

ANDROID DEVELOPMENT

LISBON, FEBRUARY 2025

ERASMUS+



Bună ziua

ANDROID DEVELOPMENT

SESSION I



**WELCOME AND
PRESENTATION**



**ANDROID AND
ANDROID STUDIO**



HELLO WORLD!



***ACTIVITIES, INTENTS
AND TOASTS***

WELCOME AND PRESENTATION



Presentation of the internship tutor



Presentation of the participants



Introduction to the internship program



The presentation of the host organization

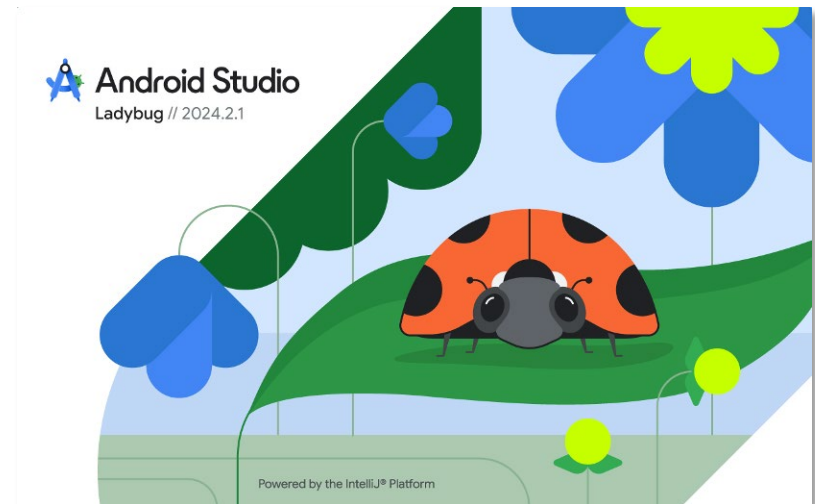
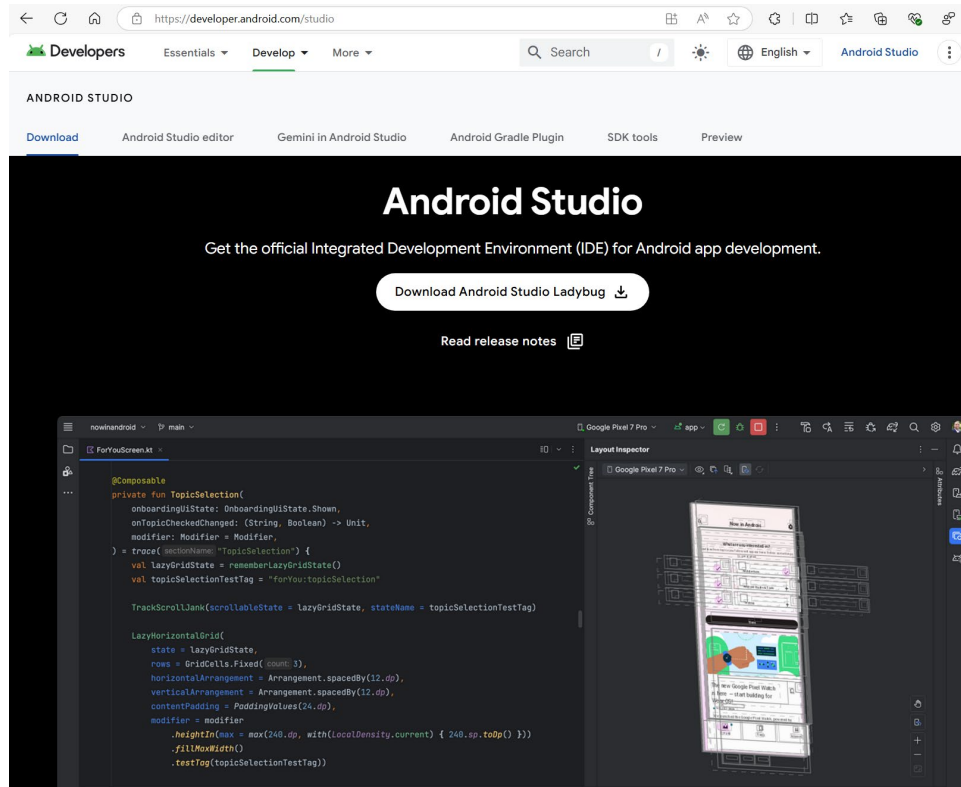
ANDROID

- Android OS is a Linux-based mobile operating system that primarily runs on smartphones and tablets
- System inside 2.5 billion active devices
- Android Inc. founded in October 2003
- Initial release: Android 1.0 September 2008
- Latest release: Android 15, October 2024

Android 

DOWNLOAD AND INSTALL ANDROID STUDIO

<https://developer.android.com/studio>



NATIVE APPS VS PROGRESSIVE WEB APPS

	Native Apps	Web Apps
Access	Installed directly on the device from app stores like the Apple App Store or Google Play Store.	Run in a web browser and do not require installation. Examples include Gmail and Google Docs.
Development	More expensive and time-consuming as they are built specifically for each operating system (iOS, Android).	Easier and cheaper to develop with a single codebase that works across multiple platforms.
Functionality	Can access all device features such as GPS, camera, and push notifications.	Limited to what web browsers support and cannot access device-specific features like GPS or camera.
Performance	Faster and more responsive due to direct interaction with the device's hardware and software.	Generally slower and less responsive compared to native apps.
Updates	Users need to download updates from the app store.	Updates are applied centrally and immediately available to all users.
User Experience	Generally more consistent and engaging, providing a smoother and more integrated experience.	Can vary based on the browser and may not be as smooth or responsive as native apps.

DRAG AND DROP TOOLS

App creation



Kodular: Kodular Creator is a powerful online tool that lets you create Android apps without writing any code. You can drag and drop components, join blocks, and create your app easily. It supports a wide range of features and components.



Adalo: Adalo is another no-code platform that allows you to build custom mobile and web apps. You can drag and drop components to design your app, and it offers features like custom fonts, color palettes, and integration with various data sources.



Thunkable: Thunkable provides a drag-and-drop interface to create Android and iOS apps. It offers various components to enhance your app with animations, maps, and more.

App prototyping



Figma: A design and prototyping tool that allows you to create high-fidelity interactive prototypes. With Figma, you can create and share realistic prototypes with no code required.



Miro: Offers a collaborative platform to quickly create interactive prototypes. It is ideal for transforming concepts into wireframes and mockups, facilitating rapid validation and iteration of designs.



Andromo: Provides a detailed guide on how to create mobile app prototypes. It highlights the importance of testing prototypes in real-world situations and using live content to get more accurate feedback.

JAVA VS KOTLIN



Java

- **Established Language:** Java has been around since 1995 and has a large, mature ecosystem.
- **Verbose Syntax:** Java code tends to be more verbose, requiring more lines of code to accomplish tasks.
- **Checked Exceptions:** Java uses checked exceptions, which can make error handling more explicit but also more cumbersome.
- **Community and Libraries:** Java has a vast community and a plethora of libraries and frameworks.
- **Performance:** Generally, Java performs well and is optimized for the JVM (Java Virtual Machine).

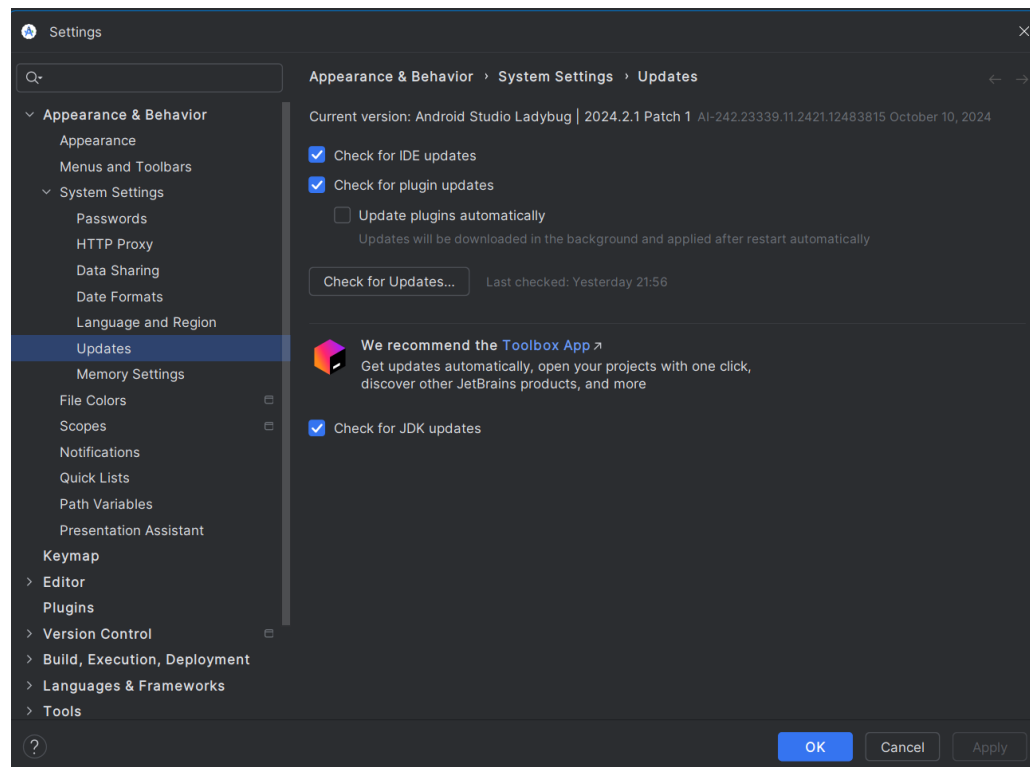
Kotlin



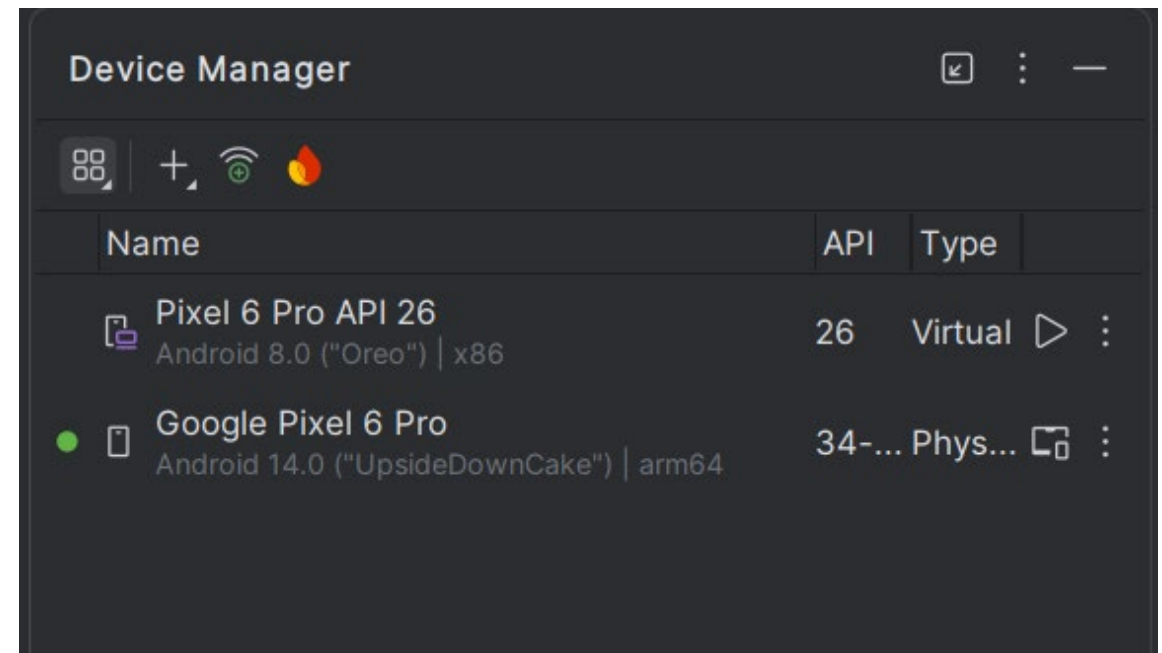
- **Modern Language:** Kotlin was introduced by JetBrains in 2011 and officially supported by Google for Android development since 2017.
- **Concise Syntax:** Kotlin is known for its concise and expressive syntax, reducing boilerplate code.
- **Null Safety:** Kotlin has built-in null safety, which helps prevent null pointer exceptions.
- **Interoperability:** Kotlin is fully interoperable with Java, meaning you can use Java libraries and frameworks seamlessly.
- **Coroutines:** Kotlin supports coroutines for asynchronous programming, making it easier to handle tasks like network calls and database operations.

ANDROID STUDIO

CHECK UPDATES AND VIRTUAL DEVICES

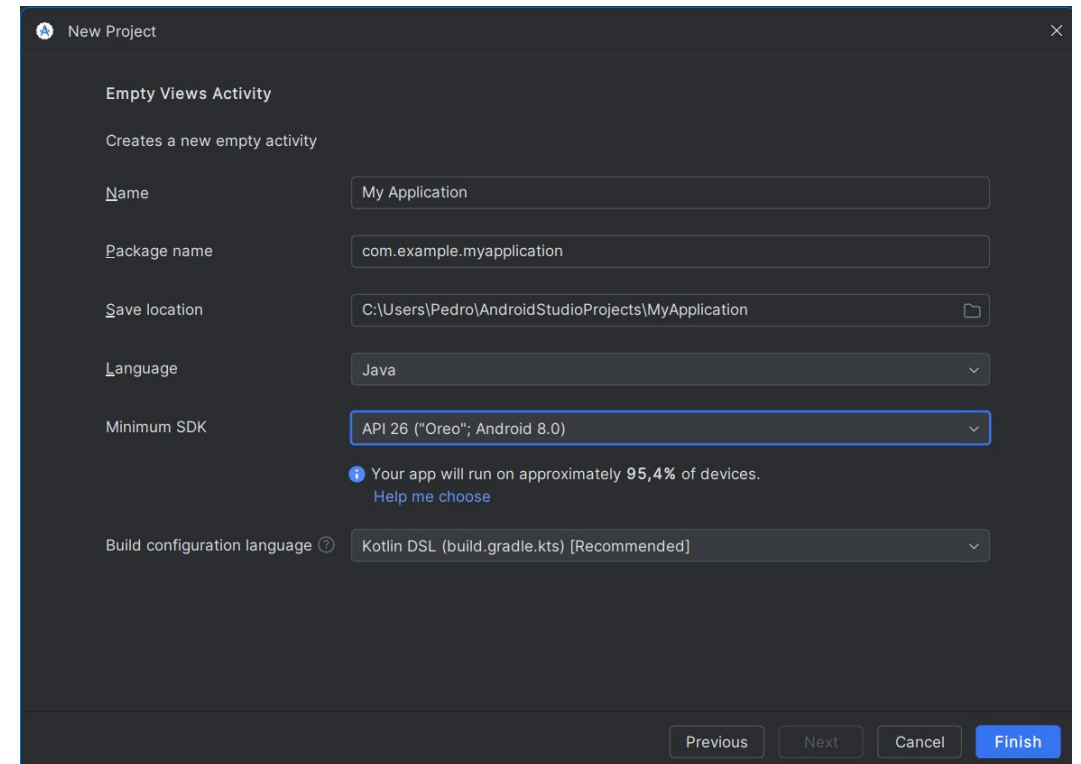
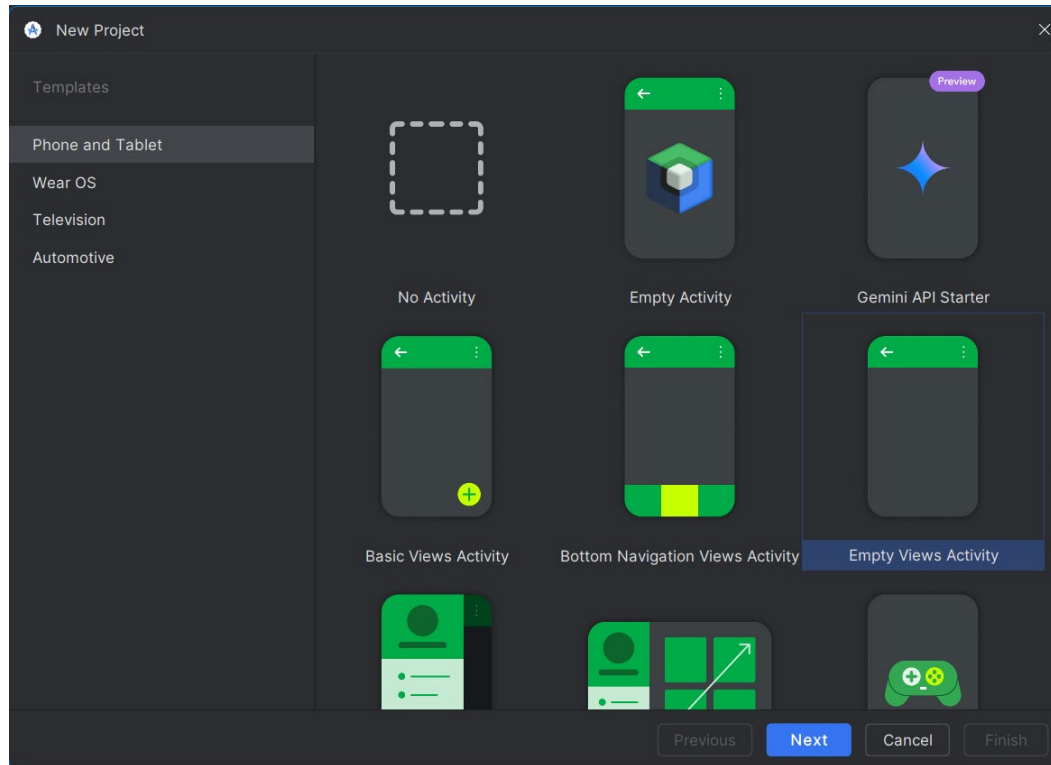


View » Tool Windows » Device Manager

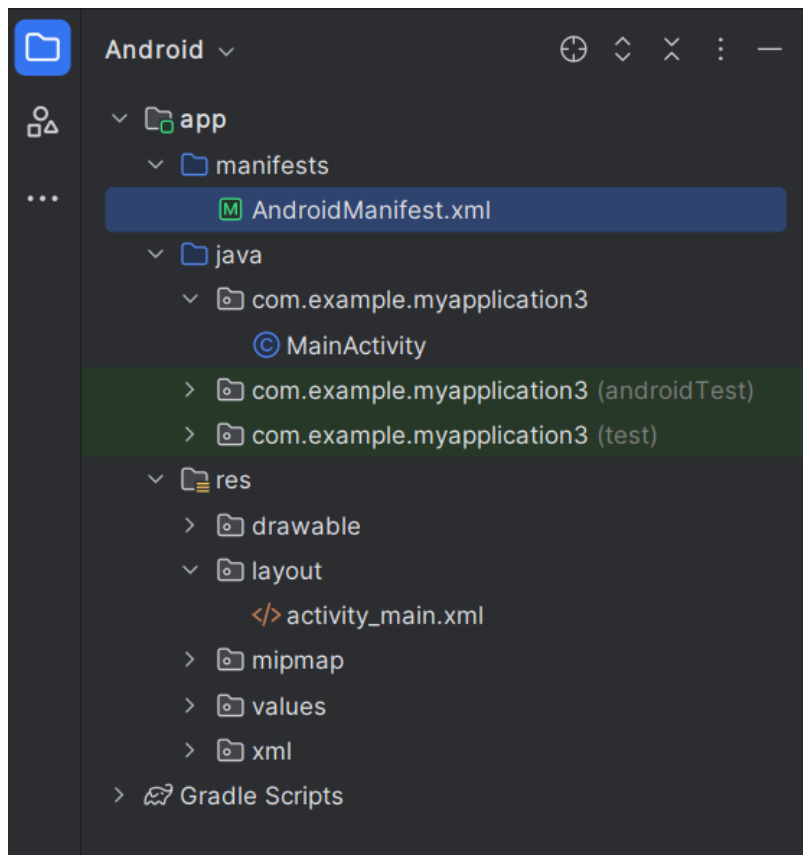


HELLO WORLD!

New project / Empty views Activity / settings (unique name, java, API 26+)



HELLO WORLD!



- **AndroidManifest.xml**

It provides essential information about your app to the Android system, which the system must have before it can run any of the app's code.

- **MainActivity.java**

Source code for the activity. It typically serves as the main entry point for the app's user interface.

- **Activity_main.xml**

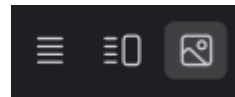
Layout file that defines the user interface for the main activity of an application in XML.

HELLO WORLD!

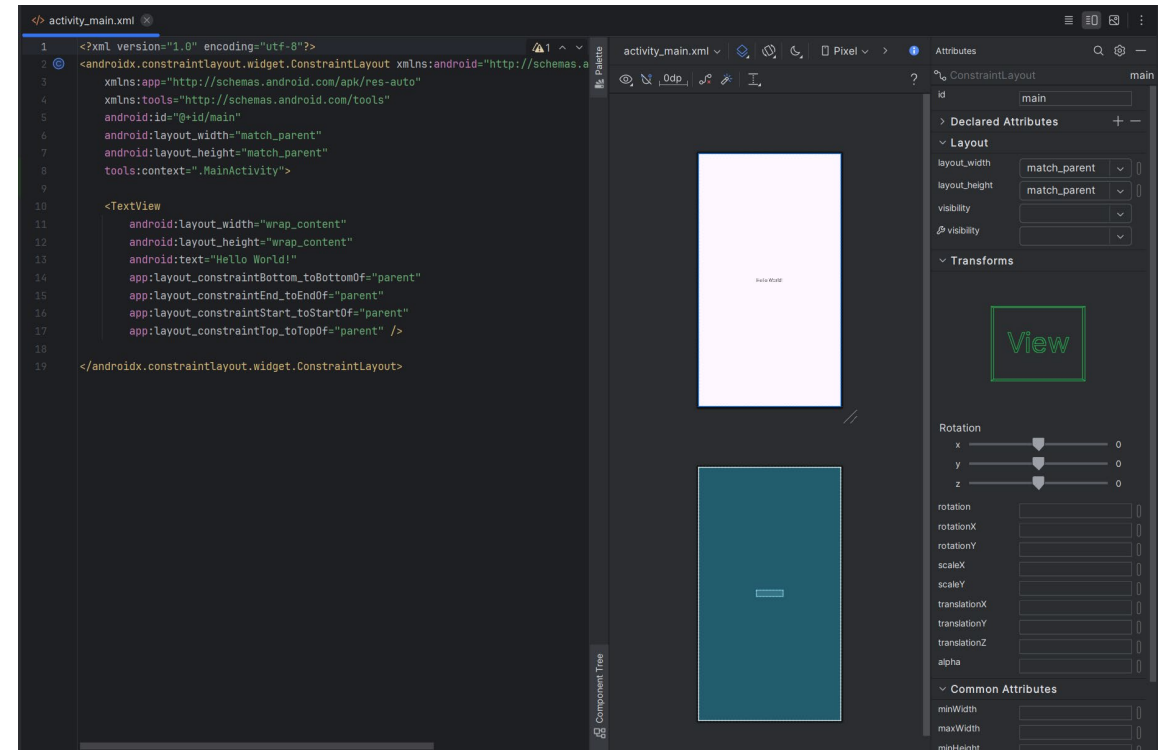
Activity_main.xml

On res / layout

To edit you can code or design



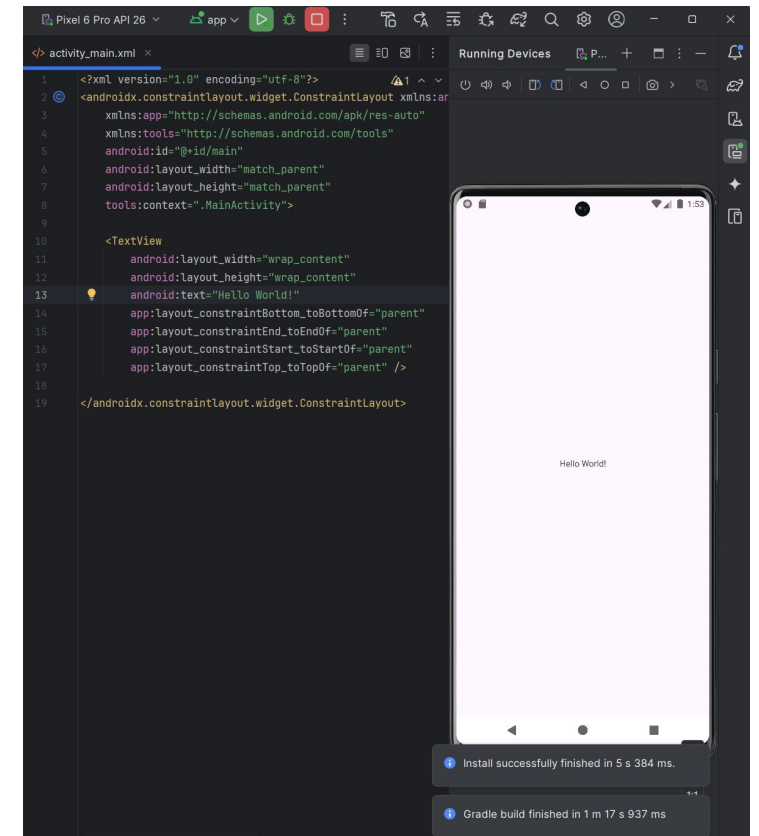
- Change text, in code
 - `android:text="Hello World!!"`
 - `android:textSize="50sp"`
- Use a background
 - Grab a file
 - Copy into res / drawable (adapt name, only lowercase, number ou underscore)
 - `android:background="@drawable/<filename>"`



HELLO WORLD!

Test on virtual device

- Click play
- The app will be prepared and run in the virtual device



TEST ON PHYSICAL DEVICES

- **Using a USB Cable ou via Using Wi-Fi (Android 11 and Higher)**
- **Enable Developer Options:**
 1. On your Android device, go to **Settings > About phone** (or Settings > About device, or Settings > About > Software information).
 2. Tap **Build number** seven times until you see a message saying “You are now a developer!”.
 3. Go back to **Settings > System > Developer options**.
 4. Enable **USB debugging** or **Wireless debugging**
- **Connect or pair your device**
- **Run Your App**

ACTIVITIES & INTENTS

An **Activity** is a crucial component that represents a single screen with a user interface. Each activity is a subclass of the Activity class and serves as the entry point for interacting with the user.

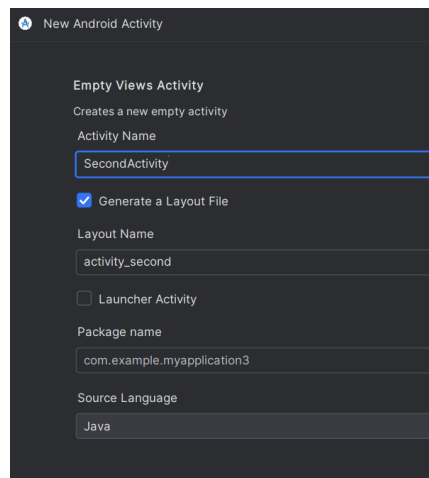
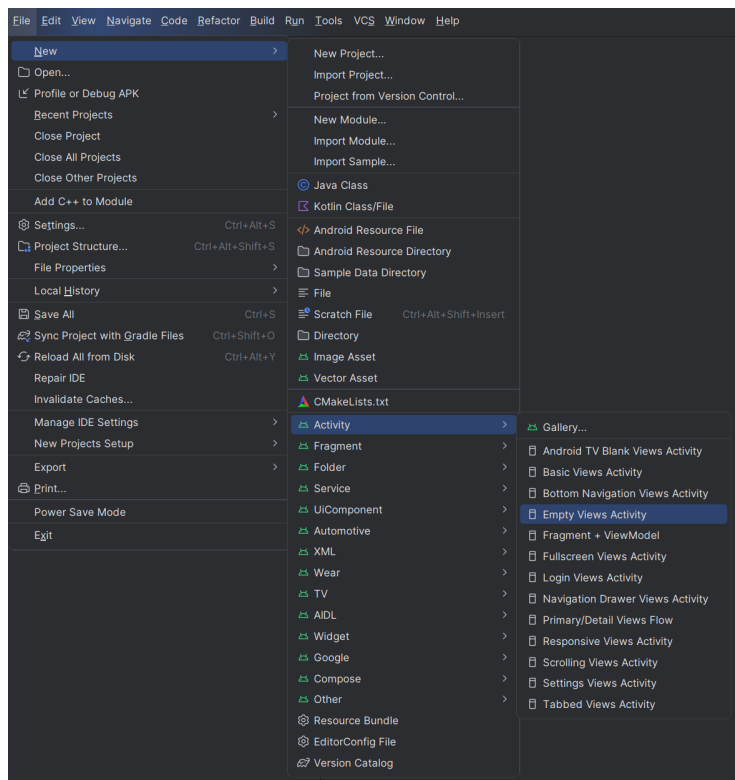
- **User Interface:** An activity provides the window in which the app draws its UI. This window typically fills the screen but can also be smaller and float on top of other windows.
- **Lifecycle:** Activities go through a lifecycle, which includes stages like onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy(). These stages help manage the activity's state as the user navigates through the app.
- **Multiple Activities:** Most apps consist of multiple activities, each representing a different screen or function. For example, an email app might have one activity for the inbox and another for composing a new email.
- **Interaction:** Activities can start other activities, both within the same app and in other apps. This allows for a seamless user experience across different parts of the app and even between different apps.

An **Intent** is a messaging object used to request an action from another app component. Intents facilitate communication between different components of an app, such as activities, services, and broadcast receivers. Here are the main types and uses of intents:

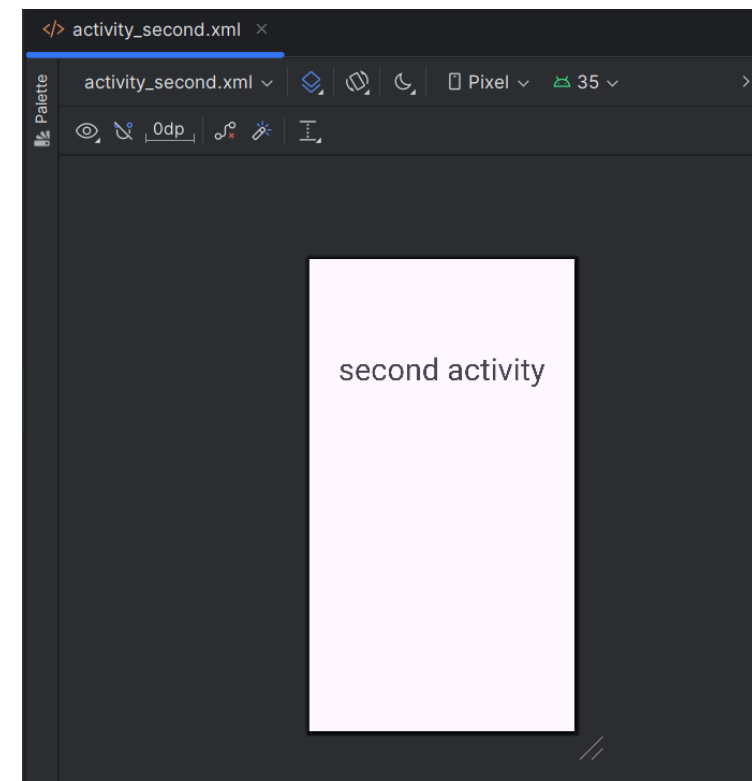
- **Explicit Intents:** These specify the exact component to start by using the component's class name. They are typically used to start a component within the same app.
- **Implicit Intents:** These do not specify a specific component. Instead, they declare a general action to perform, allowing any app that can handle the action to respond.

ACTIVITIES & INTENTS

Add a second activity:
New » Activity » Empty views activity

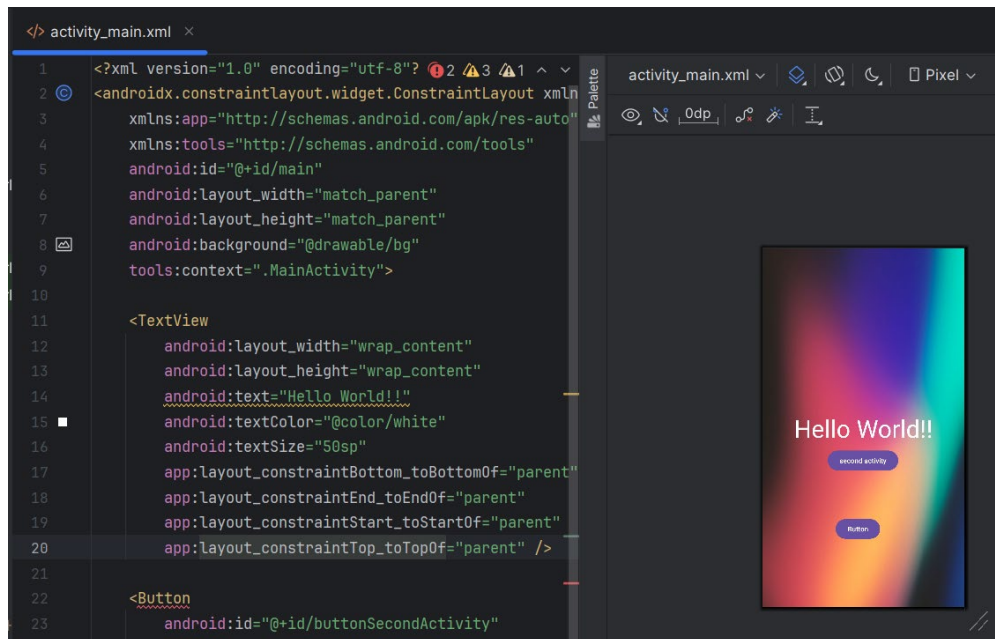


Add some content



ACTIVITIES & INTENTS

Add a button on main activity



Add an intent to change to the second

(on the onCreate method, import classes)

```
Button button2d=findViewById(R.id.button2);
button2d.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent=new Intent( packageContext: MainActivity.this, SecondActivity.class);
        startActivity(intent);
    }
});
```

Place correctly

(after the super.onCreate and setContentView)

ACTIVITIES & INTENTS

```
Button button2d=findViewById(R.id.button2);
button2d.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent=new Intent( packageContext: MainActivity.this, SecondActivity.class);
        startActivity(intent);
    }
});
```

ACTIVITIES & INTENTS

Add a button and an intent to another app (YouTube for instance, implicit)

```
Button button2 =findViewById(R.id.button2);
button2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(Intent.ACTION_VIEW);
        intent.setData(Uri.parse(uriString: "https://www.youtube.com/@PorFalarNoutraCoisa"));
        startActivity(intent);
    }
});
```

ACTIVITIES & INTENTS

Sending extra info on intent

```
Intent intent=new Intent( packageContext: MainActivity.this, SecondActivity.class);

//attach the key value pair using putExtra to this intent
String user_name = "Pedro";
intent.putExtra( name: "USER_NAME", user_name);

//starting the activity
startActivity(intent);
```

Get the extra info (on the second activity, writing to logcat)

```
//get the current intent
Intent intent = getIntent();

//get the attached extras from the intent
// we should use the same key as we used to attach the data.
String user_name = intent.getStringExtra( name: "USER_NAME");

Log.w( tag: "myApp", msg: "USER_NAME: " + user_name);
```

TOASTS



- A Toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. Toasts automatically disappear after a timeout.

```
Button button=findViewById(R.id.buttonSecondActivity);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent=new Intent(packageContext: MainActivity.this, SecondActivity.class);
        startActivity(intent);

        Toast toast=Toast.makeText(getApplicationContext(), text: "Hello toast",Toast.LENGTH_SHORT);
        toast.show();
    }
});
```

`Toast.makeText(getApplicationContext(), "Hello,World!", Toast.LENGTH_SHORT).show();`

(Import class)





MULȚUMESC

THANK YOU

OBRIGADO

SEE YOU TOMORROW!