Лабораторна робота №8. Зберігання даних засобами Android

Мета: вивчити порядок роботи з базами даних SQLite та Firebase в Android.

Теоретичні відомості

Для зберігання даних у Android передбачено декілька можливостей. На вибір методу зберігання даних впливають такі особливості додатку як об'єм необхідної пам'яті, типи даних для зберігання, права доступу до даних для різних користувачів, чи повинні дані залишатися після видалення додатку.

Загалом, можна виділити наступні методи зберігання даних:

1. Внутрішнє файлове сховище.
2. Зовнішнє файлове сховище.
3. Зберігання у файлах даних «ключ-значення».
4. Локальна база даних.
5. Хмарна база даних.

Внутрішнє файлове сховище (https://developer.android.com/training/data-storage/files.html#WriteInternalStorage). За замовченням, дані, які зберігаються у внутрішньому файловому сховищі приватні для даного додатку, тобто інші додатки та користувачі (крім root) не мають до них доступу. При видаленні додатку видаляються і дані з внутрішнього файлового сховища.

Внутрішнє сховище специфікується ім'ям додатку, який його використовує, в системній директорії , для доступу до якої використовується спеціалізоване API.

Таким чином, часто внутрішнє файлове сховище використовується для зберігання службових даних, до яких користувач не повинен мати доступу.

Особливості внутрішнього файлового сховища (internal storage)

1. Приватне для додатку.
2. Дані видаляються після видалення додатку.
3. Користувач не має прямого доступу.

Більш докладно про роботу з

Функції доступу та запису\читання даних з внутрішнього сховища.

- **getFilesDir().** Повертає об’єкт класу **File**, який використовується для представлення шляху файлів або директорій.

```
• File file = new File(context.getFilesDir(), filename);
```

- **getCacheDir().** Повертає File-об’єкт внутрішнього файлового сховища тимчасових файлів.

Для створення файлів в одній з цих двох директорій можна використати наступну конструкцію:

```
String filename = "myfile";
String fileContents = "Hello world!";
FileOutputStream outputStream;

try {
    outputStream = openFileOutput(filename, Context.MODE_PRIVATE);
    outputStream.write(fileContents.getBytes());
    outputStream.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

- **createTempFile().** Створення тимчасового файлу, який доступний тільки даному додатку.

```
private File getTempFile(Context context, String url) {
    File file;
    try {
        String fileName = Uri.parse(url).getLastPathSegment();
        file = File.createTempFile(fileName, null, context.getCacheDir());
    } catch (IOException e) {
        // Error while creating file
    }
    return file;
}
```

- **openFileInput(name)**. Відкриття існуючого файлу.
- **getDir(name, mode)**. Створює нову директорію (або відкриває існуючу) в файловій системі.

Зовнішнє файлове сховище (https://developer.android.com/training/data-storage/files.html#WriteExternalStorage).

Особливості роботи:

1. Файли доступні для інших додатків та користувачів (мультимедіа файли, текстові файли тощо). Зазвичай, файли розміщуються на SD-картці.

2. Public files. Доступні для додатків та користувачів. Залишаються після видалення додатку.

3. Private files. Належать додатку та видаляються при його видаленні.

Перед початком роботи з зовнішнім файловим сховищем необхідно в маніфест файлі вказати дозвіл на зберігання файлів (рис. 3) та виконати перевірку на доступність (рис. 4).

```
<manifest ...>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    ...
</manifest>
```

Рис. 3

```
/* Checks if external storage is available for read and write */
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

/* Checks if external storage is available to at least read */
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

Рис. 4

Зберігання файлів у public директорію:

```
public File getPublicAlbumStorageDir(String albumName) {
    // Get the directory for the user's public pictures directory.
    File file = new File(Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_PICTURES), albumName);
    if (!file.mkdirs()) {
        Log.e(LOG_TAG, "Directory not created");
    }
    return file;
}
```

Рис. 5

Зберігання файлів у private директорію:

```
public File getPrivateAlbumStorageDir(Context context, String albumName) {
    // Get the directory for the app's private pictures directory.
    File file = new File(context.getExternalFilesDir(
            Environment.DIRECTORY_PICTURES), albumName);
    if (!file.mkdirs()) {
        Log.e(LOG_TAG, "Directory not created");
    }
    return file;
}
```

Рис. 6

Зберігання у файлах даних «ключ-значення» (https://developer.android.com/training/data-storage/shared-preferences.html).
Особливості збереження даних у вигляді «ключ-значення»

1. Використовується для збереження відносно невеликого об’єму даних у зовнішньому файлі.
2. SharedPreferences API для управління записом/читанням.

Методи збереження даних у вигляді «ключ-значення»

```
Context context = getActivity();
SharedPreferences sharedPref = context.getSharedPreferences(
        getString(R.string.preference_file_key), Context.MODE_PRIVATE);
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
```

Рис. 7

Запис даних у файл:

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putInt(getString(R.string.saved_high_score_key), newHighScore);
editor.commit();
```

Рис. 8

Читання даних з файлу:

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
int defaultValue = getResources().getInteger(R.integer.saved_high_score_default_key);
int highScore = sharedPref.getInt(getString(R.string.saved_high_score_key), defaultValue);
```

Рис. 9

Робота з локальною базою даних (https://developer.android.com/training/data-storage/room/index.html).
Особливості:

1. Використання бібліотеки Room у якості проміжного шару між додатком та SQLite.
2. Робота безпосередньо з базою даних SQLite.

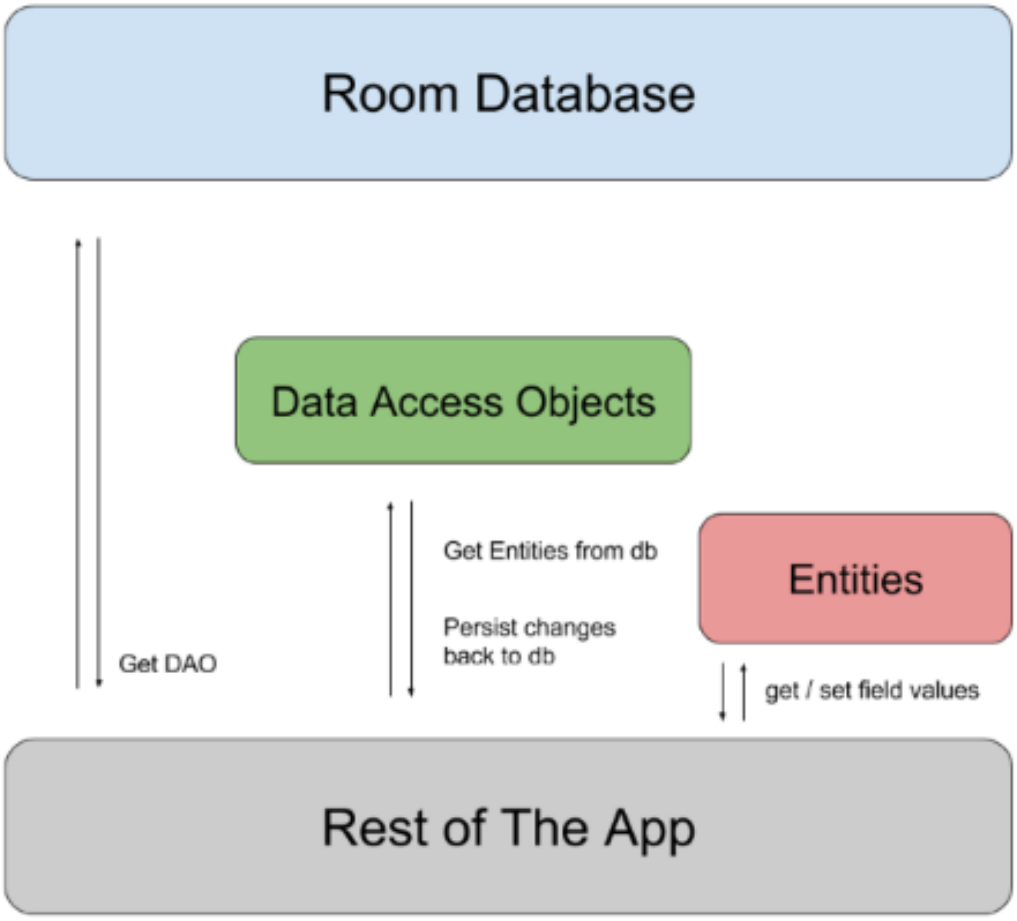

Рис. 10

Залежності build.gradle при додаванні Room в проект:

```
dependencies {
    implementation "android.arch.persistence.room:runtime:1.0.0"
    annotationProcessor "android.arch.persistence.room:compiler:1.0.0"
    ...}
```

Опис сутності:

User.java

```java
@Entity
public class User {
    @PrimaryKey
    private int uid;

    @ColumnInfo(name = "first_name")
    private String firstName;

    @ColumnInfo(name = "last_name")
    private String lastName;

    // Getters and setters are ignored for brevity,
    // but they're required for Room to work.
}
```

Рис. 11

Опис методів маніпулювання з даними:

UserDao.java

```java
@Dao
public interface UserDao {
    @Query("SELECT * FROM user")
    List<User> getAll();

    @Query("SELECT * FROM user WHERE uid IN (:userIds)")
    List<User> loadAllByIds(int[] userIds);

    @Query("SELECT * FROM user WHERE first_name LIKE :first AND "
           + "last_name LIKE :last LIMIT 1")
    User findByName(String first, String last);

    @Insert
    void insertAll(User... users);

    @Delete
    void delete(User user);
}
```

Рис. 12

AppDatabase.java

```java
@Database(entities = {User.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {
    public abstract UserDao userDao();
}

AppDatabase db = Room.databaseBuilder(getApplicationContext(),
        AppDatabase.class, "database-name").build();
```

Рис. 13

```
public class App extends Application {

    public static App instance;

    private AppDatabase database;

    @Override
    public void onCreate() {
        super.onCreate();
        instance = this;
        database = Room.databaseBuilder(this, AppDatabase.class, "database")
                .build();
    }

    public static App getInstance() {
        return instance;
    }

    public AppDatabase getDatabase() {
        return database;
    }
}
```

Рис. 14

Робота з Firebase Realtime Database (https://developer.android.com/studio/write/firebase.html).

Додавання Firebase у проект (рис. 15)

Use the Firebase Assistant

Tools>Firebase>Assistant

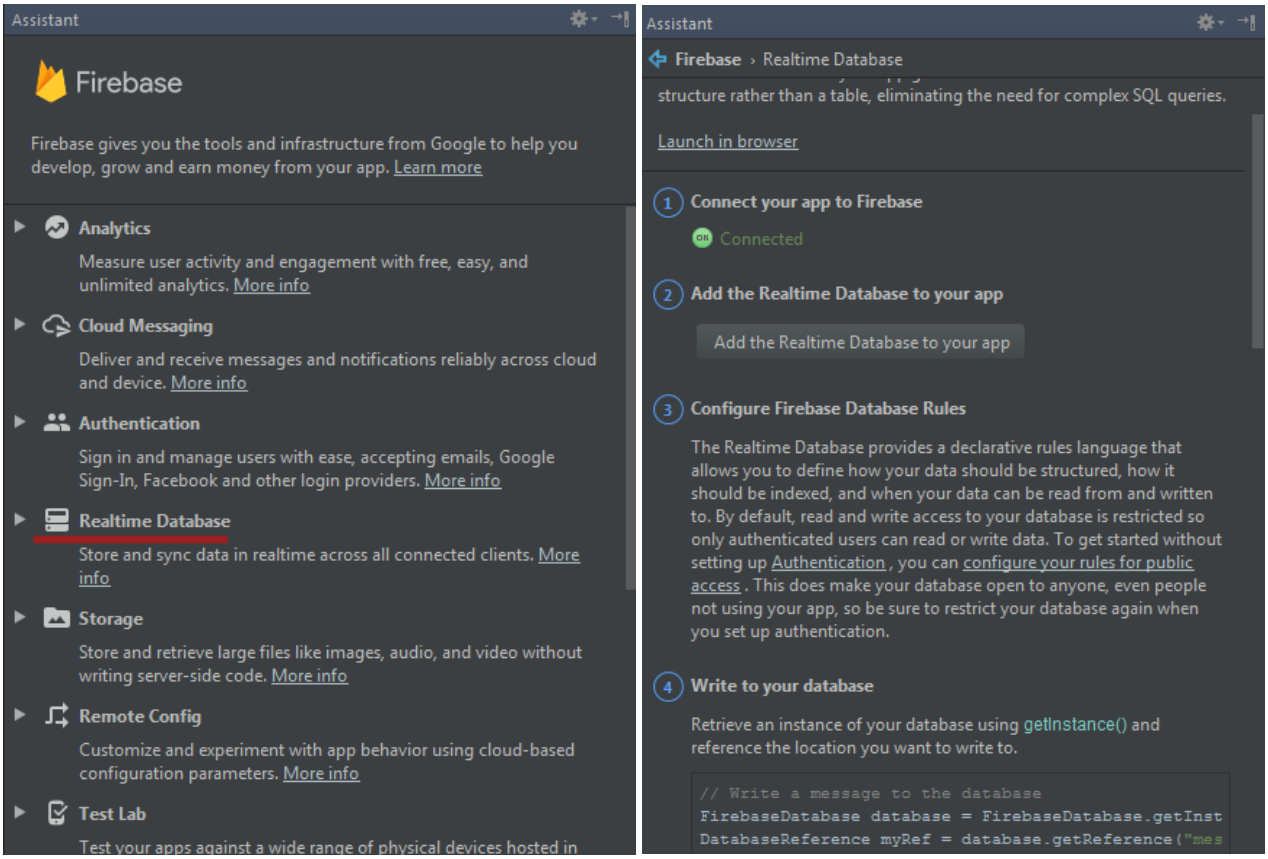
Assistant
Firebase
Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. Learn more
Analytics
Measure user activity and engagement with free, easy, and unlimited analytics. More info
Cloud Messaging
Deliver and receive messages and notifications reliably across cloud and device. More info
Authentication
Sign in and manage users with ease, accepting emails, Google Sign-In, Facebook and other login providers. More info
Realtime Database
Store and sync data in realtime across all connected clients. More info
Storage
Store and retrieve large files like images, audio, and video without writing server-side code. More info
Remote Config
Customize and experiment with app behavior using cloud-based configuration parameters. More info
Test Lab
Test your apps against a wide range of physical devices hosted in
Assistant
Firebase › Realtime Database
structure rather than a table, eliminating the need for complex SQL queries.
Launch in browser
1 Connect your app to Firebase
Connected
2 Add the Realtime Database to your app
Add the Realtime Database to your app
3 Configure Firebase Database Rules
The Realtime Database provides a declarative rules language that allows you to define how your data should be structured, how it should be indexed, and when your data can be read from and written to. By default, read and write access to your database is restricted so only authenticated users can read or write data. To get started without setting up Authentication , you can configure your rules for public access . This does make your database open to anyone, even people not using your app, so be sure to restrict your database again when you set up authentication.
4 Write to your database
Retrieve an instance of your database using getInstance() and reference the location you want to write to.
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInst
DatabaseReference myRef = database.getReference("mes

Рис. 15

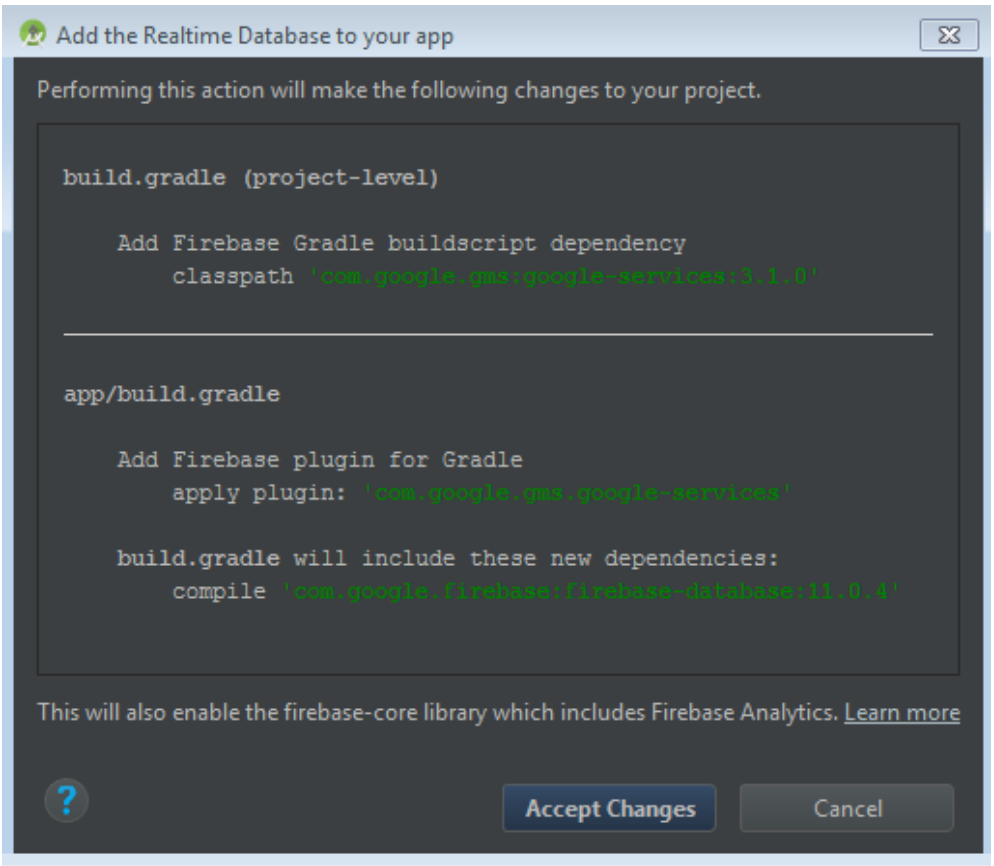
Add the Realtime Database to your app
Performing this action will make the following changes to your project.
build.gradle (project-level)
Add Firebase Gradle buildscript dependency
classpath 'com.google.gms:google-services:3.1.0'
app/build.gradle
Add Firebase plugin for Gradle
apply plugin: 'com.google.gms.google-services'
build.gradle will include these new dependencies:
compile 'com.google.firebase:firebase-database:11.0.4'
This will also enable the firebase-core library which includes Firebase Analytics. Learn more
Accept Changes
Cancel

Рис. 16

Database Rules [Firebase console]

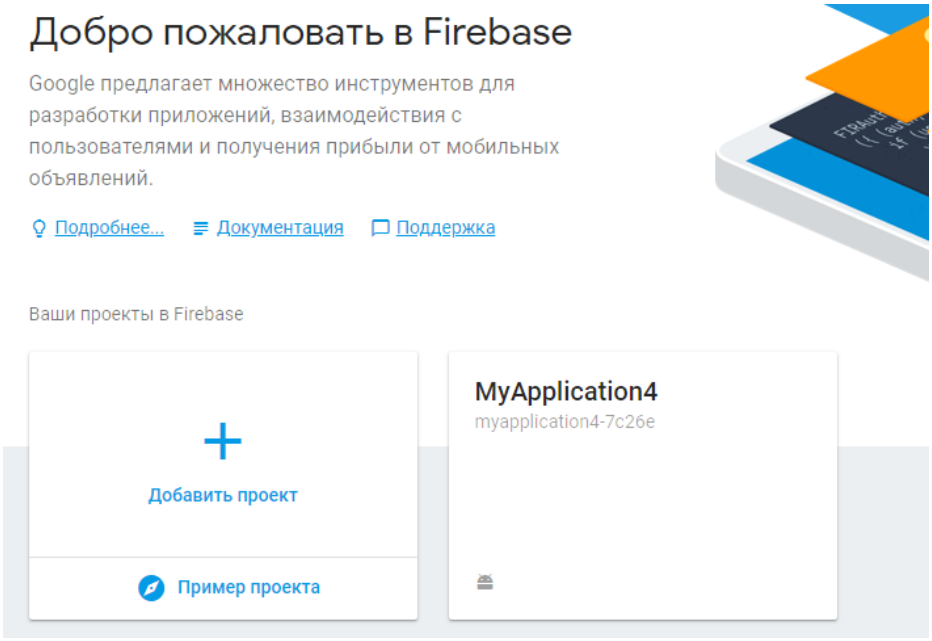

Рис. 17

Write to the database

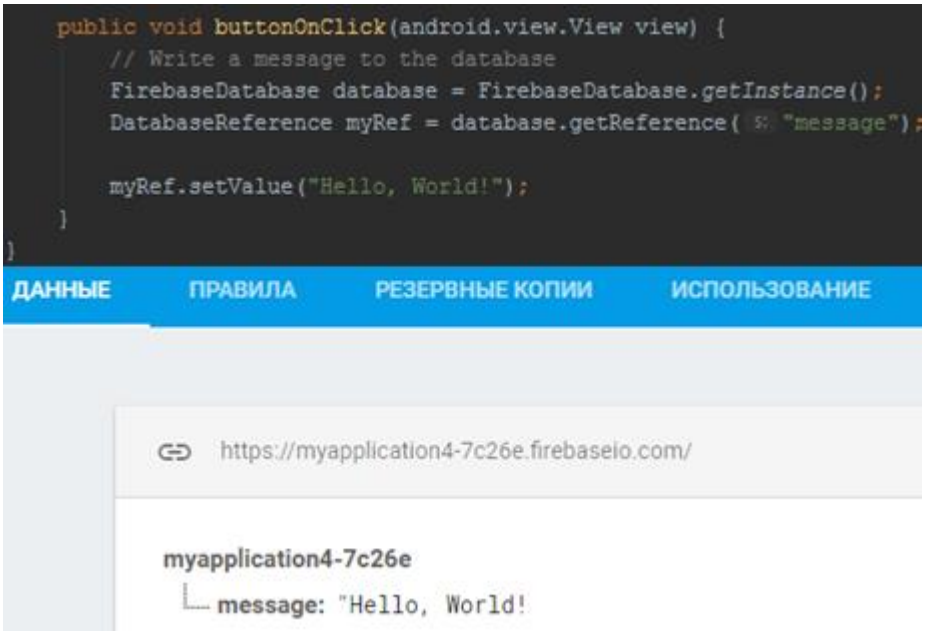

Рис. 18

Write to the database

```
public class MainActivity extends AppCompatActivity {
    Button dbButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final FirebaseDatabase database = FirebaseDatabase.getInstance();
        final DatabaseReference myRef = database.getReference("message");
        dbButton = (Button) findViewById(R.id.button2);
        dbButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Write a message to the database
                myRef.setValue("Hello, World!");
            }
        });
    }
}
```

Рис. 19

Read from the database

```
public void button2OnClick(android.view.View view) {
    // Read from the database
    myRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            // This method is called once with the initial value and again
            // whenever data at this location is updated.
            String value = dataSnapshot.getValue(String.class);
            Log.d(TAG, msg: "Value is: " + value);
        }

        @Override
        public void onCancelled(DatabaseError error) {
            // Failed to read value
            Log.w(TAG, msg: "Failed to read value.", error.toException());
        }
    });
}
```

Рис. 20

Read from the database

```
public class MainActivity extends AppCompatActivity {
    Button dbButton;
    TextView dataTextView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final FirebaseDatabase database = FirebaseDatabase.getInstance();
        final DatabaseReference myRef = database.getReference("message");
        // Read from the database
        myRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                // This method is called once with the initial value and again
                // whenever data at this location is updated.
                String value = dataSnapshot.getValue(String.class);
                dataTextView.setText(value);
            }

            @Override
            public void onCancelled(DatabaseError error) {
                // Failed to read value
                dataTextView.setText("Error " + error.toString());
            }
        });
        dbButton = (Button) findViewById(R.id.button2);
        dbButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Write a message to the database
                myRef.setValue("Hello, World!");
            }
        });
        dataTextView = (TextView) findViewById(R.id.textView);
    }
}
```

Рис. 21

Authentication in Firebase

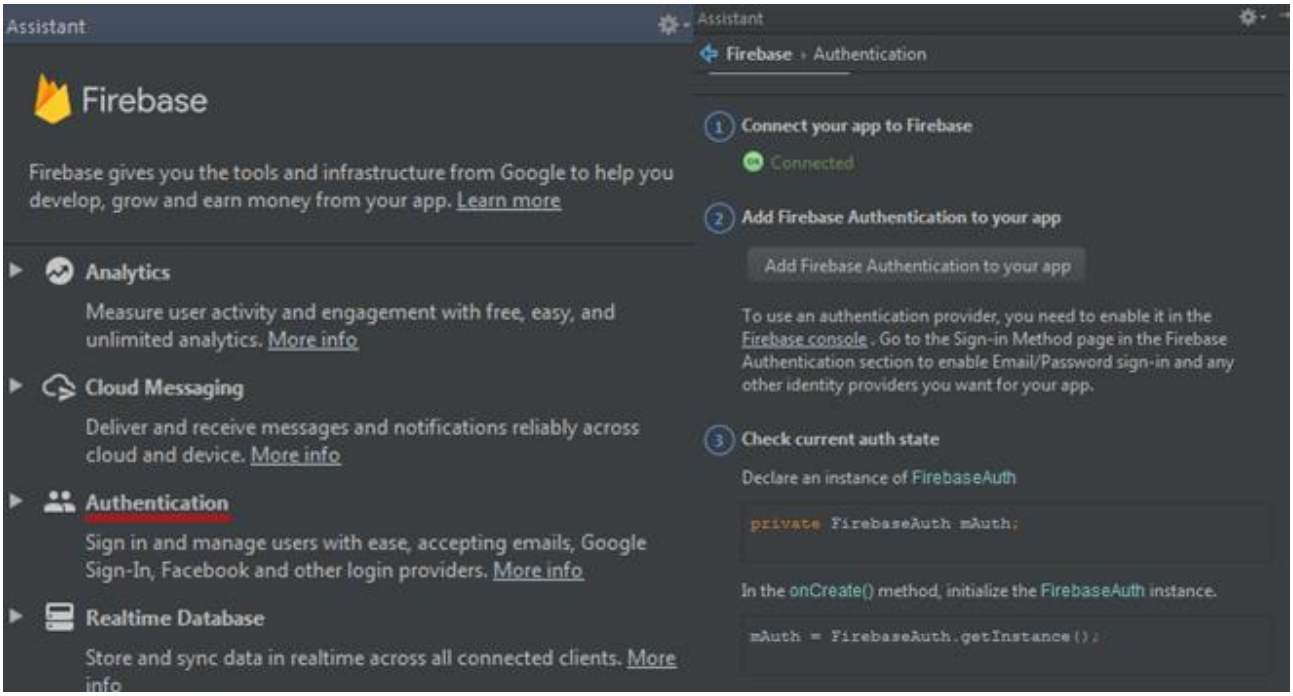
Assistant
Firebase
Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. Learn more
Analytics
Measure user activity and engagement with free, easy, and unlimited analytics. More info
Cloud Messaging
Deliver and receive messages and notifications reliably across cloud and device. More info
Authentication
Sign in and manage users with ease, accepting emails, Google Sign-In, Facebook and other login providers. More info
Realtime Database
Store and sync data in realtime across all connected clients. More info
Assistant
Firebase › Authentication
1 Connect your app to Firebase
Connected
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 Check current auth state
Declare an instance of FirebaseAuth
private FirebaseAuth mAuth;
In the onCreate() method, initialize the FirebaseAuth instance.
mAuth = FirebaseAuth.getInstance();

Рис. 22

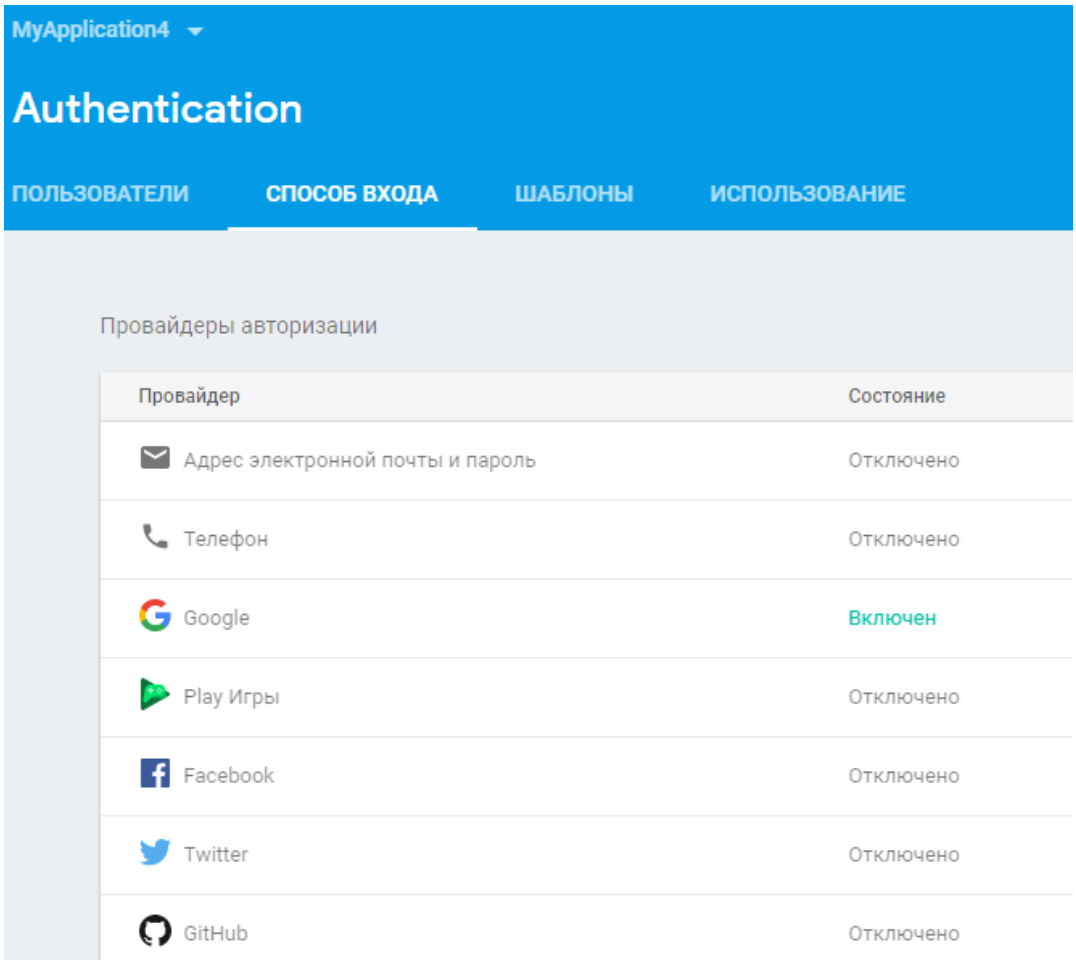
MyApplication4
Authentication
ПОЛЬЗОВАТЕЛИ
СПОСОБ ВХОДА
ШАБЛОНЫ
ИСПОЛЬЗОВАНИЕ
Провайдеры авторизации
Провайдер
Состояние
Адрес электронной почты и пароль
Отключено
Телефон
Отключено
Google
Включен
Play Игры
Отключено
Facebook
Отключено
Twitter
Отключено
GitHub
Отключено

Рис. 23

Authentication in Firebase

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    implementation 'com.google.firebase:firebase-database:11.0.4'
    implementation 'com.google.firebase:firebase-auth:11.0.4'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
}
```

Рис. 24

Завдання до лабораторної роботи

1. Додати у проект логічної гри з попередніх лабораторних робіт можливість ведення статистики ігор за допомогою локальною або хмарної бази даних з виводом статистики на останньому activity.

Контрольні запитання

1. Які способи збереження даних в Android ви знаєте?
2. Які існують особливості використання файлових сховищ?
3. Який порядок роботи з локальною та хмарною базою даних?