


Aqui está o **Passo 3 detalhado**, onde criamos o modelo de dados para o aplicativo **Tech Burger** e definimos as classes **Burger** e **Order**, necessárias para estruturar o menu de hambúrgueres e gerenciar os pedidos.

3. Criar o Modelo de Dados (Burger e Pedido)

Passo 3.1: Criar a Classe `Burger` (Modelo dos Hambúrgueres)

1. Na pasta `java/com.example.techburger`, crie um novo arquivo Kotlin chamado `Burger.kt`.
2. Adicione o seguinte código para a classe `Burger`, que irá representar cada hambúrguer no menu:

kotlin

 Copiar código

```
package com.example.techburger // Classe que representa um hambúrguer no menu
data class Burger( val name: String, // Nome do hambúrguer
val description: String, // Descrição do hambúrguer
val price: Double, // Preço do hambúrguer
val imageResId: Int // ID do recurso da imagem (drawable) )
```

Explicação:

- **name:** O nome do hambúrguer (por exemplo, "Cheeseburger").
- **description:** Uma breve descrição do hambúrguer.
- **price:** O preço do hambúrguer em formato decimal.
- **imageResId:** O ID da imagem associada ao hambúrguer (você irá usar imagens armazenadas na pasta `drawable`).

Nota: Certifique-se de ter imagens na pasta `drawable` para cada hambúrguer que deseja exibir no menu.

Passo 3.2: Criar a Classe `Order` (Modelo para Gerenciar Pedidos)

A classe `Order` vai representar os itens adicionados ao carrinho, junto com a quantidade selecionada pelo usuário.

1. Na mesma pasta `java/com.example.techburger`, crie um novo arquivo Kotlin chamado `Order.kt`.
2. Adicione o seguinte código para a classe `Order`:

kotlin

 Copiar código

```
package com.example.techburger // Classe que representa um pedido de um hambúrguer
data class Order( val burger: Burger, // O hambúrguer que está sendo pedido
var quantity: Int // Quantidade desse hambúrguer no pedido ) { // Função para calcular o
preço total do pedido
fun calculateTotal(): Double { return burger.price * quantity }
}
```

Explicação:


- **burger:** A instância do hambúrguer que o cliente está pedindo.
- **quantity:** A quantidade desse hambúrguer no pedido.
- **calculateTotal():** Um método que multiplica o preço do hambúrguer pela quantidade para calcular o total do pedido.

Passo 3.3: Adicionar o Método para Manipular o Carrinho de Pedidos

Agora que temos as classes `Burger` e `Order`, vamos criar uma lista que armazenará os pedidos no carrinho e métodos para adicionar, remover e calcular o total do carrinho.

1. Crie uma nova classe chamada `CartManager.kt` na pasta `java/com.example.techburger`.
2. Adicione o seguinte código para gerenciar os pedidos no carrinho:

kotlin

 Copiar código

```
package com.example.techburger // Classe responsável por gerenciar o carrinho de
pedidos object CartManager { private val orders = mutableListOf<Order>() // Lista de
pedidos // Função para adicionar um hambúrguer ao carrinho fun addBurgerToCart(burger:
Burger) { val existingOrder = orders.find { it.burger.name == burger.name } if
(existingOrder != null) { // Se o hambúrguer já estiver no carrinho, aumenta a
quantidade existingOrder.quantity++ } else { // Se for um novo hambúrguer, adiciona ao
carrinho com quantidade 1 orders.add(Order(burger, 1)) } } // Função para remover um
hambúrguer do carrinho fun removeBurgerFromCart(burger: Burger) { val orderToRemove =
orders.find { it.burger.name == burger.name } if (orderToRemove != null) {
orders.remove(orderToRemove) } } // Função para calcular o total do carrinho fun
calculateCartTotal(): Double { return orders.sumOf { it.calculateTotal() } } // Função
para obter a lista de pedidos fun getOrders(): List<Order> { return orders } // Função
para limpar o carrinho fun clearCart() { orders.clear() } }
```

Explicação:

- **orders:** Uma lista que armazena os pedidos atuais.
- **addBurgerToCart(burger: Burger):** Adiciona um hambúrguer ao carrinho. Se o hambúrguer já existir no carrinho, a quantidade é aumentada.
- **removeBurgerFromCart(burger: Burger):** Remove um hambúrguer do carrinho.
- **calculateCartTotal():** Calcula o valor total do carrinho somando o total de cada pedido.
- **getOrders():** Retorna a lista atual de pedidos no carrinho.
- **clearCart():** Limpa todos os itens do carrinho.

Passo 3.4: Conectar o Carrinho à Interface no `MainActivity.kt`

Agora que temos o gerenciamento do carrinho, podemos conectá-lo aos botões da interface principal (Ver Carrinho e Finalizar Pedido) no arquivo `MainActivity.kt`.

1. Abra o arquivo `MainActivity.kt` e adicione a seguinte lógica ao método `onCreate`:

```
// Configurar Adapter burgerAdapter = BurgerAdapter(burgerList) { burger -> //
Adicionar o hambúrguer ao carrinho quando for clicado
CartManager.addBurgerToCart(burger) Toast.makeText(this, "${burger.name} adicionado ao
carrinho", Toast.LENGTH_SHORT).show() } // Configurar botões
viewCartButton.setOnClickListener { val cartItems = CartManager.getOrders() if
(cartItems.isEmpty()) { Toast.makeText(this, "Carrinho está vazio",
Toast.LENGTH_SHORT).show() } else { val total = CartManager.calculateCartTotal()
Toast.makeText(this, "Total do carrinho: R$ %.2f".format(total),
Toast.LENGTH_SHORT).show() } } checkoutButton.setOnClickListener { val total =
CartManager.calculateCartTotal() if (total > 0) { Toast.makeText(this, "Pedido
finalizado! Total: R$ %.2f".format(total), Toast.LENGTH_SHORT).show()
CartManager.clearCart() // Limpar o carrinho após finalizar o pedido } else {
Toast.makeText(this, "Carrinho está vazio", Toast.LENGTH_SHORT).show() } }
```

Explicação:

- Quando o usuário clica em um hambúrguer, ele é adicionado ao carrinho através do método `CartManager.addBurgerToCart()`.
- Ao clicar no botão "Ver Carrinho", o aplicativo exibe o total atual do carrinho. Se o carrinho estiver vazio, uma mensagem correspondente é exibida.
- O botão "Finalizar Pedido" calcula o total e, se houver itens no carrinho, exibe o total final e limpa o carrinho.

Com isso, o **Passo 3** está completo. Agora, você tem as classes **Burger** e **Order** para modelar os hambúrgueres e os pedidos, além de um gerenciador de carrinho funcional que interage com a interface principal do aplicativo **Tech Burger**.