Agora vamos detalhar o **Passo 5**, focado na **interface de usuário e usabilidade** do aplicativo de barbearia. Este passo inclui a utilização de padrões de **Material Design**, aprimoramento da interface para melhor usabilidade e a implementação de uma **navegação fluida** entre telas.

5. **Interface e Usabilidade (Frontend com XML e Kotlin)**

5.1 Material Design: Usando Componentes Padrão

O **Material Design** é um guia de design desenvolvido pelo Google que foca em uma interface mais intuitiva, acessível e responsiva. Vamos implementar alguns componentes principais do Material Design, como **AppBar**, **Floating Action Button (FAB)**, e **BottomNavigationView**.

5.1.1 Dependências de Material Design no `build.gradle`

Antes de começar, vamos garantir que as dependências do Material Design estejam no arquivo `build.gradle`:

groovy

Copiar código

```
implementation 'com.google.android.material:material:1.8.0'
```

5.1.2 Criando um AppBar com Menu de Ações

O **AppBar** é a barra superior do aplicativo que pode conter ícones de navegação e ações específicas. Vamos configurá-la para exibir o nome do aplicativo e um menu de ações, como "Perfil" e "Sair".

Arquivo XML para o AppBar (`activity_main.xml`):

xml

Copiar código

```
<com.google.android.material.appbar.AppBarLayout android:layout_width="match_parent" android:layout_height="wrap_content"
android:theme="@style/ThemeOverlay.MaterialComponents.Dark.ActionBar"> <androidx.appcompat.widget.Toolbar
android:id="@+id/toolbar" android:layout_width="match_parent" android:layout_height="?attr/actionBarSize" android:background="?
attr/colorPrimary" android:elevation="4dp" app:title="BarbeariaApp" app:popupTheme="@style/ThemeOverlay.MaterialComponents.Light"
/> </com.google.android.material.appbar.AppBarLayout>
```

5.1.3 Definindo o Menu no AppBar

Crie um arquivo de menu em `res/menu/main_menu.xml` para definir as ações que estarão na AppBar:

xml

Copiar código

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"> <item android:id="@+id/action_profile" android:title="Perfil"
android:icon="@drawable/ic_profile" app:showAsAction="ifRoom" /> <item android:id="@+id/action_logout" android:title="Sair"
android:icon="@drawable/ic_logout" app:showAsAction="ifRoom" /> </menu>
```

5.1.4 Implementando o AppBar no `MainActivity.kt`

Agora, vamos implementar a lógica no Kotlin para lidar com o menu de ações.

kotlin

Copiar código

```
import android.os.Bundle import android.view.Menu import android.view.MenuItem import android.widget.Toast import
androidx.appcompat.app.AppCompatActivity import androidx.appcompat.widget.Toolbar class MainActivity : AppCompatActivity() {
override fun onCreate(savedInstanceState: Bundle?) { super.onCreate(savedInstanceState) setContentView(R.layout.activity_main)
val toolbar: Toolbar = findViewById(R.id.toolbar) setSupportActionBar(toolbar) } override fun onCreateOptionsMenu(menu: Menu?):
Boolean { menuInflater.inflate(R.menu.main_menu, menu) return true } override fun onOptionsItemSelected(item: MenuItem): Boolean
{ return when (item.itemId) { R.id.action_profile -> { // Navegar para a tela de perfil Toast.makeText(this, "Perfil
selecionado", Toast.LENGTH_SHORT).show() true } R.id.action_logout -> { // Lógica de logout Toast.makeText(this, "Saindo...",
Toast.LENGTH_SHORT).show() true } else -> super.onOptionsItemSelected(item) } } }
```

Explicação:

- O **AppBar** foi configurado com dois itens: "Perfil" e "Sair".
- O menu é inflado no método `onCreateOptionsMenu` e cada item é tratado no método `onOptionsItemSelected`.

5.2 **Implementando o BottomNavigationView**

O **BottomNavigationView** é usado para navegar entre as principais seções do aplicativo, como "Agendamentos", "Histórico" e "Perfil".

Arquivo XML para o BottomNavigationView (`activity_main.xml`):

Adicione o **BottomNavigationView** logo abaixo do `ConstraintLayout` ou outro layout principal:

xml

Copiar código

```
<com.google.android.material.bottomnavigation.BottomNavigationView android:id="@+id/bottom_navigation"
android:layout_width="match_parent" android:layout_height="wrap_content" android:layout_alignParentBottom="true"
android:background="?android:attr/windowBackground" app:menu="@menu/bottom_nav_menu" />
```

Menu para o BottomNavigation (`res/menu/bottom_nav_menu.xml`):

xml

Copiar código

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"> <item android:id="@+id/navigation_agendamentos"
android:icon="@drawable/ic_agendamentos" android:title="Agendamentos" /> <item android:id="@+id/navigation_historico"
android:icon="@drawable/ic_historico" android:title="Histórico" /> <item android:id="@+id/navigation_perfil"
android:icon="@drawable/ic_perfil" android:title="Perfil" /> </menu>
```

Implementando a Navegação no `MainActivity.kt` :

kotlin

Copiar código

```
import android.os.Bundle import androidx.appcompat.app.AppCompatActivity import
com.google.android.material.bottomnavigation.BottomNavigationView class MainActivity : AppCompatActivity() { override fun
onCreate(savedInstanceState: Bundle?) { super.onCreate(savedInstanceState) setContentView(R.layout.activity_main) val
bottomNavigationView = findViewById<BottomNavigationView>(R.id.bottom_navigation)
bottomNavigationView.setOnNavigationItemSelectedListener { item -> when (item.itemId) { R.id.navigation_agendamentos -> { //
Navegar para tela de agendamentos true } R.id.navigation_historico -> { // Navegar para tela de histórico true }
R.id.navigation_perfil -> { // Navegar para tela de perfil true } else -> false } } } }
```

Explicação:

- Cada item do `BottomNavigationView` corresponde a uma seção principal do aplicativo.
- Quando um item é selecionado, a navegação para a respectiva tela é realizada.

5.3 **Implementação do Floating Action Button (FAB)**

O **Floating Action Button (FAB)** é um botão destacado para uma ação principal, como "Agendar Serviço". Vamos implementá-lo na tela principal.

Arquivo XML para o FAB (`activity_main.xml`):

xml Copiar código

```xml
<com.google.android.material.floatingactionbutton.FloatingActionButton android:id="@+id/fab_agendar"
android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_gravity="end|bottom"
android:layout_margin="16dp" android:src="@drawable/ic_agendar" android:contentDescription="Agendar Serviço" />
```

Implementação da Ação do FAB no `MainActivity.kt` :

kotlin Copiar código

```kotlin
import android.content.Intent import android.os.Bundle import
com.google.android.material.floatingactionbutton.FloatingActionButton class MainActivity : AppCompatActivity() { override fun
onCreate(savedInstanceState: Bundle?) { super.onCreate(savedInstanceState) setContentView(R.layout.activity_main) val fabAgendar
= findViewById<FloatingActionButton>(R.id.fab_agendar) fabAgendar.setOnClickListener { // Navegar para tela de agendamento val
intent = Intent(this, ScheduleActivity::class.java) startActivity(intent) } } }
```

Explicação:

- O **FAB** é posicionado na parte inferior direita da tela.
- Ao clicar no botão, o usuário é direcionado para a tela de agendamento de serviços.

5.4 **Aprimoramento de UI/UX: Adicionando Animações**

Vamos adicionar algumas animações simples para melhorar a transição entre telas e a interação com o **Floating Action Button**.

5.4.1 Animação para FAB (`res/anim/fab_slide_in.xml`):

xml Copiar código

```xml
<set xmlns:android="http://schemas.android.com/apk/res/android"> <translate android:fromYDelta="100%" android:toYDelta="0%"
android:duration="500" /> </set>
```

5.4.2 Aplicar a animação no `MainActivity.kt` :

kotlin

Copiar código

```kotlin
import android.view.animation.AnimationUtils class MainActivity : AppCompatActivity() { override fun onCreate(savedInstanceState: Bundle?) { super.onCreate(savedInstanceState) setContentView(R.layout.activity_main) val fabAgendar = findViewById<FloatingActionButton>(R.id.fab_agendar) val animation = AnimationUtils.loadAnimation(this, R.anim.fab_slide_in) fabAgendar.startAnimation(animation) fabAgendar.setOnClickListener { val intent = Intent(this, ScheduleActivity::class.java) startActivity(intent) } } }
```

Explicação:

- A animação faz o FAB deslizar de baixo para cima quando a tela é carregada.
- Usar animações sutis melhora a experiência do usuário e torna a interface mais agradável.

5.5 **Implementação de Layout Responsivo**

Garantir que a interface seja responsiva e funcione bem em diferentes tamanhos de tela é essencial. Para isso, utilize dimensões e espaçamentos proporcionais no XML, como `dp` e `sp` para garantir que o layout se adapte a dispositivos variados.

Exemplo de uso de dimensões no XML (`dimens.xml`):

xml

Copiar código

```xml
<resources> <dimen name="padding_small">8dp</dimen> <dimen name="padding_medium">16dp</dimen> <dimen name="padding_large">24dp</dimen> </resources>
```

Depois, aplique essas dimensões no layout:

xml

Copiar código

```xml
<Button android:id="@+id/agendarButton" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_margin="@dimen/padding_medium" android:text="Agendar Serviço" />
```

Conclusão

No **Passo 5**, melhoramos a interface do aplicativo utilizando componentes do **Material Design** como **AppBar**, **FAB**, e **BottomNavigationView** para criar uma navegação fluida e intuitiva. Também aprimoramos a usabilidade com animações e garantimos que o layout seja responsivo para diferentes dispositivos. Com essas melhorias, o aplicativo de barbearia oferecerá uma experiência mais moderna e agradável para os usuários.