МИНОБРНАУКИ РОССИИ

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

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

КУРСОВАЯ РАБОТА

по дисциплине «Разработка мобильных приложений»

Тема курсовой работы
Приложение-помощник по уходу за животными «MyPets»

Студент группы ИКБО-07-20 Узоров Кирилл Александрович

(подпись студента)

Руководитель курсовой работы доцент Синицын И.В.

(подпись руководителя)

Работа представлена к защите «11» июня 2022 г.

Допущен к защите «11» июня 2022 г.

Москва 2020

МИНОБРНАУКИ РОССИИ

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

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Утверждаю
Заведующий кафедрой МОСИТ
_____ Головин С.А.
«15» февраля 2022 г.

ЗАДАНИЕ
на выполнение курсовой работы по дисциплине

«Разработка программных приложений»

Студент Узоров Кирилл Александрович Группа ИКБО-07-20

Тема работы: Приложение-помощник по уходу за животными «MyPets»
Исходные данные:
Задание на курсовую работу - Приложение-помощник по уходу за животными «MyPets»
Функционал разрабатываемой системы должен предоставлять возможности полностью интерактивной системы с понятным дружественным графическим интерфейсом и обеспечивать необходимую функциональность, которая формируется в зависмости от заданной темы и предметной области изучаемых вопросов.
Перечень вопросов, подлежащих разработке, и обязательного графического материала:
Установка и настройка Android с применением виртуальных сред. Установка и настройка сред и систем программирования:Android Studio,Eclipse ide , IntelliJ IDEA, Kivi и подобные.Установка и настройка эмуляторов Android.
Изучение жизненного цикла программ и создание мобильного программного комплекса с применением языков программирования JAVA, Phyton, Kotlin,Swift, и др. (применяются по отдельному указанию руководителя), согласно темы курсового проектирования. Реализация в создаваемом программном комплексе, хранения данных в виде файлов. Обеспечение их создания, чтения, записи во внутреннем и внешнем хранилище.
Возможна реализация клиентской части на смартфоне или планшетнике через WAP или подобную систему. Обеспечение безопасности информации при работе планового программного комплекса(обеспечение работы ролевой модели безопасности).
Тестирование и диагностика созданного программного продукта.
Возможно портирование написанной программной системы на внешних хостах в сети Интернет.
Отчет по курсовой работе в виде расчетно-пояснительной записки .

Срок представления к защите курсовой работы: до «_23__»мая 2022 г.

Задание на курсовую работу выдал _____ Синицын И.В

«15» февраля 2022 г.

Задание на курсовую работу получил _____ (Узоров К.А.)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В наши дни почти каждый человек имеет собственного домашнего питомца. По этой причине, к сожалению, не редко люди допускают ошибки в уходе за животными.

Неправильное кормление, вредные активности, отсутствие внимания или наоборот постоянное взаимодействие с питомцем, не дающее ему отдохнуть и восстановить силы - всё это примеры частых ошибок, которые встречаются почти у каждого владельца домашних животных.

Это неудивительно, ведь приобрести домашнего питомца может кто угодно и для этого не нужно каких-то специальных знаний. Тем не менее, любой человек, без сомнения, хочет, чтобы его питомец был всегда здоровым, активным и приносил радость ему и его близким, был настоящим членом семьи. Для этого, крайне важно выполнять основные требования и избегать наиболее вредящих домашнему питомцу ошибок.

К сожалению, в настоящий момент, ресурсов, которые систематизировали бы большинство накопленных знаний об уходе за домашними животными не так много, а те, которые существуют, крайне немобильны, требуют подключения к интернету и не имеют понятного пользователям интерфейса, тем самым не позволяя широкому кругу людей поддерживать гармоничное развитие своих питомцев. Данную проблему может помочь решить приложение-помощник в уходе за животными «MyPets», реализации которого и посвящена данная курсовая работа

Объектом исследования данной курсовой работы являются владельцы домашних животных и их взаимодействие с питомцами.

Предметной областью данной курсовой работы является сфера ухода за животными.

Практическая значимость данной курсовой работы заключается в реализации программного продукта, способного облегчить процесс ухода за домашними питомцами.

Актуальность темы курсовой работы обусловлена тем, что огромное число людей имеют домашних питомцев и с каждым годом это количество только растёт, а значит растёт необходимость информирования владельцев животных о верных методах ухода.

Целью курсовой работы является создание мобильного сервиса, способного загружать ленту статей про животных и показывать её в удобочитаемом виде на экране пользователя. Помимо этого, мобильный сервис должен позволять пользователю делать фотографии и сохранять их в облаке. Данная функция необходима из-за того, что каждый владелец животных любит делать фото своих питомцев. Но фотографии могут теряться во множестве других, а также занимать память мобильного устройства. В создаваемом мобильном сервисе все фотографии

питомца будут храниться в одном месте во внутренней встроенной галерее, а также загружаться в облако, чтобы не занимать память смартфона.

Задачи курсовой работы:

- провести анализ и описание предметной области;
- выбрать средства разработки и выполнить их установку;
- спроектировать компоненты мобильного приложения;
- реализовать и протестировать компоненты мобильного приложения;

1. Анализ предметной области

Для более глубокого анализа предметной области я бы хотел привести статистику ВЦИОМ, которая говорит, что у 61% россиян есть домашние питомцы (ссылки на статью с данными приложены в конце курсовой работы). Это огромная цифра, означающая, что более, чем каждый второй человек в России имеет одно или нескольких домашних животных. Это не случайно, потому как в наше время завести, к примеру, кошку или собаку очень просто. Для этого не нужно иметь специальных жилищных условий, или высокий доход, достаточно приобрести питомца в зоомагазине и привезти домой. По этой причине домашние питомцы очень распространены. Они поднимают настроение своим владельцам, избавляют от чувства одиночества, помогают бороться с психическими заболеваниями, например такими, как депрессия. Но вследствие этой же причины домашние питомцы стали своеобразным товаром. Не каждый задумывается, что иметь живое существо у себя дома это ответственность и от отношения к животному зависит его здоровье и его жизнь. Для подкрепления своих слов приведу цитату из статьи «Зверушки не игрушки: как безответственность хозяев убивает питомцев» (ссылка на данный сайт приложена в конце курсовой работы): «Необходимо осознавать, сколько сил, времени и денег потребует питомец. Не все адекватно оценивают свои возможности, а когда начинаются проблемы — во всём винят животное». Из данных слов становится понятно, что проблема безответственности по отношению к домашним животным очень распространена. Страшно представить, сколько вреда может нанести человек беззащитному животному из-за незнания основных правил ухода за питомцем. По моему мнению, хотя бы частичным решением данной проблемы могло бы стать мобильное приложение-руководство по уходу за животными, с понятным пользователю интерфейсом, и основными советами, статьями, расширяющими круг знаний человека на тему ухода за домашними питомцами. Я уверен, что большинство владельцев животных желают им только добра, но не всегда знают, как правильно выразить свою любовь к ним и как реагировать на проявление любви с их стороны. Поэтому важно распространять информацию, бороться с мифами, вредящими животным, делать накопленные знания об уходе всё более и более открытыми. Эту цель и преследует моё приложение «MyPets», реализуемое в данной курсовой работе.

2. Постановка задачи на разработку программной системы

Выше описана необходимость разработки мобильного приложения, поэтому разработаем прототип мобильного сервиса, обеспечивающего хранение информации о питомце и его фотографий в БД и предоставляющего статьи с советами о правильном уходе за животными. Приложение должно иметь визуально понятный и удобный интерфейс, где пользователь будет иметь возможность вводить некоторые данные о своём питомце, такие как имя, и его тип. Впоследствии на основе полученных данных пользователь будет иметь возможность получить индивидуальный идентификатор, который позволит ему:

- Авторизовываться в системе

- Хранить фотографии в облаке по уникальному адресу и получать доступ к этим фотографиям

- Хранить информацию о питомце в базе данных

Помимо этого, приложение должно предоставлять возможность пользователю просматривать статьи с советами по уходу за животными и давать возможность прочесть эти статьи в удобном для пользователя браузере, установленном на его мобильном устройстве.

Также приложение должно реализовывать ролевую модель безопасности, предоставляя доступ к системе пользователю и администратору.

3. Средства разработки Android-приложения

Для разработки мобильного приложения под Android используют язык программирования Java. Другим языком для Android-разработки является Kotlin, представленный компанией Google на конференции Google I/O в 2017.

Kotlin совместим с Java, отличия: для первого необходимо меньше служебного кода и он легче для чтения, а, следовательно, и для разработки.

Android Studio – разработана Google и предложена в 2013 году на конференции Google I/O. Среда написана на языках Java и Kotlin, является кроссплатформенной, бесплатной и свободно распространяемой. Android Studio является максимально удобной для разработки мобильных приложений для ОС Android, имеет удобные встроенные инструменты.

Firebase — это платформа для разработки приложений, которая помогает создавать и развивать приложения и игры, которые нравятся пользователям. Поддерживается Google и пользуется доверием миллионов компаний по всему миру.

4. Установка средств разработки Android-приложения

Установка Android Studio под Windows.

Перейдём по ссылке: https://developer.android.com/studio?utm_source=udacity&utm_medium=course&utm_campaign=android_basics

Попадём на страницу показанную на рисунке 3.1.

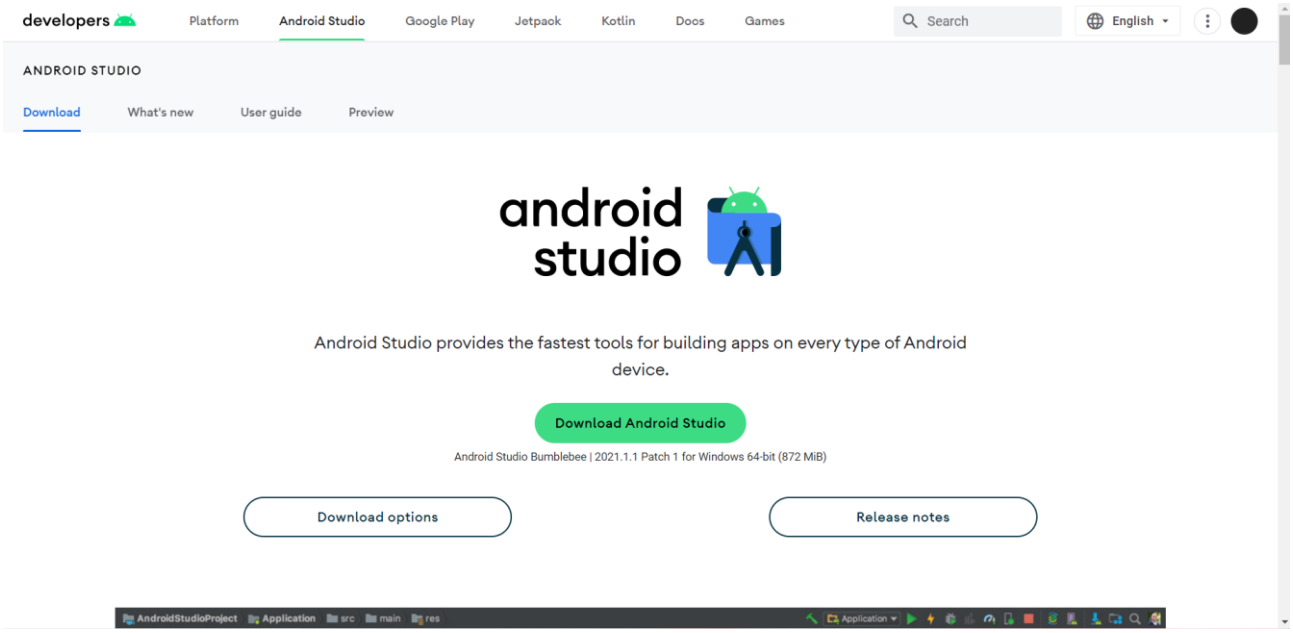

Рисунок 3.1 - Страница загрузки Android Studio

Нажмём кнопку “Download Android Studio”, дождёмся окончания загрузки, затем откроем скачанный файл и приступим к установке (см. рисунок 3.2).

Рисунок 3.2 – Мастер установки Android Studio

Нажимая на кнопку “next” попадём в диалоговое окно, где мастер установки предложить выбрать компоненты для установки (см. рисунок 3.3). Оставим настройки по умолчанию и перейдём к следующему окну.

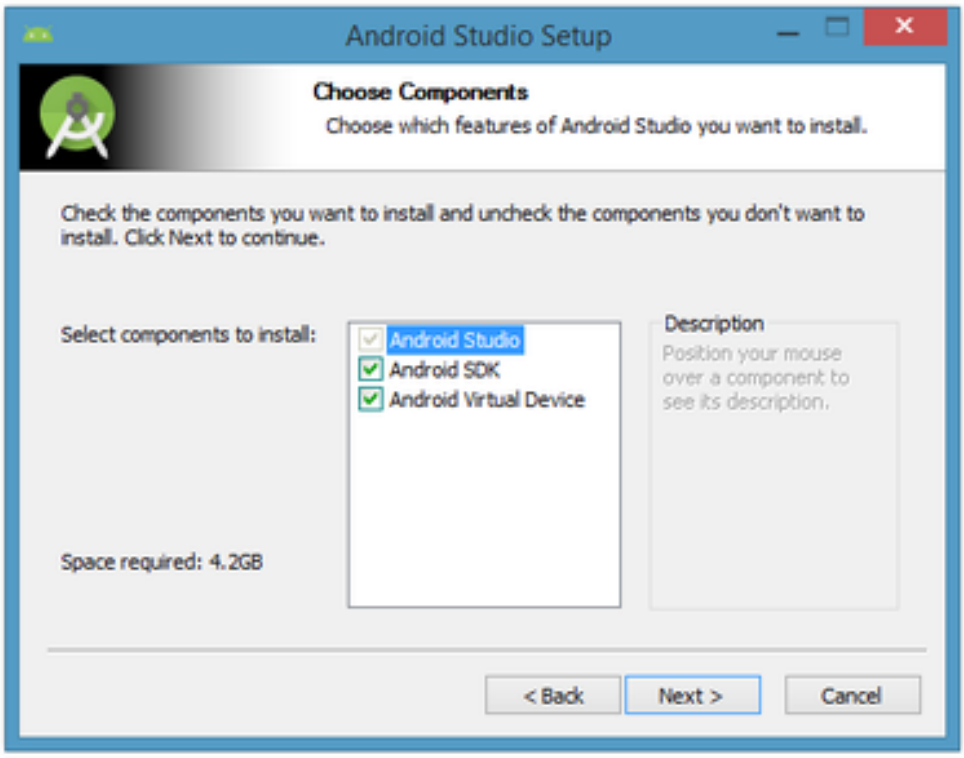

Рисунок 3.3 – Выбор устанавливаемых компонентов Android Studio

В открывшемся окне представлено лицензионное соглашение, примем его, чтобы продолжить процесс установки (см. рисунок 3.4).

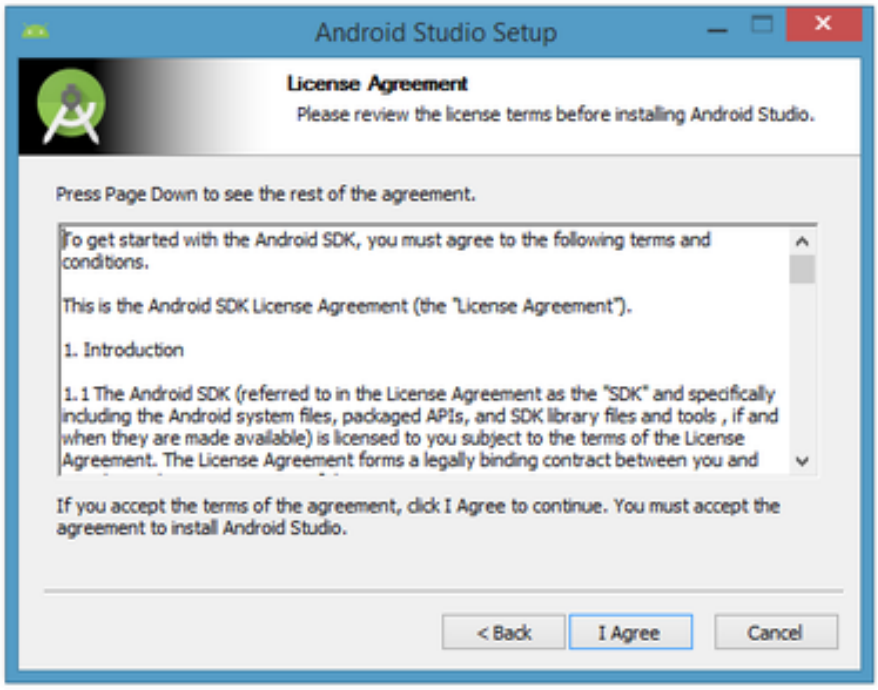

Рисунок 3.4 – Лицензионное соглашение

После нажатия кнопки “I agree”, попадём в окно, где мастер установки предложит изменить папку в которую осуществляется установка Android Studio (см. рисунок 3.5).

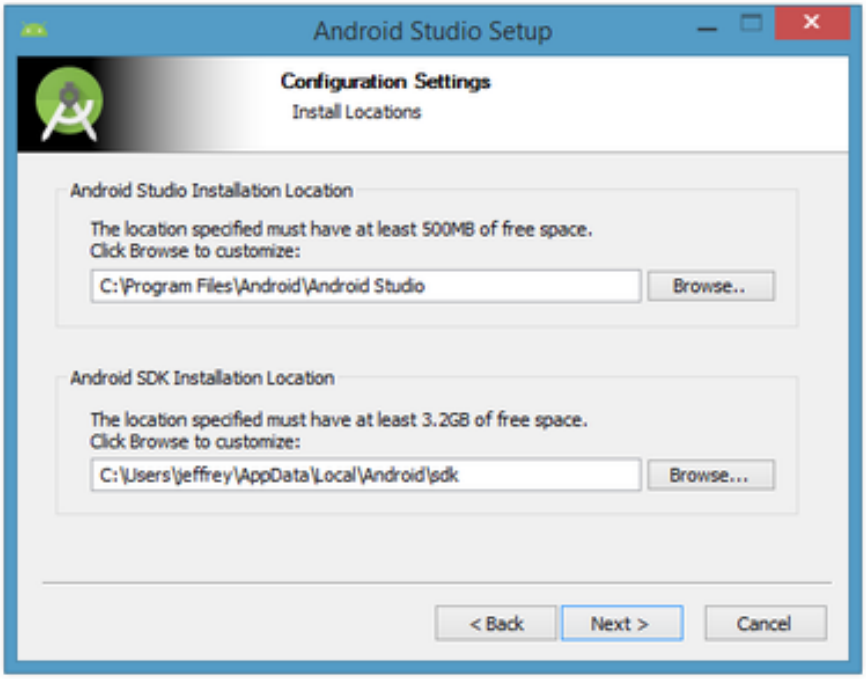

Рисунок 3.5 – Выбор пути установки

Перейдём к следующему окну, в котором мастер установки предложит выбрать папку, где будет создан ярлык для запуска программы (см. рисунок 3.6). Оставим настройки по умолчанию и нажмём кнопку “Install”.

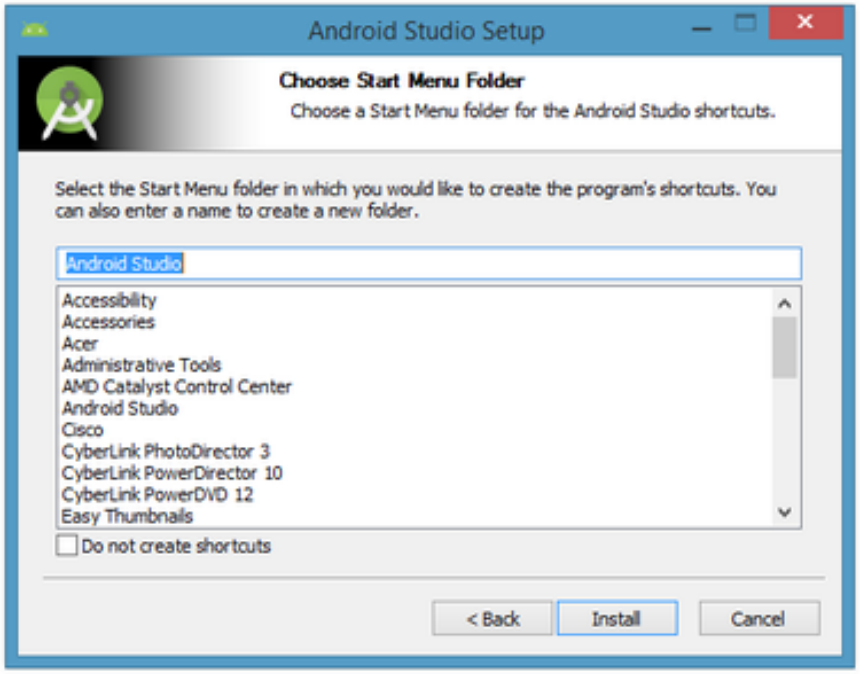

Рисунок 3.6 – Выбор папки, в которой создастся ярлык приложения

Дождёмся конца установки, после чего увидим окно, показанное на рисунке 3.7.

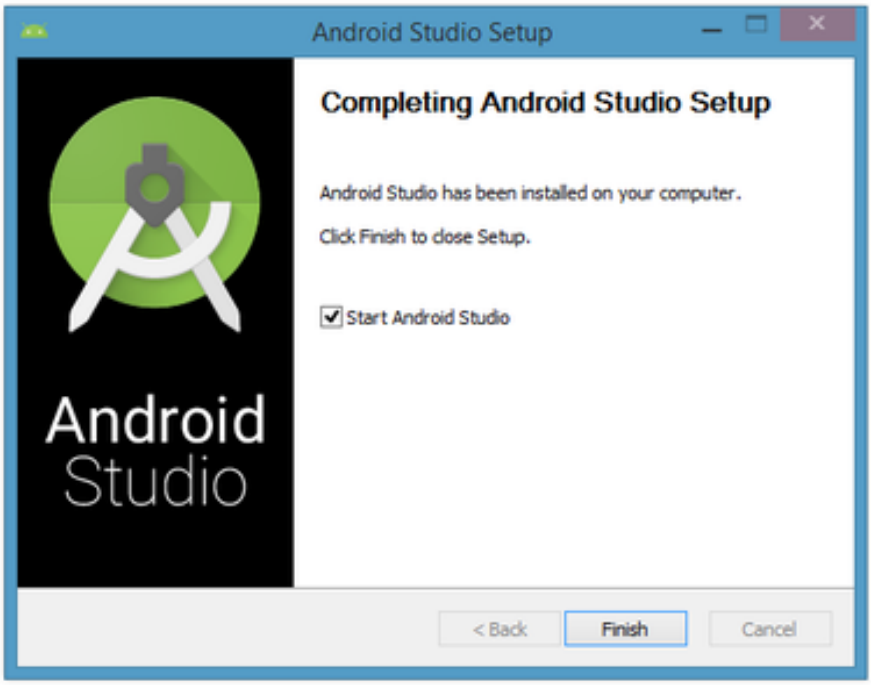

Рисунок 3.7 - Завершение установки

Таким образом, Android Studio была успешно установлена.

Подключение к Firebase

В главной консоли Firebase нажмём кнопку «Add project» (см. рисунок 3.8).

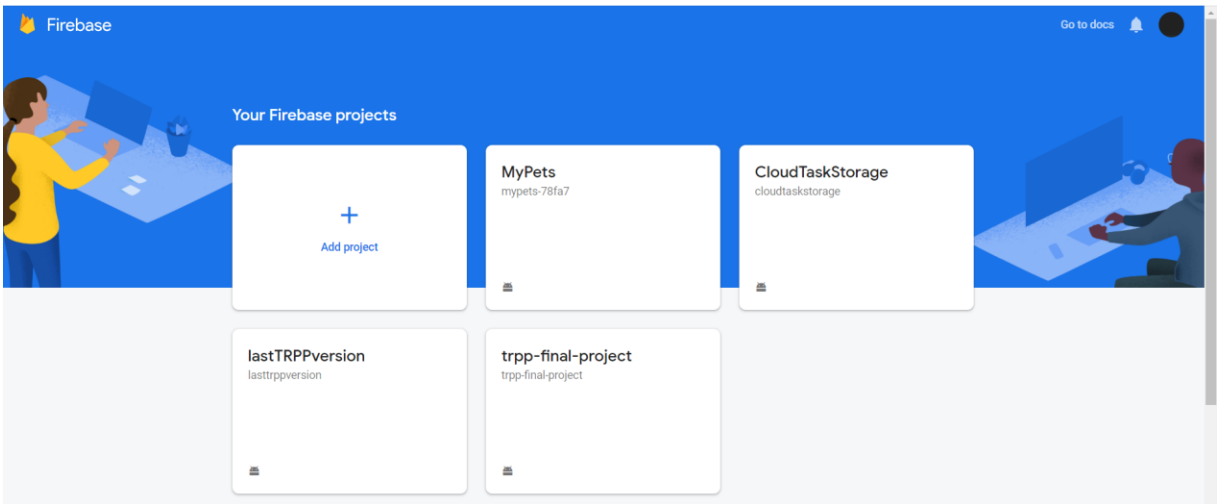

Рисунок 3.8 – Консоль Firebase

Введём имя своего проекта, пропустим страницу со сбором аналитики и нажмём «Создать проект». Firebase автоматически выделит ресурсы для нашего проекта Firebase. Когда процесс завершится, мы попадем на страницу обзора нашего проекта Firebase в консоли Firebase (см. рисунок 3.9).

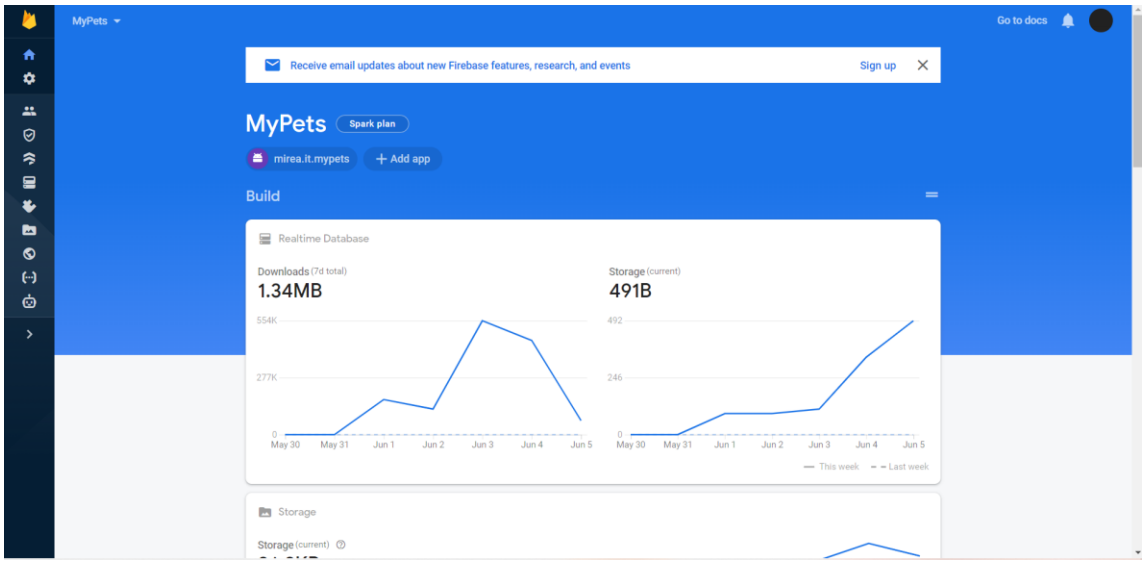

Рисунок 3.9 – Страница обзора проекта

Чтобы использовать Firebase в своем приложении для Android, нам необходимо зарегистрировать свое приложение в проекте Firebase. Для этого необходимо выполнить следующее:

- Перейдём в консоль Firebase.

- В центре страницы обзора проекта щелкнем «Добавить приложение», чтобы запустить рабочий процесс установки.

- Введём имя пакета нашего приложения в поле имени пакета Android.

- Щелкнем: «Зарегистрировать приложение».

Далее необходимо добавить конфигурационный файл Firebase. Для этого:

- Нажмём: «Загрузить google-services.json».

- Переместим файл конфигурации в каталог модуля (уровня приложения) нашего приложения.

Чтобы включить продукты Firebase в приложении, добавим плагин google-services в свои файлы Gradle. Для этого:

- В файле Gradle корневого уровня (уровня проекта) (build.gradle) добавим правила для включения подключаемого модуля Google Services Gradle (см. листинг 4.1).

Листинг 4.1 – Правила для включения Google Services

```
buildscript {

    repositories {
        // Check that you have the following line (if not, add it):
```

```
        google()  // Google's Maven repository
    }

    dependencies {
        // ...

        // Add the following line:
        classpath 'com.google.gms:google-services:4.3.10'  // Google Services plugin
    }
}

allprojects {
    // ...

    repositories {
        // Check that you have the following line (if not, add it):
        google()  // Google's Maven repository
        // ...
    }
}
```

- В файле Gradle нашего модуля (уровня приложения) применим плагин Google Services Gradle (см. листинг 4.2).

Листинг 4.2 – Применение плагина

```
apply plugin: 'com.android.application'
// Add the following line:
apply plugin: 'com.google.gms.google-services'  // Google Services plugin

android {
    // ...
}
```

- Используя Firebase Android BoM , объявим зависимости для продуктов Firebase, которые мы хотим использовать в своем приложении. Объявим их в файле Gradle нашего модуля (на уровне приложения) (обычно app/build.gradle) (см. листинг 4.3).

Листинг 4.3 – Объявление зависимостей

```
dependencies {
    // ...

    // Import the Firebase BoM
    implementation platform('com.google.firebase:firebase-bom:30.1.0')

    // When using the BoM, you don't specify versions in Firebase library
dependencies

    // Declare the dependency for the Firebase SDK for Google Analytics
    implementation 'com.google.firebase:firebase-analytics'

    // Declare the dependencies for any other desired Firebase products
    // For example, declare the dependencies for Firebase Authentication and Cloud
Firestore
    implementation 'com.google.firebase:firebase-auth'
    implementation 'com.google.firebase:firebase-firestore'
```

На этом установка средств разработки Android-приложения завершена.

5. Проектирование компонентов мобильного сервиса.

Для того чтобы начать разрабатывать приложение, его БД и писать сам код необходимо понять и прописать логику взаимодействия модулей и архитектуру этого приложения.

Архитектура приложения.

Так как в приложении будет присутствовать БД, то оно должно соответствовать архитектуре «клиент-сервер» (см. рисунок 5.1), в которой программное обеспечение разделено на клиентскую и серверную части.

Рисунок 5.1 – Архитектура «клиент-сервер»

На клиентской стороне: интерфейс пользователя; основные функции.

На клиенте – взаимодействие с пользователем посредством основных функций и интерфейса, передача его запросов на сервер Firebase, получение ответов и запросов от сервера и представление их в привычном для пользователя виде.

На серверной стороне: Сервис Firebase.

Основные модули и логика приложения.

Учитывая ТЗ, можно сказать, что всего должно быть 3 модуля: модуль интерфейса; модуль сервера; модуль обработки информации.

Каждый модуль реализует законченную функцию.

Модуль интерфейса представляет собой привычную для пользователя визуальную часть, где он вводит и получает необходимую информацию. Он поделен на несколько отдельных экранов (Activity).

Первым отображается экран авторизации (см. рисунок 5.2), где пользователь может войти в приложение, если он уже зарегистрирован в системе или зарегистрироваться. На этом экране представлены следующие элементы:

- Изображение типа ImageView с логотипом приложения

- Текстовое поле для ввода «Имя питомца»

- Текстовое поле для ввода «Пароль»

- Кнопка «Войти»

- Кнопка «Зарегистрироваться»

После нажатия пользователем кнопки «Войти», в случае успешной авторизации он попадает на приветственный экран. Если же пользователь выбирает кнопку «зарегистрироваться» он попадает на экран регистрации (см. рисунок 5.3), где может ознакомиться с краткой информацией о функционале приложения и вариантах его использования. Также на экране регистрации представлена краткая форма, где представлены следующие элементы:

- Переключатели «Тип животного» с выбором «Кот (кошка)» или «Собака»

- Текстовое поле для ввода «Имя питомца»

- Текстовое поле для ввода «Пароль»

После заполнения формы пользователю необходимо нажать кнопку «Зарегистрироваться». В случае успешной регистрации, пользователь, как и при авторизации попадает на приветственный экран (см. рисунок 5.4).

Приветственный экран содержит в себе следующие элементы:

- Изображение типа ImageView с логотипом приложения

- Текстовая надпись: «Привет, », с добавлением загруженного из базы данных имени питомца.

- Текстовая надпись с информацией об авторе, версии и годе выпуска приложения

По прошествии нескольких секунд с момента попадания пользователя на приветственный экран, перед ним автоматически откроется главный экран (см. рисунок 5.5), который выполняет функцию навигатора по приложению. На главном экране представлены следующие элементы:

- Изображение типа ImageView с логотипом приложения

- Текстовая надпись с названием приложения

- Кнопка «Статьи про животных»

- Кнопка «Галерея»

- Кнопка «Выйти из аккаунта»

При нажатии на логотип приложения пользователь попадёт на экран с общей информаций (см. рисунок 5.6), который содержит следующие элементы:

- Изображение типа ImageView с питомцем выбранного пользователем типа

- Текстовая надпись с именем питомца

- Текстовая надпись с выбранным типом питомца

- Текстовая надпись с количеством загруженных фотографий в базу данных

При нажатии на кнопку «Статьи про животных» пользователь попадёт на экран со статьями (см. рисунок 5.7), который содержит в себе элемент RecyclerView, динамически заполняемый обработанными статьями в ходе работы приложения.

При нажатии на кнопку «Галерея» пользователь попадёт на экран галереи (см. рисунок 5.8), который содержит в себе элемент ListView, динамически заполняемый фотографиями, скачанными из базы данных, в ходе работы приложения.

При нажатии на кнопку «Выйти из аккаунта» пользователь попадёт на экран авторизации, при этом в системе произойдёт его выход из аккаунта.

Модуль сервера

Модуль сервера представляет собой класс взаимодействия с сервером Firebase, на который поступают запросы со стороны приложения на обработку информации, введённой пользователем, на выдачу результата из БД, на выдачу результата из облачного хранилища, на регистрацию или авторизацию пользователя. Сервер Firebase обрабатывает информацию и передает ее обратно в приложение.

Модуль обработки информации

На **модуль обработки информации** поступает вся информация в подготовленном виде, такая как:

- Введенные данные пользователя.

- Фотографии пользователя

- Статьи о животных, полученные из интернета.

Модуль обработки информации связывает модуль сервера и модуль интерфейса, реализуя корректное отображение данных с сервера и из интернета на экране пользователя.

6. Реализация компонентов мобильного приложения

6.1 Создание проекта

Для разработки компонентов мобильного сервиса и его взаимодействия с сервером Firebase были использованы следующие языки программирования:

- Java и Kotlin – для обработки информации, работы клиентской части и взаимодействия с сервером Firebase;

Написание клиентской части происходит в программе Android Studio. Для этого создан проект с указанием основного языка программирования Java и минимальной доступной версией Android 4.0.

После создания проекта, программа автоматически сформировала необходимый минимальный набор папок и файлов.

Вся интерфейсная часть и необходимые ресурсы (например, картинки) находятся в папке «res» и ее подпапках. Вычислительная часть помещается в папку java. В папке «manifest» находится манифест приложения, где объявляется общая информация о приложении, необходимая для корректной работы приложения на мобильном устройстве.

6.2 Реализация интерфейса

Макеты экранов (Activity) приложения хранятся в файлах с разрешением xml. В данном приложении необходимо создать десять xml-макетов:

- activity_main – начальный экран приложения, куда попадает пользователь после установки. Данный экран выполняет функцию авторизации пользователя в системе.

- activity_registration – экран, который выполняет функцию регистрации пользователя в системе.

- activity_hello_page – приветственный экран при входе в приложение

- activity_general_layout – главный экран приложения, который выполняет функцию навигатора по приложению.

- activity_pet_account_page – экран с общей информацией о питомце

- activity_advices_page – экран со статьями по уходу за животными

- activity_photo_storage – экран с фотографиями питомца

- list_item – шаблон отображения статьи в элементе RecyclerView

- text_row_item – шаблон отображения фотографии в элементе ListView

Экран авторизации содержит в себе два текстовых поля для ввода имени и пароля, текст с названием приложения, картинку с логотипом приложения, две кнопки для входа и регистрации.

Экран регистрации содержит в себе три текстовых поля, которые кратко сообщают о возможностях приложения, две радиокнопки для выбора типа животного, два текстовых поля для ввода имени животного и пароля и кнопка: «Зарегистрировать питомца».

Приветственный экран содержит одну картинку с логотипом приложения и две текстовых надписи, одна из которых приветствует зарегистрированного питомца, а вторая даёт информацию об авторе приложения, версии и годе выпуска.

Главный экран содержит картинку с логотипом приложения, две кнопки, ведущие на экран со статьями и на экран галереи, текстовая надпись с названием приложения и кнопка: «Выйти из аккаунта».

Экран с общей информацией содержит в себе картинку с типом животного, три текстовых поля с именем и типом животного и с количеством загруженных фотографий в облачное хранилище. Также экран содержит информацию об авторе приложения, версию и год выпуска.

Экран со статьями содержит только элемент RecyclerView, который динамически заполняется шаблонными элементами list_item.

Экран галереи содержит панель инструментов, которая позволяет пользователю выйти на главный экран при нажатии стрелки в верхнем левом углу. Также данный экран содержит элемент ListView динамически заполняемый шаблонными элементами text_row_item

6.3 Реализация модуля обработки информации

Модуль обработки информации содержит 13 основных файлов (см. рисунок 6.3.1):

- AndroidManifest.xml – главный манифест приложения;

- MainActivity.java – описаны действия, благодаря которым осуществляется вход клиента в систему или переход на экран регистрации

- Registration.java - описаны действия, благодаря которым осуществляется регистрация клиента в системе и переход на приветственную страницу

- helloPage.java – содержит функции, которые загружают фотографии из облачного хранилища, чтобы вывести их на экране галереи и которые загружают имя питомца из базы данных, чтобы отобразить его на странице

- advicesPage.kt – содержит функции для динамического отображения данных на экране со статьями

- GeneralLayout.java – используется в качестве основного экрана, связующего все имеющиеся в приложение экраны

- petAccountPage.java – содержит методы для загрузки информации о питомце из базы данных и отображения этой информации на экране

- photoStoarage.java – содержит объявление панели инструментов и вызов фрагмента RecyclerViewFragment.java.

- ImageSaver.java – вспомогательный класс, содержащий методы сохранения фотографий в память устройства и получения этих фотографий из памяти

- RecyclerViewFragment.java – класс, описывающий фрагмент, в котором определены функции для динамического отображения фотографий в галерее

- Utils.kt – файл, содержащий только одну функцию, которая позволяет сделать запрос в сеть и получить данные

- PreferenceClass.java – вспомогательный класс, который необходим для упрощения работы с элементом SharedPreferences.

- pet.java – класс, содержащий строки, описывающие данные о питомце, необходим для отправки информации в базу данных.

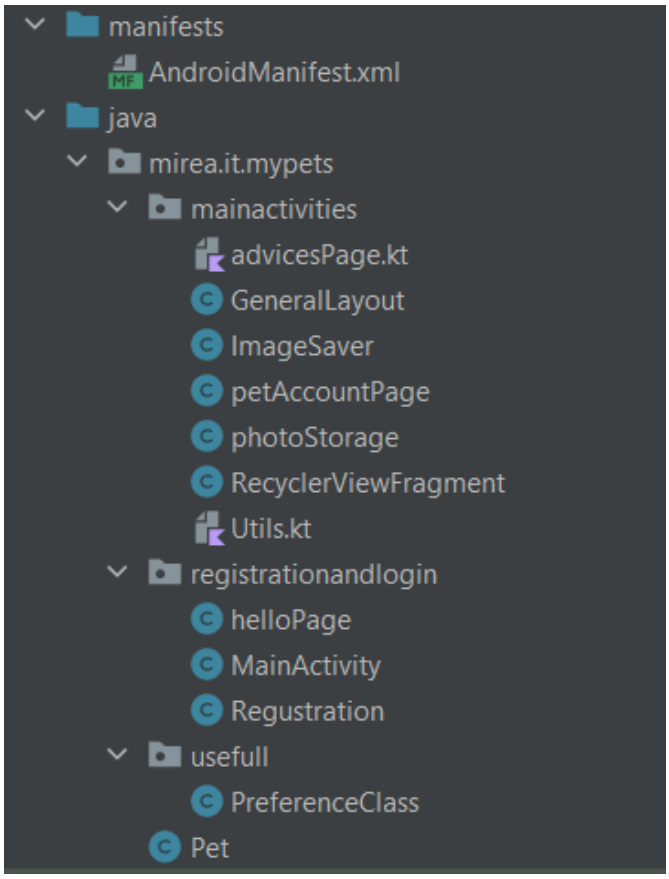

Рисунок 6.3.1 – Файлы модуля обработки данных

Рассмотрим файлы с кодом более подробно.

Файл MainActivity.java содержит следующие основные элементы:

- Переменные кнопок signInButton и signUpButton

- Поля ввода mPetName и mPassword

Файл MainActivity.java содержит следующие основные методы:

- Метод transliteration(String petName), который вызывается, чтобы создать уникальный идентификатор для питомца (см. листинг 6.3.1).

Листинг 6.3.1 – Метод transliteration(String petName)

```
public String transliteration(String petName) {
    String petEmail;
    petEmail = "";
    for (int i = 0; i < petName.length(); i++) {
        petEmail = petEmail + dictionary.get(petName.charAt(i));
    }
    petEmail = petEmail + "@mail.ru";
    return petEmail;
}
```

- Метод onClick(View view), который отслеживает какая кнопка была нажата и реализует логику в соответствии с выбранной кнопкой (см. листинг 6.3.2).

Листинг 6.3.2 – Метод onClick(View view)

```
@Override
public void onClick(View view) {
    if (view.getId() == R.id.signInButton) {
        if ((mPetName.getText()).toString().equals("")) {
            Toast.makeText(MainActivity.this, "Введите имя питомца",
Toast.LENGTH_SHORT).show();
        } else if ((mPassword.getText()).toString().equals("")) {
            Toast.makeText(MainActivity.this, "Введите пароль",
Toast.LENGTH_SHORT).show();
        } else {

SignIn(transliteration(mPetName.getText().toString().toLowerCase(Locale.ROOT)),
mPassword.getText().toString());
        }
    } else if (view.getId() == R.id.signUpButton) {
        Intent intent = new Intent(MainActivity.this, Regustration.class);
        startActivity(intent);
    }
}
```

Методы, связанные со взаимодействием с сервером описаны в пункте 6.4.

Файл Registration.java содержит следующие основные элементы:

- Две радиокнопки mCat и mDog
- Два поля ввода mPetName и mPetPassword
- Кнопка регистрации mButton

Файл Registration.java содержит следующие основные методы:

- onCheckedChanged(RadioGroup radioGroup, int i) – функция, отслеживающая изменения состояния радиокнопок (см. листинг 6.3.3).

Листинг 6.3.3 – onCheckedChanged(RadioGroup radioGroup, int i)

```
radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(RadioGroup radioGroup, int i) {
```

```
        switch (i) {
            case -1:
                Log.d("Reg", "Nothingselected");
                break;
            case R.id.Cat:
                typeOfPet = "Кот (Кошка)";
                break;
            case R.id.Dog:
                typeOfPet = "Собака";
                break;
        }
    }
});
```

- onClick(View view) – метод, отслеживающий нажатие кнопки и проверяющий корректность введённых данных (см. листинг 6.3.4).

Листинг 6.3.4 – Метод onClick(View view)

```
@Override
public void onClick(View view) {
    if (!mCat.isChecked() && !mDog.isChecked()) {
        Toast.makeText(Regustration.this, "Выберите вид животного",
Toast.LENGTH_SHORT).show();
    } else if ((mPetName.getText()).toString().equals("")) {
        Toast.makeText(Regustration.this, "Введите имя питомца",
Toast.LENGTH_SHORT).show();
    } else if ((mPassword.getText()).toString().equals("")) {
        Toast.makeText(Regustration.this, "Введите пароль",
Toast.LENGTH_SHORT).show();
    } else {

SignUp(transliteration(mPetName.getText().toString().toLowerCase(Locale.ROOT)),
mPassword.getText().toString());
    }
}
```

Методы, связанные со взаимодействием с сервером описаны в пункте 6.4.

Файл helloPage.java содержит следующие основные элементы:

- Текстовое поле helloPet.

Файл helloPage.java содержит следующие основные методы:

- stop() – метод, останавливающий работу приложения на 3 секунды и запускающий следующий экран по прошествии этого времени (см. листинг 6.3.5).

Листинг 6.3.5 – Метод stop()

```
private void stop() {
    Thread thread = new Thread() {
        @Override
        public void run() {
```

```
            try {
                TimeUnit.SECONDS.sleep(3);
                Intent intent2 = new Intent(helloPage.this, GeneralLayout.class);
                startActivity(intent2);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    };

    thread.start();
}
```

- updateCountOfImages() – метод, определяющий количество фотографий, сохранённых пользователем в облачном хранилище (см. листинг 6.3.6).

Листинг 6.3.6 – Метод updateCountOfImages()

```
public int updateCountOfImages() {
   countofimages = preferenceClass.getCount(currentUser.getUid());

    Log.d("count", "HelloPage_countofimages: " + Integer.toString(countofimages));

    return countofimages;
}
```

Методы, связанные со взаимодействием с сервером описаны в пункте 6.4.

Файл GeneralLayout.java содержит следующие основные элементы:

- Четыре кнопки galleryButton, advicesButton, accountButton и signOut

Каждая кнопка имеет собственный метод onClick(View view), который переносит пользователя на следующий экран.

Файл petAccountPage.java содержит следующие основные элементы:

- Два текстовых поля petName и petType, в которые выводятся данные полученные с сервера
- Текстовое поле cloudPhotosNumber, в которое выводится количество фотографий, сохранённых в облачном хранилище. Используется вспомогательный класс PreferenceClass.java.

Файл AdvicesPage.kt содержит следующие основные элементы:

- Переменная, содержащая запрос в интернет: request.
- RecyclerView, заполняемый статьями: vRecycleView.
- Класс RecAdapter(val items: ArrayList<FeedItems>), реализующий адаптер для элемента RecyclerView (см. листинг 6.3.7).

Листинг 6.3.7 – Класс RecAdapter(val items: ArrayList<FeedItems>)

```
class RecAdapter(val items: ArrayList<FeedItems>) : RecyclerView.Adapter<RecHolder>()
{
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecHolder {

        val inflater = LayoutInflater.from(parent!!.context)
        val view = inflater.inflate(R.layout.list_item, parent, false)
        return RecHolder(view)
    }

    override fun onBindViewHolder(holder: RecHolder, position: Int) {
        val item = items[position]

        holder?.bind(item)
    }

    override fun getItemCount(): Int {
        return items.size
    }

}
```

- Класс RecHolder(view: View), который необходим для заполнения созданного ранее шаблона полученной информацией (см. листинг 6.3.8).

Листинг 6.3.8 – Класс RecHolder(view: View)

```
class RecHolder(view: View) : RecyclerView.ViewHolder(view) {
    fun bind(item: FeedItems) {
        val vTitle = itemView.findViewById<TextView>(R.id.item_title)
        val vDesc = itemView.findViewById<TextView>(R.id.item_desc)
        val vThumb = itemView.findViewById<ImageView>(R.id.item_thumb)
        vTitle.text = item.title
        vDesc.text = item.description



        try {
            Picasso.get()
                .load(item.thumbnail)
                .resize(100, 100)
                .into(vThumb)
        } catch (e: Exception) {
            Log.e("ImgLoadingError", e.message ?: "null")
            Picasso.get()
                .load(R.drawable.catanddogicon)
                .into(vThumb)
        }

        itemView.setOnClickListener() {
            val i = Intent(Intent.ACTION_VIEW)
            i.data= Uri.parse(item.link)
            vThumb.context.startActivity(i)
        }
    }

}
```

- Класс Feed(val items: ArrayList<FeedItems>) и класс class FeedItems(val title: String, val link: String, val thumbnail: String, val description: String) – необходимые классы для разбора получаемой из интернета новостной ленты.

Файл AdvicesPage.kt содержит следующие основные методы:

- showRecView(feedList: ArrayList<FeedItems>) – метод, который назначает адаптер и менеджер для элемента vRecycleView (см. листинг 6.3.9).

Листинг 6.3.9 – Метод showRecView(feedList: ArrayList<FeedItems>)

```
fun showRecView(feedList: ArrayList<FeedItems>) {
    vRecycleView.adapter = RecAdapter(feedList)
    vRecycleView.layoutManager = LinearLayoutManager(this)
}
```

Файл RecyclerViewFragment.java содержит следующие основные элементы:

- Элемент RecyclerView с именем mRecyclerView.
- Элемент mDataset, хранящий загруженные фотографии.
- Класс CustomAdapter, представляющий собой адаптер, который заполняет галерею фотографиями (см. листинг 6.3.10).

Листинг 6.3.10 – Класс CustomAdapter

```
public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.ViewHolder> {
    private static final String TAG = "CustomAdapter";

    private Drawable[] mDataSet;

    public class ViewHolder extends RecyclerView.ViewHolder {
        private final ImageView button;

        public ViewHolder(View v) {
            super(v);
            button = (ImageView) v.findViewById(R.id.button);
            button.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                    if (getAdapterPosition() == 0)
                        dispatchTakePictureIntent();
                    Log.d(TAG, "Element " + getAdapterPosition() + " clicked.");
                }
            });

        }

        public ImageView getButton() {
            return button;
        }
    }

    public CustomAdapter(Drawable[] dataSet) {
        mDataSet = dataSet;
    }
```

```
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
        // Create a new view.
        View v = LayoutInflater.from(viewGroup.getContext())
                .inflate(R.layout.text_row_item, viewGroup, false);

        return new ViewHolder(v);
    }

    @Override
    public void onBindViewHolder(ViewHolder viewHolder, final int position) {
        Log.d(TAG, "Element " + position + " set.");

        if (position == 0)
        {
            viewHolder.getButton().setMaxWidth(1);
            viewHolder.getButton().setMaxHeight(1);
        }
        viewHolder.getButton().setImageDrawable(mDataSet[position]);
    }

    @Override
    public int getItemCount() {
        return mDataSet.length;
    }


    private void dispatchTakePictureIntent() {
        Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        try {
            startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
        } catch (ActivityNotFoundException e) {
            // display error state to the user
        }
    }

}
```

Файл RecyclerViewFragment.java содержит следующие основные методы:

- initDataset() – метод, заполняющий переменную mDataset фотографиями, которые будут выведены на экран (см. листинг 6.3.11).

Листинг 6.3.11 – Метод initDataset()

```
private void initDataset() {

    Log.d("count", "InitDatasetAgain:" + Integer.toString(countofimages));
    mDataset = new Drawable[countofimages+1];
    mDataset[0] = getResources().getDrawable(R.drawable.plus);


    for (int i = 1; i < countofimages; i++) {
        String photoName = "pet" + Integer.toString(i);
        Bitmap bitmap = new ImageSaver(getContext()).
                setFileName(photoName).
                setDirectoryName("images").
                load();

        Drawable d = new BitmapDrawable(getResources(), bitmap);
```

```
        mDataset[i] = d;
    }
}
```

6.4 Реализация модуля сервера

В качестве серверной части в данной курсовой работе используется сервис Firebase, который предоставляет систему аутентификации, базу данных и облачное хранилище.

Методы, которые используют систему аутентификации находятся в файлах MainActivity.java и Registration.java.

Для работы с системой аутентификации используются следующие элементы:

- Переменная mAuth, которая является экземпляром класса FirebaseAuth, позволяющего работать с Firebase Authentication SDK.

Для работы с системой аутентификации используются следующие методы:

- onStart() - метод проверяет авторизован ли пользователь, если да - повторная авторизация не потребуется (см. листинг 6.4.1).

Листинг 6.4.1 – Метод onStart()

```
/**
 * Метод проверяет авторизован ли пользователь, если да - повторная авторизация не
потребуется.
 */
@Override
public void onStart() {
    super.onStart();
    // Check if user is signed in (non-null) and update UI accordingly.
    FirebaseUser currentUser = mAuth.getCurrentUser();

    if (currentUser != null) {
        currentUser.reload();
    }
    if (currentUser != null) {
        Intent intent = new Intent(MainActivity.this, helloPage.class);
        startActivity(intent);
    }
}
```

- SignIn(String email, String password) - метод, который авторизует пользователя в системе (см. листинг 6.4.2).

```
/**
 * Метод, который авторизует пользователя в системе
 */
public void SignIn(String email, String password) {
    mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(this, new
OnCompleteListener<AuthResult>() {
        /**
         * Метод, который вызывается после регистрации нового пользователя с почтой и
паролем
         * В случае успешной регистрации пользователь попадает на следующую страницу.
```

```
         *
         */
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                Toast.makeText(MainActivity.this, "Авторизация успешна",
Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(MainActivity.this, helloPage.class);
                startActivity(intent);
            } else {
                Toast.makeText(MainActivity.this, "Произошла какая-то ошибка...
Попробуйте ещё раз", Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

- SignUp(String email, String password) – метод, который регистрирует пользователя в системе (см. листинг 6.4.3).

Листинг 6.4.3 – Метод SignUp

```
public void SignUp(String email, String password) {
    mAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(this,
new OnCompleteListener<AuthResult>() {
        /**
         * Метод, который вызывается после регистрации нового пользователя с почтой и
паролем.
         * В случае успешной регистрации пользователь попадает на следующую страницу.
         *
         */
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                Toast.makeText(Regustration.this, "Вы успешно зарегистрировались",
Toast.LENGTH_SHORT).show();
                wrtiteToDataBase(mPetName.getText().toString(), typeOfPet);
                Intent intent = new Intent(Regustration.this, helloPage.class);
                startActivity(intent);
            } else {
                Toast.makeText(Regustration.this, "Произошла какая-то ошибка...
Попробуйте ещё", Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

Методы, которые используют базу данных находятся в файлах helloPage.java, Registration.java и petAccountPage.java.

Для работы с базой данных используются следующие элементы:

- Элемент database типа FirebaseDatabase, который хранит ссылку на экземпляр базы данных

- Элемент myRef типа DatabaseReference, используемый для записи в базу данных

Для работы с базой данных используются следующие методы:

- wrtiteToDataBase(String name, String typeOfPet) – метод, который вызывается после регистрации нового пользователя для сохранения введённых им данных (см. листинг 6.4.4).

Листинг 6.4.4 – Метод - wrtiteToDataBase(String name, String typeOfPet)

```
private void wrtiteToDataBase(String name, String typeOfPet) {
    currentUser = mAuth.getCurrentUser();

    preferenceClass = new PreferenceClass(this);

    preferenceClass.setCount(1, currentUser.getUid());


    Pet pet = new Pet(name, typeOfPet);
    mDatabase.child("pets").child(currentUser.getUid()).setValue(pet);
}
```

- onComplete(@NonNull Task<DataSnapshot> task) – метод, который вызывается после успешного получения данных из базы данных. Отображает полученное из базы данных имя питомца на экране. (см. листинг 6.4.5).

Листинг 6.4.5 – Метод onComplete(@NonNull Task<DataSnapshot> task)

```
@Override
public void onComplete(@NonNull Task<DataSnapshot> task) {
    if (!task.isSuccessful()) {
        Log.e("firebase", "Error getting data", task.getException());
    }
    else {
        pet = task.getResult().getValue(Pet.class);
        if (pet != null) {
            helloPet = (TextView) findViewById(R.id.helloPet);
            helloPet.setText("Привет, \n".concat(pet.name));
        }
        else
        {
            helloPet = (TextView) findViewById(R.id.helloPet);
            helloPet.setText("Привет, \n".concat("Питомец"));
        }
        updateCountOfImages();
        downloadAndSavePhotos();
        stop();
    }
}
```

- onDataChange(DataSnapshot dataSnapshot) – Метод, который вызывается в случае изменения данных в БД, или при первом запуске экрана. Позволяет получить информацию о домашнем питомце и выводит её на экран (см. листинг 6.4.6).

Листинг 6.4.6 – Метод onDataChange(DataSnapshot dataSnapshot)

```
public void onDataChange(DataSnapshot dataSnapshot) {
    // Get Post object and use the values to update the UI
    pet = dataSnapshot.getValue(Pet.class);

    TextView petName = (TextView) findViewById(R.id.petSetAccountPageName);
    TextView petType = (TextView) findViewById(R.id.petSetAccountPageType);

    petName.setText(pet.name);
    petType.setText(pet.type);

    ImageView imageView = (ImageView) findViewById(R.id.accountPhoto);

    if (Objects.equals(pet.type, "Кот (Кошка)"))
    {
        Picasso.get()
                .load(R.drawable.the_red_or_white_cat_i_on_white_studio)
                .resize(100, 150)
                .into(imageView);
    }
    else
    {
        Picasso.get()
                .load(R.drawable.pexels_poodles_doodles_1458926)
                .into(imageView);
    }
    // ..
}
```

Методы, которые используют облачное хранилище находятся в файлах RecyclerViewFragment.java и helloPage.java.

Для работы с облачным хранилищем используются следующие элементы:

- Элемент storage типа FirebaseStorage, который используется для получения доступа к облачному хранилищу

Для работы с облачным хранилищем используются следующие методы:

- downloadAndSavePhotos() – метод, который подгружает фотографии с базы данных для их дальнейшего отображения в галерее (см. листинг 6.4.7).

Листинг 6.4.7 – Метод downloadAndSavePhotos()

```
private void downloadAndSavePhotos() {
    for (int i = 1; i <= countofimages; i++) {
        String photoName = "pet" + Integer.toString(i);
        int position = i;
        storageRef.child(photoName).getBytes(Long.MAX_VALUE).addOnSuccessListener(new
OnSuccessListener<byte[]>() {
            @Override
            public void onSuccess(byte[] bytes) {
                Bitmap bitmap = BitmapFactory.decodeByteArray(bytes, 0,
bytes.length);
                // Use the bytes to display the image
                new ImageSaver(helloPage.this).
                        setFileName(photoName).
```

```
                        setDirectoryName("images").
                        save(bitmap);
                Log.d("dddd", "Фотография скачана и сохранена");
                Log.d("dddd", (helloPage.this).getApplicationInfo().dataDir);
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception exception) {
                Log.d("dddd", "Фотография НЕ скачана и не сохранена");
                // Handle any errors
            }
        });
    }
}
```

- onActivityResult(int requestCode, int resultCode, Intent data) – метод, который получает фотографию, сделанную пользователем. В данном методе полученная фотография загружается в облачное хранилище (см. листинг 6.4.8).

Листинг 6.4.8 – Метод onActivityResult(int requestCode, int resultCode, Intent data)

```
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");

        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        imageBitmap.compress(Bitmap.CompressFormat.JPEG, 100, baos);
        byte[] Data = baos.toByteArray();

        Toast.makeText(getContext(), "Фото загружается в облако...",
Toast.LENGTH_SHORT).show();

        Bitmap bitmap = BitmapFactory.decodeByteArray(Data, 0, Data.length);
        // Use the bytes to display the image
        new ImageSaver(getContext()).
                setFileName(getUniceName()).
                setDirectoryName("images").
                save(bitmap);


        Log.d("count", "OnActivityResult_countofimages: " +
Integer.toString(countofimages));


        UploadTask uploadTask = storageRef.child(getUniceName()).putBytes(Data);
        uploadTask.addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception exception) {
                Log.d("Photo", "Фото НЕ сохранено");
                // Handle unsuccessful uploads
            }
        }).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                // taskSnapshot.getMetadata() contains file metadata such as size,
content-type, etc.
                // ...
                Intent intent2 = new Intent(getContext(), photoStorage.class);
                startActivity(intent2);
```

```
                preferenceClass.setCount(countofimages + 1, currentUser.getUid());

                Log.d("Photo", "Фото сохранено");
            }
        });


    }

}
```

Таким образом, нами описана реализация компонентов мобильного приложения: модуля интерфейса, модуля обработки информации, модуля сервера. Код всех компонентов программы размещён на веб-сервисе GitHub, ссылка на который представлена в приложении 1.

7. Контрольный пример работы

Для проверки корректности работы приложения реализуем контрольный пример.

При открытии приложения пользователь видит экран авторизации (см. рисунок 7.1). На данном экране размещены два поля ввода - для имени питомца и пароля - и две кнопки – для авторизации и регистрации.

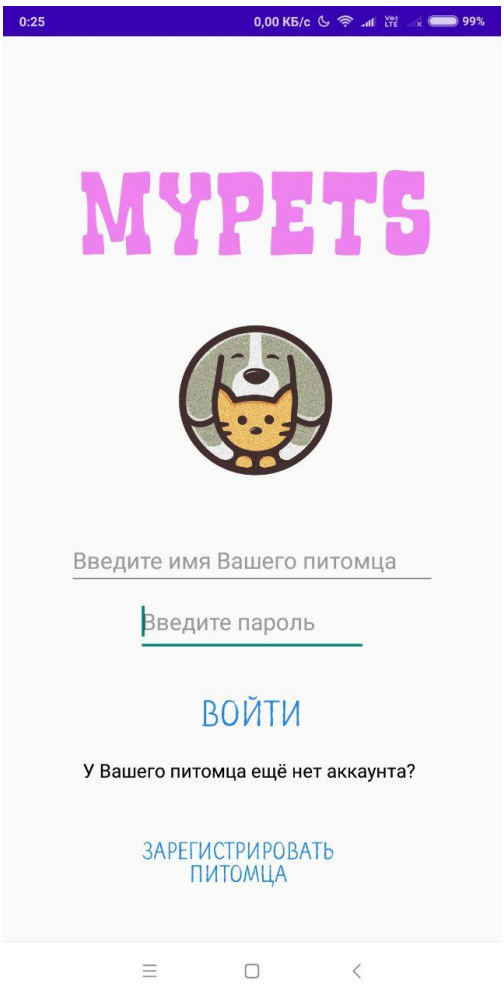

Рисунок 7.1 – Экран авторизации

Рассмотрим работу приложения на примере следующих данных:

- Тип животного: Кот;

- Имя животного: Джек;

- Пароль: 123456;

Если попробовать ввести эти данные на экране авторизации, высветится сообщение об ошибке (см. рисунок 7.2).

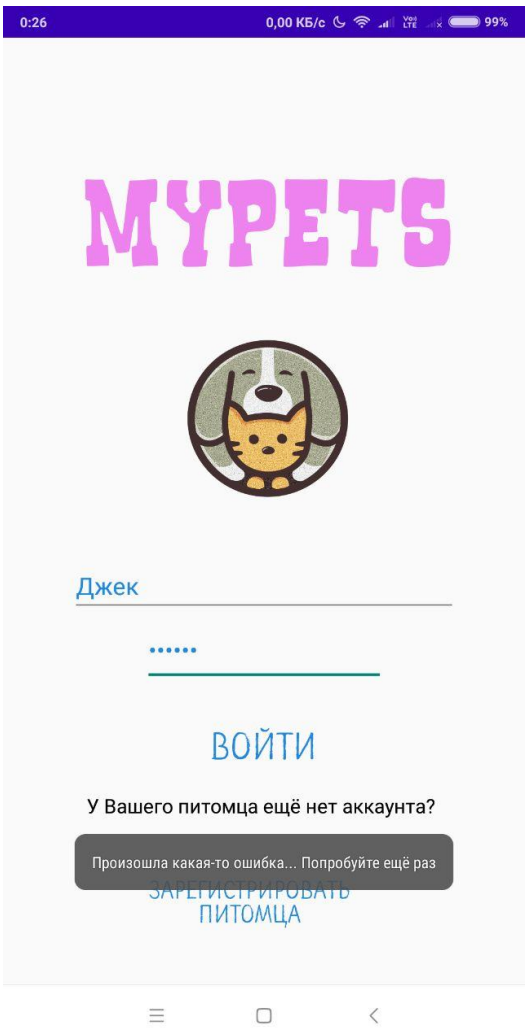

Рисунок 7.2 – Высвечивающаяся ошибка

Питомец ещё не зарегистрирован в системе. Для регистрации перейдём на экран регистрации (см. рисунок 7.3), нажав на соответствующую кнопку.

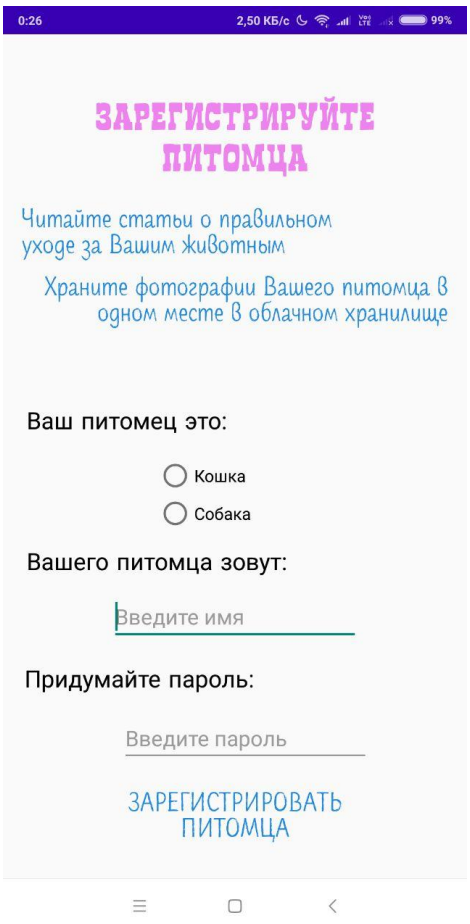

Рисунок 7.3 – Экран регистрации

На данном экране можно увидеть краткую информацию о функционале приложения, а также поля для ввода данных питомца.

Если ввести не все данные, оставив какое-то поле пустым, покажется ошибка (см. рисунок 7.4) с указанием какое поле ввода было пропущено.

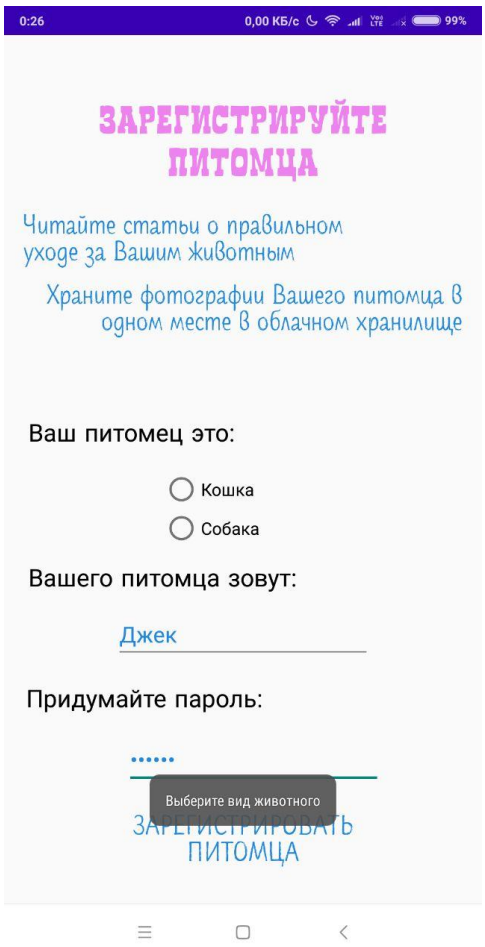

Рисунок 7.4 – Указание пропущенного поля

После ввода всех данных пользователь видит сообщение, что регистрация успешна, после чего попадает на приветственный экран (см. рисунок 7.5).

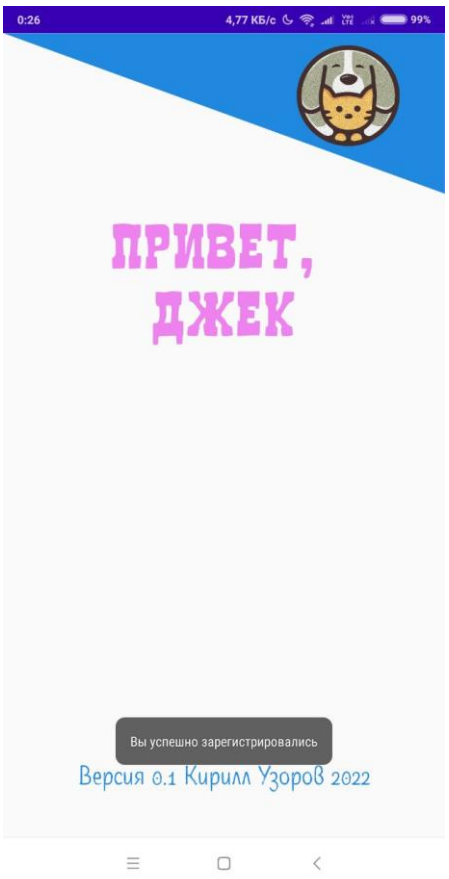

Рисунок 7.5 – Приветственный экран

На приветственном экране отображается имя питомца, который был зарегистрирован нами ранее.

По прошествии нескольких секунд перед нами откроется главный экран приложения (см. рисунок 7.6), на котором расположены кнопки ведущие в аккаунт питомца, на страницу со статьями, в галерею и кнопка выхода из аккаунта. Последовательно зайдём на каждый экран.

Рисунок 7.6 – Главный экран приложения

На экране аккаунта питомца отображена основная информация о питомце, в данном случае, мы видим имя питомца и его тип, которые были указаны нами при регистрации (см. рисунок 7.7). Также можно увидеть количество загруженных фотографий в облако. Так как аккаунт Джека был создан только сейчас, в облаке пока что не хранится его фотографий.

Рисунок 7.7 – Экран с основной информацией о питомце

На экране со статьями можно увидеть статьи с заголовком, кратким описанием и иконкой приложения (см. рисунок 7.8). При нажатии на любую статью пользователю будет предложено выбрать браузер для открытия ссылки (см. рисунок 7.9), после чего статья будет открыта в выбранном браузере (см. рисунок 7.10).

Рисунок 7.8 – Экран со статьями

Рисунок 7.9 – Выбор браузера

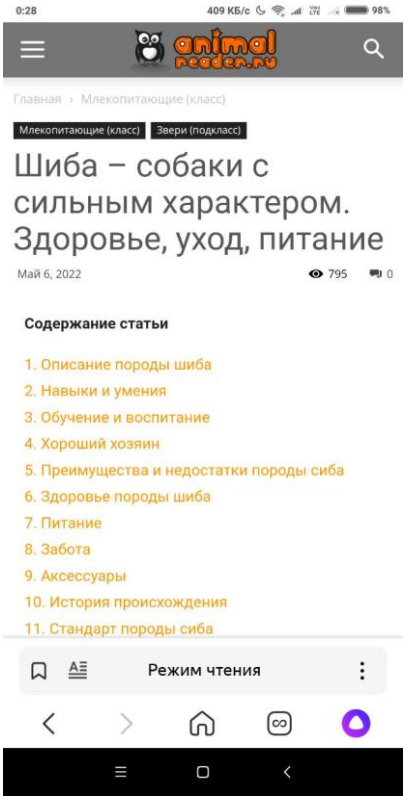

Рисунок 7.10 – Статья в браузере

На экране галереи можно увидеть панель инструментов со стрелочкой в левом верхнем углу, при нажатии на которую пользователь попадает на главную страницу (см. рисунок 7.11). Кроме этого, можно увидеть большую кнопку, на которой изображён знак «плюс». При нажатии на эту кнопку, откроется камера (см. рисунок 7.12). После того как фотография будет сделана пользователь увидит надпись, сообщающую, что фотография загружается на сервер. Через короткий промежуток времени экран обновится и на нём появится сделанное ранее фото (см. рисунок 7.13).

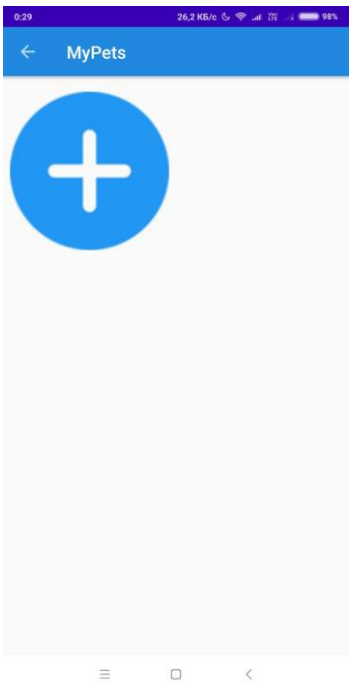

Рисунок 7.11 – Экран с галереей

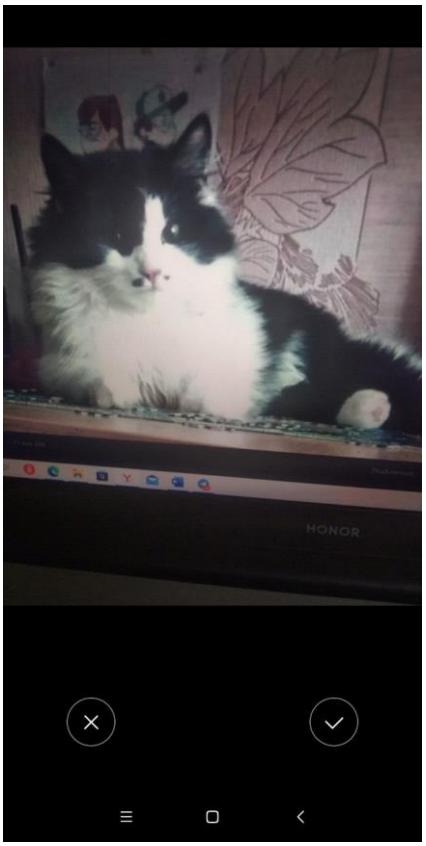

Рисунок 7.12 – Камера

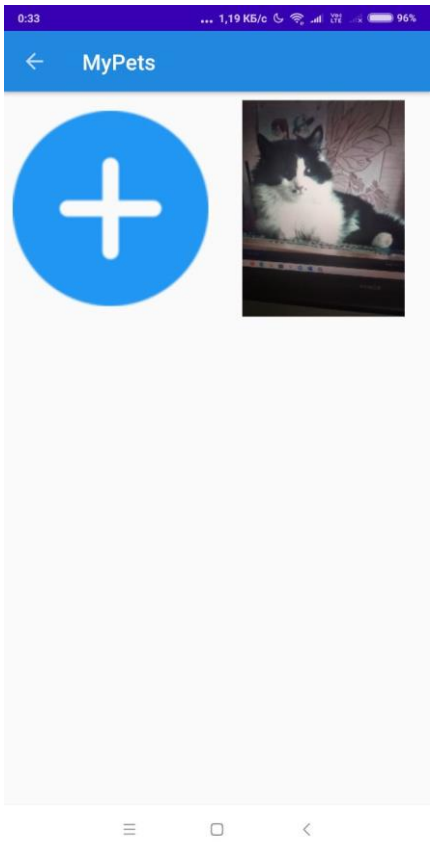

Рисунок 7.13 – Добавленное фото

При нажатии на кнопку выхода из аккаунта, пользователь попадёт на страницу авторизации. При перезапуске приложения автоматический вход в приложение больше не будет работать, пока пользователь заново не выполнит вход. Выполним

вход для Джека. Введём его имя и пароль и нажмём кнопку: «Войти». Увидим надпись об успешной авторизации и сразу же приветственный экран, ведущий на главную страницу приложения (см. рисунок 7.14).

Рисунок 7.14 – Авторизация в приложении

Таким образом, приложение продемонстрировало работоспособность и корректность результатов поставленных перед началом разработки приложения задач.

ЗАКЛЮЧЕНИЕ

В курсовой работе спроектирован и разработан мобильный сервис, способный загружать ленту статей про животных и показывать её в удобочитаемом виде на экране пользователя. Помимо этого, мобильный сервис позволяет пользователю делать фотографии и сохранять их в облаке.

Был приведён процесс установки средств, при помощи которых велась разработка мобильного сервиса. Такими средствами являются AndroidStudio и сервис Firebase.

В рамках курсовой работы были спроектированы, реализованы и протестированы компоненты мобильного приложения при помощи инструментов разработки AndroidStudio, сервиса Firebase, языков программирования Java и Kotlin.

Так как все поставленные задачи решены, то цель курсовой работы достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Android Studio and SDK tools // [Электронный ресурс] URL: https://developer.android.com/studio (дата обращения 15.05.22)
2. Firebase Documentation // [Электронный ресурс] URL: https://firebase.google.com/docs?authuser=0&hl=en (дата обращения 15.05.22)
3. Firebase Console // [Электронный ресурс] URL: https://console.firebase.google.com/ (дата обращения 15.05.22)
4. Курс «Разработка мобильных приложений для Android» // [Электронный ресурс] URL: https://stepik.org/course/5703/syllabus
5. Documentation for app developers // [Электронный ресурс] URL: https://developer.android.com/docs (дата обращения 15.05.22)
6. Сайт Александра Климова // [Электронный ресурс] URL: http://developer.alexanderklimov.ru/android/ (дата обращения 15.05.22)

ПРИЛОЖЕНИЯ

Приложение 1. Ссылка на GitHub проекта.

https://github.com/uzorov/MyPets