ADS/GTI

ATIVIDE EXTRA AULA FINAL Q.A.

VALE 0,5 PONTO EXTRA

BÔNUS: essa atividade tem bônus 0,5 ponto que será aplicado na prova semestral

Objetivo: Q.A. - Testes em Mobile e CI/CD

FERRAMENTA:

ESPRESSO

Espresso é uma ferramenta poderosa para testes de interface de usuário (UI) em aplicativos Android. A ferramenta foi desenvolvida pelo Google. Ela faz parte do conjunto de ferramentas de teste do Android, projetada para facilitar a criação de testes de interface de usuário (UI) para aplicativos Android. O objetivo do Espresso é ajudar a escrever testes confiáveis.

Principais Características do Espresso
API Simples: A API do Espresso é pequena e fácil de aprender, permitindo que você escreva testes de forma fácil.
Sincronização Automática: Espresso gerencia automaticamente a sincronização com a UI do aplicativo, garantindo que as ações e asserções sejam realizadas apenas quando a UI estiver em um estado estável.
Flexibilidade: Ele oferece várias extensões, como espresso-web para suporte a WebView, espresso-intents para validação de intents, e espresso-contrib para ações adicionais como DatePicker.

ESTRUTURA:

Nosso exemplo será estruturado da seguinte forma:

1. **Setup do projeto no Android Studio**.
2. **Criação do layout de login** com dois campos de entrada e um botão.
3. **Criação do teste automatizado** usando Espresso.
4. **Integração com GitHub Actions** para execução automática dos testes.

SETUP DO PROJETO NO ANDROID STUDIO:

1. **Abra o Android Studio** e selecione "New Project" (Novo Projeto) na tela inicial.
2. **Escolha o Tipo de Projeto**:
    - Na tela de criação do novo projeto, você verá várias opções de templates. Selecione **"Empty Activity"**. Esse template cria uma Activity vazia com um layout básico, o que facilita a criação de novas telas sem configurações adicionais.

3. **Configurações do Projeto**:
    - Preencha as informações do projeto:

   Name: Insira o nome do projeto, por exemplo, LoginApp.
   Package Name: Esse será o identificador único do seu aplicativo.
   Save location: Escolha onde o projeto será salvo em seu computador.
   Language: Escolha **Java**.
   Minimum SDK: Selecione o nível mínimo de API que seu app irá suportar. Normalmente, o Android Studio sugere uma opção que permite a maioria dos dispositivos Android.
   BUILD: Escolha **Kotlin.**

4. **Finalize a Criação**:
    - Clique em **Finish** para que o Android Studio crie o projeto.
    - Depois disso, o Android Studio irá configurar o projeto com uma estrutura padrão e incluirá uma **Empty Activity** chamada MainActivity, com seu respectivo layout XML (activity_main.xml) já vinculado.

CRIANDO O LAYOUT DE LOGIN:

5. Crie um layout simples de tela de login (activity_main.xml) com dois campos de texto (nome de usuário e senha) e um botão de login;

Código XML para activity_main.xml

```
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:gravity="center"
   android:padding="16dp">

   <EditText
      android:id="@+id/editTextUsername"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Username" />

   <EditText
      android:id="@+id/editTextPassword"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Password"
      android:inputType="textPassword" />

   <Button
```

```
        android:id="@+id/buttonLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Login" />

    <TextView
        android:id="@+id/welcomeMessage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome!"
        android:visibility="gone" />
</LinearLayout>
```

CRIAÇÃO DO CÓDIGO DE LOGIN

O código abaixo verifica se o nome de usuário e a senha são "TesteUsuario" e "senha123". Caso correto, a mensagem de boas-vindas é exibida.

No MainActivity escreva o código abaixo.

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        EditText usernameField = findViewById(R.id.editTextUsername);
        EditText passwordField = findViewById(R.id.editTextPassword);
        Button loginButton = findViewById(R.id.buttonLogin);
        TextView welcomeMessage = findViewById(R.id.welcomeMessage);

        loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String username = usernameField.getText().toString();
                String password = passwordField.getText().toString();
```

```
            if (username.equals("TesteUsuario") && password.equals("senha123")) {
                welcomeMessage.setVisibility(View.VISIBLE);
            }
        }
    });
  }
}
```

CRIADO O TESTE COM ESPRESSO

Crie um teste com Espresso para verificar se o login funciona corretamente.

LoginTest.java

Crie um arquivo de teste em:
app/src/androidTest/java/[pacote_do_seu_app]/LoginTest.java:

```
import androidx.test.ext.junit.rules.ActivityScenarioRule;
import androidx.test.ext.junit.runners.AndroidJUnit4;
import androidx.test.espresso.action.ViewActions;
import androidx.test.espresso.matcher.ViewMatchers;

import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;

import static androidx.test.espresso.Espresso.onView;
import static androidx.test.espresso.assertion.ViewAssertions.matches;
import static androidx.test.espresso.matcher.ViewMatchers.withId;
import static androidx.test.espresso.matcher.ViewMatchers.withText;
import static androidx.test.espresso.matcher.ViewMatchers.isDisplayed;

@RunWith(AndroidJUnit4.class)
public class LoginTest {

    @Rule
    public ActivityScenarioRule<MainActivity> activityRule = new
ActivityScenarioRule<>(MainActivity.class);

    @Test
    public void testSuccessfulLogin() {
        // Digita o nome de usuário

onView(withId(R.id.editTextUsername)).perform(ViewActions.typeText("TesteUsuario")
, ViewActions.closeSoftKeyboard());
```

```
        // Digita a senha

onView(withId(R.id.editTextPassword)).perform(ViewActions.typeText("senha123"),
ViewActions.closeSoftKeyboard());

        // Clica no botão de login
        onView(withId(R.id.buttonLogin)).perform(ViewActions.click());

        // Verifica se a mensagem de boas-vindas está visível
        onView(withId(R.id.welcomeMessage)).check(matches(isDisplayed()));
    }
}
```

Esse teste usa o Espresso para simular as ações do usuário e verificar se a mensagem de boas-vindas aparece após o login bem-sucedido.

CONFIGURANDO GITHUB ACTIONS P/EXECUTAR TESTES COM ESPRESSO

6. Para configurar o GitHub Actions para rodar testes automatizados, você precisa adicionar um workflow no repositório do GitHub.

Crie um arquivo chamado android-test.yml em .github/workflows/:
.github/workflows/android-test.yml

```
name: Android CI

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
```

```
    - name: Set up JDK
      uses: actions/setup-java@v1
      with:
        java-version: '11'

    - name: Cache Gradle
      uses: actions/cache@v2
      with:
        path: |
          ~/.gradle/caches
          ~/.gradle/wrapper
        key: gradle-${{ runner.os }}-${{ hashFiles('**/*.gradle*', '**/gradle-wrapper.properties') }}
        restore-keys: |
          gradle-${{ runner.os }}

    - name: Set up Android SDK
      uses: android-actions/setup-android@v2
      with:
        api-level: 30
        build-tools: 30.0.3

    - name: Build the app
      run: ./gradlew build

    - name: Run Instrumented Tests
      run: ./gradlew connectedAndroidTest
```

7. Esse workflow irá:
   Configurar o JDK e o Android SDK.
   Compilar o app.
   Executar os testes instrumentados, incluindo o teste de login que criamos.