Android
-
Firebase

Peter Borovanský
KAI, I-18
MS-Teams: 2sf3ph4, List, github
borovan 'at' ii.fmph.uniba.sk

- **Room**
  - **@Database**
  - **@Entity**
  - **@Dao**

- **Firebase**
  - **Authentification**
    - **Email/Password, Google, FB, Twitter**
  - **Realdatabase**
  - **Storage**
  - **Push notifications**

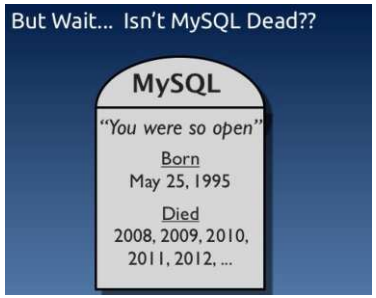
Kam (inam) uložiť naše dáta

bolo minule:

W - Windows
A - Apache
M - MySql
P - Php
WAMP - Server

- máme vlastný server – protokol najčastejšie http-https
  - najčastejšie (v bakalárkach) AMP – Apache-MySQL-PHP, prip. java-servlets
  - tak trochu doba minulá...

LinuxDays 2017:
MySQL sežere Vaše data
David Karban



bude:

- lokálne
  - SQLite https://developer.android.com/training/data-storage/sqlite
  - Room https://developer.android.com/topic/libraries/architecture/room

- cloudové úložiská a ich služby
  - Relačné - tabuľkovo orientované
    Parse.com http://parseplatform.org/ (kúpil Facebook)

  - Nerelačné - noSQL – json
    firebase.com (Google)

SQLite vs. Room

SQLite databáza:

- obsahovala veľa *boilerplate* kódu
- operácie, ktoré blokovali main-thread
- sql dotazy, ktoré sa konštruujú v run-time (nie sú predkompilované)

> **Caution:** Although these APIs are powerful, they are fairly low-level and require a great deal of time and effort to use:
>
> - There is no compile-time verification of raw SQL queries. As your data graph changes, you need to update the affected SQL queries manually. This process can be time consuming and error prone.
> - You need to use lots of boilerplate code to convert between SQL queries and data objects.
>
> For these reasons, we **highly recommended** using the Room Persistence Library as an abstraction layer for accessing information in your app's SQLite databases.

od 2017 Android má Room, ktorá umožňuje:

- používať komponenty Room (kapt - anotácie @Entity, @Dao, @Database)
- udržiavať relácie medzi entitami (kľúče - keys)
- spúšťať DB operácie mimo main-thread, prirodzene pomocou corutín
- je to odporúčaná náhrada za bývalú SQLite (existuje v Androide od API-1)

Room

```
build.gradle
dependencies {
   implementation "androidx.room:room-runtime:2.2.5"
   kapt "androidx.room:room-compiler:2.2.5"
   implementation "androidx.room:room-ktx:2.2.5"
}
```

```
build.gradle
plugins {
    . . .
    id 'kotlin-android-extensions'
    id 'kotlin-kapt'
}
```

Anotácie:

- @Database – abstraktná trieda RoomDatabase
  - Room.databaseBuilder() persistentná inštancia, dáta existujú aj po skončení procesu,
  - Room.inMemoryDatabaseBuilder() – dáta zmiznú, keď proces zanikne
- @Entity – tabuľky v SQL databáze

```
@Entity
data class Student ( … )
```

- @DAO – data access object – metódy na prístup k databáze

```
@Dao
interface StudentDAO {
    @Query("SELECT * FROM Student WHERE name = :name")
    suspend fun getName(name: String): Student?
```

https://developer.android.com/topic/libraries/architecture/room

RoomDB

malý príklad

- vytvoríme aplikáciu na registrovanie študentov s funkciami:
  - signup/login/logout/delete
- v návrhovom vzore MVVM
- s použitím corutín

Najprv si:

- obohatíme build.gradle (app) o
- room

```
implementation "androidx.room:room-runtime:2.4.3"
kapt "androidx.room:room-compiler:2.4.3"
implementation "androidx.room:room-ktx:2.4.3"
```

- coroutines

```
implementation "org.jetbrains.kotlinx:kotlinx-coroutines-core:1.6.1"
implementation "org.jetbrains.kotlinx:kotlinx-coroutines-android:1.6.1"
```

- plugins

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    id 'kotlin-android-extensions'
    id 'kotlin-kapt'}
```

Project:CoroutinesDB.zip

@Entity

```
@Entity
data class Student (
    val isic              : String,
    val name              : String,
    val passwordHash      : Int,
    @ColumnInfo(passwordHash = "password_hash")
    val description       : String) {
    @PrimaryKey(autoGenerate = true)
    var id: Long = 0
}
```

mapovanie kotlinovskej položky `passwordHash` na stĺpec tabuľky **`Student`** s názvom `password_hash`

`autoGenerate` - autoincrement primary key – najčastejšie sa implementuje ako SEQUENCE (Oracle, Postgre), resp. MS-SQL

```
CREATE TABLE (
  ID_column INT NOT NULL IDENTITY(1,1) PRIMARY KEY
```

https://developer.android.com/reference/androidx/room/Entity

Project:CoroutinesDB.zip

@Entity

foreignKeys/Embedded

```
@Entity
data class User(
    @PrimaryKey val userId: Long,
    val name: String,
    val age: Int
)
@Entity(foreignKeys = [
   ForeignKey(
     entity = User::class,
     parentColumns = ["userId"],
     childColumns = ["userOwnerId"],
     onDelete = CASCADE)]
)
```

```
data class Library(
   @PrimaryKey
   val libraryId: Long,
   val title: String,
   val userOwnerId: Long
)
data class UserAndLibrary(
   @Embedded val user: User,
   @Relation(
     parentColumn = "userId",
     entityColumn = "userOwnerId"
   )
   val library: Library
)
```

https://medium.com/@tonyowen/room-entity-annotations-379150e1ca82

Project:CoroutinesDB.zip

@Dao

data access object

```
@Dao
interface StudentDAO {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insert(student: Student): Long

    @Query("SELECT * FROM Student WHERE name = :name")
    suspend fun getName(name: String): Student?

    @Query("SELECT * FROM Student WHERE id = :id")
    suspend fun getID(id: Long): Student?

    @Query("SELECT * FROM Student WHERE isic = :isic")
    suspend fun getISIC(isic: String): Student?

    @Query("DELETE FROM Student WHERE id = :id")
    suspend fun deleteID(id: Long)

    @Insert
    suspend fun insertAll(vararg students: Student)

    @Delete
    suspend fun delete(student: Student)
}
```

Project:CoroutinesDB.zip

@Database

```
@Database(entities = arrayOf(Student::class), version = 1)
abstract class StudentDatabaseEasy: RoomDatabase() {
    abstract fun studentDAO(): StudentDAO

    fun getInstance(context: Context) = Room.databaseBuilder(
            context.applicationContext,
            StudentDatabaseEasy::class.java,
           "studentdatabase"
    ).build()
}
signup:
coroutineScope.launch {
    if (db.getName(name) != null || db.getISIC(isic) != null) {
        withContext(Dispatchers.Main) {
           error.value = "Student already exists"
        }
    } else {
        val studentId = db.insert(
                Student(isic, name, password.hashCode(), desc))
} }
```

Project:CoroutinesDB.zip

@Database

```
val coroutineScope = CoroutineScope(Dispatchers.IO)

fun login(name: String, password: String) {
    coroutineScope.launch {
    val student = db.getName(name)
    if (student == null)
        withContext(Dispatchers.Main) {
          error.value = "Student not found"
        }
    else {
        if (student.passwordHash == password.hashCode()) {
            Status.login(student)
            withContext(Dispatchers.Main) {
              logged.value = true
            }
        } else {
            withContext(Dispatchers.Main) {
              error.value = "Password is incorrect"
}   }   }   }
```

Project:CoroutinesDB.zip

Rozsiahlejší príklad @Entity

```
@Entity(foreignKeys = [ForeignKey(
        entity = Company::class,
        parentColumns = ["id"],
        childColumns = ["company_id"],
        onDelete = ForeignKey.CASCADE)])

data class Employee (
    @ColumnInfo(name = "name")
    val name: String,

    @ColumnInfo(name = "company_id")
    val companyId : Int = 0)
{
    @PrimaryKey(autoGenerate = true)
    @ColumnInfo(name = "id")
    val employeeId : Int = 0
}
```

```
@Entity(tableName = "Company")
data class Company (
    @ColumnInfo(name = "name")
    val name: String,

    @ColumnInfo(name = "date_updated")
    @TypeConverters(DateConverter::class)
    val itemUpdatedDate: Date? = null,

    @Embedded
    private val location: Location? = null,

    @Embedded(prefix = "hq_")
    private val headLocation: Location? = null,

    @Ignore
    val picture: Bitmap? = null
) {
    @PrimaryKey
    @ColumnInfo(name = "id")
    val companyId = 0
}
```

```
@Entity(tableName = "Department")
class Department(
        @ColumnInfo(name = "name")
        val name: String,

        @ColumnInfo(name = "company_id")
        val companyId : Int = 0
) {
    @PrimaryKey(autoGenerate = true)
    @ColumnInfo(name = "id")
    var id = 0
}
```

```
@Entity(primaryKeys = ["id", "code"])
class Office {
    val id : Int = 0
    var code: String
}
```

Project:CoroutinesDB.zip

Rozsiahlejší príklad @Dao

```
@Dao
interface EmployeeDao {
    @get:Query("SELECT * FROM Employee")
    val allEmployees: LiveData<List<Employee?>?>?
    @RawQuery
    fun getAllEmployeesWithLimit(query: String?): List<Employee?>?
    @Insert
    fun insertEmployee(employee: Employee?)
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insertAll(employees: List<Employee?>?)
    @Update
    fun updateEmployee(employee: Employee?)
    @Delete
    fun deleteEmployee(employee: Employee?)
}
```

```
@Dao
abstract class DepartmentDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    abstract fun insertAll(departments: List<Department?>?)
    @Insert
    abstract fun insert(product: Department?)
    @Delete
    abstract fun delete(product: Department?)
    @Transaction
    fun insertAndDeleteInTransaction(
              newDepartment: Department?,
              oldDepartment: Department?) {
        insert(newDepartment)
        delete(oldDepartment)
    }
}
```

```
@Dao
interface CompanyDao {
    @get:Query("SELECT * FROM Company")
    val allCompanies: LiveData<List<Company?>?>?
    @get:Query("SELECT * FROM Company ORDER BY name")
    val allCompaniesOrdered: LiveData<List<Company?>?>?
    @Insert
    fun insertCompany(company: Company?)
    @Query(
     "SELECT * FROM Company WHERE name LIKE :companyName")
    fun getCompanies(
              companyName: String?):
              LiveData<List<Company?>?>?
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insertAll(companies: List<Company?>?)
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insertAll(vararg companies: Company?)
    @Update
    fun updateCompany(company: Company?)
    @Update
    fun updateCompanies(vararg company: Company?)
    @Delete
    fun deleteCompany(company: Company?)
    @Delete
    fun deleteCompanies(vararg company: Company?)
}
```

Project:CoroutinesDB.zip

Cvičenie - C

Vytvorte aplikáciu, ktorá slúži na **evidenciu známok študentov** s následujúcimi entitami (verím, že tomu zápisu rozumiete). Máte urobiť Room model a minimálne základne GUI, aby bolo jasné, že viete pracovať s Room.
Základná verzia na hodnotenie:
•vie pridať položku do troch tabuliek (delete nemusíte riešiť), Znamky sú statický číselník,
•zobrazuje počet študentov, predmetov a hodnotení.
Bonus: [1 bod] nejaký listview zobrazuje všetky hodnotenia, v ľub. poradí, bez filtrov,
Bonus: [1 bod] viete zmazať študenta/predment, ktorý už má hodnotenie, CASCADE...

```
drop table Student;
create table Student (
  id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
  isic varchar(20) not null,
  meno varchar(20),
  priezvisko varchar(30),
  CONSTRAINT sid_pk PRIMARY KEY (id),
  CONSTRAINT isic_pk UNIQUE (isic)
);
insert into Student (isic, meno, priezvisko)
      values ('123456789', 'Sansa', 'Starkova');
commit;
drop table predmet;
create table Predmet (
  id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
  kod varchar(20) not null,
  nazov varchar(80),
  CONSTRAINT pid_pk PRIMARY KEY (ID),
  CONSTRAINT kod_pk UNIQUE (kod)
);
insert into Predmet (kod, nazov)
       values ('1-AIN-472/12‘,
  'Vývoj mobilných aplikácií, zimný semester 2020/2021');
commit;
```

```
create table Znamka (
  ID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
  text varchar(5),
  CONSTRAINT zid_pk PRIMARY KEY (ID),
  CONSTRAINT text_pk UNIQUE (text)
);
insert into Znamka (text) values ('A');
insert into Znamka (text) values ('B');
insert into Znamka (text) values ('C');
insert into Znamka (text) values ('D');
insert into Znamka (text) values ('E');
insert into Znamka (text) values ('Fx');
create table Hodnotenie (
  studentID NUMBER,
  predmetID NUMBER,
  znamkaID NUMBER,
  CONSTRAINT fk_student FOREIGN KEY (studentID)
              REFERENCES Student(ID),
  CONSTRAINT fk_predmet FOREIGN KEY (predmetID)
              REFERENCES Predmet(ID),
  CONSTRAINT fk_znamka FOREIGN KEY (znamkaID)
              REFERENCES Znamka(ID)
);
```

Ďalšie čítanie - Room

tutoriály

- **Room Persistence Library: Introduction**:
  https://medium.com/@magdamiu/android-room-persistence-library-97ad0d25668e
- **Room Persistence Library: Entity, Dao, Database**:
  https://medium.com/@magdamiu/android-room-components-5a7458b99191
- **Room Persistence Library: Relations**:
  https://medium.com/@magdamiu/android-room-persistence-library-relations-75bbe02e8522
- **Room Persistence Library: Queries and Migration Support**:
  https://medium.com/@magdamiu/android-room-persistence-library-queries-and-migration-support-a9f21d2dc9d8

Cvičenie - C

Malá evidencia produktov pomocou bar-code scannera

EAN 8584004040108

Horalka

See on Ebay

Brand	Sedita
Manufacturer	I.D.C. Holding
EAN	8584004040108
Country	Slovakia
Last Scan	Nov 25 2020 at 11:17 PM
GS1 Name	I.D.C. HOLDING, odš.závod Pečivárne
GS1 Address	Drieňová 3 Bratislava SK
Description	No description for 8584004040108
Barcode	8 584004 040108

Project:BarCodeScanner.zip

Pokračovanie scannera

dekódovanie Covid-Pasu

https://gae-piaz.medium.com/decode-the-eu-green-pass-qrcode-using-java-b5654e55b0fc

Parse vs. Parse Server

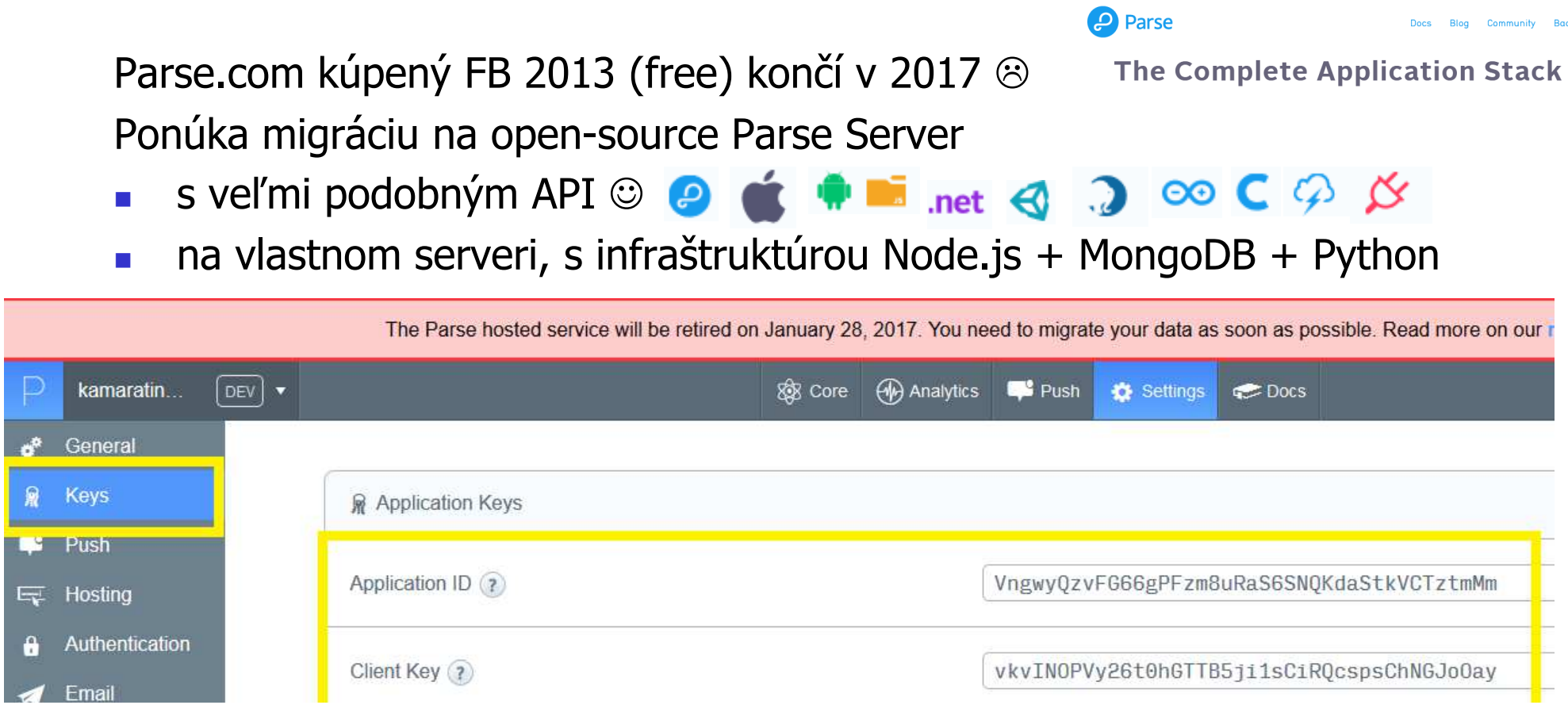
Parse + facebook

- API pre komunikáciu mobilných/web aplikácií, ukladanie/zdieľanie dát
- podporoval PUSH notifikácie pomocou Google Cloud Messaging (GCM)
  vaša aplikácia dostane notifikáciu zo servera, ak iný užívateľ vyvolá event

Parse.com kúpený FB 2013 (free) končí v 2017 ☹

Ponúka migráciu na open-source Parse Server

- s veľmi podobným API ☺
- na vlastnom serveri, s infraštruktúrou Node.js + MongoDB + Python



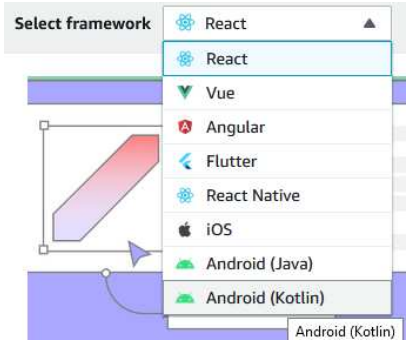
Alternatívy

(k Parse Server)

Veci zadarmo sú (často) síce najlepšie, ale zase pomíjivé

Treba pozrieť konkurenciu:

- Amazon AWS – iOS, Android, Web, ReactNative, ...
  https://aws.amazon.com/amplify/
  - Authentication
  - File Storage
  - Analytics
  - Notifications



- MS Azure Mobile App SDK – iOS, Android, Xamarin, Cordova, ...
  https://docs.microsoft.com/en-us/azure/app-service-mobile/app-service-mobile-android-how-to-use-client-library
  - Databases
    - Azure SQL
    - Azure CosmosDB
    - Azure PostgreSQL
    - Azure for MariaDB
  - Analytics

- Firebase – iOS, Android, Flutter, Web, ...
  - realtime JSON oriented DB,
  - PUSH notifikácie, analytics, REST API

Firebase

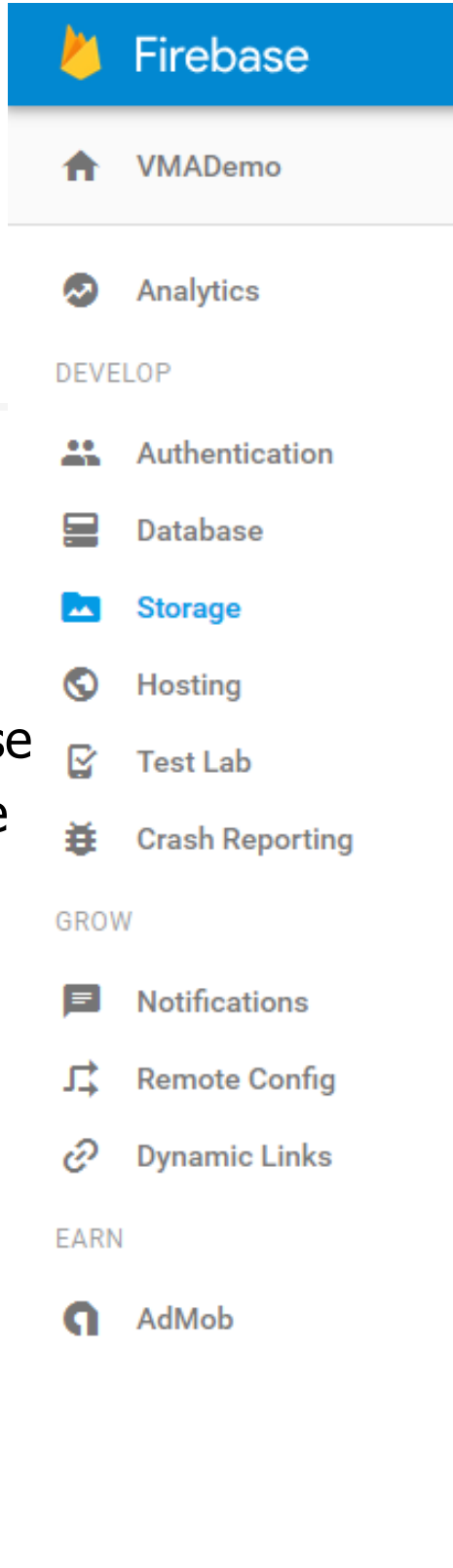
Startup Firebase(2011) kúpil Google(2014)

Ponúka služby/servisy:

- Authentification – FB/Google/Twitter/FirebaseEmail
- Realtime database – non sql store & synchronized database
- Cloud Firestore – realtime scalabe database, works off-line
- File storage – secure upload & download files
- Test lab for Android – rôzne virtuálne zariadenia
- Firebase local emulator – WebApp/Android/iOS
- Cloud messaging – push notifikácie pomocou Google Cloud Messaging/Firebase Cloud Messaging
- Cloud Functions – runs backend scripts on Google servers
- Crash reporting
- Analytics

...

https://firebase.google.com/docs?authuser=0



Firebase Products

and development platforms

Build your app

- Cloud Firestore — iOS, Android, Web
- Firebase ML — iOS, Android
- Cloud Functions — iOS, Android, Web, C++, Unity
- Authentication — iOS, Android, Web, C++, Unity
- Hosting — Web
- Cloud Storage — iOS, Android, Web, C++, Unity
- Realtime Database — iOS, Android, Web, C++, Unity

Improve app quality

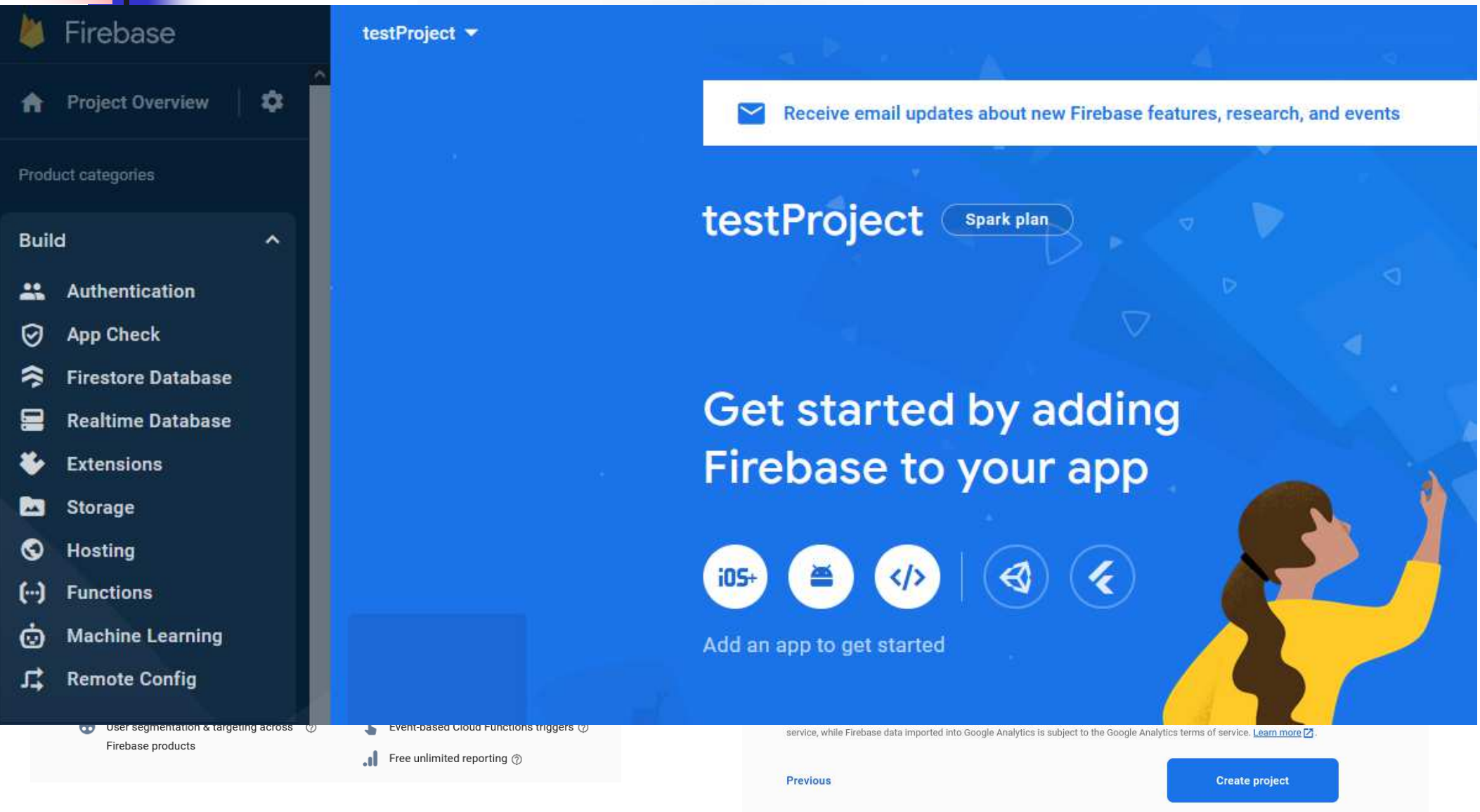
- Crashlytics — iOS, Android, Unity
- App Distribution — iOS, Android
- Performance Monitoring — iOS, Android, Web
- Test Lab — iOS, Android

Grow your business

- Analytics — iOS, Android, Web, C++, Unity
- Extensions — iOS, Android, Web, C++, Unity
- Predictions — iOS, Android, C++, Unity
- Firebase A/B Testing — iOS, Android, C++, Unity
- Cloud Messaging — iOS, Android, Web, C++, Unity
- In-App Messaging — iOS, Android
- Remote Config — iOS, Android, Web, C++, Unity
- Dynamic Links — iOS, Android, C++, Unity
- App Indexing — iOS, Android

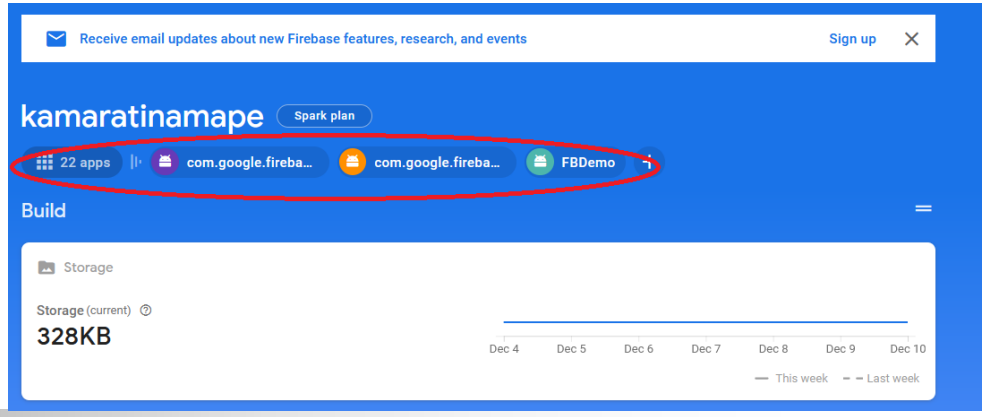
Firebase Console

prvý dotyk

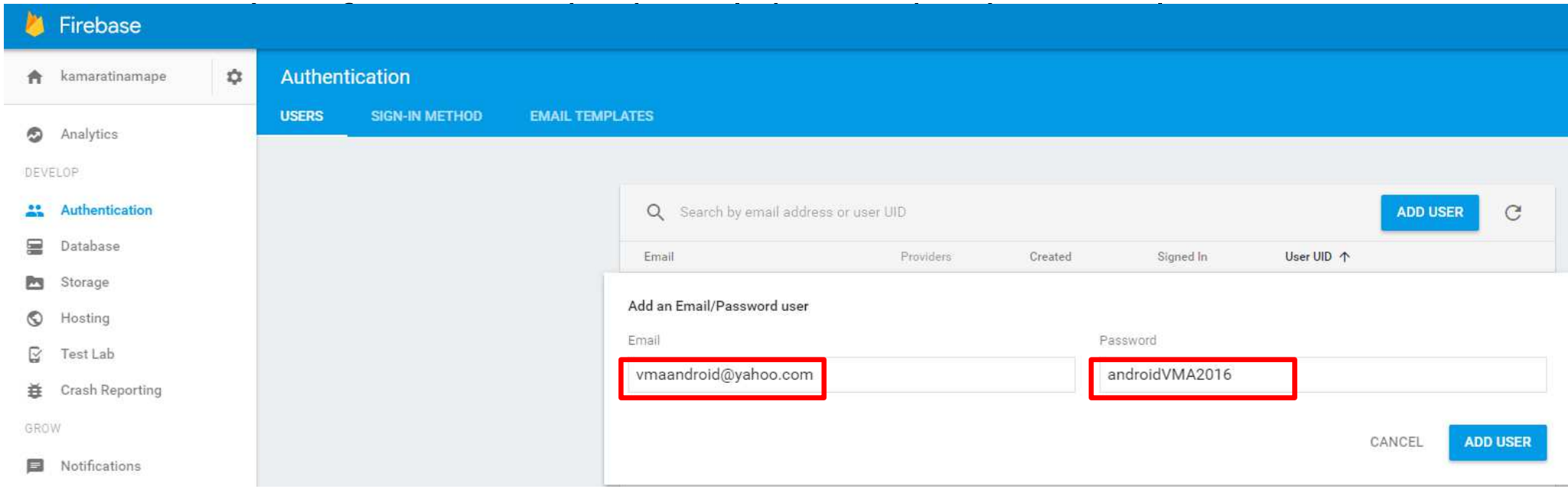


Firebase Console

v tejto prezentácii



https://console.firebase.google.com/project/kamaratinamape/database/data



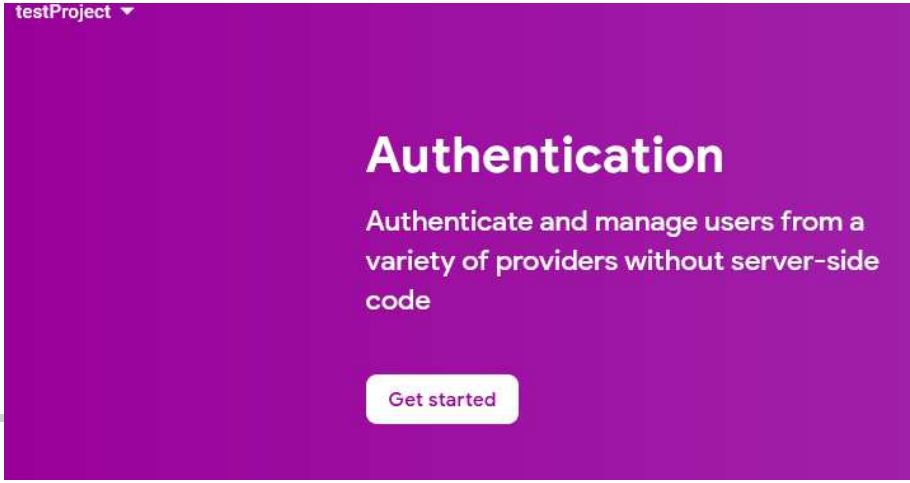
- ukážky sú robené pomocou tohoto účtu
- login a password vidíte v obrázku
- ale obávam sa, že to na inom zariadení bude chcieť potvrdenie tel.čísla ☹

https://console.firebase.google.com/project/kamaratinamape/database/data
Použite iný/váš účet
vmaandroid@yahoo.com
androidVMA2016

Authentication

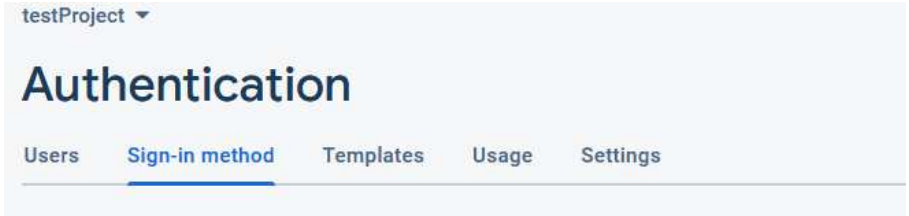
iOS Android </> C++ Unity

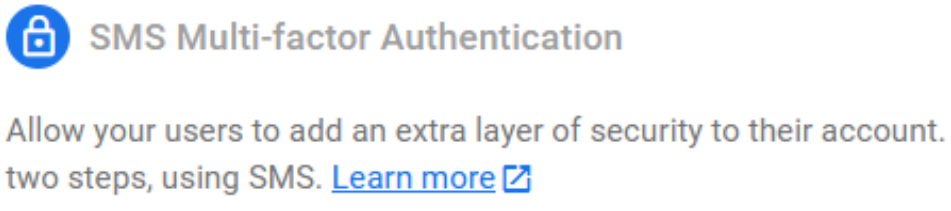
Autentifikácia

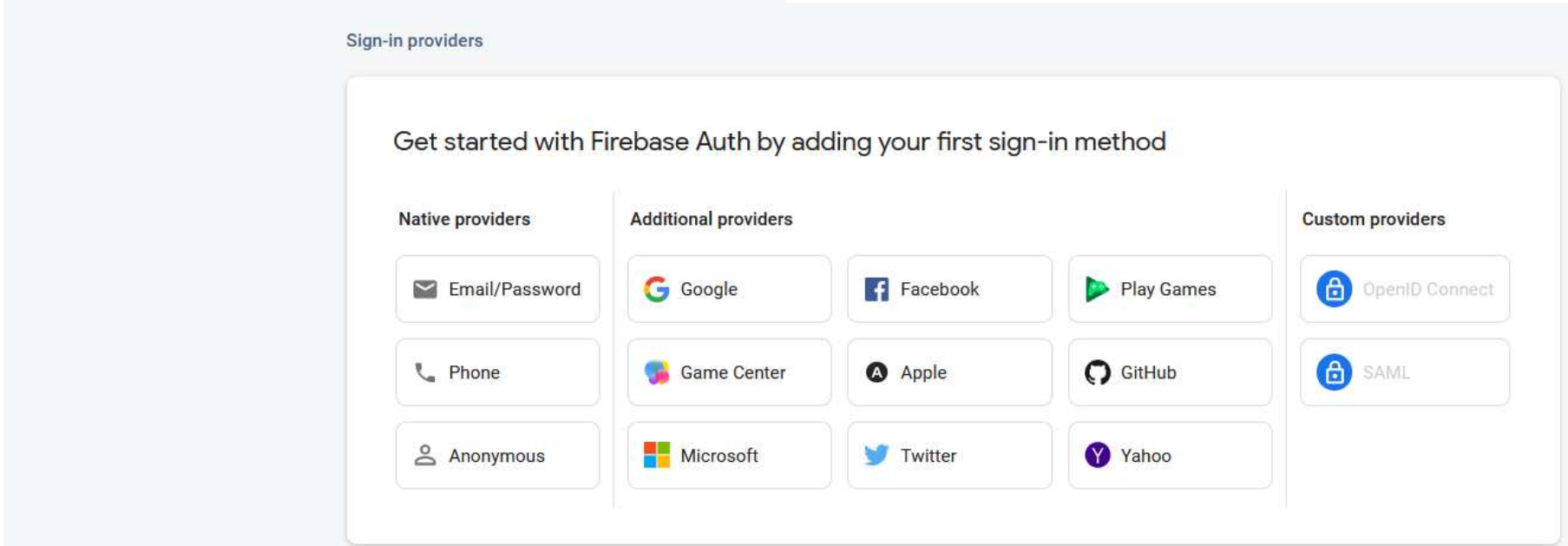


Sign-in metódy:

- vlastná autentifikácia cez email/password
  - môžete definovať viacero email-účtov (signup/signin)
- cez FB/Google/Twitter/GitHub/...





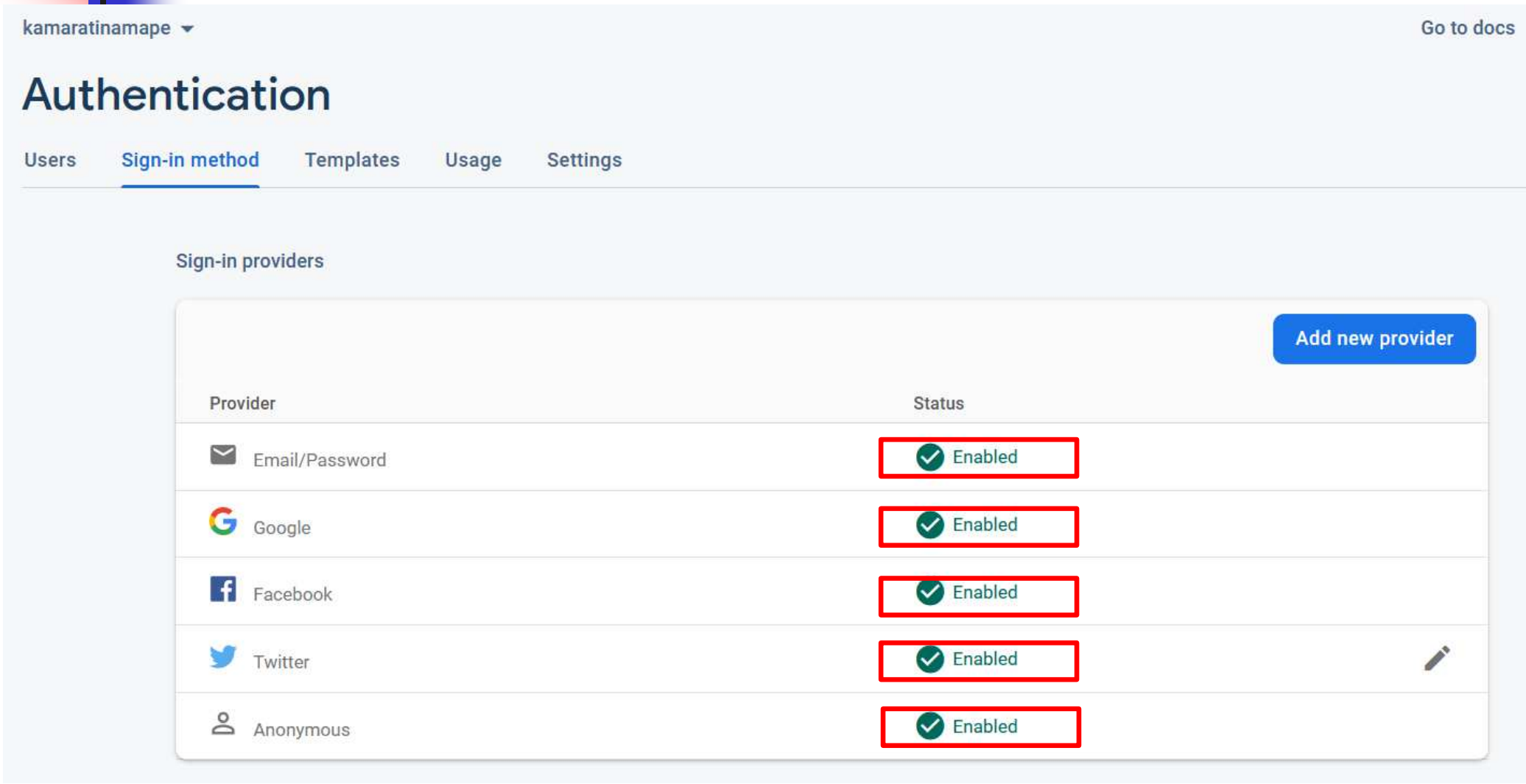


Authentication

iOS Android </> C++ Unity

Sign-in methods

https://console.firebase.google.com/u/1/project/kamaratinamape/authentication/providers

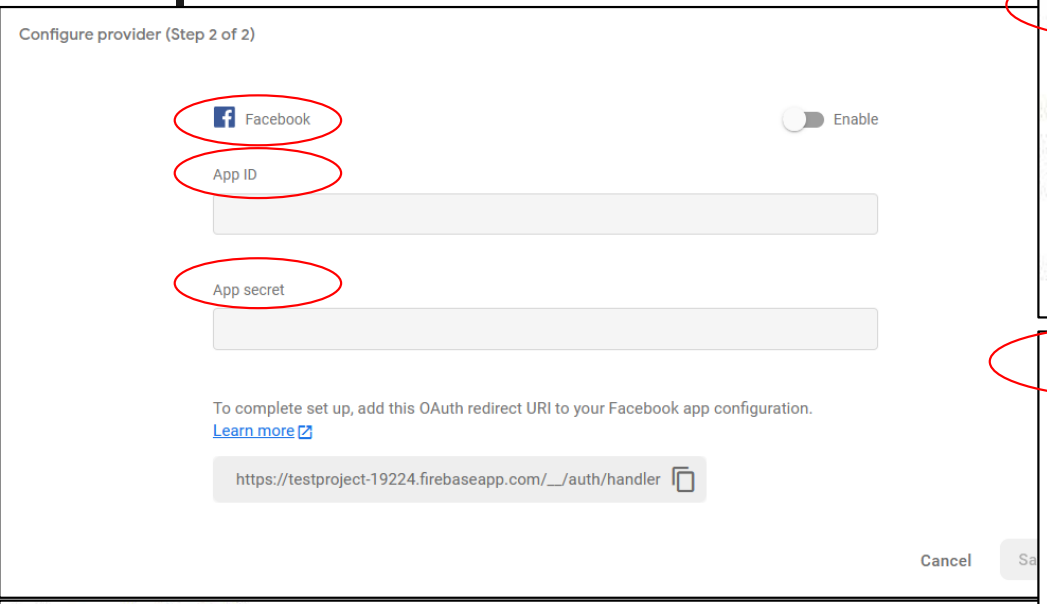


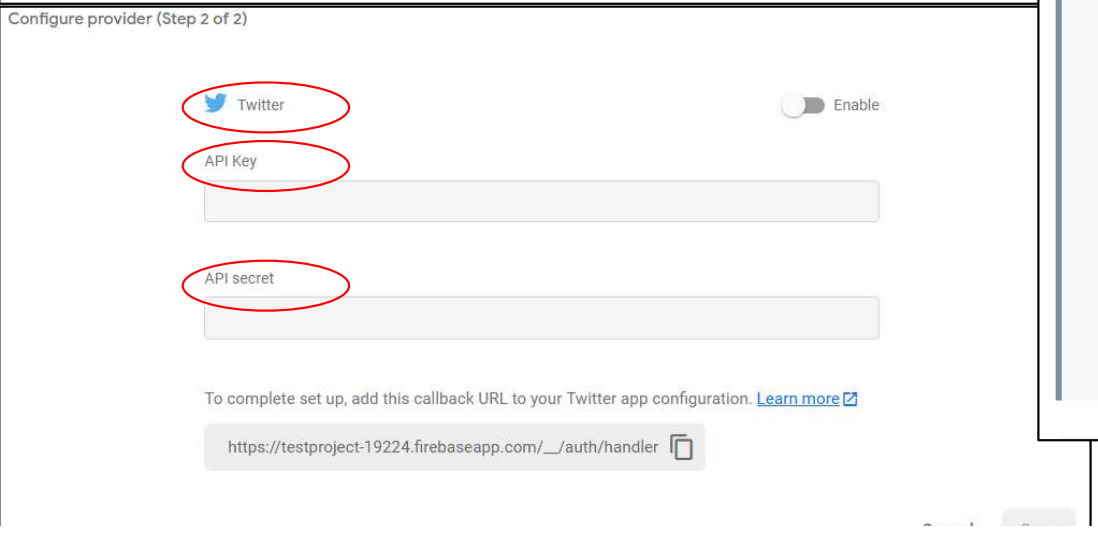
Authentication

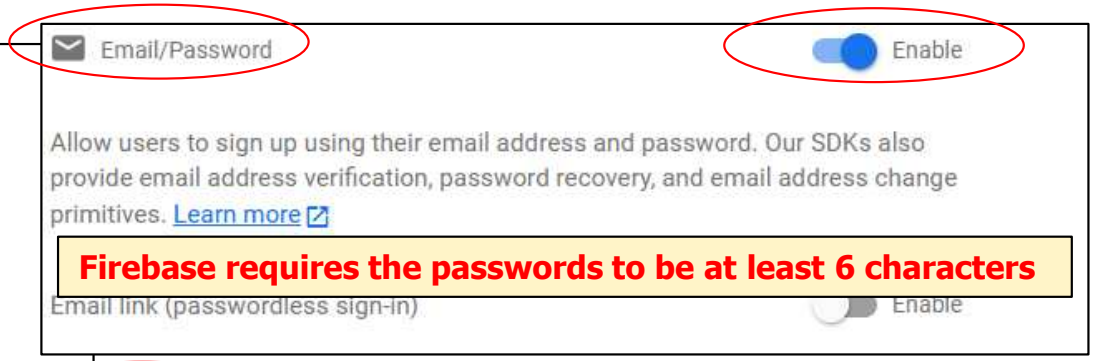
iOS Android </> C++ Unity

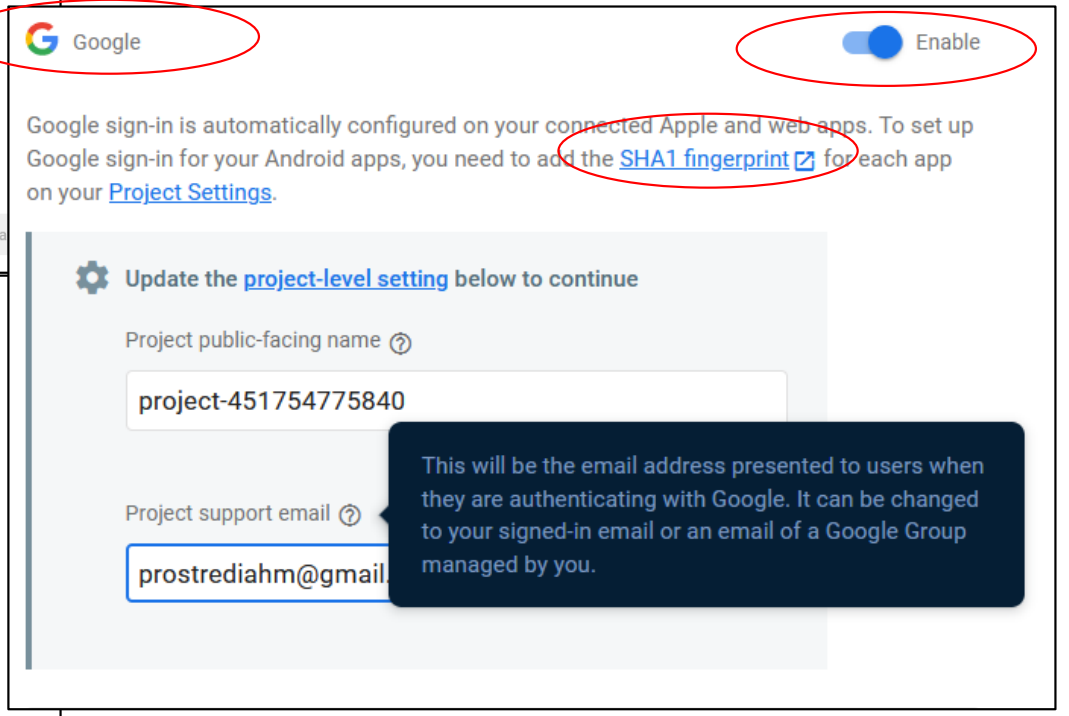
Sign-in methods

https://console.firebase.google.com/u/1/project/kamaratinamape/authentication/providers





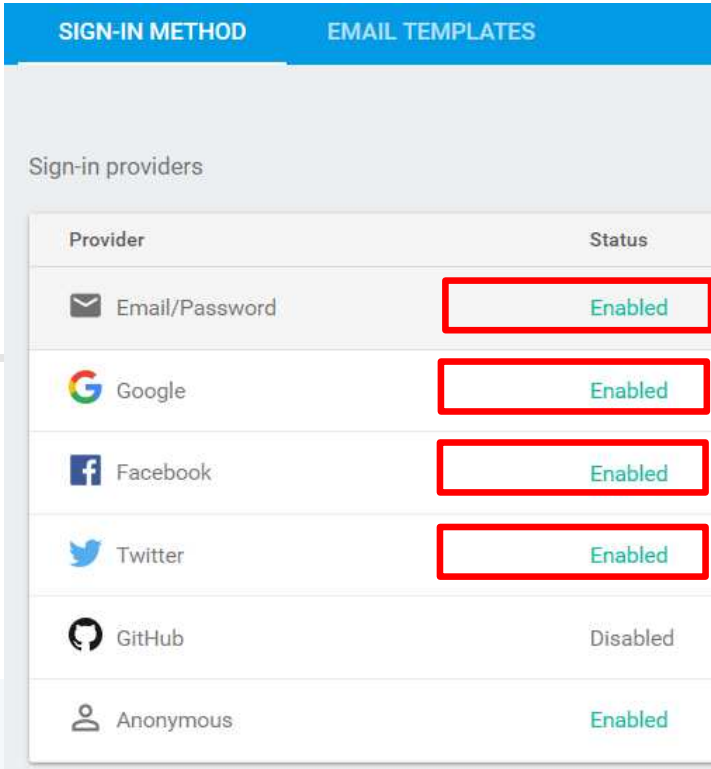


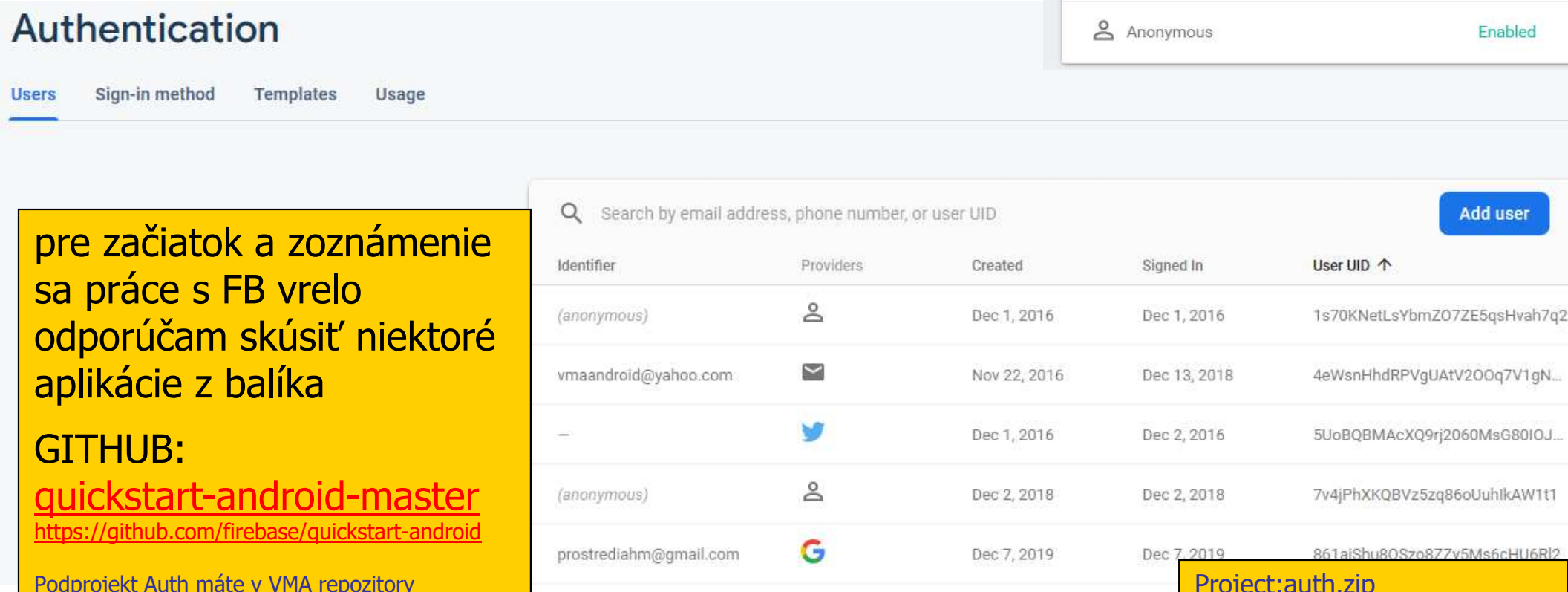


Autentifikácia

Sign-in metódy:

- vlastná autentifikácia cez email/password
  - môžete definovať viacero email-účtov
- cez FB/Google/Twitter





pre začiatok a zoznámenie sa práce s FB vrelo odporúčam skúsiť niektoré aplikácie z balíka

GITHUB:
quickstart-android-master
https://github.com/firebase/quickstart-android

Podprojekt Auth máte v VMA repozitory

Project:auth.zip
https://github.com/firebase/quickstart-android

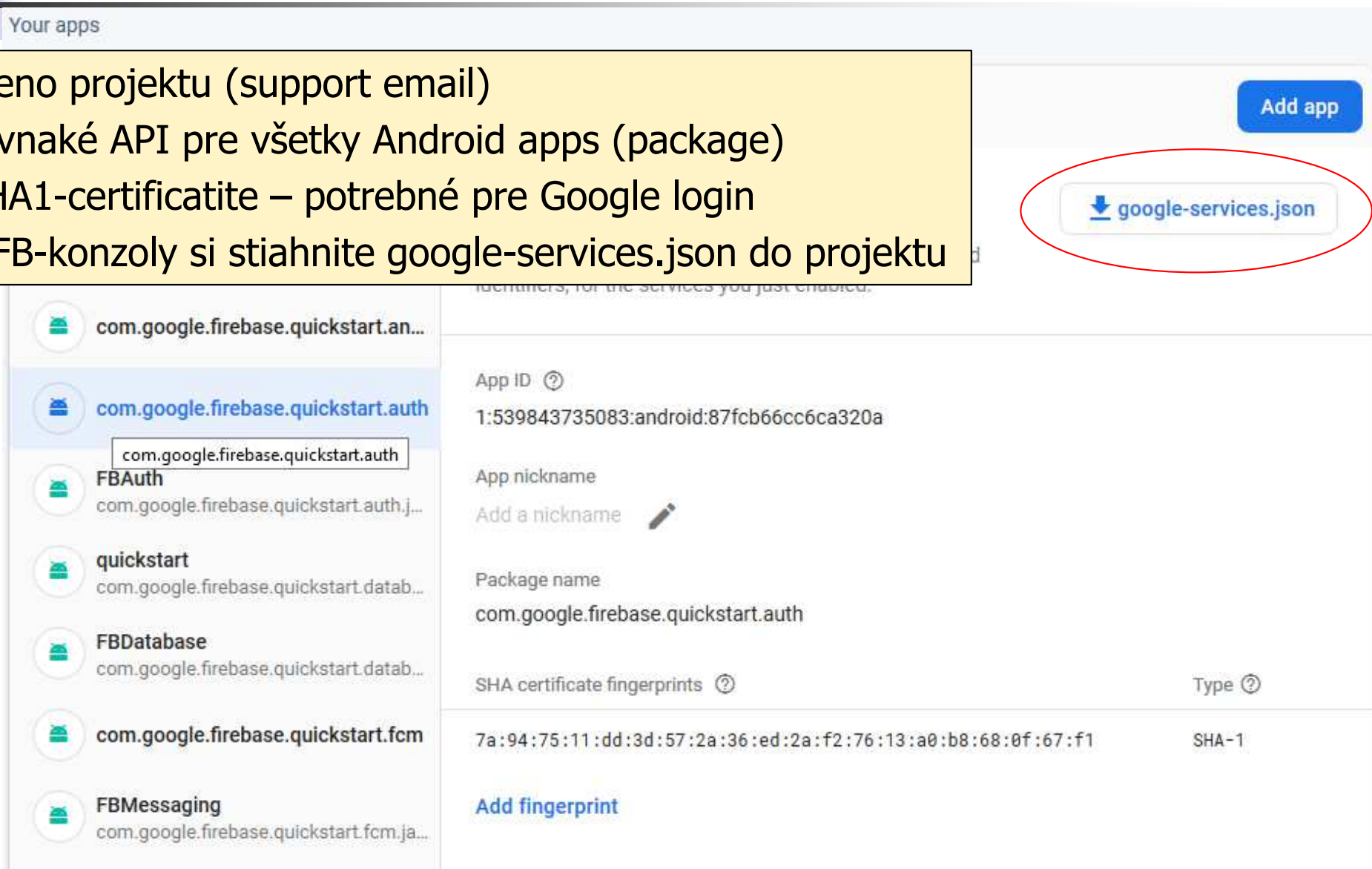
Authentication

iOS C++

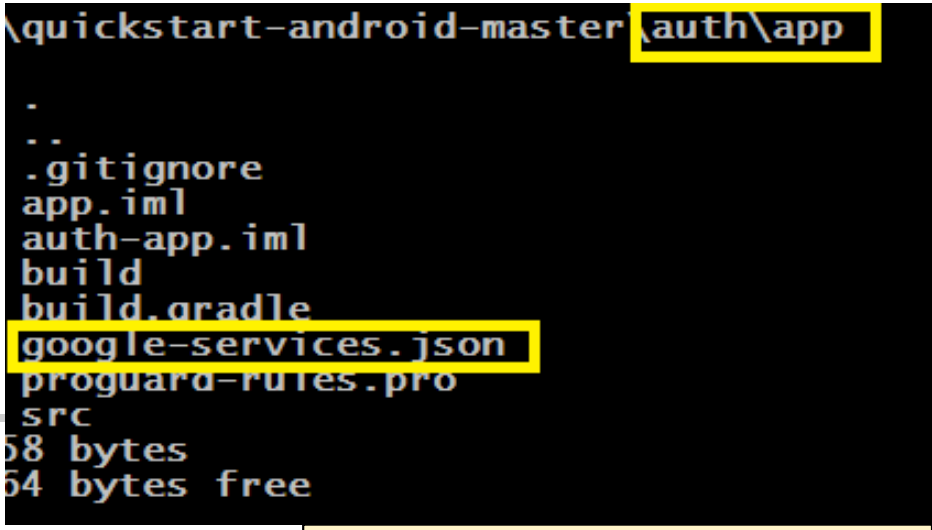
Project Setting

https://console.firebase.google.com/u/0/project/kamaratinamape/settings/general/android:com.example.firebasedemo1

- meno projektu (support email)
- rovnaké API pre všetky Android apps (package)
- SHA1-certificatite – potrebné pre Google login
- z FB-konzoly si stiahnite google-services.json do projektu



google-services .json



Tento súbor potrebujete mať v projekte, stiahnite a do pod-adresára \app


```
{"project_info": {
    "project_number": "539843735083",
    "firebase_url": "https://kamaratinamape.firebaseio.com",
    "project_id": "kamaratinamape",
    "storage_bucket": "kamaratinamape.appspot.com"
  },
  "client": [
    {"client_info": {
        "mobilesdk_app_id": "1:539843735083:android:e4c17d2977753b25",
        "android_client_info": {"package_name": "sk.uniba.fmph.dai.borovan.fbdemo"
        }
      },
      "oauth_client": [{
          "client_id": "539843735083-e4n6dg61g1npk7uka8ebf2rhcmg4t7v1.apps.googleusercontent.com",
          "client_type": 3
        }
      ],
      "api_key": [ { "current_key": "AIzaSyCbfmtnNkbnhjlqanA051uSfQl1_PTjPa8“ } ],
      "services": {
        "analytics_service": {"status": 1 },
        "appinvite_service": {"status": 1,"other_platform_oauth_client": [] },
        "ads_service": { "status": 2 }
      }
    }
  ],
  "configuration_version": "1"
}
```

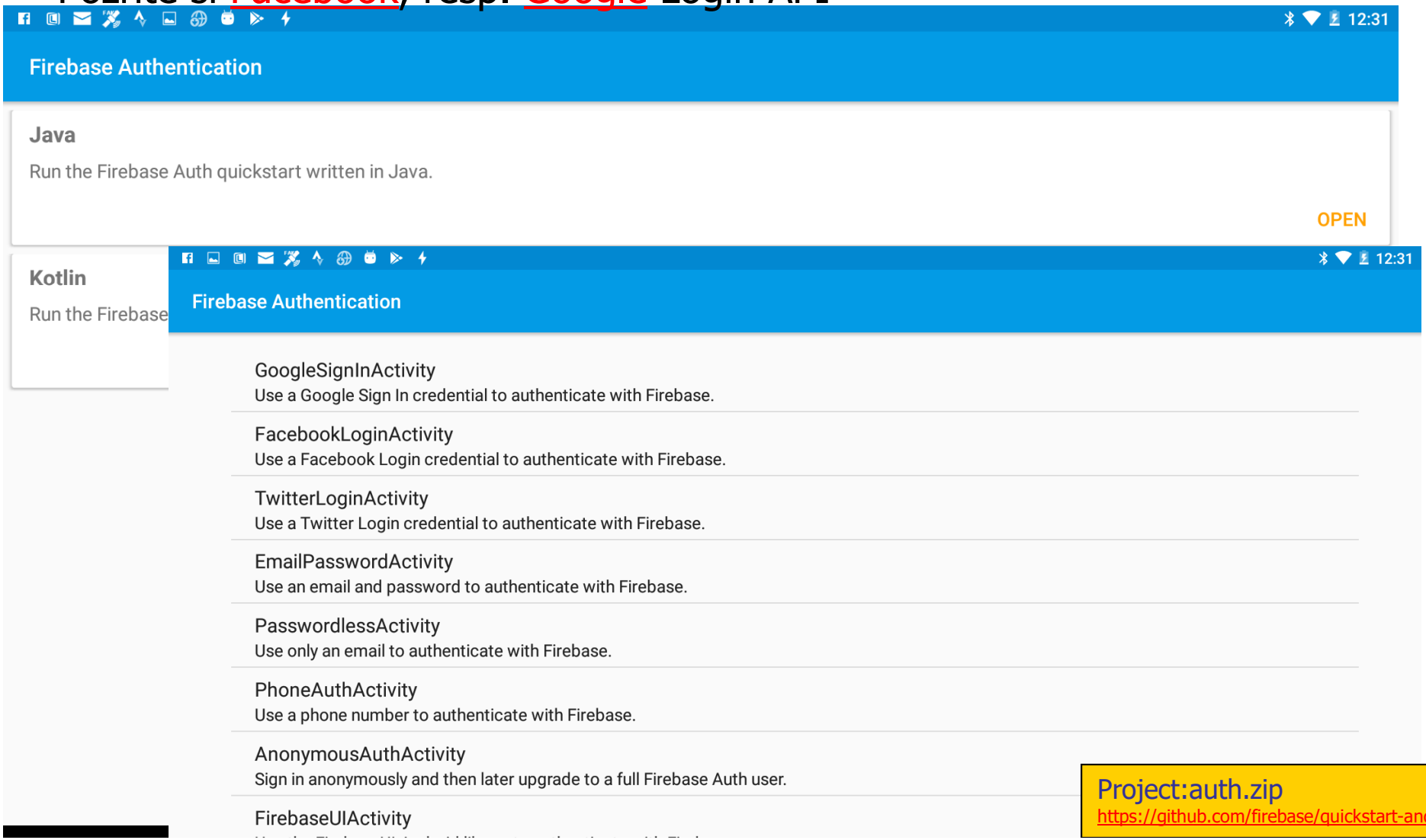

Project:auth.zip
https://github.com/firebase/quickstart-android

Authentication

iOS Android </> C++ Unity

Iná autentifikácia

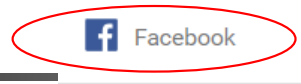
Pozrite si Facebook, resp. Google Login API

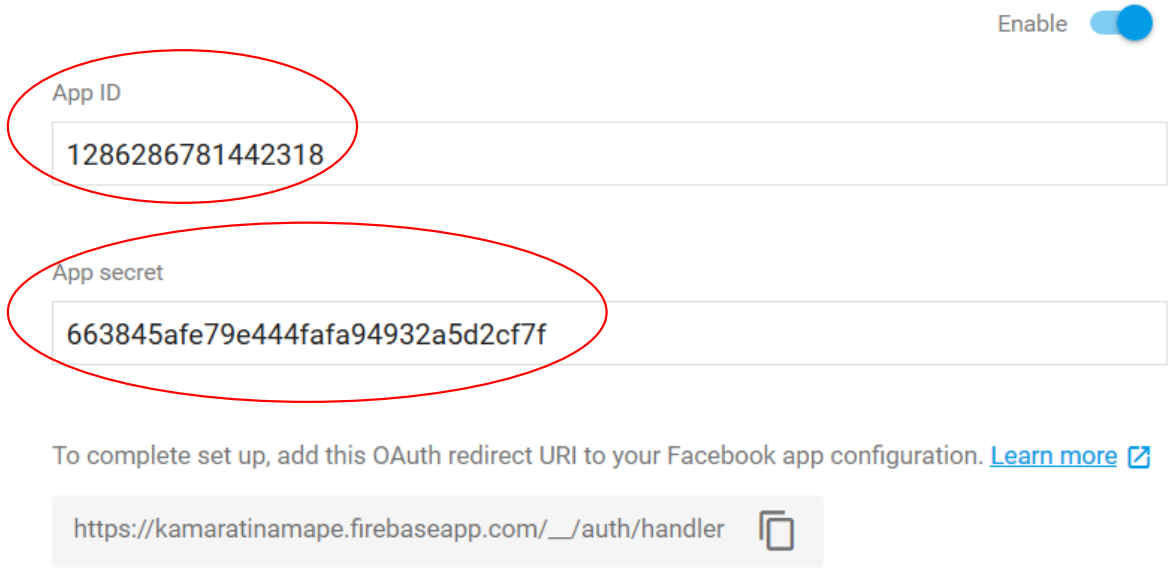


Project:auth.zip
https://github.com/firebase/quickstart-android

Autentifikácia cez FaceBook

https://firebase.google.com/docs/auth/android/facebook-login





Autentifikácia cez FB/Twitter/GitHub API:

- predpokladá, že registrujete aplikáciu na FB/Twitter/Git developerskej konzole

napr. https://developers.facebook.com/apps/1286286781442318/settings/basic/

- kde dostanete nejakú analógiu APP ID/Secret key
- tú/tie zapíšete do Firebase API vašej Firebase appky
- Firebase vám vygeneruje **google-services.json**, ktorý zakompilujete do .apk

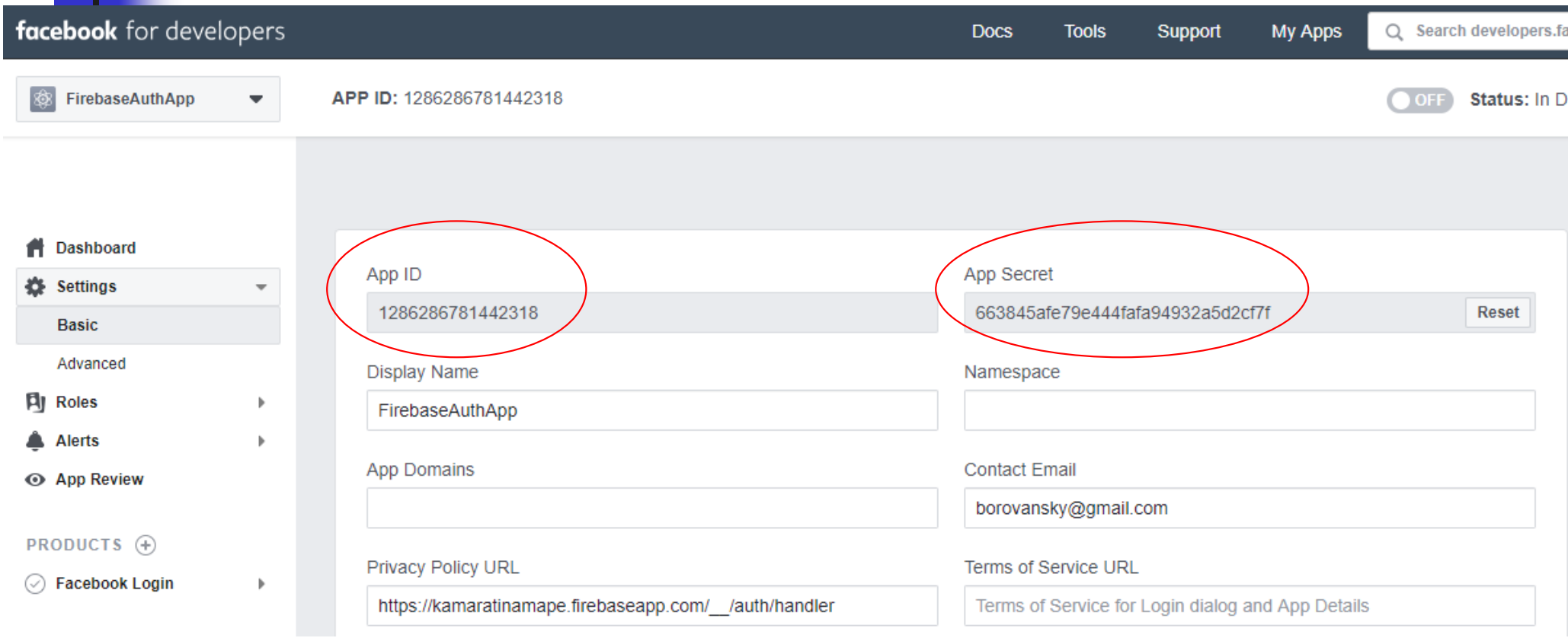
Project:auth.zip
https://github.com/firebase/quickstart-android

Authentication

iOS C++

Facebook for developers

(dev konzola od FB)

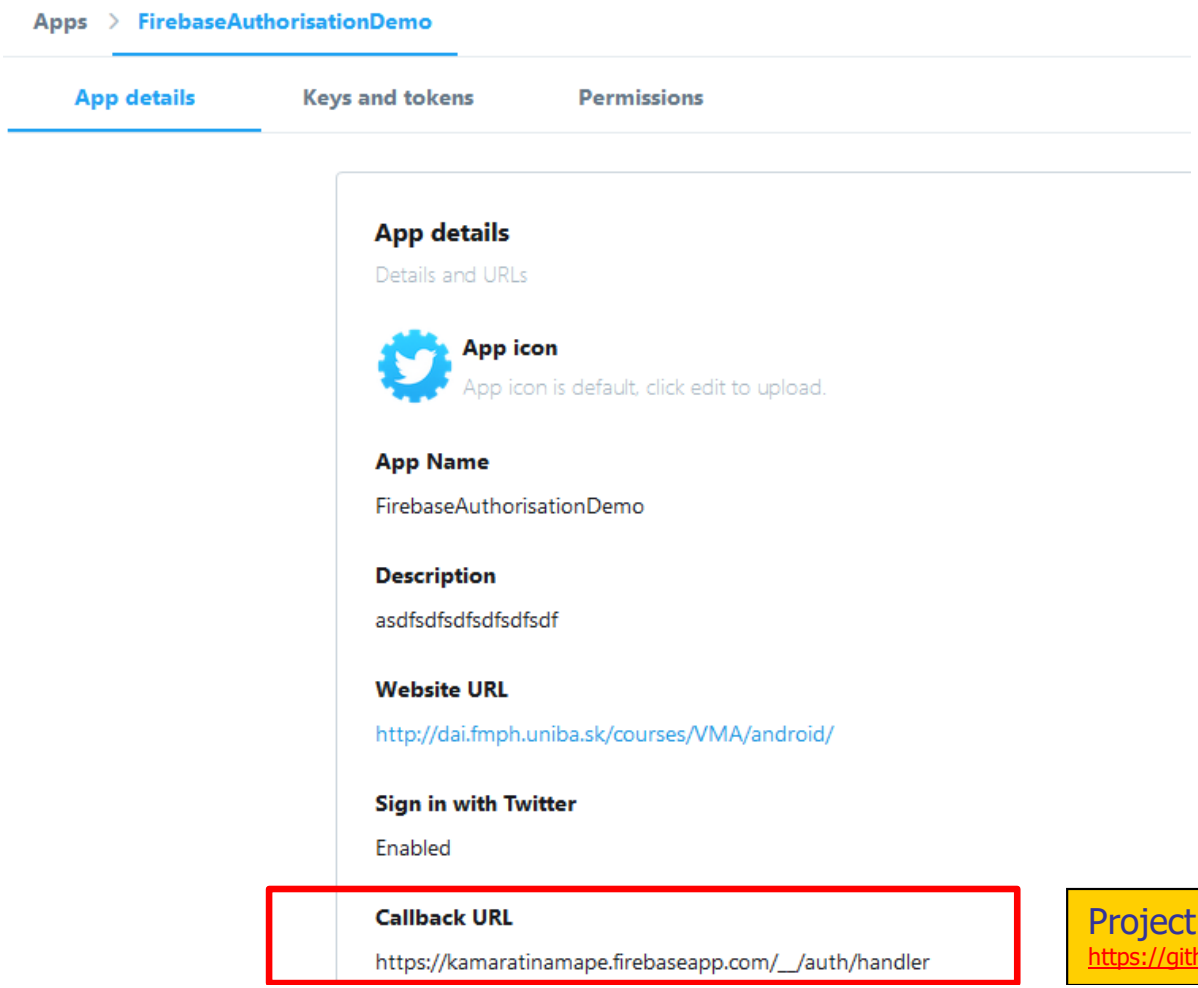


https://developers.facebook.com/apps/1286286781442318/settings/basic/

Autentifikácia cez Twitter

https://firebase.google.com/docs/auth/android/twitter-login
https://developer.twitter.com/en/apps/13160641

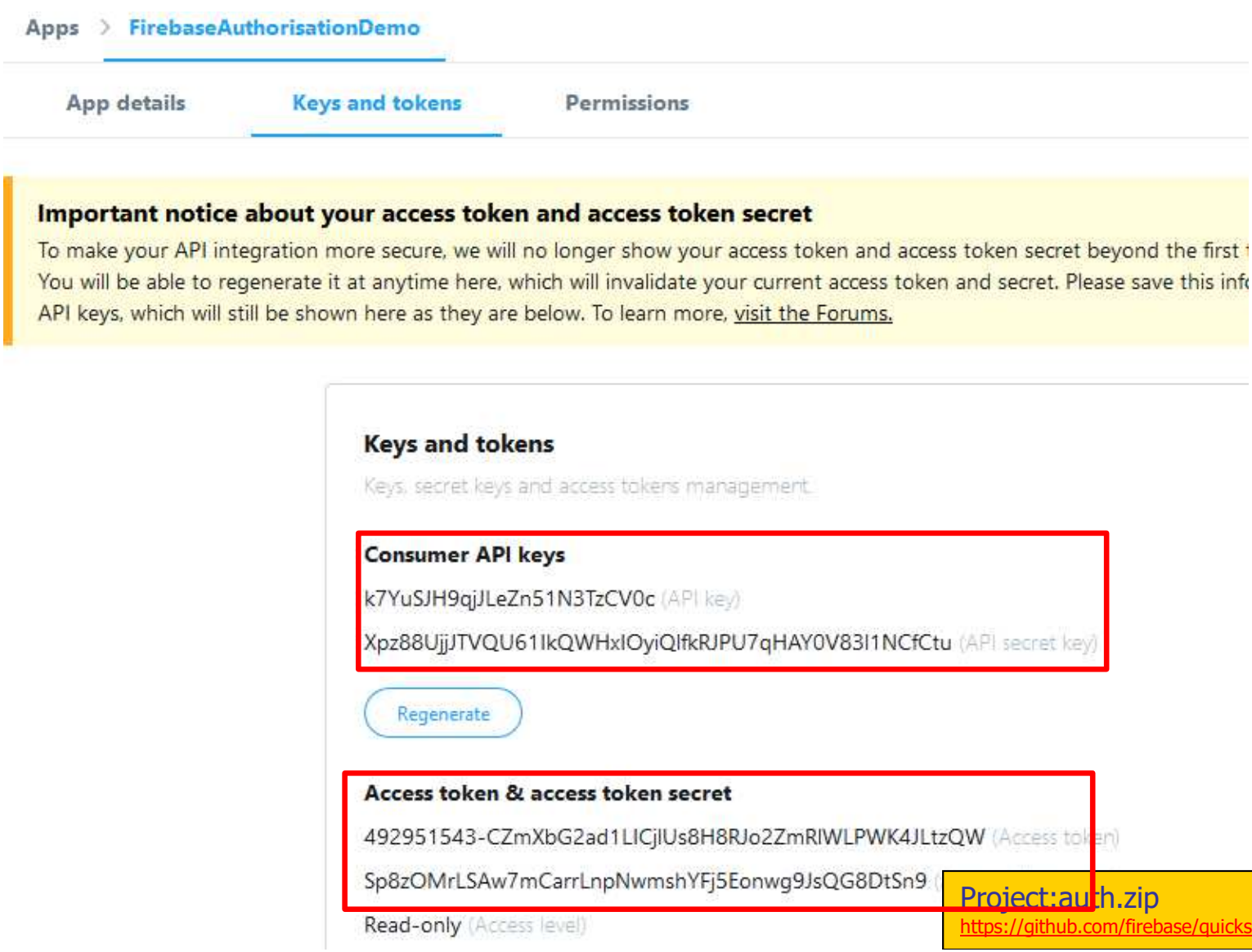
Na Twitter developerskej konzole musíte registrovať applikáciu, a získať Consumer API keys a Access token, ten prezradíte Firebase Console



Project:auth.zip
https://github.com/firebase/quickstart-android

Autentifikácia cez Twitter

https://developer.twitter.com/en/apps/13160641

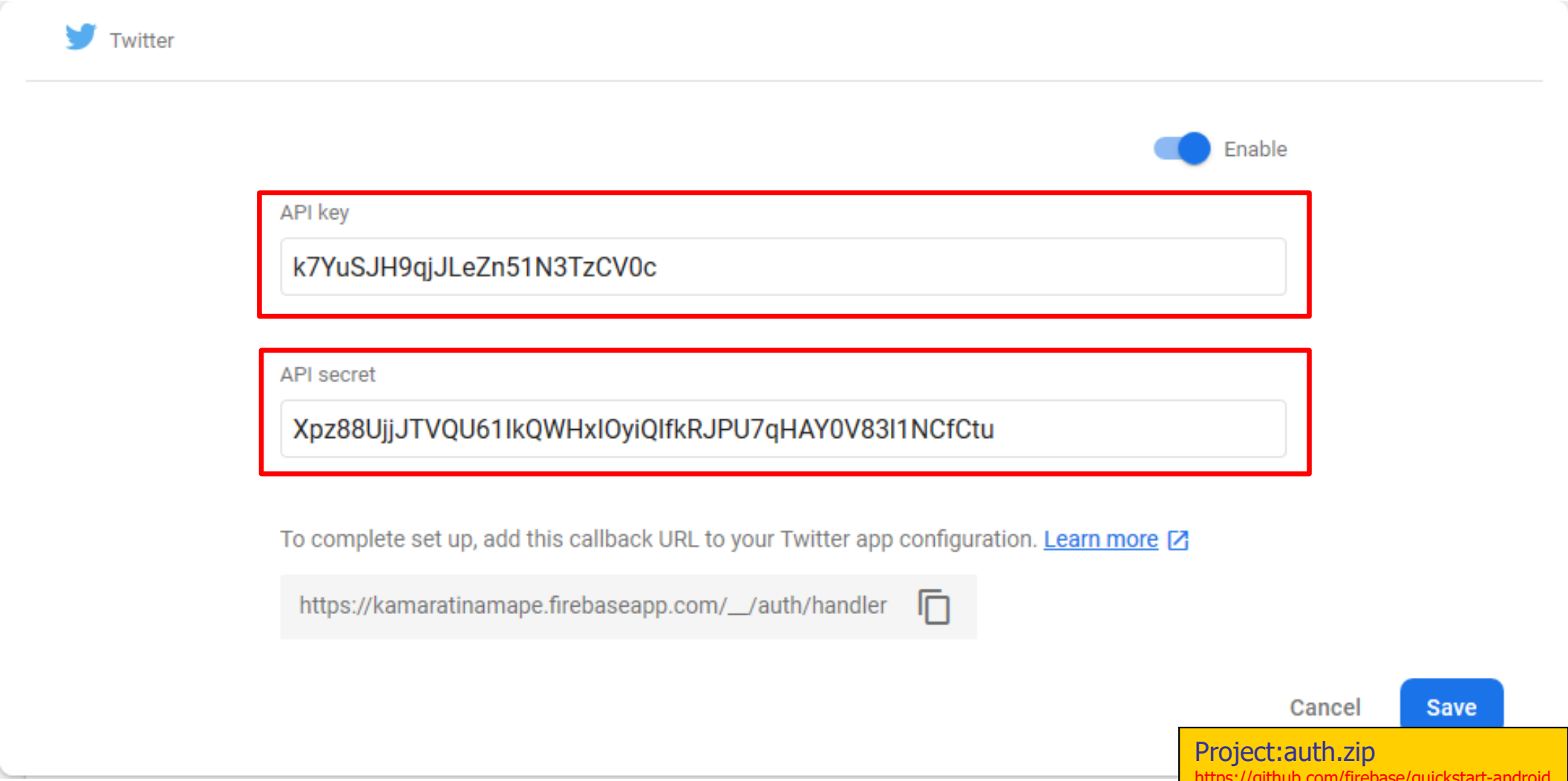


Project:auth.zip
https://github.com/firebase/quickstart-android

Autentifikácia cez Twitter

https://developer.twitter.com/en/apps/13160641

API KEY a SECRET treba vložiť do Firebase Console pre Twitter Authentification



Project:auth.zip
https://github.com/firebase/quickstart-android

Authentication

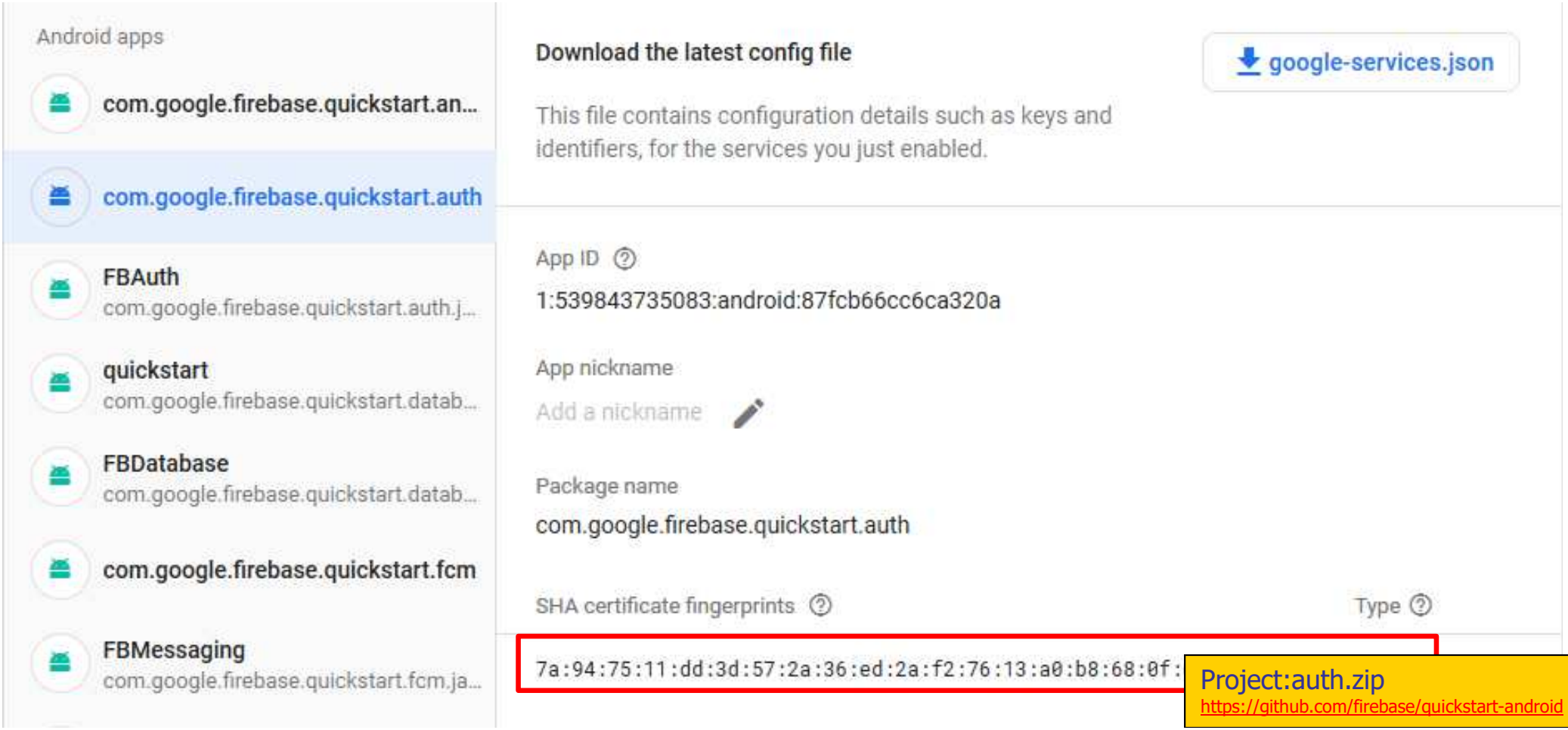
iOS Android </> C++ Unity

Autentifikácia cez Google

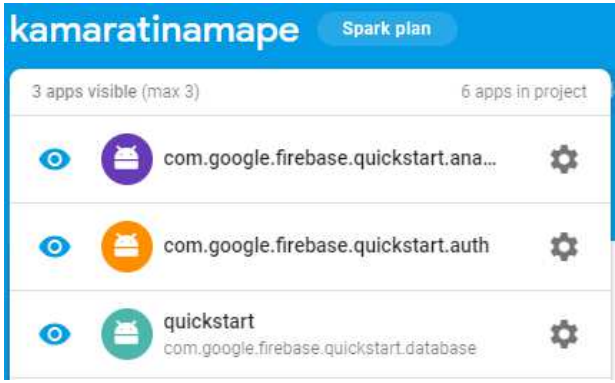
https://developers.google.com/android/guides/client-auth

Musíte do Firebase projektu/aplikácie vložiť svoj SHA1 kľúč (viac bolo minule)

Google sign-in is automatically configured on your connected iOS and web apps. To set up Google sign-in for your Android apps, you need to add the SHA1 fingerprint for each app Project Settings.

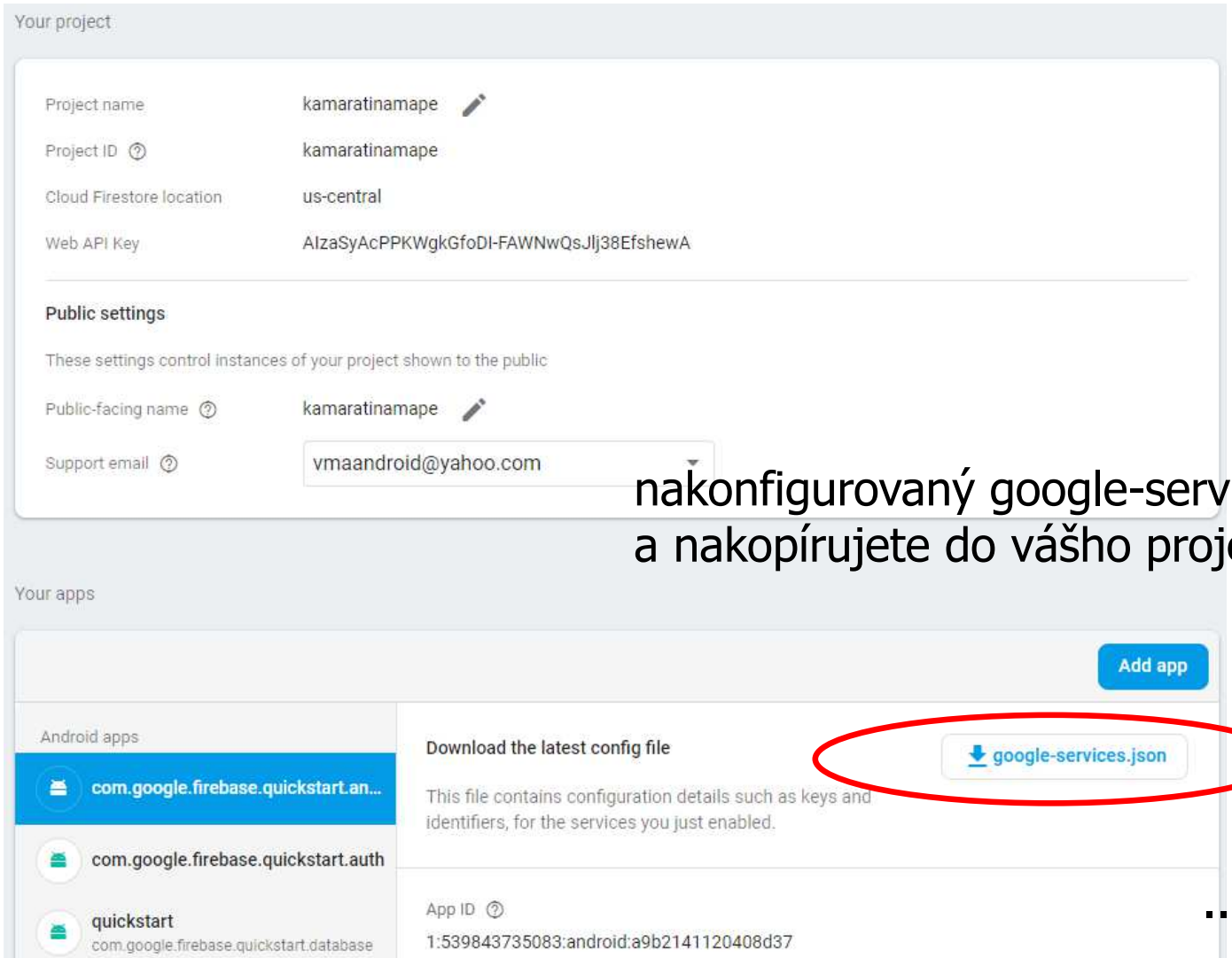


Project:auth.zip
https://github.com/firebase/quickstart-android



Registrovanie appky

v project settings registrujete aplikáciu



nakonfigurovaný google-services.json stiahnete a nakopírujete do vášho projektu, do ...app/

... a skompilujete

Registrovanie appky

1 **Register app**

This project is limited to 30 apps. You can register up to 8 more apps. Learn more

Android package name

com.example.fbdemo1

App nickname (optional)

My Android App

Debug signing certificate SHA-1 (optional)

75:11:dd:3d:57:2a:36:ed:2a:f2:76:13:a0:b8:68:0f:67:f1

Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Register app

Register app

Android package name: com.example.fbdemo1

2 **Download and then add config file**

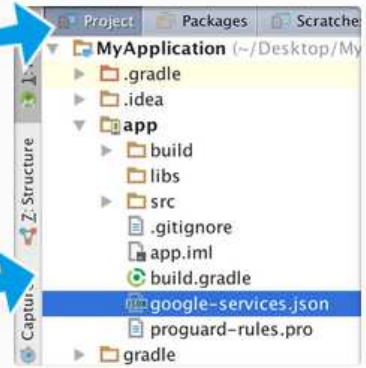
Instructions for Android Studio below | Unity C++

Download google-services.json

Switch to the **Project** view in Android Studio to see your project root directory.

Move your downloaded `google-services.json` file into your module (app-level) root directory.

google-services.json



Next

Registrovanie appky

1 **Register app**

This project is limited to 30 apps. You can register up to 8 more apps. Learn more

Android package name

com.example.fbdemo1

App nickname (optional)

My Android App

Debug signing certificate SHA-1 (optional)

75:11:dd:3d:57:2a:36:ed:2a:f2:76:13:a0:b8:68:0f:67:f1

Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Register app

Register app

Android package name: com.example.fbdemo1

2 **Download and then add config file**

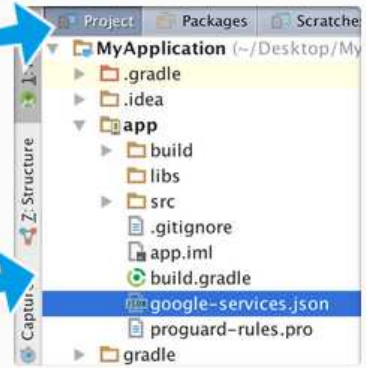
Instructions for Android Studio below | Unity C++

Download google-services.json

Switch to the **Project** view in Android Studio to see your project root directory.

Move your downloaded `google-services.json` file into your module (app-level) root directory.

google-services.json



Next

Registrovanie appky

3 Add Firebase SDK

Instructions for Gradle | Unity C++

1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plugin.

   Add the plugin as a buildscript dependency to your **project-level** `build.gradle` file:

   Root-level (project-level) Gradle file (`<project>/build.gradle`):

```
buildscript {
  repositories {
    // Make sure that you have the following two repositories
    google()  // Google's Maven repository
    mavenCentral()  // Maven Central repository
  }
  dependencies {
    ...
    // Add the dependency for the Google services Gradle plugin
    classpath 'com.google.gms:google-services:4.3.13'
  }
}

allprojects {
  ...
  repositories {
    // Make sure that you have the following two repositories
    google()  // Google's Maven repository
    mavenCentral()  // Maven Central repository
  }
}
```

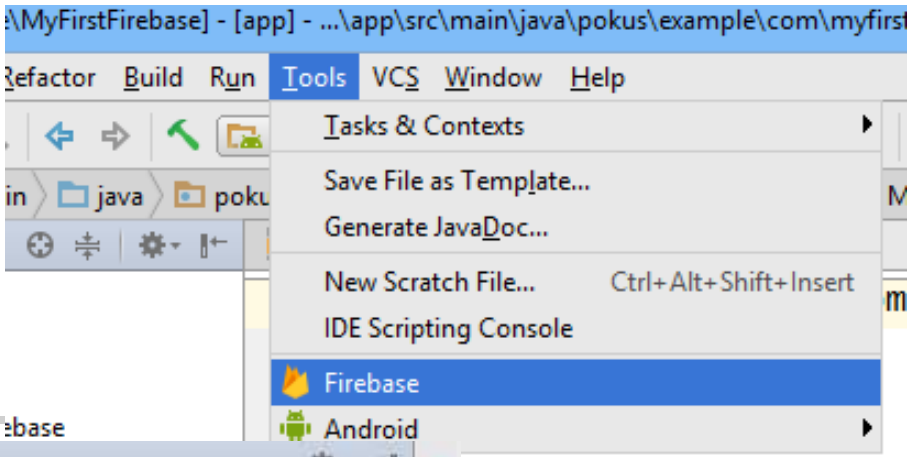
2. Then, in your **module (app-level)** `build.gradle` file, add both the `google-services` plugin and any Firebase SDKs that you want to use in your app:

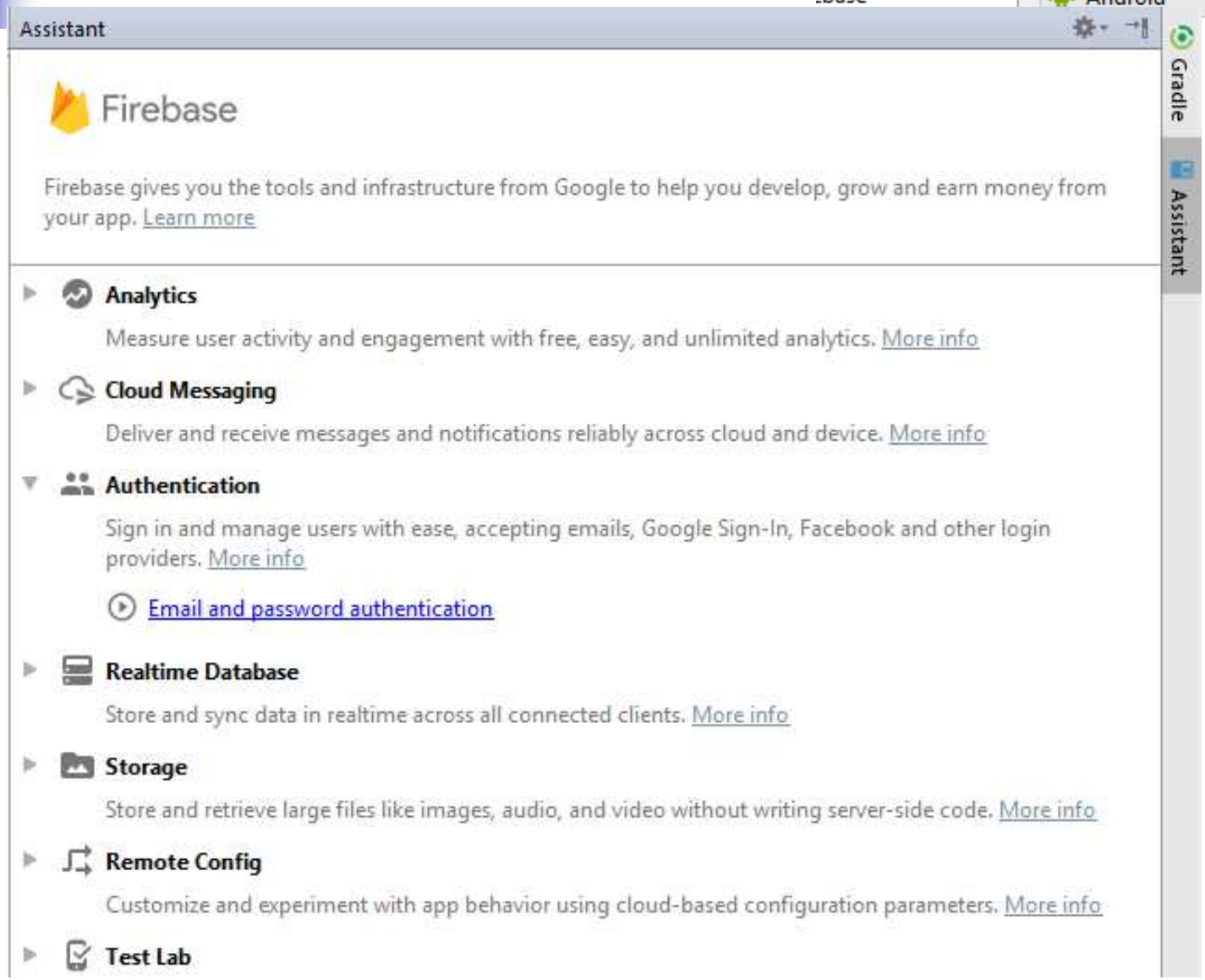
   ○ Java ◉ Kotlin

   Module (app-level) Gradle file (`<project>/<app-module>/build.gradle`):

```
plugins {
  id 'com.android.application'
  // Add the Google services Gradle plugin
  id 'com.google.gms.google-services'
  ...
}

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:31.1.1')

  // TODO: Add the dependencies for Firebase products you want to use
  // When using the BoM, don't specify versions in Firebase dependencies
  implementation 'com.google.firebase:firebase-analytics-ktx'

  // Add the dependencies for any other desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

Firebase podpora v AS



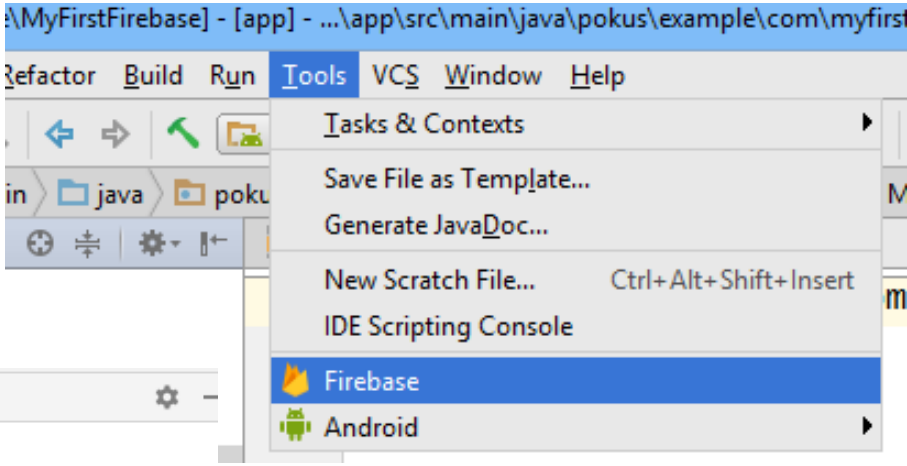


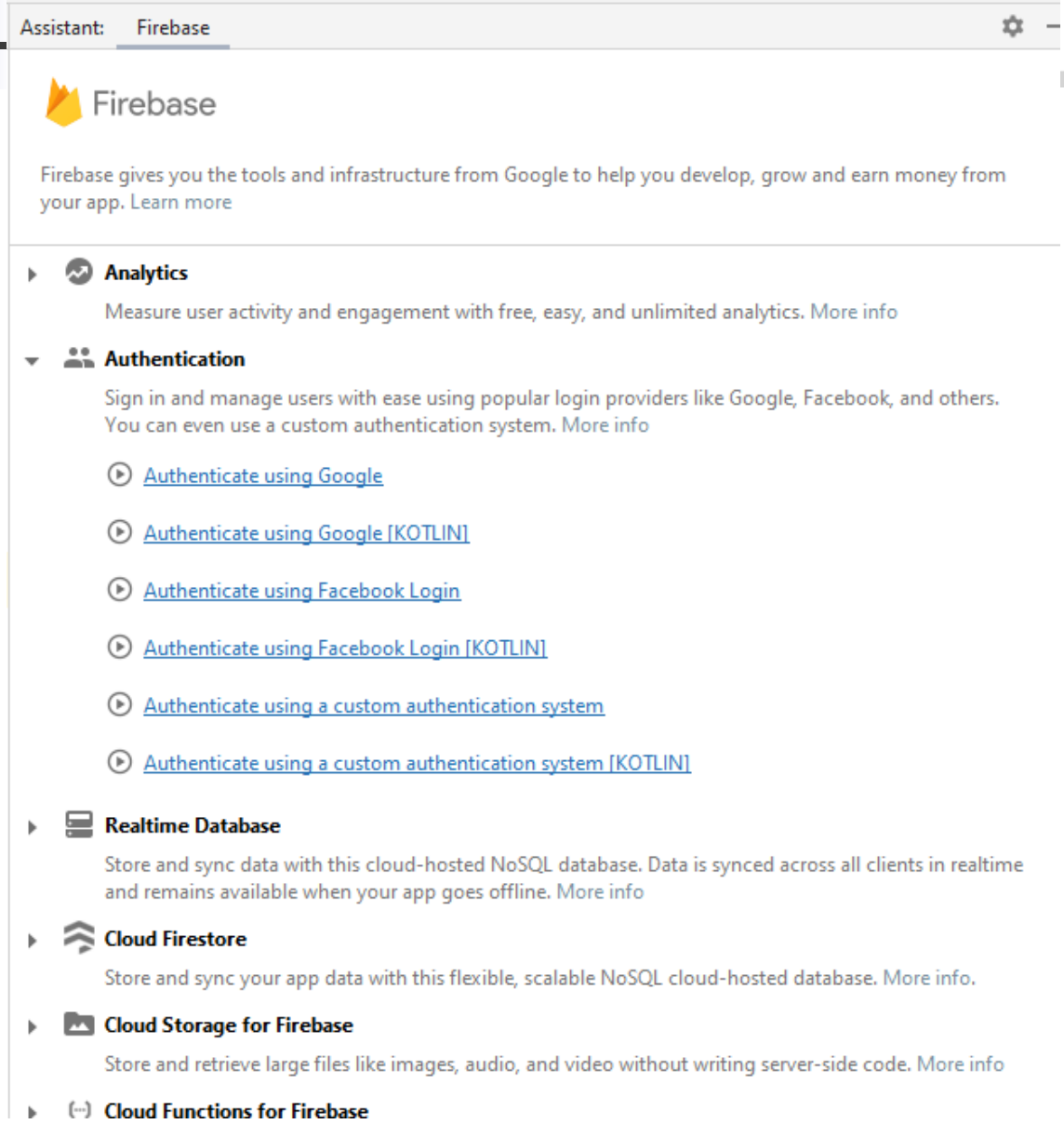
Project:FBDemo1

Authentication

iOS Android </> C++

Firebase v AS





Project:FBDemo1

Authentication

iOS Android </> C++ Unity

Autentifikačný kit v AS

Assistant

Firebase › Authentication

Email and password authentication

You can use Firebase Authentication to let your users sign in with their email addresses and passwords, and to manage your app's password-based accounts. This tutorial helps you set up an email and password system and then access information about the user.

Launch in browser

1 **Connect your app to Firebase**

Connect to Firebase

2 **Add Firebase Authentication to your app**

Add Firebase Authentication to your app

To use an authentication provider, you need to enable it in the Firebase console . Go to the Sign-in Method page in the Firebase Authentication section to enable Email/Password sign-in and any other identity providers you want for your app.

3 **Listen for auth state**

Declare the FirebaseAuth and AuthStateListener objects.

```
private FirebaseAuth mAuth;
```

```
private FirebaseAuth.AuthStateListener mAuthListener;
```

In the onCreate() method, initialize the FirebaseAuth instance and the AuthStateListener method so you can track whenever the user signs in or out.

```
mAuth = FirebaseAuth.getInstance();
```

Ponúka vám to step-by-step návod na vytvorenie rôznych typov Firebase aplikácií

Add Firebase Authentication to your app

Performing this action will make the following changes to your project.

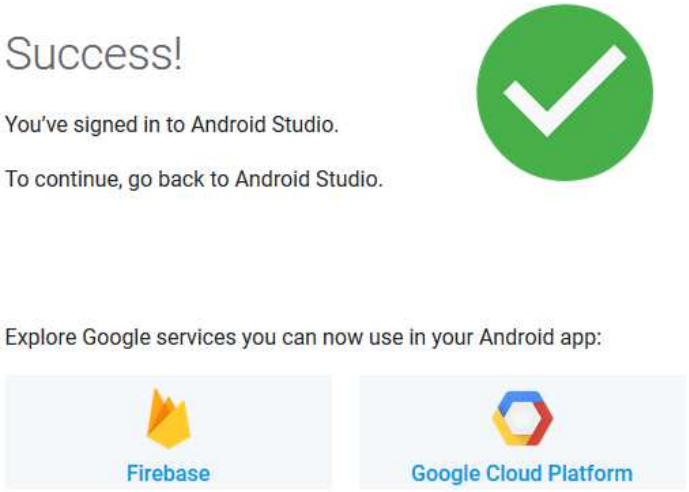
```
build.gradle (project-level)

    Add Firebase Gradle buildscript dependency
        classpath 'com.google.gms:google-services:4.0.1'

app/build.gradle

    Add Firebase plugin for Gradle
        apply plugin: 'com.google.gms.google-services'

    build.gradle will include these new dependencies:
        compile 'com.google.firebase:firebase-auth:16.0.1:15.0.0'
```

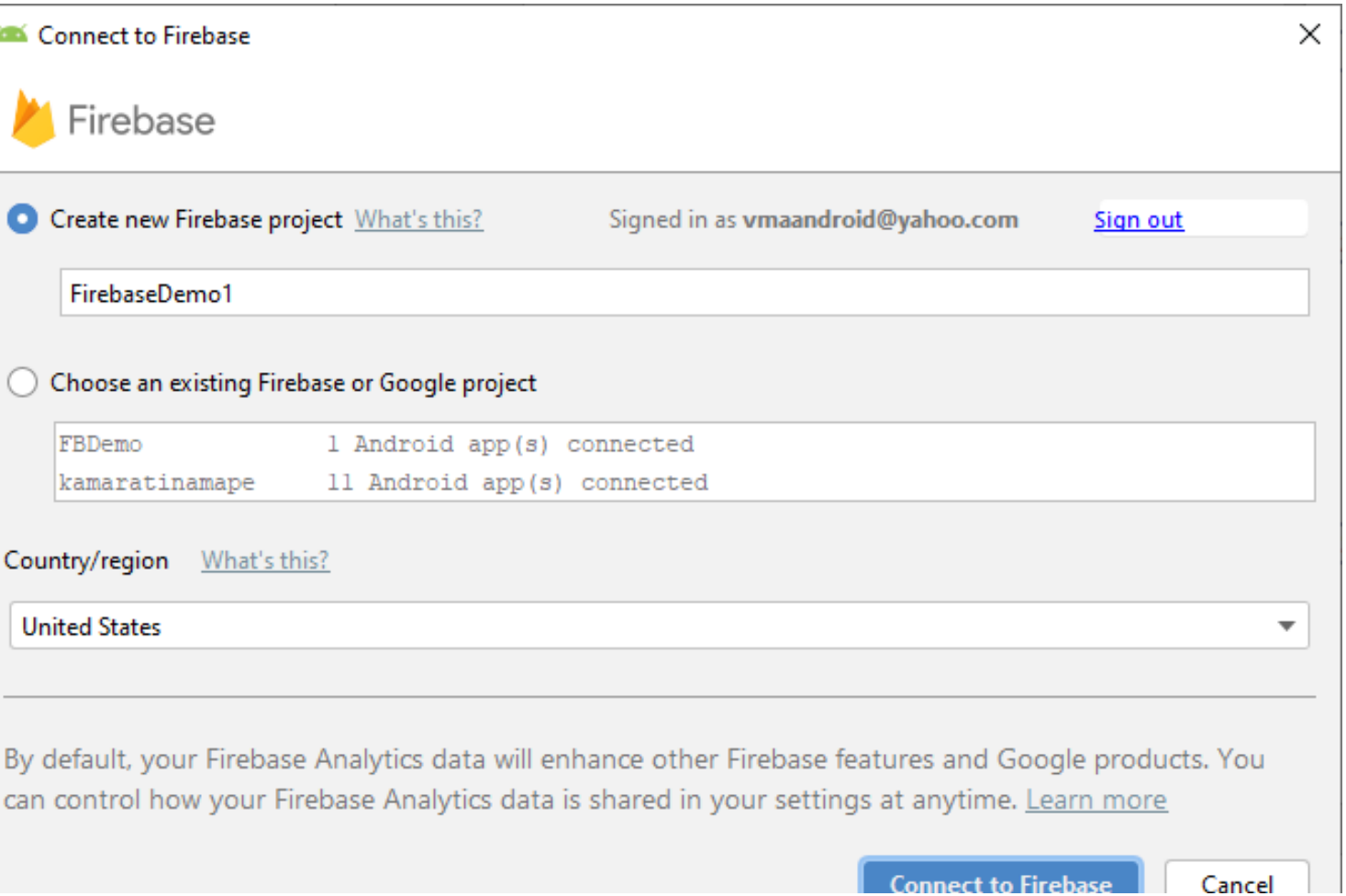
This will also enable the firebase-core library which includes Firebase Analytics. Learn more

Accept Changes Cancel

Project:FBDemo1

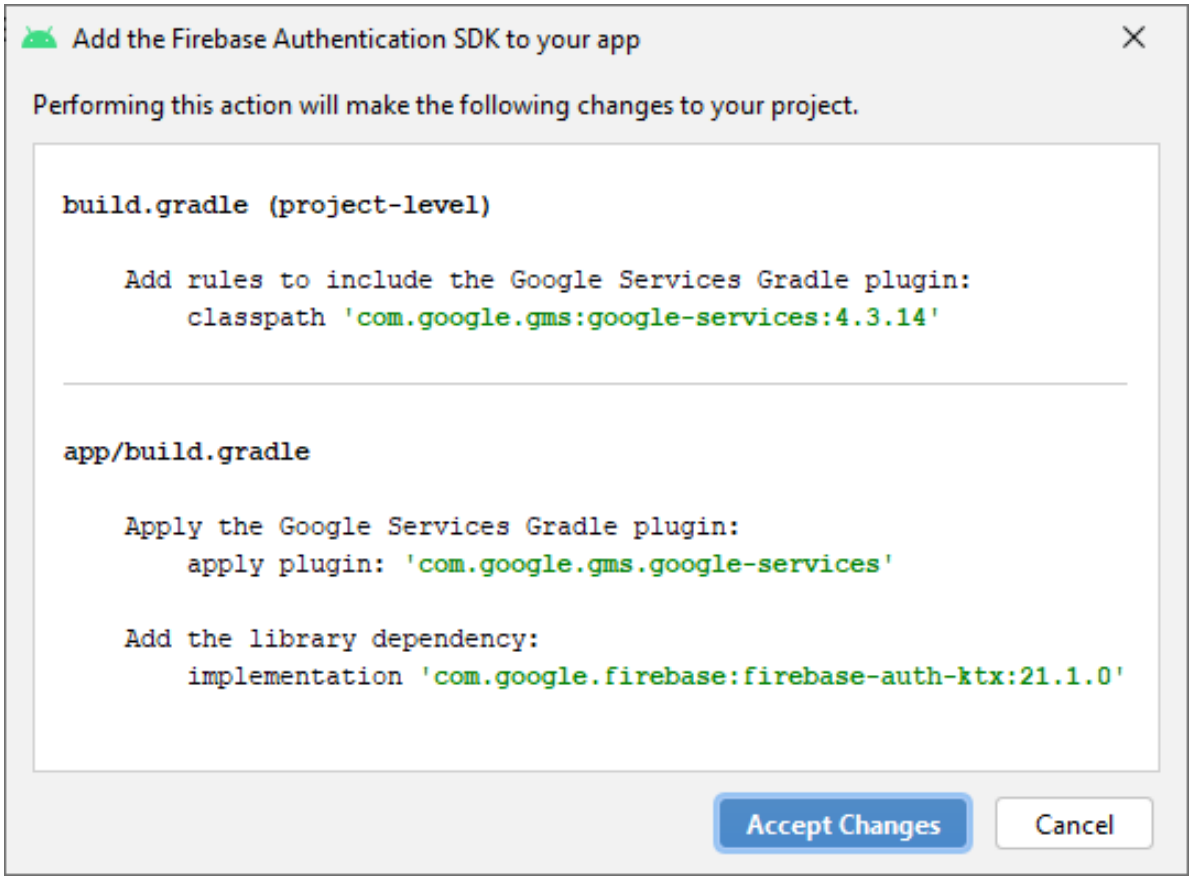
1) Connect your app to Firebase





Project:FBDemo1

2) Add Firebase Auth to your app



Project:FBDemo1

build.grade (app)

```
dependencies {
 …
    implementation 'com.google.firebase:firebase-auth-ktx:21.1.0'
}
```

```
buildscript {
    dependencies {
        classpath 'com.google.gms:google-services:4.3.14'
    }
}
plugins {
    id 'com.android.application' version '7.3.1' apply false
    id 'com.android.library' version '7.3.1' apply false
    id 'org.jetbrains.kotlin.android' version '1.7.20' apply false
}
```

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    id 'com.google.gms.google-services'
}
```

Project:FBDemo1

Project:FirebaseDemo1

Firebase Email Authentification

sign-up new user

```
lateinit var mAuth: FirebaseAuth
. . .
mAuth.createUserWithEmailAndPassword(email, password)
  .addOnCompleteListener(this, object:
     OnCompleteListener<AuthResult> {
        override fun onComplete(task: Task<AuthResult>) {
            if (task.isSuccessful()) { // Sign in success, update
               Log.d(TAG, "createUserWithEmail:success")
               val user = mAuth.currentUser
            } else { // If sign in fails,
               Log.w(TAG, "createUserWithEmail:failure",
                     task.getException())
            }
        }
    })
```

Project:FBDemo1 Project:FirebaseDemo1

Firebase Email Authentification

sign-in an existing user

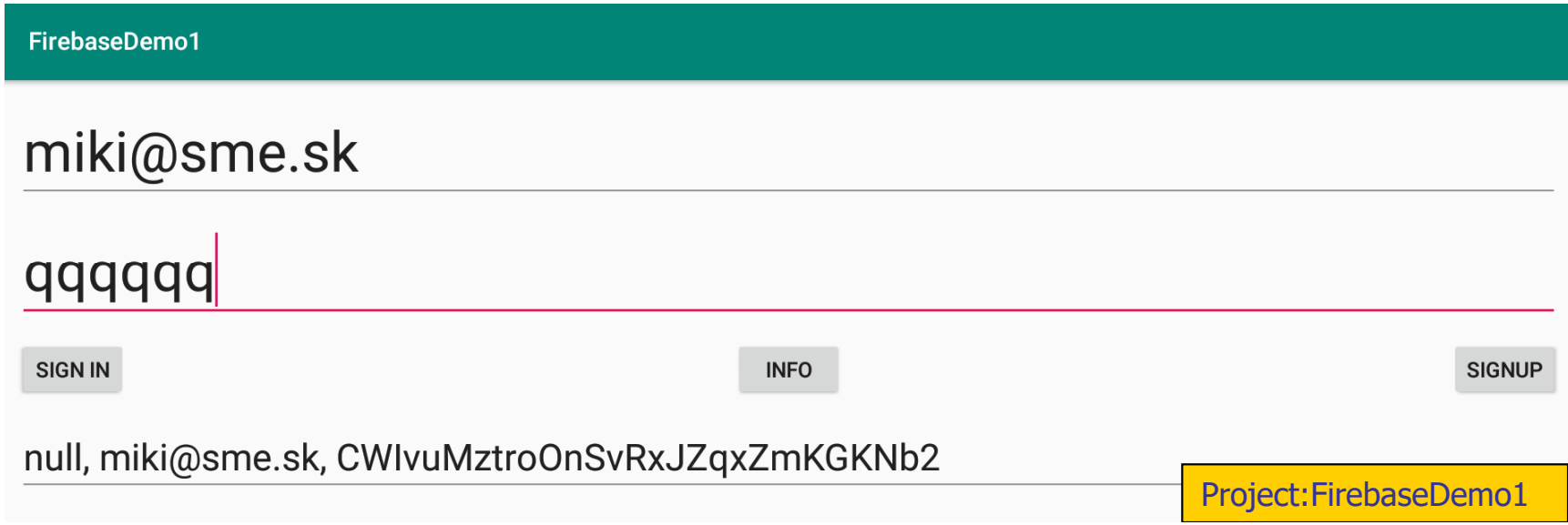
```
lateinit var mAuth: FirebaseAuth
. . .
mAuth.signInWithEmailAndPassword(email, password)
    .addOnCompleteListener(this) { task ->
        if (task.isSuccessful) { // Sign in success
            Log.d(TAG, "signInWithEmail:success")
            val user = mAuth.currentUser
        } else { // If sign in fails
            Log.w(TAG, "signInWithEmail:failure", task.exception)
        }
}
```

Project:FirebaseDemo1

Firebase Email Authentification

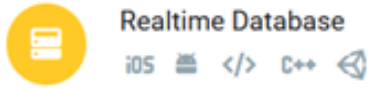
get user info

```
val user = FirebaseAuth.getInstance().currentUser
if (user != null) { // Name, email address, and profile photo Url
    val name = user.displayName
    val email = user.email
    val photoUrl: Uri? = user.photoUrl
    val emailVerified = user.isEmailVerified
    val uid = user.uid
    infoTV.setText("$name, $email, $uid")
}
```



Project:FirebaseDemo1

Firebase ako databáza



- noSQL databázy
- **Realtime Database** (efektívna pre mobilné app, synchronizácia)
  - ukladanie a synchronizácia v reálnom čase so všetkými pripojenými klientami
  - všetky dáta sú jeden veľký json dátový strom (JSON Tree)
  - existuje dávno, je stabilná, **regionálne** má veľmi slušnú latenciu

- **Cloud Firestore** (novinka)
  - Realtime updates, powerful queries, automatic scaling
  - dáta sú v kolekciách, hierarchicky organizované, sub-kolekcie, ...
  - novinka, beta r.2018, scalability, prepojenie cez viaceré dátové centrá

Obe podporujú offline support pre mobilných klientov (Android, iOS, web)
- zmeny počas off-line sa ukladajú do cache a synchronizujú, keď on-line

Realtime Database

iOS

FireStore Database

nosql databáza - rules

- Default, no access, only FB console

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if false;
    }
  }
}
```

- Public, anyone can...

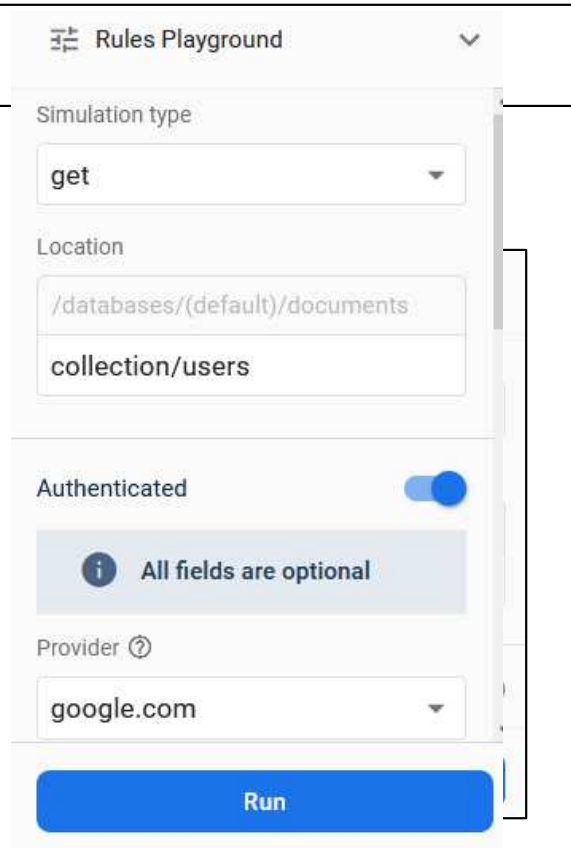
```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if true;
    }
  }
}
```

- FB/Google/Git Authentified only

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

Simulated read allowed

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if request.auth !=
    }
  }
}
```



Realtime Database

iOS

Realtime Database

nosql databáza - rules

- Default, no access, only FB console

```
// These rules don't allow anyone read or write access to your database
{
  "rules": {
    ".read": false,
    ".write": false
  }
}
```

- Public, anyone can...

```
// These rules give anyone, even people who are not users of your app,
// read and write access to your database
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

- FB/Google/Git Authentified only

```
{
  "rules": {
    "users": {
      "$uid": {
        ".read": "$uid === auth.uid",
        ".write": "$uid === auth.uid"
      }
    }
  }
}
```

https://kamaratinamape.firebaseio.com/

Default security rules require users to be authenticat

kamaratinamape

- 4eWsnHhdRPVgUAtV2OOq7V1gNOI3: "treti
- pikatchus
  - address: "treti status"
  - lati: 48
  - longi: 17
  - name: "treti"
  - time: 1480539003061
- statuses
  - druhy: "druhy status"
  - prvy: "prvy status"
  - treti: "treti status"

Realtime Database

DATA | RULES | USAGE | BACKUPS

Firebase Email Authentification

Default security rules require users to be authenticat

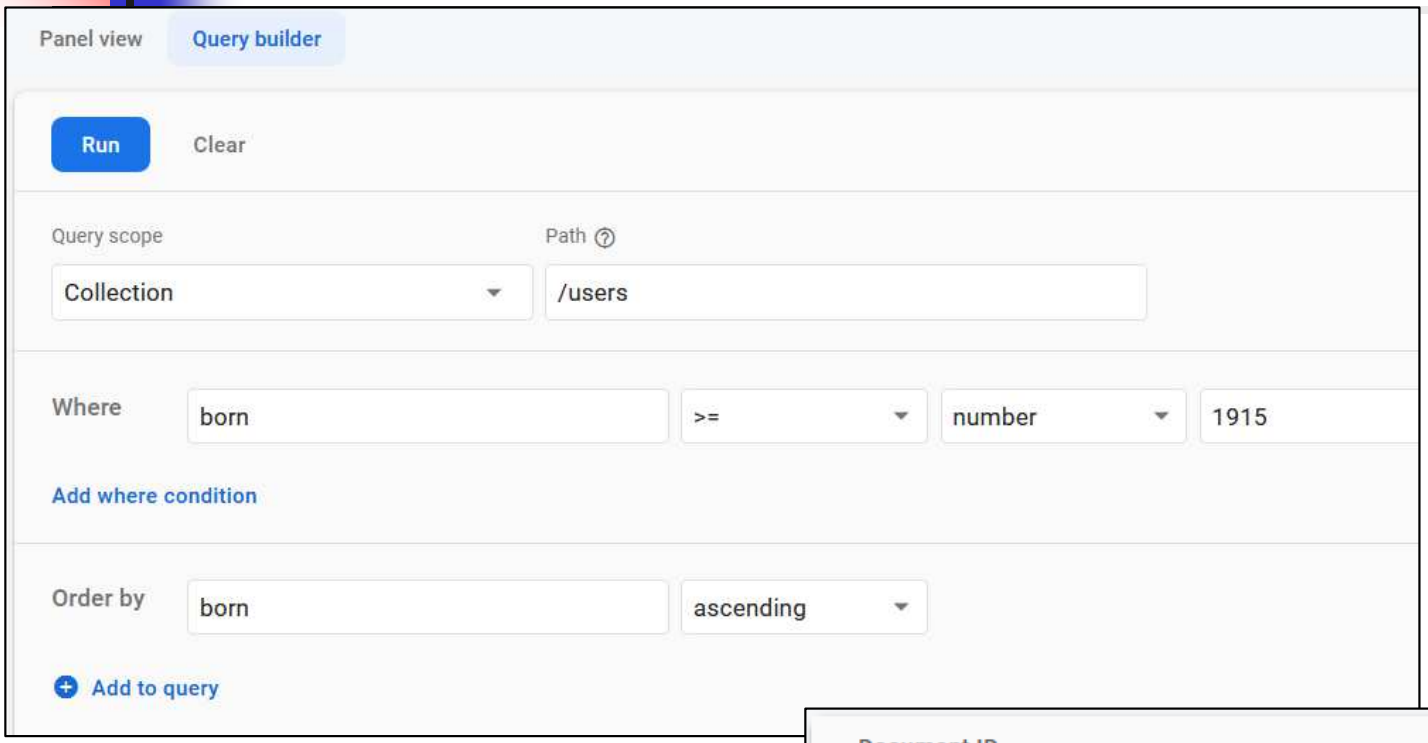
```
1  {
2    "rules": {
3      ".read": "auth != null",
4      ".write": "auth != null"
5    }
```

Realtime Database

iOS

Realtime Database

nosql databáza - rules



Document ID	born
Jcu8pq0pijBsqZKHxab0	1915
bg4ZfwhM6Q2PY5RfWpK1	1915
fQ9OVH49aAVd1O3311k0	1915
I8KGCcjVz6Eiap1Q9SjS	1919

Realtime Database

iOS

Realtime Database

zápis dát

```
databaseReference = FirebaseDatabase
        .getInstance()
        .reference
databaseReference.child("pikatchus")
    .setValue(Pokemon(name, address, lati, longi))
```

object

```
class Pokemon : Serializable
```

```
globalState == mutableMapOf<String,String>()
globalState[name]=status                // prvy -> prvy status
databaseReference.child("statuses").setValue(globalState)
```

kolekcia

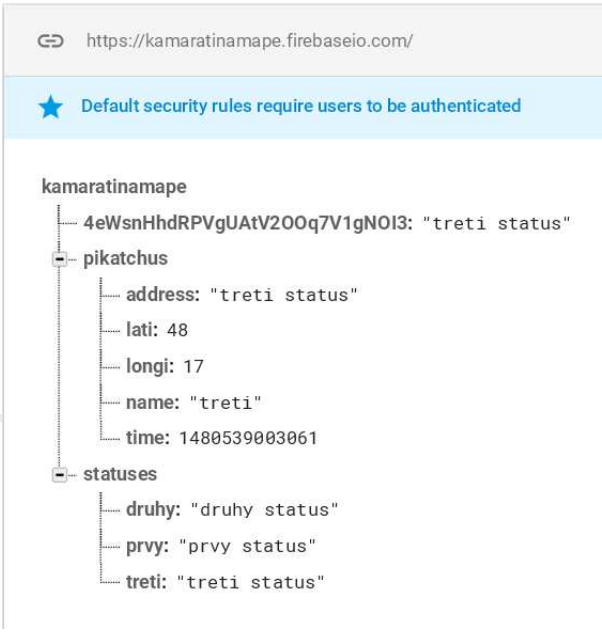
```
val user = firebaseAuth.currentUser
if (user != null) { // ak je user nalogovaný
   databaseReference.child(user.uid).setValue(status)
}
```

uid

Project:FirebaseDemo3.zip

Realtime Database

čítanie dát - synchronizácia



Pri akejkoľvek zmene sa zavolá listener

po zápise → `databaseReference.addValueEventListener(`

```
object:ValueEventListener() {
  override fun onDataChange(dataSnapshot:DataSnapshot) {
    for (child in dataSnapshot.children){ % toto sú zmeny
      val o = child.value          % (child.key, child.value)
      val str = "changed " + child.key + " is: "
      val o = child.value          % zmenil sa (key,o=value)
      if (o is Pokemon)            % zmenil sa objekt
        statusMemo.append("Pokemon has ")
      Log.d(TAG, child.key)
      editTextMemo.append(
          "changed ${child.key} is: ${o.toString()}\n")
    }
  }
})
```

object →

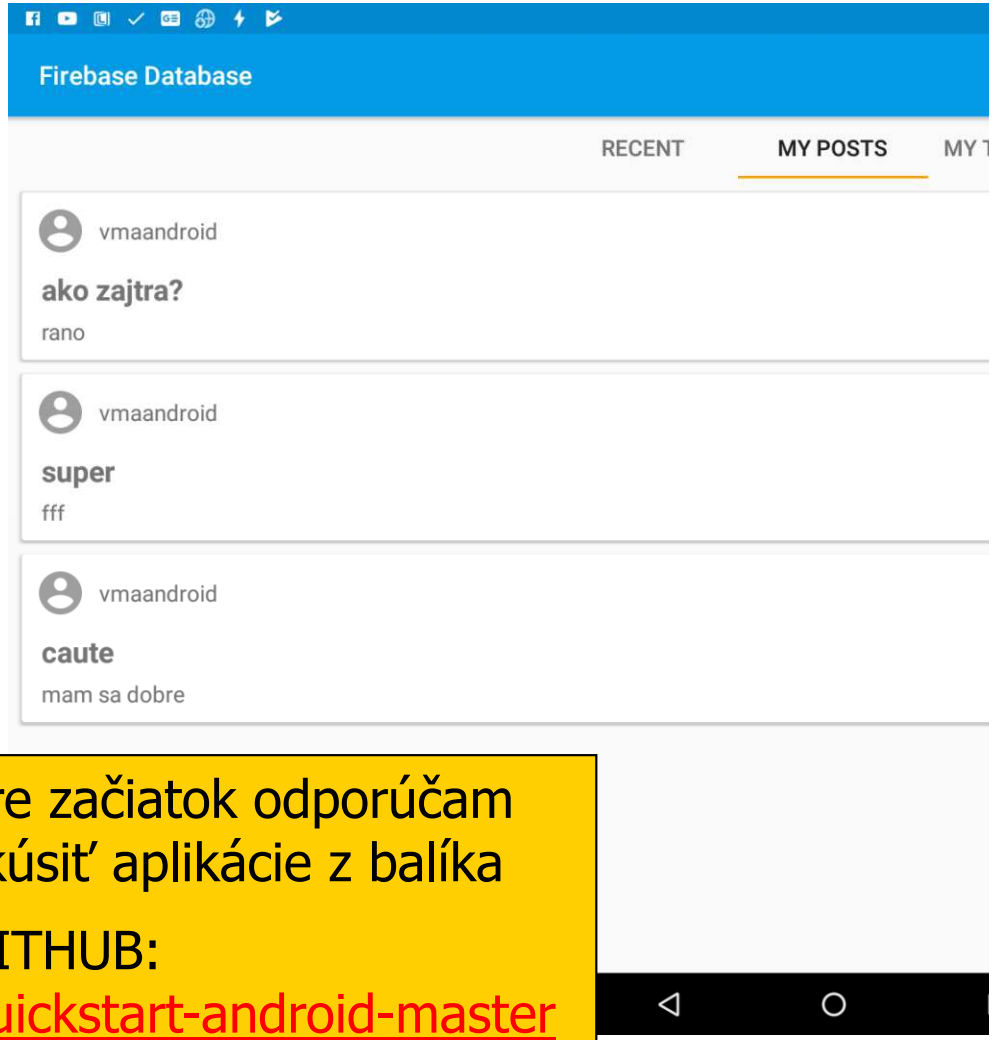
Project:FirebaseDemo3.zip

Realtime Database

iOS

Firebase DB

vyskúšajte si hotovú appku



pre začiatok odporúčam skúsiť aplikácie z balíka

GITHUB:
quickstart-android-master

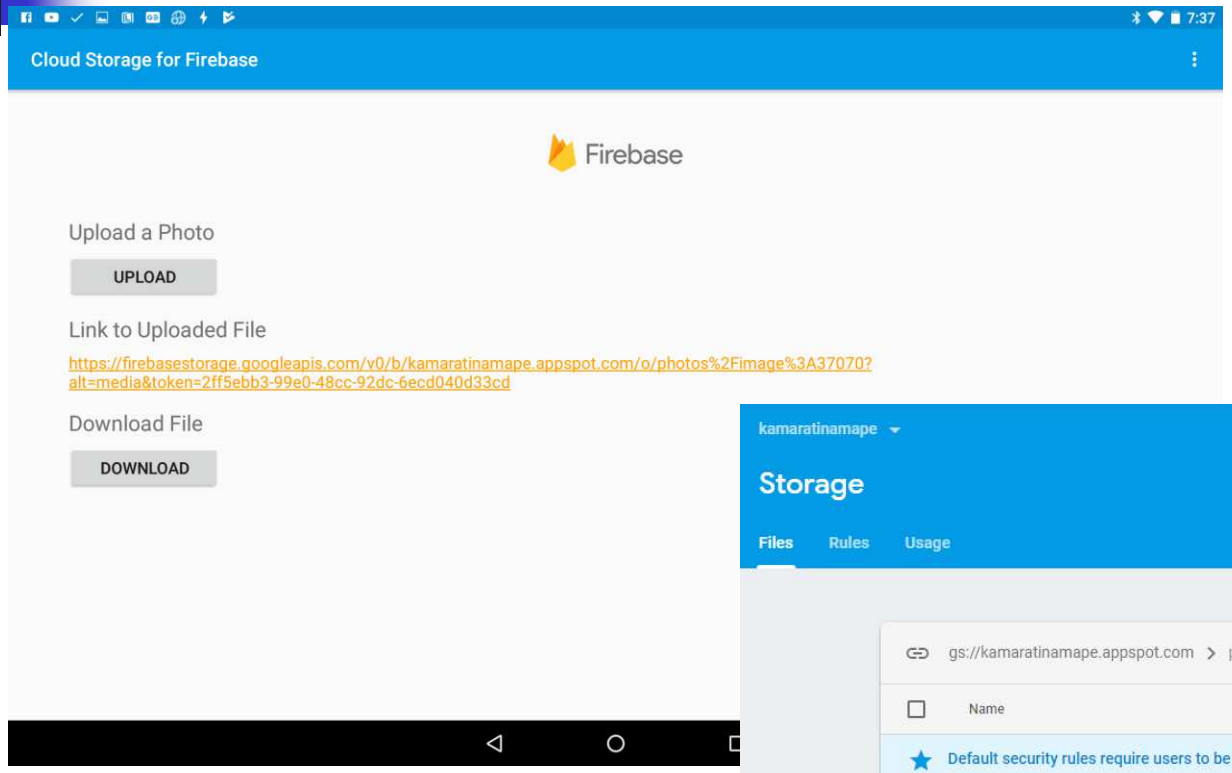
Project:database.zip

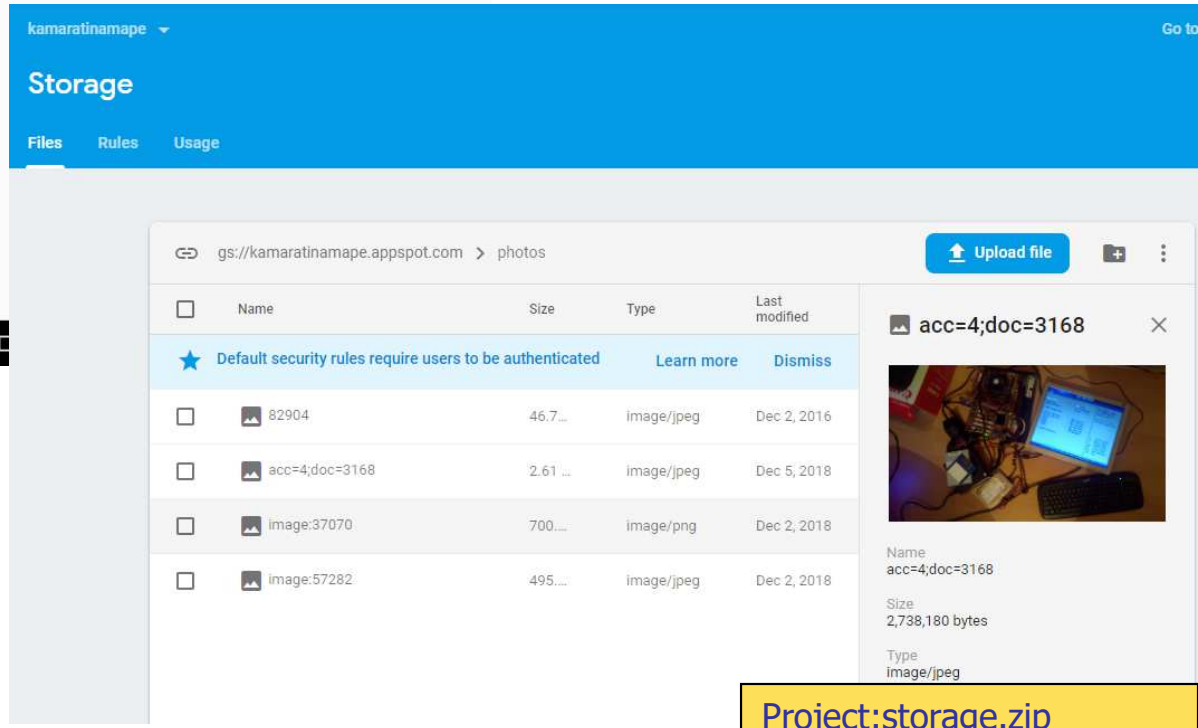
Cloud Firestore

iOS

Firebase Storage

vyskúšajte si hotovú appku





pre začiatok odporúčam skúsiť aplikácie z balíka

GITHUB:
quickstart-android-master

Project:storage.zip

Firebase Cloud Messaging

(vyskúšajte si)

pre začiatok odporúčam skúsiť aplikácie z balíka

GITHUB: quickstart-android-master

- Firebase Messaging – push notifikácie
- cross-platform (iOS, Android, ...)
- payload of up to 4kB to client app.

- Notification messages – when app is in the **background**

```
{"message":{
    "token":" chLzRZ59Svk:APA91bGEy41ulMs3qQnThxYL6VWJAOu61pIHWkGTUEHQe4rWlyWL9yutLHxiwmgYdstis7T54I68yKhwZj95TnKXjUynd4rt2oLQ1gPAaIa249g2-h4MKSg7Xkgie8uCVPx8sbB_itLr",
    "notification":{
      "title":"Notification Test",
      "body":"test"
} } }
```

- Data messages – key/value pairs received in a callback function.

```
{"message":{
    "token":" chLzRZ59Svk:APA91bGEy41ulMs3qQnThxYL6VWJAOu61pIHWkGTUEHQe4rWlyWL9yutLHxiwmgYdstis7T54I68yKhwZj95TnKXjUynd4rt2oLQ1gPAaIa249g2-h4MKSg7Xkgie8uCVPx8sbB_itLr",
    "data":{
      "Nick" : "Peter",
      "body" : "teacher",
      "Room" : "I-18"
} } }
```

Project:messaging.zip

Firebase Cloud Messaging

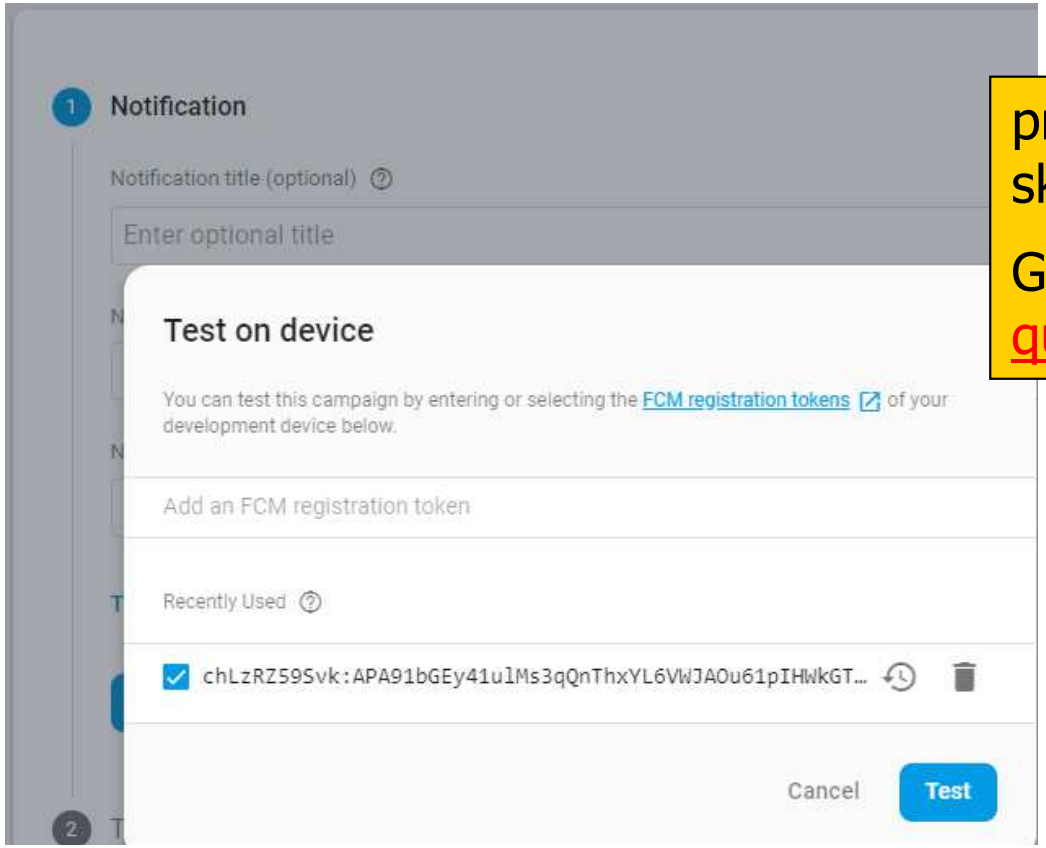
(vyskúšajte si)

Device Token

InstanceID Token:

chLzRZ59Svk:APA91bGEy41ulMs3qQnThxYL6VWJAOu61pIHWkGTUEHQe4rWlyWL9yutLHxiwmgYdstis7T54I68yKh
wZj95TnKXjUynd4rt2oLQ1gPAaIa249g2-h4MKSg7Xkgie8uCVPx8sbB_itLr

Sending a test message from Firebase Console (app is in background!)



pre začiatok odporúčam skúsiť aplikácie z balíka

GITHUB: quickstart-android-master

Project:messaging.zip