УДК 004

Реализация Google Sign-In в Android-приложении с использованием Firebase

Андриенко Иван Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается процесс интеграции Google Sign-In и Firebase Authentication в Android-приложение для обеспечения удобной и безопасной авторизации пользователей. Представлены основные этапы настройки проекта, включая добавление необходимых зависимостей, создание и настройку параметров Google Sign-In, инициализацию FirebaseAuth, а также обработку результатов входа и авторизацию пользователя в Firebase. Пример кода демонстрирует, как легко и эффективно реализовать авторизацию через Google, обеспечивая высокий уровень производительности и безопасности.
Ключевые слова: Google Sign-In, Firebase Authentication, Android, авторизация, интеграция, мобильная разработка, Android Studio, Firebase Console.

Implementing Google Sign-In in an Android Application using Firebase

Andrienko Ivan Sergeevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses the process of integrating Google Sign-In and Firebase Authentication into an Android application to ensure convenient and secure user authorization. The main stages of project configuration are presented, including adding the necessary dependencies, creating and configuring Google Sign-In parameters, initializing FirebaseAuth, as well as processing login results and user authorization in Firebase. The sample code demonstrates how to easily and effectively implement authorization through Google, providing a high level of performance and security.
Keywords: Google Sign-In, Firebase Authentication, Android, Authorization, Integration, Mobile development, Android Studio, Firebase Console.

1 Введение

1.1 Актуальность

В современном мире мобильных приложений авторизация через социальные сети и сторонние сервисы стала стандартом, обеспечивая

пользователям удобный и безопасный способ входа в приложение. Одним из наиболее популярных методов авторизации является использование учетной записи Google. Google Sign-In позволяет пользователям входить в приложения с помощью своей учетной записи Google, упрощая процесс регистрации и входа. Firebase Authentication, в свою очередь, предоставляет мощные инструменты для управления пользователями и их авторизацией, поддерживая различные методы аутентификации, включая Google Sign-In. Интеграция этих двух сервисов в Android-приложение позволяет разработчикам быстро и легко настроить безопасную и удобную авторизацию

1.2 Обзор исследований

С.А. Федоров в своей работе описывает применение Firebase для создания и масштабирования Android-приложений. Автор рассматривает основные возможности Firebase, такие как аутентификация, базы данных в реальном времени, аналитика и другие инструменты, которые помогают разработчикам создавать высокопроизводительные и масштабируемые приложения. В своей работе Р.В. Мосолов обсуждает альтернативы Firebase, такие как NoSQL СУБД GoldenRaceDB, и сравнивает их с Firebase по различным критериям, включая производительность, удобство использования и возможности масштабирования. А.С. Мосягин подробно рассматривает принципы авторизации в системе Google, приводя конкретные примеры реализации. В статье обсуждаются различные аспекты безопасности и удобства использования Google Sign-In для авторизации пользователей. К.О. Атеев в своей работе описывает создание простой аутентификации с использованием технологий React и Firebase, подчеркивая преимущества использования Firebase для управления пользовательскими данными и аутентификацией. В работе Н.С. Тимошенко, П.Е. Вдовых, Р.Е. Кутумбаев и др. исследуют создание баз данных с помощью сервиса Firebase. Авторы приводят примеры практического применения Firebase в образовательных и научных проектах, подчеркивая его простоту и эффективность в создании и управлении базами данных.

1.3 Цель исследования

Цель исследования – изучить способ интеграции Google Sign-In и Firebase Authentication в Android-приложении для обеспечения безопасной, быстрой и удобной авторизации пользователей.

2 Материалы и методы

Для реализации интеграции Google Sign-In и Firebase Authentication использовались Android Studio и язык программирования Java. В проект были добавлены зависимости для Google Sign-In и Firebase Authentication. Настройка параметров Google Sign-In осуществлялась с использованием GoogleSignInOptions, что позволяет запросить ID токен и электронную почту пользователя.

3 Результаты и обсуждения

Firebase Console — это мощный инструмент для управления проектами Firebase. Она предоставляет разработчикам удобный веб-интерфейс для настройки и управления различными сервисами Firebase, такими как аутентификация, базы данных, аналитика, облачные функции и другие. Используя Firebase Console, разработчики могут интегрировать в свои приложения широкие возможности. Такая интеграция подразумевает выполнение ряда технических действий, включая настройку API взаимодействия с Google и Firebase, а также адаптацию мобильного приложения под новые требования безопасности и удобства пользователя (рис.1).

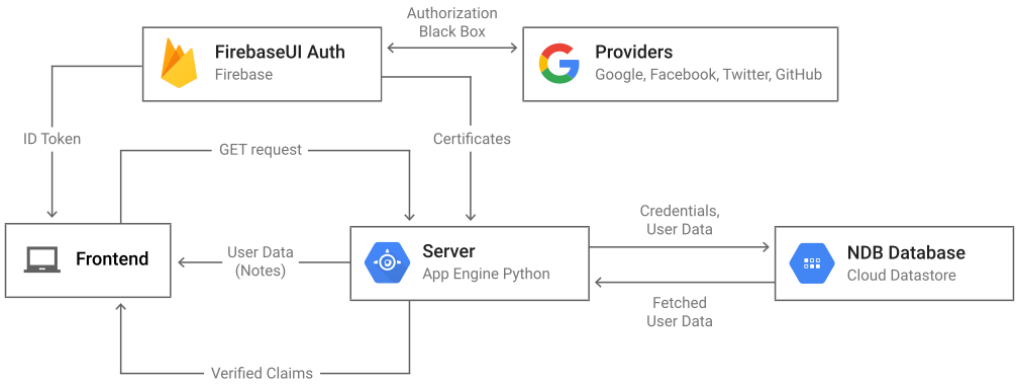


Рис. 1. Схема работы авторизации

Для реализации поставленной задачи по интеграции авторизации через сервис Google и базы данных Firebase в мобильное приложение следует произвести подбор программных средств, учитывая несколько ключевых аспектов:

1. Интеграция с сервисом Google:

Для обеспечения авторизации через Google необходимо выбрать библиотеку или SDK, которая позволит интегрировать функционал авторизации с учетными записями Google в мобильное приложение. Подходящими инструментами для этого могут быть Google Sign-In SDK или Firebase Authentication, которые предоставляют готовые решения для работы с авторизацией через Google.

2. Использование Firebase в качестве базы данных:

Для хранения пользовательских данных и управления ими на сервере потребуется база данных. Firebase предоставляет облачное решение для этой цели, обеспечивая гибкость, масштабируемость и удобный API для работы с данными. Для интеграции Firebase Database в мобильное приложение можно использовать Firebase SDK, который предоставляет набор инструментов для работы с базой данных в реальном времени.

3. Совместимость и документация:

При выборе программных средств необходимо убедиться в их совместимости с выбранной платформой разработки (например, Android или iOS) и языком программирования. Также важно обратить внимание на

качество документации и наличие примеров использования, чтобы облегчить интеграцию и разработку.

4. Безопасность и производительность:

При работе с данными пользователей важно обеспечить высокий уровень безопасности, поэтому выбранные программные средства должны предоставлять средства для защиты данных. Также необходимо оценить производительность выбранных инструментов и их способность масштабироваться при увеличении количества пользователей и объема данных.

Прежде всего, необходимо добавить Firebase в свой проект Android. В Android Studio это можно выполнить в несколько шагов. Создать свой проект в Android Studio. Перейти во вкладку Tools в верхнем меню и выбрать Firebase, что откроет панель Assistant справа. В панели Firebase Assistant раздел «Authentication», затем «Email and password authentication» для добавления возможности аутентификации через Firebase в мобильное приложение. После этих действий файлы gradle синхронизируются, что обеспечит необходимые зависимости в проекте (рис. 2).

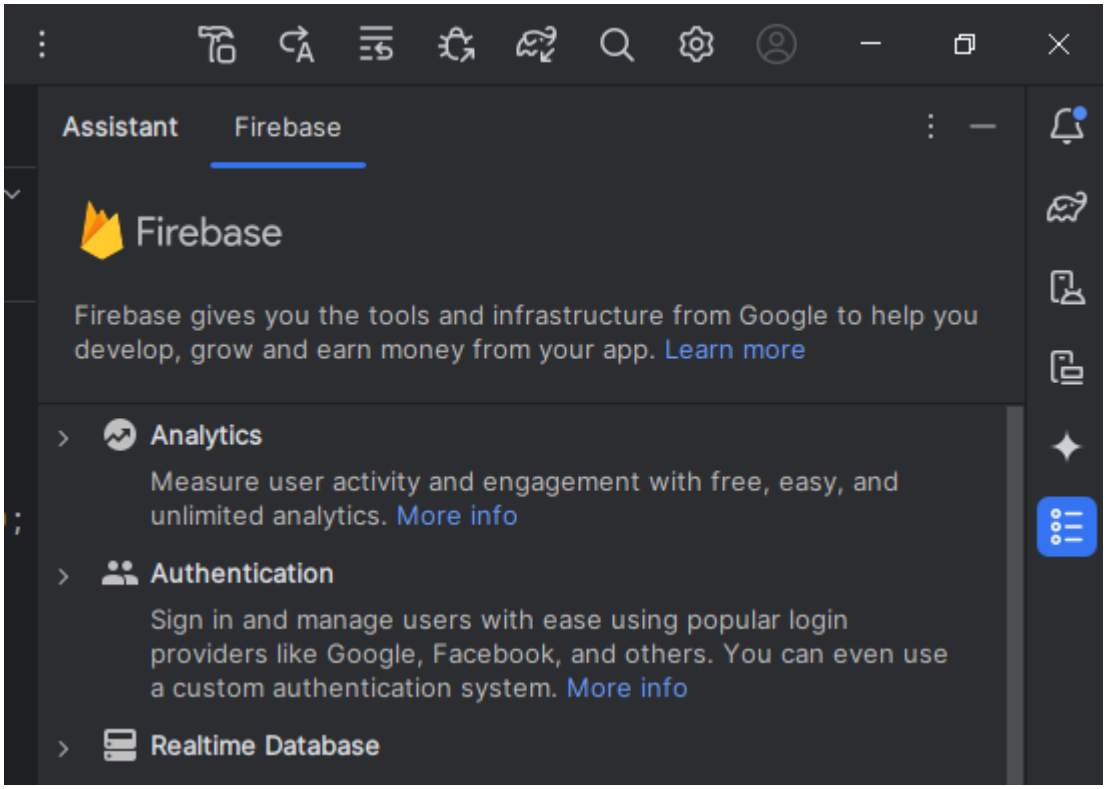


Рис. 2. Использование Firebase Assistant

В файле Gradle модуля (на уровне приложения) необходимы зависимости для аутентификации Firebase, библиотека для Android. Рекомендуется использовать Firebase Android BoM для управления версиями библиотеки. Кроме того, в рамках настройки аутентификации Firebase необходимо добавить в приложение SDK сервисы Google Play. Используя

Firebase Android BoM, приложение всегда будет использовать совместимые версии библиотек Firebase Android (рис. 3).

```
dependencies {

    implementation libs.appcompat
    implementation libs.material
    implementation libs.constraintlayout
    implementation libs.lifecycle.livedata.ktx
    implementation libs.lifecycle.viewmodel.ktx
    implementation libs.navigation.fragment
    implementation libs.navigation.ui
    implementation libs.firebase.auth
    implementation libs.activity
    implementation libs.ads.mobile.sdk
    testImplementation libs.junit
    androidTestImplementation libs.ext.junit
    androidTestImplementation libs.espresso.core

    //Firebase
    implementation("com.google.android.gms:play-services-auth:21.1.0")
    implementation("com.google.firebase:firebase-auth")
    implementation(platform("com.google.firebase:firebase-bom:32.8.1"))
```

Рис. 3. Зависимости для Firebase

При авторизации в Firebase от разработчика требуется отпечаток SHA. SHA (Secure Hash Algorithm) — это семейство криптографических хеш-функций, используемое для создания уникального, фиксированного размера хеш-значения из данных. В контексте мобильных приложений, отпечаток SHA (обычно SHA-1 или SHA-256) часто используется для идентификации и безопасности.

Для интеграции входа в Google One Tap в приложение необходимо при настройке объекта BeginSignInRequest вызвать метод setGoogleIdTokenRequestOptions(). В этом методе указывается корректный идентификатор клиента сервера с помощью метода setServerClientId(), а также необходимо убедиться, что опция фильтрации по авторизованным аккаунтам установлена в значение true с помощью метода setFilterByAuthorizedAccounts().

Для начала была добавлена проверка статуса аутентификации пользователя в приложении. В этом участке кода создается экземпляр объекта FirebaseAuth с помощью метода getInstance(). Затем получается текущий авторизованный пользователь с помощью метода getCurrentUser().

Далее используется объект Handler для запуска нового потока выполнения с нулевой задержкой. В этом потоке проверяется, авторизован ли пользователь. Если пользователь авторизован, выполняется соответствующий блок кода. В противном случае пользователь перенаправляется на экран входа, создается Intent для SignInActivity, который запускается с помощью startActivity(), а текущая активность MainActivity завершается с помощью finish(). Этот код обеспечивает контроль над авторизацией пользователей в приложении, направляя их на экран входа при

отсутствии авторизации или выполняя соответствующие действия в случае успешной авторизации (рис. 4).

```
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        mAuth = FirebaseAuth.getInstance();
        final FirebaseUser user = mAuth.getCurrentUser();

        binding.adContainerView.getViewTreeObserver().addOnGlobalLayoutListener(
                new ViewTreeObserver.OnGlobalLayoutListener() {
                    @Override
                    public void onGlobalLayout() {
                        binding.adContainerView.getViewTreeObserver().removeOnGlobalLayoutListener( victim: this);
                        mBannerAd = loadBannerAd(getAdSize());
                    }
                }
        );

        BottomNavigationView navView = findViewById(R.id.nav_view);
        AppBarConfiguration appBarConfiguration = new AppBarConfiguration.Builder(
                R.id.navigation_home, R.id.navigation_dashboard, R.id.navigation_notifications)
                .build();
        NavController navController = Navigation.findNavController( activity: this, R.id.nav_host_fragment_activity_main);
        NavigationUI.setupWithNavController(binding.navView, navController);

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                if (user != null) {
                    // Пользователь авторизован
                } else {
                    Intent signInIntent = new Intent( packageContext: MainActivity.this, SignInActivity.class);
                    startActivity(signInIntent);
                    finish();
                }
            }
        }, delayMillis: 0);
    }
```

Рис. 4. Проверка авторизации

Для реализации процесса авторизации в приложении была создана активность SignInActivity. При запуске активность происходит настройка макета и инициализация объектов для аутентификации пользователей. Используется GoogleSignInOptions для конфигурации запроса аутентификации через Google и создается GoogleSignInClient для выполнения входа через учетные данные Google. При нажатии на кнопку входа вызывается метод signIn(), который запускает активность для входа через Google и получает учетные данные. Результат входа обрабатывается в методе onActivityResult(). Если вход успешен, происходит аутентификация в Firebase с использованием полученных учетных данных Google. Метод firebaseAuthWithGoogle() используется для аутентификации в Firebase. После успешной аутентификации пользователь перенаправляется на главный экран приложения (MainActivity), а активность для входа завершается. Этот класс обеспечивает удобный процесс входа в приложение с использованием учетных данных Google и контроль над авторизацией пользователей, направляя их на экран входа при отсутствии авторизации или выполняя соответствующие действия в случае успешной авторизации (рис. 5,6).

```java
public class SignInActivity extends AppCompatActivity {

    private static final int RC_SIGN_IN = 120;  2 usages
    private FirebaseAuth mAuth;  3 usages
    private GoogleSignInClient mGoogleSignInClient;  2 usages

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_sign_in);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
        setContentView(R.layout.activity_sign_in);

        // Configure Google Sign In
        GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
                .requestIdToken(getString(R.string.default_web_client_id))
                .requestEmail()
                .build();
        mGoogleSignInClient = GoogleSignIn.getClient( activity: this, gso);

        // Firebase Auth instance
        mAuth = FirebaseAuth.getInstance();

        findViewById(R.id.sign_in_btn).setOnClickListener(view -> signIn());
    }

    private void signIn() {  1 usage
        Intent signInIntent = mGoogleSignInClient.getSignInIntent();
        startActivityForResult(signInIntent, RC_SIGN_IN);
    }
```

Рис. 5. Основная активность авторизации

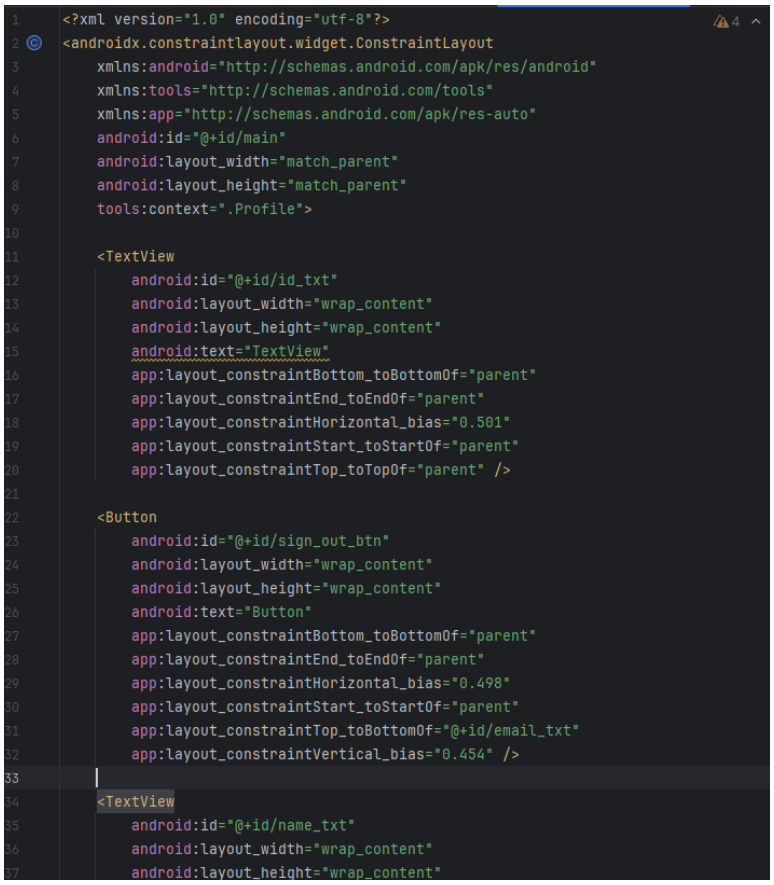
```java
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        // Result returned from launching the Intent from GoogleSignInApi.getSignInIntent(...);
        if (requestCode == RC_SIGN_IN) {
            Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
            try {
                // Google Sign In was successful, authenticate with Firebase
                GoogleSignInAccount account = task.getResult(ApiException.class);
                Log.d( tag: "SignInActivity",  msg: "firebaseAuthWithGoogle:" + account.getId());
                firebaseAuthWithGoogle(account.getIdToken());
            } catch (ApiException e) {
                // Google Sign In failed, update UI appropriately
                Log.w( tag: "SignInActivity",  msg: "Google sign in failed", e);
            }
        }
    }

    private void firebaseAuthWithGoogle(String idToken) {  1 usage
        AuthCredential credential = GoogleAuthProvider.getCredential(idToken,  s1: null);
        mAuth.signInWithCredential(credential)
                .addOnCompleteListener( activity: this, task -> {
                    if (task.isSuccessful()) {
                        // Sign in success, update UI with the signed-in user's information
                        Log.d( tag: "SignInActivity",  msg: "signInWithCredential:success");
                        FirebaseUser user = mAuth.getCurrentUser();
                        Intent intent = new Intent( packageContext: SignInActivity.this, MainActivity.class);
                        startActivity(intent);
                        finish();
                    } else {
                        // If sign in fails, display a message to the user.
                        Log.w( tag: "SignInActivity",  msg: "signInWithCredential:failure", task.getException());
                    }
                });
    }
```

Рис. 6. Основная активность авторизации

Для отображения профиля текущего пользователя была создана активность Profile. В макете были добавлены поля для вывода имени пользователя, индивидуального id и почты пользователя. При запуске активности происходит настройка макета и инициализация объектов для работы с аутентификацией пользователей. В активности используется объект FirebaseAuth для работы с аутентификацией Firebase. Текущий пользователь получается с помощью метода mAuth.getCurrentUser(). Далее находятся View-элементы, такие как TextView для отображения идентификатора, имени и электронной почты пользователя, а также Button для кнопки выхода из аккаунта. Если текущий пользователь не равен null, его данные устанавливаются в соответствующие TextView. При нажатии на кнопку выхода (sign_out_btn) вызывается метод mAuth.signOut(), после чего происходит переход на экран входа и завершение текущей активности. Этот класс обеспечивает удобное отображение профиля текущего пользователя и контроль над процессом выхода из аккаунта (рис.7, 8).



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Profile">

    <TextView
        android:id="@+id/id_txt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.501"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/sign_out_btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/email_txt"
        app:layout_constraintVertical_bias="0.454" />

    <TextView
        android:id="@+id/name_txt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

Рис. 7. Макет кабинета пользователя

```java
public class Profile extends AppCompatActivity {

    private FirebaseAuth mAuth;  3 usages
    private TextView id_txt, name_txt, email_txt;  2 usages
    private ImageView profile_image;  no usages
    private Button sign_out_btn;  2 usages

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

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

        id_txt = findViewById(R.id.id_txt);
        name_txt = findViewById(R.id.name_txt);
        email_txt = findViewById(R.id.email_txt);
        //profile_image = findViewById(R.id.profile_image);
        sign_out_btn = findViewById(R.id.sign_out_btn);

        if (currentUser != null) {
            id_txt.setText(currentUser.getUid());
            name_txt.setText(currentUser.getDisplayName());
            email_txt.setText(currentUser.getEmail());
            //Glide.with(this).load(currentUser.getPhotoUrl()).into(profile_image);
        }

        sign_out_btn.setOnClickListener(v -> {
            mAuth.signOut();
            Intent intent = new Intent( packageContext: Profile.this, SignInActivity.class);
            startActivity(intent);
            finish();
        });
    }
}
```

Рис. 8. Активность кабинета пользователя

Выводы

В данной статье рассмотрена интеграция Google Sign-In и Firebase Authentication в Android-приложение, что позволяет обеспечить безопасную, быструю и удобную авторизацию пользователей. Представленный подход демонстрирует, как можно использовать мощные возможности Firebase Console для управления проектом и его масштабирования, обеспечивая высокую производительность и улучшенный пользовательский опыт. Использование Google Sign-In предоставляет пользователям возможность входа с помощью их существующих учетных записей Google, что значительно упрощает процесс регистрации и входа. Firebase Authentication, в свою очередь, обеспечивает надежную аутентификацию и управление пользователями, что особенно важно для поддержания безопасности приложения.

Библиографический список

1. Федоров С.А. Применение Firebase в создании и масштабировании Android-приложений // Вестник науки. 2024. Т. 4. № 1 (70). С. 502-513.
2. Мосолов Р.В. О разработке NoSQL СУБД GoldenRaceDB как альтернативы Google Firebase // Электронные библиотеки. 2023. Т. 26. №

4. С. 498-517.
3. Мосягин А.С. Рассмотрение принципов авторизации в системе Google с примерами // Аллея науки. 2018. Т. 3. № 7 (23). С. 119-124.
4. Атеев К.О. Создание простой аутентификации с использованием технологий React и Firebase // Инновации. Наука. Образование. 2020. № 23. С. 322-334.
5. Тимошенко Н.С., Вдовых П.Е., Кутумбаев Р.Е., Хоркуш А.В. Создание базы данных при помощи сервиса Firebase // Инновационные технологии в образовании и науке. Сборник материалов Международной научно-практической конференции. В 2-х томах. Редколлегия: О.Н. Широков [и др.]. 2017. С. 78-81.