

Aqui está o **Passo 7 detalhado**, que envolve testar o aplicativo **Tech Burguer** para garantir que ele funcione corretamente em diferentes cenários, resolvendo possíveis problemas e preparando o app para publicação.

7. Testar o Aplicativo

O processo de teste do aplicativo é crucial para garantir que todas as funcionalidades estejam funcionando corretamente. Vamos focar em testar a usabilidade, a compatibilidade com diferentes dispositivos e a estabilidade do aplicativo.

Passo 7.1: Testar o Aplicativo no Emulador

1. Iniciar o Emulador Android:

- No Android Studio, vá para o menu **Tools > AVD Manager**.
- Crie um novo emulador Android, se necessário, escolhendo um dispositivo comum, como **Pixel 3a**, com **API 30 (Android 11)**.
- Inicie o emulador.

2. Executar o Aplicativo no Emulador:

- Clique no botão **Run** (ícone de play) na barra de ferramentas do Android Studio.
- Selecione o emulador como destino de execução.
- O aplicativo **Tech Burguer** será instalado e iniciado no emulador.

3. Verificar as Funcionalidades:

- **Navegação:** Verifique se você pode navegar entre as telas principais, como **Menu de Hambúrgueres**, **Carrinho**, **Pagamento** e **Login**.
- **Adicionar ao Carrinho:** Adicione hambúrgueres ao carrinho e verifique se a quantidade e o valor total são atualizados corretamente.
- **Remover Itens do Carrinho:** Teste a função de limpar o carrinho e verifique se o total é zerado.
- **Finalizar Pedido:** Verifique se o pedido pode ser finalizado corretamente, limpando o carrinho após a conclusão.
- **Login e Registro:** Teste o login e registro de novos usuários com o Firebase Authentication.
- **Pagamento:** Teste o fluxo de pagamento (simulado ou em produção, dependendo da implementação).

4. Verificar Mensagens de Erro:

- Certifique-se de que o aplicativo exiba mensagens apropriadas quando ocorrerem erros (por exemplo, ao tentar finalizar um pedido com o carrinho vazio ou inserir um cartão inválido).

Passo 7.2: Testar em Dispositivos Reais

Embora o emulador forneça um bom ambiente de teste, é importante verificar o comportamento do aplicativo em dispositivos reais para garantir que tudo funcione corretamente em termos de desempenho e usabilidade.

1. Conectar um Dispositivo Android Real:

- Conecte um dispositivo Android real ao computador via USB.
- Ative o modo **Depuração USB** nas opções de desenvolvedor do dispositivo.

2. Executar o Aplicativo no Dispositivo:

- No Android Studio, clique em **Run** novamente e selecione o dispositivo real como destino de execução.
- Verifique o desempenho do aplicativo no dispositivo e teste as mesmas funcionalidades mencionadas no **Passo 7.1**.

Passo 7.3: Testar a Responsividade para Diferentes Telas

1. Testar em Diferentes Tamanhos de Tela:

- No **AVD Manager**, crie emuladores com diferentes tamanhos de tela e resoluções (por exemplo, um **tablet** com tela grande e um **dispositivo menor** como o **Pixel 3a**).
- Teste o layout do aplicativo para garantir que os elementos da interface, como botões, texto e imagens, sejam exibidos corretamente e não fiquem distorcidos.

2. Verificar a Responsividade do Layout:

- Verifique se a interface é responsiva e adaptável a diferentes resoluções, orientações (paisagem e retrato) e tamanhos de tela.
- Certifique-se de que os componentes do layout, como o **RecyclerView** e os **botões**, são redimensionados adequadamente.

Passo 7.4: Testar a Estabilidade do Aplicativo

1. Testar a Estabilidade com Várias Interações:

- Teste interações rápidas e repetidas para verificar a estabilidade do aplicativo, como adicionar/remover itens do carrinho repetidamente ou alternar entre telas rapidamente.

2. Simular Conexão Lenta ou Ausente:

- Desative a conexão de internet ou simule uma conexão lenta no emulador/dispositivo.

- Verifique como o aplicativo lida com a falta de conectividade, especialmente em funcionalidades que dependem de rede, como **Login** e **Pagamento**.

3. Verificar o Comportamento Após Fechar e Reabrir o App:

- Feche o aplicativo completamente e reabra-o. Verifique se o carrinho de compras ainda contém os itens adicionados (se aplicável) e se o login do usuário é mantido.

Passo 7.5: Testar a Funcionalidade de Pagamento

1. Testar o Fluxo de Pagamento:

- Se o ambiente de pagamento está em modo de produção, execute uma transação real utilizando cartões de teste fornecidos pela **Stripe** (ou outro provedor de pagamento).
- Se estiver em modo de teste, use os cartões de teste fornecidos pelo Stripe para garantir que a transação seja processada corretamente.

2. Testar Cartões Inválidos:

- Tente inserir um cartão inválido (por exemplo, um número de cartão incorreto ou uma data de validade passada) e verifique se o sistema retorna uma mensagem de erro apropriada.

Passo 7.6: Verificar Desempenho e Consumo de Recursos

1. Monitorar o Consumo de Memória e CPU:

- No **Android Studio**, use o **Android Profiler** para monitorar o uso de CPU e memória enquanto o aplicativo está sendo executado.
- Verifique se o aplicativo não está consumindo muitos recursos, especialmente ao interagir com o carrinho ou processar pagamentos.

2. Testar o Consumo de Bateria:

- Verifique se o aplicativo tem um impacto mínimo na bateria, especialmente durante a execução em segundo plano ou em espera prolongada.

Passo 7.7: Verificar Logs e Exceções

1. Monitorar Logs de Erros:

- Use o **Logcat** no Android Studio para monitorar qualquer erro ou exceção que possa ocorrer durante a execução do aplicativo.
- Corrija possíveis falhas que forem detectadas.

2. Tratamento de Exceções:

- Verifique se todas as exceções que possam ocorrer, como erros de rede ou falhas no processamento de pagamento, estão sendo tratadas corretamente e exibem mensagens apropriadas ao usuário.

Exemplos de Código para Testes

Teste de Funcionalidade de Carrinho:


kotlin

 Copiar código

```
// Adicionar um item ao carrinho e verificar se o total é atualizado corretamente fun testAdicionarAoCarrinho() {  
CartManager.addBurgerToCart(Burger("Cheeseburger", "Descrição", 19.99, R.drawable.ic_burger)) val total =  
CartManager.calculateCartTotal() assert(total == 19.99) { "Erro ao calcular o total do carrinho" } } // Testar se o carrinho é  
limpo corretamente fun testLimparCarrinho() { CartManager.clearCart() val total = CartManager.calculateCartTotal() assert(total  
== 0.0) { "Erro ao limpar o carrinho" } }
```

Teste de Funcionalidade de Login:

kotlin

 Copiar código

```
// Teste de login com Firebase Authentication (em ambiente de teste) fun testLogin(email: String, password: String) { val auth =  
FirebaseAuth.getInstance() auth.signInWithEmailAndPassword(email, password) .addOnCompleteListener { task ->  
assert(task.isSuccessful) { "Falha no login: ${task.exception?.message}" } } }
```

Com isso, o **Passo 7** está completo. Agora, você já passou por várias fases de teste do aplicativo **Tech Burguer** para garantir que ele funcione corretamente em diferentes dispositivos, tamanhos de tela e cenários de uso.