
**UNIVERSIDADE FEDERAL DE CAMPINA GRANDE**
**CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA**
**CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**ISAÍAS MARTINS TEIXEIRA PONTES**

# APLICATIVO GERENCIADOR DE PEDIDOS DE RESTAURANTE COM CONEXÃO PARA IMPRESSORA TÉRMICA

**CAMPINA GRANDE - PB**

**2023**

**ISAÍAS MARTINS TEIXEIRA PONTES**

# APLICATIVO GERENCIADOR DE PEDIDOS DE RESTAURANTE COM CONEXÃO PARA IMPRESSORA TÉRMICA

**Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.**

**Orientador : Claudio Elízio Calazans Campelo**

**CAMPINA GRANDE - PB**

**2023**

**ISAÍAS MARTINS TEIXEIRA PONTES**

# APLICATIVO GERENCIADOR DE PEDIDOS DE RESTAURANTE COM CONEXÃO PARA IMPRESSORA TÉRMICA

**Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.**

## BANCA EXAMINADORA:

**Claudio Elízio Calazans Campelo**

**Orientador – UASC/CEEI/UFCG**

**Roberto Medeiros de Faria**

**Examinador – UASC/CEEI/UFCG**

**Francisco Vilar Brasileiro**

**Professor da Disciplina TCC – UASC/CEEI/UFCG**

**Trabalho aprovado em: 29 de JUNHO de 2023.**

**CAMPINA GRANDE - PB**

# RESUMO

Este trabalho tem como objetivo promover uma solução para pequenas e médias empresas no ramo gastronômico de delivery (entregas em domicílio), com ênfase nos estabelecimentos que, por muitas vezes, contam com uma equipe reduzida e pessoas não familiarizadas com o uso de computadores. Com a pandemia da COVID-19, o número de entregas de alimentos aumentou consideravelmente, e a maioria dessas demandas são concentradas em poucos aplicativos comerciais. No entanto, esses aplicativos apresentam alta complexidade de uso. O que prejudica os restaurantes de menor porte que não tem familiaridade com uso de computador e nem condições para compra de equipamentos necessários como impressora térmica serial que tem um custo elevado. Para solucionar esse problema, foi desenvolvido uma aplicação Android e backend que gerencia os pedidos, desde o fluxo do cardápio até a entrega. A aplicação possui funcionalidades essenciais, como a impressão do pedido por meio de impressora térmica bluetooth de baixo custo. Dessa forma, é possível atingir um público maior que depende de pedidos impressos para a comunicação entre a cozinha e os entregadores, melhorando a organização do estabelecimento com redução do tempo de entrega e prevenção de erros humanos.

# APLICATIVO GERENCIADOR DE PEDIDOS DE RESTAURANTE COM CONEXÃO PARA IMPRESSORA TÉRMICA

## ABSTRACT

This work aims to provide a solution for small and medium-sized businesses in the food delivery sector, with a focus on establishments that often have a limited workforce and individuals who are not familiar with computer usage. With the COVID-19 pandemic, the number of food deliveries has significantly increased, and most of these demands are concentrated on a few commercial applications. However, these applications are highly complex to use, which hinders smaller restaurants that lack computer literacy and cannot afford costly equipment like thermal serial printers. To address this problem, an Android application and backend were developed to manage orders, from the menu flow to delivery. The application offers essential functionalities, including printing orders using a low-cost Bluetooth thermal printer. This way, a wider audience can be reached, as many rely on printed orders for communication between the kitchen and delivery personnel, improving the establishment's organization by reducing delivery time and preventing human errors.

# Aplicativo gerenciador de pedidos de restaurante com conexão para impressora térmica

Trabalho de Conclusão de Curso

Isaias Martins Teixeira Pontes (Aluno), Cláudio Campelo (Orientador)
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Paraíba - Brasil

## RESUMO

Este trabalho tem como objetivo promover uma solução para pequenas e médias empresas no ramo gastronômico de *delivery* (entregas em domicílio), com ênfase nos estabelecimentos que, por muitas vezes, contam com uma equipe reduzida e pessoas não familiarizadas com o uso de computadores.

Com a pandemia da COVID-19, o número de entregas de alimentos aumentou consideravelmente, e a maioria dessas demandas são concentradas em poucos aplicativos comerciais. No entanto, esses aplicativos apresentam alta complexidade de uso. O que prejudica os restaurantes de menor porte que não tem familiaridade com uso de computador e nem condições para compra de equipamentos necessários como impressora térmica serial que tem um custo elevado.

Para solucionar esse problema, foi desenvolvido uma aplicação Android e *backend* que gerencia os pedidos, desde o fluxo do cardápio até a entrega. A aplicação possui funcionalidades essenciais, como a impressão do pedido por meio de impressora térmica bluetooth de baixo custo. Dessa forma, é possível atingir um público maior que depende de pedidos impressos para a comunicação entre a cozinha e os entregadores, melhorando a organização do estabelecimento com redução do tempo de entrega e prevenção de erros humanos.

## PALAVRAS-CHAVE

*Mobile*[13], *Django DRF*[6], *API*[5], *Delivery.*

## 1 INTRODUÇÃO

Após a pandemia de COVID-19, o setor de *delivery* de alimentos registrou um aumento significativo no número de pedidos, evidenciando a popularização desse serviço[11]. Segundo dados do IBGE[10], em 2021, cerca de 28 milhões de pessoas no Brasil ainda não utilizavam a internet, porém a demanda por serviços de entrega cresceu exponencialmente. O comércio eletrônico, incluindo a entrega de comida por meio de aplicativos, tornou-se uma das categorias de maior crescimento nesse período desafiador. Os aplicativos de entrega de comida foram impulsionados pelo aumento das entregas em 2020. Essas mudanças destacam a importância e a necessidade de adaptação do setor gastronômico para atender às demandas dos consumidores, oferecendo soluções práticas e acessíveis por meio do *delivery* de alimentos.

Nesse contexto, os pequenos restaurantes que realizam entrega em domicílio enfrentam grandes dificuldades por não ter um sistema consolidado para auxiliar que seja fácil de usar e não precise adquirir computador[12], tendo que, por muitas vezes, fazer trabalhos manualmente e passíveis de erros, como: anotar pedidos em papel (provenientes de ligação ou mensagem), anotar endereços, adicionar observações de cada item dos pedidos, explicar aos clientes os itens que estão disponíveis, entre outros.

Por outro lado, os clientes também enfrentam desafios na experiência de compra. Geralmente, recebem cardápios em formato PDF ou precisam conferir as opções disponíveis por meio de postagens em redes sociais. Além disso, calcular o valor total do pedido e adicionar observações personalizadas a determinados produtos pode ser difícil.

Os sistemas existentes atendem às necessidades dos grandes restaurantes ou de um público familiarizado com a utilização de computadores, mas não são adequados para aqueles que não têm acesso a um computador[12], seja por limitações financeiras ou falta de conhecimento em informática. Segundo uma pesquisa da Fundação Getúlio Vargas, atualmente existem mais de um smartphone por habitante no Brasil, totalizando 242 milhões de dispositivos móveis. Entre os dispositivos portáteis, o smartphone é o mais vendido, devido ao seu custo mais acessível e facilidade de uso. Portanto, o dispositivo é a melhor opção para inclusão das pessoas de baixa renda que muitas vezes não têm acesso a um notebook. Além disso, o smartphone é o meio mais utilizado e mais simples para que pessoas com pouca afinidade com tecnologia possam automatizar seus estabelecimentos.

Diante desse cenário, é necessário desenvolver um sistema que facilite o gerenciamento por parte dos restaurantes, por meio de um aplicativo móvel, ao mesmo tempo em que aprimore a experiência dos clientes. Assim, este artigo apresenta uma aplicação desenvolvido para restaurantes, que permite receber pedidos, imprimir em impressoras térmicas Bluetooth e gerenciar todo o fluxo do pedido, desde o recebimento até a entrega.

## 2 CASOS DE USO

A arquitetura do projeto é ilustrada na Figura 1. Nela, destacam-se três entidades principais: o *frontend*, o *backend* e o *mobile*. Essas entidades interagem entre si e realizam as diversas ações relacionadas ao gerenciamento de pedidos e cardápios.

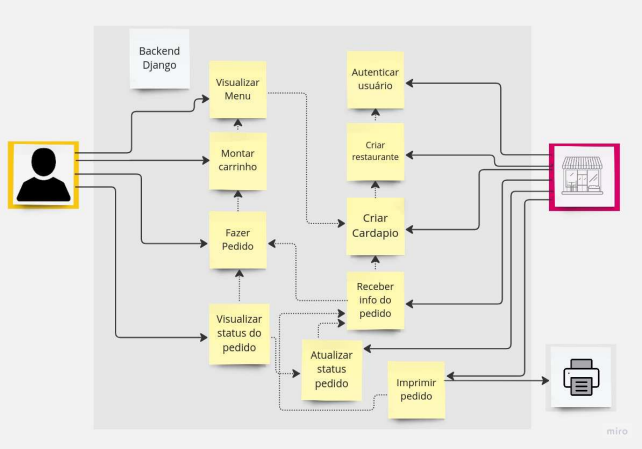


**Figura 1: Arquitetura do projeto**

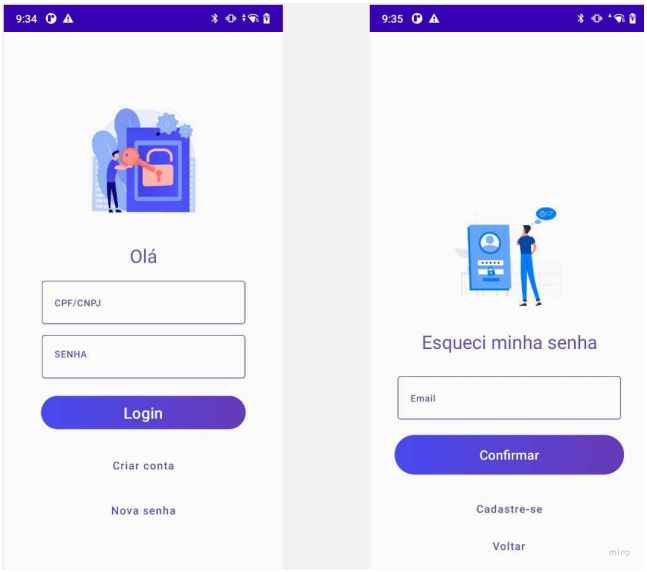


**Figura 2: Páginas de login**

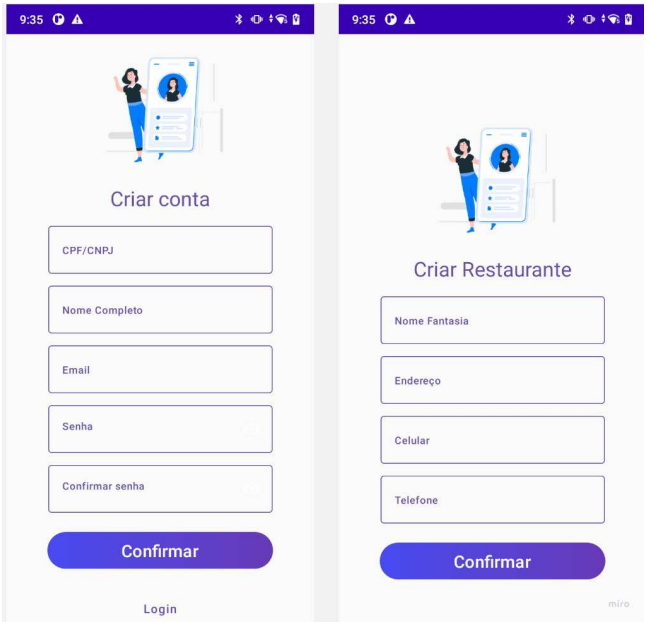


**Figura 3: Página de cadastro**

O *backend* é responsável pela lógica do negócio e executa todas as operações relacionadas à criação, atualização e leitura dos pedidos e cardápios. Ele gerencia os dados, processa as requisições e fornece as informações necessárias para o funcionamento adequado do sistema.

Por sua vez, o *frontend* é responsável pela apresentação dos dados provenientes do *backend* ao cliente. Ele oferece uma interface intuitiva e amigável, por meio da qual o cliente pode montar seu carrinho de compras e realizar o pedido. O *frontend* tem o objetivo de facilitar a interação do cliente com o sistema, tornando o processo de seleção dos itens e finalização do pedido mais simples e eficiente.

A aplicação *mobile* é utilizado na interação com o restaurante. Além de apresentar os pedidos, permitir a criação do cardápio e a edição dos itens do restaurante, ele possui uma integração especial com uma impressora térmica[14] por meio da tecnologia Bluetooth. Essa integração possibilita a impressão dos pedidos com os dados do cliente, o que confere um diferencial ao sistema. Com essa funcionalidade, é possível garantir que as informações do pedido sejam impressas de forma rápida e precisa, facilitando o processo de preparação e entrega dos alimentos.

Essa arquitetura permite uma comunicação eficiente entre as entidades e proporciona uma experiência de uso completa e integrada, tanto para o cliente quanto para o restaurante. A interação entre o *frontend*, *backend* e *mobile* é fundamental para o bom funcionamento do sistema de gerenciamento de pedidos e cardápios, garantindo agilidade, precisão e facilidade de uso.

### *2.0.1 Aplicativo Mobile.*

Como ilustrado nas Figuras 2 e 3, a aplicação *mobile* oferece funcionalidades essenciais para o restaurante, como o login e a opção de alterar a senha. Essas opções permitem que os usuários acessem o sistema de forma segura e personalizem suas credenciais de acesso de acordo com suas necessidades.

Além disso, o *backend* do sistema suporta a gestão de múltiplos restaurantes para um único usuário. Isso significa que, futuramente, será possível implementar uma funcionalidade para alternar entre diferentes estabelecimentos dentro de um único login. Essa flexibilidade proporciona um maior gerenciamento e controle para os usuários que possuem mais de um restaurante.

Essas funcionalidades do aplicativo *mobile*, aliadas ao suporte do *backend*, contribuem para uma experiência de uso completa e personalizada no gerenciamento dos estabelecimentos gastronômicos.

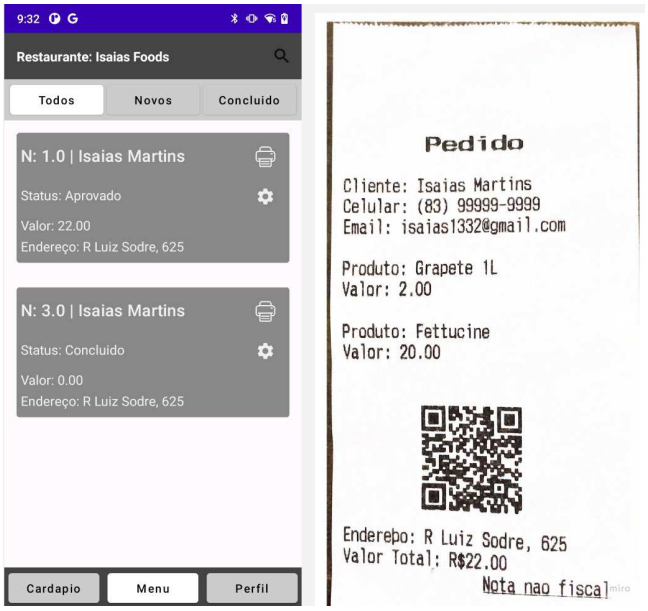


**Figura 4: Impressão de pedido.**

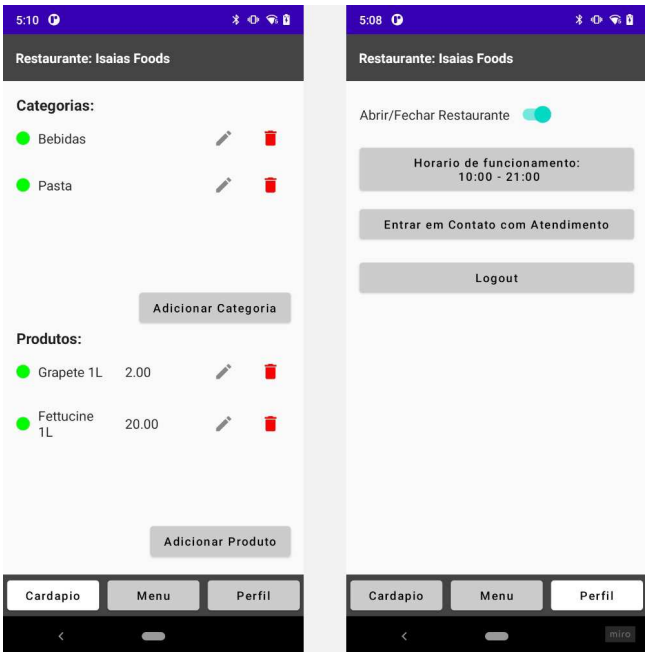


**Figura 5: Página de cardápio e Perfil**

No contexto do menu, conforme ilustrado na Figura 4, o aplicativo *mobile* oferece diversas funcionalidades que facilitam o gerenciamento do cardápio e o controle dos pedidos. Uma dessas funcionalidades é a possibilidade de alterar o status dos pedidos de forma fácil e rápida, o que permite manter um controle eficiente do processo de atendimento.

Além disso, o aplicativo *mobile* permite a impressão dos pedidos, conforme mostrado na Figura 4. Essa funcionalidade é especialmente útil em ambientes de restaurante, pois agiliza o serviço e garante a precisão dos dados. Os usuários têm a capacidade de criar categorias e produtos relacionados ao cardápio, personalizando-o de acordo com as necessidades específicas do restaurante.

Outra funcionalidade importante é a opção de ativar e desativar o restaurante. Essa flexibilidade permite ajustar o funcionamento do estabelecimento de acordo com horários de pico, dias de folga ou qualquer outra necessidade específica.

O aplicativo também oferece a opção de deslogar do sistema na aba de Perfil, permitindo que os usuários encerrem sua sessão quando necessário. Para futuras melhorias, está prevista a inclusão de uma barra de pesquisa para facilitar a busca por clientes, pedidos ou status específicos.

Toda a implementação do *backend* foi projetada levando em consideração a possibilidade de futuras modificações e melhorias, garantindo a flexibilidade e escalabilidade do sistema.

# 3 ARQUITETURA

## 3.1 Backend

O *backend* foi projetado para garantir a segurança e a separação dos dados de cada restaurante. Por meio do processo de autenticação, é possível identificar qual restaurante está logado, e o token de autenticação é utilizado para filtrar todos os dados relacionados a esse restaurante, como categorias, produtos, carrinhos e pedidos. Essa abordagem permite que a plataforma possa cadastrar e gerenciar vários restaurantes em um único sistema, proporcionando flexibilidade e escalabilidade.

Para o desenvolvimento do *backend*, utilizou-se a linguagem Python[3] em conjunto com o framework Django Rest Framework[6]. O Django Rest Framework é uma extensão do framework Django que fornece uma estrutura robusta para a construção de APIs. Além disso, o Django Rest Framework possui seu próprio ORM (Object Relational Mapper), o que simplifica o acesso e a manipulação dos dados no banco de dados.

Ao utilizar o Django Rest Framework, é possível agilizar o desenvolvimento, uma vez que ele cuida automaticamente da criação das tabelas e das operações relacionadas ao banco de dados. O *framework* também facilita a serialização e desserialização dos dados no formato JSON, tornando a comunicação com o *frontend* mais eficiente. Além disso, o Django Rest Framework cria automaticamente uma interface de administrador, o que simplifica o processo de gerenciamento dos dados do sistema sem a necessidade de conexão direta com o banco de dados.

Com essa estrutura, a plataforma oferece um ambiente seguro, flexível e eficiente para o gerenciamento de múltiplos restaurantes, além de facilitar a manutenção do sistema.

### *3.1.1 Estrutura.*

A estrutura do projeto segue o padrão MVC *(Model-View-Controller)*, com cada aplicação possuindo suas próprias pastas e responsabilidades específicas. Os modelos são responsáveis por mapear as tabelas do banco de dados e facilitar as operações de criação, leitura,

atualização e exclusão *(CRUD)*. A serialização é utilizada para converter os objetos Python em formatos adequados para a comunicação com o *frontend*, como JSON.

O Django[1] admin é uma poderosa ferramenta que oferece uma interface de administração pronta para uso, simplificando a gestão dos dados do sistema. Com ela, é possível realizar tarefas como adicionar, editar e excluir registros do banco de dados de forma fácil e intuitiva.

No projeto, foram criadas duas aplicações no Django[8]. A primeira aplicação lida com a autenticação[9], abstraindo essa etapa de todas as outras rotas e simplificando o processo de manutenção. A segunda aplicação é responsável pela parte relacionada ao Restaurante. Nessa aplicação, encontramos a definição das classes Python que representam as entidades do banco de dados, como Restaurant, Product, Order, Menu e Cart. Essas classes possuem os atributos e métodos necessários para manipular os dados correspondentes a cada entidade, facilitando a interação com o banco de dados.

Essa estrutura de pastas e organização das aplicações proporciona um desenvolvimento mais organizado e escalável, permitindo a separação de responsabilidades e facilitando a manutenção do projeto.

## 3.2 Mobile

O desenvolvimento *mobile* utilizando a biblioteca Jetpack Compose[13] proporciona uma abordagem inovadora e eficiente para a criação de interfaces de usuário. Ao contrário do desenvolvimento tradicional com XML, o Jetpack Compose permite a criação da interface de forma declarativa e direta no código Kotlin[2], oferecendo mais flexibilidade, reutilização de código e uma experiência de programação mais produtiva.

A programação reativa incorporada no Jetpack Compose permite lidar com estados e eventos de forma eficiente, possibilitando atualizações dinâmicas da interface de acordo com as interações do usuário ou mudanças nos dados. Isso resulta em interfaces fluidas e responsivas, proporcionando uma melhor experiência para o usuário.

Uma das funcionalidades do aplicativo é a utilização de *Intents*[7] para interagir com o sistema operacional e realizar a impressão dos pedidos em uma impressora térmica[14]. As Intents permitem que o aplicativo abra configurações específicas, como as de Bluetooth, e realize ações externas, como a impressão de documentos. Isso amplia a capacidade do aplicativo e possibilita a integração com outras partes do sistema operacional.

Para a autenticação e o gerenciamento de usuários, o aplicativo faz uso da API do Django, realizando chamadas *HTTP* para realizar operações de login e criação de usuários. O *Retrofit*[16] é uma biblioteca utilizada para simplificar as requisições *HTTP*, permitindo a definição de interfaces com os endpoints desejados e a realização de chamadas assíncronas de forma transparente.

O armazenamento do *token*[15] de autorização é feito utilizando o *LocalStorage*[4], uma forma de armazenamento persistente no dispositivo. O token[15] é armazenado localmente para ser utilizado nas requisições subsequentes, garantindo a autenticação e a segurança do aplicativo. Além disso, o aplicativo utiliza um Authenticator para adicionar automaticamente o *token*[15] de autorização nas requisições, facilitando o processo de autenticação em todas as chamadas realizadas.

Essas tecnologias e abordagens utilizadas no desenvolvimento *mobile* contribuem para a criação de um aplicativo eficiente, seguro e de fácil manutenção, garantindo uma experiência agradável tanto para os usuários quanto para os desenvolvedores.

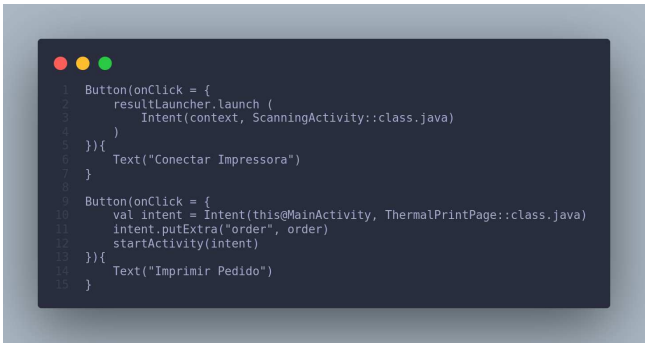


**Figura 6: Navagação entre Intents**

A navegação entre elementos Composables no Jetpack Compose permite a transição entre diferentes telas ou composições dentro do aplicativo, seguindo uma lógica de fluxo de trabalho definida. O componente *Navigation* é utilizado para definir rotas para cada tela e configurar a navegação entre elas. Isso permite que o usuário interaja com o aplicativo de forma intuitiva, navegando de uma tela para outra de acordo com a sequência desejada.

A navegação entre *intents*[7] é relevante quando é necessário interagir com recursos específicos do dispositivo ou realizar integrações externas. No contexto do desenvolvimento mencionado, foi necessário lidar com configurações de Bluetooth e conexão para impressão. Nesse caso, a navegação entre *intents*[7] permite que o aplicativo inicie ações relacionadas a esses recursos, como abrir as configurações de Bluetooth ou iniciar a impressão. Essas *intents*[7] são como atividades independentes, que podem ser acionadas a partir do aplicativo principal.

Como ilustrado pela Figura 6, é possível observar o código fonte relacionado à navegação entre *intents*[7] no contexto específico do aplicativo. Essa parte do código define a criação de *intents*[7] para imprimir pedidos e para escanear dispositivos e pareá-los via Bluetooth. Essas *intents*[7] são responsáveis por iniciar atividades externas ao aplicativo principal, permitindo a interação com recursos específicos do dispositivo.

Portanto, a navegação entre elementos Composables está relacionada à transição entre as partes do aplicativo, enquanto a navegação entre *intents*[7] é usada para interagir com recursos do dispositivo ou integrações externas. Ambas as abordagens são importantes no desenvolvimento de aplicativos no Jetpack Compose, pois proporcionam controle e fluidez na navegação, seja dentro do aplicativo ou ao interagir com recursos específicos do dispositivo.

### *3.2.1 Estrutura.*

O diretório principal do aplicativo é o app/. Dentro dele, temos os seguintes pacotes:

- api/ Contém as classes relacionadas à interação com a API do Django.[6]

- models/ Armazena as classes de modelo de dados.
- pages/ Engloba as telas do aplicativo, com subpacotes específicos para as telas de login e menu.
- repository/ Responsável pelo repositório de dados, lidando com a comunicação entre as telas e os dados.
- sharedPreferences/ Trata do armazenamento local de dados utilizando a SharedPreferences.
- ui.theme/ Define o estilo e a aparência da interface do usuário.
- ComponentsHelper/ Contém componentes de apoio que podem ser reutilizados em várias telas.
- ThermalPrintPage/ Implementa a tela específica para a impressão térmica.
- MainActivity.kt Classe responsável por iniciar o aplicativo e configurar o ambiente de execução.

Essa estrutura modularizada facilita a organização e a manutenção do código do aplicativo *Mobile*, permitindo uma separação clara de responsabilidades e facilitando a localização dos arquivos relacionados a cada funcionalidade do aplicativo.

## 4 AVALIAÇÃO

A aplicação desenvolvida ainda não foi validada por clientes reais. No entanto, foram realizados testes em três dispositivos Android: *Xiaomi MIA3 (versão 2021), Samsung Tab S6 Lite e Asus Zenfone 9*. Todos os dispositivos testados funcionaram perfeitamente, com as permissões necessárias, como acesso à localização (essencial para o Bluetooth) e acesso à internet. É importante observar que, devido à utilização do Jetpack Compose, o aplicativo requer no mínimo a versão *Android 5.0 (Lollipop), com SDK mínimo API 21*. Quanto ao *backend*, ele foi executado localmente em um notebook *Acer i7 da 6ª geração, com 8 GB de RAM*, e uma porta exposta para permitir os testes necessários.

Além disso, foram realizados testes de integração com uma impressora térmica modelo *C58A-203*, com as seguintes especificações: tamanho de 105 x 75 x 45 mm, largura de impressão de 48 mm e largura de papel de 57,5 mm. A velocidade de impressão da impressora variou entre 50 e 80 mm/s, proporcionando uma impressão rápida e eficiente dos pedidos. A Figura 4 exibe o resultado de uma impressão realizada pelo equipamento.

## 5 EXPERIÊNCIA

Durante o processo de integração da impressora térmica com o aplicativo *mobile* e *backend*, enfrentei alguns desafios significativos. No entanto, minha experiência e preparação na universidade foram fundamentais para superar essas dificuldades e progredir nesse projeto.

A formação acadêmica na universidade foi fundamental para minha preparação na integração da impressora térmica. Disciplinas como Projeto de Software, Projeto de Programação, Arquitetura de Software e Banco de Dados forneceram uma base sólida em conceitos e práticas de desenvolvimento de software. Esses conhecimentos me permitiram compreender os princípios fundamentais, aplicar metodologias adequadas e garantir uma estrutura robusta e eficiente na integração da impressora térmica. Com a base teórica e prática adquirida durante meu curso, pude enfrentar os desafios com confiança e desenvolver uma solução de qualidade.

Além da formação acadêmica, os estágios que realizei foram cruciais para minha jornada de aprendizado, especialmente no que diz respeito ao Kotlin[2], a linguagem de programação utilizada no desenvolvimento Android. Durante essas experiências profissionais, tive a oportunidade de aprofundar meus conhecimentos em Kotlin e aplicá-los em projetos práticos. Essa experiência complementar me permitiu dominar as habilidades necessárias para lidar com intents, permissões, navegação entre telas e outras tarefas relacionadas à integração da impressora térmica. Ao combinar a base sólida fornecida pela universidade com as experiências práticas nos estágios, fui capaz de superar os desafios encontrados e alcançar sucesso na integração da impressora térmica com o aplicativo *mobile* e *backend*.

Além disso, os estágios me permitiram trabalhar em um ambiente real de desenvolvimento de software, onde pude aprender a lidar com prazos, colaborar com equipes e enfrentar desafios reais. A experiência adquirida durante esses estágios foi valiosa para me preparar para os desafios encontrados durante todo o desenvolvimento da aplicação.

No geral, a universidade e os estágios desempenharam um papel fundamental em minha preparação, fornecendo uma base sólida e as habilidades técnicas necessárias para enfrentar os desafios enfrentados na integração da impressora térmica. Essas experiências me ajudaram a superar os obstáculos e a encontrar soluções eficazes para garantir o sucesso do projeto.

### 5.1 Desenvolvimento

Durante o desenvolvimento do *backend*, foram realizadas diversas etapas de produção para garantir um sistema seguro e eficiente para o gerenciamento de múltiplos restaurantes. A seguir, são listadas as etapas e o tempo aproximado dedicado a cada uma delas:

- Análise e Planejamento: Nesta fase inicial, foi realizada uma análise das necessidades e requisitos do sistema, além do planejamento das funcionalidades e da estrutura do banco de dados. Essa etapa durou cerca de uma semana.
- Autenticação: A implementação da autenticação foi uma das etapas essenciais do desenvolvimento. Foram dedicadas duas semanas para garantir a segurança e a separação dos dados de cada restaurante, permitindo que os usuários autenticados tenham acesso apenas aos dados correspondentes ao restaurante em que estão logados.
- Criação das Rotas: Após a implementação da autenticação, foi necessário criar as rotas e definir as APIs para cada funcionalidade do sistema. Essa etapa demandou cerca de um mês, levando em consideração a criação dos endpoints, a validação dos dados recebidos e o retorno das respostas adequadas.
- *Models* e Relações: A estrutura do banco de dados foi definida por meio da criação das classes Python, mapeando as tabelas e estabelecendo as relações entre os diferentes elementos do sistema, como restaurante, pedido, cliente, carrinho e produtos. Essa etapa demandou cerca de duas semanas, considerando a definição dos modelos e a configuração das relações no ORM.

- Serialização e Desserialização: O Django Rest Framework[6] facilitou a serialização e desserialização dos dados no formato JSON, melhorando a eficiência na comunicação com o *frontend*. Essa etapa levou aproximadamente uma semana, garantindo a correta transformação dos objetos Python em formatos adequados para a comunicação.
- Interface de Administração: O Django Rest Framework[6] também criou automaticamente uma interface de administração, simplificando o gerenciamento dos dados do sistema. Essa etapa demandou cerca de uma semana para personalizar a interface e adequá-la às necessidades específicas do projeto.

Durante o desenvolvimento da aplicação *Mobile* com Jetpack Compose, foram realizadas diversas etapas de produção para garantir um sistema completo e funcional. A seguir, serão listadas as etapas e o tempo aproximado dedicado a cada uma delas:

- Análise e Planejamento: Nesta fase inicial, foram analisadas as necessidades e requisitos da aplicação, além do planejamento das telas e funcionalidades. Essa etapa levou cerca de uma semana.
- Criação das Telas de *Login* e Mudança de Senha: As telas de *login* e mudança de senha são fundamentais para a segurança e autenticação dos usuários. Foram dedicadas duas semanas para implementar essas telas, garantindo uma experiência intuitiva e segura.
- Criação de Usuário e Restaurante: A funcionalidade de criação de usuário e restaurante é essencial para permitir o cadastro de novos usuários e estabelecimentos. Essa etapa demandou aproximadamente duas semanas, levando em consideração a criação das telas de cadastro e a integração com o *backend* para persistência dos dados.
- Listagem de Pedidos: A listagem de pedidos permite aos usuários visualizar os pedidos realizados. Essa etapa foi concluída em cerca de uma semana, considerando a criação da tela de listagem e a integração com o *backend* para obter os dados dos pedidos.
- Conexão com Impressora Térmica por Bluetooth e Impressão de Pedidos: A integração com uma impressora térmica por meio do Bluetooth para imprimir os pedidos é uma funcionalidade diferencial. Foram dedicadas duas semanas para implementar essa integração, garantindo uma comunicação estável e a impressão correta dos dados dos pedidos.
- Mudança de *Status*, *Logout* e Outras Funcionalidades: As funcionalidades adicionais, como a capacidade de mudar o *status* dos pedidos, *logout* do sistema e outras interações, foram implementadas em cerca de uma semana, garantindo uma experiência completa e satisfatória para os usuários.

Foi contemplado no projeto a implementação do *backend* com a autenticação e criação das rotas no Django Rest Framework[6], garantindo segurança e flexibilidade no gerenciamento dos dados dos restaurantes. Já a aplicação *Mobile* com Jetpack Compose ofereceu diversas funcionalidades, permitindo aos usuários realizar *login*, mudar senha, cadastrar usuários e restaurantes, gerenciar pedidos, se conectar com uma impressora térmica por Bluetooth, imprimir pedidos, alterar *status*, *logou* e interagir com outras funcionalidades.

| Ação | Tempo (Semanas) |
|---|---|
| **Backend** | |
| Implementação da Autenticação | 2 |
| Criação das Rotas no Django Rest Framework | 4 |
| Definição dos Models e Relações | 2 |
| Serialização e Desserialização dos Dados | 1 |
| Criação da Interface de Administração | 1 |
| **Mobile (Jetpack Compose)** | |
| Telas de Login e Mudança de Senha | 2 |
| Cadastro de Usuário e Restaurante | 2 |
| Listagem de Pedidos | 1 |
| Integração com Impressora Térmica e Impressão de Pedidos | 2 |
| Mudança de Status, Deslogar e Outras Funcionalidades | 1 |

**Tabela 1: Tempo de desenvolvimento para *Backend* e *Mobile***

O projeto, como um todo, demandou aproximadamente quatro meses para ser concluído como mostra a tabela acima.

## 5.2 Desafios

Ao desenvolver a integração da impressora térmica com o aplicativo *mobile* e *backend*, enfrentei algumas dificuldades que foram desafiadoras. Uma das primeiras foi conseguir integrar as intents no Jetpack Compose, uma vez que estava mais acostumado com a abordagem tradicional usando XML. O Jetpack Compose é uma nova maneira de construir interfaces de usuário reativas e declarativas, o que exigiu um pouco de tempo para aprender a utilizar corretamente as intents e assegurar a comunicação correta entre as telas e componentes.

Outra dificuldade foi lidar com a navegação entre as telas. Com o Jetpack Compose, a navegação é gerenciada por meio do *Navigation Component*, que possui uma abordagem diferente do fluxo de telas convencional. Foi necessário estudar e entender como configurar e navegar entre as intents de forma adequada, garantindo que o usuário pudesse avançar e retroceder nas etapas do pedido sem problemas.

Uma questão importante foi encontrar a melhor forma de passar os dados do pedido pela intent, de modo a garantir que todas as informações relevantes fossem transmitidas corretamente para a impressora térmica. Por exemplo, foi necessário analisar como serializar os dados do pedido em um formato apropriado para serem enviados pela intent e interpretados corretamente no *backend*.

Outro desafio foi lidar com as permissões necessárias para utilizar o Bluetooth no dispositivo móvel. A integração com a impressora térmica exigiu a solicitação e a concessão das permissões adequadas para estabelecer a conexão Bluetooth. Foi necessário entender como lidar com as permissões no contexto do Jetpack Compose e garantir que o aplicativo solicitasse as permissões necessárias de forma adequada, considerando as diferentes versões do sistema operacional e as políticas de permissões específicas de cada plataforma.

Para superar essas dificuldades, busquei recursos e documentações oficiais do Jetpack Compose, do *Android* e do fabricante da impressora térmica. Além disso, recorri à comunidade de desenvolvedores para obter suporte e compartilhar experiências. A colaboração com outros desenvolvedores e a participação em fóruns e grupos de discussão foram fundamentais para encontrar soluções e compartilhar conhecimentos.

## 6 TRABALHOS FUTUROS

Embora o *backend* e o *mobile* para restaurante tenham sido concluídos, ainda existem diversas oportunidades de desenvolvimento e melhorias que podem ser exploradas tanto na versão web para clientes quanto no aplicativo. A seguir, são apresentados alguns trabalhos futuros que podem ser implementados para aprimorar a plataforma:

- Desenvolvimento da Versão Web para Clientes: Uma das melhorias a serem consideradas é o desenvolvimento de uma versão web, voltada para os clientes. Isso permitiria que os clientes acessem a plataforma através de um navegador responsivo, para escolher os itens do cardápio e fazer seus pedidos.
- Implementação de Recursos de Busca: Uma adição importante seria a implementação de recursos de busca tanto na versão *mobile* quanto na versão web. Isso permitiria que os clientes pesquisassem por pedidos específicos, clientes ou produtos, facilitando a localização de informações relevantes de forma rápida e eficiente.
- Integração com Serviços de Notificação: A integração com serviços de notificação, como push notifications, pode ser implementada para manter os clientes informados sobre o status dos pedidos, promoções especiais, atualizações de cardápio, entre outros. Isso ajudaria a aumentar o engajamento e a interação dos usuários com o aplicativo.
- Integração de Redes Sociais: Integração com as redes sociais, permitindo que os usuários compartilhem seus pedidos, avaliações ou experiências gastronômicas com seus amigos e seguidores. Isso poderia aumentar a visibilidade dos restaurantes e atrair mais clientes em potencial.
- Implementação de Avaliações e Comentários: Adição de um sistema de avaliações e comentários para os restaurantes e produtos. Isso permitiria que os clientes compartilhassem suas experiências e opiniões, ajudando outros usuários a tomar decisões informadas ao escolher um restaurante ou prato.

## REPOSITÓRIO DO PROJETO

O código-fonte e a documentação do projeto estão disponíveis no seguinte repositório do GitLab: https://gitlab.com/isaiasmtp/smart-restaurant.

## AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão à minha família por todo o apoio indispensável que me foi fornecido ao longo da minha vida e, especialmente, durante minha jornada acadêmica. Sem o suporte e encorajamento de cada um, eu não estaria onde estou hoje. Minha mãe tem sido uma fonte constante de inspiração e força, sempre dedicada a me ajudar em todas as situações. Meu pai e minhas irmãs também merecem um agradecimento especial por todo o suporte e encorajamento que me deram ao longo dos anos. Suas palavras de incentivo e seu apoio emocional foram fundamentais para minha perseverança e determinação. Agradeço à minha namorada por todo o apoio, sua presença, suporte e compreensão foram um apoio essencial para enfrentar os desafios acadêmicos e profissionais. Sou grato aos professores e alunos do curso de Ciência da Computação da UFCG pelo conhecimento compartilhado ao longo dos anos. Também quero agradecer aos meus amigos por estarem sempre presentes para me ajudar. Por fim, gostaria de expressar um agradecimento especial ao Professor Cláudio Campelo por atuar como meu orientador. Sua orientação e expertise foram fundamentais para o sucesso deste trabalho.

## REFERÊNCIAS

[1] Django Software Foundation [n. d.]. *Django Documentation.* Django Software Foundation. https://docs.djangoproject.com/
[2] [n. d.]. Kotlin Documentation. https://kotlinlang.org/docs/. Acesso em: 8 de junho de 2023.
[3] [n. d.]. Python Documentation. https://docs.python.org. Acesso em: 8 de junho de 2023.
[4] 2023. LocalStorage in Android. Documento interno. https://developer.android.com/reference/android/webkit/WebView#localStorage
[5] api-design-google [n. d.]. *API Design Guide (Google).* Retrieved June 8, 2023 from https://cloud.google.com/apis/design
[6] Tom Christie. 2019. *Django REST framework - Powerful and flexible toolkit for building Web APIs.* Independently Published.
[7] Android Developers. 2023. *Android Intents.* Google. https://developer.android.com/guide/components/intents-filters
[8] Django Software Foundation. 2023. *Django Applications.* Django Software Foundation. https://docs.djangoproject.com/en/4.2/ref/applications/
[9] Django Software Foundation. 2023. *User Objects.* Django Software Foundation. https://docs.djangoproject.com/en/4.2/topics/auth/default/#user-objects
[10] G1. 2021. *Comércio eletrônico: comida por delivery e supermercados são categorias que mais crescem na pandemia.* Retrieved June 8, 2023 from https://g1.globo.com/economia/noticia/2021/05/26/comercio-eletronico-comida-por-delivery-e-supermercados-sao-categorias-que-mais-crescem-na-pandemia.ghtml
[11] G1. 2021. *Entregas de comida disparam em 2020 e apps comemoram lucros.* Retrieved June 8, 2023 from https://g1.globo.com/economia/noticia/2021/02/11/entregas-de-comida-disparam-em-2020-e-apps-comemoram-lucros.ghtml
[12] G1. 2022. *Em 2021, 28 milhões de pessoas no Brasil não usaram a internet, diz IBGE.* Retrieved June 8, 2023 from https://g1.globo.com/tecnologia/noticia/2022/09/16/em-2021-28-milhoes-de-pessoas-no-brasil-nao-usaram-a-internet-diz-ibge.ghtml
[13] jetpack-compose [n. d.]. *Documentação oficial do Jetpack Compose.* Retrieved June 8, 2023 from https://developer.android.com/jetpack/compose
[14] Sarah Johnson. 2023. Integrating Thermal Printers in Android Applications using Kotlin. *International Journal of Android Development* 5, 1 (2023), 15–28.
[15] jwt [n. d.]. *Introduction to JSON Web Tokens.* Retrieved December 14, 2022 from https://jwt.io/introduction
[16] Inc. Square. 2023. *Retrofit.* https://square.github.io/retrofit/