ЛЕКЦИЯ I

Android: Android Libs

Теодор Костадинов

Съдържание

SOFTWARE ACADEMY
Educational Center Net It

1. View Binding
2. Какво е Firebase?
3. FB Authentication
4. FB Realtime database
5. FB Storage
6. FB Cloud messaging
7. Google Maps
8. Picasso
9. Exercise

Преговор

Какво представляват shared preferences?

Какво е Room Database?

Какво представлява Entity?

Какво означава Dao и за какво се използва?

Преговор

Какво е AsyncTask?

Какво е retrofit и за какво се използва?

Какво е json?

Какво е gson?

View Binding

View Binding

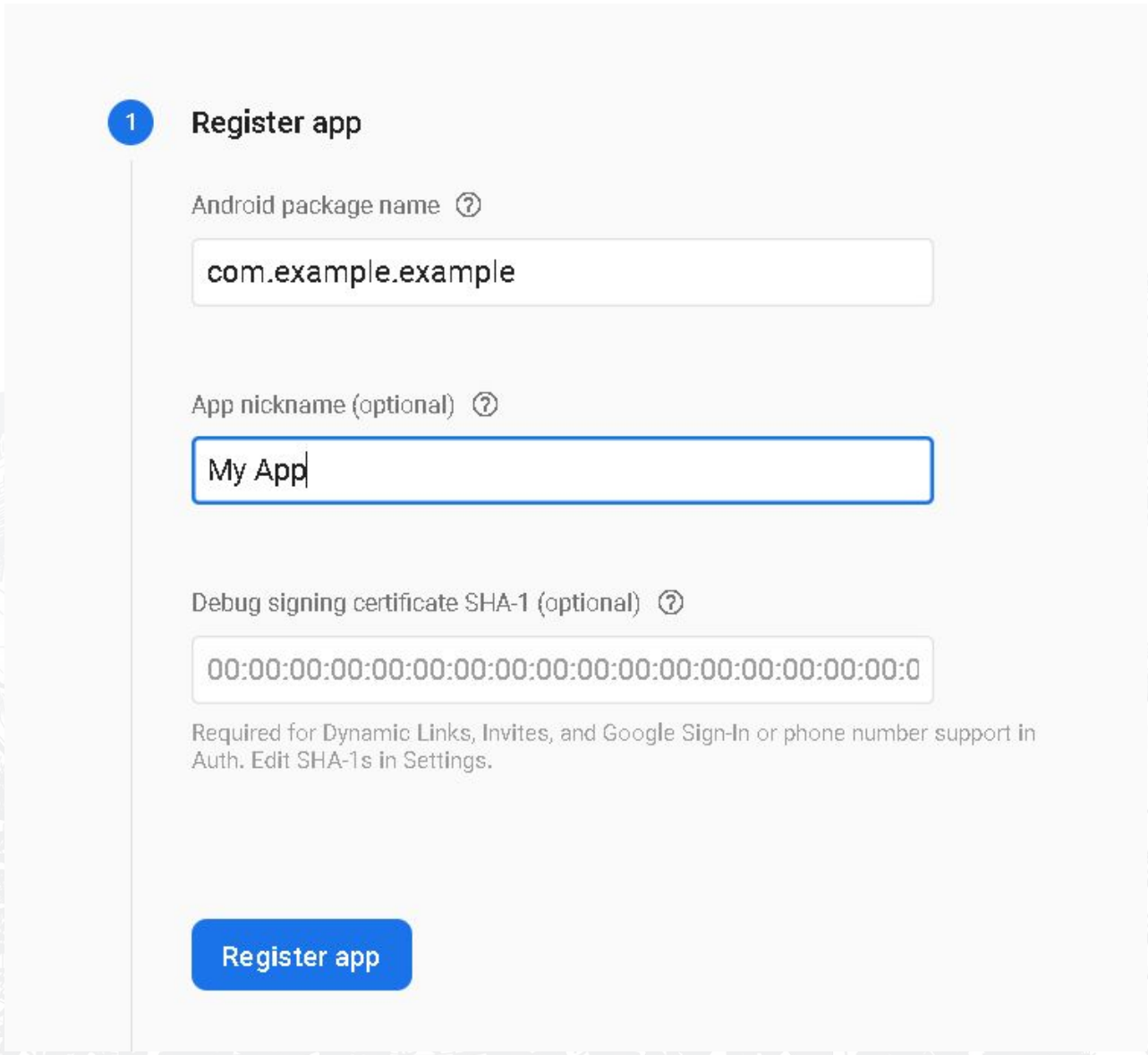
- Използването на findViewById е бавен процес, който е обвързан и с писането на много код.
- Андроид предлага алтернатива, чрез своята view binding функционалност
- Тя се пуска от грейдъл файла и чрез нея за всеки лейаут се генерира обект, който съдържа всички вюта в него

```
buildFeatures {
        viewBinding true
}
```

```
@Override
public View onCreateView (LayoutInflater inflater,
                          ViewGroup container,
                          Bundle savedInstanceState) {
    binding = ResultProfileBinding.inflate(inflater, container, false);
    View view = binding.getRoot();
    return view;
}
```

Firebase

Първи стъпки



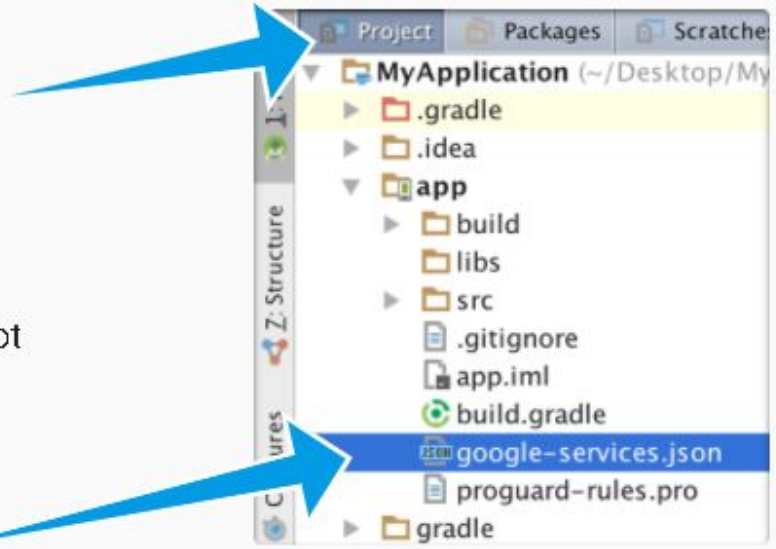
Добавяне на google-services.json

2 **Download 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 the google-services.json file you just downloaded into your Android app module root directory.

Previous Next

build gradle (project)

build.gradle(project)

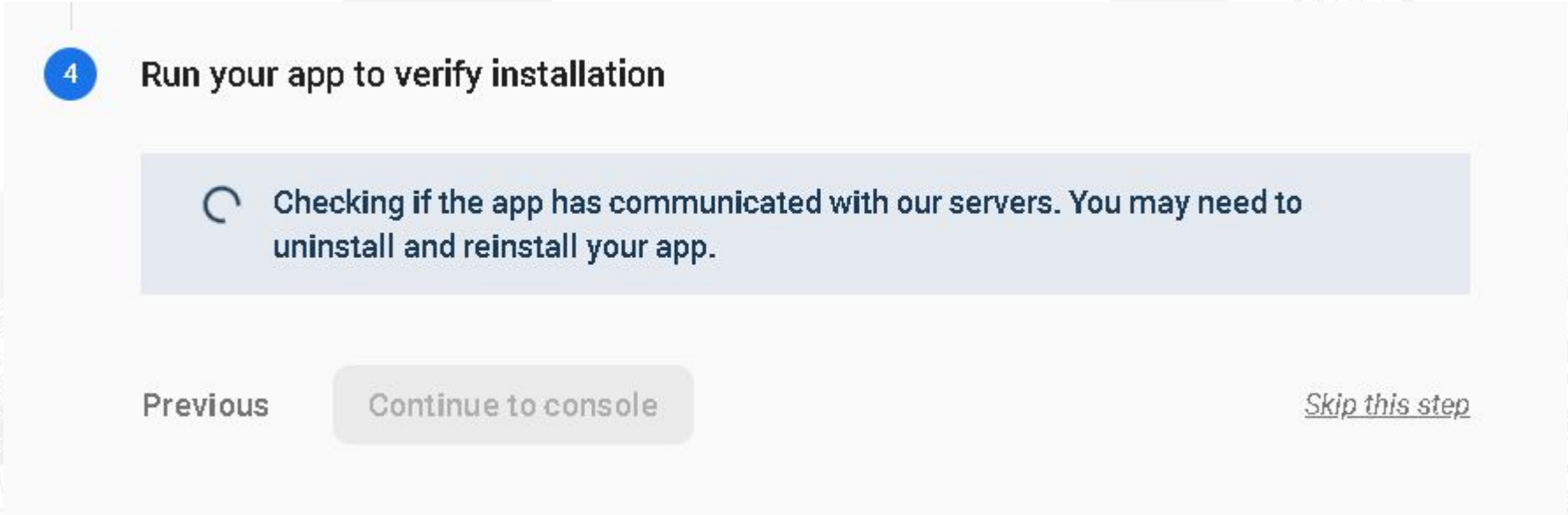
```
buildscript {
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
  }
  dependencies {
    ...
    // Add this line
    classpath 'com.google.gms:google-services:4.3.4'
  }
}
allprojects {
  ...
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
    ...
  }
}
```

build gradle (app)

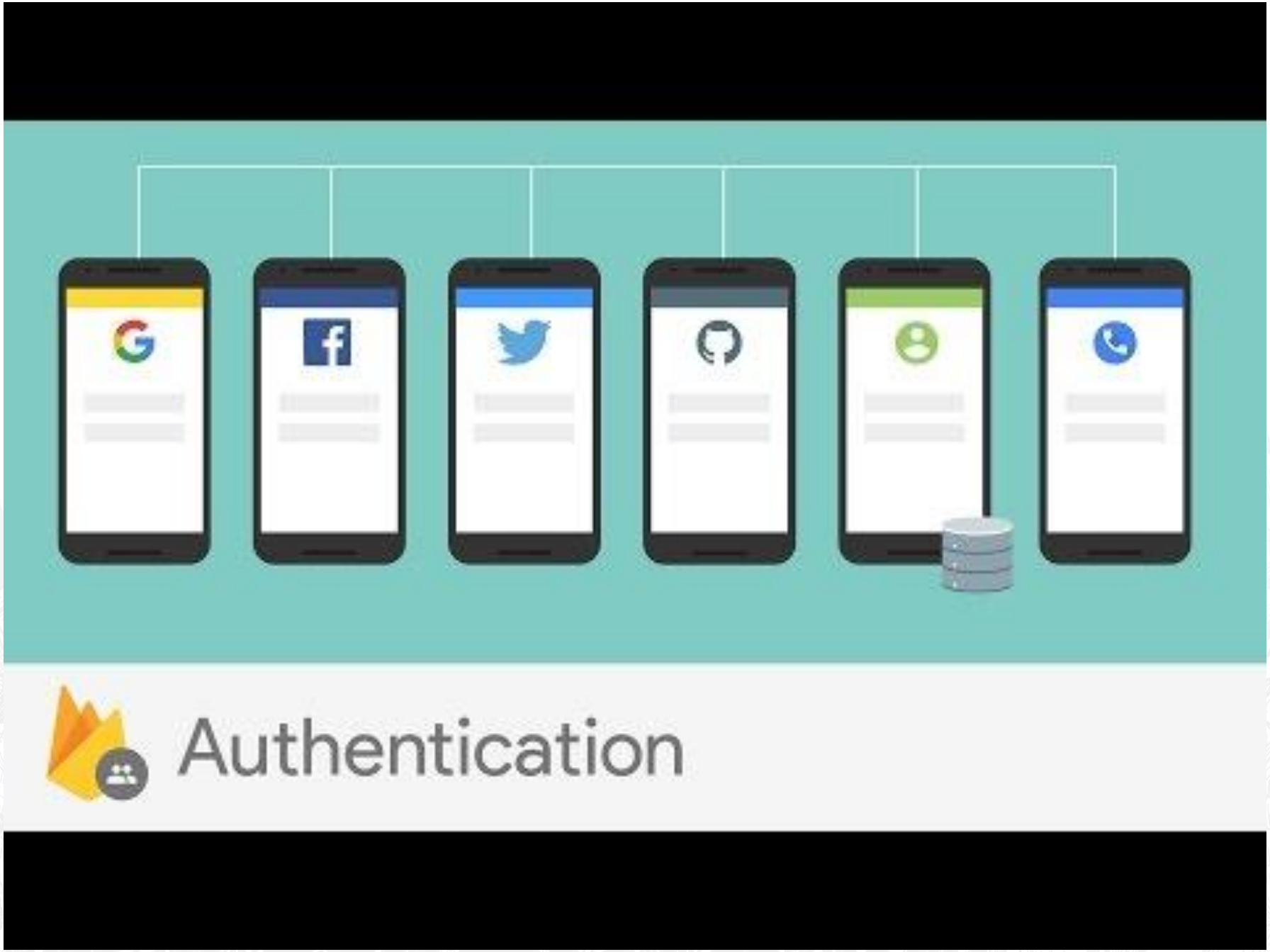
build.gradle(app)

```
apply plugin: 'com.android.application'
// Add this line
apply plugin: 'com.google.gms.google-services'

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:25.12.0')

  // Add the dependency for the Firebase SDK for Google Analytics
  // When using the BoM, don't specify versions in Firebase dependencies
  implementation 'com.google.firebase:firebase-analytics'
}
```

Верификация



Authentication


Authentication

Добавяне в проекта

```
dependencies {
    // Declare the dependency for the Firebase Authentication library
    // When NOT using the BoM, you must specify versions in Firebase library dependencies
    implementation 'com.google.firebase:firebase-auth:19.4.0'
}
```

Създаване на Auth

```
private FirebaseAuth mAuth;
// ...
// Initialize Firebase Auth
mAuth = FirebaseAuth.getInstance();
```

Проверка за потребител

```
@Override
public void onStart() {
    super.onStart();
    // Check if user is signed in (non-null) and update UI accordingly.
    FirebaseUser currentUser = mAuth.getCurrentUser();
    updateUI(currentUser);
}
```

Създаване на потребител

```
mAuth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    FirebaseUser user = mAuth.getCurrentUser();
                    updateUI(user);
                } else {
                    Log.w(TAG, "createUserWithEmail:failure", task.getException());
                    Toast.makeText(EmailPasswordActivity.this, "Authentication failed.",
                                   Toast.LENGTH_SHORT).show();
                    updateUI(null);
                }
            }
        });
```

Логване на потребител

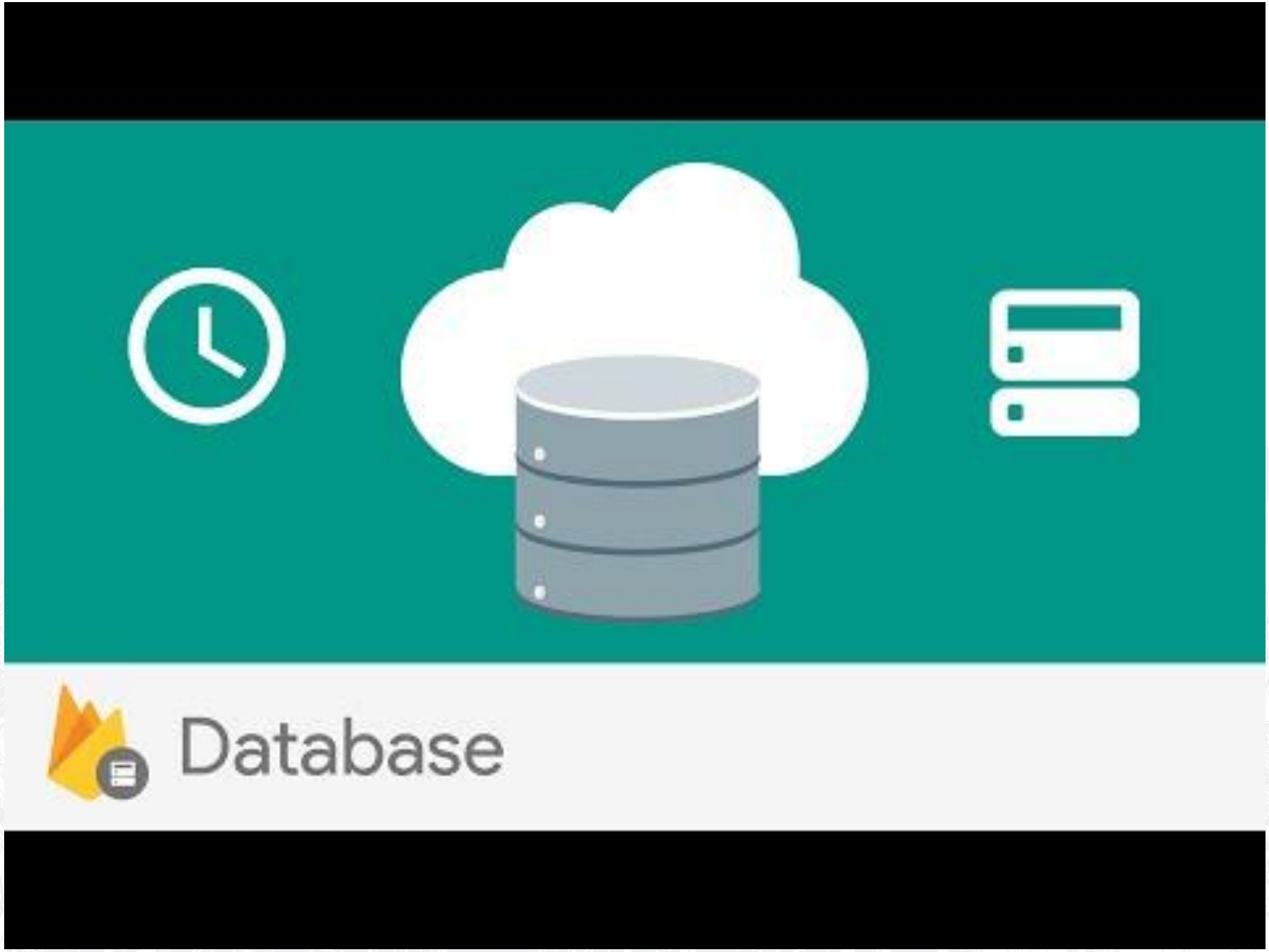
```
mAuth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    FirebaseUser user = mAuth.getCurrentUser();
                    updateUI(user);
                } else {
                    Log.w(TAG, "signInWithEmail:failure", task.getException());
                    Toast.makeText(EmailPasswordActivity.this, "Authentication failed.",
                                   Toast.LENGTH_SHORT).show();
                    updateUI(null);
                }
            }
        });
```

Sign out

За да приключим сесията на даден потребител, е достатъчно само да извикаме

```
FirebaseAuth.getInstance().signOut();
```

Realtime database


Database

Добавяне в проекта

```
dependencies {
    // Declare the dependency for the Realtime Database library
    implementation 'com.google.firebase:firebase-database:19.5.0'
}
```

Записване на прости данни

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Четене на прости данни

```
myRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        String value = dataSnapshot.getValue(String.class);
        Log.d(TAG, "Value is: " + value);
    }
    @Override
    public void onCancelled(DatabaseError error) {
        Log.w(TAG, "Failed to read value.", error.toException());
    }
});
```


chat-ef76f
Messages
-L7Ae3CJxtzYhkHJTIEQ
content: "кп"
isSentByAdmin: false
username: "Radoslav Aleksandrov"
-L7Dogaq18aXpNfYJ-bY
-L7DxmYbg2xorJNaEklq
-L7DyMZ8ToXkVkyDL3px
-L7DyQ0egwHNL0wqO64o
-L7DyaDcwvwEBBiZPsaQ


chat-ef76f
Messages
Users
0NgSeWSTtZX4QBGgjmzyf4cfWAv2
isAdmin: 1
username: "teacher"
EwJk55OadqYxCWAsgUU2eunaPgQ2
H5xxv2mTMLhqhpoFnn6oYMUiJEi1
MYC2uwNmELhtE9gfu191tolLFS63
NqATCBDuH6bwVUtlRfLTCJ2A2Aw1
QmZNtKlzYQdCdikzi2uA1D1JkPD2

Формат на данни

```
public class User {

    public String username;
    public String email;

    public User() {}
    public User(String username, String email) {
        this.username = username;
        this.email = email;
    }
}
```

Записване на данни

```
private void writeNewUser(String userId, String name, String email) {
    User user = new User(name, email);

    mDatabase.child("users").child(userId).setValue(user);
}
```

```
mDatabase.child("users").child(userId).child("username").setValue(name);
```

Изтриване на данни

```
myRef.removeValue()
```

```
myRef.setValue(null)
```

Четене на данни

```
ValueEventListener userListener = new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        User user = dataSnapshot.getValue(User.class);
    }
    @Override
    public void onCancelled(DatabaseError databaseError) {
        //log and handle error
    }
};
mUserReference.addValueEventListener(userListener);
```

Storage


Storage

Добавяне в проекта

```
dependencies {
    // Declare the dependency for the Cloud Storage library
    implementation 'com.google.firebase:firebase-storage:19.2.0'
}
```

Навигиране между локациите

"Photos"

"Photos/121"

"121"

"Photos"

```
FirebaseStorage storage = FirebaseStorage.getInstance();
storageRef = storage.getReference();

photosRef = storageRef.child("Photos");

String fileName = "121";
121Ref = imagesRef.child(fileName);

String path = 121Ref.getPath();

String name = 121Ref.getName();

photosRef = 121Ref.getParent();
```

Качване на файл

```
file = Uri.fromFile(new File("path/to/mountains.jpg"));
metadata = new StorageMetadata.Builder()
        .setContentType("image/jpeg")
        .build();
uploadTask = storageRef.child("images/"+file.getLastPathSegment()).putFile(file, metadata);
uploadTask.addOnProgressListener(new OnProgressListener<UploadTask.TaskSnapshot>() {
    @Override
    public void onProgress(UploadTask.TaskSnapshot taskSnapshot) {
        double progress = (100.0 * taskSnapshot.getBytesTransferred()) / taskSnapshot.getTotalByteCount();
        Log.d(TAG, "Upload is " + progress + "% done");
    }
}).addOnPausedListener(new OnPausedListener<UploadTask.TaskSnapshot>() {
    @Override
    public void onPaused(UploadTask.TaskSnapshot taskSnapshot) {
        Log.d(TAG, "Upload is paused");
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception exception) {
        // Handle unsuccessful uploads
    }
}).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
    @Override
    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
        // Handle successful uploads on complete
        // ...
    }
});
```

Качване на файл

```
file = Uri.fromFile(new File("path/to/mountains.jpg"));

metadata = new StorageMetadata.Builder()
        .setContentType("image/jpeg")
        .build();

uploadTask =
storageRef.child("images/"+file.getLastPathSegment()).putFile(file,
metadata);
```

Следене на прогреса

```
uploadTask.addOnProgressListener(new
OnProgressListener<UploadTask.TaskSnapshot>() {
    @Override
    public void onProgress(UploadTask.TaskSnapshot taskSnapshot) {
        double progress = (100.0 * taskSnapshot.getBytesTransferred())
                          / taskSnapshot.getTotalByteCount();
        Log.d(TAG, "Upload is " + progress + "% done");
    }
})
```

Следене за пауза

```
.addOnPausedListener(new OnPausedListener<UploadTask.TaskSnapshot>() {
    @Override
    public void onPaused(UploadTask.TaskSnapshot taskSnapshot) {
        Log.d(TAG, "Upload is paused");
    }
})
```

Error handling

```
.addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception exception) {
        // Handle unsuccessful uploads
    }
})
```

Успешно качване

```
.addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
    @Override
    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
        // Handle successful uploads on complete
        // ...
    }
});
```

Сваляне на файлове (bytes)

```
storageRef.child("users/me/profile.png").getBytes(Long.MAX_VALUE)
.addOnSuccessListener(new OnSuccessListener<byte[]>() {
    @Override
    public void onSuccess(byte[] bytes) {
        // Use the bytes to display the image
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception exception) {
        // Handle any errors
    }
});
```

Сваляне на файлове (Uri)

```
storageRef.child("users/me/profile.png").getDownloadUrl().addOnSuccessLi
stener(new OnSuccessListener<Uri>() {
    @Override
    public void onSuccess(Uri uri) {
        // Got the download URL for 'users/me/profile.png'
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception exception) {
        // Handle any errors
    }
});
```

Cloud messaging


Cloud Messaging

Добавяне в проекта

```
dependencies {
    // Declare the dependencies for the Firebase Cloud Messaging and Analytics libraries
    implementation 'com.google.firebase:firebase-messaging:20.3.0'
    implementation 'com.google.firebase:firebase-analytics:17.6.0'
}
```

Intent filter

```
<service
    android:name=".java.MyFirebaseMessagingService"
    android:exported="false">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT" />
    </intent-filter>
</service>
```

Какво е registration token?

При първото стартиране на приложението се генерира registration token за съответната му инстанция

Този token може да се променя когато:

- Приложението се преинсталира
- Приложението се инсталира на ново устройство
- Данните на приложението бъдат изтрити

Какво е registration token?

```
FirebaseMessaging.getInstance().getToken()
    .addOnCompleteListener(new OnCompleteListener<String>() {
        @Override
        public void onComplete(@NonNull Task<String> task) {
          if (!task.isSuccessful()) {
            Log.w(TAG, "Fetching FCM registration token failed",
                    task.getException());
            return;
          }
          String token = task.getResult();
          Log.d(TAG, token);
        }
    });
```

1 Notification

Notification title

Enter optional title

Notification text

Enter notification text

Notification image (optional)

Example: https://yourapp.com/image.png

Notification name (optional)

Enter optional name

Next

Device preview

This preview provides a general idea of how your message will appear on a mobile device. Actual message rendering will vary depending on the device. Test with a real device for actual results.

Send test message

Initial state | Expanded view

Android

iOS

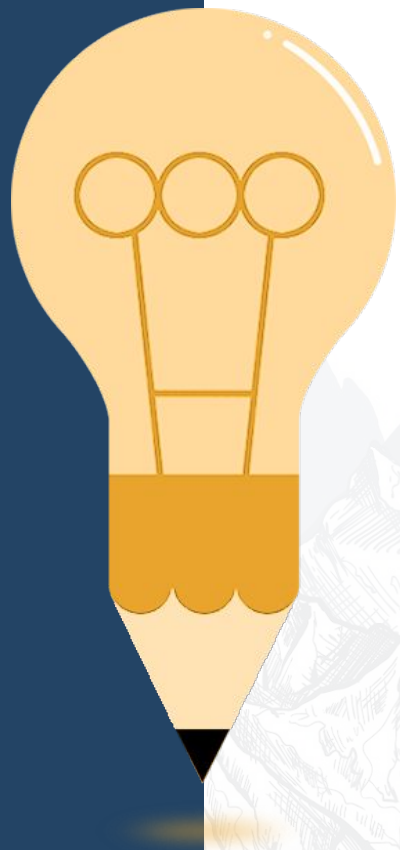
Google Maps

Google Maps

- Процесът изисква създаването на проект в Cloud Console на Google и не е най-прост
- Cloud Console е платформа подобна на FireBase, но с повече функции
-

https://developers.google.com/maps/documentation/android-sdk/start

Задача



Създайте приложение - дневник.
Записите в него съдържат:

- дата
- разказ как е минал денят ви
- настроение (може да бъде добро, средно, лошо)

Приложението да има екран за записване на преживяване и екран за показване на всички записани до сега
Използвайте Room Database

ВЪПРОСИ?

БЛАГОДАРЯ
ЗА ВНИМАНИЕТО!