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

---

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

# КУРСОВАЯ РАБОТА

по дисциплине «Проектирование и разработка мобильных приложений»

**Тема курсовой работы:**
«Информационно-справочная система " Поликлиника"»

Студент группы ИКБО-60-23 Ананбех Милана Имадовна _____ (подпись)

Руководитель курсовой работы Доцент Синицын И.В. _____ (подпись)

Работа представлена к защите «9» 06 2025 г.

Допущен к защите «9» 06 2025 г.

Москва 2025 г.

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

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

Утверждаю
И.О.Заведующего кафедрой МОСИТ
_____ Головин С.А.
«17» февраля 2025 г.

# ЗАДАНИЕ

**на выполнение курсовой работы**

по дисциплине «Проектирование и разработка мобильных приложений»

Студент Ананбех Милана Имадовна Группа ИКБО-60-23

**Тема** «Информационно-справочная система " Поликлиника"»

**Исходные данные: Разрабатываемый прототип мобильного приложения должен** предоставлять возможности полностью интерактивной системы с понятным дружественным UI 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 мобильной ОС с применением виртуальных сред. Установка и настройка IDE для мобильной разработки. Установка и настройка эмуляторов мобильного девайса.
Анализ предметной области разрабатываемой программной системы, сбор и анализ требований, составление ТЗ согласно ГОСТ 19.201-78. Реализация ролевой модели безопасности.
Изучение жизненного цикла мобильных программ и их компонентов, а также создание мобильного программного комплекса с применением языков программирования высокого уровня согласно теме курсовой работы. Реализация в создаваемом программном комплексе визуальных элементов, сервисов, методов хранения данных.
Тестирование и отладка кода созданного программного продукта, контрольные примеры работы.
Возможно портирование написанной программной системы на внешних хостах в сети Интернет.
Отчет по курсовой работе в виде пояснительной записки.

**Срок представления к защите курсовой работы:** **до** «30» мая 2025 г.

**Задание на курсовую работу выдал** _____ Синицын И.В.
*Подпись руководителя* *(ФИО руководителя)*
«17» февраля 2025 г.

**Задание на курсовую работу получил** _____ Ананбех М.И
*Подпись обучающегося* *(ФИО обучающегося)*
«17» февраля 2025 г.

# Оглавление

# ВВЕДЕНИЕ

## Цель

Цель разработки информационно-справочной системы "Поликлиника" заключается в создании удобного и интуитивно понятного инструмента, который позволит пользователям получать актуальную информацию о врачах и их специализациях, а также осуществлять поиск врачей по различным критериям. Система будет предоставлять возможность просмотра медицинских карт пользователей, включая диагнозы и назначения, а также записываться на приём к врачу в удобное для них время.

## Задачи

Для достижения поставленной цели необходимо решить следующий комплекс задач:

1) Анализ требований и планирование:

- Провести анализ требований пользователей к программному комплексу.
- Разработать план и архитектуру приложения.

2) Проектирование пользовательского интерфейса:

- Создать удобный и интуитивно понятный интерфейс для пользователей.

3) Разработка функционала для пользователей:

- Реализовать функции поиска и просмотра информации о врачах (ФИО, образование, специальность и т.д.).
- Реализовать возможность создавать и редактировать свой профиль.
- Реализовать просмотр личной карточки пациента ( диагноз, назначения и т.д.).

4) Интеграция и тестирование:

- Провести тестирование приложения для выявления и устранения ошибок.

# РАЗДЕЛ 1: ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

Для разработки мобильного приложения Информационно-справочная система "Поликлиника" было проведено исследование текущих технологий и инструментов, используемых в индустрии разработки мобильных приложений. Особое внимание уделялось выбору языка программирования, базы данных и библиотек, которые обеспечат функциональность и надежность приложения.

## 1.1 Объект исследования

Объектом исследования выступает информационно-справочная система "Поликлиника", построенная на платформе Firebase и ориентированная на хранение и оперативную подачу справочной информации о работе медицинского учреждения. Система объединяет в себе пользовательский интерфейс и облачную архитектуру с использованием Firestore.

## 1.2 Предметная область исследования

Предметная область охватывает проектирование и разработку информационно-справочных систем в сфере здравоохранения с использованием облачных технологий. В рамках данной работы рассматриваются ключевые аспекты создания системы "Поликлиника" с применением Firebase Firestore — облачной NoSQL базы данных от Google, обеспечивающей масштабируемость, надежность и доступ в реальном времени.

Основные исследуемые направления включают:

1. Пользовательский интерфейс (UI):

- Разработка удобного и доступного интерфейса, ориентированного как на пациентов, так и на сотрудников поликлиники;
- Адаптация интерфейса под различные типы устройств — от смартфонов до планшетов и ПК.

2. Работа с Firebase Firestore:

- Интеграция облачной базы данных Firestore для хранения и управления информацией о врачах, расписании приёма и медицинских услугах;
- Использование преимуществ Firestore, таких как:

- Реальное время (real-time updates) — изменения в базе данных мгновенно отражаются на клиенте без перезагрузки;
- Гибкая структура данных — возможность хранения вложенных коллекций и документов с произвольной структурой;
- Масштабируемость — система автоматически адаптируется под количество пользователей.

3. Функциональные возможности системы:

- Отображение данных о расписании врачей, кабинетах, специализациях, а также доступных услугах;
- Возможность поиска по различным критериям: имя врача, специализация, день недели и т.д.

4. Безопасность и разграничение прав доступа:

- Реализация контроля доступа с помощью Firebase Authentication и Firestore Security Rules;
- Ограничение прав пользователей: пациенты имеют только доступ к чтению данных;
- Обеспечение безопасности персональных данных в соответствии с требованиями по защите информации.

### 1.3 Анализ существующих аналогов

При создании системы "Поликлиника" целесообразно изучить существующие решения, использующие облачные технологии и предоставляющие пользователям справочную информацию в сфере медицины:

1) Госуслуги / ЕМИАС:

- Предоставляют возможность записи к врачу, просмотра расписания и информации о медучреждениях;
- Имеют веб- и мобильные интерфейсы.

2) Поликлиника.ру:

- Платформы с возможностью поиска врачей, услуг и адресов клиник;

- Часто имеют карту и фильтры по специализациям.

3) ProDoctorov:

- Система отзывов и рейтингов врачей, предоставляющая расширенную информацию о приёмах, специализациях и ценах.

Сравнение показывает, что система, построенная на Firestore, имеет преимущества за счёт высокой отзывчивости, автоматической синхронизации и встроенной системы авторизации.

### 1.4 Актуальность

С каждым годом увеличивается поток пациентов, а вместе с ним — и объёмы информации, которую требуется быстро и точно предоставить. Информационная перегрузка и недостаток централизованных решений в учреждениях здравоохранения порождают необходимость внедрения удобных цифровых систем. Ключевые причины, подтверждающие актуальность создания системы "Поликлиника":

1) Рост цифровизации медицины:

- Всё больше пользователей предпочитают получать медицинскую информацию в электронном виде, без необходимости личного визита в регистратуру.

2) Сложности в навигации и поиске информации:

- Отсутствие единой справочной платформы затрудняет доступ к расписанию врачей, перечню услуг, сведениям о квалификации специалистов.

3) Необходимость автоматизации:

- Снижение нагрузки на регистраторов и операторов, автоматизация рутинных процессов повышает эффективность работы учреждения.

4) Увеличение потребности в персонализации:

- Пациенты хотят видеть список "своих" врачей, иметь доступ к истории посещений, получать напоминания — это требует гибкой и умной системы.

## 1.5. Выбор инструментальных средств моделирования

В ходе анализа инструментальных средств для моделирования приложения, среди которых были такие варианты как: Microsoft Visual Studio, InteliJ IDEA, Qt Creator, B4X, PhoneGap была выбрана среда разработки мобильных приложений Android studio. Именно эта среда отвечает всем современным требованиям и отлично подходит для разработки программного продукта для системы android на языке Java.

# РАЗДЕЛ 2: ПРОЕКТНАЯ ЧАСТЬ

## 2.1 Определение пользовательских и функциональных требований

1. Регистрация и авторизация пользователей
   - Пользователь может зарегистрироваться с помощью email и пароля;
   - Используется Firebase Authentication для защиты доступа.
2. Поиск информации о врачах
   - Пользователь может выполнить поиск по ФИО, специальности или отделению;
   - Результаты отображаются в виде карточек с:
     - ФИО врача;
     - Специальностью;
     - Образованием и стажем.
   - Данные загружаются из Firestore и кэшируются для оффлайн-доступа.
3. Редактирование профиля пользователя
   - Каждый пользователь имеет доступ к своему профилю;
   - Доступные поля:
     - ФИО;
     - Дата рождения;
     - Адрес;
     - Номер страхового полиса;
     - Контактные данные (телефон, email).
   - Изменения автоматически синхронизируются с Firestore и сохраняются в кэш.
4. Просмотр личной медицинской карточки пациента
   - Доступна только зарегистрированному пользователю;
   - Карточка содержит:
     - Диагнозы;
     - Историю приёмов;
     - Назначения и обследования;

- Имя лечащего врача.

- Все данные защищены и доступны только владельцу (через Firestore security rules).

5. Синхронизация и работа с данными

- Все данные хранятся в Firebase Firestore, структура:
  - users/ – личные профили;
  - doctors/ – список врачей;
  - diagnoses/ – диагнозы пациентов.
- Поддержка оффлайн-доступа (Firestore сохраняет данные локально);
- Реальное обновление данных (при наличии интернета).

## 2.2 Описание используемых средств разработки

Для разработки мобильного приложения «Информационно-справочная система «Поликлиника» » были выбраны следующие технологии:

Язык программирования: Java

Java была выбрана как основной язык разработки благодаря своей зрелости и широкому распространению в экосистеме Android.

Преимущества:

- Широкая поддержка и стабильность. Java уже давно используется для создания Android-приложений, что обеспечивает богатый выбор библиотек, фреймворков и документации;
- Кроссплатформенность. Приложения, написанные на Java, могут запускаться на различных платформах с поддержкой JVM, что делает код более универсальным.

Недостатки:

- Избыточный синтаксис. По сравнению с Kotlin, Java требует больше строк кода для решения типовых задач, что делает разработку менее лаконичной;
- Уступает в современных возможностях. Java не предоставляет некоторых полезных функций, таких как встроенная защита от null, корутин или расширенных лямбда-выражений, что ограничивает гибкость разработки.

База данных: Firebase Firestore

В качестве системы хранения данных была выбрана облачная база данных Firebase Firestore, оптимально подходящая для приложений, работающих с динамической информацией.

Преимущества:

- Обновления в реальном времени. Firestore поддерживает мгновенную синхронизацию данных между клиентами, что особенно важно для актуальной медицинской информации;
- Облачная инфраструктура. Нет необходимости в настройке и обслуживании собственных серверов — все backend-операции берёт на себя Firebase;
- Гибкое масштабирование. Решение хорошо подходит как для небольших клиник, так и для масштабных сетей с большим количеством пользователей.

Недостатки:

- Стоимость. При росте числа пользователей и объема данных использование Firestore может стать затратным;
- Ограниченные возможности запросов. По сравнению с классическими реляционными базами, возможности сложных выборок и соединений в Firestore менее гибкие;
- Настройка безопасности. Требуется детальная проработка правил доступа к данным, чтобы обеспечить конфиденциальность и безопасность медицинской информации.

## 2.3 Архитектура программной системы

1. Общая структура

Приложение разработано на языке Java с использованием архитектурного паттерна MVVM (Model–View–ViewModel), что способствует разделению логики представления, бизнес-логики и модели данных. Проект организован в виде модулей и пакетов в соответствии с назначением компонентов.

2. Слой данных (data)

Пакет: com.example.healthcareapp.data

Этот слой отвечает за работу с данными: получением, хранением и передачей в приложение.

adapters:

– DoctorAdapter — адаптер для отображения списка врачей в RecyclerView;

– AdapterCallBack — интерфейс для обработки кликов/действий пользователя в списках.

model:

– Doctor — модель данных врача (ФИО, образование, специальность и т.д.).

3. Слой бизнес-логики (domain)

Пакет: com.example.healthcareapp.domain

На данный момент он пустой, но предназначен для размещения:

- Интерфейсов репозиториев;
- Сервисов;
- Бизнес-логики, обрабатывающей данные между data и ViewModels.

4. Слой представления (ui)

Пакет: com.example.healthcareapp.ui

Содержит фрагменты и интерфейсы пользователя, разделённые по подсекциям:

auth:

– LoginFragment — вход в систему;

– SignUpFragment — регистрация нового пользователя.

doctors:

– DoctorsFragment — список всех врачей;

– DetailFragment — детальная информация о враче;

– DiagnosisFragment — диагноз пациента;

– NaznachFragment — назначения;

– MedCardFragment — личная медицинская карточка пациента;

– EditProfileFragment — редактирование пользовательского профиля;

– ProfileFragment — просмотр профиля пациента;

– AnalysysFragment – анализы.

main:

– MainActivity — основная активность, в которой хостятся фрагменты и навигация.

5. ViewModel слой (ViewModels)

Пакет: com.example.healthcareapp.ViewModels

– DoctorViewModel — содержит бизнес-логику, связанную с врачами;

– ProfileViewModel — управляет состоянием и логикой профиля пользователя и его карточки.

6. Взаимодействие между слоями

[ View (Fragment) ]

↓

[ ViewModel ]

↓

[ Repository / Firestore ]

↑

[ Model (Doctor, Profile...) ]

7. Безопасность

- Аутентификация через Firebase Authentication;
- Доступ к данным регулируется через Firebase Security Rules;

- Разделение по ролям реализуется на уровне данных и правил доступа.

8. Расширяемость

Благодаря использованию MVVM и разделению по пакетам, приложение легко расширяется:

- Можно добавить модули для чата, записи к врачу, интеграции с другими сервисами.
- Легко тестировать бизнес-логику отдельно от UI.

## 2.4 Тестирование и верификация приложения

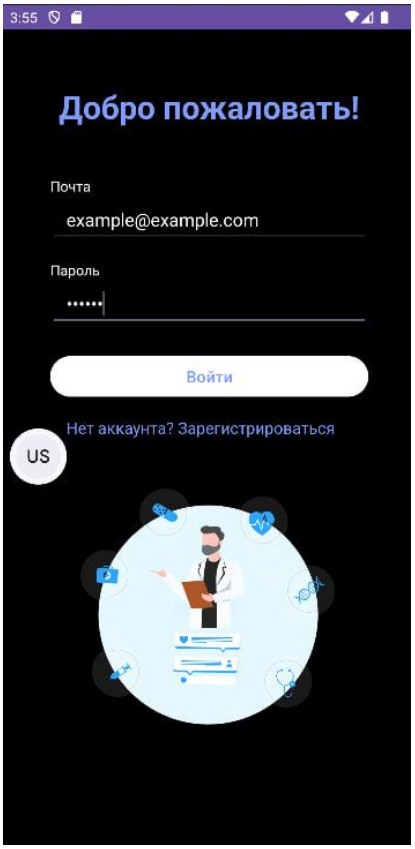


Рисунок 1 – Авторизация пользователя

Для успешной авторизации пользователя уже зарегистрировавшийся человек должен ввести почту и пароль. Форма регистрации указана ниже (рис.2).

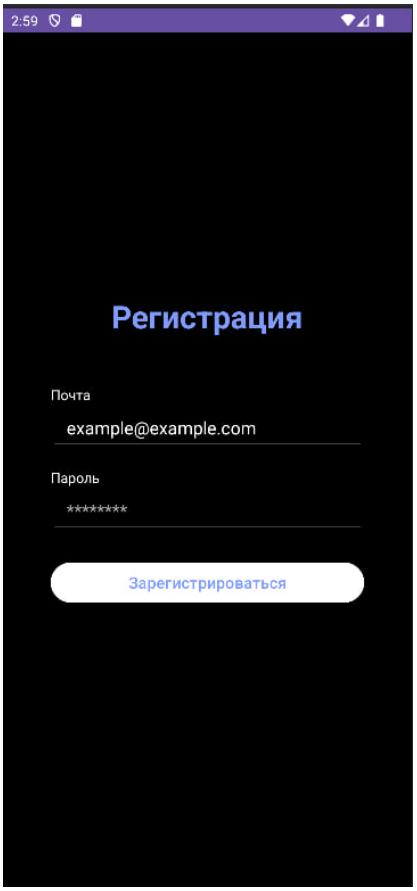


Рисунок 2 – Регистрация пользователя

После успешной авторизации пользователь видит уведомление (рис.3).

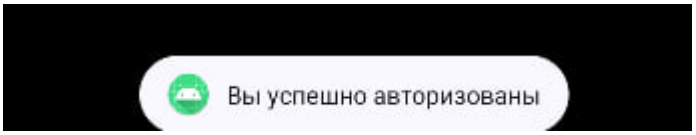


Рисунок 3 – Уведомление об успешной авторизации

Пользователь может искать нужного врача по ФИО или специальности (рис.5).

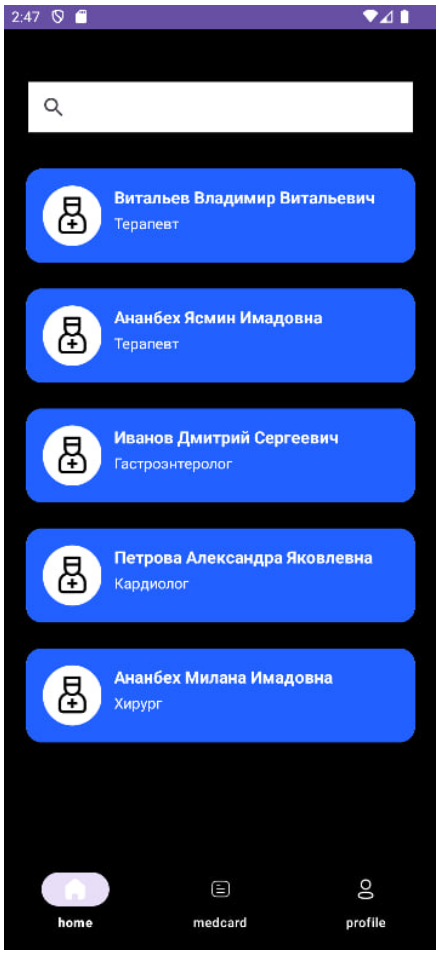


Рисунок 4 – Список врачей


Рисунок 5 – Поиск врача

Также реализована возможность просмотра информации о враче, а также запись на прием (рис.6).

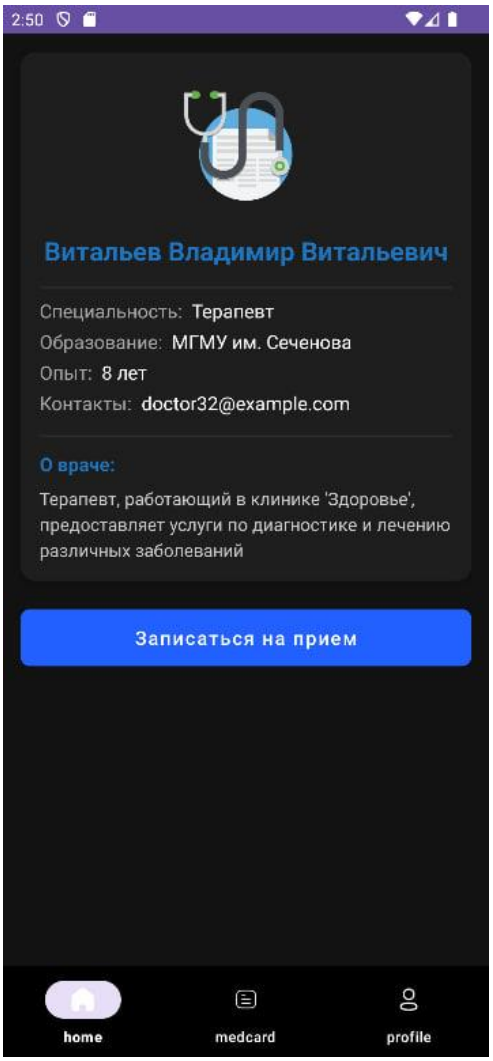


Рисунок 6 – Информация о враче

Также доступен просмотр медкарты пациента и информация о диагнозах, назначениях и анализах( рис.7,8,9,10)

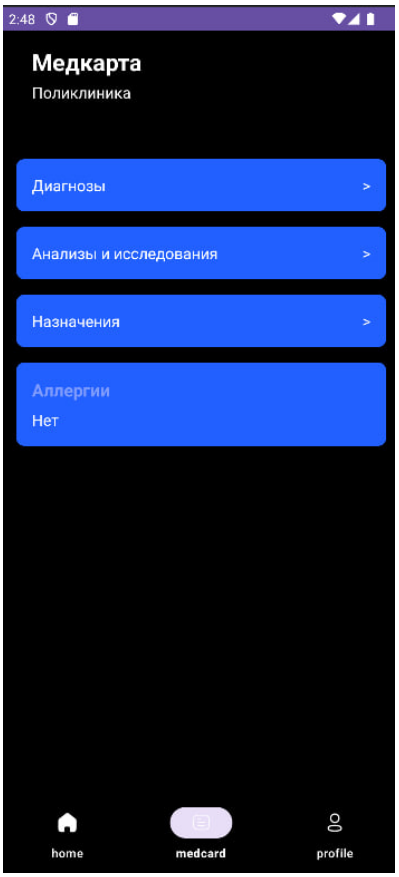


Рисунок 7 – Медкарта пациента

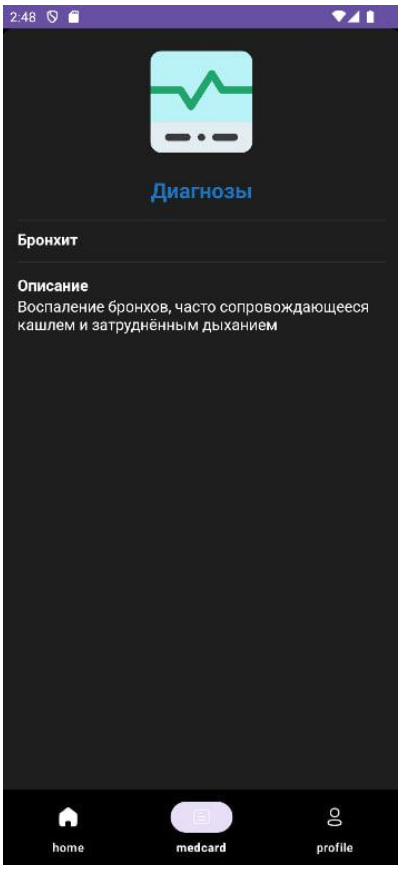


Рисунок 8 – Диагнозы пациента

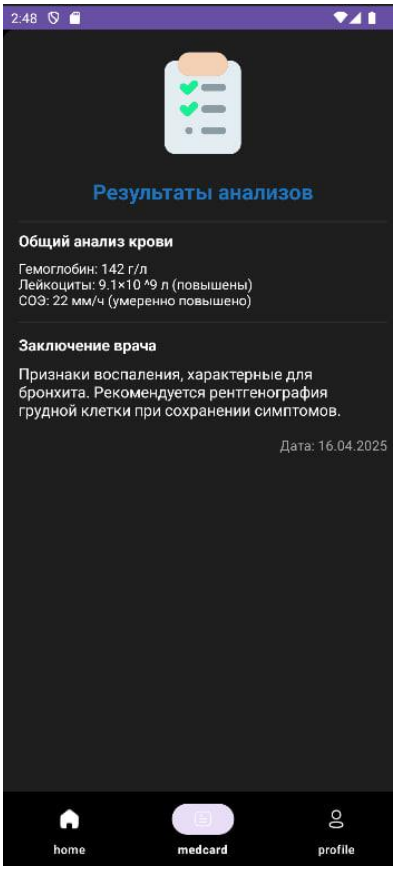


Рисунок 9 – Результаты анализов

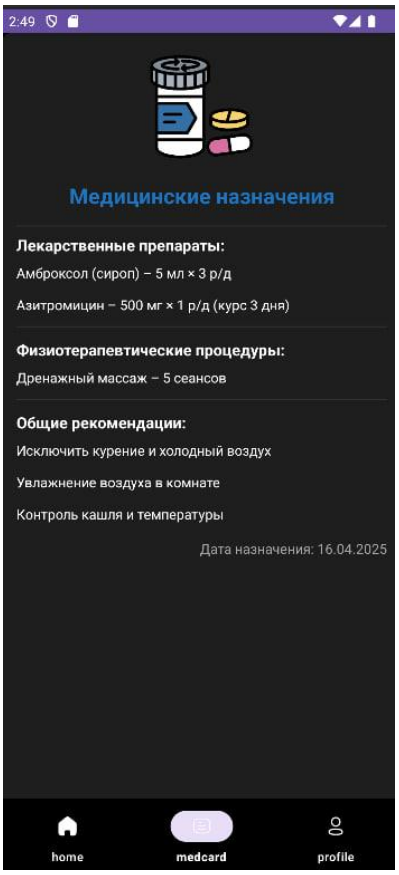


Рисунок 10 – Назначения врача

Редактирование профиля пациента (рис.11,12). Есть возможность выбора аватара изходя из пола пользователя.

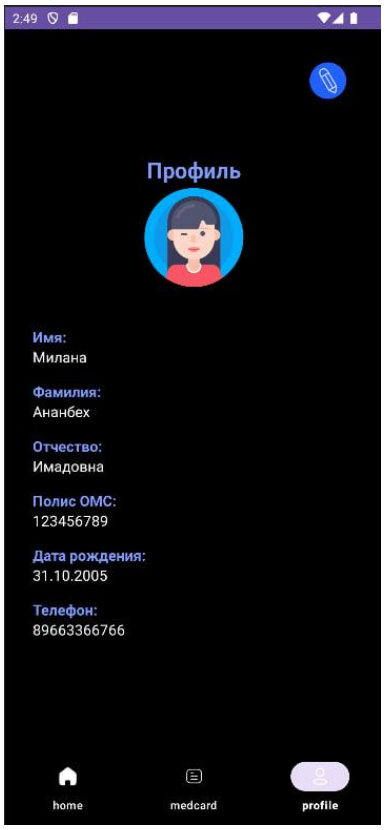


Рисунок 11 – Профиль пациента ж

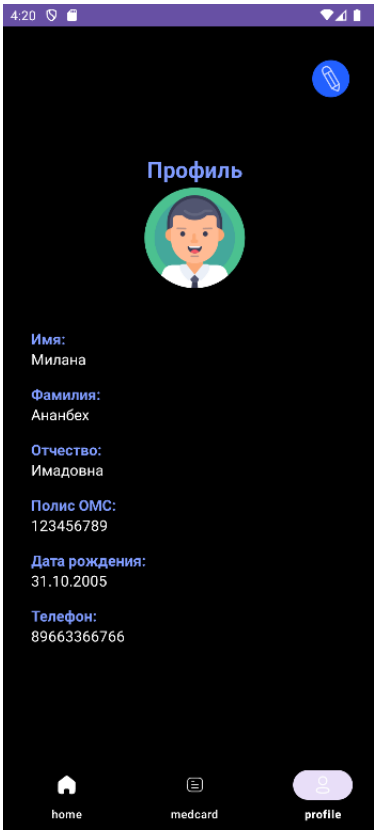


Рисунок 12 – Профиль пациента м

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была разработана информационно-справочная система «Поликлиника» в виде мобильного приложения, реализующая ключевые функции взаимодействия пользователя с медицинской информацией. Основной целью являлось создание удобного и функционального инструмента, позволяющего пациентам получать актуальные сведения о врачах, медицинских картах, диагнозах и назначениях, а также управлять личным профилем.

В процессе реализации проекта были решены следующие задачи:

- Проведён анализ требований к системе и спроектирована архитектура приложения с использованием шаблонов проектирования и разделением на слои (UI, data, domain);
- Разработан удобный пользовательский интерфейс, обеспечивающий простую навигацию по разделам: авторизация, профиль, карточка пациента, назначение, диагностика и т.д.;
- Реализованы основные функции: регистрация и вход в систему, просмотр информации о врачах, просмотр и редактирование профиля пользователя, доступ к медицинской карте и диагнозам;
- В качестве базы данных использован облачный сервис Firebase Firestore, что обеспечило надёжное хранение и масштабируемость данных;
- Для обработки данных и логики взаимодействия с интерфейсом использована архитектура MVVM с внедрением ViewModel.

Разработанное приложение обеспечивает доступность медицинской информации в удобной форме, повышая информированность пациента и эффективность взаимодействия с медицинским учреждением. Таким образом, можно заключить, что поставленные задачи успешно выполнены, а цель курсовой работы достигнута.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Синицын И.В. Методические указания по выполнению практических работ по дисциплине «Разработка мобильных приложений», 2025.

2. Синицын И.В. Лекционные материалы по дисциплине «Разработка мобильных приложений», 2025.

3. Android Developers [Электронный ресурс]. URL: https://developer.android.com/ (дата обращения 26.04.2024)

# ПРИЛОЖЕНИЕ

## Приложение 1

*Листинг файла LoginFragment.java*

```
package com.example.healthcareapp.ui.auth;

import static android.content.ContentValues.TAG;

import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;

import com.example.healthcareapp.R;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Random;

public class LoginFragment extends Fragment {

    private Button login;
    private TextView newAccText;
    private EditText emailEditText, passwordEditText;
    private FirebaseAuth auth;
    private FirebaseFirestore db;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_login, container,
false);
    }
```

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

        emailEditText = view.findViewById(R.id.email);
        passwordEditText = view.findViewById(R.id.password);
        login = view.findViewById(R.id.login);
        newAccText = view.findViewById(R.id.register_text);

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

        NavController navController = Navigation.findNavController(view);

        login.setOnClickListener(v -> handleLogin(navController));
        newAccText.setOnClickListener(v ->
navController.navigate(R.id.action_loginFragment_to_signupFragment));
    }

    private void handleLogin(NavController navController) {
        String email = emailEditText.getText().toString().trim();
        String password = passwordEditText.getText().toString().trim();

        if (email.isEmpty() || password.isEmpty()) {
            Toast.makeText(getContext(), "Поля не могут быть пустыми",
Toast.LENGTH_SHORT).show();
            return;
        }

        auth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(requireActivity(), task -> {
                    if (task.isSuccessful()) {
                        Toast.makeText(getContext(), "Вы успешно
авторизованы", Toast.LENGTH_SHORT).show();
                        FirebaseUser firebaseUser =
auth.getCurrentUser();
                        if (firebaseUser != null) {
                            saveUserToFirestore(firebaseUser, email,
navController);
                        }
                    } else {
                        Toast.makeText(getContext(), "Ошибка авторизации:
" + task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                    }
                });
    }

    private void saveUserToFirestore(FirebaseUser firebaseUser, String
email, NavController navController) {
        String uid = firebaseUser.getUid();

        DocumentReference userDocRef =
db.collection("users").document(uid);

        userDocRef.get().addOnSuccessListener(documentSnapshot -> {
            if (!documentSnapshot.exists()) {
                Map<String, Object> user = new HashMap<>();
```

```
                user.put("email", email);
                user.put("createdAt", System.currentTimeMillis());

                userDocRef.set(user)
                        .addOnSuccessListener(unused ->
assignRandomDiagnosis(uid))
                        .addOnFailureListener(e -> Log.w(TAG, "Ошибка при
создании пользователя", e));
            } else {
                Log.d(TAG, "Пользователь уже существует, пропускаем
создание.");
            }

            navController.navigate(R.id.action_login_to_doctors);
        }).addOnFailureListener(e -> Log.e(TAG, "Ошибка при проверке
существования пользователя", e));
    }

    private void assignRandomDiagnosis(String uid) {
        DocumentReference userRef = db.collection("users").document(uid);
        userRef.get().addOnSuccessListener(snapshot -> {
            if (snapshot.exists() && !snapshot.contains("diagnosis")) {


db.collection("diagnoses").get().addOnSuccessListener(task -> {
                    List<DocumentSnapshot> diagnosisList =
task.getDocuments();
                    if (!diagnosisList.isEmpty()) {
                        DocumentSnapshot randomDoc =
diagnosisList.get(new Random().nextInt(diagnosisList.size()));


                        String name = randomDoc.getString("title");
                        String description =
randomDoc.getString("description");
                        String drugs = randomDoc.getString("drugs");
                        String physio = randomDoc.getString("physio");
                        String recommendations =
randomDoc.getString("recommendations");
                        String results = randomDoc.getString("results");
                        String docResult =
randomDoc.getString("doc_result");
                        String date = randomDoc.getString("date");


                        Map<String, Object> diagnosisMap = new
HashMap<>();
                        diagnosisMap.put("title", name);
                        diagnosisMap.put("description", description);
                        diagnosisMap.put("drugs", drugs != null ? drugs :
"");
                        diagnosisMap.put("physio", physio != null ?
physio : "");
                        diagnosisMap.put("recommendations",
recommendations != null ? recommendations : "");
                        diagnosisMap.put("results", results != null ?
results : "");
                        diagnosisMap.put("docResult", docResult != null ?
```

```
docResult : "");
                        diagnosisMap.put("date", date != null ? date :
"01.01.2023"); // Дефолтная дата

                        userRef.update("diagnosis", diagnosisMap)
                                .addOnSuccessListener(aVoid ->
Log.d("Firestore", "Диагноз с полными параметрами назначен"))
                                .addOnFailureListener(e ->
Log.e("Firestore", "Ошибка при добавлении диагноза", e));
                    }
                });
            } else {
                Log.d("Firestore", "Диагноз уже существует, не
изменяем.");
            }
        }).addOnFailureListener(e -> Log.e("Firestore", "Ошибка при
получении пользователя", e));
    }
}
```

## Приложение 2

*Листинг файла SignUpFragment.java*

```
package com.example.healthcareapp.ui.auth;

import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;


import com.example.healthcareapp.R;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;


public class SignUpFragment extends Fragment {

    private FirebaseAuth auth;
    private EditText email,password;
    private Button SignInButton;
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_sign, container,
false);

        auth = FirebaseAuth.getInstance();
        SignInButton = view.findViewById(R.id.sign);
        email = view.findViewById(R.id.email);
        password = view.findViewById(R.id.password);

        SignInButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String Email = email.getText().toString();
                String Password = password.getText().toString();

auth.createUserWithEmailAndPassword(Email,Password).addOnCompleteListener
(
                        requireActivity(), new
OnCompleteListener<AuthResult>() {
                            @Override
```

```
                                public void onComplete(@NonNull
Task<AuthResult> task) {
                                    if (task.isSuccessful()){
                                        FirebaseUser user =
auth.getCurrentUser();
                                        Log.d("RRR","Sign up success");
                                    }else{
                                        Log.d("RRR","Sign up failed");
                                    }
                                }
                            }
                );
                NavController navController =
Navigation.findNavController(view);

navController.navigate(R.id.action_signupFragment_to_loginFragment);

            }
        });

        return view;
    }
}
```

## Приложение 3

*Листинг файла Doctor.java*

```
package com.example.healthcareapp.data.model;

import com.google.firebase.firestore.PropertyName;

public class Doctor {
    private String id;
    @PropertyName("last_name")
    private String last_name;
    private String name;
    private String otchestvo;
    @PropertyName("specialty")
    private String specialty;
    private String education;
    private String experience;
    private String contacts;
    private String description;

    public Doctor() {

    }

    public String getId() {
        return id;
    }

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

    @PropertyName("last_name")
    public String getLastName() {
        return last_name;
    }
    @PropertyName("last_name")
    public void setLastName(String last_name) {
        this.last_name = last_name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getOtchestvo() {
        return otchestvo;
    }

    public void setOtchestvo(String otchestvo) {
        this.otchestvo = otchestvo;
    }
    @PropertyName("specialty")
```

```
    public String getSpecialty() {
        return specialty;
    }
    @PropertyName("specialty")
    public void setSpecialty(String specialty) {
        this.specialty = specialty;
    }

    public String getEducation() {
        return education;
    }

    public void setEducation(String education) {
        this.education = education;
    }
    public String getExperience() {
        return experience;
    }

    public void setExperience(String experience) {
        this.experience = experience;
    }
    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
    public String getContacts() {
        return contacts;
    }

    public void setContacts(String contacts) {
        this.contacts = contacts;
    }

}
```

## Приложение 4

*Листинг файла DoctorsFragment.java*

```
package com.example.healthcareapp.ui.doctors;

import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.SearchView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;
import androidx.navigation.Navigation;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.healthcareapp.R;
import com.example.healthcareapp.ViewModels.DoctorViewModel;
import com.example.healthcareapp.data.adapters.AdapterCallBack;
import com.example.healthcareapp.data.adapters.DoctorAdapter;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

import java.util.ArrayList;

public class DoctorsFragment extends Fragment {

    private DoctorAdapter adapter;
    private RecyclerView recyclerView;
    private SearchView searchView;
    private DoctorViewModel viewModel;

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_doctors, container,
false);
        recyclerView = v.findViewById(R.id.my_recycler_view);
        searchView = v.findViewById(R.id.searchView);

        adapter = new DoctorAdapter(new ArrayList<>(), doctorId -> {
            Bundle args = new Bundle();
            args.putString("doctorId", doctorId);
            Navigation.findNavController(requireActivity(),
R.id.nav_host_fragment)

.navigate(R.id.action_doctorsFragment_to_doctorDetailFragment, args);
        });

        return v;
    }
```

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

        viewModel = new
ViewModelProvider(this).get(DoctorViewModel.class);

        // Подключаем адаптер
        recyclerView.setLayoutManager(new
LinearLayoutManager(getContext()));
        recyclerView.setAdapter(adapter);

        // Подписка на изменения
        viewModel.getDoctors().observe(getViewLifecycleOwner(), doctors -
> {
            adapter.updateData(doctors);
        });

        // Поиск
        searchView.setOnQueryTextListener(new
SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                searchView.clearFocus();
                return false;
            }

            @Override
            public boolean onQueryTextChange(String newText) {
                adapter.getFilter().filter(newText);
                return true;
            }
        });
    }
    @Override
    public void onResume() {
        super.onResume();
        if (viewModel != null) {
            viewModel.loadDoctors();
        }
    }

}
```

## Приложение 5

*Листинг файла MainActivity.java*

```
package com.example.healthcareapp.ui.main;

import android.os.Bundle;
import android.view.View;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.navigation.NavController;
import androidx.navigation.fragment.NavHostFragment;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;

import com.example.healthcareapp.R;
import com.example.healthcareapp.databinding.ActivityMainBinding;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.firebase.FirebaseApp;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class MainActivity extends AppCompatActivity {
    FirebaseAuth auth;
    BottomNavigationView bottomNavigationView;
    NavController navController;

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

        auth = FirebaseAuth.getInstance();
        FirebaseUser currentUser = auth.getCurrentUser();

        bottomNavigationView = findViewById(R.id.bottom_nav);
        NavHostFragment navHostFragment = (NavHostFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.nav_host_fragment);
        navController = navHostFragment.getNavController();


        NavigationUI.setupWithNavController(bottomNavigationView,
navController);


        if (currentUser == null) {
            bottomNavigationView.setVisibility(View.GONE);
            navController.navigate(R.id.loginFragment);
        } else {
            bottomNavigationView.setVisibility(View.VISIBLE);
        }


        navController.addOnDestinationChangedListener((controller,
destination, arguments) -> {
            if (destination.getId() == R.id.loginFragment ||
```

```
destination.getId() == R.id.signupFragment) {
                bottomNavigationView.setVisibility(View.GONE);
            } else {
                bottomNavigationView.setVisibility(View.VISIBLE);
            }
        });
    }
}
```

## Приложение 6

*Листинг файла DoctorAdapter.java*

```
package com.example.healthcareapp.data.adapters;

import android.annotation.SuppressLint;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Filter;
import android.widget.Filterable;
import android.widget.TextView;

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

import com.example.healthcareapp.R;
import com.example.healthcareapp.data.model.Doctor;

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

public class DoctorAdapter extends
RecyclerView.Adapter<DoctorAdapter.DoctorViewHolder> implements
Filterable {

    private final List<Doctor> filteredList = new ArrayList<>();
    private final List<Doctor> fullList = new ArrayList<>();
    private final AdapterCallBack callback;

    public DoctorAdapter(List<Doctor> doctors, AdapterCallBack callback)
{
        if (doctors != null) {
            filteredList.addAll(doctors);
            fullList.addAll(doctors);
        }
        this.callback = callback;
    }

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

    @SuppressLint("SetTextI18n")
    @Override
    public void onBindViewHolder(@NonNull DoctorViewHolder holder, int
position) {
        Doctor doctor = filteredList.get(position);

        String fullName = "";
        if (doctor.getLastName() != null) fullName +=
```

```
doctor.getLastName() + " ";
        if (doctor.getName() != null) fullName += doctor.getName() + " ";
        if (doctor.getOtchestvo() != null) fullName +=
doctor.getOtchestvo();

        holder.fullName.setText(fullName.trim().isEmpty() ? "Без имени" :
fullName.trim());
        holder.specialty.setText(doctor.getSpecialty() != null ?
doctor.getSpecialty() : "Нет специальности");

        holder.itemView.setOnClickListener(v -> {
            if (callback != null && doctor.getId() != null &&
!doctor.getId().isEmpty()) {
                callback.onClickDoctor(doctor.getId());
            }
        });
    }

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

    public void updateData(List<Doctor> newList) {
        filteredList.clear();
        fullList.clear();
        if (newList != null) {
            filteredList.addAll(newList);
            fullList.addAll(newList);
        }
        notifyDataSetChanged();
    }

    @Override
    public Filter getFilter() {
        return doctorFilter;
    }

    private final Filter doctorFilter = new Filter() {
        @Override
        protected FilterResults performFiltering(CharSequence constraint)
{
            List<Doctor> resultList = new ArrayList<>();
            if (constraint == null || constraint.length() == 0) {
                resultList.addAll(fullList);
            } else {
                String filterPattern =
constraint.toString().toLowerCase().trim();
                for (Doctor doc : fullList) {
                    if ((doc.getName() != null &&
doc.getName().toLowerCase().contains(filterPattern)) ||
                            (doc.getLastName() != null &&
doc.getLastName().toLowerCase().contains(filterPattern)) ||
                            (doc.getOtchestvo() != null &&
doc.getOtchestvo().toLowerCase().contains(filterPattern)) ||
                            (doc.getSpecialty() != null &&
doc.getSpecialty().toLowerCase().contains(filterPattern))) {
                        resultList.add(doc);
```

```
                    }
                }
            }

            FilterResults results = new FilterResults();
            results.values = resultList;
            results.count = resultList.size();
            return results;
        }

        @Override
        protected void publishResults(CharSequence constraint,
FilterResults results) {
            filteredList.clear();
            if (results.values != null) {
                filteredList.addAll((List<Doctor>) results.values);
            }
            notifyDataSetChanged();
        }
    };

    public static class DoctorViewHolder extends RecyclerView.ViewHolder
{
        TextView fullName, specialty;

        public DoctorViewHolder(@NonNull View itemView) {
            super(itemView);
            fullName = itemView.findViewById(R.id.doctor_full_name);
            specialty = itemView.findViewById(R.id.doctor_specialty);
        }
    }
}
```

## Приложение 7

*Листинг файла EditProfileFragment.java*

```
package com.example.healthcareapp.ui.doctors;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.RadioGroup;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;

import com.example.healthcareapp.R;
import com.example.healthcareapp.ViewModels.ProfileViewModel;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.HashMap;
import java.util.Map;

public class EditProfileFragment extends Fragment {

    private ImageView strelochka, profileImage;
    private FirebaseAuth auth;
    private FirebaseFirestore db;

    private Button save;
    private ProfileViewModel viewModel;
    private EditText etName, etSurname, etPatronymic, etOms, etBirthDate,
etNumber;
    private RadioGroup genderRadioGroup;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_edit_profile,
container, false);
    }

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

        viewModel = new
```

```
ViewModelProvider(requireActivity()).get(ProfileViewModel.class);
        auth = FirebaseAuth.getInstance();
        db = FirebaseFirestore.getInstance();


        strelochka = view.findViewById(R.id.strelochka);
        save = view.findViewById(R.id.saveButton);
        etName = view.findViewById(R.id.customEditTextWithMaterial);
        etSurname = view.findViewById(R.id.familia);
        etPatronymic = view.findViewById(R.id.otchestvo);
        etOms = view.findViewById(R.id.oms);
        etBirthDate = view.findViewById(R.id.birth_date);
        etNumber = view.findViewById(R.id.phone_number);
        profileImage = view.findViewById(R.id.avatarka);
        genderRadioGroup = view.findViewById(R.id.genderRadioGroup);


        NavController navController = Navigation.findNavController(view);
        strelochka.setOnClickListener(v ->
navController.navigate(R.id.action_EditProfileFragment_to_ProfileFragment
));


        genderRadioGroup.setOnCheckedChangeListener((group, checkedId) ->
{
            if (checkedId == R.id.maleRadio) {
                profileImage.setImageResource(R.drawable.male);
                viewModel.setGender("male");
            } else if (checkedId == R.id.femaleRadio) {
                profileImage.setImageResource(R.drawable.avatar);
                viewModel.setGender("female");
            }
        });


        save.setOnClickListener(v -> {
            saveDataToViewModel();
            saveDataToFirestore();
        });


        loadDataFromFirestore();
    }

    private void loadDataFromFirestore() {
        String userId = auth.getCurrentUser().getUid();

        db.collection("users").document(userId)
                .get()
                .addOnSuccessListener(documentSnapshot -> {
                    if (documentSnapshot.exists()) {
                        fillFieldsFromDocument(documentSnapshot);
                        updateViewModelFromDocument(documentSnapshot);
                    }
                })
                .addOnFailureListener(e ->
                        Toast.makeText(getContext(), "Ошибка загрузки
данных", Toast.LENGTH_SHORT).show()
```

```
                );
    }

    private void fillFieldsFromDocument(DocumentSnapshot doc) {
        etName.setText(doc.getString("name"));
        etSurname.setText(doc.getString("surname"));
        etPatronymic.setText(doc.getString("patronymic"));
        etOms.setText(doc.getString("omsNumber"));
        etBirthDate.setText(doc.getString("birthDate"));
        etNumber.setText(doc.getString("phone"));

        String gender = doc.getString("gender");
        if ("male".equals(gender)) {
            genderRadioGroup.check(R.id.maleRadio);
            profileImage.setImageResource(R.drawable.male);
        } else if ("female".equals(gender)) {
            genderRadioGroup.check(R.id.femaleRadio);
            profileImage.setImageResource(R.drawable.avatar);
        }
    }

    private void updateViewModelFromDocument(DocumentSnapshot doc) {
        viewModel.setName(doc.getString("name"));
        viewModel.setSurname(doc.getString("surname"));
        viewModel.setPatronymic(doc.getString("patronymic"));
        viewModel.setOmsNumber(doc.getString("omsNumber"));
        viewModel.setBirthDate(doc.getString("birthDate"));
        viewModel.setPhone(doc.getString("phone"));
        viewModel.setGender(doc.getString("gender"));
    }

    private void saveDataToViewModel() {
        viewModel.setName(etName.getText().toString());
        viewModel.setSurname(etSurname.getText().toString());
        viewModel.setPatronymic(etPatronymic.getText().toString());
        viewModel.setOmsNumber(etOms.getText().toString());
        viewModel.setBirthDate(etBirthDate.getText().toString());
        viewModel.setPhone(etNumber.getText().toString());

        int selectedGenderId =
genderRadioGroup.getCheckedRadioButtonId();
        if (selectedGenderId == R.id.maleRadio) {
            viewModel.setGender("male");
        } else if (selectedGenderId == R.id.femaleRadio) {
            viewModel.setGender("female");
        }
    }

    private void saveDataToFirestore() {
        String userId = auth.getCurrentUser().getUid();

        Map<String, Object> userData = new HashMap<>();
        userData.put("name", etName.getText().toString());
        userData.put("surname", etSurname.getText().toString());
        userData.put("patronymic", etPatronymic.getText().toString());
        userData.put("omsNumber", etOms.getText().toString());
        userData.put("birthDate", etBirthDate.getText().toString());
        userData.put("phone", etNumber.getText().toString());
```

```
        int selectedGenderId =
genderRadioGroup.getCheckedRadioButtonId();
        if (selectedGenderId == R.id.maleRadio) {
            userData.put("gender", "male");
        } else if (selectedGenderId == R.id.femaleRadio) {
            userData.put("gender", "female");
        }

        db.collection("users").document(userId)
                .update(userData)
                .addOnSuccessListener(aVoid ->
                        Toast.makeText(getContext(), "Данные успешно
сохранены", Toast.LENGTH_SHORT).show()
                )
                .addOnFailureListener(e ->
                        Toast.makeText(getContext(), "Ошибка сохранения
данных", Toast.LENGTH_SHORT).show()
                );
    }
}
```

## Приложение 8

*Листинг файла ProfileFragment.java*

```
package com.example.healthcareapp.ui.doctors;

import android.os.Bundle;
import android.util.Log;
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.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;
import androidx.navigation.NavController;
import androidx.navigation.fragment.NavHostFragment;

import com.example.healthcareapp.R;
import com.example.healthcareapp.ViewModels.ProfileViewModel;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;

public class ProfileFragment extends Fragment {

    private ImageView iconn, avatar;
    private TextView name, last_name, otchestvo, oms, date, phone;

    private FirebaseFirestore db;
    private FirebaseAuth auth;

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_profile, container,
false);
        iconn = v.findViewById(R.id.icon12);
        avatar = v.findViewById(R.id.avatarka);
        name = v.findViewById(R.id.nameValue);
        last_name = v.findViewById(R.id.surnameValue);
        otchestvo = v.findViewById(R.id.middleNameValue);
        oms = v.findViewById(R.id.omsValue);
        date = v.findViewById(R.id.birthDateValue);
        phone = v.findViewById(R.id.phone);
        return v;
    }

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

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

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

        db.collection("users").document(userId)
                .addSnapshotListener(new
EventListener<DocumentSnapshot>() {
                    @Override
                    public void onEvent(@Nullable DocumentSnapshot value,
@Nullable FirebaseFirestoreException error) {
                        if (error != null) {
                            Log.e("Firestore error", error.getMessage());
                            return;
                        }

                        if (value != null && value.exists()) {
                            String nameValue = value.getString("name");
                            String surnameValue =
value.getString("surname");
                            String patronymicValue =
value.getString("patronymic");
                            String omsValue =
value.getString("omsNumber");
                            String birthDateValue =
value.getString("birthDate");
                            String phoneValue = value.getString("phone");
                            String genderValue =
value.getString("gender");

                            name.setText(nameValue != null ? nameValue :
"");
                            last_name.setText(surnameValue != null ?
surnameValue : "");
                            otchestvo.setText(patronymicValue != null ?
patronymicValue : "");
                            oms.setText(omsValue != null ? omsValue :
"");
                            date.setText(birthDateValue != null ?
birthDateValue : "");
                            phone.setText(phoneValue != null ? phoneValue
: "");

                            if ("male".equalsIgnoreCase(genderValue)) {
                                avatar.setImageResource(R.drawable.male);
                            } else if
("female".equalsIgnoreCase(genderValue)) {

avatar.setImageResource(R.drawable.avatar);
                            } else {

avatar.setImageResource(R.drawable._303173_account_avatar_client_man_pers
on_icon);
                            }
```

```
                        }
                    }
                });

        ProfileViewModel viewModel = new
ViewModelProvider(requireActivity()).get(ProfileViewModel.class);

        if (viewModel.getName() != null)
name.setText(viewModel.getName());
        if (viewModel.getSurname() != null)
last_name.setText(viewModel.getSurname());
        if (viewModel.getPatronymic() != null)
otchestvo.setText(viewModel.getPatronymic());
        if (viewModel.getOmsNumber() != null)
oms.setText(viewModel.getOmsNumber());
        if (viewModel.getBirthDate() != null)
date.setText(viewModel.getBirthDate());
        if (viewModel.getPhone() != null)
phone.setText(viewModel.getPhone());

        NavHostFragment navHostFragment = (NavHostFragment) getActivity()
                .getSupportFragmentManager()
                .findFragmentById(R.id.nav_host_fragment);

        if (navHostFragment != null) {
            NavController navController =
navHostFragment.getNavController();

            iconn.setOnClickListener(v -> {
                if (navController != null) {

navController.navigate(R.id.action_profilefragment_to_profile_edit_fragme
nt);
                }
            });
        }
    }
}
```

## Приложение 9

*Листинг файла ProfileViewModel.java*

```
package com.example.healthcareapp.ViewModels;

import androidx.lifecycle.ViewModel;

public class ProfileViewModel extends ViewModel {
    private String name = "";
    private String surname = "";
    private String patronymic = "";
    private String omsNumber = "";
    private String birthDate = "";

    private String phone = "";

    private String gender ="";

    public String getGender() { return gender; }
    public void setGender(String gender) { this.gender = gender; }


    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public String getSurname() { return surname; }
    public void setSurname(String surname) { this.surname = surname; }

    public String getPatronymic() { return patronymic; }
    public void setPatronymic(String patronymic) { this.patronymic =
patronymic; }

    public String getOmsNumber() { return omsNumber; }
    public void setOmsNumber(String omsNumber) { this.omsNumber =
omsNumber; }

    public String getBirthDate() { return birthDate; }
    public void setBirthDate(String birthDate) { this.birthDate =
birthDate; }

    public String getPhone(){ return phone;}
    public void setPhone(String phone){ this.phone = phone;}
}
```

## Приложение 10

*Листинг файла DoctorViewModel.java*

```
package com.example.healthcareapp.ViewModels;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.healthcareapp.data.model.Doctor;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;

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

public class DoctorViewModel extends ViewModel {

    private final MutableLiveData<List<Doctor>> doctorsLiveData = new
MutableLiveData<>();
    private final FirebaseFirestore db = FirebaseFirestore.getInstance();

    public LiveData<List<Doctor>> getDoctors() {
        if (doctorsLiveData.getValue() == null ||
doctorsLiveData.getValue().isEmpty()) {
            loadDoctors();
        }
        return doctorsLiveData;
    }

    public void loadDoctors() {
        db.collection("doctor")
                .get()
                .addOnSuccessListener(snapshot -> {
                    List<Doctor> list = new ArrayList<>();
                    for (DocumentSnapshot doc : snapshot.getDocuments())
{
                        Doctor doctor = doc.toObject(Doctor.class);
                        if (doctor != null) {
                            doctor.setId(doc.getId());
                            list.add(doctor);
                        }
                    }
                    doctorsLiveData.setValue(list);
                });
    }

}
```