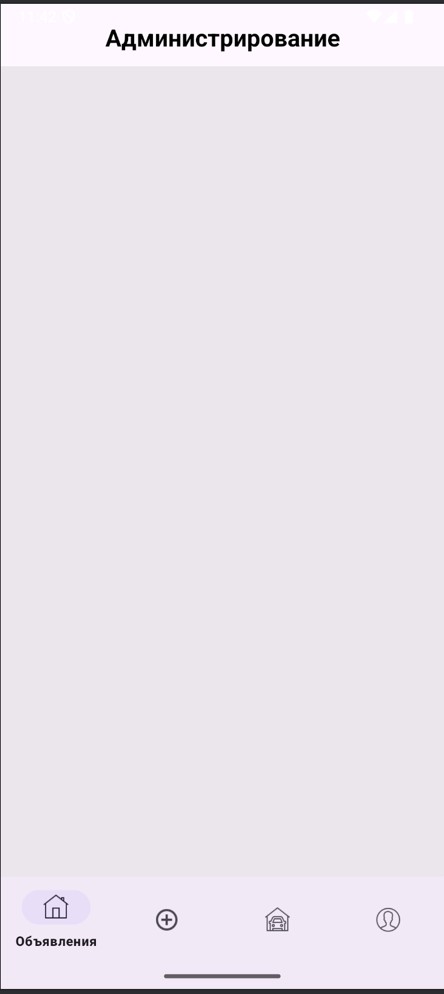


Рисунок 34 – Администрирование объявлений

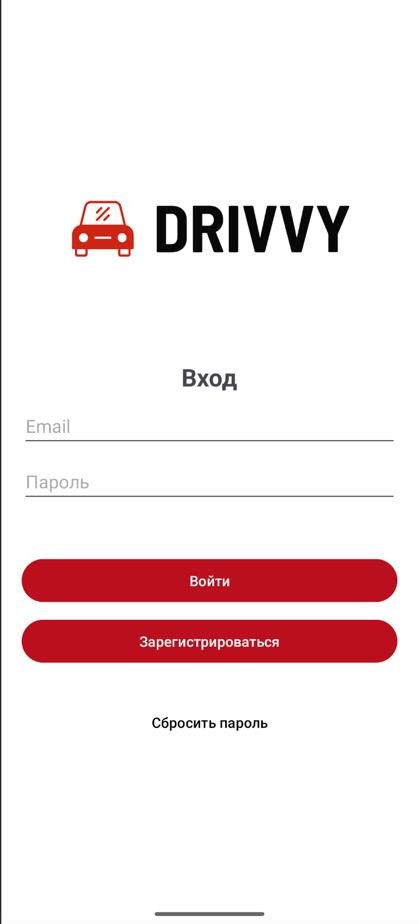


Рисунок 35 – Страница атворизации

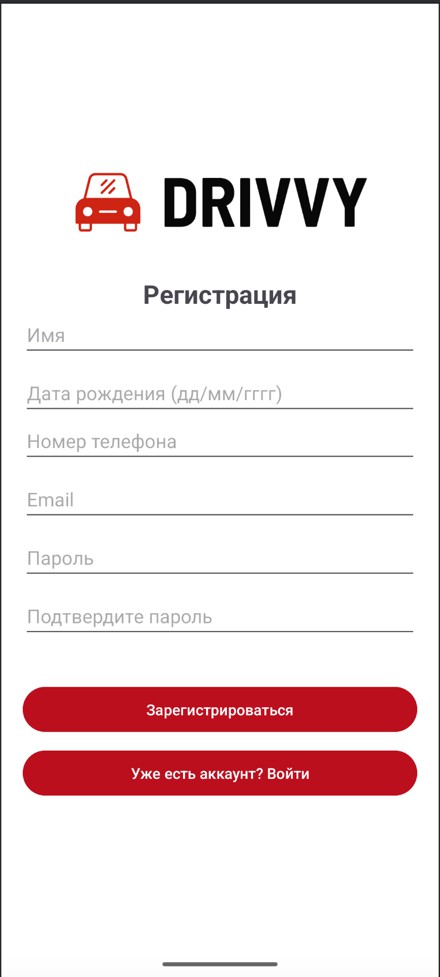


Рисунок 36 – Страница регистрации

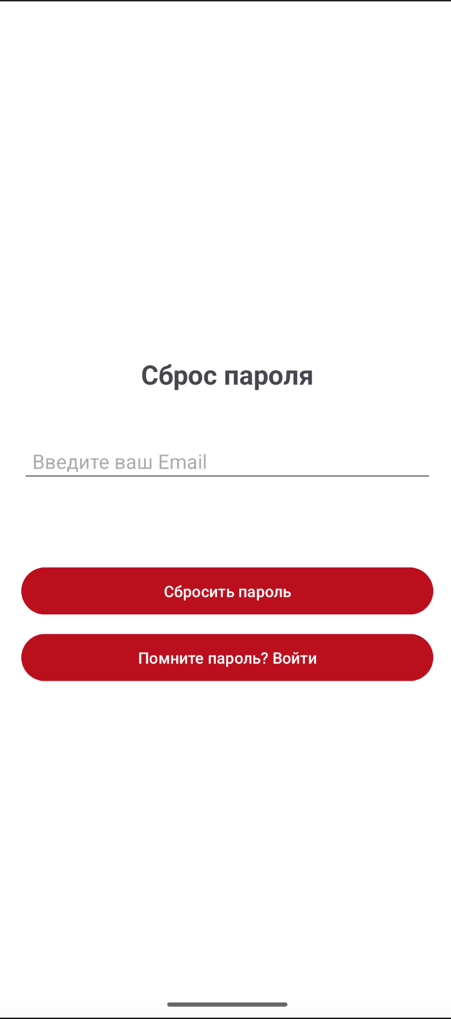


Рисунок 37 – Страница сброса пароля

## ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

## Инструментальные средства

Для хранения данных в мобильном приложении Drivee используется облачная база данных Firebase Firestore, которая предоставляет высокую производительность, масштабируемость и поддержку работы в реальном времени. Это решение обеспечивает надежное хранение информации о пользователях, объявлениях, избранных автомобилях, бортжурналах и другой сопутствующей информации.

В качестве среды разработки использовалась Android Studio, являющаяся стандартной IDE для создания мобильных приложений на платформе Android. Программная логика приложения реализована на языке Java, что позволило добиться стабильности, гибкости и удобной интеграции с Firebase-сервисами.

Разработка пользовательского интерфейса велась с использованием языка разметки XML, применяемого для создания всех экранов приложения: авторизации, регистрации, главной страницы, профиля пользователя, бортжурнала, страниц создания объявлений и фильтрации.

Для реализации работы с изображениями автомобилей и профилей применялись инструменты работы с Base64 кодированием, а для отображения изображений внутри элементов управления — адаптеры RecyclerView и ViewPager2.

Все сетевые взаимодействия, процессы авторизации, регистрации, восстановления пароля, создания объявлений, а также управление профилем были организованы через стандартные библиотеки Android SDK и интеграцию с Firebase Authentication, Firebase Firestore и Firebase Storage.

## Отладка программы

Отладка программы проводилась с использованием встроенных инструментов отладки, предоставляемых средой разработки Android Studio. Процесс отладки осуществлялся в режиме реального времени на эмуляторе

устройства Android и на реальных устройствах, что позволило эффективно отслеживать ошибки в логике приложения и корректировать их.

Основными типами ошибок в процессе тестирования являлись:

Ошибки привязки данных к элементам пользовательского интерфейса из-за неправильного указания идентификаторов id в файлах разметки XML;

Ошибки взаимодействия с базой данных Firebase, такие как неправильная структура запросов к коллекциям или попытки чтения несуществующих полей документов;

Ошибки обработки изображений, связанные с кодированием и декодированием изображений в формате Base64;

Ошибки навигации между активностями, вызванные неправильной передачей данных через Intent;

Ошибки валидации данных при регистрации, авторизации и создании объявлений.

Особое внимание при отладке уделялось проверке корректности выполнения запросов к серверу Firebase:

Чтение данных из коллекций объявлений и пользователей; Сохранение новых документов в базе данных;

Удаление объектов (объявлений, избранных машин, записей в бортжурнале);

Изменение данных профиля пользователя и объявлений.

Дополнительно выявленные ошибки сериализации объектов и несоответствия моделей данных были устранены путем согласования структуры моделей на стороне приложения с фактической структурой коллекций в Firebase.

## Защитное программирование

При разработке мобильного приложения Drivee были реализованы принципы защитного программирования для повышения устойчивости системы к некорректным действиям пользователей и сбоям в процессе работы.

Меры защитного программирования включали:

Проверка всех вводимых данных на соответствие требованиям: Ввод электронной почты проверяется на корректность формата.

Поля пароля проверяются на минимальную длину и наличие обязательных символов (цифры, буквы в разных регистрах).

Проверка на совпадение пароля и подтверждения пароля при регистрации.

При создании объявления все текстовые поля проверяются на заполненность, числовые поля — на корректность числового ввода.

Обработка ошибок сетевого взаимодействия с Firebase:

При потере подключения к интернету пользователю выводятся понятные сообщения об ошибке.

Повторная попытка выполнения операций при неудачных сетевых запросах.

Обработка возможных ошибок загрузки изображений:

Проверка наличия выбранного изображения перед его загрузкой.

Уведомление пользователя в случае неудачной загрузки или повреждения файла.

Защита от некорректной навигации:

Все переходы между активностями сопровождаются проверкой передаваемых данных (Intent extras).

При отсутствии необходимых данных активность корректно завершается с выводом сообщения пользователю.

Проверка прав доступа:

Перед выполнением действий, требующих специальных разрешений (например, вызов по телефону), выполняется проверка наличия разрешения.

При отсутствии разрешения запрашивается его предоставление у пользователя.

Обработка исключительных ситуаций (try-catch):

Основные операции взаимодействия с базой данных, обработка изображений и операций с файлами обернуты в блоки try-catch для перехвата возможных исключений без аварийного завершения приложения.

Применение данных принципов позволило значительно повысить надежность работы приложения, минимизировать количество аварийных завершений и обеспечить удобство для пользователей даже в непредвиденных ситуациях.

## Характеристики программы

В Таблице 7 представлена характеристика программы в виде перечисления всех модулей.

Таблица 7 - Характеристика программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Используемые библиотеки | Разме р файл а в  Кб | Допол нитель ные файлы |
| 1 | 2 | 3 | 4 | 5 |
| 1 | AndroidManifest.xml | <manifest xmlns:android="[http://schemas.an](http://schemas.an/) droid.com/apk/res/android" xmlns:tools="[http://schemas.andr](http://schemas.andr/) oid.com/tools">  <uses-feature android:name="android.hardware. telephony"  android:required="false" />  <uses-permission android:name="android.permissio n.CALL\_PHONE" />  <uses-permission android:name="android.permissio n.ACCESS\_FINE\_LOCATION"  />  <uses-permission android:name="android.permissio n.ACCESS\_COARSE\_LOCATIO  N" /> | 2 Кб | Нет |
| 2 | AdminActivity.java | import android.content.Intent; import android.os.Bundle; import android.util.Log; import  androidx.appcompat.app.AppCom patActivity; | 5 Кб | Нет |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Используемые библиотеки | Разме р файл  а в Кб | Допол нитель ные файлы |
| 1 | 2 | 3 | 4 | 5 |
|  |  | import androidx.recyclerview.widget.Lin earLayoutManager;  import androidx.recyclerview.widget.Rec yclerView;  import com.google.android.material.botto mnavigation.BottomNavigationVi ew;  import com.google.firebase.firestore.Fire baseFirestore;  import com.google.firebase.firestore.Que ryDocumentSnapshot;  import java.util.ArrayList;  import java.util.List; |  |  |
| 3 | AdminBortjournalActivity.ja va | import android.content.Intent; import android.os.Bundle; import android.view.View; import android.widget.Toast; import  androidx.appcompat.app.AppCom patActivity;  import androidx.recyclerview.widget.Lin earLayoutManager;  import androidx.recyclerview.widget.Rec yclerView;  import com.google.firebase.firestore.Fire baseFirestore;  import com.google.firebase.firestore.Que ryDocumentSnapshot;  import java.util.ArrayList; import java.util.List; | 4 Кб | Нет |
| 4 | AdminBortjournalAdapter.jav a | import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup;  import android.widget.Button; import android.widget.TextView; | 5 Кб | Нет |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Используемые библиотеки | Разме р файл  а в Кб | Допол нитель ные файлы |
| 1 | 2 | 3 | 4 | 5 |
|  |  | import androidx.annotation.NonNull; import androidx.recyclerview.widget.Rec yclerView;  import java.util.List; |  |  |
| 5 | AdminCarAdapter.java | import android.content.Context; import android.content.Intent; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import android.widget.Button; import android.widget.TextView; import android.widget.Toast; import androidx.annotation.NonNull; import androidx.recyclerview.widget.Rec yclerView;  import com.google.firebase.firestore.Fire baseFirestore;  import java.util.List; | 6 Кб | Нет |
| 6 | AllActivity.java | import android.content.Intent; import android.os.Bundle; import android.util.Log;  import android.view.MenuItem; import android.view.View; import android.widget.ArrayAdapter; import android.widget.Button; import android.widget.ImageButton; import android.widget.Spinner; import android.widget.TextView; import android.widget.Toast; import androidx.annotation.NonNull; import  androidx.appcompat.app.AppCom patActivity;  import  androidx.recyclerview.widget.Lin earLayoutManager; | 9 Кб | Нет |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Используемые библиотеки | Разме р файл  а в Кб | Допол нитель ные файлы |
| 1 | 2 | 3 | 4 | 5 |
|  |  | import androidx.recyclerview.widget.Rec yclerView;  import com.google.android.material.botto mnavigation.BottomNavigationVi ew;  import com.google.firebase.auth.Firebase Auth;  import com.google.firebase.firestore.Fire baseFirestore;  import com.google.firebase.firestore.Que ryDocumentSnapshot;  import java.util.ArrayList; import java.util.HashSet; import java.util.List;  import java.util.Map; import java.util.Set; |  |  |
| 7 | BortjournalActivity.java | import android.content.Intent; import android.os.Bundle; import android.widget.Button; import  androidx.appcompat.app.AppCom patActivity;  import androidx.recyclerview.widget.Lin earLayoutManager;  import androidx.recyclerview.widget.Rec yclerView;  import com.google.firebase.firestore.Fire baseFirestore;  import com.google.firebase.firestore.Que ry;  import com.google.firebase.firestore.Que ryDocumentSnapshot;  import java.util.ArrayList;  import java.util.List; | 2 Кб | Нет |
| 8 | BortjournalAdapter.java | import android.content.Context;  import android.content.Intent; | 3 Кб | Нет |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Используемые библиотеки | Разме р файл  а в Кб | Допол нитель ные файлы |
| 1 | 2 | 3 | 4 | 5 |
|  |  | import android.graphics.Bitmap; import android.graphics.BitmapFactory; import android.graphics.Canvas; import android.graphics.Paint; import android.graphics.PorterDuffXfer mode;  import android.graphics.Rect; import android.graphics.RectF; import android.graphics.PorterDuff; import android.util.Base64; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import android.widget.ImageView; import android.widget.TextView; import androidx.annotation.NonNull; import androidx.recyclerview.widget.Rec yclerView;  import com.google.firebase.firestore.Fire baseFirestore;  import java.util.List; |  |  |
| 9 | BortjournalDetailActivity.jav a | import android.os.Bundle;  import android.widget.TextView; import androidx.appcompat.app.AppCom  patActivity; | 2 Кб | Нет |
| 10 | BortjournalPost.java | package com.example.drivee; | 4 Кб | Нет |
| 11 | Car.java | import android.util.Log; import  com.google.android.gms.common  .api.internal.IStatusCallback; | 2 Кб | Нет |
| 12 | CarAdapter.java | import android.content.Context; import android.content.Intent; import android.graphics.Bitmap; import android.graphics.BitmapFactory;  import android.util.Base64; | 3 Кб | Нет |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Используемые библиотеки | Разме р файл  а в Кб | Допол нитель ные файлы |
| 1 | 2 | 3 | 4 | 5 |
|  |  | import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import android.widget.ImageButton; import android.widget.ImageView; import android.widget.TextView; import android.widget.Toast; import androidx.annotation.NonNull; import androidx.recyclerview.widget.Rec yclerView;  import com.google.firebase.auth.Firebase Auth;  import com.google.firebase.firestore.Fiel dValue;  import com.google.firebase.firestore.Fire baseFirestore;  import java.text.DecimalFormat;  import java.util.List; |  |  |
| 13 | CarAdapterUser.java | 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.Rec yclerView;  import com.bumptech.glide.Glide; import com.google.firebase.firestore.Fire baseFirestore;  import java.util.List; | 3 Кб | Нет |
| 14 | CarForProfile.java | Нет | 1 Кб | Нет |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Используемые библиотеки | Разме р файл  а в Кб | Допол нитель ные файлы |
| 1 | 2 | 3 | 4 | 5 |
| 15 | CenterActivity.java | import android.content.Intent; import android.content.pm.PackageMana ger;  import android.graphics.Bitmap; import android.graphics.BitmapFactory; import android.net.Uri;  import android.os.Bundle; import android.util.Base64; import android.util.Log; import android.widget.ImageView;  import android.widget.TextView; import android.widget.Toast; import androidx.appcompat.app.AppCom patActivity;  import androidx.viewpager2.widget.View Pager2;  import com.google.firebase.Timestamp; import com.google.firebase.firestore.Doc umentReference;  import com.google.firebase.firestore.Doc umentSnapshot;  import com.google.firebase.firestore.Fire baseFirestore;  import java.text.SimpleDateFormat; import java.util.ArrayList; import java.util.Date;  import java.util.List; import java.util.Locale; | 4 Кб | Нет |
| 16 | CreateListingActivity.java | package com.example.drivee; import android.Manifest; import  android.content.pm.PackageMana ger;  import androidx.core.app.ActivityCompa  t; | 8 Кб | Нет |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Используемые библиотеки | Разме р файл  а в Кб | Допол нитель ные файлы |
| 1 | 2 | 3 | 4 | 5 |
|  |  | import androidx.core.content.ContextCo mpat;  import com.google.android.gms.location. FusedLocationProviderClient; import com.google.android.gms.location. LocationServices;  import android.location.Address; import android.location.Geocoder; import java.util.Locale;  import android.app.Activity; import android.content.Intent; import android.content.pm.PackageMana ger;  import android.graphics.Bitmap; import android.graphics.BitmapFactory; import android.location.Address; import android.location.Geocoder; import android.net.Uri;  import android.os.Bundle; import android.util.Base64; import android.view.View; import android.widget.ArrayAdapter; import android.widget.Button; import android.widget.EditText; import android.widget.ImageView; import android.widget.Spinner; import android.widget.Toast; import androidx.annotation.NonNull; import androidx.annotation.Nullable; import  androidx.appcompat.app.AppCom patActivity;  import androidx.core.app.ActivityCompa  t; |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Используемые библиотеки | Разме р файл  а в Кб | Допол нитель ные файлы |
| 1 | 2 | 3 | 4 | 5 |
|  |  | import androidx.core.content.ContextCo mpat;  import com.google.firebase.auth.Firebase Auth;  import com.google.firebase.firestore.Fire baseFirestore;  import java.io.ByteArrayOutputStream; import java.io.InputStream; import java.util.ArrayList; import java.util.HashMap; import java.util.List;  import java.util.Locale; import java.util.Map; |  |  |
| 17 | CreatePostActivity.java | import android.os.Bundle; import android.text.TextUtils; import android.widget.ArrayAdapter; import android.widget.Button; import android.widget.EditText; import android.widget.Spinner; import android.widget.Toast; import androidx.annotation.Nullable; import  androidx.appcompat.app.AppCom patActivity;  import com.google.firebase.auth.Firebase Auth;  import com.google.firebase.firestore.Fiel dValue;  import com.google.firebase.firestore.Fire baseFirestore;  import java.util.ArrayList; import java.util.HashMap; import java.util.List;  import java.util.Map; | 3 Кб | Нет |
| 18 | FavoritesActivity.java | import android.content.Intent; import android.os.Bundle;  import android.util.Log; | 2 Кб | Нет |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Используемые библиотеки | Разме р файл  а в Кб | Допол нитель ные файлы |
| 1 | 2 | 3 | 4 | 5 |
|  |  | import android.view.View; import android.widget.TextView; import  androidx.appcompat.app.AppCom patActivity;  import androidx.recyclerview.widget.Lin earLayoutManager;  import androidx.recyclerview.widget.Rec yclerView;  import com.google.firebase.auth.Firebase Auth;  import com.google.firebase.firestore.Fire baseFirestore;  import java.util.ArrayList; import java.util.List; |  |  |
| 19 | FilterActivity.java | import android.content.Intent; import android.os.Bundle; import android.text.TextUtils; import android.widget.ArrayAdapter; import android.widget.Button; import android.widget.EditText;  import android.widget.Spinner; | 2 Кб | Нет |
| 20 | ImagePagerAdapter.java | import android.content.Context; import android.graphics.Bitmap; import android.graphics.BitmapFactory; import android.util.Base64; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import android.widget.ImageView; import androidx.annotation.NonNull; import  androidx.viewpager2.widget.View Pager2;  import androidx.recyclerview.widget.Rec  yclerView; | 2 Кб | Нет |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Используемые библиотеки | Разме р файл  а в Кб | Допол нитель ные файлы |
| 1 | 2 | 3 | 4 | 5 |
| 21 | MainActivity.java | import android.content.Intent; import android.os.Bundle; import android.view.View; import android.widget.EditText; import android.widget.Toast; import android.widget.Button; import  androidx.appcompat.app.AppCom patActivity;  import com.google.firebase.auth.Firebase Auth;  import com.google.firebase.auth.Firebase User;  import com.google.firebase.firestore.Doc umentSnapshot;  import  com.google.firebase.firestore.Fire baseFirestore; | 2 Кб | Нет |
| 22 | ManageModelsActivity.java | import android.os.Bundle; import android.util.Log; import android.view.View; import android.widget.\*; import  androidx.appcompat.app.AppCom patActivity;  import  com.google.firebase.firestore.\*; import java.util.\*; | 2 Кб | Нет |
| 23 | RegActivity.java | import android.content.Intent; import android.os.Bundle; import android.view.View; import android.widget.Button; import android.widget.EditText; import android.widget.Toast; import  androidx.appcompat.app.AppCom patActivity;  import com.google.firebase.auth.Firebase Auth;  import com.google.firebase.auth.Firebase  User; | 3 Кб | Нет |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Используемые библиотеки | Разме р файл  а в Кб | Допол нитель ные файлы |
| 1 | 2 | 3 | 4 | 5 |
|  |  | import com.google.firebase.firestore.Fire baseFirestore;  import java.util.HashMap; import java.util.Map; |  |  |
| 24 | ResetPasswordActivity.java | import android.os.Bundle; import android.view.View; import android.widget.Button; import android.widget.EditText; import android.widget.Toast; import  androidx.appcompat.app.AppCom patActivity;  import com.google.firebase.auth.Firebase  Auth; | 1 Кб | Нет |
| 25 | SelectBrandActivity.java | import android.content.Intent; import android.os.Bundle; import android.util.Log; import android.view.View; import android.widget.ArrayAdapter; import android.widget.Button; import android.widget.Spinner; import android.widget.Toast; import  androidx.appcompat.app.AppCom patActivity;  import com.google.firebase.firestore.Doc umentSnapshot;  import com.google.firebase.firestore.Fire baseFirestore;  import java.util.ArrayList;  import java.util.List; import java.util.Map; | 2 Кб | Нет |
| 26 | SelectModelActivity.java | import android.content.Intent; import android.os.Bundle; import android.util.Log; import android.view.View; import android.widget.ArrayAdapter; import android.widget.Button;  import android.widget.Spinner; import android.widget.Toast; | 2 Кб | Нет |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Используемые библиотеки | Разме р файл  а в Кб | Допол нитель ные файлы |
| 1 | 2 | 3 | 4 | 5 |
|  |  | import androidx.appcompat.app.AppCom patActivity;  import com.google.firebase.firestore.Doc umentSnapshot;  import com.google.firebase.firestore.Fire baseFirestore;  import java.util.ArrayList; import java.util.List;  import java.util.Map; |  |  |
| 27 | SplashActivity.java | import android.content.Intent; import android.os.Bundle; import android.os.Handler; import  androidx.appcompat.app.AppCom patActivity; | 1 Кб | Нет |
| 28 | User.java | Нет | 1 Кб | Нет |
| 29 | UserCar.java | Нет | 1 Кб | Нет |
| 30 | UserProfileActivity.java | import android.content.Intent; import android.graphics.Bitmap; import android.graphics.BitmapFactory; import android.net.Uri;  import android.os.Bundle; import android.util.Base64; import android.util.Log; import android.widget.Button; import android.widget.ImageView;  import android.widget.TextView; import android.widget.Toast; import androidx.appcompat.app.AppCom patActivity;  import androidx.recyclerview.widget.Lin earLayoutManager;  import androidx.recyclerview.widget.Rec yclerView;  import com.google.firebase.auth.Firebase  Auth; | 3 Кб | Нет |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Используемые библиотеки | Разме р файл  а в Кб | Допол нитель ные файлы |
| 1 | 2 | 3 | 4 | 5 |
|  |  | import com.google.firebase.firestore.Fire baseFirestore;  import com.google.firebase.firestore.Que ryDocumentSnapshot;  import java.io.ByteArrayOutputStream; import java.io.FileNotFoundException; import java.io.InputStream;  import java.util.ArrayList; import java.util.List; |  |  |

## ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта на тему «Разработка мобильного приложения "Drivee" для публикации и поиска автомобильных объявлений» было разработано мобильное приложение Drivee на языке программирования Java с использованием среды разработки Android Studio и облачной базы данных Firebase Firestore.

Целью проекта являлось упрощение процессов покупки, продажи автомобилей, управления объявлениями и взаимодействия между пользователями в мобильной среде.

В процессе выполнения курсового проекта были сформированы требования к системе, спроектированы архитектура приложения и базы данных, разработан пользовательский интерфейс и реализован основной функционал, включая авторизацию, регистрацию, создание, поиск и фильтрацию объявлений, работу с избранным, создание записей в бортжурнале, а также административные функции для управления контентом. Для хранения данных применялась облачная СУБД Firebase Firestore,

для интеграции карт и геолокации — Yandex MapKit. Для оформления документации использовался текстовый редактор Microsoft Word 2016. База данных размещалась в облаке, что позволило обеспечить доступ к данным в реальном времени.

В процессе разработки возникали ошибки, связанные с обработкой сетевых запросов, сериализацией данных и обработкой изображений. Ошибки устранялись через дополнительную проверку данных, отладку сетевых взаимодействий и оптимизацию механизмов загрузки изображений.

Результаты тестирования показали, что основной функционал приложения работает стабильно, корректно обрабатывает пользовательский ввод, обеспечивает выполнение бизнес-процессов, предусмотренных техническим заданием. Имеются некоторые незначительные возможности для дальнейшего улучшения интерфейса и расширения возможностей фильтрации.

В целом, мобильное приложение Drivee полностью соответствует заявленным требованиям, демонстрирует стабильную работу, удобство использования и готово к эксплуатации пользователями.

## СПИСОК ИСПОЛЬЗОВАННЫХ МАТЕРИАЛОВ

1. Создание адаптивного интерфейса для Android-приложений.

URL: <https://developer.android.com/guide/topics/ui/declaring-layout> (Дата обращения: 01.04.2025)

1. Руководство по работе с Firebase Firestore в Android. URL: https://firebase.google.com/docs/firestore/quickstart (Дата обращения: 01.04.2025)
2. Авторизация и регистрация через Firebase Authentication. URL: https://firebase.google.com/docs/auth/android/start (Дата обращения: 02.04.2025)
3. Настройка восстановления пароля в Firebase. URL: https://firebase.google.com/docs/auth/android/manage- users#send\_a\_password\_reset\_email (Дата обращения: 05.04.2025)
4. Добавление ViewPager2 и перелистывания фотографий в Android. URL: <https://developer.android.com/guide/navigation/navigation-swipe-view-2> (Дата обращения: 05.04.2025)
5. Работа с RecyclerView: отображение списков в Android. URL: <https://developer.android.com/guide/topics/ui/layout/recyclerview> (Дата обращения: 06.04.2025)
6. Добавление избранного с помощью Firebase. URL: [https://stackoverflow.com/questions/48650898/add-to-favorites-in-firebase-](https://stackoverflow.com/questions/48650898/add-to-favorites-in-firebase-android) [android](https://stackoverflow.com/questions/48650898/add-to-favorites-in-firebase-android) (Дата обращения: 08.04.2025)
7. Создание фильтрации данных в приложении Android. URL: [https://stackoverflow.com/questions/24090968/filtering-recyclerview-using-](https://stackoverflow.com/questions/24090968/filtering-recyclerview-using-a-searchview) [a-searchview](https://stackoverflow.com/questions/24090968/filtering-recyclerview-using-a-searchview) (Дата обращения: 09.04.2025)

## Интеграция карт и геолокации с помощью Yandex MapKit. URL: https://yandex.ru/dev/maps/mobile-

sdk/doc/android/quickstart/concepts/android-quickstart-docpage/ (Дата обращения: 09.04.2025)

1. Подключение и настройка Bottom Navigation View. URL: <https://developer.android.com/guide/navigation/navigation-ui> (Дата обращения: 12.04.2025)

## Как работать с изображениями в Firebase Storage и отображать их в приложении.

URL: https://firebase.google.com/docs/storage/android/start (Дата обращения: 01.04.2025)

1. Права доступа в Android (запрос на доступ к звонкам, памяти и интернету). URL: <https://developer.android.com/training/permissions/requesting> (Дата обращения: 15.04.2025)
2. Отправка звонка с приложения Android. URL: [https://stackoverflow.com/questions/4275679/how-to-make-a-phone-call-in-](https://stackoverflow.com/questions/4275679/how-to-make-a-phone-call-in-android) [android](https://stackoverflow.com/questions/4275679/how-to-make-a-phone-call-in-android) (Дата обращения: 01.04.2025)
3. Создание форм регистрации с проверкой введённых данных. URL: <https://developer.android.com/guide/topics/ui/form-validation> (Дата обращения: 15.04.2025)
4. Работа с фрагментами в Android и их навигацией. URL: <https://developer.android.com/guide/fragments> (Дата обращения: 15.04.2025)
5. Создание custom-адаптеров для RecyclerView. URL: <https://developer.android.com/guide/topics/ui/layout/recyclerview#java> (Дата обращения: 15.04.2025)
6. Основы архитектуры MVVM в Android. URL: <https://developer.android.com/jetpack/guide> (Дата обращения: 15.04.2025)

## Работа с облачными функциями в Firebase для сложной логики. URL: https://firebase.google.com/docs/functions (Дата обращения: 15.04.2025)

1. Интеграция платежных систем в Android-приложения. URL: <https://developer.android.com/google/play/billing> (Дата обращения: 16.04.2025)
2. Рекомендации по производительности мобильных приложений. URL: <https://developer.android.com/topic/performance> (Дата обращения: 16.04.2025)

ПРИЛОЖЕНИЕ А. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

АННОТАЦИЯ

В данном программном документе приведено техническое задание на разработку программы для покупки и продажи автомобилей Drivvy.

В данном программном документе, в разделе «Введение» указано наименование, краткая характеристика области применения программы (программного изделия).

В разделе «Основания для разработки» указаны документы, на основании которых ведется разработка, наименование и условное обозначение темы разработки.

В данном программном документе, в разделе «Назначение разработки» указано функциональное и эксплуатационное назначение программы (программного изделия).

Раздел «Требования к программе» содержит следующие подразделы:

* требования к функциональным характеристикам;
* требования к надежности;
* условия эксплуатации;
* требования к составу и параметрам технических средств;
* требования к информационной и программной совместимости;
* специальные требования.

В данном программном документе, в разделе «Требования к программной документации» указаны предварительный состав программной документации и специальные требования к ней.

В разделе «Технико-экономические показатели» указаны: ориентировочная экономическая эффективность, предполагаемая годовая потребность, экономические преимущества разработки.

В данном программном документе, в разделе «Стадии и этапы

## разработки» установлены необходимые стадии разработки, этапы и содержание работ.

В разделе «Порядок контроля и приемки» должны быть указаны виды испытаний и общие требования к приемке работы.

## СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ
   1. [Наименование программы 5](#_bookmark32)
   2. [Краткая характеристика области применения программы 5](#_bookmark33)
2. [ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ 6](#_bookmark34)
   1. [Основание для проведения разработки 6](#_bookmark35)
   2. [Наименование и условное обозначение для разработки 6](#_bookmark36)
3. [НАЗНАЧЕНИЕ РАЗРАБОТКИ 7](#_bookmark37)
   1. [Функциональное назначение программы 7](#_bookmark38)
   2. [Эксплуатационное назначение программы 7](#_bookmark39)
4. [ТРЕБОВАНИЯ К ПРОГРАММЕ 8](#_bookmark40)
   1. [Требования к функциональным характеристикам 8](#_bookmark41)
   2. [Требования к надежности 14](#_bookmark42)
   3. [Условия эксплуатации 15](#_bookmark43)
   4. [Требования к составу и параметрам технических средств 15](#_bookmark44)
   5. [Требования к информационной и программной совместимости 16](#_bookmark45)

[4.7. Специальные требования 16](#_bookmark46)

1. [ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ 17](#_bookmark47)
   1. [Предварительный состав программной документации 17](#_bookmark48)
   2. [Специальные требования к программной документации 17](#_bookmark49)
2. [ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 19](#_bookmark50)
   1. [Ориентировочная экономическая эффективность 19](#_bookmark51)
   2. [Предполагаемая годовая потребность 19](#_bookmark52)
   3. [Экономические преимущества разработки 19](#_bookmark53)
3. [СТАДИИ И ЭТАПЫ РАЗРАБОТКИ 20](#_bookmark54)
   1. [Стадии разработки 20](#_bookmark55)
   2. [Этапы разработки 20](#_bookmark56)
   3. [Содержание работ по этапам 20](#_bookmark57)
   4. [Исполнители 21](#_bookmark58)
4. [ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ 22](#_bookmark59)
   1. [Порядок контроля 22](#_bookmark60)
   2. [Порядок приемки 22](#_bookmark61)
5. ВВЕДЕНИЕ

## Наименование программы

Наименование - «Разработка мобильного приложения для продажи и покупки автомобилей».

## Краткая характеристика области применения программы

Приложение предназначено к применению покупки и продаж автомобилей.

## ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

## Основание для проведения разработки

Основанием для проведения разработки является задание по курсовому проекту по МДК 01.01 «Разработка программных модулей».

## Наименование и условное обозначение для разработки

Наименование темы разработки - «Мобильное приложение покупки и продаж автомобилей».

## НАЗНАЧЕНИЕ РАЗРАБОТКИ

## Функциональное назначение программы

Функциональным назначением программы является обеспечение удобного и безопасного взаимодействия между продавцами и покупателями транспортных средств. Она позволяет размещать и просматривать объявления, фильтровать предложения по различным параметрам, а также связываться с другими пользователями напрямую.

## Эксплуатационное назначение программы

Программа предназначена для эксплуатации в подразделениях, занимающихся куплей-продажей автомобилей, таких как автосалоны, дилерские центры, а также частными продавцами и покупателями (далее – Пользователь).

Конечным пользователем программы будут:

1. Владельцы автомобилей, желающие продать свой транспорт
2. Покупатели, ищущие автомобиль по заданным параметрам
3. Автодилеры и коммерческие организации, работающие в сфере торговли транспортными средствами

## ТРЕБОВАНИЯ К ПРОГРАММЕ

## Требования к функциональным характеристикам

* + 1. Требования к составу выполняемых функций

Программа должна обеспечивать выполнение следующих функций:

Регистрация и авторизация пользователей: создание учетной записи. Авторизация через логин и пароль, Восстановление доступа к аккаунту.

Управление объявлениями: размещение объявлений о продаже автомобилей, удаление объявлений, добавление фото, описания и характеристик авто.

Управление записями: размещение записей о авто, добавление описания заголовков и пробега авто.

Поиск и фильтрация: поиск автомобилей по марке, модели, цене, году выпуска, пробегу и другим параметрам, фильтрация объявлений по заданным критериям, сортировка результатов поиска.

Связь между пользователями: активный номер телефона продавца.

Безопасность и проверка данных: модерация объявлений для предотвращения мошенничества.

* + 1. Требования к переменным

Имена переменных должны быть осмысленными и отражать их назначение. Необходимо использовать стиль camelCase (например, productPrice, userLogin).

Тип данных переменной должен точно соответствовать её содержанию: числовые значения (цена, пробег) хранятся как int или float, текстовые данные (модель, марка, цвет) — как string, логические значения (например, isVerified) — как boolean. Все переменные должны быть инициализированы перед использованием. При работе с чувствительными данными, такими как контактная информация пользователей, необходимо учитывать вопросы безопасности и конфиденциальности.

4.1.3 Требования к функциям приложения

Приложение должно обеспечивать выполнение следующих функций:

1. Регистрация и авторизация пользователей

Создание аккаунта (продавец, покупатель, автодилер).

Восстановление доступа через e-mail или номер телефона. Авторизация по почте и паролю.

1. Управление объявлениями.

Размещение объявлений с фото, описанием и характеристиками автомобиля.

Удаление объявлений.

1. Управление записями.

Размещение записей с заголовком, описанием и пробегом автомобиля.

1. Поиск и фильтрация.

Поиск автомобилей по марке, модели, году выпуска, цене, пробегу и другим параметрам.

Фильтрация объявлений по типу кузова, городу и другим критериям.

Сортировка результатов поиска по дате публикации, цене, пробегу и популярности и т.д.

1. Безопасность и проверка данных.

Модерация объявлений и записей бортжурнала для предотвращения мошенничества.

Защита данных пользователей (шифрование, безопасное хранение паролей).

1. Управление личным кабинетом.

Личный кабинет пользователя с историей просмотров, избранными объявлениями.

* + 1. Требования к окнам приложения

Окно авторизации и регистрации должно содержать: авторизацию, регистрацию, сброс пароля.

Главное окно должно содержать: фильтрацию и поиск по объявлениям, вывод всех объявлений, навигация по приложению (аккаунт, избранное, главная страница)

Окно бортжурнала должно содержать: создание записи в бортжурнал, просмотр записей бортжурнала.

Окно аккаунта должно содержать: возможность просмотра фотографии профиля, изменения фотографии пользователя, удаления фотографии пользователя, вывод почты аккаунта, вывод имя аккаунта, вывод номера телефона аккаунта, вывод объявлений созданных пользователем включая статус его создания, возможность удаления созданного пользователем объявления.

Окно избранного должно содержать вывод всех объявлений приложения, которые пользователь добавил путем нажатия в объявлении на кнопку звездочки.

Окно администратора должно содержать: возможность просматривать объявления, модерировать их (удалять и принимать), добавлять марки машин, добавлять модели машин по маркам, переходить на страницу всех.

* + 1. Требования к ролям пользователей Администратор:

Имеет возможность принять или удалить объявление пользователя. Имеет возможность принять или удалить запись в бортжурнал пользователя

Пользователь:

Может просматривать каталог автомобилей, фильтровать их по различным параметрам (марка, модель, цена, год выпуска и т.д.). Может

добавить автомобиль в избранное. Может создать, редактировать и удалить свое объявление о продаже

* + 1. Требования к добавлению объявления:

Пользователи должны иметь возможность добавлять объявления о продаже автомобиля. Все обязательные поля должны быть четко обозначены и обязательными для заполнения. Все поля должны иметь валидацию для предотвращения заполнения пользователем некорректной формы данных.

Поля:

Марка и модель автомобиля – выбираются на страницах выбора автомобиля при создании объявления.

Год выпуска – от 1800 до настоящего года. Числовой тип данных. VIN – строго 17 символов.

Пробег – числовой тип данных. Цена – числовой тип данных.

Описание автомобиля – текстовое поле для подробного описания состояния, характеристик и особенностей автомобиля.

Фотографии – минимум 3 фотографии, включая переднюю, боковую и заднюю части автомобиля. Фотографии должны быть загружены в стандартных форматах (JPEG, PNG).

Цвет автомобиля – только цвет.

Тип кузова – седан, хэтчбек, внедорожник и другие.

Тип двигателя – бензиновый, дизельный, электрический и т.д.

Технические характеристики – мощность двигателя (числовой тип данных), трансмиссия, тип привода (передний, полный

* + 1. Правила оформления и модерации:

Все объявления должны соответствовать установленным правилам (например, запрет на ложную информацию, запрещенные товары или услуги).

После отправки, объявление должно пройти модерацию администратором, в ходе которой проверяется корректность данных, фотографии и описание.

* + 1. Требования к смене пароля:

Доступность функции смены пароля:

Функция смены пароля должна быть доступна для всех авторизованных пользователей через страницу авторизации и регистрации, на которой необходимо нажать на кнопку «сброс пароля», после чего ввести почту ранее созданного аккаунта, на которую приходит ссылка на WEB-страницу сброса пароля.

Требования к новому паролю:

Новый пароль должен быть не менее 8 символов.

Все данные, связанные с сменой пароля, должны передаваться по защищенному каналу (например, через HTTPS).

Пароли должны храниться в зашифрованном виде в базе данных, с использованием современных алгоритмов хеширования.

* + 1. Процесс регистрации пользователя:

Новый пользователь должен иметь возможность зарегистрироваться приложении через специальную форму регистрации. Регистрация должна быть доступна через основные способы:

Электронная почта и пароль. Поля для регистрации:

Электронная почта – для подтверждения учетной записи и связи с пользователем.

Пароль – новый пользователь должен создать безопасный пароль (с требованиями безопасности, как указано в разделе "Требования к смене пароля").

Имя – для идентификации на платформе.

Номер телефона - для связи с пользователем других пользователей. Необязательные поля для заполнения.

Возможность добавить фото профиля (фотография не является обязательной)

Валидация данных:

Все введенные данные должны проверяться на корректность:

Электронная почта должна быть в правильном формате (например, [user@example.com).](mailto:user@example.com)

Пароль должен соответствовать минимальным требованиям безопасности (не менее 8 символов).

Номер телефона (если используется) должен быть в правильном формате (с кодом страны и без ошибок).

В случае ошибок валидации, система должна уведомить Требования к модерации объявлений и отзывов

При создании объявления или записи в бортжурнал пользователем они отправляются на модерацию (на аккаунт администратора). Администратор в праве принять или удалить объявление или запись без обоснования причин.

* + 1. Требования к авторизации

Авторизация должна выполняться через страницу авторизации и регистрации и должна включать в себя ввод почты, а так же пароля (указываются при регистрации).

* + 1. Требования к модерации объявления.

Возможность просматривать объявления, модерировать их (удалять и принимать), добавлять марки машин, добавлять модели машин по маркам, переходить на страницу всех машин.

* + 1. Требования к подключению базы данных / API

Подключение к базе данных (далее - БД), происходит на уровне работы самого приложения. Роль БД выполняет FireStore Database (FireBase)

* + 1. Требования к дизайну приложения

Простота и удобство: Минимальное количество шагов для выполнения

основных действий.

Адаптивность: Корректное отображение на мобильных устройствах и планшетах.

Современный стиль: Использование актуальных UI/UX трендов, четкие шрифты, удобные кнопки.

## Требования к надежности

* + 1. Требования к обеспечению надежного (устойчивого) функционирования программы

Надежное (устойчивое) функционирование программы должно быть обеспечено организацией бесперебойного питания технических средств.

* + 1. Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств не должно превышать времени, необходимого на перезагрузку операционной системы и запуск программы, при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

* + 1. Отказы из-за некорректных действий оператора

Отказы программы возможны вследствие некорректных действий пользователя при взаимодействии с операционной и информационной системой.

Во избежание возникновения отказов программы по указанной выше причине следует провести инструктаж с конечным пользователем о правильном эксплуатации информационной системы и убедиться в том, что конечный пользователь имеет опыт использования ЭВМ.

## Условия эксплуатации

* + 1. Климатические условия эксплуатации

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

* + 1. Требования к видам обслуживания

Требования к видам обслуживания не предъявляются.

* + 1. Требования к численности и квалификации персонала

Сотрудники должны иметь базовое умение работы с персональным компьютером и таблицами.

## Требования к составу и параметрам технических средств

В состав технических средств для разработки и эксплуатации мобильного приложения Drivee должны входить следующие компоненты:

Мобильные устройства: Приложение должно поддерживаться на смартфонах и планшетах с операционными системами Android.

Разрешение экрана: Минимальное разрешение экрана для Android устройств — 480x720 точек1.

Оперативная память (ОЗУ): Не менее 512 МБ на мобильном устройстве.

Дисковое пространство: Не менее 200 МБ на мобильном устройстве.

Для обеспечения работы серверной части приложения необходимы:

Сервер базы данных: Система должна использовать надежную базу данных – FireStore Database (Firebase).

Процессор сервера: Процессор с тактовой частотой не менее 2 ГГц. Оперативная память сервера: Объем ОЗУ не менее 8 ГБ.

Жесткий диск сервера: Объем дискового пространства не менее 500 ГБ SSD.

Система должна поддерживать подключение по Gigabit Ethernet или Wi-Fi с скоростью передачи данных не ниже 100 Мбит/с4.

## Требования к информационной и программной совместимости

* + 1. Требования к информационным структурам и методам решения

Требования к информационным структурам на входе и выходе, а также к методам решения не предъявляются.

* + 1. Требование к исходному коду и языкам программирования

Исходные коды программы должны быть реализованы на языке Java и Xml. В качестве интегрированной среды разработки программы должна быть использована среда Android Studio (локализованная, русская версия).

* + 1. Требования к программным средствам, используемым программой На компьютере должна быть операционная система Windows 10 или Windows 11, а также запущен сервер с базой данных FireStore Database

(Firebase).

* 1. Требования к защите информации и программ

Требования к защите информации и программ не предъявляются.

## Специальные требования

Специальные требования не предъявляются.

## ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

## Предварительный состав программной документации

Состав программной документации должен включать в себя:

1. Приложение А. Техническое задание;
2. Пояснительная записка;
3. Приложение Б. Сценарий и результаты тестовых испытаний;
4. Приложение В. Руководство пользователя;
5. Приложение Г. Текст программы;

## Специальные требования к программной документации

Техническое задание оформлялось по ГОСТам: 19.201-78 и 19.106-

78.

Поля макета технической записки:

* правое поле – 20 мм,
* левое поле – 35 мм
* верхнее поле – 25 мм,
* нижнее поле – 20 мм.

Текст должен быть отпечатан машинным способом на принтере на

одной стороне листа.

Шрифт должен быть с "засечками". Рекомендуется Times New Roman. Цвет шрифта должен быть черным.

Размер шрифта (его высота в кеглях):

|  |  |
| --- | --- |
| - для основного текста: | 14, |
| - для заголовка | 14, |
| - для таблиц, схем и приложений | 12, |
| - для колонтитулов (к каждому листу П3) | 12. |

Межстрочный интервал равен:

* для основного текста 1,5;
* для таблиц, схем и приложений 1.

Текст содержит отступ каждого абзаца - 1,25 см.

Основной текст работы (проекта) печатается 1,5 междустрочным интервалом компьютерного набора.

Выравнивание текста устанавливается «По ширине страницы».

## ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

## Ориентировочная экономическая эффективность

Ориентировочная экономическая эффективность не рассчитывается.

## Предполагаемая годовая потребность

Предполагаемая годовая потребность не рассчитывается.

## Экономические преимущества разработки

Экономические преимущества разработки не рассчитываются.

## СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

## Стадии разработки

Разработка должна быть проведена в соответствии со следующими стадиями:

1. Анализ предметной области. Разработка технического задания;
2. Проектирование. Раздел 2. Специальная часть ПЗ;
3. Рабочий проект;
4. Тестирование. Разработка сценария и результатов тестирования.
5. Внедрение. Разработка Руководства пользователя и Текста программы.

## Этапы разработки

Сроки сдачи этапов разработки формируются исходя из требований заказчика.

Окончательный срок сдачи всего проекта – 16.04.2025.

## Содержание работ по этапам

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1. Постановка задачи;
2. Определение и уточнение требований к техническим средствам;
3. Определение требований к программе;
4. Определение стадий, этапов и сроков разработки программы и документации на неё;
5. Выбор языков программирования;
6. согласование и утверждение технического задания.

На этапе проектирования должны быть выполнены перечисленные ниже работы:

1. Предварительная разработка входных и выходных данных;
2. Уточнение методов решения задачи;
3. Разработка общего описания алгоритма решения задач;
4. Согласование и утверждения схем проектирования.

На этапе подготовки готового программного проекта должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию.

На этапе разработки пояснительной записки должны быть выполнены перечисленные ниже работы:

1. Общая часть;
2. Специальная часть;
3. Технологическая часть;
4. Заключение.

На этапе разработки результатов тестовых испытаний должны быть выполнены перечисленные ниже виды работ:

1. Разработка, согласование и утверждение программы и методики испытаний;
2. Проведение приемо-сдаточных испытаний;
3. Корректировка программы и программной документации по результатам испытаний.

На этапе разработки руководства пользователя должны быть описаны следующие пункты:

1. Описание установки информационной системы;
2. Описание эксплуатации информационной системы;
3. Описание деинсталляции информационной системы.

На этапе подготовки скрипта базы данных и текста программы необходимо выполнить работу по созданию соответствующего приложения.

## Исполнители

Руководитель разработки Горбунова М.А.

Исполнитель Пушкин И.А.

## ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

## Порядок контроля

Контроль выполнения работ проводится исполнителем проекта и заказчиком проекта в момент завершения каждого этапа /стадии.

Стадии разработки проекта указаны в разделе «7.1 Стадии разработки».

## Порядок приемки

Исполнитель передает заказчику техническую документацию и разработанный проект в электронном виде.

Все документы должны быть подписаны исполнителем проекта и утверждены заказчиком.

Состав документации определен в разделе 5.1. «Предварительный состав программной документации».

ПРИЛОЖЕНИЕ Г. СЦЕНАРИЙ И РЕЗУЛЬТАТЫ ТЕСТОВЫХ

ИСПЫТАНИЙ

АННОТАЦИЯ

В данном программном документе приведен сценарий тестовых испытаний и результаты тестовых испытаний на проверку соответствия разрабатываемой программы на соответствие функционалу по техническому заданию.

В разделе «Цель испытаний» указана цель проведения испытаний. В разделе «Требования к программе» указаны требования,

подлежащие проверке во время испытаний и заданные в пояснительной записке на программу.

В данном программном документе, в разделе «Требования к программной документации» указан состав программной документации, предъявляемый на испытания.

В разделе «Средства и порядок испытаний» указаны технические и программные средства, используемые во время испытаний, а также порядок проведения испытаний.

В разделе «Методы испытаний» приведено описание используемых методов испытаний.

В разделе «Тестовые примеры» приведены таблицы с результатами тестовых испытаний.

В данном программном документе, в разделе «Результаты тестирования», представлены таблицы с примерами результатами тестовых испытаний.

# ОБЪЕКТ ИСПЫТАНИЙ

* 1. Наименование объекта

Наименование – «Разработка мобильного приложения для продажи и покупки автомобилей».

* 1. Область применения объекта

Программа Мобильное приложение «Drivee» предназначено для автоматизации процессов публикации, поиска и управления объявлениями о продаже автомобилей. Разработанное решение позволяет пользователям удобно размещать свои объявления, находить интересующие автомобили по заданным параметрам, а также вести бортовой журнал эксплуатации транспортного средства. Приложение упрощает взаимодействие между продавцами и покупателями автомобилей, обеспечивая быструю публикацию объявлений и удобный поиск по множеству критериев.

* 1. Обозначение испытуемой программы. Обозначение – «Drivee».

# ЦЕЛЬ ИСПЫТАНИЙ

Целью проведения испытаний является проверка соответствия разработанной программы требованиям, изложенным в документе

«Пояснительная записка». Испытания будут созданы на основе обязательного для разработки функционала.

# ТРЕБОВАНИЯ К ПРОГРАММЕ

## Схема тестирования

На Рисунке 1 представлена схема тестирования настольного приложения «Drivee». Схема разделена на функционально тестирование, тестирование производительности и проверку интерфейса.

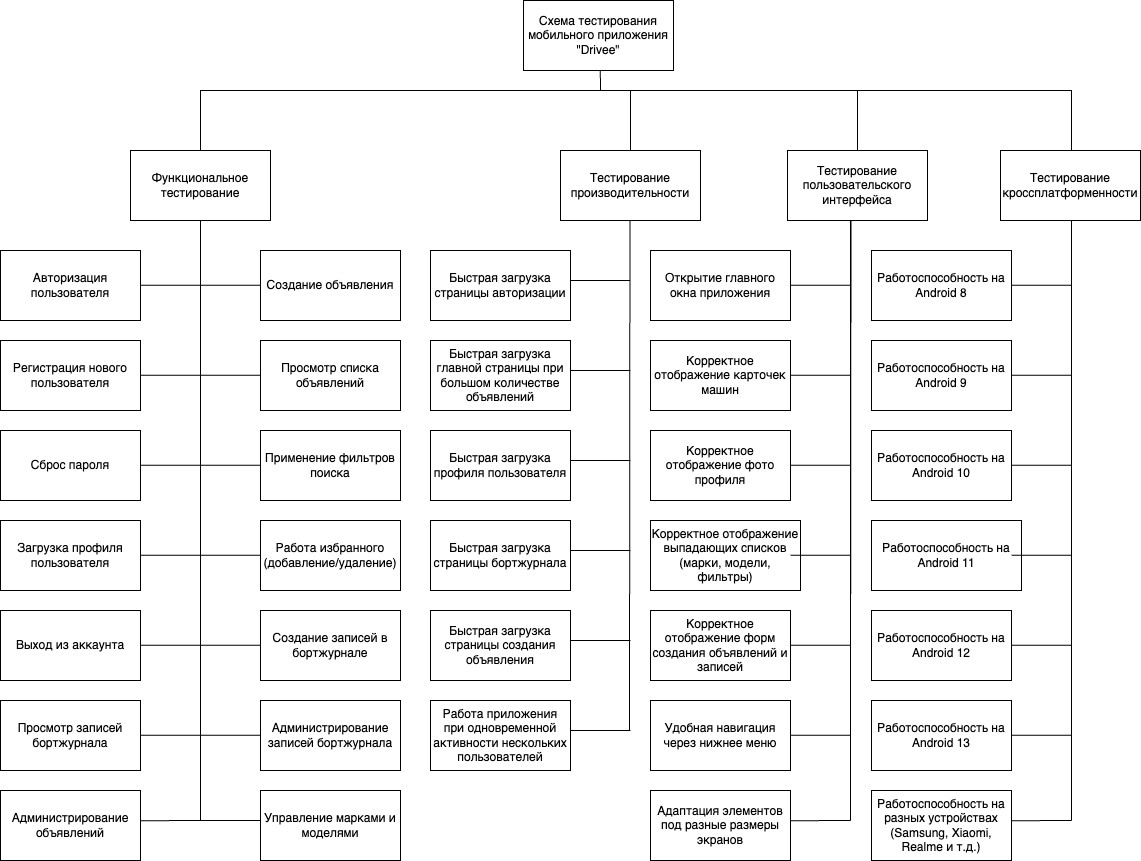


Рисунок 1 - Схема тестирования приложения

* 1. Требования, подлежащие проверке
     + Возможность запуска приложения на слабом устройстве;
     + Возможность работы приложения несколькими пользователями одновременно (регистрация/авторизация);
     + Возможность работы приложения одним пользователем (личный кабинет, профиль, объявления);
     + Возможность регистрации нового пользователя;
     + Возможность авторизации зарегистрированного пользователя;
     + Возможность сброса пароля пользователем;
     + Возможность создания нового объявления;
     + Возможность загрузки и отображения фото профиля;
     + Возможность редактирования фото профиля;
     + Возможность удаления своих объявлений;
     + Возможность работы с избранным (добавление/удаление машины в избранное);
     + Возможность создания записи в бортжурнале;
     + Возможность просмотра и открытия записи бортжурнала;
     + Возможность применения фильтров поиска машин (фильтрация по параметрам);
     + Возможность сортировки списка машин (по цене, по пробегу, по дате);
     + Возможность расширенного поиска машин по дополнительным параметрам;
     + Возможность выхода из аккаунта;
     + Возможность модерации объявлений администратором (подтверждение/удаление);
     + Возможность создания марок и моделей авто через админ- панель;
     + Возможность модерации записей бортжурнала администратором;
     + Возможность смены настроек профиля пользователя (фото, email, телефон);
     + Работоспособность программы на разных версиях Android (8, 9, 10, 11, 12, 13);
     + Работоспособность на разных устройствах (Samsung, Xiaomi, Realme, и др.).

# СРЕДСТВА И ПОРЯДОК ИСПЫТАНИЙ

## Состав технических средств, используемых для проведения тестировании информационной системы представлен в Таблице 1.

Таблица 1 - Технические средства

|  |  |  |
| --- | --- | --- |
| № | Тип оборудования | Наименование оборудования |
| 1 | 2 | 3 |
| 1 | Смартфон | Samsung Galaxy S21 |
| 2 | Экран | 6.2" Dynamic AMOLED 2X |
| 3 | Разрешение экрана | 2400×1080 пикселей |
| 4 | Процессор | Exynos 2100 / Snapdragon 888 |
| 5 | Количество ядер процессора | 8 |
| 6 | Оперяативная память | 8 ГБ LPDDR5 |
| 7 | Конфигурация накопителей | 128 ГБ встроенной памяти |
| 8 | ОС устройства | Android 13 |

## Состав программных средств, используемых для проведения тестировании информационной системы представлен в Таблице 2.

Таблица 2 - Программные средства

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тип средства | Название средства | Назначение |
| 1 | 2 | 3 | 4 |
| 1 | Среда разработки | Android Studio Hedgehog (2023.1.1) | Разработка и тестирование  мобильного приложения |

Испытания должны проводиться поэтапно согласно разделу

«Требования к программе» настоящего документа.

# МЕТОДЫ ИСПЫТАНИЙ

* 1. По формальности тестирования.

Тестирование по тестам – тестирование по предварительно написанным тест-кейсам.

* 1. По исполнению кода.

Динамическое тестирование - во время тестирования код исполняется.

* 1. По уровню тестирования.

Системное – проверка работы всей системы на соответствие заявленным требованиям к программному продукту.

* 1. По целям.

Функциональное тестирование направлено на проверку того, какие функции ПО реализованы, и того, насколько верно они реализованы.

* 1. По степени автоматизации.

Ручное – тестировщик проверяет программу вручную.

* 1. По позитивности сценария.

Позитивным – проверка ПО на соответствие ожидаемому поведению.

Негативным – проверяет, будет ли ПО работать в случае, когда поведение пользователя отличается от ожидаемого.

* 1. По знанию системы.

Тестирование «белого ящика» и «черного ящика» – тестирование с доступом и без доступа к коду продукта.

* 1. По разработке тестовых испытаний.

На основе требований – требование было определено до начала тестирования.

# ТЕСТОВЫЕ ПРИМЕРЫ

Все тестовые данные для проведения тестирования приложения по созданию типизированных документов представлены в Таблицах 3-14. Для именования видов тестирования их обозначения будут сокращены до первых букв названия:

* Тестирование функциональной корректности (Functional correctness) - FC;
* Тестирование производительности (Performance testing) – P;
* Тестирование графического интерфейса (GUI testing) - GUI;
* Тестирование кроссплатформенности (Cross platform testing)

– CP.

* 1. Проверка авторизации пользователя

Для прохождения процедуры необходимо ввести корректные логин и пароль. При успешной авторизации пользователь должен попасть на

главную страницу.

Таблица 3 – Авторизация

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | FC | Ввод валидного email и  пароля | Пользователь успешно входит  в систему |
| 2 | FC | Ввод невалидного  email/пароля | Отображается сообщение об  ошибке входа |

## Проверка регистрации нового пользователя

Для прохождения процедуры необходимо заполнить все обязательные поля корректными данными и нажать кнопку "Зарегистрироваться".

Таблица 4 – Регистрация

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | FC | Ввод корректных данных в  форму | Успешная регистрация и  переход к авторизации |
| 2 | FC | Ввод некорректного email | Отображение сообщения об  ошибке формата email |

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 3 | FC | Ввод несовпадающих  паролей | Отображение сообщения об  ошибке пароля |

## Проверка сброса пароля

Для прохождения процедуры необходимо ввести зарегистрированный email и запросить ссылку на сброс пароля. На email должно прийти письмо.

Таблица 5 – Сброс пароля

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | FC | Ввод существующего email  для сброса | Получение письма для смены  пароля |
| 2 | FC | Ввод несуществующего  email для сброса | Отображение ошибки "Email  не найден" |

## Проверка загрузки профиля пользователя

При входе в систему должна корректно загружаться фотография профиля, имя, email и телефон пользователя.

Таблица 6 - Профиль

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | FC | Открытие страницы  профиля | Отображение имени, номера  телефона, email и фото |

## Проверка выхода из аккаунта

При нажатии на кнопку выхода пользователь должен вернуться на страницу авторизации.

Таблица 7 - Выход

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | FC | Нажатие кнопки "Выйти из  аккаунта" | Открытие страницы  авторизации |

## Проверка создания объявления

Для создания объявления необходимо заполнить все обязательные поля и прикрепить хотя бы одно фото.

Таблица 8 – Выход из аккаунта

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | FC | Заполнение всех полей объявления | Успешное создание  объявления и переход на главную страницу |
| 2 | FC | Пропуск обязательного  поля | Отображение сообщения об  ошибке |

## Проверка просмотра списка объявлений

После авторизации или сброса фильтров должен корректно загружаться список всех активных объявлений.

Таблица 9 – Список объявлений

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | FC | Открытие списка  объявлений | Отображение всех объявлений |

## Проверка применения фильтров поиска

При выборе параметров фильтрации должны корректно отбираться машины по заданным критериям.

Таблица 10 - Фильтры

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | FC | Применение фильтра по  марке и модели | Отображение только  соответствующих объявлений |
| 2 | FC | Применение фильтра по  цене | Отображение объявлений в  указанном диапазоне |

## Проверка работы избранного

При нажатии на иконку избранного объявление должно добавляться/удаляться из списка избранных.

Таблица 11 - Избранное

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | FC | Добавление объявления в  избранное | Объявление добавлено в  список избранного |
| 2 | FC | Удаление объявления из  избранного | Объявление удалено из списка  избранного |

## Проверка создания записей в бортжурнале

После заполнения всех обязательных полей в бортжурнале запись должна успешно создаться.

Таблица 12 – Создание записи

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | FC | Заполнение всех полей  записи | Создание новой записи в  бортжурнале |

## Проверка просмотра записей бортжурнала

При открытии бортжурнала должны корректно отображаться все опубликованные записи.

Таблица 13 – Просмотр записей

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | FC | Открытие списка записей | Отображение всех записей  бортжурнала |

## Проверка администрирования объявлений

Администратор должен иметь возможность подтверждать или удалять объявления.

Таблица 14 – Администрирование объявлений

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | FC | Нажать "Принять" на  объявлении | Статус объявления меняется на  "approved" |
| 2 | FC | Нажать "Удалить" на объявлении | Объявление удаляется из базы данных |

## Проверка администрирования записей бортжурнала

Администратор может подтверждать или удалять записи в бортжурнале.

Таблица 15 – Администрирование записей

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | FC | Нажать "Принять" на  записи | Запись становится видимой  для всех |
| 2 | FC | Нажать "Удалить" на  записи | Запись удаляется из базы  данных |

## Проверка управления марками и моделями

Администратор может добавлять новые марки и модели автомобилей.

Таблица 16 – Управление марками и моделями

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | FC | Добавить новую марку | Добавить новую модель к  марке |
| 2 | FC | Добавить новую модель к  марке | Модель добавляется к  выбранной марке |

## Быстрая загрузка страницы авторизации

Для прохождения процедуры необходимо получить время отклика от сервера, которое не должно превышать 0.5 секунд. Кроме этого, мобильное устройство (МУ) должно иметь стабильное подключение к интернету.

Таблица 17 - Быстрая загрузка страницы авторизации

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | P | Открытие страницы  авторизации | Загрузка менее 2 секунд |

## Быстрая загрузка главной страницы при большом количестве объявлений

В базе данных размещено не менее 100 объявлений. Пользователь успешно вошел в приложение. При переходе на главную страницу все объявления должны корректно отображаться.

Таблица 18 - Быстрая загрузка главной страницы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | P | Открытие главной страницы с 500+  объявлениями | Загрузка менее 3 секунд |

## Быстрая загрузка профиля пользователя

Пользователь должен быть авторизован в системе. При переходе на вкладку "Профиль" должны загружаться актуальные данные (имя, телефон, email).

Таблица 19 - Быстрая загрузка профиля пользователя

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | P | Открытие профиля  пользователя | Загрузка менее 2 секунд |

## Быстрая загрузка страницы бортжурнала

Пользователь должен быть авторизован. В базе данных должны быть размещены записи бортжурнала. При переходе на вкладку "Бортжурнал" все записи должны быстро загружаться без видимых задержек.

Таблица 20 - Быстрая загрузка страницы бортжурнала

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | P | Открытие страницы  бортжурнала | Загрузка менее 2 секунд |

* 1. Быстрая загрузка страницы создания объявления

Пользователь должен быть авторизован. При нажатии на кнопку

"Создать объявление" приложение должно быстро открывать страницу создания объявления.

Таблица 21 - Быстрая загрузка страницы создания объявления

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | P | Открытие страницы  создания объявления | Загрузка менее 2 секунд |

## Проверка работы приложения при активности нескольких пользователей

Несколько пользователей одновременно используют приложение: публикуют объявления, добавляют машины в избранное, редактируют профиль. Сервер должен корректно обрабатывать все запросы без ошибок.

Таблица 22 – Несколько активных пользователей

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | P | Одновременная работа 10 пользователей | Корректная работа без ошибок |

## Открытие главного окна приложения

Приложение установлено и запускается впервые. Пользователь видит страницу авторизации при открытии.

Таблица 23 - Открытие главного окна приложения

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | GUI | Открытие главной  страницы | Корректное отображение  элементов |

## Корректное отображение карточек машин

На главной странице приложения должны быть загружены объявления о продаже автомобилей. Каждая карточка должна содержать фото, название машины, пробег, цену, статус (если в профиле).

Таблица 24 - Отображение карточек машин

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | GUI | Просмотр карточек на главной странице | Правильное отображение фото, цены, марки, модели,  пробега |

## Корректное отображение фото профиля

Пользователь загрузил фото профиля. Фото должно отображаться круглым, без обрезания, без искажений.

Таблица 25 - Отображение фото профиля

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | GUI | Загрузка фото профиля | Отображение фото в виде  круга |

## Корректное отображение выпадающих списков

В системе заданы марки и модели автомобилей. Фильтры поиска должны быть настроены. При нажатии на выпадающий список корректно открывается список доступных опций.

Таблица 26 – Выпадающие списки

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | GUI | Открытие списка марок и  моделей | Отображение корректного  списка без ошибок |

* 1. Корректное отображение форм создания объявлений и записей Пользователь должен иметь возможность создать новое объявление

или запись в бортжурнале. Все поля формы должны быть видимыми и

правильно расположены.

Таблица 27 – Формы создания объявлений

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | GUI | Открытие страницы  создания объявления | Корректная разметка всех  полей |

## Удобная навигация через нижнее меню

В приложении должно быть настроено нижнее меню навигации (главная, избранное, создать объявление, бортжурнал, профиль).

Таблица 28 – Навигация

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | GUI | Нажатие на кнопки  нижнего меню | Быстрый переход на  соответствующие экраны |

## Адаптация элементов под разные размеры экранов

Приложение должно корректно отображать весь контент на разных устройствах с различными размерами экранов (смартфоны, планшеты).

Таблица 29 - Адаптивность

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | GUI | Открытие на устройстве с  маленьким экраном | Корректное отображение всех  элементов |
| 2 | GUI | Открытие на устройстве с  большим экраном | Отсутствие растягивания и  наложения |

## Проверка работоспособности на Android 8

Приложение установлено на устройстве с Android 8. Все основные функции должны работать без ошибок: авторизация, регистрация, публикация объявлений, фильтрация машин, бортжурнал.

Таблица 30 - Android 8

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | CP | Установка приложения на  Android 8 | Корректная работа  приложения |

## Проверка работоспособности на Android 9

Приложение установлено на устройстве с Android 9. Все функции работают корректно без вылетов и ошибок.

Таблица 31 - Android 9

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | CP | Установка приложения на  Android 9 | Корректная работа  приложения |

## Проверка работоспособности на Android 10

Приложение установлено на устройстве с Android 10. Ожидается корректная работа интерфейса и функций.

Таблица 32 - Android 10

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | CP | Установка приложения на  Android 10 | Корректная работа  приложения |

## Проверка работоспособности на Android 11

Приложение установлено на устройстве с Android 11. Проверка стабильности работы всех страниц и модулей.

Таблица 33 - Android 11

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | CP | Установка приложения на  Android 11 | Корректная работа  приложения |

## Проверка работоспособности на Android 12

Приложение должно корректно работать на Android 12, без багов и вылетов.

Таблица 34 - Android 12

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | CP | Установка приложения на  Android 12 | Корректная работа  приложения |

## Проверка работоспособности на Android 13

Приложение должно быть совместимо с последней версией Android 13, поддерживая все функции.

Таблица 35 - Android 13

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | CP | Установка приложения на  Android 13 | Корректная работа  приложения |

## Проверка работоспособности на разных устройствах

Приложение тестируется на различных брендах устройств для проверки универсальности и совместимости.

Таблица 36 - Проверка работоспособности на разных устройствах

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вид  тестирования | Действие (входное  значение) | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | CP | Установка на устройства  Samsung, Xiaomi, Realme | Корректная работа  приложения |

# РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Тестирование проводится по тестовым данным, представленным в Таблицах 3-36. Результаты тестирования представлены в Таблицах 37-70.

* 1. Проверка авторизации пользователя

Для прохождения процедуры необходимо ввести корректные логин и пароль. При успешной авторизации пользователь должен попасть на

главную страницу.

Таблица 37 – Авторизация

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Открыть приложение, ввести Email [aa@gmail.com,](mailto:aa@gmail.com) ввести Пароль xyzzytui,  нажать кнопку "Войти" | Открывается главная страница пользователя | Открывается  главная страница пользователя | Пройден |
| 2 | Открыть приложение, ввести Email  [qq@gmail.com,](mailto:qq@gmail.com) ввести Пароль xyzzytui, нажать кнопку "Войти" | Открывается панель администратора | Открывается панель администратора | Пройден |
| 3 | Открыть приложение, ввести Email [aa@gmail.com,](mailto:aa@gmail.com) ввести неверный пароль wrongpass, нажать  кнопку "Войти" | Появляется ошибка  "Неверный email или пароль" | Появляется ошибка "Неверный email или пароль" | Пройден |

## Проверка регистрации нового пользователя

Для прохождения процедуры необходимо заполнить все обязательные поля корректными данными и нажать кнопку "Зарегистрироваться".

Таблица 38 – Регистрация

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Нажать кнопку  "Зарегистрироваться", ввести Имя Иван Иванов, | Открывается страница авторизации | Открывается страница авторизации | Пройден |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
|  | дату рождения 01.01.2000,  телефон +79991234567, email [ivan@example.com,](mailto:ivan@example.com) пароль Password123, подтверждение пароля Password123, нажать кнопку  "Зарегистрироваться" |  |  |  |
| 2 | Нажать кнопку  "Зарегистрироваться", ввести Имя Анна Петрова, дату рождения 05.05.1998,  телефон +79991112233, email [anna@example.com,](mailto:anna@example.com) пароль Password123, подтверждение пароля  Password321, нажать кнопку "Зарегистрироваться" | Появляется сообщение "Пароли не совпадают" | Появляется сообщение "Пароли не совпадают" | Пройден |
| 3 | Нажать кнопку  "Зарегистрироваться", ввести Имя Иван Иванов, дату рождения 01.01.2000,  телефон +79991234567, email [aa@gmail.com,](mailto:aa@gmail.com) пароль Password123, подтверждение пароля Password123, нажать кнопку  "Зарегистрироваться" | Появляется сообщение "Пользователь с таким email уже зарегистрирован" | Появляется сообщение "Пользователь с таким email уже зарегистрирован" | Пройден |

## Проверка сброса пароля

Для прохождения процедуры необходимо ввести зарегистрированный email и запросить ссылку на сброс пароля. На email должно прийти письмо.

Таблица 39 – Сброс пароля

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | На странице авторизации нажать "Сбросить пароль", ввести Email [aa@gmail.com,](mailto:aa@gmail.com) нажать кнопку "Сбросить пароль" | Приходит ссылка на почту для сброса пароля | Приходит ссылка на почту для сброса пароля | Пройден |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 2 | Перейти по ссылке из письма, ввести новый пароль  Newpass123, подтвердить новый пароль Newpass123, нажать кнопку "Сбросить пароль" | Появляется сообщение "Пароль успешно изменен" | Появляется сообщение "Пароль успешно изменен" | Пройден |
| 3 | На странице сброса пароля нажать "Помните пароль? Войти" | Открывается страница авторизации | Открывается страница авторизации | Пройден |

## Проверка загрузки профиля пользователя

При входе в систему должна корректно загружаться фотография профиля, имя, email и телефон пользователя.

Таблица 40 - Профиль

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Авторизоваться в системе,  перейти на страницу "Профиль" | Отображаются имя пользователя, номер  телефона, email и фотография профиля | Отображаются имя пользователя, номер телефона, email и фотография профиля | Пройден |

## Проверка выхода из аккаунта

При нажатии на кнопку выхода пользователь должен вернуться на страницу авторизации.

Таблица 41 - Выход

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Перейти на страницу "Профиль", нажать | Открывается страница авторизации | Открывается страница авторизации | Пройден |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
|  | кнопку "Выйти из аккаунта" |  |  |  |

## Проверка создания объявления

Для создания объявления необходимо заполнить все обязательные поля и прикрепить хотя бы одно фото.

Таблица 42 – Выход из аккаунта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Нажать кнопку "Создать объявление", выбрать марку Toyota, нажать  кнопку "Далее" | Открывается выбор модели | Открывается выбор модели | Пройден |
| 2 | Выбрать модель Camry, нажать кнопку "Далее" | Открывается форма создания объявления | Открывается форма создания объявления | Пройден |
| 3 | Загрузить 5 фотографий машины, заполнить поля: VIN 123456789, год  выпуска 2020, цвет Черный, объем двигателя 2.5, мощность двигателя  200, пробег 30000,  количество владельцев 1,  цена 2500000, описание Отличное состояние, город Москва, выбрать тип кузова Седан, выбрать трансмиссию Автомат, выбрать топливо Бензин, выбрать привод Передний,  нажать кнопку "Создать объявление" | Объявление успешно создается и пользователь переходит на главную страницу | Объявление успешно создается и пользователь переходит на главную страницу | Пройден |

## Проверка просмотра списка объявлений

После авторизации или сброса фильтров должен корректно загружаться список всех активных объявлений.

Таблица 43 – Список объявлений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Открыть главную страницу | Загружается список всех доступных  объявлений | Загружается список всех доступных объявлений | Пройден |

## Проверка применения фильтров поиска

При выборе параметров фильтрации должны корректно отбираться машины по заданным критериям.

Таблица 44 - Фильтры

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | На главной странице выбрать марку  Toyota, модель Camry | Отображаются только объявления Toyota Camry | Отображаются только объявления Toyota Camry | Пройден |
| 2 | Нажать кнопку "Параметры",  установить фильтры: год выпуска от 2015 до 2021, цена от 1000000 до 3000000,  нажать кнопку "Применить" | Отображаются только объявления, соответствующие указанным  фильтрам | Отображаются только объявления, соответствующие указанным фильтрам | Пройден |

## Проверка работы избранного

При нажатии на иконку избранного объявление должно добавляться/удаляться из списка избранных.

Таблица 45 - Избранное

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | На карточке объявления нажать на  кнопку | Машина добавляется в раздел  "Избранное" | Машина добавляется в раздел "Избранное" | Пройден |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие  (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
|  | "Избранное" |  |  |  |
| 2 | В разделе "Избранное" нажать еще раз на кнопку "Избранное" на машине | В разделе "Избранное" нажать еще раз на кнопку "Избранное" на машине | Машина удаляется из избранного | Пройден |

## Проверка создания записей в бортжурнале

После заполнения всех обязательных полей в бортжурнале запись должна успешно создаться.

Таблица 46 – Создание записи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Перейти в раздел "Бортжурнал", нажать кнопку "Создать запись" | Открывается форма создания  записи | Открывается форма создания записи | Пройден |
| 2 | Выбрать марку Toyota, выбрать модель Camry, ввести заголовок Плановое ТО, ввести текст записи  Замена масла, ввести пробег 30000, нажать кнопку "Опубликовать" | Запись  сохраняется и отображается в списке | Запись  сохраняется и отображается в списке | Пройден |

## Проверка просмотра записей бортжурнала

При открытии бортжурнала должны корректно отображаться все опубликованные записи.

Таблица 47 – Просмотр записей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Перейти в раздел  "Бортжурнал" | Отображаются все созданные  записи | Отображаются все созданные записи | Пройден |
| 2 | Нажать на любую запись в  списке | Открывается подробная информация о  записи | Открывается подробная информация о записи | Пройден |

## Проверка администрирования объявлений

Администратор должен иметь возможность подтверждать или удалять объявления.

Таблица 48 – Администрирование объявлений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие  (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Авторизоваться как администратор, перейти в  раздел  "Модерация объявлений" | Загружается список неподтвержденных объявлений | Загружается список неподтвержденных объявлений | Пройден |
| 2 | Нажать кнопку "Принять" на объявлении | Статус объявления меняется на "approved" | Статус объявления меняется на "approved" | Пройден |
| 3 | Нажать кнопку  "Удалить" на объявлении | Объявление  удаляется из базы данных | Объявление удаляется из базы данных | Пройден |

## Проверка администрирования записей бортжурнала

Администратор может подтверждать или удалять записи в бортжурнале.

Таблица 49 – Администрирование записей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Авторизоваться как администратор, перейти в  раздел "Бортжурнал" | Загружается список записей, ожидающих модерации | Загружается список записей, ожидающих модерации | Пройден |
| 2 | Нажать кнопку "Принять" на записи | Статус записи меняется на  "Одобрено" | Статус записи меняется на "Одобрено" | Пройден |
| 3 | Нажать кнопку "Удалить" на записи | Запись удаляется из базы  данных | Запись удаляется из базы данных | Пройден |

z

* 1. Проверка управления марками и моделями

Администратор может добавлять новые марки и модели автомобилей.

Таблица 50 – Управление марками и моделями

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Перейти в раздел "Марки и модели", ввести новую марку  Honda, нажать кнопку "Добавить марку" | Марка Honda добавляется в список | Марка Honda добавляется в список | Пройден |
| 2 | Выбрать марку Honda, ввести модель Accord, нажать кнопку "Добавить модель" | Модель Accord появляется в списке моделей выбранной  марки | Модель Accord появляется в списке  моделей выбранной марки | Пройден |

## Быстрая загрузка страницы авторизации

Для прохождения процедуры необходимо получить время отклика от сервера, которое не должно превышать 0.5 секунд. Кроме этого, мобильное устройство (МУ) должно иметь стабильное подключение к интернету.

Таблица 51 - Быстрая загрузка страницы авторизации

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Открыть приложение, перейти на экран авторизации | Страница авторизации загружается менее чем за 1 секунду | Страница авторизации загружается менее чем за 1 секунду | Пройден |

## Быстрая загрузка главной страницы при большом количестве объявлений

В базе данных размещено не менее 100 объявлений. Пользователь успешно вошел в приложение. При переходе на главную страницу все объявления должны корректно отображаться.

Таблица 52 - Быстрая загрузка главной страницы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Авторизоваться как пользователь, перейти на  главную страницу | Главная страница с 50+  объявлениями загружается  менее чем за 2 секунды | Главная страница с 50+ объявлениями загружается менее чем за 2 секунды | Пройден |

## Быстрая загрузка профиля пользователя

Пользователь должен быть авторизован в системе. При переходе на вкладку "Профиль" должны загружаться актуальные данные (имя, телефон, email).

Таблица 53 - Быстрая загрузка профиля пользователя

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Авторизоваться как  пользователь, | Страница профиля  загружается | Страница профиля загружается менее чем за  1 секунду | Пройден |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие  (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
|  | нажать на  вкладку "Профиль" | менее чем  за 1 секунду |  |  |

## Быстрая загрузка страницы бортжурнала

Пользователь должен быть авторизован. В базе данных должны быть размещены записи бортжурнала. При переходе на вкладку "Бортжурнал" все записи должны быстро загружаться без видимых задержек.

Таблица 54 - Быстрая загрузка страницы бортжурнала

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие  (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Перейти на вкладку "Бортжурнал"  в нижнем меню | Страница бортжурнала загружается  менее чем за 2 секунды | Страница бортжурнала загружается менее чем за 2 секунды | Пройден |

## Быстрая загрузка страницы создания объявления

Пользователь должен быть авторизован. При нажатии на кнопку "Создать объявление" приложение должно быстро открывать страницу создания объявления.

Таблица 55 - Быстрая загрузка страницы создания объявления

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Перейти на  вкладку "Создать объявление", выбрать марку и модель | Страница создания объявления открывается  менее чем за 1 секунду | Страница создания объявления открывается  менее чем за 1 секунду | Пройден |

## Проверка работы приложения при активности нескольких пользователей

Несколько пользователей одновременно используют приложение: публикуют объявления, добавляют машины в избранное, редактируют профиль. Сервер должен корректно обрабатывать все запросы без ошибок.

Таблица 56 – Несколько активных пользователей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Одновременно с разных аккаунтов загрузить  главную страницу | Все пользователи видят данные без подвисаний или ошибок | Все пользователи видят данные без подвисаний или ошибок | Пройден |
| 2 | Одновременно 5  пользователей создают  объявления | Все  объявления успешно создаются без ошибок  сохранения | Все объявления успешно создаются без ошибок сохранения | Пройден |

## Открытие главного окна приложения

Приложение установлено и запускается впервые. Пользователь видит страницу авторизации при открытии.

Таблица 57 - Открытие главного окна приложения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие  (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Установить и открыть приложение | Корректное отображение стартового  окна (авторизация) | Корректное отображение стартового окна (авторизация) | Пройден |

## Корректное отображение карточек машин

На главной странице приложения должны быть загружены объявления о продаже автомобилей. Каждая карточка должна содержать фото, название машины, пробег, цену, статус (если в профиле).

Таблица 58 - Отображение карточек машин

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | № | Действие (входное  значение) | Ожидаемый результат | Фактический результат |
| 1 | 1 | 2 | 3 | 4 |
| 1 | Перейти на главную страницу после авторизации | Карточки машин  отображаются с  изображением, маркой, моделью, ценой,  пробегом | Карточки машин отображаются с  изображением, маркой, моделью, ценой, пробегом | Пройден |

## Корректное отображение фото профиля

Пользователь загрузил фото профиля. Фото должно отображаться круглым, без обрезания, без искажений.

Таблица 59 - Отображение фото профиля

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие  (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Загрузить фото профиля через "Выбрать  фото" | Фото  отображается в профиле в круглом  формате | Фото отображается в профиле в круглом формате | Пройден |

## Корректное отображение выпадающих списков

В системе заданы марки и модели автомобилей. Фильтры поиска должны быть настроены. При нажатии на выпадающий список корректно открывается список доступных опций.

Таблица 60 – Выпадающие списки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Нажать на выпадающий список  марки | Список отображается и позволяет | Список отображается и позволяет выбрать марку | Пройден |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие  (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
|  | автомобиля | выбрать  марку |  |  |
| 2 | Нажать на выпадающий список модели  автомобиля | Список отображается и меняется в зависимости от  выбранной марки | Список отображается и меняется в зависимости от выбранной марки | Пройден |
| 3 | Нажать на фильтр цвета, кузова, топлива | Выпадающие списки  открываются корректно,  доступны все значения | Выпадающие списки  открываются корректно, доступны все значения | Пройден |

* 1. Корректное отображение форм создания объявлений и записей Пользователь должен иметь возможность создать новое объявление

или запись в бортжурнале. Все поля формы должны быть видимыми и

правильно расположены.

Таблица 61 – Формы создания объявлений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие  (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Перейти на страницу создания объявления | Все поля для заполнения отображаются корректно | Все поля для заполнения отображаются корректно | Пройден |
| 2 | Перейти на страницу создания записи в бортжурнал | Поля заголовка, текста,  выбора авто  отображаются корректно | Поля заголовка, текста, выбора авто  отображаются корректно | Пройден |

## Удобная навигация через нижнее меню

В приложении должно быть настроено нижнее меню навигации (главная, избранное, создать объявление, бортжурнал, профиль).

Таблица 62 – Навигация

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Нажать на кнопки "Главная", "Избранное", "Создать", "Бортжурнал", "Профиль" в  нижнем меню | Открываются соответствующие страницы без задержек и ошибок | Открываются соответствующие страницы без задержек и ошибок | Пройден |

## Адаптация элементов под разные размеры экранов

Приложение должно корректно отображать весь контент на разных устройствах с различными размерами экранов (смартфоны, планшеты).

Таблица 63 - Адаптивность

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Открыть приложение на разных устройствах (смартфон, планшет) | Элементы интерфейса корректно  масштабируются, тексты не  обрезаются | Элементы интерфейса корректно  масштабируются,  тексты не обрезаются | Пройден |

## Проверка работоспособности на Android 8

Приложение установлено на устройстве с Android 8. Все основные функции должны работать без ошибок: авторизация, регистрация, публикация объявлений, фильтрация машин, бортжурнал.

Таблица 64 - Android 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Установить приложение на  устройство с | Приложение устанавливается успешно | Приложение устанавливается успешно | Пройден |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие  (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
|  | Android 8 |  |  |  |
| 2 | Открыть приложение и выполнить авторизацию | Окно авторизации и главная страница  работают корректно | Окно авторизации и главная страница  работают корректно | Пройден |
| 3 | Проверить работу фильтров, профиля, избранного | Все функции работают без сбоев | Все функции работают без сбоев | Пройден |

## Проверка работоспособности на Android 9

Приложение установлено на устройстве с Android 9. Все функции работают корректно без вылетов и ошибок.

Таблица 65 - Android 9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие  (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Установить приложение на устройство с Android 9 | Приложение устанавливается успешно | Приложение устанавливается успешно | Пройден |
| 2 | Открыть приложение и войти | Стартовая страница отображается  корректно | Стартовая страница отображается корректно | Пройден |
| 3 | Проверить создание объявления и запись в бортжурнал | Все функции работают корректно | Все функции работают корректно | Пройден |

## Проверка работоспособности на Android 10

Приложение установлено на устройстве с Android 10. Ожидается корректная работа интерфейса и функций.

Таблица 66 - Android 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие  (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Установить  приложение на Android 10 | Установка  проходит без ошибок | Установка проходит без ошибок | Пройден |
| 2 | Авторизоваться в системе | Открывается главная  страница | Открывается главная страница | Пройден |
| 3 | Проверить переходы между  страницами | Навигация работает без сбоев | Навигация работает без сбоев | Пройден |

## Проверка работоспособности на Android 11

Приложение установлено на устройстве с Android 11. Проверка стабильности работы всех страниц и модулей.

Таблица 67 - Android 11

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие  (входное значение) | Ожидаемый результат | Фактический результат | Статус теста  (пройден/не пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Установить приложение на Android  11 | Приложение устанавливается | Приложение устанавливается | Пройден |
| 2 | Войти в систему | Данные  пользователя загружаются | Данные пользователя загружаются | Пройден |
| 3 | Выполнить создание объявления | Объявление успешно добавляется | Объявление успешно добавляется | Пройден |

## Проверка работоспособности на Android 12

Приложение должно корректно работать на Android 12, без багов и вылетов.

Таблица 68 - Android 12

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Установить приложение на  Android 12 | Процесс установки  завершен | Процесс установки завершен | Пройден |
| 2 | Авторизоваться | Вход выполнен  корректно | Вход выполнен корректно | Пройден |
| 3 | Проверить работу фильтров и избранного | Фильтрация и  добавление в избранное работают  корректно | Фильтрация и  добавление в избранное работают корректно | Пройден |

## Проверка работоспособности на Android 13

Приложение должно быть совместимо с последней версией Android 13, поддерживая все функции.

Таблица 69 - Android 13

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Установить приложение на  Android 13 | Приложение запускается  без ошибок | Приложение запускается без  ошибок | Пройден |
| 2 | Авторизация пользователя | Вход в аккаунт без  сбоев | Вход в аккаунт без сбоев | Пройден |
| 3 | Проверить  создание записи в бортжурнал | Запись  успешно сохраняется | Запись успешно сохраняется | Пройден |

## Проверка работоспособности на разных устройствах

Приложение тестируется на различных брендах устройств для проверки универсальности и совместимости.

Таблица 70 - Проверка работоспособности на разных устройствах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Действие (входное  значение) | Ожидаемый результат | Фактический результат | Статус теста (пройден/не  пройден) |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Установить приложение на Samsung, Xiaomi, Realme | Установка на устройства Samsung, Xiaomi, Realme | Установка на устройства Samsung, Xiaomi, Realme | Пройден |
| 2 | Открыть приложение и авторизоваться | Приложение стабильно  работает на всех  устройствах | Приложение стабильно работает на всех устройствах | Пройден |
| 3 | Проверить создание и просмотр  объявлений | Создание и просмотр  работают  корректно | Создание и просмотр работают корректно | Пройден |

# ОЦЕНКА КАЧЕСТВА ПРОЕКТА

## Для оценки качества программного продукта были использованы требования и качества оценки программного обеспечения, согласно ГОСТ Р ИСО/МЭК 25010-2015. Оценка функциональной надежности приведена в Таблице 27. Нумерация пунктов стандартов происходит согласно определениям в пункте 4 ГОСТа, указанного выше.

Таблица 71 - Отображение функциональной надежности

|  |  |  |  |
| --- | --- | --- | --- |
| № | Пункт ГОСТ 25010-  2015 | Настоящий стандарт | Функциональная надежность |
| 1 | 2 | 3 | 4 |
| 1 | 4.1.2 | Производительность | Нагрузочное тестирование функциональной части программы показало незначительные  различия с обычным тестированием в категории времени отклика сервера |
| 2 | 4.1.4 | Свобода от риска | Программа не защищена от критических ошибок, приводящих к выходу из приложения |
| 4 | 4.2.1 | Функциональная пригодность | Решение полностью раскрывает  представленную в теме проблему создания документации |
| 6 | 4.2.3 | Совместимость | Программа не вызывает конфликтов с другими  программами, а также работает на нескольких операционных системах |
| 7 | 4.2.4 | Удобство использования | Для удобства использования реализованы подсказки и справочник внутри программы |
| 8 | 4.2.5 | Надежность | Решение предоставляет защиту всех данных пользователя и обеспечивает целостность  данных при их отправке с клиента на сервер. |
| 9 | 4.2.5.2 | Готовность | Программный продукт полностью готов, функционал не нуждается в изменениях |
| 10 | 4.2.6.1 | Конфиденциальность | Данные пользователей шифруются как на стороне клиента, так и на стороне сервера |
| 11 | 4.2.7 | Сопровождаемость | Фукнционал нуждается в сопровождении двумя квалифицированными в разработке  программных продуктов сотрудниками для починки возникающих ошибок. Не реализована система обновления версий для изменения  программы. |
| 12 | 4.2.8 | Переносимость | Программа подразумевает функционал для  переноса данных пользователя с одного устройства на другое |

После проведения оценки качества программного продукта было выяснено, что программный продукт производителен, функционально пригоден, совместим со средой эксплуатации, удобен, надёжен, защищён и переносим.

В целом выполненное решение по покупке и продаж автомобилей оценивается как качественное и рекомендовано к распространению среди конечных пользователей программы.

# ПРИЛОЖЕНИЕ Д. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ АННОТАЦИЯ

В данном программном документе приведено руководство пользователя на разработку программы для поддержания работы мобильного приложения «Drivee».

В данном программном документе, в разделе «Назначение программы», указано функциональное и эксплуатационное назначение программы.

В разделе «Условия выполнения программы» указана информация о максимальных (рекомендуемых) и минимальных технических средствах для использования программы.

В разделе «Выполнение программы» подробно описан жизненный цикл программы. Описаны действия для загрузки, для запуска и для удаления программы. Также описаны команды и ответы программы на действия пользователя.

В разделе «Сообщения оператору» представлены все сообщения, выводимые оператору во время эксплуатации программы.

# НАЗНАЧЕНИЕ ПРОГРАММЫ

Функциональным назначением приложения «Drivee» является предоставление пользователям возможностей для поиска, продажи и покупки автомобилей, ведения записей о своем транспорте, а также управления личным профилем.

Программа должна эксплуатироваться открыто.

Конечными пользователями программы должны являются подразделения, занимающиеся куплей-продажей автомобилей, таких как автосалоны, дилерские центры, а также частные продавцы и покупатели.

# УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

## В Таблице 1 представлены максимальные (или рекомендуемые) и минимальные технические средства для использования программы.

Таблица 1 - Технические средства

|  |  |  |
| --- | --- | --- |
| № | Тип оборудования | Наименование оборудования |
| Максимальные или рекомендуемые | | |
| 1 | Размер экрана | 6.5" |
| 2 | Разрешение экрана | 2400×1080 |
| 3 | Процессор | Snapdragon 870 / Dimensity 1200 |
| 4 | Количество ядер процессора | 8 |
| 5 | Оперативная память | 6–8 ГБ |
| 6 | Тип накопителя | Встроенный (UFS 2.1/2.2) |
| 7 | Объем накопителя | от 128 ГБ |
| 8 | Операционная система | Android 11, 12, 13 |
| 9 | Связь | Стабильное подключение к интернету (Wi-Fi /4G / 5G) |
| Минимальные | | |
| 1 | Размер экрана | 5.0" |
| 2 | Разрешение экрана | 1280×720 |
| 3 | Процессор | Snapdragon 450 / MediaTek Helio P22 |
| 4 | Количество ядер процессора | 4 |
| 5 | Оперативная память | 2–3 ГБ |
| 6 | Тип накопителя | Встроенный (eMMC) |
| 7 | Объем накопителя | от 32 ГБ |
| 8 | Операционная система | Android 8.0 и выше |
| 9 | Связь | Стабильное подключение к интернету (3G / 4G) |

# ВЫПОЛНЕНИЕ ПРИЛОЖЕНИЯ

## Действия для загрузки приложения

Настольное приложение «Drivee» устанавливается в виде архива, указанынм на Рисунке 1. Архив можно получить с помощью физического носителя или скачать из Google Drive. Ссылка: https://drive.google.com/drive/folders/1ZnGyWfIyQVOotFzRWHYU15Nw3gRK 5wcu?usp=sharing

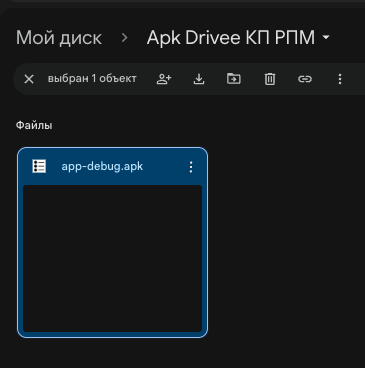


Рисунок 1 - Инсталлятор

## Действия для запуска приложения

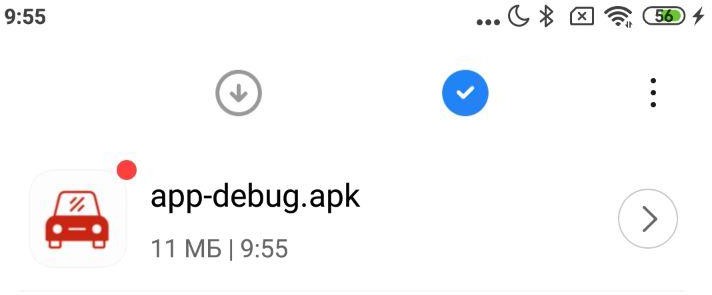
После установки инсталлятора необходимо запустить скачанный файл app-debug.apk.

Рисунок 2 - Файл запуска приложения

Далее, стандартный установщик android сообщит о том, что

приложение не требует специальных разрешений. Необходимо нажать

«Установить».

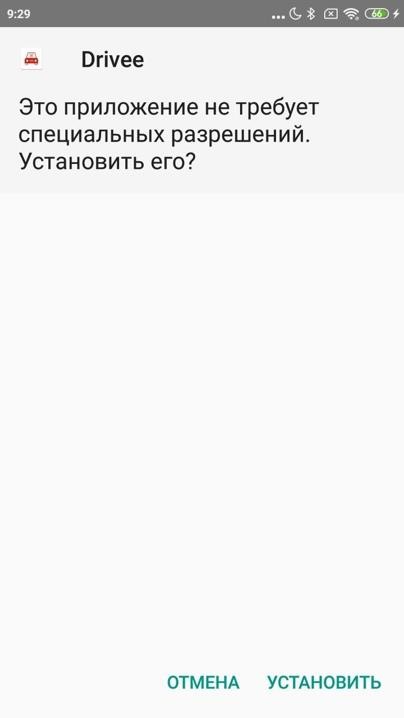


Рисунок 3 – Установка часть 1

## Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение Контент, сгенерированный ИИ, может содержать ошибки.Затем, установщик сообщит о том, что загрузка началась. Необходимо подождать около одной минуты.

Рисунок 4 – Установка часть 2

## После выполненных действий на рабочем столе android устройства появится приложение «Drivee».



Рисунок 5 – Рабочий стол

## Выполнение приложения с описанием функций

При запуске приложения первой открывается страница авторизации. На ней пользователь видит два текстовых поля для ввода логина (email) и пароля, а также три кнопки: «Войти», «Зарегистрироваться» и «Сбросить пароль». Пользователь может ввести свои учетные данные и нажать «Войти». При правильном вводе данных (например, [aa@gmail.com](mailto:aa@gmail.com) и xyzzytui) открывается главная страница пользователя. При вводе данных администратора [(qq@gmail.com](mailto:(qq@gmail.com) и xyzzytui) открывается административная панель. Если пользователь еще не зарегистрирован, он может перейти на страницу регистрации.

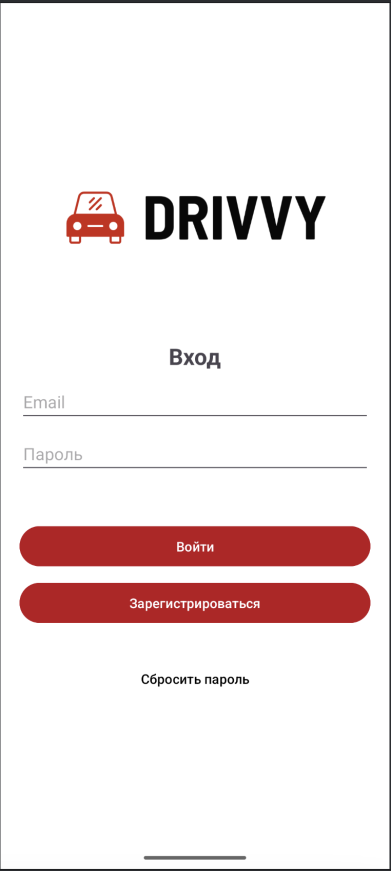


Рисунок 6 – Страница авторизации

## На странице регистрации пользователю предлагается заполнить несколько полей: имя, дата рождения, номер телефона, адрес электронной почты, пароль и подтверждение пароля. После заполнения всех полей и нажатия на кнопку регистрации данные отправляются в базу данных. При успешной регистрации пользователь автоматически возвращается на страницу авторизации, где может авторизоваться, используя введенные ранее данные.

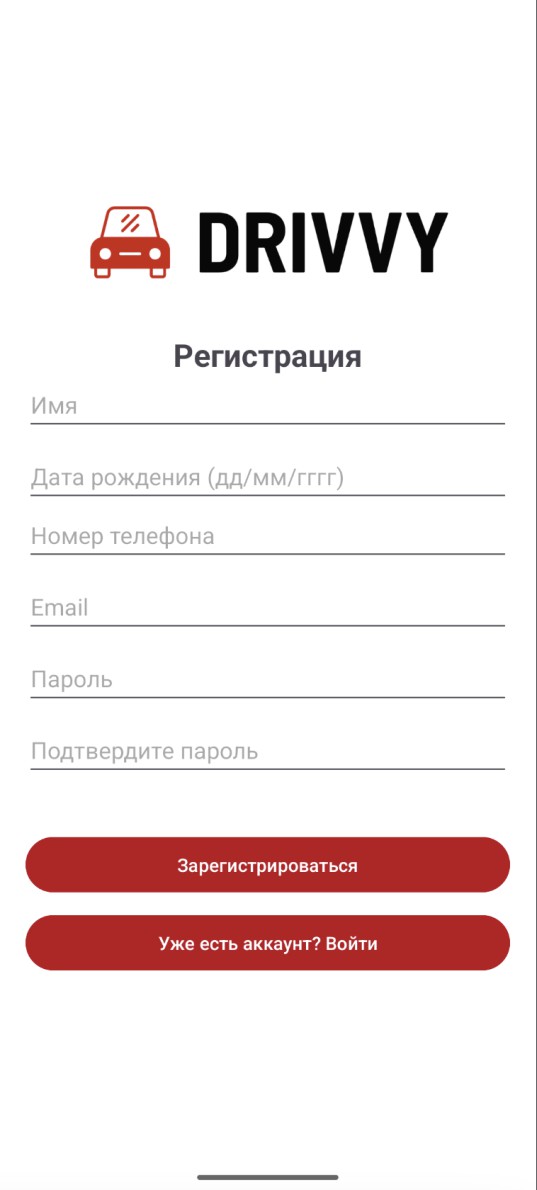


Рисунок 7 – Страница регистрации

## На этой странице находится одно текстовое поле для ввода электронной почты и две кнопки: «Сбросить пароль» и «Помните пароль? Войти». При вводе email и нажатии кнопки «Сбросить пароль» на указанную почту отправляется ссылка для сброса пароля. Перейдя по ссылке, пользователь попадает на форму смены пароля. Если пользователь передумал сбрасывать пароль, он может нажать на «Помните пароль? Войти», чтобы вернуться на страницу авторизации.

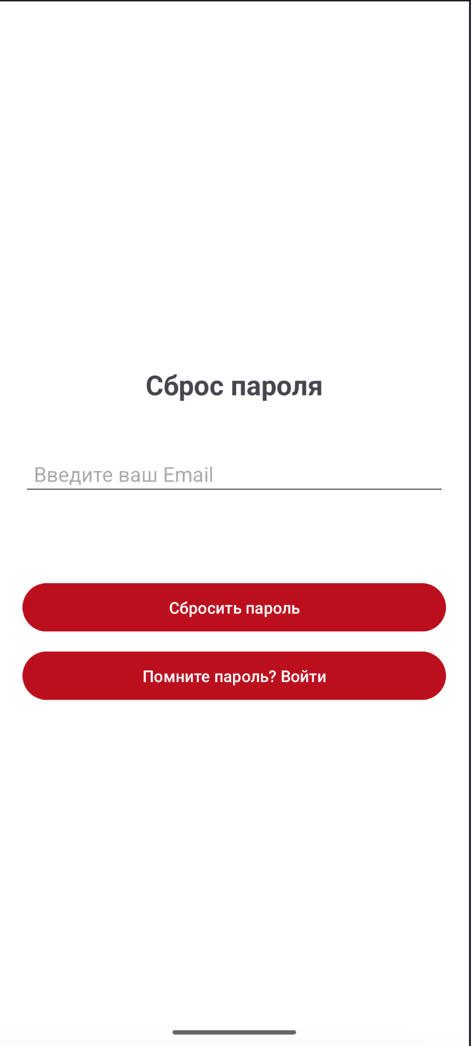


Рисунок 8 – Страница сброса пароля

На главной странице отображается количество доступных объявлений, а слева расположен выпадающий список для сортировки объявлений: по цене (по возрастанию/убыванию) и по пробегу. Ниже находятся выпадающие списки для выбора марки и модели авто. Еще ниже расположены две кнопки:

«Сбросить» (обнуляет выбранные фильтры) и «Параметры» (переход на страницу расширенного поиска). Под ними выводятся карточки объявлений: каждая карточка содержит фото, цену, марку, модель, пробег и кнопку

«Избранное». Нажатие на карточку открывает подробную информацию о выбранной машине.

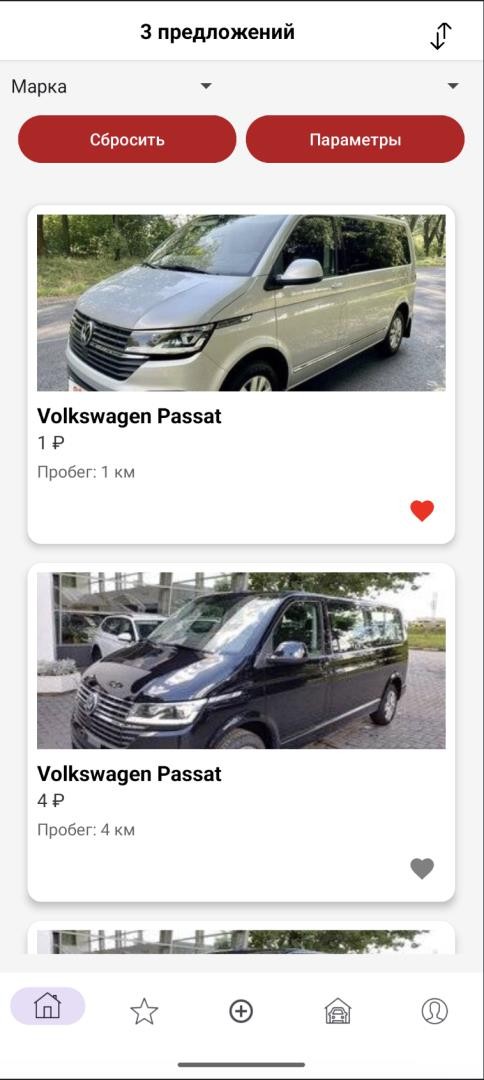


Рисунок 9 – Страница «Главная» пользователя

## На этой странице пользователь может задать дополнительные фильтры поиска. Доступны поля для выбора года выпуска (от/до), объема двигателя (от/до), мощности (от/до), количества владельцев (до), цены (от/до), города поиска, а также выпадающие списки для выбора цвета, типа кузова, типа трансмиссии и типа топлива. Внизу страницы две кнопки: «Применить» для запуска поиска по заданным параметрам и «Отменить» для возврата на главную страницу без применения фильтров.

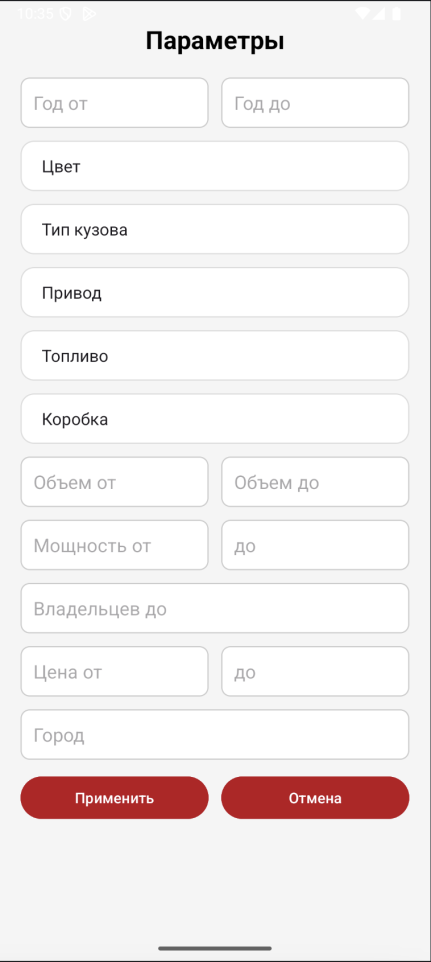


Рисунок 10 - Страница расширенного поиска (параметры).

## При нажатии на карточку машины с главной страницы открывается страница с полной информацией о машине: фотографии, цена, марка, модель, объем двигателя, мощность, год выпуска, цвет, тип кузова, тип топлива, тип трансмиссии, привод, количество владельцев, VIN-номер, пробег, описание, дата публикации. Также отображаются контактные данные продавца: имя, телефон и аватар.

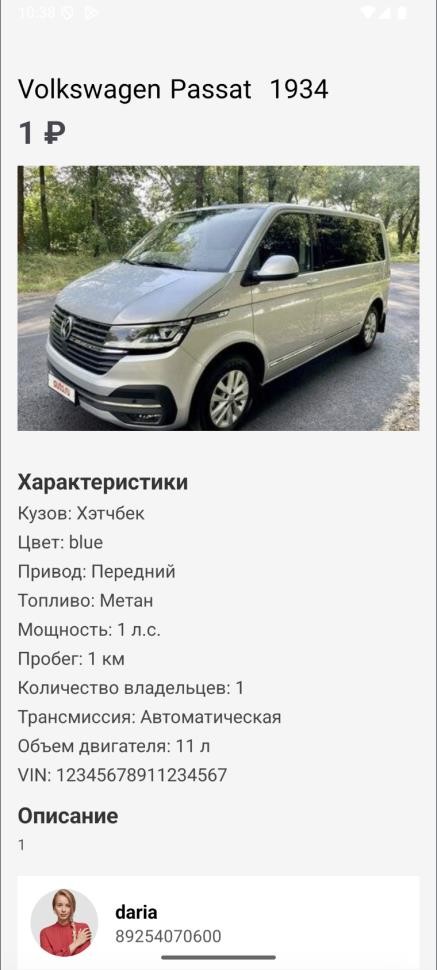


Рисунок 11 - Карточка машины (подробная информация).

## На всех основных страницах внизу экрана доступно меню с пятью кнопками: «Главная», «Избранное», «Создать объявление», «Бортжурнал»,

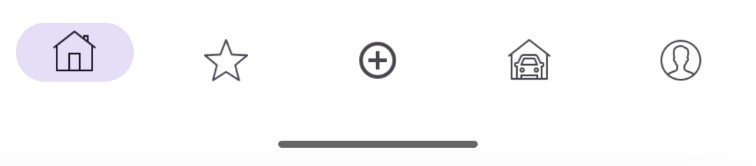
«Профиль». Каждая кнопка приводит к соответствующему разделу приложения.

Рисунок 12 - Bottom Navigation (нижнее меню навигации).

## На этой странице можно создать новое объявление о продаже автомобиля. Сначала выбирается марка авто, затем модель. После выбора открывается форма добавления объявления, где заполняются все параметры автомобиля, загружаются до пяти фотографий, указываются VIN, описание,

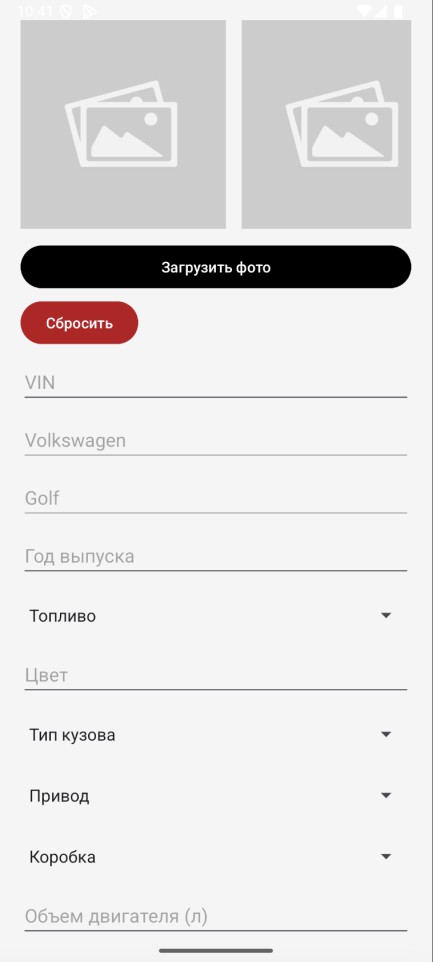
цена и другая информация. Есть кнопки «Загрузить фото», «Сбросить фото» и «Создать объявление».

Рисунок 13 - Страница создания объявления.

## Профиль пользователя отображает фотографию профиля, имя, телефон и email. Есть две кнопки: «Выбрать фото» (для смены аватара) и «Выйти из аккаунта» (переход обратно на страницу авторизации). Также выводятся все объявления пользователя с возможностью их удаления.

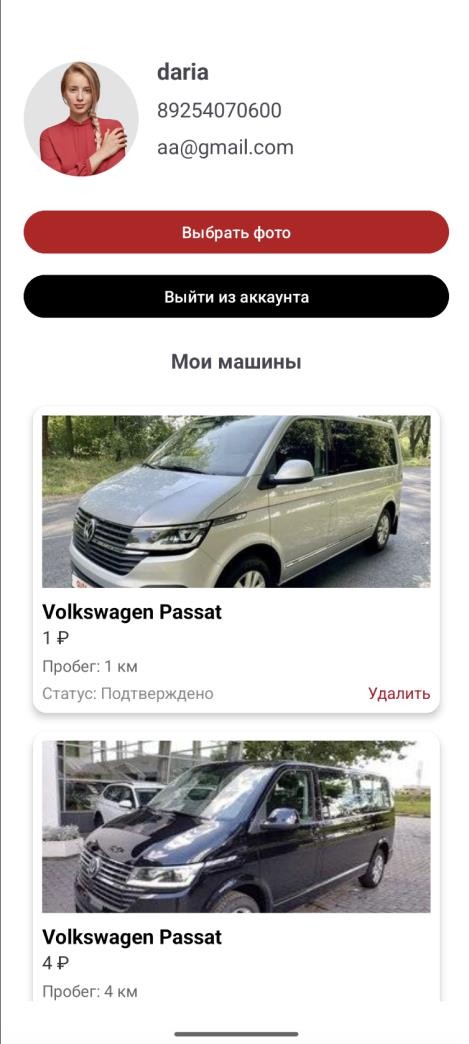


Рисунок 14 – Страница профиля.

## В разделе избранного отображаются машины, которые пользователь отметил сердечком на главной странице. Отсюда можно перейти к подробному просмотру машины или удалить её из избранного.

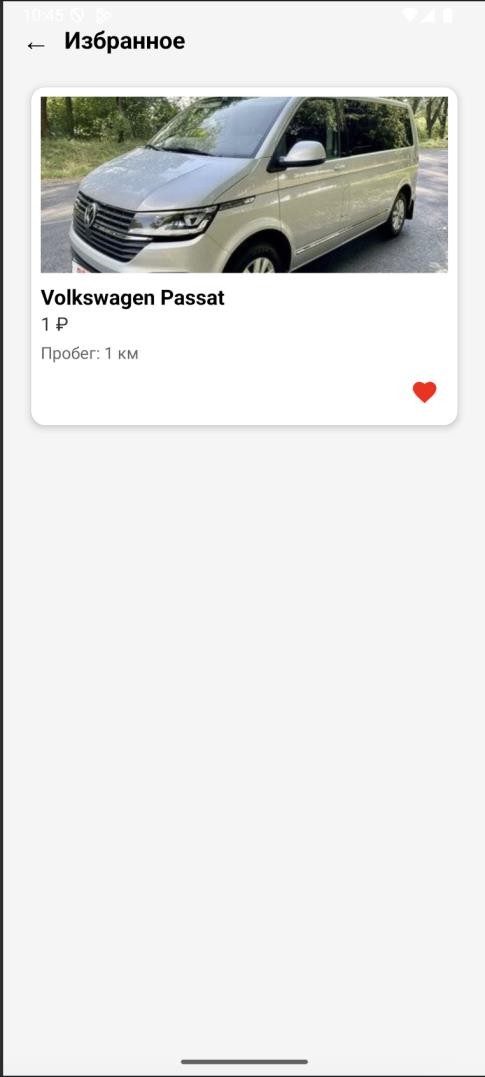


Рисунок 15 - Страница избранного.

## На странице бортжурнала пользователь видит созданные записи о своих поездках или событиях, связанных с автомобилями. Есть кнопка

«Создать запись», при нажатии на которую открывается форма добавления нового поста: марка, модель, заголовок, текст записи и пробег.

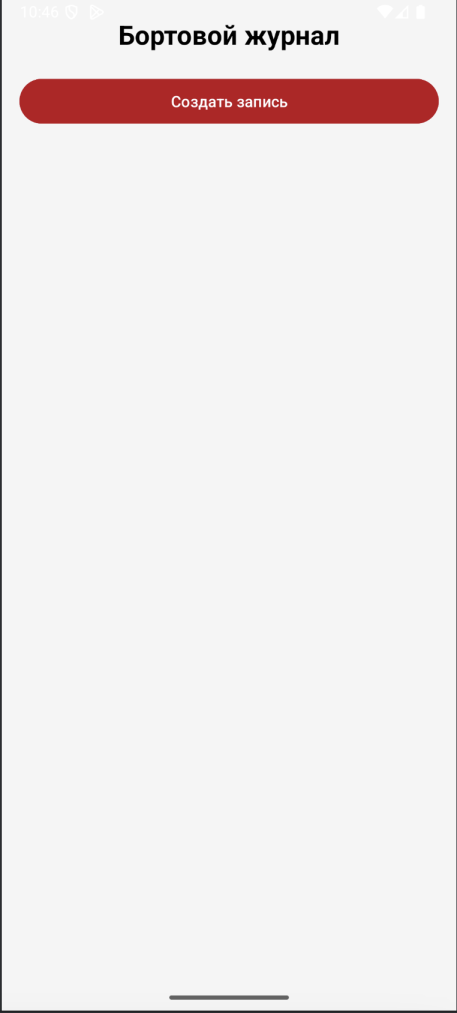


Рисунок 16 - Страница бортжурнала.

## На форме создания записи заполняются поля: марка авто, модель авто, заголовок записи, текст записи и пробег. После нажатия кнопки

«Опубликовать» запись появляется в бортжурнале.

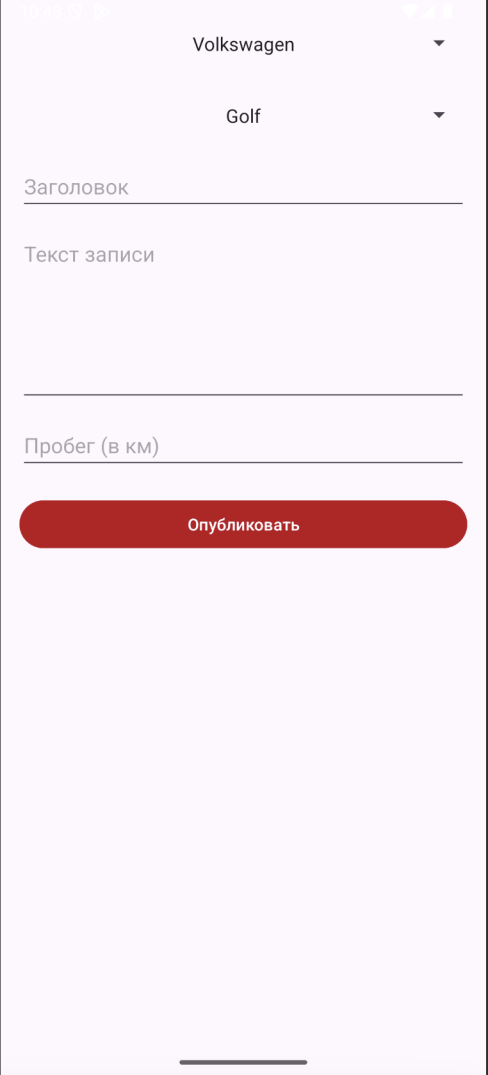


Рисунок 17 - Страница создания записи в бортжурнале.

## После авторизации администратор попадает на страницу управления объявлениями. Здесь выводятся все объявления со статусом «На проверке». Для каждого объявления доступны две кнопки: «Принять» (одобрить публикацию) и «Удалить» (удалить объявление). При нажатии на карточку объявления открывается его полное описание.

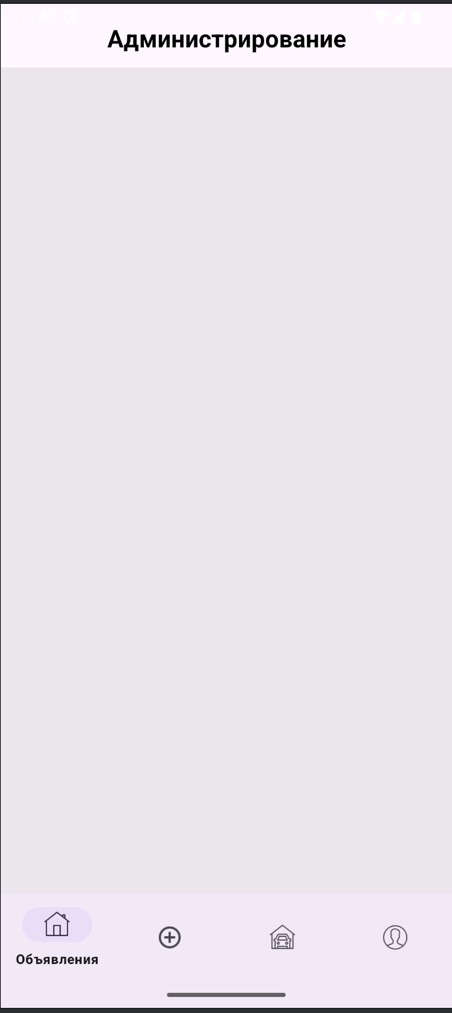


Рисунок 18 - Страница администратора (модерация объявлений).

## На этой странице администратор может добавлять новые марки и модели автомобилей. Для добавления модели требуется сначала выбрать марку из выпадающего списка.

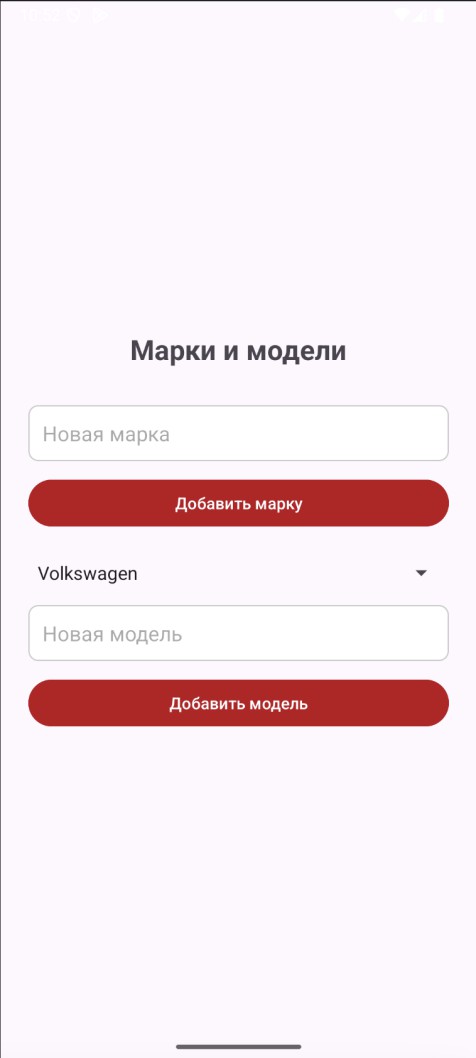


Рисунок 19 - Страница управления марками и моделями.

## Аналогично модерации объявлений, администратор может просматривать и модерировать записи в бортжурнале пользователей, используя кнопки «Принять» или «Удалить» для каждой записи.

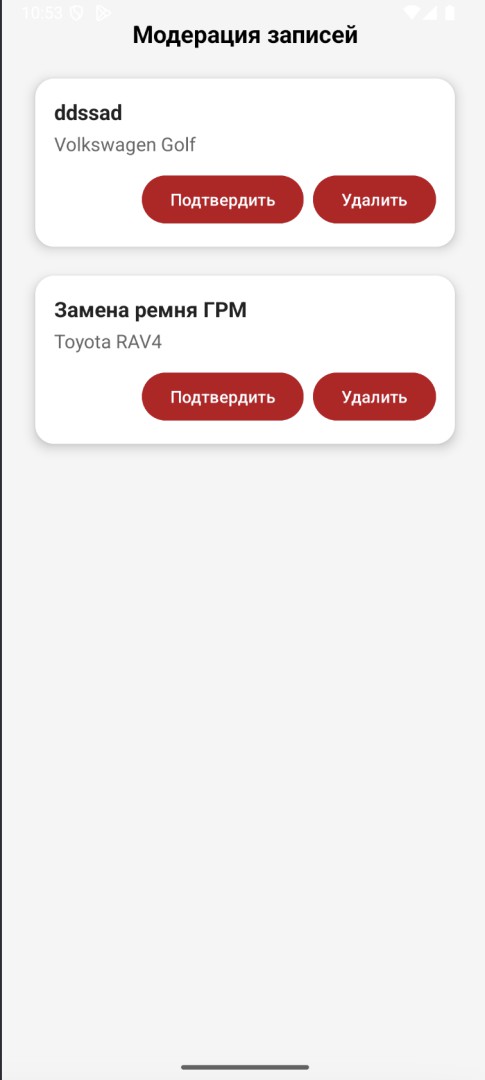


Рисунок 20 - Страница модерации записей бортжурнала.

# СООБЩЕНИЯ ОПЕРАТОРУ

## В Таблице 4 представлены все сообщения, выводимые оператору во время эксплуатации программы.

Таблица 2 - Сообщения оператору

|  |  |  |  |
| --- | --- | --- | --- |
| № | Текст сообщения | Содержание сообщения | Действия оператора |
| 1 | 2 | 3 | 4 |
| 1 | Неверный логин или пароль | Сообщение о некорректных данных для входа | Нажать на кнопку «ОК»,  проверить правильность ввода email и пароля |
| 2 | Пользователь с таким email уже  существует | Сообщение о невозможности зарегистрироваться с  существующим email | Нажать на кнопку «ОК», ввести другой email |
| 3 | Произошла  ошибка при сбросе пароля | Сообщение об ошибке отправки ссылки на почту | Нажать на кнопку «ОК», проверить корректность email |
| 4 | Пароли не совпадают | Сообщение о несоответствии пароля и подтверждения  пароля при регистрации | Нажать на кнопку «ОК», ввести совпадающие пароли |
| 5 | Все поля  обязательны для заполнения | Сообщение о незаполненных  обязательных полях регистрации или объявления | Нажать на кнопку «ОК», заполнить все поля формы |
| 6 | Ошибка  загрузки изображения | Сообщение об ошибке при  выборе фотографии для профиля или объявления | Нажать на кнопку «ОК»,  попробовать выбрать другое изображение |
| 7 | Ошибка загрузки данных | Сообщение о неполадках при загрузке объявлений или  записей бортжурнала | Нажать на кнопку «ОК», перезагрузить приложение или  повторить попытку позже |
| 8 | Объявление  успешно создано | Сообщение об успешном добавлении нового объявления | Нажать на кнопку «ОК», вернуться на главную страницу |
| 9 | Запись успешно создана | Сообщение об успешной публикации записи в  бортжурнал | Нажать на кнопку «ОК», вернуться на страницу  бортжурнала |
| 10 | Объявление удалено | Сообщение об успешном удалении объявления | Нажать на кнопку «ОК»,  список объявлений обновится автоматически |

ПРИЛОЖЕНИЕ Б. ТЕКСТ ПРОГРАММЫ АННОТАЦИЯ

В данном программном документе приведен текст программы для поддержания работы мобильного приложения «Drivee».

В данном программном документе, в разделе «Тест программы», указана информация о информационной системе, области применения программы, модулей программы и код каждого программного модуля.

1. МОДУЛИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

В Таблице 1 представлено описание всех классов и модулей программы, количество строк верстки или кода и размер файла.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Описание | Количество строк | Размер  файла в Кб |
| 1 | 2 | 3 | 4 | 5 |
| 1 | AndroidManifest.xml | конфигурация приложения: разрешения, иконки,  список активити | 54 | 2 Кб |
| 2 | AdminActivity.java | панель администратора, просмотр необработанных  объявлений | 93 | 5 Кб |
| 3 | AdminBortjournalActivity.jav a | панель администратора для управления  бортжурналами | 86 | 4 Кб |
| 4 | AdminBortjournalAdapter.java | адаптер для отображения  бортжурналов в админке | 88 | 5 Кб |
| 5 | AdminCarAdapter.java | адаптер для списка  машин в админке. | 110 | 6 Кб |
| 6 | AllActivity.java | главный экран со всеми  объявлениями + фильтры и навигация. | 391 | 9 Кб |
| 7 | BortjournalActivity.java | экран бортжурнала  пользователя | 62 | 2 Кб |
| 8 | BortjournalAdapter.java | адаптер для списка  записей в бортжурнале | 123 | 3 Кб |
| 9 | BortjournalDetailActivity.java | экран подробностей по  записи в бортжурнале | 30 | 2 Кб |
| 10 | BortjournalPost.java | модель (POJO) для  записи в бортжурнал | 80 | 4 Кб |
| 11 | Car.java | модель машины | 47 | 2 Кб |
| 12 | CarAdapter.java | адаптер для отображения  карточек машин | 198 | 3 Кб |
| 13 | CarAdapterUser.java | адаптер для отображения машин текущего  пользователя | 86 | 3 Кб |
| 14 | CarForProfile.java | отдельная модель  машины для профиля | 33 | 1 Кб |
| 15 | CenterActivity.java | экран карточки машины | 182 | 4 Кб |
| 16 | CreateListingActivity.java | создание нового  объявления | 305 | 8 Кб |
| 17 | CreatePostActivity.java | создание записи в  бортжурнал | 134 | 3 Кб |
| 18 | FavoritesActivity.java | избранные объявления  пользователя | 106 | 2 Кб |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Описание | Количество строк | Размер  файла в Кб |
| 1 | 2 | 3 | 4 | 5 |
| 19 | FilterActivity.java | расширенные фильтры поиска | 106 | 2 Кб |
| 20 | ImagePagerAdapter.java | адаптер для ViewPager с  фото машины | 68 | 2 Кб |
| 21 | MainActivity.java | экран авторизации | 94 | 2 Кб |
| 22 | ManageModelsActivity.java | админка: управление  марками и моделями | 97 | 2 Кб |
| 23 | RegActivity.java | экран регистрации | 114 | 3 Кб |
| 24 | ResetPasswordActivity.java | сброс пароля | 47 | 1 Кб |
| 25 | SelectBrandActivity.java | выбор марки авто. | 80 | 2 Кб |
| 26 | SelectModelActivity.java | выбор модели авто. | 95 | 2 Кб |
| 27 | SplashActivity.java | splash-экран при запуске | 22 | 1 Кб |
| 28 | User.java | модель пользователя | 30 | 1 Кб |
| 29 | UserCar.java | модель машины,  привязанной к пользователю | 48 | 1 Кб |
| 30 | UserProfileActivity.java | профиль пользователя | 197 | 3 Кб |
| 31 | circle.xml | закруглённый стиль | 5 | 1 Кб |
| 32 | edittext\_background.xml | фон для полей ввода | 3 | 1 Кб |
| 33 | ic\_heart\_filled.xml | иконка «лайка»  (избранное) | 9 | 1 Кб |
| 34 | ic\_heart\_outline.xml | иконка «лайка» —  пустая | 9 | 1 Кб |
| 35 | ic\_launcher\_background.xml | стандартные ресурсы  иконки | 75 | 2 Кб |
| 36 | ic\_launcher\_foreground.xml | стандартные ресурсы  иконки | 30 | 1 Кб |
| 37 | spinner\_background.xml | фон для выпадающего  списка (Spinner). | 6 | 1 Кб |
| 38 | splash\_background.xml | фон для splash-экрана. | 4 | 1 Кб |
| 39 | activity\_admin.xml | UI экрана AdminActivity | 45 | 1 Кб |
| 40 | activity\_admin\_bortjournal.xm  l | UI для  AdminBortjournalActivity | 26 | 1 Кб |
| 41 | activity\_all.xml | UI для экрана со всеми  объявлениями | 107 | 3 Кб |
| 42 | activity\_center.xml | UI карточки машины | 160 | 3 Кб |
| 43 | activity\_create\_listing.xml | UI создания нового  объявления | 292 | 4 Кб |
| 44 | activity\_favorites.xml | UI избранного | 49 | 1 Кб |
| 45 | activity\_filter.xml | UI экрана фильтрации | 216 | 4 Кб |
| 46 | activity\_main.xml | UI авторизации | 69 | 1 Кб |
| 47 | activity\_manage\_models.xml | UI управления  марками/моделями. | 68 | 1 Кб |
| 48 | activity\_reg.xml | UI регистрации | 91 | 2 Кб |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Описание | Количество строк | Размер  файла в Кб |
| 1 | 2 | 3 | 4 | 5 |
| 49 | activity\_reset\_password.xml | UI сброса пароля | 41 | 1 Кб |
| 50 | activity\_select\_brand.xml | UI выбора марки | 33 | 1 Кб |
| 51 | activity\_select\_model.xml | UI выбора модели | 33 | 1 Кб |
| 52 | activity\_splash.xml | UI заставки | 14 | 1 Кб |
| 53 | activity\_user\_profile.xml | UI профиля пользователя | 113 | 2 Кб |
| 54 | bortjournal\_activity.xml | UI экрана с записями  бортжурнала | 28 | 1 Кб |
| 55 | bortjournal\_detail\_activity.xml | UI экрана с деталями записи | 38 | 1 Кб |
| 56 | create\_post\_activity.xml | UI создания записи в  бортжурнал | 60 | 2 Кб |
| 57 | item\_admin\_bortjournal.xml | элемент списка записей в  админке | 61 | 2 Кб |
| 58 | item\_bortjournal.xml | элемент записи в  бортжурнале | 70 | 2 Кб |
| 59 | item\_car.xml | карточка машины. | 86 | 2 Кб |
| 60 | item\_car\_admin.xml | карточка машины для  админки | 63 | 2 Кб |
| 61 | item\_car\_image.xml | элемент изображения машины | 5 | 1 Кб |
| 62 | item\_car\_user.xml | карточка машины в  профиле пользователя | 86 | 2 Кб |
| 63 | item\_image.xml | отдельный элемент  изображения | 6 | 1 Кб |
| 64 | bottom\_nav\_menu.xml | нижняя навигация | 11 | 1 Кб |
| 65 | bottom\_navigation\_menu.xml | нижняя навигация | 19 | 1 Кб |
| 66 | colors.xml | определение цветов в  приложении | 6 | 1 Кб |
| 67 | strings.xml | строковые ресурсы | 3 | 1 Кб |
| 68 | styles.xml | стили | 8 | 1 Кб |
| 69 | values.xml | общее хранилище  ресурсов | 9 | 1 Кб |
| 70 | build.gradle | конфигурация сборки  приложения | 50 | 1 Кб |
| 71 | libs.versions.toml | зависимости, версии  библиотек | 29 | 1 Кб |

1. ИСХОДНЫЙ КОД ПРОГРАММЫ
2. AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="<http://schemas.android.com/apk/res/android>" xmlns:tools="<http://schemas.android.com/tools>">

<uses-feature android:name="android.hardware.telephony"

android:required="false" />

<uses-permission android:name="android.permission.CALL\_PHONE" />

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

<application android:allowBackup="true"

android:dataExtractionRules="@xml/data\_extraction\_rules" android:fullBackupContent="@xml/backup\_rules" android:icon="@mipmap/ic\_launcher" android:roundIcon="@mipmap/ic\_launcher\_round" android:label="@string/app\_name" android:supportsRtl="true" android:theme="@style/Theme.Drivee" tools:targetApi="31">

<activity android:name=".MainActivity" android:exported="true">

<intent-filter>

<action android:name="android.intent.action.MAIN"/>

<category android:name="android.intent.category.LAUNCHER"/>

</intent-filter>

</activity>

<activity android:name=".RegActivity" />

<activity android:name=".SplashActivity" />

<activity android:name=".FavoritesActivity" />

<activity android:name=".SelectModelActivity" />

<activity android:name=".ManageModelsActivity" />

<activity android:name=".AdminBortjournalActivity" />

<activity android:name=".FilterActivity" />

<activity android:name=".CreateListingActivity" />

<activity android:name=".ResetPasswordActivity" />

<activity android:name=".CenterActivity" />

<activity android:name=".SelectBrandActivity" />

<activity android:name=".AllActivity" />

<activity android:name=".BortjournalDetailActivity" />

<activity android:name=".BortjournalActivity" />

<activity android:name=".CreatePostActivity" />

<activity android:name=".UserProfileActivity" />

<activity android:name=".AdminActivity" />

</application>

</manifest>

## AdminActivity.java

package com.example.drivee;

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

import androidx.appcompat.app.AppCompatActivity; import androidx.recyclerview.widget.LinearLayoutManager; import androidx.recyclerview.widget.RecyclerView;

import com.google.android.material.bottomnavigation.BottomNavigationView; import com.google.firebase.firestore.FirebaseFirestore;

import com.google.firebase.firestore.QueryDocumentSnapshot;

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

public class AdminActivity extends AppCompatActivity { private RecyclerView recyclerView;

private AdminCarAdapter adminCarAdapter;

private List<Car> carList;

private BottomNavigationView bottomNavigationView; private FirebaseFirestore db;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_admin);

recyclerView = findViewById(R.id.recyclerView); recyclerView.setLayoutManager(new LinearLayoutManager(this));

carList = new ArrayList<>();

adminCarAdapter = new AdminCarAdapter(this, carList); recyclerView.setAdapter(adminCarAdapter);

db = FirebaseFirestore.getInstance();

bottomNavigationView = findViewById(R.id.bottomNavigationView); bottomNavigationView.setOnItemSelectedListener(item -> {

if (item.getItemId() == R.id.nav\_manage\_models) {

Intent intent = new Intent(AdminActivity.this, ManageModelsActivity.class); startActivity(intent);

return true;

}

else if (item.getItemId() == R.id.nav\_manage\_bort) {

Intent intent = new Intent(AdminActivity.this, AdminBortjournalActivity.class); startActivity(intent);

return true;

}

return false;

});

loadPendingCars();

}

private void loadPendingCars() { db.collection("cars")

.whereEqualTo("status", "pending")

.get()

.addOnCompleteListener(task -> { if (task.isSuccessful()) {

carList.clear();

for (QueryDocumentSnapshot document : task.getResult()) { String photoone = document.getString("photoone");

if (photoone == null) photoone = "";

long mileage = document.getLong("mileage") != null ? document.getLong("mileage") : 0;

Car car = new Car( document.getId(), document.getString("brand"), document.getString("model"), photoone,

mileage, document.getLong("price")

);

carList.add(car);

}

adminCarAdapter.notifyDataSetChanged();

} else {

Log.e("AdminActivity", "Ошибка загрузки машин", task.getException());

}

});

}

}

## AdminBortjournalActivity.java

package com.example.drivee;

import android.content.Intent; import android.os.Bundle; import android.view.View; import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity; import androidx.recyclerview.widget.LinearLayoutManager; import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.firestore.FirebaseFirestore;

import com.google.firebase.firestore.QueryDocumentSnapshot;

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

public class AdminBortjournalActivity extends AppCompatActivity { private RecyclerView recyclerView;

private FirebaseFirestore db;

private List<BortjournalPost> postList = new ArrayList<>(); private AdminBortjournalAdapter adapter;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_admin\_bortjournal);

recyclerView = findViewById(R.id.adminRecyclerView); recyclerView.setLayoutManager(new LinearLayoutManager(this));

db = FirebaseFirestore.getInstance();

adapter = new AdminBortjournalAdapter(postList, this::openDetail, this::approvePost, this::deletePost); recyclerView.setAdapter(adapter);

loadPendingPosts();

}

private void loadPendingPosts() { db.collection("bortjornal")

.whereEqualTo("status", "pending")

.get()

.addOnSuccessListener(querySnapshot -> { postList.clear();

for (QueryDocumentSnapshot doc : querySnapshot) { BortjournalPost post = doc.toObject(BortjournalPost.class); post.setId(doc.getId());

postList.add(post);

}

adapter.notifyDataSetChanged();

})

.addOnFailureListener(e -> Toast.makeText(this, "Ошибка загрузки", Toast.LENGTH\_SHORT).show());

}

private void openDetail(BortjournalPost post) {

Intent intent = new Intent(this, BortjournalDetailActivity.class); intent.putExtra("brand", post.getBrand()); intent.putExtra("model", post.getModel()); intent.putExtra("title", post.getTitle()); intent.putExtra("content", post.getContent()); startActivity(intent);

}

private void approvePost(BortjournalPost post) { db.collection("bortjornal").document(post.getId())

.update("status", "approved")

.addOnSuccessListener(unused -> {

Toast.makeText(this, "Подтверждено", Toast.LENGTH\_SHORT).show();

postList.remove(post); adapter.notifyDataSetChanged();

});

}

private void deletePost(BortjournalPost post) { db.collection("bortjornal").document(post.getId())

.delete()

.addOnSuccessListener(unused -> {

Toast.makeText(this, "Удалено", Toast.LENGTH\_SHORT).show(); postList.remove(post);

adapter.notifyDataSetChanged();

});

}

}

## AdminBortjournalAdapter.java

package com.example.drivee;

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

import android.view.ViewGroup; import android.widget.Button; import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView; import java.util.List;

public class AdminBortjournalAdapter extends RecyclerView.Adapter<AdminBortjournalAdapter.ViewHolder> {

public interface OnPostOpenListener { void onOpen(BortjournalPost post);

}

public interface OnPostApproveListener { void onApprove(BortjournalPost post);

}

public interface OnPostDeleteListener { void onDelete(BortjournalPost post);

}

private List<BortjournalPost> posts; private OnPostOpenListener openListener;

private OnPostApproveListener approveListener; private OnPostDeleteListener deleteListener;

public AdminBortjournalAdapter(List<BortjournalPost> posts,

OnPostOpenListener openListener, OnPostApproveListener approveListener, OnPostDeleteListener deleteListener) {

this.posts = posts; this.openListener = openListener;

this.approveListener = approveListener; this.deleteListener = deleteListener;

}

@NonNull @Override

public AdminBortjournalAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) { View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item\_admin\_bortjournal, parent, false); return new ViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull AdminBortjournalAdapter.ViewHolder holder, int position) { BortjournalPost post = posts.get(position);

holder.brandModelTextView.setText(post.getBrand() + " " + post.getModel()); holder.titleTextView.setText(post.getTitle());

holder.itemView.setOnClickListener(v -> {

if (openListener != null) openListener.onOpen(post);

});

holder.approveButton.setOnClickListener(v -> {

if (approveListener != null) approveListener.onApprove(post);

});

holder.deleteButton.setOnClickListener(v -> {

if (deleteListener != null) deleteListener.onDelete(post);

});

}

@Override

public int getItemCount() { return posts.size();

}

public static class ViewHolder extends RecyclerView.ViewHolder { TextView titleTextView, brandModelTextView;

Button approveButton, deleteButton;

public ViewHolder(@NonNull View itemView) { super(itemView);

brandModelTextView = itemView.findViewById(R.id.adminBrandModel); titleTextView = itemView.findViewById(R.id.adminTitle);

approveButton = itemView.findViewById(R.id.approveButton); deleteButton = itemView.findViewById(R.id.deleteButton);

}

}

}

## AdminCarAdapter.java

package com.example.drivee;

import android.content.Context; import android.content.Intent; import android.view.LayoutInflater; import android.view.View;

import android.view.ViewGroup; import android.widget.Button; import android.widget.TextView; import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView; import com.google.firebase.firestore.FirebaseFirestore; import java.util.List;

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

private Context context; private List<Car> carList;

public AdminCarAdapter(Context context, List<Car> carList) { this.context = context;

this.carList = carList;

}

@NonNull @Override

public CarViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) { View itemView = LayoutInflater.from(context).inflate(R.layout.item\_car\_admin, parent, false); return new CarViewHolder(itemView);

}

@Override

public void onBindViewHolder(@NonNull CarViewHolder holder, int position) { Car car = carList.get(position);

holder.tvBrand.setText(car.getBrand()); holder.tvModel.setText(car.getModel()); holder.tvPrice.setText(String.format("%d руб.", car.getPrice()));

holder.btnAccept.setOnClickListener(v -> { holder.btnAccept.setEnabled(false); // анти-спам updateCarStatus(car.getId(), "approved", position);

});

holder.btnDelete.setOnClickListener(v -> { holder.btnDelete.setEnabled(false); deleteCar(car.getId(), position);

});

holder.itemView.setOnClickListener(v -> {

Intent intent = new Intent(context, CenterActivity.class); intent.putExtra("carId", car.getId()); context.startActivity(intent);

});

}

@Override

public int getItemCount() { return carList.size();

}

public class CarViewHolder extends RecyclerView.ViewHolder { TextView tvBrand, tvModel, tvPrice;

Button btnAccept, btnDelete;

public CarViewHolder(@NonNull View itemView) { super(itemView);

tvBrand = itemView.findViewById(R.id.tvBrand); tvModel = itemView.findViewById(R.id.tvModel); tvPrice = itemView.findViewById(R.id.tvPrice); btnAccept = itemView.findViewById(R.id.btnAccept); btnDelete = itemView.findViewById(R.id.btnDelete);

}

}

private void updateCarStatus(String carId, String status, int position) { FirebaseFirestore db = FirebaseFirestore.getInstance(); db.collection("cars").document(carId)

.update("status", status)

.addOnSuccessListener(aVoid -> {

Toast.makeText(context, "Объявление утверждено!", Toast.LENGTH\_SHORT).show(); carList.remove(position);

notifyItemRemoved(position);

})

.addOnFailureListener(e -> {

Toast.makeText(context, "Ошибка: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

});

}

private void deleteCar(String carId, int position) { FirebaseFirestore db = FirebaseFirestore.getInstance(); db.collection("cars").document(carId)

.delete()

.addOnSuccessListener(aVoid -> {

Toast.makeText(context, "Объявление удалено!", Toast.LENGTH\_SHORT).show(); carList.remove(position);

notifyItemRemoved(position);

})

.addOnFailureListener(e -> {

Toast.makeText(context, "Ошибка: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

});

}

}

## AllActivity.java

package com.example.drivee;

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

import android.view.MenuItem; import android.view.View;

import android.widget.ArrayAdapter; import android.widget.Button; import android.widget.ImageButton; import android.widget.Spinner; import android.widget.TextView; import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity; import androidx.recyclerview.widget.LinearLayoutManager; import androidx.recyclerview.widget.RecyclerView;

import com.google.android.material.bottomnavigation.BottomNavigationView; import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.firestore.FirebaseFirestore;

import com.google.firebase.firestore.QueryDocumentSnapshot;

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

public class AllActivity extends AppCompatActivity { private RecyclerView recyclerView;

private CarAdapter carAdapter;

private Button resetFiltersButton, paramsButton;

private List<Car> carList;

private Spinner brandSpinner, modelSpinner; private List<String> brandList = new ArrayList<>();

private List<String> modelList = new ArrayList<>(); private String selectedBrand = null;

private String selectedModel = null;

private ImageButton filterButton; private FirebaseFirestore db;

private Set<String> favoriteCarIds = new HashSet<>(); private FirebaseAuth auth;

private TextView adsCountTextView;

private final BottomNavigationView.OnNavigationItemSelectedListener navListener = new BottomNavigationView.OnNavigationItemSelectedListener() {

@Override

public boolean onNavigationItemSelected(@NonNull MenuItem item) { int id = item.getItemId();

if (id == R.id.nav\_home) {

startActivity(new Intent(AllActivity.this, AllActivity.class)); return true;

} else if (id == R.id.nav\_saved) {

startActivity(new Intent(AllActivity.this, FavoritesActivity.class)); return true;

} else if (id == R.id.nav\_create) {

startActivity(new Intent(AllActivity.this, SelectBrandActivity.class)); return true;

} else if (id == R.id.nav\_profile) {

startActivity(new Intent(AllActivity.this, UserProfileActivity.class)); return true;

} else if (id == R.id.nav\_garage) {

startActivity(new Intent(AllActivity.this, BortjournalActivity.class)); return true;

}

return false;

}

};

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_all);

db = FirebaseFirestore.getInstance(); auth = FirebaseAuth.getInstance();

carList = new ArrayList<>();

carAdapter = new CarAdapter(this, carList);

recyclerView = findViewById(R.id.recyclerView); recyclerView.setLayoutManager(new LinearLayoutManager(this)); recyclerView.setAdapter(carAdapter);

adsCountTextView = findViewById(R.id.adsCountTextView); filterButton = findViewById(R.id.filterButton);

brandSpinner = findViewById(R.id.brandSpinner); modelSpinner = findViewById(R.id.modelSpinner); resetFiltersButton = findViewById(R.id.resetFiltersButton); paramsButton = findViewById(R.id.paramsButton); paramsButton.setOnClickListener(v -> {

Intent intent = new Intent(this, FilterActivity.class); startActivityForResult(intent, 101);

});

filterButton.setOnClickListener(v -> showFilterMenu()); resetFiltersButton.setOnClickListener(v -> {

brandSpinner.setSelection(0); modelSpinner.setSelection(0); selectedBrand = null; selectedModel = null; loadCars();

});

loadFavoriteCars(); loadCars(); loadBrands();

BottomNavigationView bottomNavigationView = findViewById(R.id.bottomNavigation); bottomNavigationView.setOnNavigationItemSelectedListener(navListener);

}private void applyAdvancedFilters(Intent data) { db.collection("cars")

.whereEqualTo("status", "approved")

.get()

.addOnSuccessListener(queryDocumentSnapshots -> { carList.clear();

for (QueryDocumentSnapshot doc : queryDocumentSnapshots) { boolean isValid = true;

Long year = doc.getLong("year");

Double volume = doc.getDouble("volume"); Long horsepower = doc.getLong("horsepower"); Long owners = doc.getLong("owners");

Long price = doc.getLong("price");

String color = doc.getString("color"); String body = doc.getString("body"); String drive = doc.getString("drive");

String fuel = doc.getString("fuel");

String transmission = doc.getString("transmission"); String city = doc.getString("city");

isValid &= checkRange(year, data.getStringExtra("yearFrom"), data.getStringExtra("yearTo"));

isValid &= checkRange(volume, data.getStringExtra("volumeFrom"), data.getStringExtra("volumeTo")); isValid &= checkRange(horsepower, data.getStringExtra("powerFrom"), data.getStringExtra("powerTo")); isValid &= checkRange(owners, null, data.getStringExtra("ownersTo"));

isValid &= checkRange(price, data.getStringExtra("priceFrom"), data.getStringExtra("priceTo"));

isValid &= checkString(color, data.getStringExtra("color")); isValid &= checkString(body, data.getStringExtra("body")); isValid &= checkString(drive, data.getStringExtra("drive")); isValid &= checkString(fuel, data.getStringExtra("fuel"));

isValid &= checkString(transmission, data.getStringExtra("transmission")); isValid &= checkString(city, data.getStringExtra("city"));

if (isValid) {

String photoone = doc.getString("photoone"); if (photoone == null) photoone = "";

long mileage = doc.getLong("mileage") != null ? doc.getLong("mileage") : 0; Car car = new Car(

doc.getId(),

doc.getString("brand"), doc.getString("model"), photoone,

mileage, doc.getLong("price")

);

car.setFavorite(favoriteCarIds.contains(car.getId())); carList.add(car);

}

}

adsCountTextView.setText("Найдено объявлений: " + carList.size()); carAdapter.notifyDataSetChanged();

});

}

private boolean checkRange(Number value, String fromStr, String toStr) { if (value == null) return false;

try {

double doubleValue = value.doubleValue();

if (fromStr != null && doubleValue < Double.parseDouble(fromStr)) return false; if (toStr != null && doubleValue > Double.parseDouble(toStr)) return false;

} catch (NumberFormatException e) { return false;

}

return true;

}

private boolean checkString(String fieldValue, String expected) { if (expected == null) return true; // Не фильтруем

return expected.equalsIgnoreCase(fieldValue);

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) { super.onActivityResult(requestCode, resultCode, data);

if (requestCode == 101 && resultCode == RESULT\_OK && data != null) { applyAdvancedFilters(data);

}

}

private void loadFavoriteCars() {

String userId = auth.getCurrentUser().getUid();

db.collection("users").document(userId).get()

.addOnSuccessListener(documentSnapshot -> {

if (documentSnapshot.exists() && documentSnapshot.contains("favorites")) { List<String> favoriteCarIdsList = (List<String>) documentSnapshot.get("favorites"); if (favoriteCarIdsList != null) {

favoriteCarIds.addAll(favoriteCarIdsList);

}

}

})

.addOnFailureListener(e -> Log.e("AllActivity", "Ошибка загрузки избранных машин", e));

}

private void showFilterMenu() {

android.widget.PopupMenu popupMenu = new android.widget.PopupMenu(this, filterButton); popupMenu.getMenu().add("Цена ↑");

popupMenu.getMenu().add("Цена ↓"); popupMenu.getMenu().add("Новые");

popupMenu.setOnMenuItemClickListener(item -> { String selected = item.getTitle().toString();

switch (selected) { case "Цена ↑":

carList.sort((a, b) -> Long.compare(a.getPrice(), b.getPrice())); break;

case "Цена ↓":

carList.sort((a, b) -> Long.compare(b.getPrice(), a.getPrice())); break;

case "Новые":

carList.sort((a, b) -> b.getId().compareTo(a.getId())); break;

}

carAdapter.notifyDataSetChanged(); return true;

});

popupMenu.show();

}

private void loadBrands() { db.collection("dfd").document("models")

.get()

.addOnSuccessListener(documentSnapshot -> { if (documentSnapshot.exists()) {

Map<String, Object> data = documentSnapshot.getData(); if (data != null) {

brandList.clear(); brandList.add("Марка");

brandList.addAll(data.keySet());

ArrayAdapter<String> brandAdapter = new ArrayAdapter<>(this, android.R.layout.simple\_spinner\_item, brandList);

brandAdapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item); brandSpinner.setAdapter(brandAdapter);

brandSpinner.setOnItemSelectedListener(new android.widget.AdapterView.OnItemSelectedListener() { @Override

public void onItemSelected(android.widget.AdapterView<?> parent, View view, int position, long id) { if (position == 0) return;

selectedBrand = brandList.get(position); loadModels(selectedBrand, data);

}

}

}

});

}

@Override

public void onNothingSelected(android.widget.AdapterView<?> parent) {}

});

private void loadModels(String brand, Map<String, Object> data) { Object rawModels = data.get(brand);

if (rawModels instanceof List) {

List<String> models = (List<String>) rawModels; modelList.clear();

modelList.add("Модель"); modelList.addAll(models);

ArrayAdapter<String> modelAdapter = new ArrayAdapter<>(this, android.R.layout.simple\_spinner\_item, modelList);

modelAdapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item); modelSpinner.setAdapter(modelAdapter);

modelSpinner.setOnItemSelectedListener(new android.widget.AdapterView.OnItemSelectedListener() { @Override

public void onItemSelected(android.widget.AdapterView<?> parent, View view, int position, long id) { if (position == 0) return;

selectedModel = modelList.get(position); filterCars();

}

@Override

public void onNothingSelected(android.widget.AdapterView<?> parent) {}

});

}

}

private void filterCars() { db.collection("cars")

.whereEqualTo("status", "approved")

.whereEqualTo("brand", selectedBrand)

.whereEqualTo("model", selectedModel)

.get()

.addOnSuccessListener(queryDocumentSnapshots -> { carList.clear();

for (QueryDocumentSnapshot document : queryDocumentSnapshots) { String photoone = document.getString("photoone");

if (photoone == null) photoone = "";

long mileage = document.getLong("mileage") != null ? document.getLong("mileage") : 0; Car car = new Car(

document.getId(),

document.getString("brand"), document.getString("model"), photoone,

mileage, document.getLong("price")

);

car.setFavorite(favoriteCarIds.contains(car.getId())); carList.add(car);

}

adsCountTextView.setText("Найдено объявлений: " + carList.size()); carAdapter.notifyDataSetChanged();

});

}

private long getMileage(Car car) { try {

java.lang.reflect.Field mileageField = car.getClass().getDeclaredField("mileage"); mileageField.setAccessible(true);

Object value = mileageField.get(car);

return value instanceof Long ? (Long) value : 0L;

} catch (Exception e) { return 0L;

}

}

private void loadCars() { db.collection("cars")

.whereEqualTo("status", "approved")

.get()

.addOnCompleteListener(task -> { if (task.isSuccessful()) {

carList.clear();

for (QueryDocumentSnapshot document : task.getResult()) { String photoone = document.getString("photoone");

if (photoone == null) { photoone = "";

}

long mileage = document.getLong("mileage") != null ? document.getLong("mileage") : 0; Car car = new Car(

document.getId(), document.getString("brand"), document.getString("model"), photoone,

mileage, document.getLong("price")

);

car.setFavorite(favoriteCarIds.contains(car.getId())); carList.add(car);

}

adsCountTextView.setText(carList.size() + " предложений"); carAdapter.notifyDataSetChanged();

} else {

Log.e("AllActivity", "Ошибка загрузки машин", task.getException());

}

});

}

}

## BortjournalActivity.java;

package com.example.drivee;

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

import androidx.appcompat.app.AppCompatActivity; import androidx.recyclerview.widget.LinearLayoutManager; import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.firestore.FirebaseFirestore; import com.google.firebase.firestore.Query;

import com.google.firebase.firestore.QueryDocumentSnapshot;

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

public class BortjournalActivity extends AppCompatActivity { private RecyclerView postsRecyclerView;

private Button createPostButton;

private FirebaseFirestore db;

private List<BortjournalPost> posts = new ArrayList<>(); private BortjournalAdapter adapter;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.bortjournal\_activity);

postsRecyclerView = findViewById(R.id.postsRecyclerView); createPostButton = findViewById(R.id.createPostButton);

db = FirebaseFirestore.getInstance();

adapter = new BortjournalAdapter(this, posts);

postsRecyclerView.setLayoutManager(new LinearLayoutManager(this)); postsRecyclerView.setAdapter(adapter);

createPostButton.setOnClickListener(v -> { startActivity(new Intent(this, CreatePostActivity.class));

});

loadPosts();

}

private void loadPosts() { db.collection("bortjournal")

.whereEqualTo("status", "approved")

.orderBy("timestamp", Query.Direction.DESCENDING)

.get()

.addOnSuccessListener(queryDocumentSnapshots -> { posts.clear();

for (QueryDocumentSnapshot doc : queryDocumentSnapshots) { BortjournalPost post = doc.toObject(BortjournalPost.class); posts.add(post);

}

adapter.notifyDataSetChanged();

});

}

}

## BortjournalAdapter.java;

package com.example.drivee;

import android.content.Context; import android.content.Intent; import android.graphics.Bitmap;

import android.graphics.BitmapFactory; import android.graphics.Canvas;

import android.graphics.Paint;

import android.graphics.PorterDuffXfermode; import android.graphics.Rect;

import android.graphics.RectF; import android.graphics.PorterDuff; import android.util.Base64;

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

import android.view.ViewGroup; import android.widget.ImageView; import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView; import com.google.firebase.firestore.FirebaseFirestore; import java.util.List;

public class BortjournalAdapter extends RecyclerView.Adapter<BortjournalAdapter.PostViewHolder> {

private Context context;

private List<BortjournalPost> postList;

private FirebaseFirestore db = FirebaseFirestore.getInstance();

public BortjournalAdapter(Context context, List<BortjournalPost> postList) { this.context = context;

this.postList = postList;

}

@NonNull @Override

public PostViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) { View view = LayoutInflater.from(context).inflate(R.layout.item\_bortjournal, parent, false); return new PostViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull PostViewHolder holder, int position) { BortjournalPost post = postList.get(position);

holder.brandModelTextView.setText(post.getBrand() + " " + post.getModel()); holder.titleTextView.setText(post.getTitle()); holder.mileageTextView.setText("Пробег: " + post.getMileage() + " км");

// Загружаем имя и аватар пользователя db.collection("users").document(post.getUserId())

.get()

.addOnSuccessListener(doc -> {

String name = doc.getString("name");

String imageBase64 = doc.getString("profile\_image");

if (name != null) { holder.userNameTextView.setText(name);

}

if (imageBase64 != null && !imageBase64.isEmpty()) { try {

byte[] decoded = Base64.decode(imageBase64, Base64.DEFAULT);

Bitmap bitmap = BitmapFactory.decodeByteArray(decoded, 0, decoded.length); holder.profileImageView.setImageBitmap(getCircularBitmap(bitmap));

} catch (Exception e) { e.printStackTrace();

}

} else {

holder.profileImageView.setImageResource(R.drawable.logo);

}

});

holder.itemView.setOnClickListener(v -> {

Intent intent = new Intent(context, BortjournalDetailActivity.class); intent.putExtra("brand", post.getBrand());

intent.putExtra("model", post.getModel()); intent.putExtra("title", post.getTitle()); intent.putExtra("content", post.getContent()); context.startActivity(intent);

});

}

@Override

public int getItemCount() { return postList.size();

}

public static class PostViewHolder extends RecyclerView.ViewHolder {

TextView brandModelTextView, titleTextView, mileageTextView, userNameTextView; ImageView profileImageView;

public PostViewHolder(@NonNull View itemView) { super(itemView);

brandModelTextView = itemView.findViewById(R.id.brandModelTextView); titleTextView = itemView.findViewById(R.id.titleTextView); mileageTextView = itemView.findViewById(R.id.mileageTextView); userNameTextView = itemView.findViewById(R.id.userNameTextView); profileImageView = itemView.findViewById(R.id.profileImageView);

}

}

private Bitmap getCircularBitmap(Bitmap bitmap) {

int size = Math.min(bitmap.getWidth(), bitmap.getHeight());

Bitmap output = Bitmap.createBitmap(size, size, Bitmap.Config.ARGB\_8888);

Canvas canvas = new Canvas(output); final Paint paint = new Paint();

final Rect rect = new Rect(0, 0, size, size); final RectF rectF = new RectF(rect);

paint.setAntiAlias(true);

canvas.drawCircle(size / 2f, size / 2f, size / 2f, paint); paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC\_IN)); canvas.drawBitmap(bitmap, null, rect, paint);

return output;

}

}

## BortjournalDetailActivity.java;

package com.example.drivee; import android.os.Bundle;

import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class BortjournalDetailActivity extends AppCompatActivity { private TextView detailBrandModel, detailTitle, detailContent;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.bortjournal\_detail\_activity);

detailBrandModel = findViewById(R.id.detailBrandModel); detailTitle = findViewById(R.id.detailTitle);

detailContent = findViewById(R.id.detailContent);

String brand = getIntent().getStringExtra("brand"); String model = getIntent().getStringExtra("model"); String title = getIntent().getStringExtra("title");

String content = getIntent().getStringExtra("content");

detailBrandModel.setText(brand + " " + model); detailTitle.setText(title); detailContent.setText(content);

}

}

## BortjournalPost.java;

package com.example.drivee; public class BortjournalPost {

private String id; private String brand; private String model; private String title; private String content; private String userId; private long mileage; private String status;

public BortjournalPost() {

}

public String getId() { return id;

}

public void setId(String id) { this.id = id;

}

public String getBrand() { return brand;

}

public void setBrand(String brand) { this.brand = brand;

}

public String getModel() { return model;

}

public void setModel(String model) { this.model = model;

}

public String getTitle() { return title;

}

public void setTitle(String title) { this.title = title;

}

public String getContent() { return content;

}

public void setContent(String content) { this.content = content;

}

public String getUserId() { return userId;

}

public void setUserId(String userId) { this.userId = userId;

}

public long getMileage() { return mileage;

}

public void setMileage(long mileage) { this.mileage = mileage;

}

public String getStatus() { return status;

}

public void setStatus(String status) { this.status = status;

}

}

## Car.java;

package com.example.drivee; import android.util.Log;

import com.google.android.gms.common.api.internal.IStatusCallback;

public class Car { private String id; private String brand; private String model;

private String photoBase64; private long mileage; private long price;

private boolean isFavorite; private String status;

// Конструктор

public Car(String id, String brand, String model, String photoBase64, long mileage, long price) { this.id = id;

this.brand = brand; this.model = model;

this.photoBase64 = photoBase64; this.mileage = mileage;

this.price = price;

}

public String getId() { return id; }

public String getBrand() { return brand; } public String getModel() { return model; }

public String getPhotoBase64() { return photoBase64; } public long getMileage() { return mileage; }

public long getPrice() { return price; }

public boolean isFavorite() { return isFavorite; }

public void setFavorite(boolean favorite) { isFavorite = favorite; } public String getStatus() { return status; }

public void setStatus(String status) { this.status = status; }

}

## CarAdapter.java;

package com.example.drivee;

import android.content.Context; import android.content.Intent; import android.graphics.Bitmap;

import android.graphics.BitmapFactory; import android.util.Base64;

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

import android.view.ViewGroup; import android.widget.ImageButton; import android.widget.ImageView; import android.widget.TextView; import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.auth.FirebaseAuth; import com.google.firebase.firestore.FieldValue;

import com.google.firebase.firestore.FirebaseFirestore;

import java.text.DecimalFormat; import java.util.List;

public class CarAdapter extends RecyclerView.Adapter<CarAdapter.CarViewHolder> { private Context context;

private List<Car> carList;

private boolean showStatus = true; private boolean showFavorite = true;

private FirebaseAuth auth = FirebaseAuth.getInstance(); private FirebaseFirestore db = FirebaseFirestore.getInstance();

public CarAdapter(Context context, List<Car> carList) {

this(context, carList, true, false);

}

public CarAdapter(Context context, List<Car> carList, boolean showFavorite, boolean showStatus) { this.context = context;

this.carList = carList; this.showFavorite = showFavorite; this.showStatus = showStatus;

}

@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.carTitleTextView.setText(car.getBrand() + " " + car.getModel()); DecimalFormat formatter = new DecimalFormat("#,###");

String formattedPrice = formatter.format(car.getPrice()).replace(",", " "); holder.carPriceTextView.setText(formattedPrice + " ₽");

if (showStatus && holder.carStatusTextView != null && car.getStatus() != null) { holder.carStatusTextView.setVisibility(View.VISIBLE);

String statusTranslated; switch (car.getStatus()) {

case "approved":

statusTranslated = "Подтверждено"; break;

case "pending":

statusTranslated = "На проверке"; break;

case "rejected":

statusTranslated = "Отклонено"; break;

default:

statusTranslated = car.getStatus(); break;

}

holder.carStatusTextView.setText("Статус: " + statusTranslated);

} else if (holder.carStatusTextView != null) { holder.carStatusTextView.setVisibility(View.GONE);

}

// Фото из base64

String base64 = car.getPhotoBase64();

if (base64 != null && !base64.isEmpty()) { try {

byte[] decodedBytes = Base64.decode(base64, Base64.DEFAULT);

Bitmap bitmap = BitmapFactory.decodeByteArray(decodedBytes, 0, decodedBytes.length); holder.carImageView.setImageBitmap(bitmap);

} catch (Exception e) { e.printStackTrace();

holder.carImageView.setImageResource(R.drawable.logo);

}

} else {

holder.carImageView.setImageResource(R.drawable.logo);

}

// Кнопка "избранное" if (showFavorite) {

holder.favoriteButton.setVisibility(View.VISIBLE); updateFavoriteIcon(holder.favoriteButton, car.isFavorite());

holder.favoriteButton.setOnClickListener(v -> { boolean isNowFavorite = !car.isFavorite();

car.setFavorite(isNowFavorite); updateFavoriteIcon(holder.favoriteButton, isNowFavorite); updateFavoriteInFirebase(car.getId(), isNowFavorite);

});

} else {

holder.favoriteButton.setVisibility(View.GONE); // скрываем в профиле

}

if (showStatus && holder.deleteButton != null) { holder.deleteButton.setVisibility(View.VISIBLE); holder.deleteButton.setOnClickListener(v -> {

String carId = car.getId(); db.collection("cars").document(carId)

.delete()

.addOnSuccessListener(unused -> { carList.remove(position); notifyItemRemoved(position);

Toast.makeText(context, "Машина удалена", Toast.LENGTH\_SHORT).show();

})

.addOnFailureListener(e -> Toast.makeText(context, "Ошибка при удалении", Toast.LENGTH\_SHORT).show());

});

} else if (holder.deleteButton != null) { holder.deleteButton.setVisibility(View.GONE);

}

long mileage = 0; try {

java.lang.reflect.Field mileageField = car.getClass().getDeclaredField("mileage"); mileageField.setAccessible(true);

Object value = mileageField.get(car);

if (value instanceof Long) mileage = (Long) value;

} catch (Exception e) { e.printStackTrace();

}

holder.carMileageTextView.setText("Пробег: " + mileage + " км");

holder.itemView.setOnClickListener(v -> {

Intent intent = new Intent(context, CenterActivity.class); intent.putExtra("carId", car.getId()); context.startActivity(intent);

});

}

@Override

public int getItemCount() { return carList.size();

}

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

TextView carTitleTextView, carPriceTextView, carStatusTextView, deleteButton; TextView carMileageTextView;

ImageButton favoriteButton;

public CarViewHolder(@NonNull View itemView) { super(itemView);

carImageView = itemView.findViewById(R.id.carImageView); carTitleTextView = itemView.findViewById(R.id.carTitleTextView); carPriceTextView = itemView.findViewById(R.id.carPriceTextView); carMileageTextView = itemView.findViewById(R.id.carMileageTextView); carStatusTextView = itemView.findViewById(R.id.carStatusTextView); favoriteButton = itemView.findViewById(R.id.favoriteButton);

deleteButton = itemView.findViewById(R.id.deleteCarButton); // Добавили

}

}

private void updateFavoriteIcon(ImageButton button, boolean isFavorite) { if (isFavorite) {

button.setImageResource(R.drawable.ic\_heart\_filled);

} else {

button.setImageResource(R.drawable.ic\_heart\_outline);

}

}

private void updateFavoriteInFirebase(String carId, boolean add) { String userId = auth.getCurrentUser().getUid();

if (add) { db.collection("users").document(userId)

.update("favorites", FieldValue.arrayUnion(carId));

} else {

db.collection("users").document(userId)

.update("favorites", FieldValue.arrayRemove(carId));

}

}

}

## CarAdapterUser.java;

package com.example.drivee;

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.google.firebase.firestore.FirebaseFirestore; import java.util.List;

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

private Context context; private List<UserCar> carList;

private FirebaseFirestore db = FirebaseFirestore.getInstance();

public CarAdapterUser(Context context, List<UserCar> carList) { this.context = context;

this.carList = carList;

}

@NonNull @Override

public CarViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) { View view = LayoutInflater.from(context).inflate(R.layout.item\_car\_user, parent, false); return new CarViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull CarViewHolder holder, int position) { UserCar car = carList.get(position);

holder.title.setText(car.getBrand() + " " + car.getModel()); holder.price.setText(car.getPrice() + " ₽"); holder.status.setText("Статус: " + car.getStatus());

// Загрузка фото

if (car.getPhotoUrl() != null && !car.getPhotoUrl().isEmpty()) { Glide.with(context)

.load(car.getPhotoUrl())

.into(holder.imageView);

} else {

holder.imageView.setImageResource(R.drawable.logo);

}

holder.deleteButton.setOnClickListener(v -> {

db.collection("cars").document(car.getId())

.delete()

.addOnSuccessListener(unused -> { carList.remove(position); notifyItemRemoved(position); notifyItemRangeChanged(position, carList.size());

Toast.makeText(context, "Объявление удалено", Toast.LENGTH\_SHORT).show();

});

}

})

.addOnFailureListener(e -> {

Toast.makeText(context, "Ошибка удаления", Toast.LENGTH\_SHORT).show();

});

@Override

public int getItemCount() { return carList.size();

}

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

TextView title, price, status, deleteButton;

public CarViewHolder(@NonNull View itemView) { super(itemView);

imageView = itemView.findViewById(R.id.carImageView); title = itemView.findViewById(R.id.carTitleTextView); price = itemView.findViewById(R.id.carPriceTextView); status = itemView.findViewById(R.id.carStatusTextView);

deleteButton = itemView.findViewById(R.id.deleteCarButton);

}

}

}

## CarForProfile.java;

package com.example.drivee; public class CarForProfile {

private String id; private String brand; private String model; private long price; private String status;

public CarForProfile(String id, String brand, String model, long price, String status) { this.id = id;

this.brand = brand; this.model = model; this.price = price; this.status = status;

}

public String getId() { return id; }

public void setId(String id) { this.id = id; }

public String getBrand() { return brand; }

public void setBrand(String brand) { this.brand = brand; }

public String getModel() { return model; }

public void setModel(String model) { this.model = model; }

public long getPrice() { return price; }

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

public String getStatus() { return status; }

public void setStatus(String status) { this.status = status; }

}

## CenterActivity.java;

package com.example.drivee;

import android.content.Intent;

import android.content.pm.PackageManager; import android.graphics.Bitmap;

import android.graphics.BitmapFactory; import android.net.Uri;

import android.os.Bundle; import android.util.Base64; import android.util.Log;

import android.widget.ImageView; import android.widget.TextView; import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity; import androidx.viewpager2.widget.ViewPager2;

import com.google.firebase.Timestamp;

import com.google.firebase.firestore.DocumentReference; import com.google.firebase.firestore.DocumentSnapshot; import com.google.firebase.firestore.FirebaseFirestore;

import java.text.SimpleDateFormat; import java.util.ArrayList;

import java.util.Date; import java.util.List; import java.util.Locale;

public class CenterActivity extends AppCompatActivity {

private static final String TAG = "CenterActivity"; private ViewPager2 carImagePager;

private ImageView userAvatar;

private TextView userName, userPhone;

private TextView vinTextView, dataPublicTextView, bodyTextView, brandTextView, colorTextView, descriptionTextView; private TextView driveTextView, fuelTextView, horsepowerTextView, mileageTextView, modelTextView;

private TextView ownersTextView, priceTextView, transmissionTextView, volumeTextView, yearTextView;

private FirebaseFirestore db; private String carId;

private ImagePagerAdapter imageAdapter;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_center);

carId = getIntent().getStringExtra("carId"); if (carId == null || carId.isEmpty()) {

Log.e(TAG, "Ошибка: carId не передан!"); finish();

return;

}

vinTextView = findViewById(R.id.vinTextView); userAvatar = findViewById(R.id.userAvatar); userName = findViewById(R.id.userName); userPhone = findViewById(R.id.userPhone); bodyTextView = findViewById(R.id.bodyTextView);

brandTextView = findViewById(R.id.brandTextView); colorTextView = findViewById(R.id.colorTextView); descriptionTextView = findViewById(R.id.descriptionTextView); driveTextView = findViewById(R.id.driveTextView); fuelTextView = findViewById(R.id.fuelTextView); horsepowerTextView = findViewById(R.id.horsepowerTextView); mileageTextView = findViewById(R.id.mileageTextView); modelTextView = findViewById(R.id.modelTextView); carImagePager = findViewById(R.id.carImagePager); dataPublicTextView = findViewById(R.id.dataPublicTextView); ownersTextView = findViewById(R.id.ownersTextView); priceTextView = findViewById(R.id.priceTextView);

transmissionTextView = findViewById(R.id.transmissionTextView);

volumeTextView = findViewById(R.id.volumeTextView); yearTextView = findViewById(R.id.yearTextView);

db = FirebaseFirestore.getInstance();

List<String> initialPhotoUrls = new ArrayList<>(); imageAdapter = new ImagePagerAdapter(this, initialPhotoUrls); carImagePager.setAdapter(imageAdapter);

loadCarDetails();

TextView phoneTextView = findViewById(R.id.userPhone); phoneTextView.setOnClickListener(v -> {

String phoneNumber = phoneTextView.getText().toString().trim(); Intent callIntent = new Intent(Intent.ACTION\_CALL); callIntent.setData(Uri.parse("tel:" + phoneNumber));

if (checkSelfPermission(android.Manifest.permission.CALL\_PHONE) == PackageManager.PERMISSION\_GRANTED)

{

startActivity(callIntent);

} else {

requestPermissions(new String[]{android.Manifest.permission.CALL\_PHONE}, 1);

}

});

}

@Override

public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) { super.onRequestPermissionsResult(requestCode, permissions, grantResults);

if (requestCode == 1) {

if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION\_GRANTED) { Toast.makeText(this, "Разрешение получено, нажмите еще раз для звонка", Toast.LENGTH\_SHORT).show();

} else {

Toast.makeText(this, "Разрешение на звонок не предоставлено", Toast.LENGTH\_SHORT).show();

}

}

}

private void loadCarDetails() {

DocumentReference docRef = db.collection("cars").document(carId);

docRef.get().addOnCompleteListener(task -> { if (task.isSuccessful()) {

DocumentSnapshot document = task.getResult(); if (document.exists()) {

vinTextView.setText("VIN: " + document.getString("vin")); bodyTextView.setText("Кузов: " + document.getString("body")); brandTextView.setText(document.getString("brand")); colorTextView.setText("Цвет: " + document.getString("color")); descriptionTextView.setText(document.getString("description")); driveTextView.setText("Привод: " + document.getString("drive"));

Timestamp timestamp = document.getTimestamp("DataPublic"); if (timestamp != null) {

Date date = timestamp.toDate();

SimpleDateFormat sdf = new SimpleDateFormat("dd MMMM yyyy", Locale.getDefault()); dataPublicTextView.setText(sdf.format(date));

}

fuelTextView.setText("Топливо: " + document.getString("fuel")); horsepowerTextView.setText("Мощность: " + document.getLong("horsepower") + " л.с."); mileageTextView.setText("Пробег: " + document.getLong("mileage") + " км"); modelTextView.setText(document.getString("model")); ownersTextView.setText("Количество владельцев: " + document.getLong("owners")); priceTextView.setText(document.getLong("price") + " ₽"); transmissionTextView.setText("Трансмиссия: " + document.getString("transmission")); volumeTextView.setText("Объем двигателя: " + document.get("volume") + " л"); yearTextView.setText(String.valueOf(document.getLong("year")));

List<String> photoBase64List = new ArrayList<>();

String[] photoFields = {"photoone", "phototwo", "photothree", "photofour", "photofive"};

for (String field : photoFields) {

String base64 = document.getString(field); if (base64 != null && !base64.isEmpty()) {

photoBase64List.add(base64);

}

}

String ownerId = document.getString("ownerId"); if (ownerId != null) {

db.collection("users").document(ownerId).get()

.addOnSuccessListener(userDoc -> { if (userDoc.exists()) {

userName.setText(userDoc.getString("name")); userPhone.setText(userDoc.getString("phone"));

String profileBase64 = userDoc.getString("profile\_image"); if (profileBase64 != null && !profileBase64.isEmpty()) {

byte[] decodedBytes = Base64.decode(profileBase64, Base64.DEFAULT);

Bitmap bitmap = BitmapFactory.decodeByteArray(decodedBytes, 0, decodedBytes.length); userAvatar.setImageBitmap(bitmap);

} else {

userAvatar.setImageResource(R.drawable.user\_prof\_1);

}

}

})

.addOnFailureListener(e -> Log.e(TAG, "Ошибка загрузки данных пользователя", e));

}

imageAdapter.updateImages(photoBase64List);

} else {

Log.e(TAG, "Документ не найден!");

}

} else {

Log.e(TAG, "Ошибка при загрузке данных", task.getException());

}

});

}

}

## CreateListingActivity.java;

package com.example.drivee; import android.Manifest;

import android.content.pm.PackageManager; import androidx.core.app.ActivityCompat; import androidx.core.content.ContextCompat;

import com.google.android.gms.location.FusedLocationProviderClient; import com.google.android.gms.location.LocationServices;

import android.location.Address; import android.location.Geocoder; import java.util.Locale;

import android.app.Activity; import android.content.Intent;

import android.content.pm.PackageManager; import android.graphics.Bitmap;

import android.graphics.BitmapFactory; import android.location.Address; import android.location.Geocoder; import android.net.Uri;

import android.os.Bundle; import android.util.Base64; import android.view.View;

import android.widget.ArrayAdapter; import android.widget.Button; import android.widget.EditText; import android.widget.ImageView; import android.widget.Spinner; import android.widget.Toast;

import androidx.annotation.NonNull; import androidx.annotation.Nullable;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.app.ActivityCompat; import androidx.core.content.ContextCompat;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.firestore.FirebaseFirestore;

import java.io.ByteArrayOutputStream; import java.io.InputStream;

import java.util.ArrayList; import java.util.HashMap; import java.util.List; import java.util.Locale; import java.util.Map;

public class CreateListingActivity extends AppCompatActivity { private List<ImageView> imagePreviews = new ArrayList<>();

private EditText editVin, editBrand, editModel, editYear, editColor, editVolume, editHorsepower,

editMileage, editOwners, editPrice, editDescription, editCity;

private Spinner spinnerDrive, spinnerTransmission, spinnerFuel, spinnerBody; private Button btnCreateListing, btnUploadPhoto, btnResetPhotos;

private final List<Uri> selectedImages = new ArrayList<>(); private static final int MAX\_PHOTOS = 5;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_create\_listing);

if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_FINE\_LOCATION)

!= PackageManager.PERMISSION\_GRANTED) {

ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.ACCESS\_FINE\_LOCATION}, 1);

} else {

getCityFromLocation();

}

imagePreviews.add(findViewById(R.id.imagePreview1)); imagePreviews.add(findViewById(R.id.imagePreview2)); imagePreviews.add(findViewById(R.id.imagePreview3)); imagePreviews.add(findViewById(R.id.imagePreview4)); imagePreviews.add(findViewById(R.id.imagePreview5));

editVin = findViewById(R.id.editVin); editBrand = findViewById(R.id.editBrand); editModel = findViewById(R.id.editModel); editYear = findViewById(R.id.editYear); editColor = findViewById(R.id.editColor); editVolume = findViewById(R.id.editVolume);

editHorsepower = findViewById(R.id.editHorsepower); editMileage = findViewById(R.id.editMileage); editOwners = findViewById(R.id.editOwners); editPrice = findViewById(R.id.editPrice); editDescription = findViewById(R.id.editDescription); editCity = findViewById(R.id.editCity);

spinnerDrive = findViewById(R.id.spinnerDrive); spinnerTransmission = findViewById(R.id.spinnerTransmission); spinnerFuel = findViewById(R.id.spinnerFuel);

spinnerBody = findViewById(R.id.spinnerBody);

btnCreateListing = findViewById(R.id.btnCreateListing); btnUploadPhoto = findViewById(R.id.btnUploadPhoto); btnResetPhotos = findViewById(R.id.btnRemovePhoto5);

String selectedBrand = getIntent().getStringExtra("selectedBrand"); String selectedModel = getIntent().getStringExtra("selectedModel");

if (selectedBrand != null && !selectedBrand.isEmpty()) { editBrand.setText(selectedBrand); editBrand.setEnabled(false);

}

if (selectedModel != null && !selectedModel.isEmpty()) {

editModel.setText(selectedModel); editModel.setEnabled(false);

}

setupSpinners();

btnUploadPhoto.setOnClickListener(v -> openGallery()); btnCreateListing.setOnClickListener(v -> saveListing()); btnResetPhotos.setOnClickListener(v -> resetPhotos());

}

private void getCityFromLocation() {

if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {

return;

}

FusedLocationProviderClient fusedLocationClient = LocationServices.getFusedLocationProviderClient(this); Geocoder geocoder = new Geocoder(this, Locale.getDefault());

fusedLocationClient.getLastLocation().addOnSuccessListener(location -> { if (location != null) {

try {

List<Address> addresses = geocoder.getFromLocation(location.getLatitude(), location.getLongitude(), 1); if (addresses != null && !addresses.isEmpty()) {

String city = addresses.get(0).getLocality(); if (city != null) {

editCity.setText(city);

}

}

} catch (Exception e) { e.printStackTrace();

}

}

});

}

@Override

public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) { super.onRequestPermissionsResult(requestCode, permissions, grantResults);

if (requestCode == 1 && grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION\_GRANTED) { getCityFromLocation();

}

}

private void setupSpinners() {

String[] driveOptions = {"Привод", "Полный", "Передний", "Задний"};

String[] transmissionOptions = {"Коробка", "Автоматическая", "Механика", "Вариатор", "Робот"}; String[] fuelOptions = {"Топливо", "Бензин", "Дизель", "Гибрид", "Электро", "ГБО", "Метан"}; String[] bodyOptions = {"Тип кузова", "Купе", "Седан", "Хэтчбек", "Кроссовер", "Внедорожник"};

spinnerDrive.setAdapter(new ArrayAdapter<>(this, android.R.layout.simple\_spinner\_dropdown\_item, driveOptions)); spinnerTransmission.setAdapter(new ArrayAdapter<>(this, android.R.layout.simple\_spinner\_dropdown\_item,

transmissionOptions));

spinnerFuel.setAdapter(new ArrayAdapter<>(this, android.R.layout.simple\_spinner\_dropdown\_item, fuelOptions)); spinnerBody.setAdapter(new ArrayAdapter<>(this, android.R.layout.simple\_spinner\_dropdown\_item, bodyOptions));

}

private void openGallery() {

if (selectedImages.size() < MAX\_PHOTOS) {

Intent intent = new Intent(Intent.ACTION\_PICK); intent.setType("image/\*"); intent.putExtra(Intent.EXTRA\_ALLOW\_MULTIPLE, true); startActivityForResult(intent, 1);

}

}

@Override

protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) { super.onActivityResult(requestCode, resultCode, data);

if (requestCode == 1 && resultCode == Activity.RESULT\_OK && data != null) { if (data.getClipData() != null) {

int count = data.getClipData().getItemCount();

for (int i = 0; i < count && selectedImages.size() < MAX\_PHOTOS; i++) { Uri imageUri = data.getClipData().getItemAt(i).getUri(); selectedImages.add(imageUri); imagePreviews.get(i).setImageURI(imageUri);

}

} else if (data.getData() != null && selectedImages.size() < MAX\_PHOTOS) { Uri imageUri = data.getData();

selectedImages.add(imageUri); imagePreviews.get(0).setImageURI(imageUri);

}

}

}

private void resetPhotos() { selectedImages.clear();

for (ImageView imageView : imagePreviews) { imageView.setImageResource(android.R.color.darker\_gray);

}

}

private void saveListing() {

String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();

if (spinnerDrive.getSelectedItemPosition() == 0) {

Toast.makeText(this, "Выберите тип привода", Toast.LENGTH\_SHORT).show(); return;

}

if (spinnerTransmission.getSelectedItemPosition() == 0) {

Toast.makeText(this, "Выберите тип коробки", Toast.LENGTH\_SHORT).show(); return;

}

if (spinnerFuel.getSelectedItemPosition() == 0) {

Toast.makeText(this, "Выберите тип топлива", Toast.LENGTH\_SHORT).show(); return;

}

if (spinnerBody.getSelectedItemPosition() == 0) {

Toast.makeText(this, "Выберите тип кузова", Toast.LENGTH\_SHORT).show(); return;

}

String vin = editVin.getText().toString().trim(); if (vin.length() != 17) {

Toast.makeText(this, "VIN должен содержать 17 символов", Toast.LENGTH\_SHORT).show(); return;

}

int year = parseIntSafe(editYear.getText().toString());

int currentYear = java.util.Calendar.getInstance().get(java.util.Calendar.YEAR);

if (year < 1850 || year > currentYear) {

Toast.makeText(this, "Год выпуска должен быть между 1850 и " + currentYear, Toast.LENGTH\_SHORT).show(); return;

}

String color = editColor.getText().toString().trim(); if (!color.matches("^[А-Яа-яA-Za-z\\s]+$")) {

Toast.makeText(this, "Цвет должен содержать только буквы", Toast.LENGTH\_SHORT).show(); return;

}

FirebaseFirestore db = FirebaseFirestore.getInstance();

Map<String, Object> carData = new HashMap<>(); carData.put("vin", vin);

carData.put("brand", editBrand.getText().toString().trim()); carData.put("model", editModel.getText().toString().trim()); carData.put("year", year);

carData.put("ownerId", userId);

carData.put("color", editColor.getText().toString().trim()); carData.put("volume", parseIntSafe(editVolume.getText().toString())); carData.put("horsepower", parseIntSafe(editHorsepower.getText().toString())); carData.put("mileage", parseIntSafe(editMileage.getText().toString()));

carData.put("owners", parseIntSafe(editOwners.getText().toString())); carData.put("price", parseIntSafe(editPrice.getText().toString())); carData.put("description", editDescription.getText().toString().trim()); carData.put("city", editCity.getText().toString().trim()); carData.put("drive", spinnerDrive.getSelectedItem().toString());

carData.put("transmission", spinnerTransmission.getSelectedItem().toString()); carData.put("fuel", spinnerFuel.getSelectedItem().toString()); carData.put("body", spinnerBody.getSelectedItem().toString()); carData.put("status", "pending");

for (int i = 0; i < 5; i++) {

String keyName = "photo" + (i == 0 ? "one" : i == 1 ? "two" :

i == 2 ? "three" :

i == 3 ? "four" : "five");

if (i < selectedImages.size()) { try {

InputStream inputStream = getContentResolver().openInputStream(selectedImages.get(i)); Bitmap bitmap = BitmapFactory.decodeStream(inputStream);

ByteArrayOutputStream outputStream = new ByteArrayOutputStream(); bitmap.compress(Bitmap.CompressFormat.JPEG, 70, outputStream); byte[] imageBytes = outputStream.toByteArray();

String base64 = Base64.encodeToString(imageBytes, Base64.DEFAULT); // Кодируем изображение в Base64 carData.put(keyName, base64); // Добавляем в коллекцию Firestore

} catch (Exception e) { e.printStackTrace(); carData.put(keyName, "");

}

} else {

carData.put(keyName, "");

}

}

db.collection("cars")

.add(carData)

.addOnSuccessListener(documentReference -> {

Toast.makeText(this, "Объявление добавлено и ожидает модерации!", Toast.LENGTH\_SHORT).show(); Intent intent = new Intent(CreateListingActivity.this, AllActivity.class);

startActivity(intent); finish();

})

.addOnFailureListener(e -> {

Toast.makeText(this, "Ошибка: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

});

}

private int parseIntSafe(String value) { try {

return Integer.parseInt(value);

} catch (NumberFormatException e) { return 0;

}

}

}

## CreatePostActivity.java;

package com.example.drivee;

import android.os.Bundle; import android.text.TextUtils;

import android.widget.ArrayAdapter; import android.widget.Button; import android.widget.EditText; import android.widget.Spinner; import android.widget.Toast;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.auth.FirebaseAuth; import com.google.firebase.firestore.FieldValue;

import com.google.firebase.firestore.FirebaseFirestore;

import java.util.ArrayList; import java.util.HashMap; import java.util.List; import java.util.Map;

public class CreatePostActivity extends AppCompatActivity { private Spinner brandSpinner, modelSpinner;

private EditText titleEditText, contentEditText, mileageEditText; private Button publishButton;

private FirebaseFirestore db; private FirebaseAuth auth;

private Map<String, List<String>> brandModelMap = new HashMap<>(); @Override

protected void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState); setContentView(R.layout.create\_post\_activity);

brandSpinner = findViewById(R.id.brandSpinner); modelSpinner = findViewById(R.id.modelSpinner); titleEditText = findViewById(R.id.titleEditText); contentEditText = findViewById(R.id.contentEditText); mileageEditText = findViewById(R.id.mileageEditText); publishButton = findViewById(R.id.publishButton);

db = FirebaseFirestore.getInstance(); auth = FirebaseAuth.getInstance();

loadBrands();

brandSpinner.setOnItemSelectedListener(new android.widget.AdapterView.OnItemSelectedListener() {

@Override public void onItemSelected(android.widget.AdapterView<?> parent, android.view.View view, int pos, long id)

{

String selectedBrand = brandSpinner.getSelectedItem().toString(); loadModelsForBrand(selectedBrand);

}

@Override public void onNothingSelected(android.widget.AdapterView<?> parent) {}

});

publishButton.setOnClickListener(v -> publishPost());

}

private void loadBrands() { db.collection("dfd").document("models").get().addOnSuccessListener(documentSnapshot -> {

if (documentSnapshot.exists()) {

Map<String, Object> data = documentSnapshot.getData(); if (data != null) {

List<String> brands = new ArrayList<>(); for (String key : data.keySet()) {

brands.add(key);

brandModelMap.put(key, (List<String>) data.get(key));

}

});

}

}

ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple\_spinner\_item, brands); adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item); brandSpinner.setAdapter(adapter);

}

private void loadModelsForBrand(String brand) { List<String> models = brandModelMap.get(brand); if (models != null) {

ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple\_spinner\_item, models); adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);

modelSpinner.setAdapter(adapter);

}

}

private void publishPost() {

String brand = brandSpinner.getSelectedItem().toString(); String model = modelSpinner.getSelectedItem().toString(); String title = titleEditText.getText().toString().trim();

String content = contentEditText.getText().toString().trim(); String mileageStr = mileageEditText.getText().toString().trim();

if (TextUtils.isEmpty(brand) || TextUtils.isEmpty(model) || TextUtils.isEmpty(title)

|| TextUtils.isEmpty(content) || TextUtils.isEmpty(mileageStr)) {

Toast.makeText(this, "Пожалуйста, заполните все поля", Toast.LENGTH\_SHORT).show(); return;

}

long mileage; try {

mileage = Long.parseLong(mileageStr);

} catch (NumberFormatException e) {

Toast.makeText(this, "Некорректный пробег", Toast.LENGTH\_SHORT).show(); return;

}

String userId = auth.getCurrentUser().getUid();

Map<String, Object> postMap = new HashMap<>(); postMap.put("brand", brand);

postMap.put("model", model); postMap.put("title", title); postMap.put("content", content); postMap.put("mileage", mileage); postMap.put("userId", userId); postMap.put("status", "pending");

postMap.put("timestamp", FieldValue.serverTimestamp());

db.collection("bortjornal")

.add(postMap)

.addOnSuccessListener(documentReference -> {

Toast.makeText(this, "Запись отправлена на модерацию", Toast.LENGTH\_SHORT).show(); finish();

})

.addOnFailureListener(e -> {

Toast.makeText(this, "Ошибка при сохранении", Toast.LENGTH\_SHORT).show();

});

}

}

## FavoritesActivity.java;

package com.example.drivee;

import android.content.Intent; import android.os.Bundle; import android.util.Log; import android.view.View;

import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity; import androidx.recyclerview.widget.LinearLayoutManager; import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.firestore.FirebaseFirestore;

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

public class FavoritesActivity extends AppCompatActivity { private RecyclerView recyclerView;

private CarAdapter carAdapter; private List<Car> favoriteCars; private FirebaseFirestore db; private FirebaseAuth auth; private View progressBar;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_favorites);

recyclerView = findViewById(R.id.recyclerViewFavorites); progressBar = findViewById(R.id.progressBar); recyclerView.setLayoutManager(new LinearLayoutManager(this));

favoriteCars = new ArrayList<>();

carAdapter = new CarAdapter(this, favoriteCars); recyclerView.setAdapter(carAdapter);

db = FirebaseFirestore.getInstance(); auth = FirebaseAuth.getInstance();

TextView backButton = findViewById(R.id.backButton); backButton.setOnClickListener(v -> {

startActivity(new Intent(FavoritesActivity.this, AllActivity.class)); finish();

});

loadFavoriteCars();

}

private void loadFavoriteCars() { progressBar.setVisibility(View.VISIBLE); String userId = auth.getCurrentUser().getUid();

db.collection("users").document(userId).get()

.addOnSuccessListener(documentSnapshot -> {

if (documentSnapshot.exists() && documentSnapshot.contains("favorites")) { List<String> favoriteCarIds = (List<String>) documentSnapshot.get("favorites"); if (favoriteCarIds != null && !favoriteCarIds.isEmpty()) {

fetchCarsOneByOne(favoriteCarIds);

} else {

progressBar.setVisibility(View.GONE);

}

} else {

progressBar.setVisibility(View.GONE);

}

})

.addOnFailureListener(e -> {

Log.e("FavoritesActivity", "Ошибка загрузки избранного", e); progressBar.setVisibility(View.GONE);

});

}

private void fetchCarsOneByOne(List<String> carIds) { favoriteCars.clear();

for (String carId : carIds) { db.collection("cars").document(carId).get()

.addOnSuccessListener(documentSnapshot -> { if (documentSnapshot.exists()) {

long mileage = documentSnapshot.getLong("mileage") != null ? documentSnapshot.getLong("mileage") : 0;

Car car = new Car( documentSnapshot.getId(), documentSnapshot.getString("brand"), documentSnapshot.getString("model"),

documentSnapshot.getString("photoone"), mileage, documentSnapshot.getLong("price")

);

car.setFavorite(true);

favoriteCars.add(car); carAdapter.notifyDataSetChanged();

}

progressBar.setVisibility(View.GONE);

})

.addOnFailureListener(e -> {

Log.e("FavoritesActivity", "Ошибка загрузки машины", e); progressBar.setVisibility(View.GONE);

});

}

}

}

## FilterActivity.java;

package com.example.drivee;

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

import android.widget.ArrayAdapter; import android.widget.Button; import android.widget.EditText; import android.widget.Spinner;

import androidx.appcompat.app.AppCompatActivity; public class FilterActivity extends AppCompatActivity {

private EditText yearFrom, yearTo, volumeFrom, volumeTo, powerFrom, powerTo, ownersTo, priceFrom, priceTo, cityInput; private Spinner colorSpinner, bodySpinner, driveSpinner, fuelSpinner, transmissionSpinner;

private Button applyButton, cancelButton;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_filter);

yearFrom = findViewById(R.id.yearFrom); yearTo = findViewById(R.id.yearTo); volumeFrom = findViewById(R.id.volumeFrom); volumeTo = findViewById(R.id.volumeTo); powerFrom = findViewById(R.id.powerFrom); powerTo = findViewById(R.id.powerTo); ownersTo = findViewById(R.id.ownersTo); priceFrom = findViewById(R.id.priceFrom); priceTo = findViewById(R.id.priceTo);

cityInput = findViewById(R.id.cityInput);

colorSpinner = findViewById(R.id.colorSpinner); bodySpinner = findViewById(R.id.bodySpinner); driveSpinner = findViewById(R.id.driveSpinner); fuelSpinner = findViewById(R.id.fuelSpinner);

transmissionSpinner = findViewById(R.id.transmissionSpinner);

applyButton = findViewById(R.id.applyButton); cancelButton = findViewById(R.id.cancelButton);

setupSpinners();

applyButton.setOnClickListener(v -> returnFilters()); cancelButton.setOnClickListener(v -> finish());

}

private void setupSpinners() {

String[] fuels = {"Топливо", "Бензин", "Дизель", "Гибрид", "Электро", "ГБО", "Метан"}; String[] bodies = {"Тип кузова", "Купе", "Седан", "Хэтчбек", "Кроссовер", "Внедорожник"}; String[] drives = {"Привод", "Полный", "Передний", "Задний"};

String[] transmissions = {"Коробка", "Автоматическая", "Механика", "Вариатор", "Робот"};

String[] colors = {"Цвет", "Красный", "Черный", "Белый", "Синий", "Зеленый", "Желтый", "Серый"};

setupSpinner(colorSpinner, colors); setupSpinner(bodySpinner, bodies); setupSpinner(driveSpinner, drives); setupSpinner(fuelSpinner, fuels); setupSpinner(transmissionSpinner, transmissions);

}

private void setupSpinner(Spinner spinner, String[] values) {

ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple\_spinner\_item, values); adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item); spinner.setAdapter(adapter);

}

private void returnFilters() { Intent result = new Intent();

result.putExtra("yearFrom", getTextOrNull(yearFrom)); result.putExtra("yearTo", getTextOrNull(yearTo)); result.putExtra("volumeFrom", getTextOrNull(volumeFrom)); result.putExtra("volumeTo", getTextOrNull(volumeTo)); result.putExtra("powerFrom", getTextOrNull(powerFrom)); result.putExtra("powerTo", getTextOrNull(powerTo)); result.putExtra("ownersTo", getTextOrNull(ownersTo)); result.putExtra("priceFrom", getTextOrNull(priceFrom)); result.putExtra("priceTo", getTextOrNull(priceTo)); result.putExtra("city", getTextOrNull(cityInput));

result.putExtra("color", getSpinnerValue(colorSpinner)); result.putExtra("body", getSpinnerValue(bodySpinner)); result.putExtra("drive", getSpinnerValue(driveSpinner)); result.putExtra("fuel", getSpinnerValue(fuelSpinner)); result.putExtra("transmission", getSpinnerValue(transmissionSpinner));

setResult(RESULT\_OK, result); finish();

}

private String getTextOrNull(EditText editText) { String text = editText.getText().toString().trim(); return TextUtils.isEmpty(text) ? null : text;

}

private String getSpinnerValue(Spinner spinner) { if (spinner.getSelectedItemPosition() == 0) {

return null;

}

return spinner.getSelectedItem().toString();

}

}

## ImagePagerAdapter.java;

package com.example.drivee;

import android.content.Context; import android.graphics.Bitmap;

import android.graphics.BitmapFactory; import android.util.Base64;

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

import android.view.ViewGroup; import android.widget.ImageView;

import androidx.annotation.NonNull;

import androidx.viewpager2.widget.ViewPager2; import androidx.recyclerview.widget.RecyclerView;

import java.util.List;

public class ImagePagerAdapter extends RecyclerView.Adapter<ImagePagerAdapter.ImageViewHolder> {

private final Context context;

private List<String> photoBase64List;

public ImagePagerAdapter(Context context, List<String> photoBase64List) { this.context = context;

this.photoBase64List = photoBase64List;

}

@NonNull @Override

public ImageViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) { View view = LayoutInflater.from(context).inflate(R.layout.item\_car\_image, parent, false); return new ImageViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull ImageViewHolder holder, int position) { String base64Image = photoBase64List.get(position);

if (!base64Image.isEmpty()) {

byte[] decodedString = Base64.decode(base64Image, Base64.DEFAULT);

Bitmap decodedByte = BitmapFactory.decodeByteArray(decodedString, 0, decodedString.length); holder.imageView.setImageBitmap(decodedByte);

} else {

holder.imageView.setImageResource(R.drawable.logo);

}

}

@Override

public int getItemCount() { return photoBase64List.size();

}

public void updateImages(List<String> newPhotoBase64List) { this.photoBase64List = newPhotoBase64List; notifyDataSetChanged();

}

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

public ImageViewHolder(View itemView) { super(itemView);

imageView = itemView.findViewById(R.id.imageView);

}

}

}

## MainActivity.java;

package com.example.drivee;

import android.content.Intent; import android.os.Bundle; import android.view.View; import android.widget.EditText; import android.widget.Toast; import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity; import com.google.firebase.auth.FirebaseAuth; import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.firestore.DocumentSnapshot; import com.google.firebase.firestore.FirebaseFirestore;

public class MainActivity extends AppCompatActivity { private EditText emailEditText, passwordEditText; private Button loginButton, registerButton;

private FirebaseAuth mAuth;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_main);

Button resetPasswordButton = findViewById(R.id.btnResetPassword); resetPasswordButton.setOnClickListener(v -> {

startActivity(new Intent(MainActivity.this, ResetPasswordActivity.class));

});

mAuth = FirebaseAuth.getInstance(); emailEditText = findViewById(R.id.email);

passwordEditText = findViewById(R.id.passwordtxt); loginButton = findViewById(R.id.btnVhod); registerButton = findViewById(R.id.btnReg);

loginButton.setOnClickListener(new View.OnClickListener() { @Override

public void onClick(View v) { loginUser();

}

});

registerButton.setOnClickListener(new View.OnClickListener() { @Override

public void onClick(View v) {

Intent intent = new Intent(MainActivity.this, RegActivity.class); startActivity(intent);

}

});

}

private void loginUser() {

String email = emailEditText.getText().toString().trim();

String password = passwordEditText.getText().toString().trim();

if (email.isEmpty() || password.isEmpty()) {

Toast.makeText(MainActivity.this, "Заполните все поля", Toast.LENGTH\_SHORT).show(); return;

}

mAuth.signInWithEmailAndPassword(email, password)

.addOnCompleteListener(this, task -> { if (task.isSuccessful()) {

FirebaseUser user = mAuth.getCurrentUser(); if (user != null) {

FirebaseFirestore db = FirebaseFirestore.getInstance(); db.collection("users").document(user.getUid()).get()

.addOnCompleteListener(roleTask -> {

if (roleTask.isSuccessful() && roleTask.getResult().exists()) { DocumentSnapshot document = roleTask.getResult(); String role = document.getString("role");

if ("admin".equals(role)) {

startActivity(new Intent(MainActivity.this, AdminActivity.class));

} else {

startActivity(new Intent(MainActivity.this, AllActivity.class));

}

finish();

} else {

Toast.makeText(MainActivity.this, "Ошибка загрузки роли", Toast.LENGTH\_SHORT).show();

}

});

}

} else {

Toast.makeText(MainActivity.this, "Ошибка авторизации: " + task.getException().getMessage(), Toast.LENGTH\_LONG).show();

}

});

}

}

## ManageModelsActivity.java;

package com.example.drivee;

import android.os.Bundle; import android.util.Log; import android.view.View; import android.widget.\*;

import androidx.appcompat.app.AppCompatActivity; import com.google.firebase.firestore.\*;

import java.util.\*;

public class ManageModelsActivity extends AppCompatActivity { EditText brandInput, modelInput;

Button addBrandBtn, addModelBtn;

Spinner brandSpinner; FirebaseFirestore db; ArrayAdapter<String> brandAdapter;

List<String> brandList = new ArrayList<>(); Map<String, List<String>> allData = new HashMap<>(); DocumentReference docRef;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_manage\_models);

brandInput = findViewById(R.id.brandInput); modelInput = findViewById(R.id.modelInput); addBrandBtn = findViewById(R.id.addBrandBtn); addModelBtn = findViewById(R.id.addModelBtn); brandSpinner = findViewById(R.id.brandSpinner);

db = FirebaseFirestore.getInstance();

docRef = db.collection("dfd").document("models");

brandAdapter = new ArrayAdapter<>(this, android.R.layout.simple\_spinner\_item, brandList); brandAdapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item); brandSpinner.setAdapter(brandAdapter);

loadBrands(); addBrandBtn.setOnClickListener(v -> {

String brand = brandInput.getText().toString().trim();

if (!brand.isEmpty()) {

docRef.update(brand, new ArrayList<String>())

.addOnSuccessListener(unused -> {

Toast.makeText(this, "Марка добавлена", Toast.LENGTH\_SHORT).show(); loadBrands();

})

.addOnFailureListener(e -> {

Map<String, Object> newBrand = new HashMap<>(); newBrand.put(brand, new ArrayList<String>()); docRef.set(newBrand, SetOptions.merge());

Toast.makeText(this, "Марка создана", Toast.LENGTH\_SHORT).show(); loadBrands();

});

}

});

addModelBtn.setOnClickListener(v -> {

String model = modelInput.getText().toString().trim();

String selectedBrand = brandSpinner.getSelectedItem().toString();

if (!model.isEmpty() && selectedBrand != null) { docRef.get().addOnSuccessListener(snapshot -> {

List<String> models = (List<String>) snapshot.get(selectedBrand); if (models == null) models = new ArrayList<>();

if (!models.contains(model)) { models.add(model); docRef.update(selectedBrand, models);

Toast.makeText(this, "Модель добавлена", Toast.LENGTH\_SHORT).show();

} else {

Toast.makeText(this, "Модель уже существует", Toast.LENGTH\_SHORT).show();

}

});

}

}

});

private void loadBrands() { docRef.get().addOnSuccessListener(documentSnapshot -> {

if (documentSnapshot.exists()) { brandList.clear(); allData.clear();

Map<String, Object> data = documentSnapshot.getData(); for (String brand : data.keySet()) {

brandList.add(brand);

allData.put(brand, (List<String>) data.get(brand));

}

});

}

}

}

brandAdapter.notifyDataSetChanged();

## RegActivity.java;

package com.example.drivee;

import android.content.Intent; import android.os.Bundle; import android.view.View; import android.widget.Button; import android.widget.EditText; import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.auth.FirebaseAuth; import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.firestore.FirebaseFirestore;

import java.util.HashMap; import java.util.Map;

public class RegActivity extends AppCompatActivity {

private EditText emailEditText, passwordEditText, confirmPasswordEditText, nameEditText, phoneEd itText, dobEditText; private Button registerButton, backToLoginButton;

private FirebaseAuth mAuth; private FirebaseFirestore db;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_reg);

mAuth = FirebaseAuth.getInstance(); db = FirebaseFirestore.getInstance();

emailEditText = findViewById(R.id.email); passwordEditText = findViewById(R.id.passwordtxt);

confirmPasswordEditText = findViewById(R.id.confirmPassword); nameEditText = findViewById(R.id.nameEditText);

phoneEditText = findViewById(R.id.phoneEditText); dobEditText = findViewById(R.id.dobEditText); registerButton = findViewById(R.id.btnRegister);

backToLoginButton = findViewById(R.id.btnBackToLogin);

registerButton.setOnClickListener(new View.OnClickListener() { @Override

public void onClick(View v) { registerUser();

}

});

backToLoginButton.setOnClickListener(new View.OnClickListener() { @Override

public void onClick(View v) {

startActivity(new Intent(RegActivity.this, MainActivity.class)); finish();

}

});

}

private void registerUser() {

String email = emailEditText.getText().toString().trim();

String password = passwordEditText.getText().toString().trim();

String confirmPassword = confirmPasswordEditText.getText().toString().trim(); String name = nameEditText.getText().toString().trim();

String phone = phoneEditText.getText().toString().trim(); String dob = dobEditText.getText().toString().trim();

if (email.isEmpty() || password.isEmpty() || confirmPassword.isEmpty() || name.isEmpty() || phone.isEmpty() || dob.isEmpty()) {

Toast.makeText(RegActivity.this, "Заполните все поля", Toast.LENGTH\_SHORT).show(); return;

}

if (!password.equals(confirmPassword)) {

Toast.makeText(RegActivity.this, "Пароли не совпадают", Toast.LENGTH\_SHORT).show(); return;

}

mAuth.createUserWithEmailAndPassword(email, password)

.addOnCompleteListener(this, task -> { if (task.isSuccessful()) {

FirebaseUser user = mAuth.getCurrentUser(); saveUserData(user.getUid(), name, email, phone, dob);

Toast.makeText(RegActivity.this, "Добро пожаловать, " + user.getEmail(), Toast.LENGTH\_SHORT).show();

startActivity(new Intent(RegActivity.this, CenterActivity.class)); finish();

} else {

Toast.makeText(RegActivity.this, "Ошибка регистрации: " + task.getException().getMessage(), Toast.LENGTH\_LONG).show();

}

});

}

private void saveUserData(String userId, String name, String email, String phone, String dob) { Map<String, Object> user = new HashMap<>();

user.put("name", name); user.put("email", email); user.put("phone", phone); user.put("role", "user"); user.put("date\_of\_birth", dob);

user.put("account\_creation\_date", System.currentTimeMillis());

db.collection("users")

.document(userId)

.set(user)

.addOnSuccessListener(aVoid -> {

Toast.makeText(RegActivity.this, "Пользователь зарегистрирован", Toast.LENGTH\_SHORT).show();

})

.addOnFailureListener(e -> {

Toast.makeText(RegActivity.this, "Ошибка сохранения данных", Toast.LENGTH\_SHORT).show();

});

}

}

## ResetPasswordActivity.java;

package com.example.drivee;

import android.os.Bundle; import android.view.View; import android.widget.Button; import android.widget.EditText; import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity; import com.google.firebase.auth.FirebaseAuth;

public class ResetPasswordActivity extends AppCompatActivity { private EditText emailEditText;

private Button resetPasswordButton; private FirebaseAuth mAuth;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_reset\_password);

emailEditText = findViewById(R.id.emailEditText); resetPasswordButton = findViewById(R.id.btnResetPassword);

mAuth = FirebaseAuth.getInstance(); resetPasswordButton.setOnClickListener(v -> {

String email = emailEditText.getText().toString().trim();

if (email.isEmpty()) {

Toast.makeText(ResetPasswordActivity.this, "Введите ваш Email", Toast.LENGTH\_SHORT).show(); return;

}

mAuth.sendPasswordResetEmail(email)

.addOnCompleteListener(task -> { if (task.isSuccessful()) {

Toast.makeText(ResetPasswordActivity.this, "Инструкция по сбросу пароля отправлена на почту", Toast.LENGTH\_LONG).show();

} else {

Toast.makeText(ResetPasswordActivity.this, "Ошибка: " + task.getException().getMessage(), Toast.LENGTH\_LONG).show();

}

});

});

}

}

## SelectBrandActivity.java;

package com.example.drivee;

import android.content.Intent; import android.os.Bundle; import android.util.Log; import android.view.View;

import android.widget.ArrayAdapter; import android.widget.Button; import android.widget.Spinner; import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.firestore.DocumentSnapshot; import com.google.firebase.firestore.FirebaseFirestore;

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

public class SelectBrandActivity extends AppCompatActivity { private Spinner brandSpinner;

private Button nextButton;

private FirebaseFirestore db; private List<String> brandList;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_select\_brand);

brandSpinner = findViewById(R.id.brandSpinner); nextButton = findViewById(R.id.btnNext);

db = FirebaseFirestore.getInstance(); brandList = new ArrayList<>();

loadBrands(); nextButton.setOnClickListener(v -> {

String selectedBrand = brandSpinner.getSelectedItem().toString(); if (selectedBrand.equals("Выберите марку")) {

Toast.makeText(this, "Пожалуйста, выберите марку", Toast.LENGTH\_SHORT).show();

} else {

Intent intent = new Intent(SelectBrandActivity.this, SelectModelActivity.class); intent.putExtra("selectedBrand", selectedBrand);

startActivity(intent);

}

});

}

private void loadBrands() { db.collection("dfd").document("models")

.get()

.addOnCompleteListener(task -> { if (task.isSuccessful()) {

DocumentSnapshot document = task.getResult(); if (document.exists()) {

Map<String, Object> data = document.getData(); if (data != null) {

brandList.add("Выберите марку");

for (Object value : data.keySet()) { // Используем keySet(), так как марки — это ключи brandList.add(value.toString());

}

ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple\_spinner\_item, brandList);

adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item); brandSpinner.setAdapter(adapter);

}

} else {

Log.e("SelectBrandActivity", "Документ models не найден"); Toast.makeText(this, "Марки не найдены", Toast.LENGTH\_SHORT).show();

}

} else {

Log.e("SelectBrandActivity", "Ошибка загрузки: ", task.getException()); Toast.makeText(this, "Ошибка загрузки марок", Toast.LENGTH\_SHORT).show();

}

});

}

}

## SelectModelActivity.java;

package com.example.drivee;

import android.content.Intent; import android.os.Bundle; import android.util.Log; import android.view.View;

import android.widget.ArrayAdapter; import android.widget.Button; import android.widget.Spinner; import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.firestore.DocumentSnapshot; import com.google.firebase.firestore.FirebaseFirestore;

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

public class SelectModelActivity extends AppCompatActivity { private Spinner modelSpinner;

private Button nextButton; private FirebaseFirestore db; private List<String> modelList; private String selectedBrand;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_select\_model);

modelSpinner = findViewById(R.id.modelSpinner); nextButton = findViewById(R.id.btnNext);

db = FirebaseFirestore.getInstance(); modelList = new ArrayList<>();

selectedBrand = getIntent().getStringExtra("selectedBrand"); if (selectedBrand == null || selectedBrand.isEmpty()) {

Toast.makeText(this, "Ошибка: марка не выбрана", Toast.LENGTH\_SHORT).show(); finish();

return;

}

loadModels(); nextButton.setOnClickListener(v -> {

String selectedModel = modelSpinner.getSelectedItem().toString(); if (selectedModel.equals("Выберите модель")) {

Toast.makeText(this, "Пожалуйста, выберите модель", Toast.LENGTH\_SHORT).show();

} else {

Intent intent = new Intent(SelectModelActivity.this, CreateListingActivity.class); intent.putExtra("selectedBrand", selectedBrand); intent.putExtra("selectedModel", selectedModel);

startActivity(intent);

}

});

}

private void loadModels() { db.collection("dfd").document("models")

.get()

.addOnCompleteListener(task -> { if (task.isSuccessful()) {

DocumentSnapshot document = task.getResult(); if (document.exists()) {

Map<String, Object> data = document.getData();

if (data != null && data.containsKey(selectedBrand)) { List<String> models = (List<String>) data.get(selectedBrand); if (models != null && !models.isEmpty()) {

modelList.add("Выберите модель"); modelList.addAll(models);

ArrayAdapter<String> adapter = new ArrayAdapter<>(this,

android.R.layout.simple\_spinner\_item, modelList); adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item); modelSpinner.setAdapter(adapter);

} else {

Toast.makeText(this, "Модели для " + selectedBrand + " не найдены", Toast.LENGTH\_SHORT).show();

}

} else {

Log.e("SelectModelActivity", "Марка " + selectedBrand + " не найдена"); Toast.makeText(this, "Марка не найдена", Toast.LENGTH\_SHORT).show();

}

} else {

Log.e("SelectModelActivity", "Документ models не найден"); Toast.makeText(this, "Модели не найдены", Toast.LENGTH\_SHORT).show();

}

} else {

Log.e("SelectModelActivity", "Ошибка загрузки: ", task.getException()); Toast.makeText(this, "Ошибка загрузки моделей", Toast.LENGTH\_SHORT).show();

}

});

}

}

## SplashActivity.java;

package com.example.drivee;

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

import androidx.appcompat.app.AppCompatActivity;

public class SplashActivity extends AppCompatActivity { private static final int SPLASH\_TIME\_OUT = 3000;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_splash);

new Handler().postDelayed(() -> {

Intent intent = new Intent(SplashActivity.this, MainActivity.class); startActivity(intent);

finish();

}, SPLASH\_TIME\_OUT);

}

}

## User.java;

package com.example.drivee; public class User {

private String name; private String email; private String phone;

public User() {

}

public User(String name, String email, String phone) { this.name = name;

this.email = email; this.phone = phone;

}

public String getName() { return name;

}

public String getEmail() { return email;

}

public String getPhone() { return phone;

}

}

## UserCar.java;

package com.example.drivee; public class UserCar {

private String id; private String brand; private String model; private String photoUrl; private long price; private String status;

public UserCar(String id, String brand, String model, String photoUrl, long price, String status) { this.id = id;

this.brand = brand; this.model = model; this.photoUrl = photoUrl; this.price = price; this.status = status;

}

public String getId() { return id;

}

public String getBrand() { return brand;

}

public String getModel() { return model;

}

public String getPhotoUrl() { return photoUrl;

}

public long getPrice() { return price;

}

public String getStatus() { return status;

}

public void setStatus(String status) { this.status = status;

}

}

## UserProfileActivity.java;

package com.example.drivee;

import android.content.Intent; import android.graphics.Bitmap;

import android.graphics.BitmapFactory; import android.net.Uri;

import android.os.Bundle; import android.util.Base64; import android.util.Log;

import android.widget.Button; import android.widget.ImageView; import android.widget.TextView; import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity; import androidx.recyclerview.widget.LinearLayoutManager; import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.firestore.FirebaseFirestore;

import com.google.firebase.firestore.QueryDocumentSnapshot;

import java.io.ByteArrayOutputStream; import java.io.FileNotFoundException; import java.io.InputStream;

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

public class UserProfileActivity extends AppCompatActivity { private static final int PICK\_IMAGE\_REQUEST = 1;

private ImageView profileImage; private Button selectImageBtn;

private TextView userName, userPhone, userEmail; private RecyclerView userCarsRecyclerView; private List<Car> userCars = new ArrayList<>(); private CarAdapter carAdapter;

private FirebaseFirestore db; private FirebaseAuth mAuth;

private Uri imageUri; private String userId;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_user\_profile);

profileImage = findViewById(R.id.profileImage); selectImageBtn = findViewById(R.id.selectImageBtn); userName = findViewById(R.id.userName); userPhone = findViewById(R.id.userPhone); userEmail = findViewById(R.id.userEmail);

userCarsRecyclerView = findViewById(R.id.userCarsRecyclerView);

mAuth = FirebaseAuth.getInstance(); db = FirebaseFirestore.getInstance();

userId = mAuth.getCurrentUser().getUid();

carAdapter = new CarAdapter(this, userCars, false, true); userCarsRecyclerView.setLayoutManager(new LinearLayoutManager(this)); userCarsRecyclerView.setAdapter(carAdapter);

selectImageBtn.setOnClickListener(v -> openFileChooser()); Button logoutButton = findViewById(R.id.logoutButton); logoutButton.setOnClickListener(v -> {

FirebaseAuth.getInstance().signOut();

Intent intent = new Intent(UserProfileActivity.this, MainActivity.class); intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK); // очищаем back

stack

startActivity(intent);

});

loadUserProfile(); loadUserCars();

}

private void loadUserProfile() {

db.collection("users").document(userId)

.get()

.addOnSuccessListener(documentSnapshot -> { if (documentSnapshot.exists()) {

userName.setText(documentSnapshot.getString("name")); userPhone.setText(documentSnapshot.getString("phone")); userEmail.setText(documentSnapshot.getString("email"));

String profileImageBase64 = documentSnapshot.getString("profile\_image"); if (profileImageBase64 != null && !profileImageBase64.isEmpty()) {

Bitmap decodedImage = decodeBase64(profileImageBase64); profileImage.setImageBitmap(decodedImage);

} else {

profileImage.setImageResource(R.drawable.logo);

}

}

})

.addOnFailureListener(e -> {

Toast.makeText(this, "Ошибка загрузки данных", Toast.LENGTH\_SHORT).show();

});

}

private void loadUserCars() { db.collection("cars")

.get()

.addOnSuccessListener(queryDocumentSnapshots -> { userCars.clear();

for (QueryDocumentSnapshot document : queryDocumentSnapshots) { String ownerId = document.getString("ownerId");

if (ownerId != null && ownerId.equals(userId)) { String id = document.getId();

String brand = document.getString("brand"); String model = document.getString("model");

String photo = document.getString("photoone"); if (photo == null) photo = "";

long price = document.getLong("price") != null ? document.getLong("price") : 0;

String status = document.getString("status") != null ? document.getString("status") : "Неизвестно"; long mileage = document.getLong("mileage") != null ? document.getLong("mileage") : 0;

Car car = new Car(id, brand, model, photo, mileage, price);

try {

java.lang.reflect.Field field = car.getClass().getDeclaredField("status"); field.setAccessible(true);

field.set(car, status);

} catch (Exception e) { e.printStackTrace();

}

userCars.add(car);

}

}

carAdapter.notifyDataSetChanged();

})

.addOnFailureListener(e -> {

Toast.makeText(this, "Ошибка при загрузке машин", Toast.LENGTH\_SHORT).show();

});

}

private void openFileChooser() {

Intent intent = new Intent(Intent.ACTION\_PICK); intent.setType("image/\*");

startActivityForResult(intent, PICK\_IMAGE\_REQUEST);

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) { super.onActivityResult(requestCode, resultCode, data);

if (requestCode == PICK\_IMAGE\_REQUEST && resultCode == RESULT\_OK && data != null && data.getData() != null) {

imageUri = data.getData(); try {

InputStream imageStream = getContentResolver().openInputStream(imageUri); Bitmap selectedImage = BitmapFactory.decodeStream(imageStream); profileImage.setImageBitmap(selectedImage);

String imageBase64 = encodeImageToBase64(selectedImage); saveImageToFirestore(imageBase64);

} catch (FileNotFoundException e) { e.printStackTrace();

Toast.makeText(this, "Ошибка при загрузке изображения", Toast.LENGTH\_SHORT).show();

}

}

}

private String encodeImageToBase64(Bitmap bitmap) {

ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); bitmap.compress(Bitmap.CompressFormat.PNG, 100, byteArrayOutputStream); byte[] byteArray = byteArrayOutputStream.toByteArray();

return Base64.encodeToString(byteArray, Base64.DEFAULT);

}

private Bitmap decodeBase64(String base64String) {

byte[] decodedBytes = Base64.decode(base64String, Base64.DEFAULT);

return BitmapFactory.decodeByteArray(decodedBytes, 0, decodedBytes.length);

}

private void saveImageToFirestore(String imageBase64) { db.collection("users").document(userId)

.update("profile\_image", imageBase64)

.addOnSuccessListener(aVoid -> {

Toast.makeText(this, "Изображение обновлено", Toast.LENGTH\_SHORT).show();

})

.addOnFailureListener(e -> {

Toast.makeText(this, "Ошибка при обновлении изображения", Toast.LENGTH\_SHORT).show();

});

}

}

## circle.xml;

<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="<http://schemas.android.com/apk/res/android>" android:shape="oval">

<solid android:color="#DDDDDD" /> <!-- можно заменить на белый или прозрачный -->

</shape>

## edittext\_background.xml;

<shape xmlns:android="<http://schemas.android.com/apk/res/android>">

<solid android:color="@android:color/white"/>

<corners android:radius="8dp"/>

<stroke

android:width="1dp" android:color="#CCCCCC"/>

<padding android:left="12dp" android:top="8dp" android:right="12dp" android:bottom="8dp"/>

</shape>

## ic\_heart\_filled.xml;

<vector xmlns:android="<http://schemas.android.com/apk/res/android>" android:width="24dp"

android:height="24dp" android:viewportWidth="24" android:viewportHeight="24">

<path

android:fillColor="#FF0000"

android:pathData="M12,21.35l-1.45,-1.32C5.4,15.36 2,12.28 2,8.5C2,5.42 4.42,3 7.5,3C9.24,3 10.91,3.81

12,5.09C13.09,3.81 14.76,3 16.5,3C19.58,3 22,5.42 22,8.5C22,12.28 18.6,15.36 13.45,19.68L12,21.35z"/>

</vector>

## ic\_heart\_outline.xml;

<vector xmlns:android="<http://schemas.android.com/apk/res/android>" android:width="24dp"

android:height="24dp" android:viewportWidth="24" android:viewportHeight="24">

<path

android:fillColor="#808080"

android:pathData="M12,21.35l-1.45,-1.32C5.4,15.36 2,12.28 2,8.5C2,5.42 4.42,3 7.5,3C9.24,3 10.91,3.81

12,5.09C13.09,3.81 14.76,3 16.5,3C19.58,3 22,5.42 22,8.5C22,12.28 18.6,15.36 13.45,19.68L12,21.35z"/>

</vector>

## ic\_launcher\_background.xml;

<?xml version="1.0" encoding="utf-8"?>

<vector

android:height="108dp" android:width="108dp" android:viewportHeight="108" android:viewportWidth="108"

xmlns:android="<http://schemas.android.com/apk/res/android>">

<path android:fillColor="#3DDC84" android:pathData="M0,0h108v108h-108z"/>

<path android:fillColor="#00000000" android:pathData="M9,0L9,108" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M19,0L19,108" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M29,0L29,108" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M39,0L39,108" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M49,0L49,108" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M59,0L59,108" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M69,0L69,108" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M79,0L79,108" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M89,0L89,108" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M99,0L99,108" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M0,9L108,9" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M0,19L108,19" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M0,29L108,29" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M0,39L108,39" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M0,49L108,49" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M0,59L108,59" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M0,69L108,69" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M0,79L108,79"

android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M0,89L108,89" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M0,99L108,99" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M19,29L89,29" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M19,39L89,39" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M19,49L89,49" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M19,59L89,59" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M19,69L89,69" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M19,79L89,79" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M29,19L29,89" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M39,19L39,89" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M49,19L49,89" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M59,19L59,89" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M69,19L69,89" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

<path android:fillColor="#00000000" android:pathData="M79,19L79,89" android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

</vector>

## ic\_launcher\_foreground.xml;

<vector xmlns:android="<http://schemas.android.com/apk/res/android>" xmlns:aapt="<http://schemas.android.com/aapt>" android:width="108dp"

android:height="108dp" android:viewportWidth="108" android:viewportHeight="108">

<path android:pathData="M31,63.928c0,0 6.4,-11 12.1,-13.1c7.2,-2.6 26,-1.4 26,-1.4l38.1,38.1L107,108.928l-32,- 1L31,63.928z">

<aapt:attr name="android:fillColor">

<gradient android:endX="85.84757" android:endY="92.4963" android:startX="42.9492" android:startY="49.59793" android:type="linear">

<item

android:color="#44000000" android:offset="0.0" />

<item

android:color="#00000000" android:offset="1.0" />

</gradient>

</aapt:attr>

</path>

<path

android:fillColor="#FFFFFF" android:fillType="nonZero"

android:pathData="M65.3,45.828l3.8,-6.6c0.2,-0.4 0.1,-0.9 -0.3,-1.1c-0.4,-0.2 -0.9,-0.1 -1.1,0.3l-3.9,6.7c-6.3,-2.8 -13.4,-2.8 -

19.7,0l-3.9,-6.7c-0.2,-0.4 -0.7,-0.5 -1.1,-0.3C38.8,38.328 38.7,38.828 38.9,39.228l3.8,6.6C36.2,49.428 31.7,56.028

31,63.928h46C76.3,56.028 71.8,49.428 65.3,45.828zM43.4,57.328c-0.8,0 -1.5,-0.5 -1.8,-1.2c-0.3,-0.7 -0.1,-1.5 0.4,-2.1c0.5,-0.5

1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1 1.2,1.8C45.3,56.528 44.5,57.328 43.4,57.328L43.4,57.328zM64.6,57.328c-0.8,0 -1.5,-0.5 -1.8,-

1.2s-0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1 1.2,1.8C66.5,56.528 65.6,57.328 64.6,57.328L64.6,57.328z"

android:strokeWidth="1" android:strokeColor="#00000000" />

</vector>

## spinner\_background.xml;

<shape xmlns:android="<http://schemas.android.com/apk/res/android>" android:shape="rectangle">

<solid android:color="@android:color/white"/>

<corners android:radius="12dp"/>

<stroke android:width="1dp" android:color="#DDDDDD"/>

</shape>

## splash\_background.xml;

<?xml version="1.0" encoding="utf-8"?>

<layer-list xmlns:android="<http://schemas.android.com/apk/res/android>">

<item android:drawable="@color/white" />

</layer-list>

## activity\_admin.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>" xmlns:tools="<http://schemas.android.com/tools>" android:layout\_width="match\_parent" android:layout\_height="match\_parent" tools:context=".AdminActivity">

<TextView android:id="@+id/adminTitleText" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Администрирование" android:textSize="22sp" android:textStyle="bold" android:textColor="#000000" android:layout\_marginTop="16dp"

app:layout\_constraintTop\_toTopOf="parent" app:layout\_constraintStart\_toStartOf="parent" app:layout\_constraintEnd\_toEndOf="parent" android:gravity="center" />

<androidx.recyclerview.widget.RecyclerView android:id="@+id/recyclerView" android:layout\_width="0dp" android:layout\_height="0dp" android:layout\_marginTop="12dp"

app:layout\_constraintTop\_toBottomOf="@+id/adminTitleText" app:layout\_constraintBottom\_toTopOf="@+id/bottomNavigationView" app:layout\_constraintStart\_toStartOf="parent" app:layout\_constraintEnd\_toEndOf="parent" />

<com.google.android.material.bottomnavigation.BottomNavigationView android:id="@+id/bottomNavigationView" android:layout\_width="0dp"

android:layout\_height="wrap\_content" android:background="?android:attr/windowBackground" app:menu="@menu/bottom\_nav\_menu" app:layout\_constraintBottom\_toBottomOf="parent" app:layout\_constraintStart\_toStartOf="parent" app:layout\_constraintEnd\_toEndOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

## activity\_admin\_bortjournal.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="16dp" android:background="#F5F5F5">

<TextView android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Модерация записей" android:textSize="20sp" android:textStyle="bold" android:textColor="#000" android:gravity="center" android:paddingBottom="12dp" />

<androidx.recyclerview.widget.RecyclerView android:id="@+id/adminRecyclerView" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:clipToPadding="false" android:paddingBottom="16dp" />

</LinearLayout>

## activity\_all.xml;

<LinearLayout 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"

android:orientation="vertical" android:background="#F5F5F5">

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="50dp" android:background="#FFFFFF" android:orientation="horizontal" android:gravity="center\_vertical" android:paddingStart="16dp" android:paddingEnd="16dp" android:elevation="2dp">

<TextView android:id="@+id/adsCountTextView" android:layout\_width="0dp" android:layout\_height="wrap\_content" android:layout\_weight="1" android:text="0 объявлений" android:textSize="18sp" android:textStyle="bold" android:gravity="center" android:textColor="#000000" />

<ImageButton android:layout\_marginTop="4dp" android:id="@+id/filterButton" android:layout\_width="40dp" android:layout\_height="40dp"

android:background="?attr/selectableItemBackgroundBorderless" android:contentDescription="Фильтр" android:src="@drawable/sor"

android:scaleType="fitCenter" android:padding="8dp" />

</LinearLayout>

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="43dp" android:gravity="center\_vertical" android:orientation="horizontal" android:padding="2dp">

<Spinner android:id="@+id/brandSpinner" android:layout\_width="0dp" android:layout\_height="wrap\_content" android:layout\_weight="1" />

<View

android:layout\_width="12dp" android:layout\_height="wrap\_content" />

<Spinner android:id="@+id/modelSpinner" android:layout\_width="0dp" android:layout\_height="wrap\_content" android:layout\_weight="1" />

</LinearLayout>

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="horizontal" android:gravity="center" android:paddingHorizontal="16dp" android:layout\_marginBottom="8dp">

<Button

android:id="@+id/resetFiltersButton" android:layout\_width="0dp" android:layout\_height="wrap\_content" android:layout\_weight="1" android:text="Сбросить" android:backgroundTint="@color/red" android:textColor="@android:color/white" android:layout\_marginEnd="8dp"/>

<Button

android:id="@+id/paramsButton" android:layout\_width="0dp" android:layout\_height="wrap\_content" android:layout\_weight="1" android:text="Параметры" android:backgroundTint="@color/red" android:textColor="@android:color/white"/>

</LinearLayout>

<androidx.recyclerview.widget.RecyclerView android:id="@+id/recyclerView" android:layout\_width="match\_parent" android:layout\_height="0dp" android:layout\_weight="1" android:padding="16dp" />

<com.google.android.material.bottomnavigation.BottomNavigationView android:id="@+id/bottomNavigation" android:layout\_width="match\_parent"

android:layout\_height="90dp" android:background="@color/white" app:menu="@menu/bottom\_navigation\_menu" />

</LinearLayout>

## activity\_center.xml;

<?xml version="1.0" encoding="utf-8"?>

<android.widget.ScrollView xmlns:android="<http://schemas.android.com/apk/res/android>" android:layout\_width="match\_parent"

android:layout\_height="match\_parent" xmlns:app="<http://schemas.android.com/apk/res-auto>" android:fillViewport="true">

<LinearLayout android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" android:orientation="vertical" android:padding="16dp" android:background="#F5F5F5">

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="horizontal" android:layout\_marginTop="50dp" android:gravity="left">

<TextView android:id="@+id/brandTextView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textSize="25sp" android:textColor="#000"/>

<TextView android:id="@+id/modelTextView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textSize="25sp" android:layout\_marginStart="8dp" android:textColor="#000"/>

<TextView android:id="@+id/editVolume" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textSize="25sp" android:layout\_marginStart="8dp" android:textColor="#000"/>

<TextView android:id="@+id/yearTextView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textSize="25sp" android:layout\_marginStart="8dp" android:textColor="#000"/>

</LinearLayout>

<TextView android:id="@+id/priceTextView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textSize="30sp" android:textStyle="bold" android:layout\_marginTop="4dp"/>

<androidx.viewpager2.widget.ViewPager2 android:id="@+id/carImagePager" android:layout\_width="match\_parent" android:layout\_height="250dp" android:layout\_marginTop="12dp"/>

<TextView android:id="@+id/dataPublicTextView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textSize="12sp" android:textColor="#777" android:layout\_marginTop="4dp" android:layout\_gravity="end"/>

<TextView android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Характеристики" android:textSize="21sp"

android:textStyle="bold" android:textColor="#333" android:layout\_marginTop="12dp"/>

<TextView android:id="@+id/bodyTextView" style="@style/CarSpecItem"/>

<TextView android:id="@+id/colorTextView" style="@style/CarSpecItem"/>

<TextView android:id="@+id/driveTextView" style="@style/CarSpecItem"/>

<TextView android:id="@+id/fuelTextView" style="@style/CarSpecItem"/>

<TextView android:id="@+id/horsepowerTextView" style="@style/CarSpecItem"/>

<TextView android:id="@+id/mileageTextView" style="@style/CarSpecItem"/>

<TextView android:id="@+id/ownersTextView" style="@style/CarSpecItem"/>

<TextView android:id="@+id/transmissionTextView" style="@style/CarSpecItem"/>

<TextView android:id="@+id/volumeTextView" style="@style/CarSpecItem"/>

<TextView android:id="@+id/vinTextView" style="@style/CarSpecItem"/>

<TextView android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Описание" android:textSize="21sp" android:textStyle="bold" android:textColor="#333" android:layout\_marginTop="12dp"/>

<TextView android:id="@+id/descriptionTextView" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:textSize="14sp" android:textColor="#555" android:layout\_marginTop="4dp"/>

<LinearLayout android:id="@+id/userInfoBlock" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="horizontal" android:layout\_marginTop="20dp" android:padding="12dp" android:background="@android:color/white" android:gravity="center\_vertical" android:elevation="2dp">

<com.google.android.material.imageview.ShapeableImageView android:id="@+id/userAvatar"

android:layout\_width="64dp" android:layout\_height="64dp" android:layout\_marginEnd="16dp" android:scaleType="centerCrop" android:src="@drawable/user\_prof\_1" app:shapeAppearanceOverlay="@style/CircleImage" />

<LinearLayout android:layout\_width="0dp" android:layout\_height="wrap\_content" android:layout\_weight="1" android:orientation="vertical">

<TextView android:id="@+id/userName" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textStyle="bold" android:textSize="18sp" android:textColor="#000" />

<TextView android:id="@+id/userPhone" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textSize="16sp" android:textColor="#555" />

</LinearLayout>

</LinearLayout>

</LinearLayout>

</android.widget.ScrollView>

## activity\_create\_listing.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" android:background="#F5F5F5">

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="vertical" android:padding="20dp">

<HorizontalScrollView android:layout\_width="match\_parent" android:layout\_height="200dp" android:layout\_marginBottom="10dp">

<LinearLayout android:orientation="horizontal" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content">

<LinearLayout android:orientation="vertical" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:gravity="center">

<ImageView android:id="@+id/imagePreview1" android:layout\_width="200dp" android:layout\_height="200dp" android:scaleType="centerCrop" android:src="@drawable/pusto"

android:background="@android:color/darker\_gray" android:layout\_marginEnd="10dp"/>

<Button

android:id="@+id/btnRemovePhoto1" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Сбросить" android:layout\_marginTop="5dp" android:backgroundTint="@color/red" android:textColor="@android:color/white"/>

</LinearLayout>

<LinearLayout android:orientation="vertical" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:gravity="center">

<ImageView android:id="@+id/imagePreview2" android:layout\_width="200dp" android:layout\_height="200dp" android:scaleType="centerCrop" android:src="@drawable/pusto"

android:background="@android:color/darker\_gray" android:layout\_marginEnd="10dp"/>

<Button

android:id="@+id/btnRemovePhoto2" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Сбросить" android:layout\_marginTop="5dp" android:backgroundTint="@color/red" android:textColor="@android:color/white"/>

</LinearLayout>

<LinearLayout android:orientation="vertical" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:gravity="center">

<ImageView android:id="@+id/imagePreview3" android:layout\_width="200dp" android:layout\_height="200dp" android:scaleType="centerCrop" android:src="@drawable/pusto"

android:background="@android:color/darker\_gray" android:layout\_marginEnd="10dp"/>

<Button

android:id="@+id/btnRemovePhoto3" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Сбросить" android:layout\_marginTop="5dp" android:backgroundTint="@color/red" android:textColor="@android:color/white"/>

</LinearLayout>

<LinearLayout android:orientation="vertical" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:gravity="center">

<ImageView android:id="@+id/imagePreview4" android:layout\_width="200dp" android:layout\_height="200dp" android:scaleType="centerCrop" android:src="@drawable/pusto"

android:background="@android:color/darker\_gray" android:layout\_marginEnd="10dp"/>

<Button

android:id="@+id/btnRemovePhoto4" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Сбросить" android:layout\_marginTop="5dp" android:backgroundTint="@color/red" android:textColor="@android:color/white"/>

</LinearLayout>

<LinearLayout android:orientation="vertical" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:gravity="center">

<ImageView android:id="@+id/imagePreview5" android:layout\_width="200dp" android:layout\_height="200dp" android:scaleType="centerCrop" android:src="@drawable/pusto"

android:background="@android:color/darker\_gray" android:layout\_marginEnd="10dp"/>

</LinearLayout>

</LinearLayout>

</HorizontalScrollView>

<Button

android:id="@+id/btnUploadPhoto" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Загрузить фото" android:backgroundTint="@color/black" android:textColor="@android:color/white"/>

<Button

android:id="@+id/btnRemovePhoto5" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Сбросить" android:layout\_marginTop="5dp" android:backgroundTint="@color/red" android:textColor="@android:color/white"/>

<EditText android:id="@+id/editVin"

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="VIN" android:inputType="text" android:layout\_marginTop="10dp"/>

<EditText android:id="@+id/editBrand"

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Марка" android:inputType="text" android:layout\_marginTop="10dp"/>

<EditText android:id="@+id/editModel"

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Модель" android:inputType="text" android:layout\_marginTop="10dp"/>

<EditText android:id="@+id/editYear"

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Год выпуска" android:inputType="number" android:layout\_marginTop="10dp"/>

<Spinner android:id="@+id/spinnerFuel" android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" android:layout\_marginTop="10dp" />

<EditText android:id="@+id/editColor"

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Цвет" android:inputType="text" android:layout\_marginTop="10dp"/>

<Spinner android:id="@+id/spinnerBody" android:layout\_width="match\_parent" android:layout\_height="wrap\_content"

android:layout\_marginTop="10dp"/>

<Spinner android:id="@+id/spinnerDrive" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:layout\_marginTop="10dp"/>

<Spinner android:id="@+id/spinnerTransmission" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:layout\_marginTop="10dp"/>

<EditText android:id="@+id/editVolume" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Объем двигателя (л)" android:inputType="numberDecimal" android:layout\_marginTop="10dp"/>

<EditText android:id="@+id/editHorsepower" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Мощность (л.с.)" android:inputType="number" android:layout\_marginTop="10dp"/>

<EditText android:id="@+id/editMileage" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Пробег (км)" android:inputType="number" android:layout\_marginTop="10dp"/>

<EditText android:id="@+id/editOwners" android:layout\_width="match\_parent" android:layout\_height="wrap\_content"

android:hint="Количество владельцев" android:inputType="number" android:layout\_marginTop="10dp"/>

<EditText android:id="@+id/editPrice"

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Цена (₽)" android:inputType="number" android:layout\_marginTop="10dp"/>

<EditText android:id="@+id/editDescription" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Описание" android:inputType="textMultiLine" android:layout\_marginTop="10dp"/>

<EditText android:id="@+id/editCity"

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Город" android:inputType="text" android:layout\_marginTop="10dp"/>

<Button

android:id="@+id/btnCreateListing" android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" android:text="Создать объявление" android:layout\_marginTop="20dp" android:backgroundTint="@color/red" android:textColor="@android:color/white"/>

</LinearLayout>

</ScrollView>

## activity\_favorites.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="16dp" android:background="#F5F5F5">

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="horizontal" android:gravity="center\_vertical">

<TextView android:id="@+id/backButton" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="←" android:textSize="24sp"

android:textColor="@android:color/black" android:paddingEnd="12dp" />

<TextView android:id="@+id/titleText"

android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Избранное" android:textStyle="bold" android:textSize="20sp" android:textColor="#000000" />

</LinearLayout>

<androidx.recyclerview.widget.RecyclerView android:id="@+id/recyclerViewFavorites" android:layout\_width="match\_parent" android:layout\_height="0dp" android:layout\_weight="1" android:layout\_marginTop="16dp" />

<ProgressBar android:id="@+id/progressBar" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:layout\_gravity="center" android:visibility="gone" />

</LinearLayout>

## activity\_filter.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" android:background="#F5F5F5">

<LinearLayout

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:padding="20dp" android:orientation="vertical">

<TextView android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Параметры" android:textSize="24sp" android:textStyle="bold" android:textColor="#000000" android:gravity="center" android:layout\_marginBottom="20dp"/>

<LinearLayout android:orientation="horizontal" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:layout\_marginBottom="12dp">

<EditText android:id="@+id/yearFrom" android:layout\_width="0dp" android:layout\_weight="1" android:layout\_height="48dp" android:hint="Год от" android:inputType="number"

android:background="@drawable/edittext\_background" android:padding="12dp"/>

<View android:layout\_width="12dp" android:layout\_height="match\_parent"/>

<EditText android:id="@+id/yearTo" android:layout\_width="0dp" android:layout\_weight="1" android:layout\_height="48dp" android:hint="Год до" android:inputType="number"

android:background="@drawable/edittext\_background" android:padding="12dp"/>

</LinearLayout>

<Spinner android:id="@+id/colorSpinner" android:layout\_width="match\_parent" android:layout\_height="48dp" android:layout\_marginBottom="12dp"

android:background="@drawable/spinner\_background" android:paddingStart="12dp" android:paddingEnd="12dp" />

<Spinner android:id="@+id/bodySpinner" android:layout\_width="match\_parent" android:layout\_height="48dp" android:layout\_marginBottom="12dp"

android:background="@drawable/spinner\_background" android:paddingStart="12dp" android:paddingEnd="12dp" />

<Spinner android:id="@+id/driveSpinner" android:layout\_width="match\_parent" android:layout\_height="48dp" android:layout\_marginBottom="12dp"

android:background="@drawable/spinner\_background" android:paddingStart="12dp" android:paddingEnd="12dp" />

<Spinner android:id="@+id/fuelSpinner" android:layout\_width="match\_parent" android:layout\_height="48dp" android:layout\_marginBottom="12dp"

android:background="@drawable/spinner\_background" android:paddingStart="12dp"

android:paddingEnd="12dp" />

<Spinner android:id="@+id/transmissionSpinner" android:layout\_width="match\_parent" android:layout\_height="48dp" android:layout\_marginBottom="12dp"

android:background="@drawable/spinner\_background" android:paddingStart="12dp" android:paddingEnd="12dp" />

<LinearLayout android:orientation="horizontal" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:layout\_marginBottom="12dp">

<EditText android:id="@+id/volumeFrom" android:hint="Объем от" android:inputType="numberDecimal" android:layout\_weight="1" android:layout\_width="0dp" android:layout\_height="48dp"

android:background="@drawable/edittext\_background" android:padding="12dp"/>

<View android:layout\_width="12dp" android:layout\_height="match\_parent"/>

<EditText android:id="@+id/volumeTo" android:hint="Объем до" android:inputType="numberDecimal" android:layout\_weight="1" android:layout\_width="0dp" android:layout\_height="48dp"

android:background="@drawable/edittext\_background" android:padding="12dp"/>

</LinearLayout>

<LinearLayout android:orientation="horizontal" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:layout\_marginBottom="12dp">

<EditText android:id="@+id/powerFrom" android:hint="Мощность от" android:inputType="number" android:layout\_weight="1" android:layout\_width="0dp" android:layout\_height="48dp"

android:background="@drawable/edittext\_background" android:padding="12dp"/>

<View android:layout\_width="12dp" android:layout\_height="match\_parent"/>

<EditText android:id="@+id/powerTo" android:hint="до" android:inputType="number" android:layout\_weight="1" android:layout\_width="0dp" android:layout\_height="48dp"

android:background="@drawable/edittext\_background" android:padding="12dp"/>

</LinearLayout>

<EditText android:id="@+id/ownersTo"

android:layout\_width="match\_parent" android:layout\_height="48dp" android:hint="Владельцев до" android:inputType="number" android:background="@drawable/edittext\_background" android:padding="12dp" android:layout\_marginBottom="12dp"/>

<LinearLayout android:orientation="horizontal" android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" android:layout\_marginBottom="12dp">

<EditText android:id="@+id/priceFrom" android:hint="Цена от" android:inputType="number" android:layout\_weight="1" android:layout\_width="0dp" android:layout\_height="48dp"

android:background="@drawable/edittext\_background" android:padding="12dp"/>

<View android:layout\_width="12dp" android:layout\_height="match\_parent"/>

<EditText android:id="@+id/priceTo" android:hint="до" android:inputType="number" android:layout\_weight="1" android:layout\_width="0dp" android:layout\_height="48dp"

android:background="@drawable/edittext\_background" android:padding="12dp"/>

</LinearLayout>

<EditText android:id="@+id/cityInput"

android:layout\_width="match\_parent" android:layout\_height="48dp" android:hint="Город" android:inputType="text"

android:background="@drawable/edittext\_background" android:padding="12dp" android:layout\_marginBottom="12dp"/>

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="horizontal" android:gravity="center">

<Button

android:id="@+id/applyButton" android:layout\_width="0dp" android:layout\_weight="1" android:layout\_height="wrap\_content" android:text="Применить" android:backgroundTint="@color/red" android:textColor="@android:color/white"/>

<View android:layout\_width="12dp" android:layout\_height="match\_parent"/>

<Button

android:id="@+id/cancelButton" android:layout\_width="0dp" android:layout\_weight="1" android:layout\_height="wrap\_content" android:text="Отмена" android:backgroundTint="@color/red" android:textColor="@android:color/white"/>

</LinearLayout>

</LinearLayout>

</ScrollView>

## activity\_main.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:gravity="center"

android:padding="20dp" android:background="@color/white">

<ImageView android:layout\_width="300dp" android:layout\_height="100dp"

android:contentDescription="Логотип компании" android:layout\_marginBottom="60dp" android:src="@drawable/logo" />

<TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Вход" android:textSize="24sp" android:textStyle="bold" android:layout\_marginTop="20dp"/>

<EditText android:id="@+id/email"

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Email" android:inputType="textEmailAddress" android:layout\_marginTop="10dp"/>

<EditText android:id="@+id/passwordtxt" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Пароль" android:inputType="textPassword" android:layout\_marginTop="10dp"/>

<Button

android:id="@+id/btnVhod" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Войти" android:height="50dp" android:layout\_marginTop="50dp" android:backgroundTint="@color/red"/>

<Button

android:id="@+id/btnReg" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Зарегистрироваться" android:height="50dp" android:layout\_marginTop="10dp" android:layout\_marginBottom="30dp" android:backgroundTint="@color/red"/>

<Button

android:id="@+id/btnResetPassword" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Сбросить пароль" android:height="50dp" android:textColor="@color/black" android:backgroundTint="@color/white"/>

</LinearLayout>

## activity\_manage\_models.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" android:padding="24dp">

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:orientation="vertical"

android:gravity="center\_horizontal|center\_vertical">

<TextView android:id="@+id/titleText"

android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Марки и модели" android:textSize="24sp" android:textStyle="bold" android:layout\_marginBottom="32dp" />

<EditText android:id="@+id/brandInput" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Новая марка" android:padding="12dp"

android:background="@drawable/edittext\_background" />

<Button

android:id="@+id/addBrandBtn" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:layout\_marginTop="12dp" android:text="Добавить марку" android:backgroundTint="@color/red" android:textColor="@android:color/white" />

<View

android:layout\_width="match\_parent" android:layout\_height="24dp" />

<Spinner android:id="@+id/brandSpinner" android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

<EditText android:id="@+id/modelInput" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:layout\_marginTop="16dp" android:hint="Новая модель" android:padding="12dp"

android:background="@drawable/edittext\_background" />

<Button

android:id="@+id/addModelBtn" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:layout\_marginTop="12dp" android:text="Добавить модель" android:backgroundTint="@color/red" android:textColor="@android:color/white" />

</LinearLayout>

</ScrollView>

## activity\_reg.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:gravity="center" android:padding="20dp" android:background="@color/white">

<ImageView android:layout\_width="300dp" android:layout\_height="100dp"

android:contentDescription="Логотип компании" android:layout\_marginBottom="20dp" android:src="@drawable/logo" />

<TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Регистрация" android:textSize="24sp" android:textStyle="bold" android:layout\_marginTop="0dp"/>

<EditText android:id="@+id/nameEditText" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Имя" />

<EditText android:id="@+id/dobEditText" android:layout\_width="match\_parent" android:layout\_height="wrap\_content"

android:hint="Дата рождения (дд/мм/гггг)" android:inputType="date" android:layout\_marginTop="10dp"/>

<EditText android:id="@+id/phoneEditText" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Номер телефона" android:inputType="phone" />

<EditText android:id="@+id/email"

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Email" android:inputType="textEmailAddress" android:layout\_marginTop="10dp"/>

<EditText android:id="@+id/passwordtxt" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Пароль" android:inputType="textPassword" android:layout\_marginTop="10dp"/>

<EditText android:id="@+id/confirmPassword" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Подтвердите пароль" android:inputType="textPassword" android:layout\_marginTop="10dp"/>

<Button

android:id="@+id/btnRegister" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Зарегистрироваться" android:height="50dp" android:layout\_marginTop="40dp" android:backgroundTint="@color/red"/>

<Button

android:id="@+id/btnBackToLogin" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Уже есть аккаунт? Войти" android:height="50dp" android:layout\_marginTop="10dp" android:layout\_marginBottom="30dp" android:backgroundTint="@color/red"/>

</LinearLayout>

## activity\_reset\_password.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:gravity="center" android:padding="20dp" android:background="@color/white">

<TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Сброс пароля" android:textSize="24sp" android:textStyle="bold" android:layout\_marginTop="20dp" android:layout\_marginBottom="40dp"/>

<EditText android:id="@+id/emailEditText" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Введите ваш Email" android:inputType="textEmailAddress" android:padding="10dp" android:layout\_marginBottom="20dp"/>

<Button

android:id="@+id/btnResetPassword" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Сбросить пароль" android:textColor="@color/white" android:backgroundTint="@color/red" android:height="50dp" android:layout\_marginTop="50dp"/>

<Button

android:id="@+id/btnBackToLogin" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Помните пароль? Войти" android:height="50dp" android:layout\_marginTop="10dp" android:backgroundTint="@color/red"/>

</LinearLayout>

## activity\_select\_brand.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:gravity="center"

android:padding="20dp" android:background="@color/white">

<TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Выберите марку автомобиля" android:textSize="24sp" android:textStyle="bold" android:textColor="@android:color/black" android:layout\_marginBottom="20dp"/>

<Spinner android:id="@+id/brandSpinner" android:layout\_width="match\_parent" android:layout\_height="wrap\_content"

android:layout\_marginBottom="20dp"/>

<Button

android:id="@+id/btnNext" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Далее" android:backgroundTint="@color/red" android:textColor="@android:color/white" android:height="50dp"/>

</LinearLayout>

## activity\_select\_model.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:gravity="center" android:padding="20dp" android:background="@color/white">

<TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Выберите модель автомобиля" android:textSize="24sp" android:textStyle="bold" android:textColor="@android:color/black" android:layout\_marginBottom="20dp"/>

<Spinner android:id="@+id/modelSpinner" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:layout\_marginBottom="20dp"/>

<Button

android:id="@+id/btnNext" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Далее" android:backgroundTint="@color/red" android:textColor="@android:color/white" android:height="50dp"/>

</LinearLayout>

## activity\_splash.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:gravity="center" android:orientation="vertical" android:background="@color/white">

<ImageView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:contentDescription="App Logo" android:src="@drawable/logo" />

</LinearLayout>

## activity\_user\_profile.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:gravity="center" android:padding="20dp" android:background="@color/white">

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="horizontal" android:gravity="center\_vertical" android:layout\_marginTop="30dp" android:layout\_marginBottom="20dp">

<ImageView android:id="@+id/profileImage" android:layout\_width="100dp" android:layout\_height="100dp" android:layout\_marginEnd="16dp" android:scaleType="centerCrop" android:src="@drawable/user\_prof\_1" android:background="@drawable/circle" android:clipToOutline="true" />

<LinearLayout android:layout\_width="0dp" android:layout\_height="wrap\_content" android:orientation="vertical"

android:layout\_weight="1">

<TextView android:id="@+id/userName" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Имя" android:textSize="20sp" android:textStyle="bold"

android:layout\_marginBottom="8dp" />

<TextView android:id="@+id/userPhone" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Телефон" android:textSize="18sp"

android:layout\_marginBottom="8dp" />

<TextView android:id="@+id/userEmail" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Email"

android:textSize="18sp" android:layout\_marginBottom="20dp" />

</LinearLayout>

</LinearLayout>

<!-- Раздел для кнопок -->

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="vertical" android:gravity="center" android:layout\_marginBottom="20dp">

<Button

android:id="@+id/selectImageBtn" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:layout\_marginBottom="1dp" android:backgroundTint="@color/red" android:text="Выбрать фото" android:textSize="14sp" android:minHeight="45dp" />

<Button

android:id="@+id/logoutButton" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:layout\_marginTop="10dp" android:backgroundTint="@color/black" android:text="Выйти из аккаунта" android:textColor="@color/white" android:textSize="14sp" android:minHeight="45dp" />

</LinearLayout>

<TextView android:id="@+id/userCarsTitle" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Мои машины" android:textSize="18sp" android:textStyle="bold" android:layout\_marginTop="1dp" android:layout\_marginBottom="10dp"/>

<androidx.recyclerview.widget.RecyclerView android:id="@+id/userCarsRecyclerView" android:layout\_width="match\_parent" android:layout\_height="0dp" android:layout\_marginTop="10dp" android:layout\_marginBottom="20dp" android:layout\_weight="1" />

</LinearLayout>

## bortjournal\_activity.xml;

<LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>" android:orientation="vertical" android:layout\_width="match\_parent" android:layout\_height="match\_parent"

android:padding="16dp" android:background="#F5F5F5">

<TextView android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Бортовой журнал" android:textSize="24sp" android:textStyle="bold" android:textColor="#000000" android:gravity="center" android:layout\_marginBottom="20dp"/>

<Button

android:id="@+id/createPostButton" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Создать запись" android:backgroundTint="@color/red" android:textColor="@android:color/white" android:layout\_marginBottom="12dp"/>

<androidx.recyclerview.widget.RecyclerView android:id="@+id/postsRecyclerView" android:layout\_width="match\_parent" android:layout\_height="match\_parent" />

</LinearLayout>

## bortjournal\_detail\_activity.xml;

<ScrollView xmlns:android="<http://schemas.android.com/apk/res/android>" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:padding="16dp">

<LinearLayout android:orientation="vertical" android:layout\_width="match\_parent" android:layout\_height="wrap\_content">

<TextView android:id="@+id/detailBrandModel" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Toyota Camry" android:textSize="16sp" android:textStyle="bold" android:textColor="#555555" />

<TextView android:id="@+id/detailTitle" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Заголовок" android:textSize="20sp" android:textStyle="bold" android:textColor="#000000" android:layout\_marginTop="8dp" />

<TextView android:id="@+id/detailContent" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Текст" android:textSize="16sp" android:layout\_marginTop="12dp" android:textColor="#333333" />

</LinearLayout>

</ScrollView>

## create\_post\_activity.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:padding="16dp">

<LinearLayout android:orientation="vertical" android:layout\_width="match\_parent" android:layout\_height="wrap\_content">

<Spinner

android:id="@+id/brandSpinner" android:layout\_width="match\_parent" android:layout\_height="45dp" android:layout\_marginBottom="16dp" android:padding="12dp" android:textAlignment="center" android:spinnerMode="dropdown" />

<Spinner android:id="@+id/modelSpinner" android:layout\_width="match\_parent" android:layout\_height="45dp" android:layout\_marginBottom="16dp" android:padding="12dp" android:textAlignment="center" android:spinnerMode="dropdown" />

<EditText android:id="@+id/titleEditText" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Заголовок"

android:layout\_marginBottom="12dp" />

<EditText android:id="@+id/contentEditText" android:layout\_width="match\_parent" android:layout\_height="150dp" android:gravity="top" android:hint="Текст записи" android:layout\_marginBottom="12dp" />

<EditText android:id="@+id/mileageEditText" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Пробег (в км)" android:inputType="number" android:layout\_marginBottom="20dp" />

<Button

android:id="@+id/publishButton" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:backgroundTint="@color/red" android:text="Опубликовать" />

</LinearLayout>

</ScrollView>

## item\_admin\_bortjournal.xml;

<?xml version="1.0" encoding="utf-8"?>

<androidx.cardview.widget.CardView 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="wrap\_content" android:layout\_margin="12dp" app:cardCornerRadius="16dp" app:cardElevation="6dp" android:backgroundTint="#FFFFFF">

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="vertical" android:padding="16dp">

<TextView android:id="@+id/adminTitle" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"

android:text="Заголовок поста" android:textStyle="bold" android:textSize="18sp" android:textColor="#222222" />

<TextView android:id="@+id/adminBrandModel" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:layout\_marginTop="4dp" android:text="Марка Модель" android:textSize="16sp" android:textColor="#666666" />

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="horizontal" android:layout\_marginTop="12dp" android:gravity="end">

<Button

android:id="@+id/approveButton" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:backgroundTint="@color/red" android:text="Подтвердить" android:textColor="#FFFFFF" android:layout\_marginEnd="8dp"/>

<Button

android:id="@+id/deleteButton" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:backgroundTint="@color/red" android:text="Удалить" android:textColor="#FFFFFF" />

</LinearLayout>

</LinearLayout>

</androidx.cardview.widget.CardView>

## item\_bortjournal.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="12dp" android:elevation="6dp" card\_view:cardCornerRadius="16dp" card\_view:cardUseCompatPadding="true" android:background="@android:color/white">

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="vertical" android:padding="16dp">

<TextView android:id="@+id/brandModelTextView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textStyle="bold" android:textSize="16sp" android:textColor="#111111" android:layout\_marginBottom="4dp" />

<TextView

android:id="@+id/titleTextView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textStyle="bold" android:textSize="18sp" android:textColor="#222222" android:layout\_marginBottom="6dp" />

<TextView android:id="@+id/mileageTextView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textColor="#666666" android:textSize="14sp" android:layout\_marginBottom="10dp" />

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="horizontal" android:gravity="center\_vertical">

<ImageView android:id="@+id/profileImageView" android:layout\_width="42dp" android:layout\_height="42dp" android:layout\_marginEnd="12dp" android:scaleType="centerCrop" android:background="@drawable/circle" android:src="@drawable/user" />

<TextView android:id="@+id/userNameTextView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textSize="16sp" android:textColor="#444444" android:text="Имя пользователя" />

</LinearLayout>

</LinearLayout>

</androidx.cardview.widget.CardView>

## item\_car.xml;

<?xml version="1.0" encoding="utf-8"?>

<androidx.cardview.widget.CardView 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="wrap\_content" android:layout\_margin="8dp" app:cardCornerRadius="12dp" app:cardElevation="4dp">

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="vertical" android:padding="8dp">

<ImageView android:id="@+id/carImageView" android:layout\_width="match\_parent" android:layout\_height="150dp" android:scaleType="centerCrop"

android:background="@android:color/darker\_gray" />

<TextView android:id="@+id/carTitleTextView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"

android:textSize="18sp" android:textStyle="bold" android:layout\_marginTop="8dp" android:textColor="#000" />

<TextView android:id="@+id/carPriceTextView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textSize="16sp" android:textColor="#333" />

<TextView android:id="@+id/carMileageTextView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Пробег: 0 км" android:textSize="14sp" android:textColor="#666" android:layout\_marginTop="4dp" />

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="horizontal" android:layout\_marginTop="4dp" android:gravity="center\_vertical">

<TextView android:id="@+id/carStatusTextView" android:layout\_width="0dp" android:layout\_height="wrap\_content" android:layout\_weight="1" android:text="Статус:" android:textSize="14sp" android:textColor="#888" />

<TextView android:id="@+id/deleteCarButton" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Удалить" android:textColor="#B00020" android:textSize="14sp" android:paddingStart="12dp"

android:background="@android:color/transparent" />

</LinearLayout>

<ImageButton android:id="@+id/favoriteButton" android:layout\_width="40dp" android:layout\_height="40dp" android:layout\_gravity="end"

android:background="?attr/selectableItemBackgroundBorderless" android:padding="8dp" android:src="@drawable/ic\_heart\_outline" android:contentDescription="Добавить в избранное" />

</LinearLayout>

</androidx.cardview.widget.CardView>

## item\_car\_admin.xml;

<?xml version="1.0" encoding="utf-8"?>

<androidx.cardview.widget.CardView xmlns:android="<http://schemas.android.com/apk/res/android>" android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" android:layout\_margin="8dp" android:elevation="4dp">

<LinearLayout

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="vertical" android:padding="16dp">

<TextView android:id="@+id/tvBrand"

android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Brand" android:textSize="18sp" android:textStyle="bold" />

<TextView android:id="@+id/tvModel"

android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Model" android:textSize="16sp" />

<TextView android:id="@+id/tvPrice"

android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Price" android:textSize="16sp"

android:textColor="@android:color/holo\_green\_dark" />

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="horizontal" android:gravity="end" android:layout\_marginTop="8dp">

<Button

android:id="@+id/btnAccept" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Принять" android:backgroundTint="@color/red"

android:layout\_marginEnd="8dp" />

<Button

android:id="@+id/btnDelete" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:backgroundTint="@color/red" android:text="Удалить" />

</LinearLayout>

</LinearLayout>

</androidx.cardview.widget.CardView>

## item\_car\_image.xml;

<ImageView xmlns:android="<http://schemas.android.com/apk/res/android>" android:id="@+id/imageView"

android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:scaleType="centerCrop"/>

## item\_car\_user.xml;

<?xml version="1.0" encoding="utf-8"?>

<androidx.cardview.widget.CardView 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="wrap\_content" android:layout\_margin="8dp" app:cardCornerRadius="12dp" app:cardElevation="4dp">

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="vertical" android:padding="8dp">

<ImageView android:id="@+id/carImageView" android:layout\_width="match\_parent" android:layout\_height="150dp" android:scaleType="centerCrop"

android:background="@android:color/darker\_gray" />

<TextView android:id="@+id/carTitleTextView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textSize="18sp" android:textStyle="bold" android:layout\_marginTop="8dp" android:textColor="#000" />

<TextView android:id="@+id/carPriceTextView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textSize="16sp" android:textColor="#333" />

<TextView android:id="@+id/carMileageTextView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Пробег: 0 км" android:textSize="14sp" android:textColor="#666" android:layout\_marginTop="4dp" />

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="horizontal" android:layout\_marginTop="4dp" android:gravity="center\_vertical">

<TextView android:id="@+id/carStatusTextView" android:layout\_width="0dp" android:layout\_height="wrap\_content" android:layout\_weight="1" android:text="Статус:" android:textSize="14sp" android:textColor="#888" />

<TextView android:id="@+id/deleteCarButton" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Удалить" android:textColor="#B00020" android:textSize="14sp" android:paddingStart="12dp"

android:background="@android:color/transparent" />

</LinearLayout>

<ImageButton android:id="@+id/favoriteButton" android:layout\_width="40dp"

android:layout\_height="40dp" android:layout\_gravity="end"

android:background="?attr/selectableItemBackgroundBorderless" android:padding="8dp" android:src="@drawable/ic\_heart\_outline" android:contentDescription="Добавить в избранное" />

</LinearLayout>

</androidx.cardview.widget.CardView>

## item\_image.xml;

<?xml version="1.0" encoding="utf-8"?>

<ImageView xmlns:android="<http://schemas.android.com/apk/res/android>" android:id="@+id/imageView"

android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:scaleType="centerCrop"/>

## 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\_manage\_models" android:icon="@drawable/pluss" android:title="Марки и модели" />

<item

android:id="@+id/nav\_manage\_bort" android:icon="@drawable/garage" android:title="Бортжурнал" />

</menu>

## bottom\_navigation\_menu.xml;

<!-- res/menu/bottom\_navigation\_menu.xml -->

<menu xmlns:android="<http://schemas.android.com/apk/res/android>">

<item

android:id="@+id/nav\_home" android:icon="@drawable/home" />

<item

android:id="@+id/nav\_saved" android:icon="@drawable/star" />

<item

android:id="@+id/nav\_create" android:icon="@drawable/pluss" />

<item

android:id="@+id/nav\_garage" android:icon="@drawable/garage" />

<item

android:id="@+id/nav\_profile" android:icon="@drawable/user" />

</menu>

## colors.xml;

<?xml version="1.0" encoding="utf-8"?>

<resources>

<color name="black">#FF000000</color>

<color name="white">#FFFFFFFF</color>

<color name="red">#ba0f1d</color>

</resources>

## strings.xml;

<resources>

<string name="app\_name">Drivee</string>

</resources>

## styles.xml;

<?xml version="1.0" encoding="utf-8"?>

<resources>

<style name="CircleImage" parent="">

<item name="cornerFamily">rounded</item>

<item name="cornerSize">50%</item>

</style>

</resources>

## values.xml;

<?xml version="1.0" encoding="utf-8"?>

<resources>

<string name="gcm\_defaultSenderId" translatable="false">1034230179468</string>

<string name="google\_api\_key" translatable="false">AIzaSyBBQRNebcXKiFtSWDjlTZE56yuXzamgpGA</string>

<string name="google\_app\_id" translatable="false">1:1034230179468:android:36f2445a4d7d74f680bda9</string>

<string name="google\_crash\_reporting\_api\_key" translatable="false">AIzaSyBBQRNebcXKiFtSWDjlTZE56yuXzamgpGA</string>

<string name="google\_storage\_bucket" translatable="false">drivee-42a66.firebasestorage.app</string>

<string name="project\_id" translatable="false">drivee-42a66</string>

</resources>

## build.gradle;

plugins { alias(libs.plugins.android.application)

alias(libs.plugins.google.gms.google.services)

}

android {

namespace 'com.example.drivee' compileSdk 35

defaultConfig {

applicationId "com.example.drivee" minSdk 24

targetSdk 35

versionCode 1

versionName "1.0"

testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"

}

buildTypes { release {

minifyEnabled false

proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'

}

}

compileOptions {

sourceCompatibility JavaVersion.VERSION\_11 targetCompatibility JavaVersion.VERSION\_11

}

}

dependencies {

implementation 'com.github.bumptech.glide:glide:4.12.0' implementation libs.firebase.firestore

implementation libs.play.services.location

annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0' implementation libs.appcompat

implementation libs.material

implementation 'com.squareup.retrofit2:retrofit:2.9.0' implementation 'com.squareup.retrofit2:converter-gson:2.9.0' implementation 'com.squareup.okhttp3:okhttp:4.9.3' implementation libs.activity

implementation libs.constraintlayout

implementation 'com.squareup.picasso:picasso:2.71828' implementation libs.firebase.auth

testImplementation libs.junit androidTestImplementation libs.ext.junit androidTestImplementation libs.espresso.core implementation 'androidx.viewpager2:viewpager2:1.0.0'

}

## libs.versions.toml;

[versions] agp = "8.8.0"

junit = "4.13.2"

junitVersion = "1.2.1"

espressoCore = "3.6.1"

appcompat = "1.7.0"

material = "1.12.0"

activity = "1.10.1"

constraintlayout = "2.2.1"

firebaseAuth = "23.2.0"

googleGmsGoogleServices = "4.4.2"

firebaseFirestore = "25.1.3"

playServicesLocation = "21.3.0"

[libraries]

junit = { group = "junit", name = "junit", version.ref = "junit" }

ext-junit = { group = "androidx.test.ext", name = "junit", version.ref = "junitVersion" }

espresso-core = { group = "androidx.test.espresso", name = "espresso-core", version.ref = "espressoCore" } appcompat = { group = "androidx.appcompat", name = "appcompat", version.ref = "appcompat" }

material = { group = "com.google.android.material", name = "material", version.ref = "material" } activity = { group = "androidx.activity", name = "activity", version.ref = "activity" }

constraintlayout = { group = "androidx.constraintlayout", name = "constraintlayout", version.ref = "constraintlayout" } firebase-auth = { group = "com.google.firebase", name = "firebase-auth", version.ref = "firebaseAuth" }

firebase-firestore = { group = "com.google.firebase", name = "firebase-firestore", version.ref = "firebaseFirestore" } play-services-location = { group = "com.google.android.gms", name = "play-services-location", version.ref = "playServicesLocation" }

[plugins]

android-application = { id = "com.android.application", version.ref = "agp" }

google-gms-google-services = { id = "com.google.gms.google-services", version.ref = "googleGmsGoogleServices" }

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет им. Г.В. Плеханова» Московский приборостроительный техникум

## Проверка на уникальность авторского текста выпускного квалификационного проекта

|  |  |
| --- | --- |
| На тему: | Разработка мобильного приложения для продажи и покупки  автомобилей |

ПУШКИН ИЛЬЯ АЛЕКСАНДРОВИЧ

Студент 3 курса группы П50-3-22

по специальности 09.02.07 «Информационные системы и программирование» квалификация: Программист

Уникальность текста проверялась на сайте [www.antiplagiat.ru.](http://www.antiplagiat.ru/) Процент оригинальности содержимого пояснительной записки составил 100%. Результат представлен на Рисунке 1.

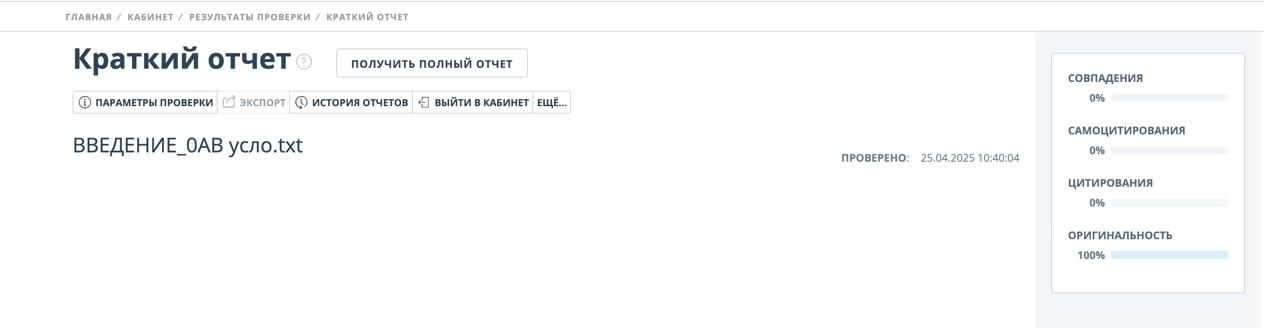


Рисунок 1 - Проверка на уникальность текста пояснительной записки