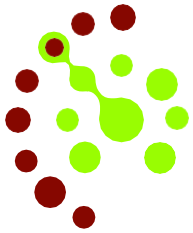


IPBeja

INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão Curso de Tecnologias Web e Dispositivos Moveis

Gestão de pedidos de um restaurante

Afonso Moisão (25912) e Tiago Marcos (25252)

Beja, 13 de julho de 2024

INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão Curso

de Tecnologias Web e Dispositivos Moveis

Gestão de pedidos de um restaurante

Afonso Moisão (25912) e Tiago Marcos (25252)

Orientado por:
Luis Rosário, João Paulo Trindade, Henrique Água-Doce e Luis Garcia,
IPBeja

Relatório de projeto
final

2024

Resumo

A aplicação de gestão de pedidos de restaurante desenvolvida neste projeto é uma solução abrangente para otimizar as operações internas de um restaurante. Utilizando Kotlin e Android Studio, a aplicação oferece uma interface intuitiva e prática, permitindo que os funcionários adicionem itens aos pedidos, calculem contas automaticamente e façam uma gestão das mesas de maneira eficiente. A base de dados robusta utilizada assegura a gestão segura e eficiente de informações, desde pedidos a faturação diária. A estética e design foram aprimorados com o uso do Adobe Illustrator, garantindo uma experiência visual atraente e funcional. Com funcionalidades que incluem gestão de pedidos em tempo real, visualização gráfica do layout das mesas e controle de faturação diária, a aplicação proporciona uma solução moderna e eficiente para restaurantes.

Palavras-chave: gestão de pedidos, restaurante, Kotlin, Android Studio, base de dados, Adobe Illustrator, faturamento diário, visualização gráfica.

Agradecimentos

Em primeiro lugar, gostaríamos de expressar nossa profunda gratidão a todos os formadores que nos acompanharam ao longo deste ano de curso. Sua dedicação, paciência e conhecimento foram fundamentais para nosso crescimento e aprendizado. Cada projeto realizado, cada desafio superado e cada nova habilidade adquirida foram frutos do empenho e do apoio inestimável que recebemos de vocês.

Agradecemos ao coordenador do curso, Luís Rosário por toda a disponibilidade e apoio na realização desde projeto.

Queremos estender nossos sinceros agradecimentos aos nossos amigos e colegas pelo incentivo e disponibilidade. Vocês fizeram deste ano uma experiência mais enriquecedora e significativa.

Por fim, mas não menos importante, expressamos nossa gratidão às nossas famílias, por todo o apoio e paciência com que acompanharam neste percurso.

Índice

Índice de imagens

Capítulo 1

Introdução

A nossa aplicação de gestão de pedidos de restaurante é uma solução abrangente e eficiente para otimizar as operações internas. Desenvolvida com o objetivo de facilitar o fluxo de trabalho dos funcionários, ela oferece uma experiência intuitiva e prática dentro do contexto do restaurante.

A aplicação possui menus organizados com elementos visuais claros e de fácil compreensão, garantindo que os funcionários possam navegar pelo sistema com facilidade. Através de uma interface intuitiva, é possível adicionar itens ao pedido de forma simples e rápida, inserindo apenas o número da mesa. A conta é calculada automaticamente com base nos pedidos da mesa selecionada, otimizando o processo de faturamento.

Para suportar essas funcionalidades, utilizamos uma base de dados robusta para armazenar toda a ementa do restaurante, bem como todos os movimentos, desde pedidos a contas. Esta base de dados permite uma gestão eficiente e segura das informações, garantindo que todas as transações sejam registadas e processadas corretamente.

A aplicação foi desenvolvida na linguagem Kotlin, utilizando o software Android Studio, o que nos permitiu criar uma solução moderna e eficiente para dispositivos móveis. A escolha dessas tecnologias assegura uma performance otimizada e uma experiência de utilização aprimorada, facilitando a adoção e utilização da aplicação pelos funcionários do restaurante.

Por fim para a realização de toda estética desde o protótipo da estrutura da aplicação a criação do logotipo, ícones e imagens foi utlizado o software de edição Adobe Illustrator.

Enquadramento teórico

Linguagens utilizadas no projeto

- Kotlin

Kotlin é uma linguagem de programação moderna criada pela JetBrains, conhecida por sua sintaxe concisa, segurança e compatibilidade total com Java. Lançada oficialmente em 2011, Kotlin foi projetada para reduzir o código boilerplate e melhorar a legibilidade, tornando o desenvolvimento mais eficiente.

A interoperabilidade com Java é um grande destaque de Kotlin, permitindo que os desenvolvedores integrem facilmente bibliotecas Java e migrem projetos existentes para Kotlin sem complicações. Suportada de maneira robusta no Android Studio e outras IDEs modernas, Kotlin oferece ferramentas avançadas de desenvolvimento, incluindo auto-completar, refatoração e depuração, facilitando a vida dos desenvolvedores.

Além de ser amplamente utilizada no desenvolvimento Android, Kotlin também se adapta a outras áreas como desenvolvimento web, servidor e multiplataforma, permitindo o compartilhamento de código entre diferentes plataformas. Desde que foi adotada oficialmente pelo Google como uma linguagem de primeira classe para desenvolvimento Android em 2017, Kotlin tem crescido rapidamente em popularidade, apoiada por uma comunidade de desenvolvedores ativa e em expansão.

Softwares utilizados no projeto

- ### Android Studio

Android Studio é o ambiente de desenvolvimento integrado (IDE) oficial para a criação de aplicativos Android. Lançado pela Google em 2013, é baseado no IntelliJ IDEA da JetBrains, oferecendo um conjunto abrangente de ferramentas e recursos para facilitar o desenvolvimento de aplicações Android.

Android Studio inclui um editor de código inteligente com destaque de sintaxe, autocompletar, e suporte para refatoração, o que ajuda os desenvolvedores a escrever e otimizar o código de maneira eficiente.
O IDE oferece um emulador de Android robusto que permite testar aplicativos em uma variedade de dispositivos virtuais com diferentes configurações e versões do sistema operacional, sem a necessidade de dispositivos físicos. Isso acelera o processo de desenvolvimento e depuração
Outra característica importante do Android Studio é o suporte integrado para o sistema de build Gradle, que automatiza a compilação, teste e implantação de aplicativos, facilitando a gestão de dependências e garantindo builds consistentes.

Com atualizações regulares e suporte para as mais recentes APIs do Android, Android Studio é uma ferramenta essencial para desenvolvedores que desejam criar aplicativos Android de alta qualidade e manter-se atualizados com as melhores práticas e novas tecnologias no ecossistema Android.

Softwares utilizados no projeto

- Adobe illustrator

Adobe Illustrator é um software de design gráfico vetorial amplamente utilizado por profissionais para criar uma variedade de elementos visuais, incluindo ilustrações, logotipos, ícones, gráficos e layouts complexos. Lançado pela Adobe Systems em 1987, Illustrator é conhecido por suas poderosas ferramentas de desenho que permitem a criação de gráficos escaláveis, ou seja, que podem ser redimensionados sem perda de qualidade.

Illustrator oferece uma vasta gama de recursos, como ferramentas de desenho precisas, manipulação de vetores, tipografia avançada e suporte para múltiplas pranchetas, facilitando o trabalho em projetos complexos. Com sua integração com outros produtos da Adobe, como Photoshop e InDesign, ele permite fluxos de trabalho eficientes e consistentes.

Utilizado em várias indústrias, desde design gráfico e editorial até web design e animação, Adobe Illustrator é essencial para quem busca criar gráficos profissionais e de alta qualidade.

Capítulo 2

Layout e Interatividade

Ecrã 1: Inicial

Figura 1 - Ecrã Inicial

Ecrã 1 – Detalhamento visual

A página inicial tem o fundo cinzento claro ***#F5F5F5****, o logotipo centrado no topo da página, logo abaixo mostra o titulo "Bem-vindos ao Restaurante Fastio" a* ***24pt, negrito****,* ***alinhado ao centro****, e " Escolha a sua mesa" a* ***18pt, regular, alinhado ao centro,*** *ambas na cor do logotipo Castanho escuro* ***#5D4037****, por fim contém os botões para selecionar a mesa, quando disponível aparece a castanho claro* ***#A1887F 16pt, negrito***, quando ocupadas aparece a vermelho ***#D32F2F***, o botão "Todas as contas" vai de uma ponta a outra do ecrã na mesma cor dos anteriores, todos eles têm o texto a branco ***#FFFFFF 16pt, negrito.***

Ecrã 1 - Interatividade

A página Inicial serve para selecionar uma mesa dentro das disponíveis (os botões castanhos) que leva para o menu principal que irá ser descrito a seguir, por fim tem um botão que permite ver todos os registadas do dia e oferece um feedback do que o restaurante faturou.

Ecrã 2: Categoria de Ementa

Figura 2 - Ecrã Categoria de Ementa

Ecrã 2 – Detalhamento visual

O menu principal contém o fundo cinzento-claro ***#F5F5F5*****,** no topo do ecrã tem o título “Menu” a ***32pt negrito, alinhado ao centro***, e o número da mesa selecionada para fazer o pedido a ***20pt regular, alinhado ao centro*** na parte central da tela estão 6 botões com imagens para dividir a ementa em grupos, cada imagem representa a sua categoria, e contem o seu nome abaixo da imagem, por fim contém dois botões um escrito “Conta” e outro “Voltar” ambos azuis ***#02678C*** com o texto a branco ***#FFFFFF***

Ecrã 2 - Interatividade

O menu principal serve para selecionar o tipo de ementa conforme o pedido feito, no caso de querer uma garrafa de vinho, seria clicar nas bebidas e iria para outra tela com todas as bebidas disponíveis que será a seguinte tela a ser exibida.

Tem o botão da conta que leva a uma página onde mostra a soma do valor de todos os pedidos feitos pela mesa selecionada, e por último tem também um botão para voltar à página anterior.

Ecrã 3: Ementa

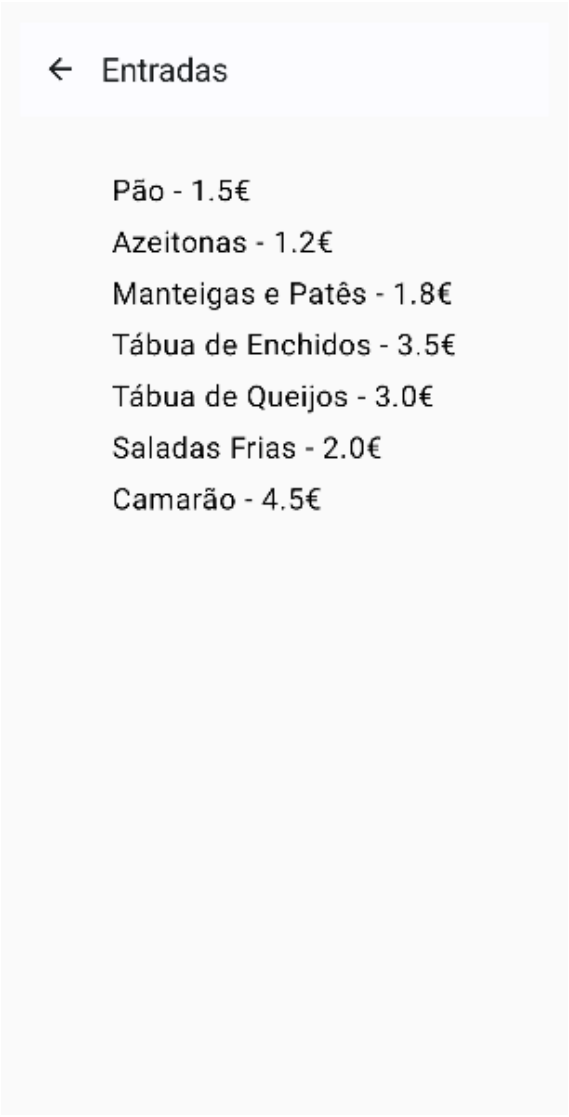


Figura 3 - Ecrã Ementa

Ecrã 3 – Detalhamento visual

O ecrã das ementas apresenta a cor cinzento-claro como fundo ***#F5F5F5,*** no topo do ecrã contém *o ícone de uma seta para voltar ao ecrã anterior, ao lado,* um título a dizer o tipo de ementa por exemplo como mostra a *figura 3* “Entradas”, que fica ***alinhado ao centro a 24pt***, ***negrito,*** cinzento-escuro ***#212121***, logo abaixo ta descrito a variedade de pratos a ***18pt regular***.

Ecrã 3 - Interatividade

Dentro da categoria escolhida vai ter uma diversidade de itens a escolher, quando clicado em algum vai aparecer outro background, que será exibido na seguinte página, o mesmo permite inserir a quantidade desejada e alguma nota se necessário.

Ecrã 4: Adicionar Item

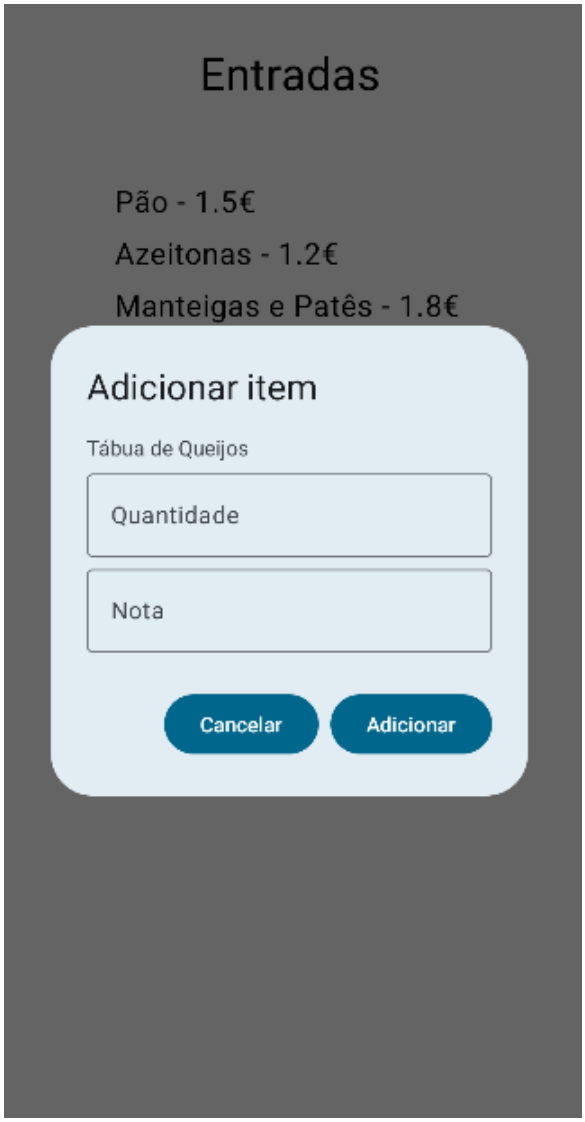


Figura 4 - Ecrã Adicionar Item

Ecrã 4 – Detalhamento visual

O background é cinzento-claro ***#E1EDF3,*** no topo tem o título “adicionar item” a ***24pt, negrito, alinhado à esquerda***, logo abaixo aparece o item selecionado no caso da *figura 4* “Tábua de queijos”, ***18pt, regular***, ***alinhado à esquerda*** de seguida existem duas caixas de texto escrito “Quantidade” e outra abaixo escrito “Nota” ambas a ***18pt, regular***.

No final tem dois botões azuis ***#02678C*** um escrito “Cancelar” e “Adicionar” com o texto a branco ***#FFFFFF 18pt, negrito***.

Ecrã 4 - Interatividade

Ao selecionar um item vai aparecer duas caixas de texto a primeira para inserir a quantidade e a seguinte para escrever alguma nota se necessário, por fim tem o botão “Adicionar” que irá fazer o pedido, e o botão “Cancelar”.

Ecrã 5: Conta

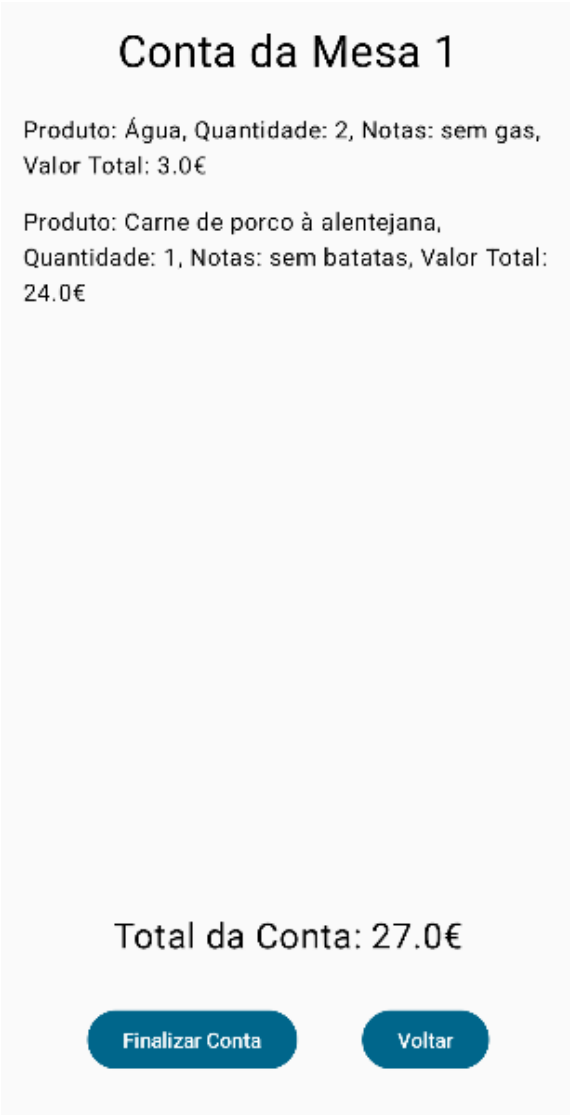


Figura 5 - Ecrã Conta

Ecrã 5 – Detalhamento visual

O ecrã conta contém o fundo a cinzento-claro ***#F5F5F5,*** no topo tem um título com o número da mesa selecionada como demonstra a *figura 5 "Conta da Mesa 2" com a formatação* ***24pt, negrito, alinhado ao centro***, de seguida aparece todos os pedidos da respetiva mesa na ordem: Produto, Quantidade, Notas, Valor Total, na formatação ***18pt, regular, alinhado à esquerda,*** na parte inferior do ecrã aparece o texto "Total da Conta" a ***20pt, negrito***, por último dois botões "Finalizar Conta" e "Voltar" ambos a azul ***#02678C*** com o texto ***18pt, negrito***.

Ecrã 5 - Interatividade

Ao entrar no ecrã conta irá mostrar a respetiva mesa, como mostra na *figura 5* "Conta da Mesa 2", apresenta a lista e os detalhes dos pedidos feitos na respetiva mesa, no final faz a soma total de tudo, ao finalizar a conta libera a mesa para fazer novos pedidos e ser utilizada por novos clientes.

Ecrã 6: Faturação Diário

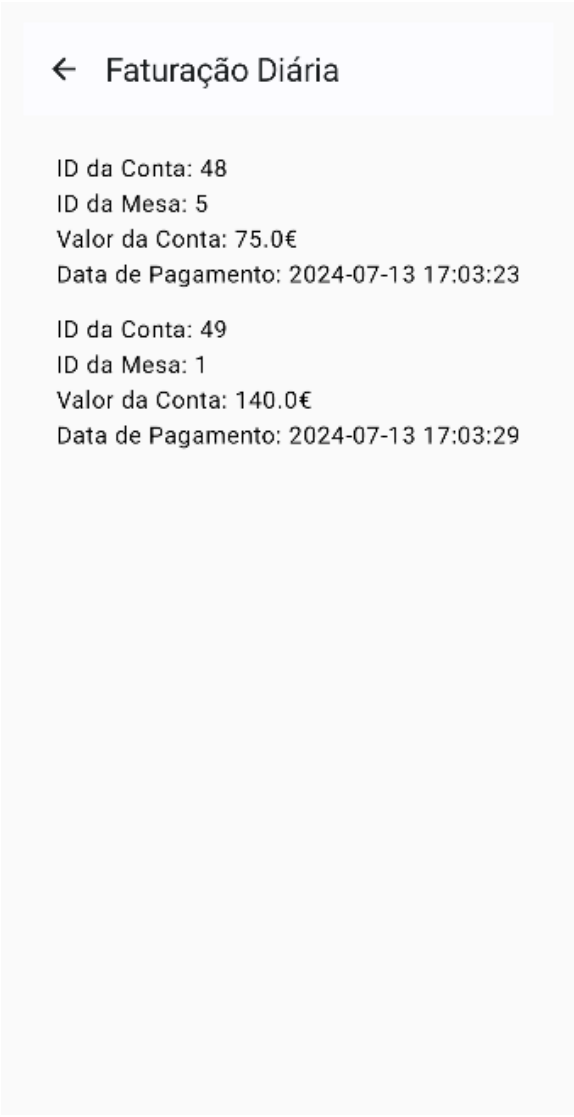


Figura 6 - Ecrã Faturameto Diário

Ecrã 5 – Detalhamento visual

O ecrã conta contém o fundo a cinzento-claro ***#F5F5F5***, no início contém o ícone de uma seta para voltar ao ecrã anterior, ao lado, o título “Todas as Contas” com a formatação ***24pt, negrito, alinhado ao centro***, no restante ecrã apresenta em texto todas as contas feitas no dia, ou seja, “ID da Conta”, “ID da Mesa”, “Valor da Conta” e “Data de Pagamento” tudo com a formatação ***18pt, regular, alinhado à esquerda.***

Ecrã 5 - Interatividade

Ao entrar no ecrã que está ilustrado na *figura 6*, contém os registos de todas as contas feitas do dia, ou seja, é bastante útil para no final de cada dia os funcionários, donos, quem administra o estabelecimento etc.…, conseguir ter um feedback do que foi feito no dia.

Funcionalidades da Aplicação – Resumo

Gestão de Pedidos em Tempo Real: A funcionalidade de gestão de pedidos em tempo real permite que os funcionários registem, modifiquem e visualizem pedidos dos clientes de forma instantânea. Os funcionários podem registar novos pedidos diretamente na interface da aplicação, escolhendo itens do menu de forma intuitiva. Esta funcionalidade também permite que os pedidos sejam modificados facilmente antes de serem enviados para a cozinha, incluindo a adição ou remoção de itens e a alteração de quantidades.

Por fim calcula a conta da mesa selecionada, somando todos os pedidos feitos na mesmo, a aplicação contém também uma opção que permite ver toda faturação diária.

Gestão de Mesas: A funcionalidade de gestão de mesas ajuda a organizar e otimizar o uso do espaço no restaurante. Com a visualização gráfica do layout do restaurante, os funcionários podem ver quais mesas estão ocupadas ou disponíveis, otimizando o uso do espaço disponível.

Capítulo 3

Código Kotlin – Partes Principais

Código 1: Navegação da aplicação

TelaSobremesas.kt | TelaTodasAsContas.kt | ViewModel.kt | RestauranteFastioDB.kt | ConfigDB.kt | NavMe

```kotlin
package com.example.restaurantefastio2

import ...

@RequiresApi(Build.VERSION_CODES.O)  Tiago Marcos
@Composable
fun NavMenu() {
    val navController = rememberNavController()

    NavHost(navController, startDestination = "TelaMesa") {
        composable( route: "TelaMesa") { TelaMesa(navController) }
        composable(
            route: "TelaMenu/{mesaNumber}",
            arguments = listOf(navArgument( name: "mesaNumber") { defaultValue = "1" })
        ) { backStackEntry ->
            val mesaNumber = backStackEntry.arguments?.getString( key: "mesaNumber")?.toIntOrNull() ?: 1
            TelaMenu(navController, mesaNumber)
        }
        composable(
            route: "TelaEntradas/{mesaNumber}",
            arguments = listOf(navArgument( name: "mesaNumber") { defaultValue = "1" })
        ) { backStackEntry ->
            val mesaNumber = backStackEntry.arguments?.getString( key: "mesaNumber")?.toIntOrNull() ?: 1
            TelaEntradas(navController, mesaNumber)
        }
        composable(
            route: "TelaBebidas/{mesaNumber}",
            arguments = listOf(navArgument( name: "mesaNumber") { defaultValue = "1" })
        ) { backStackEntry ->
            val mesaNumber = backStackEntry.arguments?.getString( key: "mesaNumber")?.toIntOrNull() ?: 1
            TelaBebidas(navController, mesaNumber)
```

Figura 7 - Navegação pelas telas

Este código configura a estrutura de navegação da aplicação, permitindo que o utilizador navegue entre diferentes telas (como menu, entradas, bebidas, etc.) através do *NavController* do Navigation Component. Cada ecrã recebe o número da mesa como parâmetro quando necessário, facilitando a personalização e interação fluida do utilizador com o aplicativo.

Código 2: Interface da tela inicial

```
@Composable  Tiago Marcos
fun TelaMesa(navController: NavController, viewModel: ViewModel = viewModel()) {
    LaunchedEffect(Unit) {
        viewModel.loadPedidosMesas()
    }

    val pedidosMesas by viewModel.pedidos_mesas.collectAsState()

    Box(
        modifier = Modifier
            .fillMaxSize()
            .background(Color( color: 0xFFF5F5F5))
            .padding(16.dp)
    ) {
        Column(
            modifier = Modifier
                .fillMaxSize(),
            verticalArrangement = Arrangement.Center,
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Image(
                painter = painterResource(id = R.drawable.logo),
                contentDescription = null,
                modifier = Modifier
                    .size(250.dp)
                    .padding(10.dp)
                    .align(Alignment.CenterHorizontally)
            )

            Text(
```

Figura 8 - Código tela inicial

Este código implementa a interface da tela inicial de um restaurante, permitindo aos utilizadores escolher uma mesa disponível para visualizar o menu correspondente. A tela é estilizada com cores e fontes específicas para refletir a identidade visual do restaurante. O uso de *Jetpack Compose* facilita a construção de interfaces de utilizador dinâmicas e responsivas no Android.

Código 3: Organiza a tela de menu

Figura 9 - Código tela Menu

Este código organiza o ecrã de menu de forma clara e eficiente, permitindo aos utilizadores selecionar diferentes categorias de itens do menu e realizar ações como verificar a conta ou voltar à tela anterior. A navegação entre as telas é gerida pelo *NavHostController*, proporcionando uma experiência ao utlizador intuitiva e fluida.

Código 4: Ecrãs das Ementas

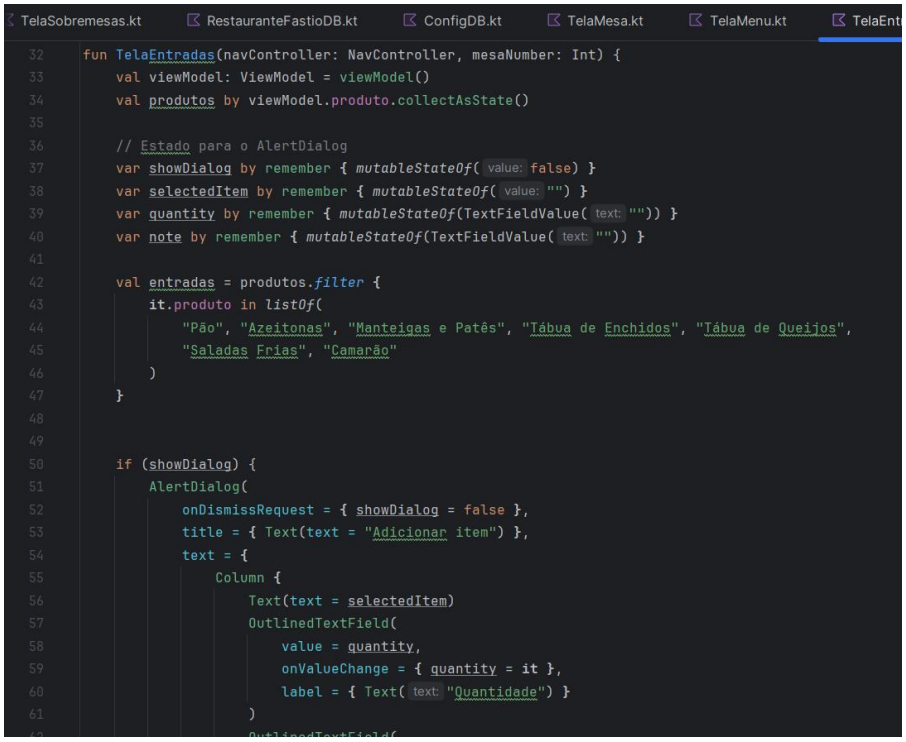


Figura 10 - Código das Ementas

Este código organiza a tela de entradas da aplicação, permitindo aos utilizadores selecionar e adicionar itens ao pedido. A navegação e interatividade são geridas pelo *NavController* e *ViewModel*, proporcionando uma experiência ao utilizador eficiente e intuitiva.

Código 5: Ecrã Conta

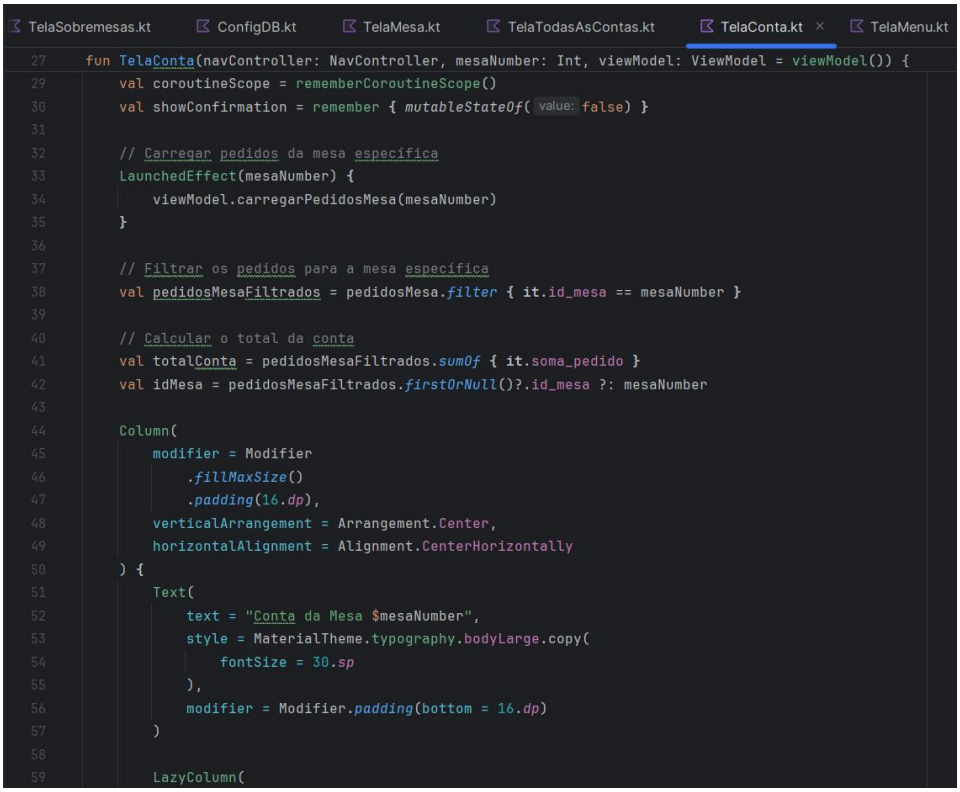


Figura 11 - Código da Conta

Este código organiza o ecrã da "Conta", permitindo aos utilizadores visualizar e finalizar a conta de uma mesa específica. A estrutura da interface é projetada para exibir informações detalhadas sobre os pedidos e o total da conta de maneira clara e acessível.

Código 6: Ecrã Faturação Diária

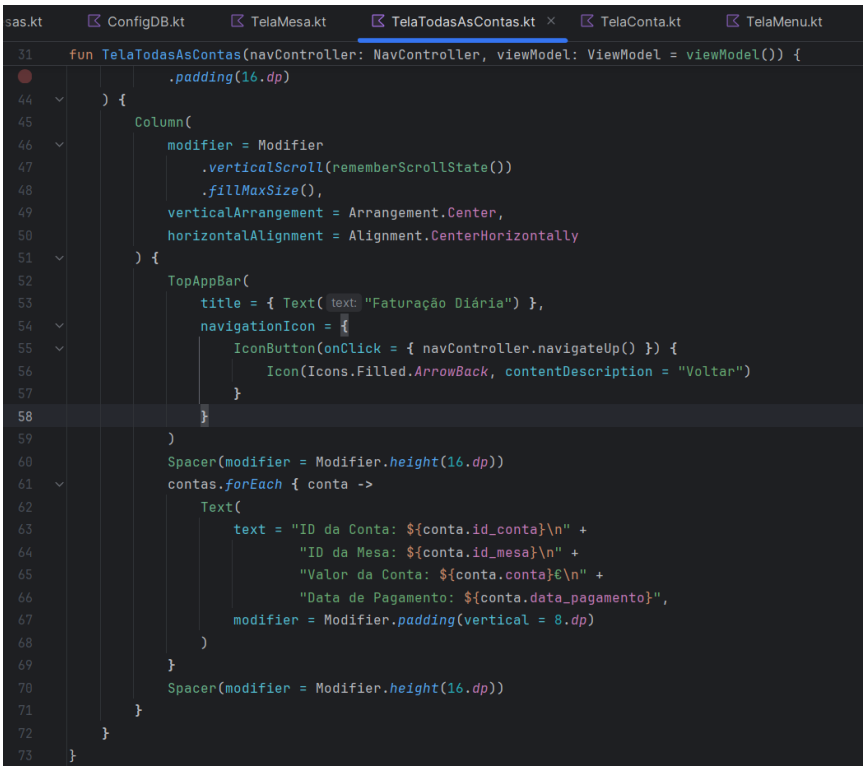


Figura 12 - Ecrã Faturação Diária

Este código é projetado para apresentar uma visão geral de todas as contas finalizadas no dia, de maneira clara e organizada. A navegação é facilitada pelo *NavController*, permitindo ao utilizador retornar ao ecrã anterior. A interação com os dados é gerida pelo *ViewModel*, que utiliza corrotinas para carregar as contas de forma assíncrona, garantindo uma experiência ao utilizador eficiente e intuitiva. A interface simples e funcional assegura que os utilizadores possam visualizar rapidamente todas as informações relevantes sobre as contas.

Código 7: Tabelas da base de dados

Figura 13 - Código das tabelas da base de dados

O codigo descrito na figura 13 faz a criação das tabelas, *mesas* com as colunas “id_mesa” e “nome”; *produto*, com as colunas “id_produto”, “produto e “preco”, *pedidos_mesa* com as colunas “id_mesa”, “produto”, categoria“, “quant”, “notas” e “soma_pedido”, por fim a tabela *conta* com as colunas “id_mesa“, “conta” e ”data_pagamento”.

Código 8: Dao

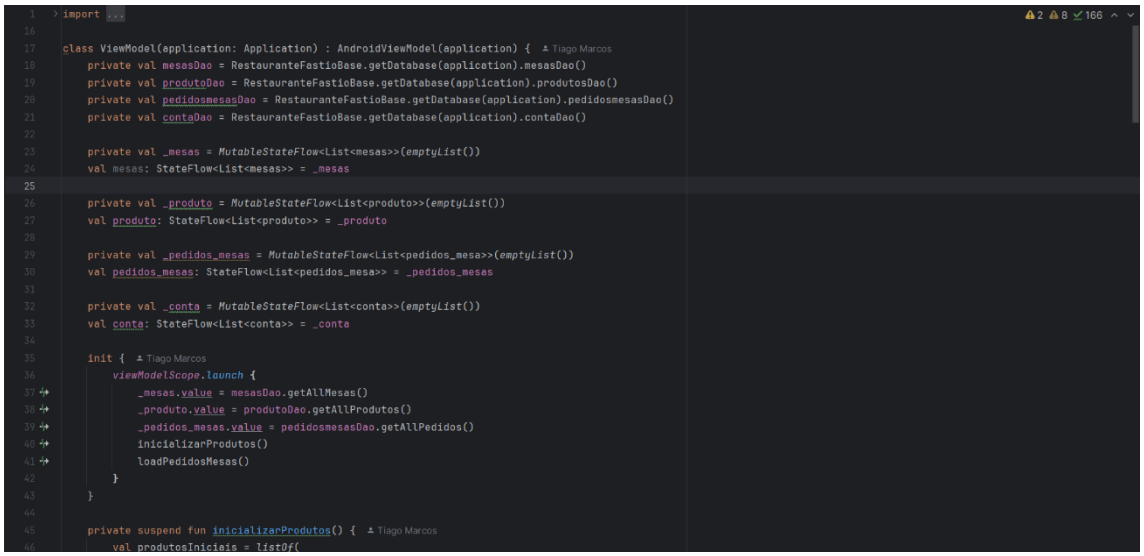
```
1  package com.example.restaurantefastio2.db
2
3  import androidx.room.*
4  @Dao  Tiago Marcos
5  interface ProdutosDao {
6      @Insert  Tiago Marcos
7      suspend fun insert(produto: produto)
8      @Query("SELECT * FROM produto")  Tiago Marcos
9      suspend fun getAllProdutos(): List<produto>
10     @Insert(onConflict = OnConflictStrategy.REPLACE)  Tiago Marcos
11     suspend fun insertAll(produtos: List<produto>)
12     @Query("DELETE FROM produto")  Tiago Marcos
13     suspend fun deleteAll()
14 }
15
```

Figura 14 - Código Dao

Os DAOs são fundamentais para a arquitetura de software, fornecendo uma camada clara de abstração entre a lógica de negócios e a persistência de dados. Eles promovem uma maior modularidade, testabilidade e manutenção do código, contribuindo para uma aplicação mais robusta e escalável.

Neste caso a *figura 14* mostra o da tabela produtos, mas as restante apresentam a mesma estrutura.

Código 9: ViewModel



```
import ...

class ViewModel(application: Application) : AndroidViewModel(application) {  Tiago Marcos
    private val mesasDao = RestauranteFastioBase.getDatabase(application).mesasDao()
    private val produtoDao = RestauranteFastioBase.getDatabase(application).produtosDao()
    private val pedidosmesasDao = RestauranteFastioBase.getDatabase(application).pedidosmesasDao()
    private val contaDao = RestauranteFastioBase.getDatabase(application).contaDao()

    private val _mesas = MutableStateFlow<List<mesas>>(emptyList())
    val mesas: StateFlow<List<mesas>> = _mesas

    private val _produto = MutableStateFlow<List<produto>>(emptyList())
    val produto: StateFlow<List<produto>> = _produto

    private val _pedidos_mesas = MutableStateFlow<List<pedidos_mesa>>(emptyList())
    val pedidos_mesas: StateFlow<List<pedidos_mesa>> = _pedidos_mesas

    private val _conta = MutableStateFlow<List<conta>>(emptyList())
    val conta: StateFlow<List<conta>> = _conta

    init {  Tiago Marcos
        viewModelScope.launch {
            _mesas.value = mesasDao.getAllMesas()
            _produto.value = produtoDao.getAllProdutos()
            _pedidos_mesas.value = pedidosmesasDao.getAllPedidos()
            inicializarProdutos()
            loadPedidosMesas()
        }
    }

    private suspend fun inicializarProdutos() {  Tiago Marcos
        val produtosIniciais = listOf(
```

Figura 15 - ViewModel

Este *ViewModel* centraliza a lógica de negócios e interações de dados da aplicação, garantindo uma arquitetura limpa e eficiente. Utilizando corrotinas para operações assíncronas, o ViewModel assegura que as operações da base de dados não bloqueiem a UI. Através do uso de StateFlow, o estado da UI é reativo às mudanças nos dados, proporcionando uma experiência ao utilizador fluida e responsiva. Navegação e interatividade são geridas pelo NavController e ViewModel.

Conclusão

Em síntese, o desenvolvimento da aplicação de gestão de pedidos para restaurante representou um marco significativo na otimização das operações diárias de estabelecimentos gastronômicos. Através da utilização de tecnologias modernas como Kotlin e Android Studio, conseguimos criar uma solução robusta e intuitiva que não só facilita o fluxo de trabalho dos funcionários, mas também melhora a experiência geral dos clientes.

A aplicação proporciona uma interface intuitiva que permite aos funcionários realizar pedidos de forma rápida e precisa, contribuindo para uma gestão eficiente das mesas e pedidos em tempo real. A integração com uma base de dados sólida garante que todas as transações sejam registadas de maneira segura e acessível, facilitando a gestão financeira e operacional do restaurante.

Glossário

UI	User Interface
API	Application Programming Interface
Boilerplate	Código padrão e repetitivo
Corrotinas	Recurso que facilita a programação assíncrona
Gradle	Sistema de automação de builds

Anexos

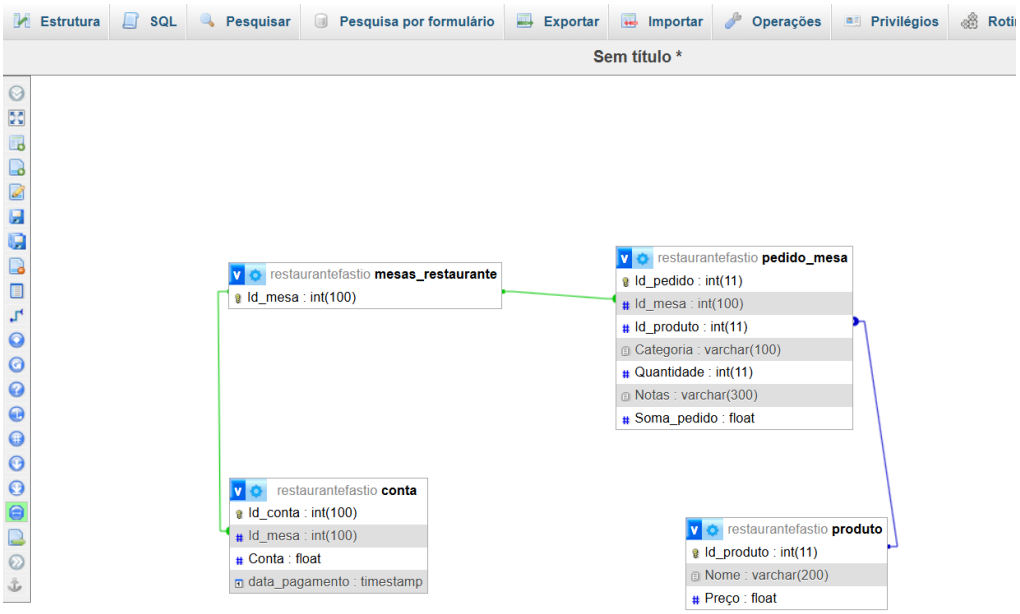


Figura 16 - Modelo Base de Dados em SQL

Figura 17 - Software Adobe Illustrator