PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO.
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS.

AED

BRUNO CAMARGO MANSO

GOIÂNIA, GO
2021

BRUNO CAMARGO MANSO

AED

Matéria: Desenho de Aplicativos para Dispositivos Móveis

Orientador: Fabricio Schlag

GOIÂNIA, GO

2021

Sumário:

O *Framework* Android

O Android é um sistema operacional que provê seu próprio Framework, segundo o site sobre desenvolvimento em Android:

> *— Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language —*
>
> *SOFTWARE ENGINEERING, 09/10/2011*

Ou seja, é uma pilha de softwares que incluem o sistema operacional, micro serviços para aplicações principais. Provê ferramentas e APIs necessárias para o desenvolvimento em sua plataforma usando a linguagem JAVA.

No presente trabalho, mostraremos como se desenvolve uma camada de apresentação e também quais são as maneiras de se persistir dados usando as ferramentas fornecidas por esse Sistema Operacional, por fim também será exposto aqui a melhor forma de se fazer um deploy de uma aplicação.

Importante ressaltar que o framework nativo do Android pode ser sobrescrito por outros frameworks que estão disponíveis no mercado, como o *Flutter* que usa a linguagem *Dart*, assim como o *React Native* que usa a linguagem *Typescript*. No presente trabalho exaltaremos apenas o *Framework* nativo do Android.

Android e Camada de Apresentação

A arquitetura MVVM — Model-View-ViewModel — é uma arquitetura otimizada originalmente criada para separar a interface do usuário da lógica de negócio utilizando amplamente o recurso de *Data Binding* (vinculação de dados).

> *— Aplicações arquitetadas desta forma têm uma camada **ViewModel** distinta que não possui dependências de sua interface de usuário —*
>
> *JOSÉ CARLOS MACORATTI, 2021*

Em Android o MVVM é dividido em *Activities* e *Fragments* para as *Views* e *ViewModels* e não devem conter lógicas e regras de negócios (OBJECTIVE, 07/01/2020).

```java
public class MainActivity extends AppCompatActivity {

    Button btn, btn_tela2, btn_sair;
    EditText editText;

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

        btn = findViewById(R.id.btn_sair);
        editText = findViewById(R.id.editText);
        btn_tela2 = findViewById(R.id.btn_tela2);
        btn_sair = findViewById(R.id.btn_sair);

        btn_sair.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                sair();
            }
        });

        btn_tela2.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {
               String aux = editText.getText().toString();

               Intent intent = new Intent(MainActivity.this, SecondScreen.class);
               intent.putExtra("message", aux);
               startActivity(intent);

           }
       });
    }
    public void setSecond (String aux) {
        Intent intent = new Intent(MainActivity.this, SecondScreen.class);
        intent.putExtra("message", aux);

        startActivity(intent);
    }

    public void sair() {
        finish();
    }
```

Acima temos uma implementação de uma atividade *MainActivity*, que instancia o método OnCreate servindo para fazer a vinculação de dados entre a *MainView* contendo elementos como *buttons* e *editTexts.*

```java
public class MainActivity extends AppCompatActivity {
    private PrimeiroFragment adaptador;
    private ViewPager viewPager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.atividade_principal);
        adaptador = new PrimeiroFragment(getSupportFragmentManager(),1);
        viewPager = findViewById(R.id.atividade);
        setupViewPager(viewPager);

    }
    protected void setupViewPager (ViewPager v) {
        PrimeiroFragment adaptador = new PrimeiroFragment(getSupportFragmentManager(), 1);
        adaptador.AddFragment(new Fragment1(),"fragment_1");
        adaptador.AddFragment(new Fragment2(),"fragment_2");
        v.setAdapter((adaptador));
    }
    public void setViewPager(int num) {
        viewPager.setCurrentItem(num);
    }
}
```

Acima, exemplificamos como as fragments são gerenciadas pela Atividade Principal. Abaixo um *Fragment* que pode ser acessado quando o movimento de deslizar para esquerda é feito. Deixando a primeira *Fragment* e indo para segunda *Fragment*.

```java
public class Fragment1 extends Fragment {
    Button btn_tela2, btn_fragmento_1, btn_fragmento_2;
    MediaPlayer som1;
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_1, container,false);
        btn_tela2 = view.findViewById(R.id.btn_tela2);
        btn_fragmento_1 = view.findViewById(R.id.btn_fragment_1);
        btn_fragmento_2 = view.findViewById(R.id.btn_fragment_2);
        som1 = MediaPlayer.create(getActivity(), R.raw.blop);

        btn_tela2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i = new Intent(getContext(), SegundaTela.class);
                startActivity(i);
//                finish();
            }
        });
        btn_fragmento_1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                ((MainActivity)getActivity()).setViewPager(0);
                som1.start();
            }
        });

        btn_fragmento_2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                ((MainActivity)getActivity()).setViewPager(1);
                som1.start();
            }
        });
        return view;
    }
}
```

Tais *Fragments* têm conteúdos diferentes e podem ser acessados de forma facilitada oferecendo interatividade e versatilidade ao usuário.

Android e Camada de Persistência

A persistência de dados são formas de armazenar dados evitando que estes se percam ao reiniciar uma aplicação. Existem diversas maneiras de se persistir dados em android. Cada maneira depende do objetivo e da arquitetura escolhida.

Shared Preferences

Salva pares de valores simples em strings, persiste de forma privada em um dicionário. Usado amplamente para manter configurações e também para manter sessões. Sua implementação segue a seguinte lógica abaixo:

```
SharedPreferences pref =
    PreferenceManager.getDefaultSharedPreferences(this);
String username = pref.getString("username", "n/a");
```

```
SharedPreferences pref =
    PreferenceManager.getDefaultSharedPreferences(this);
Editor edit = pref.edit();
edit.putString("username", "billy");
edit.putString("user_id", "65");
edit.commit();
```

O exemplo acima demonstra como deixar usuário e senha salvo no sistema, para que o usuário permaneça autenticado mesmo depois de fechar a aplicação.

Local Files

Como o termo já diz, é uma maneira de persistir dados no armazenamento interno ou externo ao dispositivo. Usado para *blobs* — arquivos PDF, por exemplo — ou até mesmo arquivos de *cache* também necessários para manter sessões entre outras coisas. A seguir, sua implementação:

```
// Use Activity method to create a file in the writeable directory
FileOutputStream fos = openFileOutput("filename", MODE_WORLD_WRITEABLE);
// Create buffered writer
BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(fos));
writer.write("Hi, I'm writing stuff");
writer.close();
```

```
BufferedReader input = null;
input = new BufferedReader(
new InputStreamReader(openFileInput("filename")));
String line;
StringBuffer buffer = new StringBuffer();
while ((line = input.readLine()) != null) {
  buffer.append(line + "\n");
}
String text = buffer.toString();
```

Os métodos acima implementados permitem não só o salvamento mas também a recuperação de dados. No exemplo temos, no primeiro bloco, a escrita de uma *string* no armazenamento usando o *BufferedWriter*, em forma de arquivo. Abaixo, utilizando o *BufferedReader* um laço de repetição que faz a leitura da frase anteriormente guardada.

SQLite

Persiste dados em tabelas de banco de dados, tal banco pode ser modelado especificamente para uma determinada aplicação. É utilizado para persistir dados complexos e interdependentes. Veremos, a seguir, sua implementação:

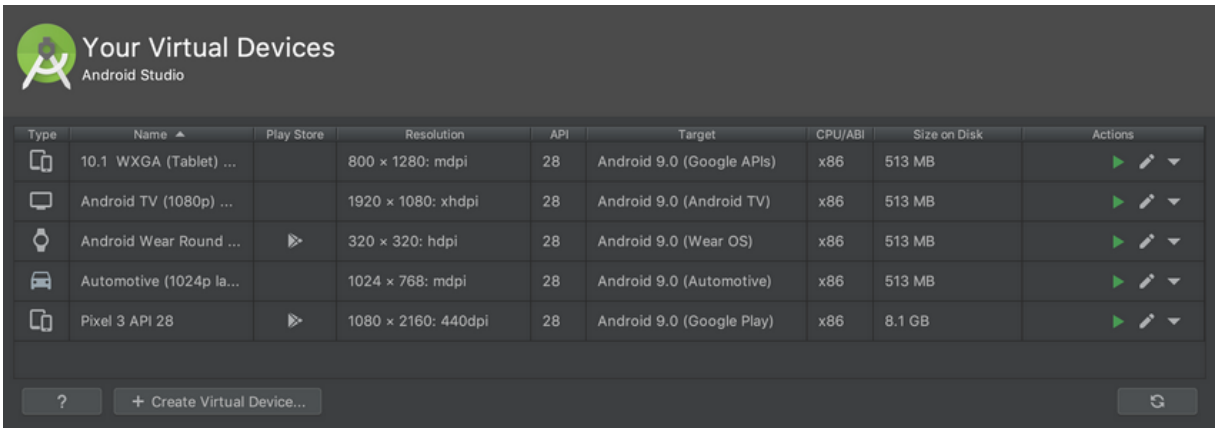
```
public class TodoItemDatabase extends SQLiteOpenHelper {
    public TodoItemDatabase(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    // This is where we need to write create table statements.
    // This is called when database is created.
    @Override
    public void onCreate(SQLiteDatabase db) {
        // SQL for creating the tables
    }

    // This method is called when database is upgraded like
    // modifying the table structure,
    // adding constraints to database, etc
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion,
        int newVersion) {
        // SQL for upgrading the tables
    }
}
```

SQLiteOpenHelper usa requisições, e pode atuar criando, carregando, atualizando ou deletando informações ou um conjunto de informações selecionadas por linhas de código com *strings* que são enviadas para o banco de dados.

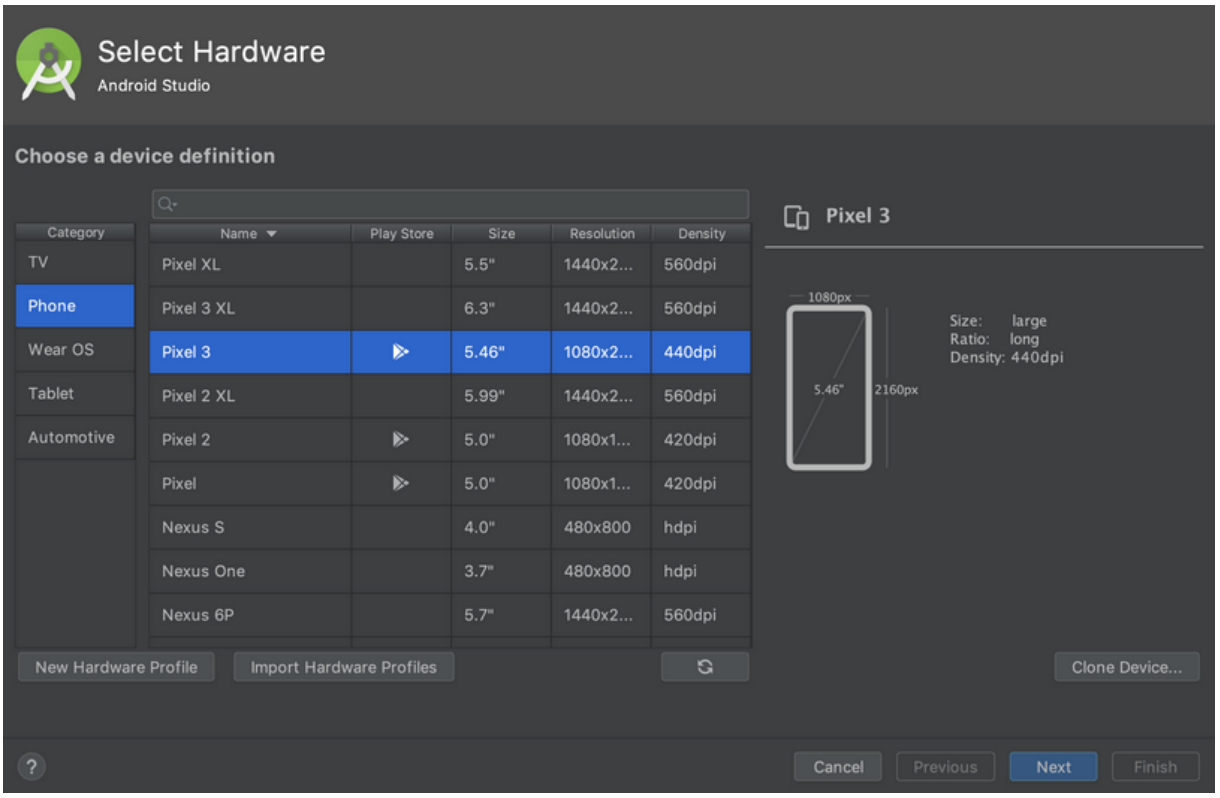
Deploying

Para ‘lançar’ o aplicativo recém criado (ANDROID DEVELOPERS, 2021), ou fazer seu *Deploy*, os seguintes passos são necessários:

- No menu, *Tools*
- Clique em *AVD Manager*, a seguinte tela aparecerá:

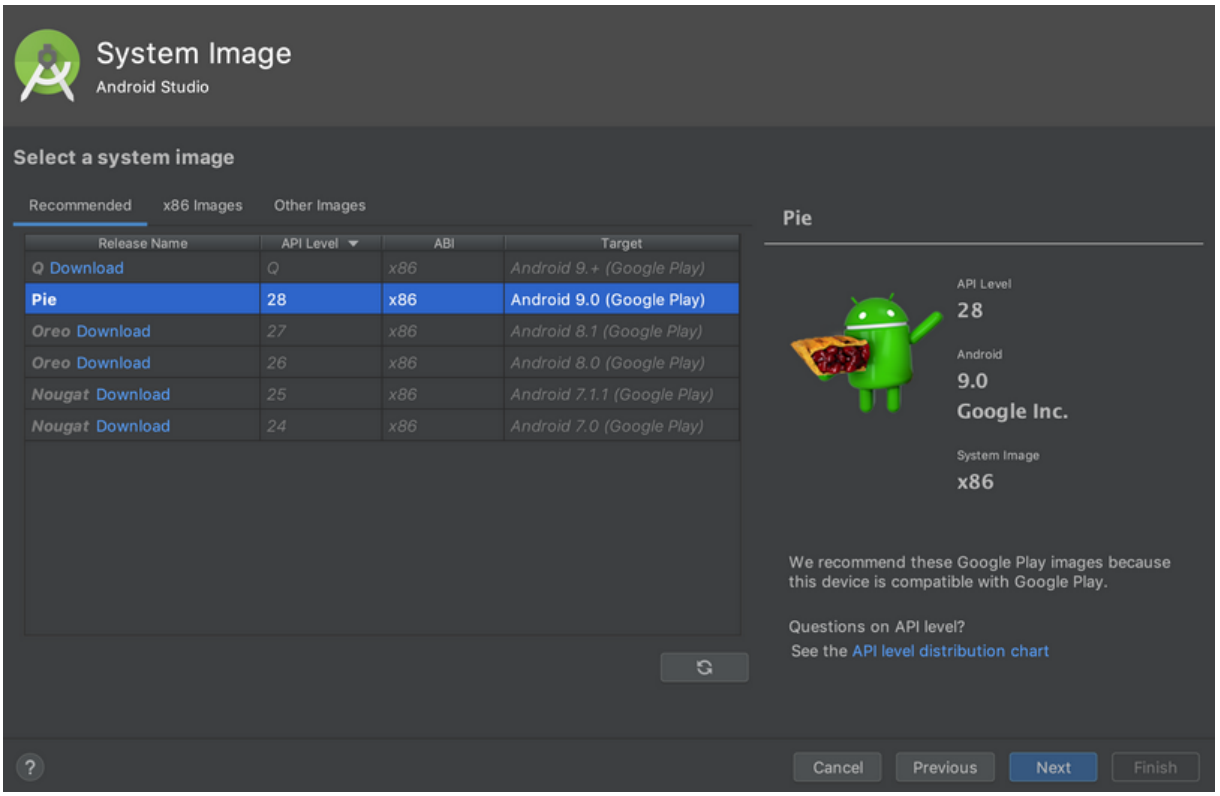


- A tela acima possivelmente aparecerá vazia
- Clique no botão *Create Virtual Device*
- Selecione o *Hardware* desejado e compatível ao seu ambiente

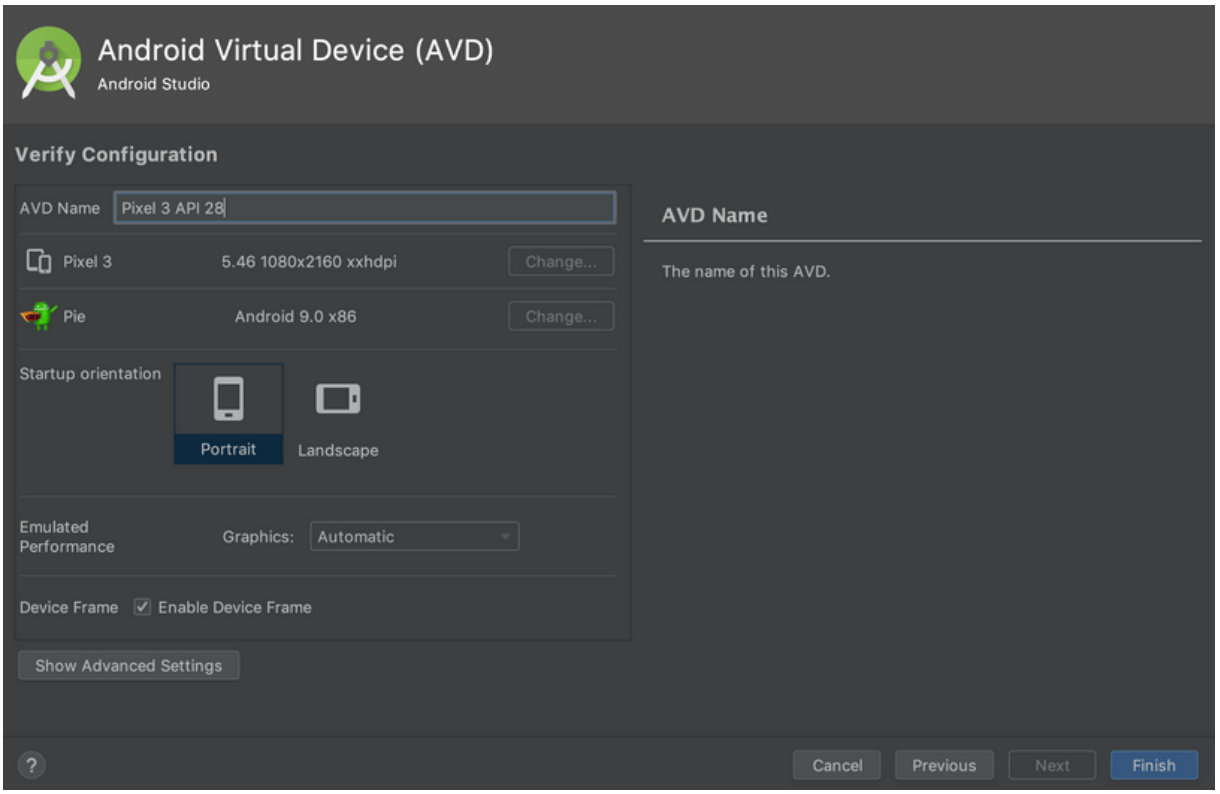


- O símbolo da *Play Store* aparecerá quando for suportado pelo dispositivo

- Em seguida selecione qual Sistema Operacional do dispositivo:



- Uma nova tela de configurações opcionais aparecerá:



- Nesse momento o Android Studio irá baixar e instalar o dispositivo e seu sistema
- Tudo pronto! Basta apertar o botão de play como ilustrado na figura abaixo:

Bibliografia

SOFTWARE ENGINEERING. **Is Android a language or a framework/platform?** 09/10/2011. Disponível em: <https://softwareengineering.stackexchange.com/questions/51769/is-android-a-language-or-a-framework-platform> Acesso em: 26 maio 2021.

CODEPATH. **Persisting Data to the Device**. 2021. Disponível em: <https://guides.codepath.com/android/Persisting-Data-to-the-Device>. Acesso em: 26 maio 2020.

OBJECTIVE. **Clean Architecture com MVVM: o que é, vantagens e como utilizar em aplicações Android**. 07/01/2020. Disponível em: <https://www.objective.com.br/insights/clean-architecture-com-mvvm/>. Acesso em: 26 maio 2021.

JOSÉ CARLOS MACORATTI. **Compreendendo o padrão MVVM : Model-View-ViewModel**. 2021. Disponível em: <http://www.macoratti.net/16/09/net_mvvm1.htm>. Acesso em: 26 maio 2021.

ANDROID DEVELOPERS. **Create and manage virtual devices**. 2021. Disponível em: <https://developer.android.com/studio/run/managing-avds>. Acesso em: 26 maio 2021.