

**ATENÇÃO:** aqui constam somente as páginas que tinham alguma anotação na revisão.

## COMANDA DIGITAL DE BARES E RESTAURANTE

Guilherme Adriano Schafer, Dalton Solano dos Reis – Orientador

Curso de Bacharel em Ciência da Computação

Departamento de Sistemas e Computação

Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

gschafer@furb.br, dalton@furb.br

**Resumo:** Este artigo apresenta o desenvolvimento de uma aplicação web para restaurantes ou lanchonetes que visa auxiliar os clientes na experiência dentro do estabelecimento. O aplicativo foi desenvolvido em C# com Visual Studio 2019 e React Native com Visual Studio Code e bibliotecas React Native. O objetivo principal é facilitar o cliente na visualização do cardápio e na solicitação do pedido e pagamento online, utilizando somente o dispositivo móvel do próprio cliente.

**Palavras-chave:** React Native. C#. Comanda Digital. Restaurantes. Lanchonetes. Pagamento online. Mundipagg

Acho que podes aumentar a descrição do Resumo.

### 1 INTRODUÇÃO

“Acredito que muitos já se depararam com uma cena constrangedora ao serem atendidos por alguém mal-humorado ou com a ‘cara marrada’” (PUDENCE, 2018). Muitos restaurantes estão perdendo clientes devido ao mal atendimento dos funcionários e uma solução para os donos de restaurantes seria investir cada vez mais nas tecnologias atuais, principalmente nos aplicativos móveis.

De acordo com Magalhães (2018)

Se é uma citação direta então usar o formato de acordo com a ABNT.

Antigamente, a tecnologia era considerada um luxo, exclusivo para poucos, mas, com a busca por mais praticidade e agilidade, as modernidades tecnológicas estão disponíveis para todos e são essenciais na rotina de cada um. Por isso, colocar um restaurante na era digital é fundamental para satisfazer clientes exigentes.

“Cada vez mais as pessoas usam aplicativos para tarefas do cotidiano – e reservar mesas em restaurantes é uma delas” (FERNANDA, 2018). Existem muitos aplicativos que facilitam a rotina das pessoas, desde a realização de compras, transportes, comunicação e até no segmento de alimentação para entrega na residência do usuário. A maioria instiga o usuário a ficar dentro da sua residência e poucos incentivam o usuário a ir até o estabelecimento.

Dante do que foi exposto, este trabalho propõe a criação de um aplicativo móvel, para realizar o controle de comandas de um restaurante ou lanchonete, onde o cliente pode visualizar o cardápio do estabelecimento, podendo fazer os pedidos de bebidas ou comidas através do aplicativo sem a utilização de um garçom e até realizar o pagamento utilizando cartões de crédito ou débito.

### 2 FUNDAMENTAÇÃO TEÓRICA

Nessa seção são apresentados os principais assuntos pertinentes à aplicação, transformação digital em foodservice, compras online, PWA (progressive web apps) e os trabalhos correlatos.

#### 2.1 TRANSFORMAÇÃO DIGITAL EM FOODSERVICE

No mundo onde cada vez mais estamos conectados, os estabelecimentos de alimentação precisam se adaptar a nova era, e a transformação digital pode auxiliar o estabelecimento a ser manter no mercado ou até mesmo a entrar no mercado de negócios.

Segundo a organização SiriusDecisions, 67% da jornada do comprador é realizado de modo digital. Para Scuadra (2018), isso significa que mesmo antes de chegar ao estabelecimento, o cliente já pesquisou sobre o restaurante, leu avaliações, procurou os pratos servidos e muitas das vezes já possui uma opinião sobre os produtos e serviços disponibilizados.

A transformação digital não está somente ligada ao marketing digital nas redes sociais, segundo Scuadra (2018), existem alguns assuntos a serem abordados para conseguir efetivamente realizar uma transformação digital, como, se relacionar com os clientes, automatizar a gestão, alimentos de qualidade e uma implantação de um cardápio digital.

ATENÇÃO: seção 2.1 com pouco conteúdo, escrever mais.

Aconselhasse fortemente não ter parágrafos com só UMA frase.

RESTAURANTES

para bares e restaurantes que

Dúvida...  
Por que usou 2 IDEs?

Bares

desenvolveu

um bar ou restaurante

Progressive Web Apps (PWA)

sociais. Segundo

## 2.2 COMPRAS ONLINE

Segundo Schnaider (2020), as compras em lojas virtuais em e-commerce brasileiro registraram um crescimento de 47% no primeiro trimestre, maior alta em 20 anos. Cada vez mais a tecnologia está sendo inserida dentro de várias áreas de negócios, a facilidade e rapidez que a tecnologia chama a atenção ainda mais dos clientes.

Uma pesquisa realizada pelo Paypal em 2019, 80% dos brasileiros usa seus smartphones para comprar online. Além de construir uma aplicação a loja/estabelecimento também precisam, otimizar o processo de checkout do produto, elaborar um app exclusivo, construir um canal de atendimento, investir em campanhas publicitárias e construir um newsletter compatível com o negócio.

Segundo Moshin (2020), 7 a cada 10 compras realizadas pela internet são pagas com cartão de crédito. Dessa maneira o pagamento online, auxilia em mais uma opção para os clientes, aumentando o público a ser alcançado pelo estabelecimento. Não sendo necessário o cliente sair de casa com seus cartões e somente com o próprio dispositivo móvel agrada as pessoas cada dia mais.

### 2.3 PWA

Com a era digital cada vez mais forte no mundo atual, o aumento de dispositivo moveis e aplicativos disponibilizados nas lojas mais conhecidas, instalar um aplicativo para cada loja ou estabelecimento virtual, pode acabar se tornando um problema ao usuário no qual deseja usufruir de toda as regalias disponibilizadas. **Parágrafo c/ 1 só frase.**

“Em 2015, a Google apresentou o Progressive Web Apps, o PWA. Essa tecnologia se tornou tendência no mercado mobile, pois investe na experiencia do usuário, transformando um site em uma tecnologia semelhante a um app, mas sem necessidade de instalação” (Daniel Galvão, 2019).

Com isso um estabelecimento, poderia alcançar um número maior de clientes, pois alcança todos os dispositivos moveis, independe de marca ou sistema operacional, não ocupando espaço de memória no dispositivo e utilizando um tráfego menor de internet.

### 2.4 TRABALHOS CORRELATOS

**ATENÇÃO: seção 2.3 com pouco conteúdo, escrever mais.**

Nessa seção são apresentados três trabalhos correlatos que possuem características semelhantes a proposta deste trabalho. O quadro 1 descreve o trabalho de Araujo (2016), que desenvolveu uma aplicação multiplataforma para localização e recomendação de restaurantes. No quadro 2 o protótipo de Koglin Junior (2018) detalha o desenvolvimento de um aplicativo para pagamento de estacionamento. No quadro 3 o trabalho de Borba Junior (2015) detalha o aplicativo, para controle e agendamento de medicamentos.

Quadro 1 – FIND FOOD: Aplicativo para localização e recomendação de restaurantes

|                                |   |
|--------------------------------|---|
| Referência                     | Araujo (2016)   |
| Objetivos                      | Criar uma aplicação móvel para pagamento localização e recomendação de restaurantes próximos.   |
| Principais funcionalidades     | Conforme Figura 1, permite ao usuário localizar os restaurantes mais próximos da sua localização atual, realizar pesquisas dos restaurantes mais bem avaliados por outros usuários, realizar as avaliações dos vários restaurantes e ainda traçar a rota via Global Position System (GPS) para chegar até o restaurante desejado. |
| Ferramentas de desenvolvimento | Framework Ionic <sup>13</sup> , Google Maps (por meio de Application Program Interface – API ), Global Position System (GPS) <sup>14</sup> .  |
| Resultados e conclusões        | O trabalho desenvolvidó atingiu seus objetivos, que eram apresentar opções de restaurantes próximos à localização atual do usuário e permitir que os usuários avaliem os restaurantes a fim de auxiliar outros usuários na escolha de bons restaurantes.  |

Fonte: elaborado pelo autor.

O aplicativo permite encontrar os restaurantes mais bem avaliados, nas aproximações dos usuários podendo da mesma maneira traçar uma rota para se locomover até o restaurante desejado. Pode-se ainda fazer avaliações positivas ou negativas em relação ao restaurante, disponibilizando essas informações para outros usuários.

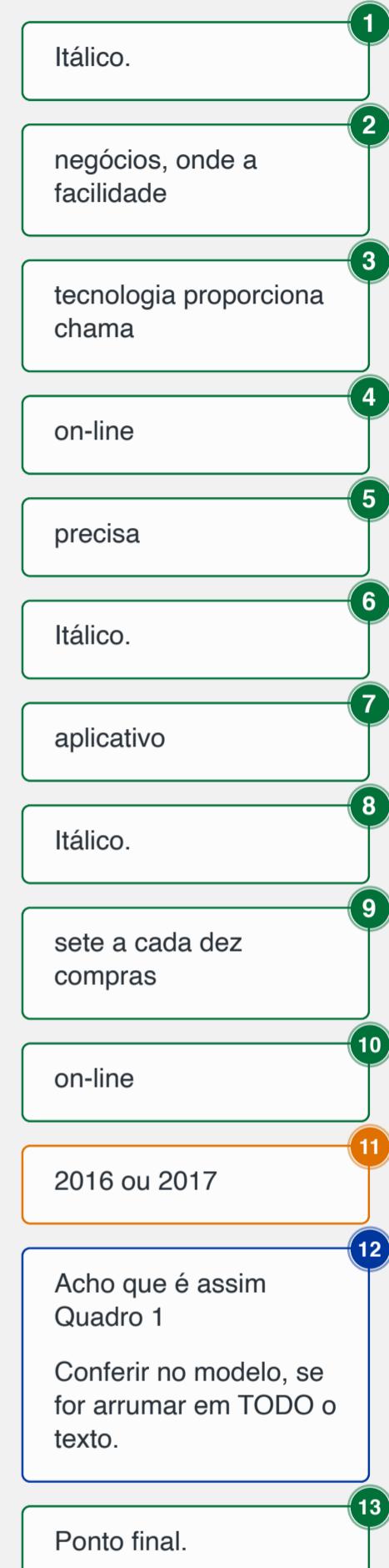
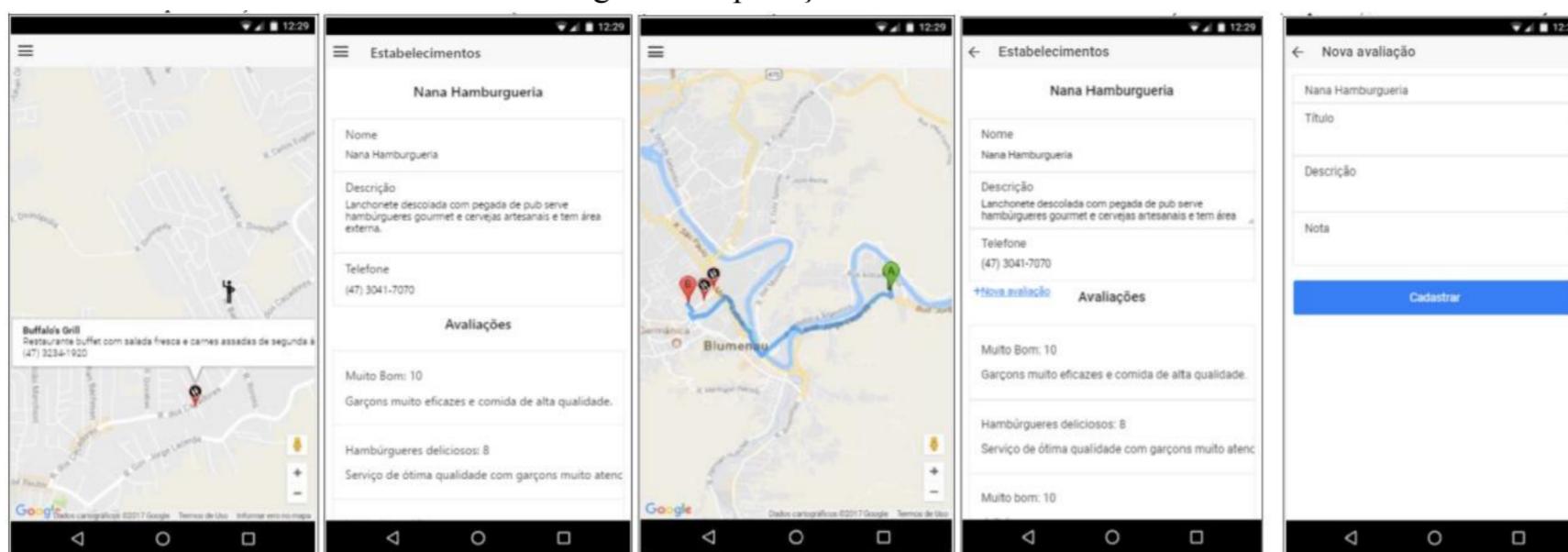


Figura 1 - Aplicação FIND FOOD



Fonte: Araujo (2017).

Quadro 2 – ESTACIONE: Protótipo de aplicativo para pagamento móvel de estacionamento

|                                |  |
|--------------------------------|--|
| Referência                     | Koglin Junior (2016)   |
| Objetivos                      | Criar um protótipo de aplicação móvel para pagamento de estacionamento                               |
| Principais funcionalidades     | Buscar estacionamentos próximos da localização e realizar o pagamento através de cartões de crédito. |
| Ferramentas de desenvolvimento | Framework Ionic, Java, banco de dados Postgress, Node JS, hospedado no Heroku                        |
| Resultados e conclusões        | O aplicativo alcançou o objetivo proposto, apresentando os estacionamentos credenciados em um mapa.  |

Fonte: elaborado pelo autor.

Conforme visualizado na Figura 2, o usuário realiza o login na aplicação e visualiza a localização atual. Através do menu lateral possibilita buscar um estacionamento, incluindo o nome de uma rua para visualizar os estacionamentos no mapa para realizar a estadia e após realizar a estadia o pagamento pode ser feito através do cartão.

1 2016 ou 2018

2 Ponto final.

3 Ponto final.

4 Itálico.



Fonte: Koglin Junior (2018)

5 Ponto final.

Segundo Koglin Junior (2018) o aplicativo alcançou o objetivo proposto, disponibilizando um mapa com as informações dos estacionamentos credenciados, visualizando o tempo de estadia e o pagamento tudo através do seu próprio dispositivo móvel.

Quadro 3 – Aplicativo Mobile para controle e agendamento de medicamentos

|                                |  |
|--------------------------------|--|
| Referência                     | Borba Junior (2015)  |
| Objetivos                      | Criar um aplicativo para uma pessoa ou responsável receber um aviso da necessidade de tomar medicamentos.  |
| Principais funcionalidades     | Cadastrar prescrições diárias a pacientes e receber notificações no horário exato de cada um dos remédios. |
| Ferramentas de desenvolvimento | HTML, CSS, Javascript, Jquery e Bootstrap e framework Apache Cordova.                                      |
| Resultados e                   | O aplicativo atendeu o principal objetivo, que era auxiliar um usuário a tomar os remédios na hora certa.  |

|            |             |
|------------|-------------|
| conclusões | hora exata. |
|------------|-------------|

Fonte: elaborado pelo autor.

O aplicativo auxilia o usuário a controlar esse consumo de remédios para que não haja nenhum esquecimento ou erro na repetição de remédios. O usuário pode cadastrar todos os remédios ou vários pacientes caso o responsável controle o remédio de mais de 1 uma pessoa, possibilitando analisar as prescrições diárias de cada paciente, recebendo notificações no horário exato de cada um dos remédios.

Figura 3 – Aplicação “Aplicativo mobile para controle e agendamento de medicamentos”



Fonte: Borba Junior (2015)

Conforme na Figura 3, primeiramente o usuário deve selecionar uma das opções: Remédios, Pacientes e Prescrições. Em cada uma das opções o usuário tem a possibilidade de cadastrar, editar, visualizar e excluir.

### 3 DESCRIÇÃO

Esta seção apresenta os detalhes de especificação, implementação e operacionalidade da aplicação. A aplicação possui quatro tipos de usuários diferentes e dispõem de comportamentos e telas diferentes para cada tipo. Para tanto, são apresentadas em dois blocos. O primeiro demonstra como foi realizado a especificação e o segundo a implementação e a operacionalidade.

#### 3.1 ESPECIFICAÇÃO

O objetivo da aplicação é melhorar a experiência do cliente nos restaurantes, proporcionando uma rapidez, nas tarefas mais comuns de um restaurante, por exemplo, agilizar a solicitação das comidas e bebidas, o cliente controlar a própria conta, facilitar e agilizar o pagamento dos pedidos solicitados, acompanhar o status do seu pedido.

Por tanto, os Requisitos Funcionais (RF) e Não Funcionais (RNF) são apresentados no Quadro 4 e Quadro 5 respectivamente e foram utilizados como base para o desenvolvimento dessa aplicação. Os requisitos foram especificados antes do início das implementações.

Quadro 4- Requisitos Funcionais

|       |  |
|-------|--|
| RF001 | Permitir o usuário se registrar                    |
| RF002 | Permitir o usuário realizar o login com o cadastro |
| RF003 | Cadastrar comidas e bebidas                        |
| RF004 | Visualizar o cardápio de comidas e bebidas         |
| RF005 | Realizar pedidos de comidas e bebidas              |
| RF006 | Acompanhar o ciclo de vida dos pedidos realizados  |
| RF007 | Realizar o pagamento dos pedidos via aplicação     |
| RF008 | Atualizar o status do pedido pela cozinha          |
| RF009 | Atualizar o status do pedido pelo bar e garçons    |

Fonte: elaborado pelo autor

Quadro 5- Requisitos Não Funcionais

|        |  |
|--------|--|
| RNF001 | Acessar aplicação via QR CODE                            |
| RNF002 | Possibilitar acesso por qualquer dispositivo móvel       |
| RNF003 | Implementar camada do cliente na linguagem React Native  |
| RNF004 | Implementar camada do servidor na linguagem .NET CORE C# |
| RNF005 | Utilizar uma base de dados SQL SERVER                    |

Fonte: elaborado pelo autor

Todos as camadas da aplicação (cliente, servidor, base de dados) estão hospedados na nuvem da Microsoft Azure. O Azure proporciona facilidades no momento de subir a aplicação para a nuvem, por possuir uma variedade de

- 1 auxiliou
- 2 controlar o seu consumo
- 3 remédios (Figura 3).
- 4 Fonte courier.
- 5 diferentes, e
- 6 dispõem
- 7 nos bares e restaurantes
- 8 um bar ou restaurante. Por
- 9 solicitados e acompanhar
- 10 e Requisitos Não
- 11 respectivamente, e
- 12 implementações e serviram para guiar o seu desenvolvimento.
- 13 Itálico.
- 14 ciclo
- 15 estabelecimento
- 16 Ponto final.
- 17 Server
- 18 Ponto final.
- 19 servidor e base de dados

servidores, inclusão de certificados HTTPS, registros na internet, integração fácil com produtos da Microsoft, por exemplo, a camada de servidor desenvolvida em .NET CORE C#.

Figura 4 – Dashboard Azure

The screenshot shows the Microsoft Azure dashboard. At the top, there's a search bar and a user profile. Below it, a navigation bar includes 'Create a resource' (with a plus icon), 'SQL databases', 'DevOps Starter', 'App Services', 'All resources', 'Virtual machines', 'Storage accounts', 'Azure Database for PostgreSQL...', 'Azure Cosmos DB', and 'More services'. A yellow arrow points from the 'Create a resource' button to the text 'Toda figura deve ser citada e explicada no texto antes da figura.' which is overlaid on the dashboard. The 'Recent resources' section lists several items with their names, types, and last viewed times:

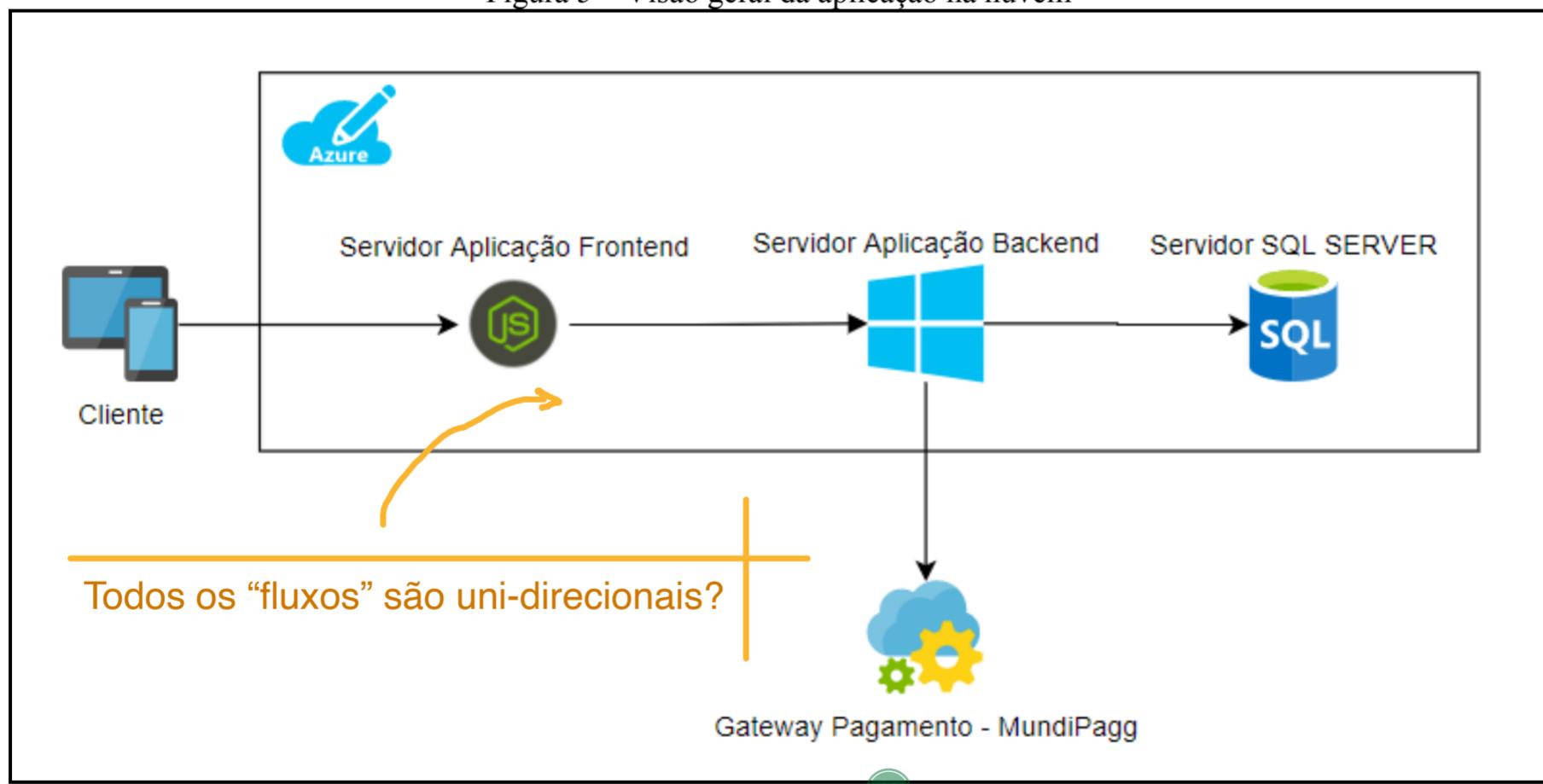
| Name   | Type             | Last Viewed  |
|--|------------------|--------------|
| comandaDigitalDb (comandadigital/comandaDigitalDb) | SQL database     | a week ago   |
| ComandaFrontend                                    | App Service      | a week ago   |
| comandaDigitalBackEnd                              | App Service      | 2 weeks ago  |
| Azure subscription 1                               | Subscription     | a month ago  |
| comandaDigital                                     | Resource group   | a month ago  |
| ASP-comandaDigital-b718                            | App Service plan | 3 months ago |

Fonte: elaborado pelo autor

O código fonte da aplicação ficou hospedado no Github, para uma melhor integração com o Microsoft Azure, por serem os dois produtos Microsoft. O Azure já possui configurações pré-definidas no momento de hospedar a aplicação, assim toda atualização realizada no [github](#) automaticamente realiza um nova versão para cada servidor, realizados algumas etapas configuradas, como o [build](#) da aplicação para garantir que não possui nenhum erro.

A camada do cliente fica hospedado em um servidor NodeJS, por ter sido desenvolvido em React Native. A camada do servidor fica em um servidor Windows, por se tratar de uma aplicação .NET CORE, o Azure disponibiliza uma configuração padrão de hospedagem de aplicações .NET. O cliente acessa o frontend através de uma URL pré-definida no momento da criação do servidor, para que seja realizado a comunicação com a aplicação backend e através da logica incluída no backend os dados são armazenados no banco de dados SQL SERVER.

Figura 5 – Visão geral da aplicação na nuvem



Fonte: elaborado pelo autor

Para realizar os pagamentos dos pedidos através do aplicativo será utilizado o gateway de pagamento da [MundiPagg](#). A plataforma da MundiPagg disponibiliza várias APIs (Application Programming Interface) para realizar

- 1 Ponto final.
- 2 Github
- 3 Itálico.
- 4 CORE. O Azure também
- 5 Itálico.
- 6 Itálico.
- 7 backend. E através
- 8 lógica
- 9 Itálico.
- 10 Server
- 11 Ponto final.
- 12 foi
- 13 Itálico.
- 14 Application Programming Interface (APIs)

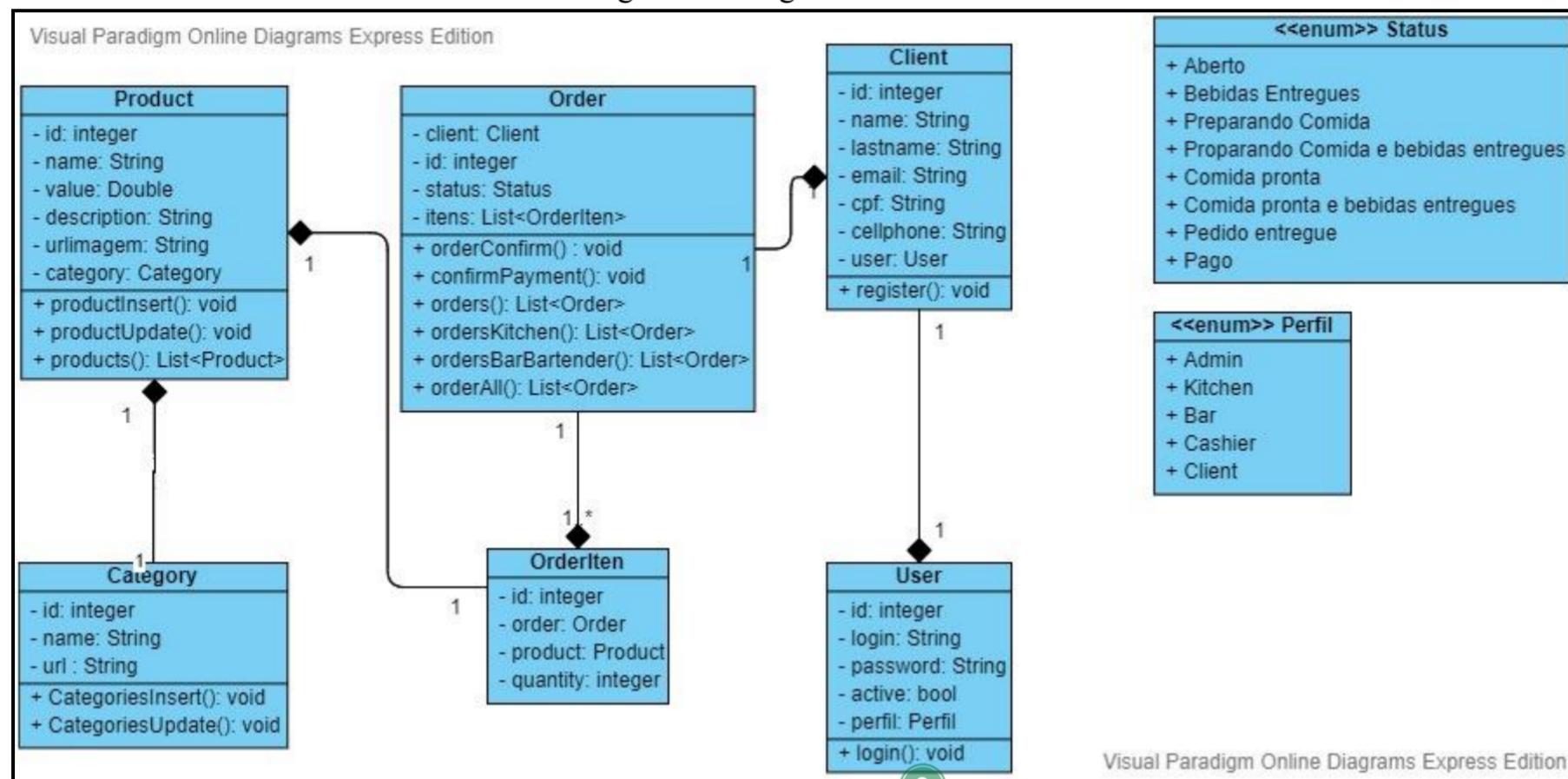
Os nomes de todas as classes, métodos, etc que aparecem no texto de ser em fonte courier.

as cobranças em cartões de créditos, agendas cobranças recorrentes, armazenar dados de cartões com segurança. Além de um dashboard completo, para verificar e extrair dados de todo o tráfego de dados transmitido para a plataforma.

### 3.2 IMPLEMENTAÇÃO

A aplicação foi dividida em classes, cada classe representa algum objeto físico ou fictício, as principais classes são, produto, pedidos e cliente. A classe de produto representa cada uma das comidas ou bebidas disponibilizadas no cardápio. A classe pedidos representa cada solicitação dos produtos disponibilizados no cardápio, seria o pedido realizado pelo um garçom normalmente. As informações do cliente para identificação de cada um e para vincular os pedidos a um cliente está representado na classe cliente.

Figura 6 – Diagrama de classes



Fonte: elaborado pelo autor

Os enumeradores são usados para controlar o ciclo de vida de um pedido, com isso o cliente consegue acompanhar o status atualizado, vindo da cozinha ou bar. O perfil de cada usuário é controlado pelo enum Perfil, onde distingue o tipo de cada usuário e os requisitos que serão apresentados aos usuários na camada frontend.

Figura 7 – Telas de Login e Registro

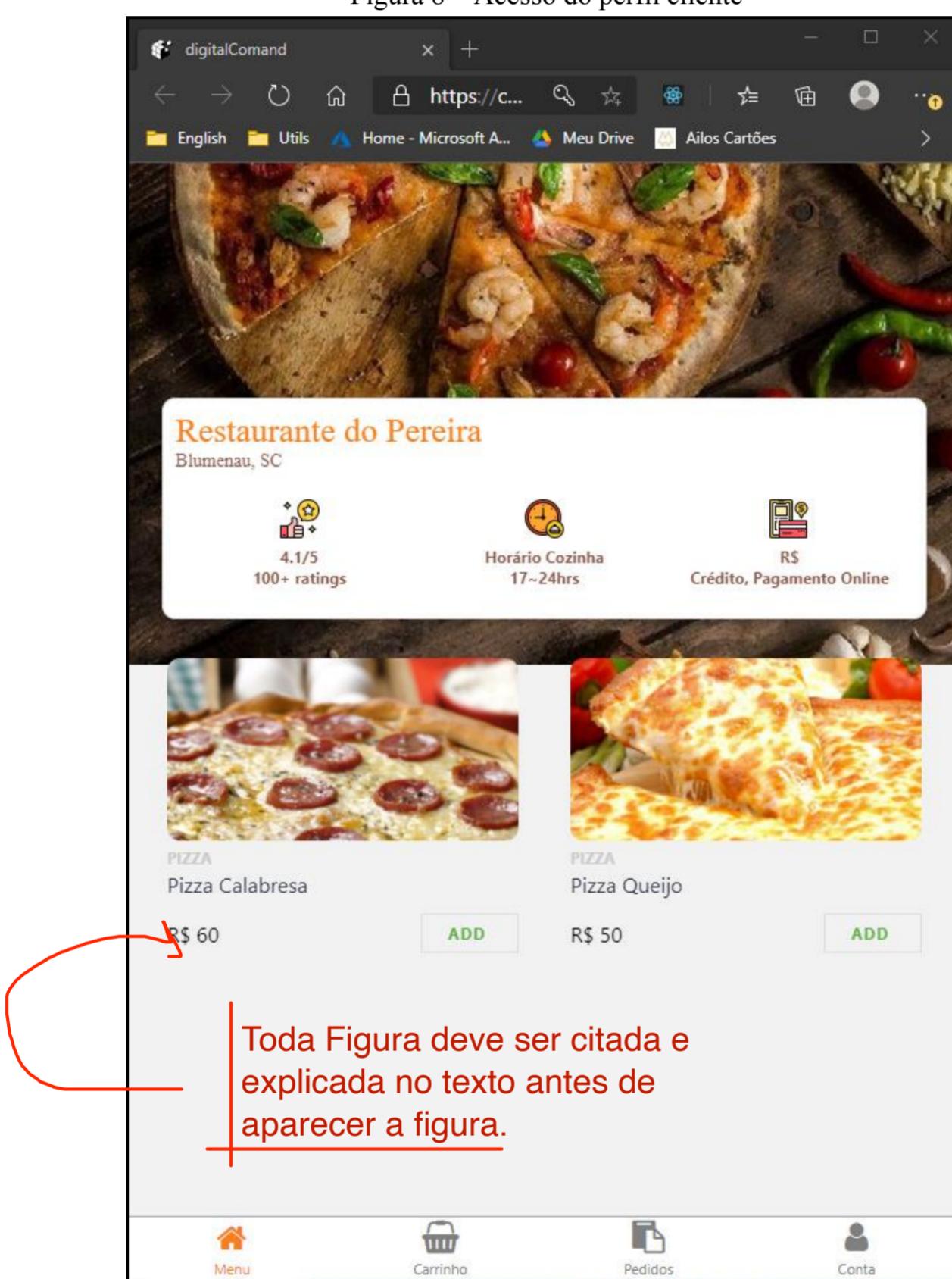
Toda Figura deve ser citada e explicada no texto antes de aparecer a figura.

- 1 tráfego
- 2 Itálico.
- 3 fictício. As
- 4 são: produto
- 5 cliente, está
- 6 Ponto final.
- 7 ciclo
- 8 enumerador Perfil  
No caso Perfil em fonte courier.
- 9 Itálico.
- 10 Ponto final.

Para o cliente acessar a aplicação a camada **fronted** disponibiliza **duas interfaces em React Native**, acessadas via browser, onde o usuário conseguirá realizar o registro com os dados pessoais e a interface de **login**. Os usuários com perfis que serão utilizados pelo **restaurante** já foram criados no momento da implantação da aplicação na nuvem.

O perfil de cliente possui algumas opções de telas em um menu inferior da aplicação, opções como o **Menu/Cardápio**, carrinho para acompanhar os itens adicionados e acompanhar o valor do seu pedido, os pedidos já realizados e seus respectivos **status** e uma opção da conta para realizar o **logoff** da aplicação. Além das informações básicas do **restaurante** que serão visualizados em tela.

Figura 8 – Acesso do perfil cliente



Fonte: elaborado pelo autor.

Para controlar os itens adicionados no carrinho através do botão **ADD** de cada produto do cardápio, a aplicação **fronted** armazena essas informações no próprio browser do dispositivo através de uma biblioteca **AsyncStorage** no qual utiliza o **LocalStorage** do browser. O método **handleCart** chamado através do botão **ADD**, armazena as informações do produto com a quantidade selecionada, dentro do browser, sem a necessidade de requisição para o servidor **backend**.

- 1** Remover.
- 2** Itálico.
- 3** Itálico.
- 4** estabelecimento
- 5** aplicação: opções
- 6** Fonte courier.
- 7** status, e uma
- 8** Itálico.
- 9** estabelecimento
- 10** Fonte courier.
- 11** Itálico.
- 12** Fonte courier.
- 13** Fonte courier.
- 14** Fonte courier.
- 15** Não itálico.  
Fonte courier.
- 16** Itálico.

Figura 9 – Método handleCart

```

handleCart = (item, qty) => {
  const itemDetails = Object.assign(item, { "Quantity": qty });
  const itemKey = `orderDetails${itemDetails.id}`;

  AsyncStorage.setItem(itemKey, JSON.stringify(itemDetails)).then(
    () => {
      this.getCartItems();
    }
  );
}

```

Fonte courier.

A figura deve ser citada no texto.

Fonte: elaborado pelo autor.

O método “getCartItems” chamado dentro do **handleCart**, obtém as informações do produto incluído dentro do LocalStorage para somar os valores totais de preço, quantidade de produtos selecionados e remover os produtos que foram zerados. Além de realizar esses cálculos através do método “getCartItemDetails”, ainda é realizado a atualização das variáveis de controle do frontend para visualizar a barra de itens adicionados no carrinho através do método “updateCartState”.

Figura 10 – Método getCartItems

```

getCartItems = () => {
  AsyncStorage.getAllKeys().then(orderList => {
    if (orderList.length !== 0) {
      AsyncStorage.multiGet(orderList).then(response => {
        let cartDetails = this.getCartItemDetails(response);
        this.updateCartState(cartDetails.totalQuantity, cartDetails.totalPrice, cartDetails.allItems);
      });
    } else {
      this.setState({
        allItems: []
      });
    }
  });
}

```

2 Remover aspas duplas.  
Fonte courier.

3 Fonte courier.

4 handleCart obtém

5 Fonte courier.

6 Remover aspas duplas.  
Fonte courier.

7 Itálico.

8 Remover aspas duplas.  
Fonte courier.

9 Fonte courier.

10 APÊNDICE

11 Remover aspas duplas.  
Fonte courier.

12 inferior. Dessa

Fonte: elaborado pelo autor.

Como pode ser visto no apêndice A, após a chamada do método “updateCartState” a barra de visualização parcial do carrinho aparece próximo do menu inferior, dessa maneira o cliente realiza seu controle de itens adicionados antes mesmo de visualizar o carrinho por completo.

Figura 11 – Método confirmar pedido

```

[HttpPost]
[Route("orderConfirm")]
[Authorize]
0 references
public IActionResult OrderConfirm([FromBody] List<OrderDto> orderItens)
{
  try
  {
    var userLogin = User.Claims.LastOrDefault(c => c.Type == ClaimTypes.Name).Value;

    orderProvider.ConfirmOrder(orderItens, userLogin);
    return Ok();
  }
  catch (Exception e)
  {
    return BadRequest(e.Message);
  }
}

```

Fonte:elaborado pelo autor

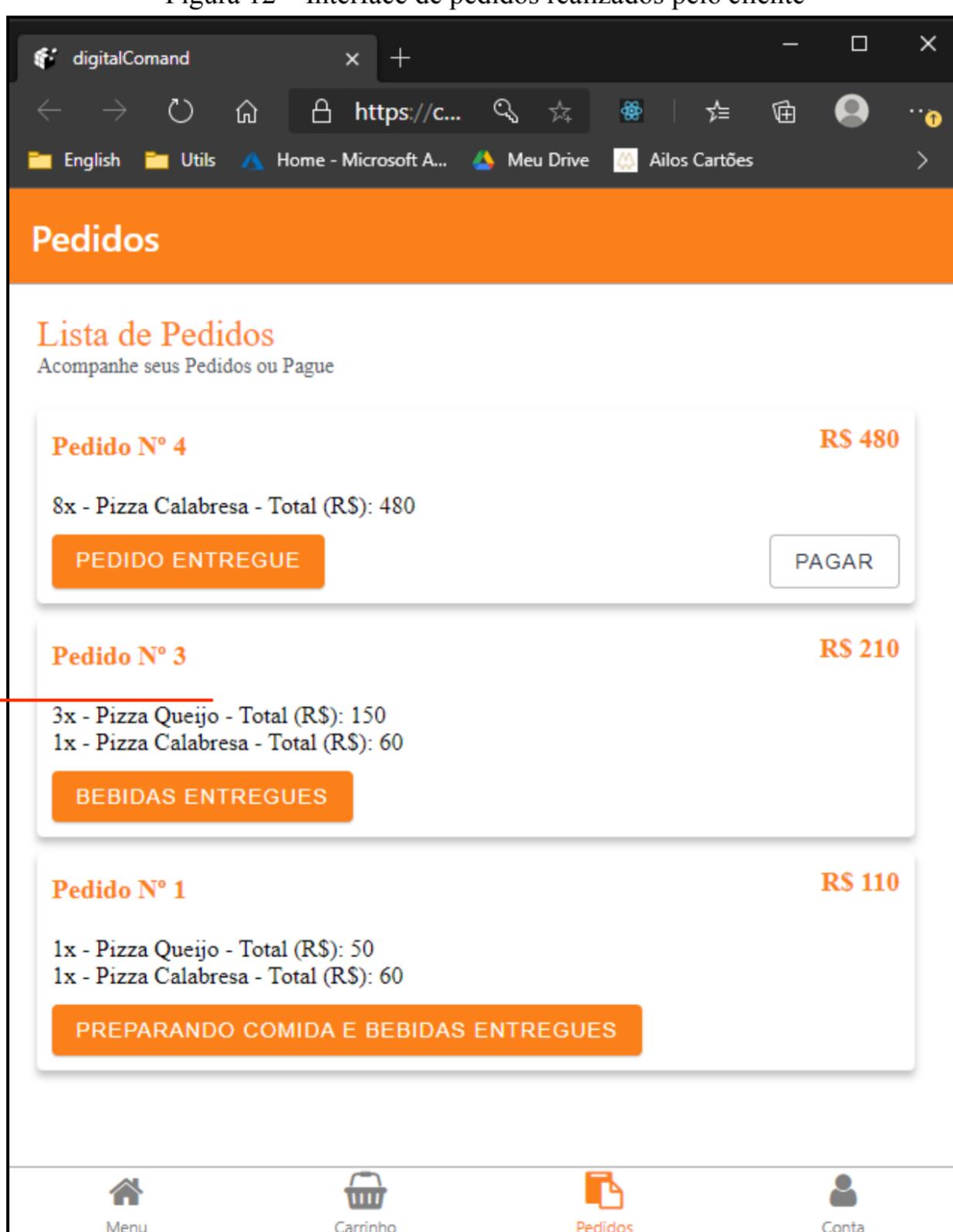
A citação no texto da figura deve aparecer antes da figura.

Para tanto, ao clicar no menu **Carrinho** a visualização do carrinho por completo será apresentada, com isso os produtos adicionados, o valor e quantidade de cada um e o valor total do pedido serão demonstrados em tela. Além de um botão “**Confirmar Pedido**”, onde ao ser clicado será direcionado pela primeira vez em todo o ciclo de vida do carrinho para a camada do servidor, para ser persistido as informações do pedido e ser apresentado o pedido para preparação da cozinha e do bar.

A comunicação com a camada do servidor é realizada através de **endpoint REST**, como pode ser visto na Figura 11 acima, ao confirmar o pedido será consumido o **endpoint de confirmar o pedido**, no qual possui o objetivo de persistir todas as informações do pedido na base de dados. Para identificar qual é o cliente/usuário que está realizando a solicitação do pedido, é obtido o usuário através do **token** passado ao servidor, como pode ser visto na Figura 11.

No menu **Pedido**, no canto inferior, o cliente pode visualizar todos os pedidos realizados e acompanhar o status de cada um, a opção de pagamento via cartão de crédito será visualizada quando o pedido já foi totalmente entregue.

Figura 12 – Interface de pedidos realizados pelo cliente



Fonte: elaborado pelo cliente.

A figura deve ser citada e explicada no texto.

Para realizar o pagamento o cliente será direcionado a uma interface para adicionar os dados do cartão, após adicionar os dados e clicar no botão **Pagar**, será consumido um **endpoint “confirmPayment”**, no qual executa a comunicação com a Mundipagg, para assim validar com a operadora do cartão e receber uma confirmação do pagamento.

Toda a comunicação com a Mundipagg é através de **API REST** (Representational State Transfer), utilizando uma autenticação de chaves geradas pela própria plataforma, a plataforma disponibiliza interfaces para visualizar todas as informações armazenadas, por exemplo, todo o histórico de cobranças realizados com o respectivo status de cada

- 1 Fonte courier.
- 2 Remover aspas duplas.  
Fonte courier.
- 3 Fonte courier.
- 4 Remover.
- 5 Norma ABNT, o texto que cita a figura deve aparecer antes da figura.
- 6 Figura 11. Ao
- 7 Fonte courier.
- 8 pedido é
- 9 Fonte courier.
- 10 Fonte courier.
- 11 um. A
- 12 cartão. Após
- 13 Fonte courier.
- 14 Fonte courier.
- 15 Pagar será
- 16 Remover aspas duplas.  
Fonte courier.
- 17 API REpresentational State Transfer (REST),
- 18 plataforma. A plataforma

cobrança, clientes cadastrados automaticamente ao gerar a cobrança, possibilidade de tentar novamente gerar a cobrança em caso de alguma falha.

Figura 13 – Plataforma MundiPagg

| ID                  | CÓDIGO      | NOME              | VALOR   | STATUS | CRIADO EM             |
|---------------------|-------------|-------------------|---------|--------|-----------------------|
| or_dMjGj0ULNhABeQw  | 68RMUJ2XU1G | Guilherme Schafer | R\$0.01 | Pago   | 24/11/2020 - 14:50:13 |
| or_vbp1OLc5s01jMzn  | R14WAN3SGP  | guilherme schafer | R\$0.01 | Pago   | 17/11/2020 - 21:42:59 |
| or_Mr5K4AmTDFj4R3dm | Y74EFRV57Y  | guilherme schafer | R\$0.01 | Pago   | 17/11/2020 - 21:42:27 |
| or_d7g0RMQuTWRzx8w  | 17WVFEIMN8  | guilherme schafer | R\$0.01 | Pago   | 17/11/2020 - 21:15:44 |
| or_g2nQp4nCVqB8Rk1  | 63DHXJA9JG  | guilherme schafer | R\$0.01 | Pago   | 17/11/2020 - 20:58:53 |
| or_EOMa3g8fGIW0QR8K | 59VY1F4FN9  | guilherme schafer | R\$0.01 | Pago   | 17/11/2020 - 19:17:23 |
| or_vGKZIRPsmHdRZLk3 | 8JNZF2556D  | guilherme schafer | R\$0.01 | Pago   | 12/11/2020 - 15:39:11 |
| or_NykbaLvlitQv2d8B | 83Q0HGQHZ8  | guilherme schafer | R\$0.01 | Pago   | 12/11/2020 - 15:36:08 |
| or_OyVlxji7GuMjLRn  | LRWWSVJAEX  | guilherme schafer | R\$0.01 | Falha  | 12/11/2020 - 15:19:50 |
| or_3X7gdZmivigJLp54 | 4EN9HWAY6   | guilherme schafer | R\$0.01 | Falha  | 12/11/2020 - 15:15:44 |
| or_Jw9y0Z7iaeCibK20 | L6KNHE9C9R  | guilherme schafer | R\$0.01 | Falha  | 12/11/2020 - 15:11:43 |
| or_3kna7e345ruj0v11 | XY3WH7PID3  | guilherme schafer | R\$0.01 | Falha  | 12/11/2020 - 14:29:14 |

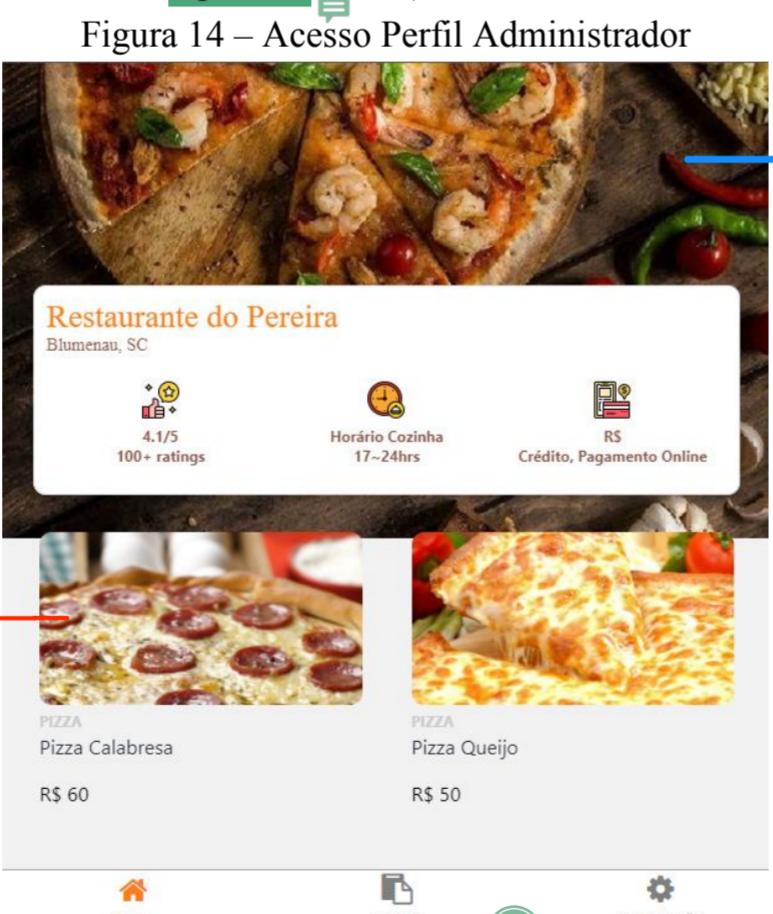
Fonte: elaborado pelo autor

A figura deve ser citada no texto.

A plataforma contém um ambiente de testes para a simulações de todos as funções disponibilizadas, no caso das cobranças em cartão de crédito, utilizado nessa implementação, existem números de cartões fictícios no qual simulam todos os status de retorno possíveis em uma cobrança. Como pode ser visto na Figura 13 acima, algumas cobranças foram realizadas com sucesso e outras com falha em um ambiente de teste (conforme apresentado na parte superior esquerda da Figura 13).

Para realizar a cobrança o método “confirmPayment”, utilizou o endpoint de Pedidos, quando gerado um pedido na plataforma MundiPagg, automaticamente é gerado uma cobrança no cartão do cliente, passando alguns dados do cliente para identificação, além dos dados do cartão para a cobrança (conforme pode ser visto no Apêndice B).

O cardápio online disponibilizado para o cliente visualizar os produtos e realizar os pedidos, conforme descritos acima, serão populados através de dois cadastros que serão existentes no perfil de administrador, os cadastros de categorias do produto e os produtos. Os cadastros serão acessados através da opção Configurações na barra de menu no canto inferior (conforme pode ser visto no Apêndice C e D).



Fonte: elaborado pelo autor

1 cobrança, e possibilidade

2 Ponto final.

3 plataforma MundiPagg contém

4 disponibilizadas. No

5 crédito utilizado

6 implementação podem usar números

7 fictícios, no

8 Remover.

9 ABNT, texto que cita figura deve aparecer antes da figura.

10 Fonte courier.

11 Pedidos. Quando

12 Remover vírgula.

13 Remover aspas duplas.

14 Fonte courier.

15 APÊNDICE

16 que existem no

17 Fonte courier.

18 APÊNDICE

19 Ponto final.

O ciclo de vida de um pedido realizado pelo cliente, pode haver alguns status, aonde irão indicar o momento que se encontra o pedido do cliente, esses status serão atualizados principalmente por dois perfis, cozinha e bar/garçom. A cozinha irá atualizar o status do pedido em dois momentos, quando estiver preparando a comida do pedido solicitado e quando estiver com os pratos prontos. O bar/garçom irá atualizar em outros momentos, quando entregar as bebidas solicitadas e quando entregar as comidas preparadas pela cozinha.

Após todos os produtos solicitados pelo cliente foram preparados e entregues ao cliente, o status do pedido será Pedido Entregue, nesse momento será liberado ao cliente a opção de realizar o pagamento e com isso o ciclo de vida do pedido se encerra para pedido Pago.

A rotina que executa todas as atualizações do pedido, menos o status de Pago, é o método `updateStatusOrder`, no qual identifica de qual perfil está sendo atualizado o status, através dos parâmetros recebidos, irá atualizar o status do pedido conforme o status atual.

Figura 15 – Método `UpdateStatusOrder`

```
public void UpdateStatusOrder(OrderDto orderDto)
{
    var order = unitOfWork.OrderRepository.GetOne(o => o.Id == orderDto.Id);

    if(order != null)
    {
        if(orderDto.Status == (int)Status.DrinksSent)
        {
            if(order.Status == Status.Open)
            {
                order.Status = Status.DrinksSent;
            }
            else if(order.Status == Status.InProgress)
            {
                order.Status = Status.InProgressDrinksSent;
            }
            else if (order.Status == Status.FoodFinishedAndDrinksSent)
            {
                order.Status = Status.OrderSent;
            }
            else if(order.Status == Status.FoodFinished)
            {
                order.Status = Status.FoodFinishedAndDrinksSent;
            }
        }
        else if (orderDto.Status == (int)Status.InProgress)
        {
            if (order.Status == Status.Open)
            {
                order.Status = Status.InProgress;
            }
            else if (order.Status == Status.DrinksSent)
            {
                order.Status = Status.InProgressDrinksSent;
            }
        }
        else if(orderDto.Status == (int)Status.FoodFinished)
        {
            if (order.Status == Status.InProgressDrinksSent)
            {
                order.Status = Status.FoodFinishedAndDrinksSent;
            }
            else if (order.Status == Status.InProgress)
            {
                order.Status = Status.FoodFinished;
            }
        }
        else if(orderDto.Status == (int)Status.OrderSent)
        {
            order.Status = Status.OrderSent;
        }
    }

    unitOfWork.Commit();
}
```

A figura deve ser citada e explicada no texto.

Qual método?

Fonte: elaborado pelo autor

Quando o método recebe por parâmetros “DrinkSent” ou “OrderSent”, a aplicação identifica que a atualização está sendo realizada pelo perfil de bar/garçom e analisando o status atual do pedido atualiza para o que faz o sentido no ciclo de vida do pedido. Ao receber “InProgress” ou “FoodFinished”, é identificado que o prato está sendo preparado ou já se encontra pronto, portanto, a atualização está sendo realizado pelo perfil da cozinha.

As interfaces liberadas para os perfis cozinha e bar/garçom são praticamente as mesmas, com alguns ajustes realizados no momento de chamar a aplicação backend para atualizar o status. Ao realizar o login, são apresentados os pedidos realizados por clientes no qual necessitam de alguma ação do respectivo perfil, por exemplo, se todos os pratos

cliente. Esses

Fonte courier.

Entregue. Nesse

Pedido Pago

Fonte courier.

Fonte courier.

Fonte courier.

recebidos, e irá

updateStatusOrder. No

Fonte courier.

Ponto final.

Remover aspas duplas.

Fonte courier.

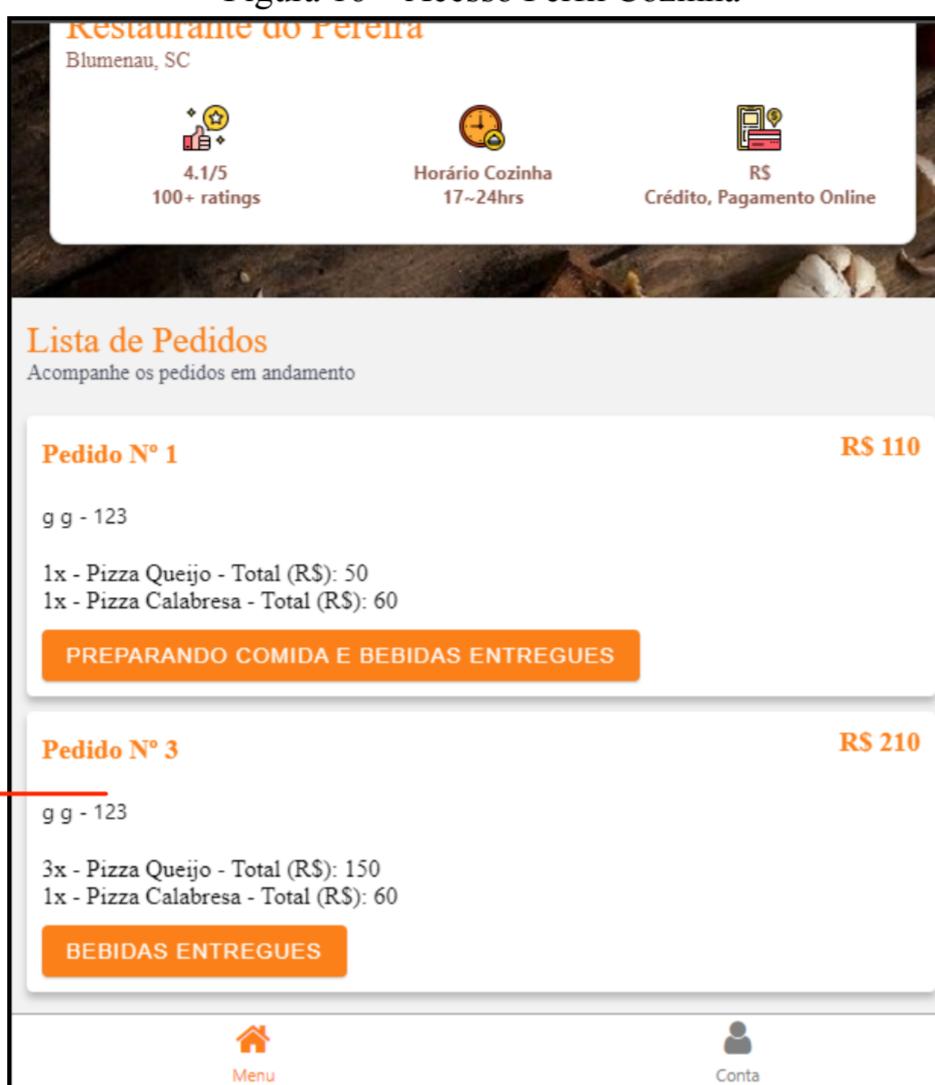
Itálico.

Itálico.

perfil. Por

do pedido já foram preparados e o pedido foi atualizado pela cozinha, não será mais apresentado na interface. O mesmo acontece para o perfil de bar/garçom, ao cliente realizar o pagamento do pedido.

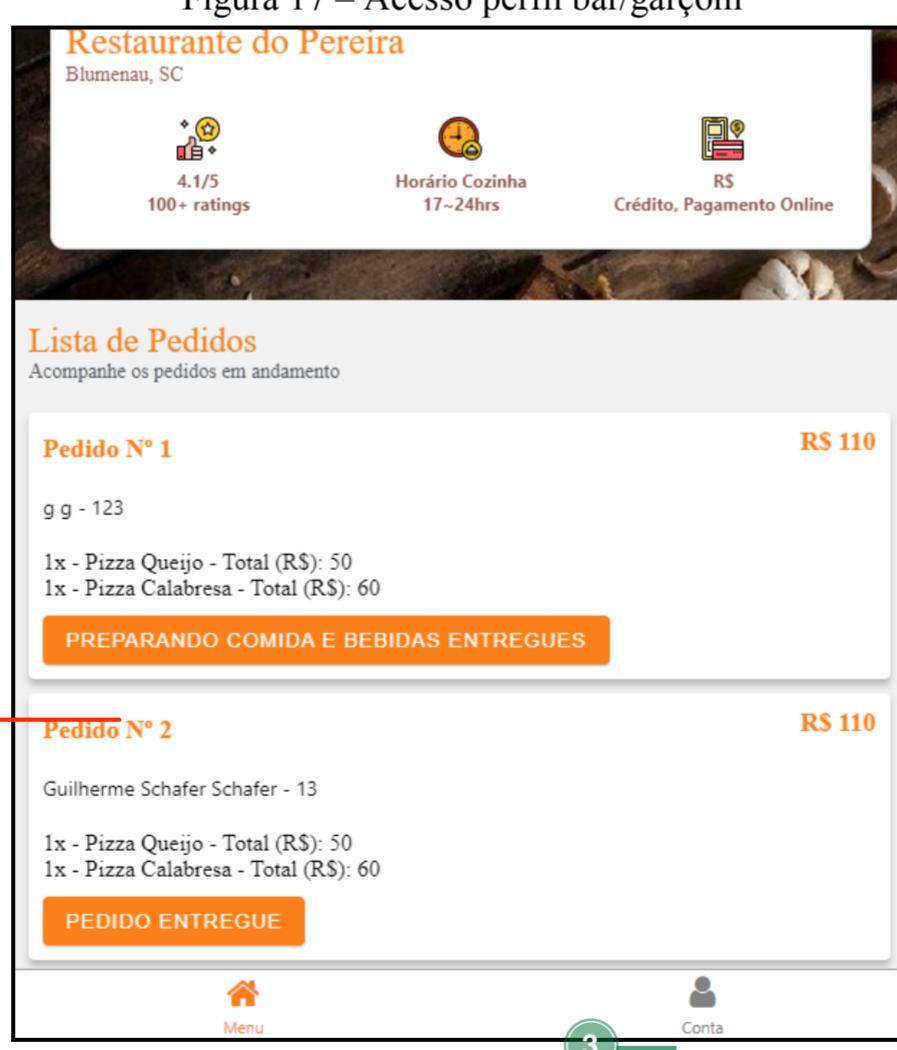
Figura 16 – Acesso Perfil Cozinha



Fonte: elaborado pelo autor

A atualização do status do pedido por um perfil, vai impactar na interface do outro, ou seja, caso a cozinha atualize o pedido indicando que a comida se encontra pronta, o status do pedido será atualizado automaticamente na interface do bar/garçom, para indicar ao garçom a necessidade de entregar o prato diretamente ao cliente.

Figura 17 – Acesso perfil bar/garçom



1 Ponto final.

2 outro. Ou

3 Ponto final.

Todas as consultas realizadas pelo frontend para apresentar os dados das interfaces, todas foram feitas através de endpoints API REST, construídos na aplicação do servidor backend. Praticamente todos os endpoints do servidor necessitam de um Bearer Token, gerado no momento do login, onde dentro do token criptografado, existem algumas informações como o perfil, usuário logado, data de expiração do token. O token é gerado pelo método GenerateToken apresentado na Figura 18.

Figura 18 – Método GenerateToken

```
 1  U references
 2  public static string GenerateToken(User user)
 3  {
 4    var tokenHandler = new JwtSecurityTokenHandler();
 5    var key = Encoding.ASCII.GetBytes(Settings.Secret);
 6    var tokenDescriptor = new SecurityTokenDescriptor
 7    {
 8      Subject = new ClaimsIdentity(new Claim[]
 9      {
10        new Claim(ClaimTypes.Name, user.Login.ToString()),
11        new Claim(ClaimTypes.Role, user.Perfil.ToString())
12      }),
13      Expires = DateTime.UtcNow.AddHours(2),
14      SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
15    };
16    var token = tokenHandler.CreateToken(tokenDescriptor);
17    return tokenHandler.WriteToken(token);
18 }
```

Fonte: elaborado pelo autor

Na camada de frontend o método bootstrapAsync valida o perfil adicionado ao token para realizar o direcionamento correto para as rotas da aplicação e caso a data de expiração do token já tenha sido alcançado realiza a remoção das informações do LocalStorage do browser e direciona para tela de login novamente.

Figura 19 – Método bootstrapAsync

```
 1  _bootstrapAsync = async () => {
 2    const userToken = await AsyncStorage.getItem('userToken');
 3
 4    if (userToken == undefined || userToken == null) {
 5      this.props.navigation.navigate('Auth');
 6    }
 7    else {
 8      let decodeToken = jwt_decode(userToken);
 9      console.log(decodeToken);
10      if (decodeToken.exp > (new Date().getTime() + 1) / 1000) {
11        if (decodeToken.role == 0) {
12          this.props.navigation.navigate('HomeAdmin');
13        }
14        else if (decodeToken.role == 1) {
15          this.props.navigation.navigate('HomeKitchen');
16        }
17        else if (decodeToken.role == 2) {
18          this.props.navigation.navigate('HomeBarBartender');
19        }
20        else if (decodeToken.role == 4) {
21          this.props.navigation.navigate('Home');
22        }
23        You, a week ago • telas rotas de cozinha e bar
24      }
25      else {
26        await AsyncStorage.clear();
27        this.props.navigation.navigate('Auth');
28      }
29    }
30    // This will switch to the App screen or Auth screen and this loading
31    // screen will be unmounted and thrown away.
32  };
33 }
```

Fonte: elaborado pelo autor

- 1 Itálico.
- 2 interfaces foram
- 3 Fonte courier.
- 4 Itálico.
- 5 Fonte courier.
- 6 Fonte courier.
- 7 Itálico.
- 8 login. Aonde
- 9 Itálico.
- 10 Itálico.
- 11 Fonte courier.
- 12 Fonte courier.
- 13 Ponto final.
- 14 Itálico.
- 15 Fonte courier.
- 16 Itálico.
- 17 aplicação. E caso
- 18 Itálico
- 19 Fonte courier.
- 20 Itálico.
- 21 Fonte courier.
- 22 Ponto final.

E a comparação com os trabalhos correlatos?

A seção 4.1 está muito “truncada” ..  
descrever mais .. outros testes.  
No servidor, no MundiPagg, etc.

## 4 RESULTADOS

Esta seção apresenta os testes realizados com a aplicação e os resultados obtidos através destes testes. A seção foi dividida em duas subseções. A primeira demonstra os testes das funcionalidades realizadas durante a fase de implementação. A segunda subseção descreve os testes realizados pelo professor Dalton simulando a utilização em um estabelecimento.

### 4.1 TESTE INICIAIS

Durante a implementação da ferramenta foram feitos testes em todas as funcionalidades, em todos os tipos de dispositivos (Android, iOS, Desktop), para tentar encontrar e sanar possíveis falhas. No decorrer dos testes foi visto que houve erros principalmente na parte de interface de usuários, dependendo o tamanho do dispositivo do usuário, algumas informações acabam ultrapassando o limite da tela, por isso foi preciso ajustar alguns textos e modificar a visualização de alguns componentes. Alguns componentes acabavam escondendo alguns campos, conforme eram apresentadas mensagens de validações de campos, por não estar sendo tratado no componente uma barra de rolagem.

Não houve grandes complicações na utilização da aplicação em celulares Android mais antigos, foi utilizado para teste um dispositivo na versão 5.0.2 e não encontrou nenhum problema em abrir a aplicação via chrome v86, foi possível executar praticamente todas as funções do aplicativo.

### 4.2 TESTES COM USUARIOS

A escrever conforme retorno do dalton

Chegassem a fazer um questionário?  
Seria muito bom testar com mais  
usuários, que não estivessem  
envolvidos no desenvolvimento do TCC.

## 5 CONCLUSÕES

Apesar do pequeno número de avaliações, considera-se que a aplicação cumpriu o principal objetivo, facilitando ao cliente do restaurante realizar os pedidos no estabelecimento, apresentando o pedido para as áreas responsáveis e permitindo o pagamento realizado através do aplicativo, dessa maneira diminuindo a interação do cliente com funcionários do restaurante.

A aplicação por ser uma aplicação web desenvolvida em React Native, conseguiu cumprir um segundo objetivo que seria executar em todas as plataformas e isso foi alcançado com sucesso, conforme descrito nos testes, a aplicação funciona até em dispositivos mais antigos, dessa maneira podendo alcançar um maior número de pessoas, nos mais variados tipos de restaurantes.

Por fim, ao realizar os testes finais do aplicativo, foram identificados alguns pontos a melhorar no aplicativo. As possíveis extensões encontradas para este trabalho são sugeridas a seguir:

- implementar um filtro por categoria no cardápio para que o cliente tenha a possibilidade de selecionar a categoria de produtos que mais gosta;
- implementar o perfil para o caixa do restaurante, para controlar os pedidos a serem pagos e até realizar uma baixa nos pedidos dos clientes que desejarem pagar diretamente no caixa;
- implementar outras possibilidades de pagamento, além do cartão de crédito, como cartão de débito, picpay e outras formas, visto que existem plataformas de integração que possibilitam essa melhoria;
- melhorar a visualização dos pedidos na parte do administrador, para que possa ser realizado consultas de pedidos antigos e até extração de relatórios;
- implementar a autenticação via redes sociais;

Conclusão “truncada”.

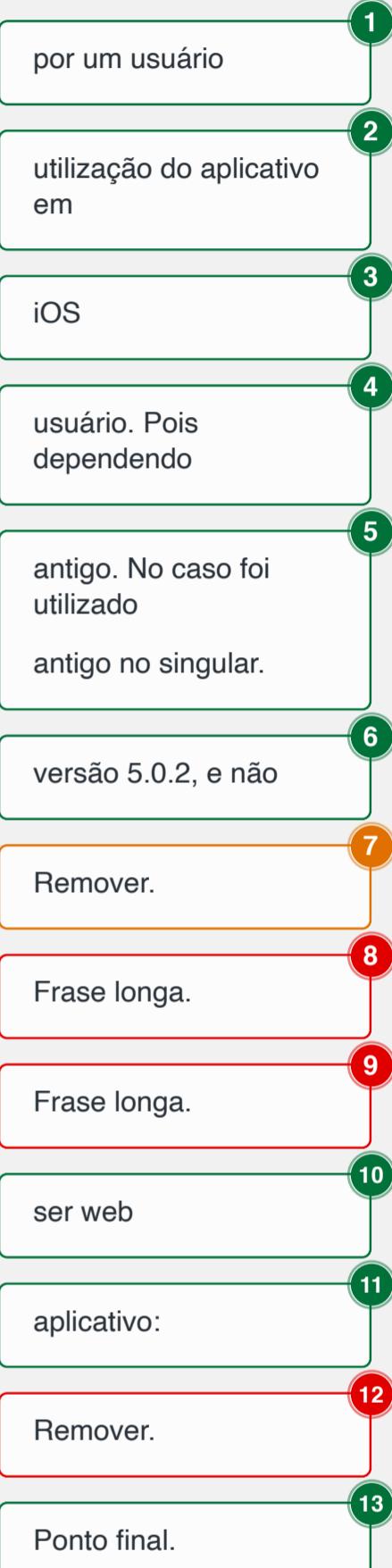
## REFERÊNCIAS

ARAUJO, Thiago Lippel de; **Find Food**: Aplicativo para Localização e recomendação de restaurantes; 2017. 51f Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação – Centro de Ciências Exatas e Naturais; Universidade Regional de Blumenau, Blumenau).

BORBA JUNIOR, José Celso de Borba; **Aplicativo mobile para controle e agendamento de consumo de medicamentos**; 2015. 42f Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação – Centro de Ciências Exatas e Naturais; Universidade Regional de Blumenau, Blumenau).

FERNANDA, Raisa **Conheça as principais tendências tecnológicas em gestão de restaurantes**, [2018]. Disponível em: <https://www.ticket.com.br/blog/inovacao-e-tecnologia/conheca-as-principais-tendencias-tecnologicas-em-gestao-de-restaurantes>. Acesso em: 27 jul. 2018.

<sup>14</sup> GALVÃO, Daniel **Crescimento do mercado mobile altera comportamento sobre uso de apps**, [2019]. Disponível em: <https://digitalks.com.br/artigos/crescimento-do-mercado-mobile-altera-comportamento-sobre-uso-de-apps/>. Acesso em: 28 nov. 2020.



Referência não citada no texto.

KOGLIN JUNIOR, Paulo Arnoldo; **Estacione**: protótipo de aplicativo para pagamento móvel de estacionamento; 2018. 85f Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação – Centro de Ciências Exatas e Naturais; Universidade Regional de Blumenau, Blumenau.

MAGALHÃES, Casa Restaurante na era digital: como satisfazer clientes exigentes?, [2018]. Disponível em:  
<https://www.casamagalhaes.com.br/blog/tecnologia/restaurante-na-era-digital>. Acesso em: 10 out. 2018.

MOHSHIN, Maryam **9 estatísticas sobre compras online para o ano de 2020**, [2020]. Disponível em:  
<https://www.oberlo.com.br/blog/estatisticas-compras-online#:~:text=A%20cada%2010%20compras%20realizadas%20no%20e%2Dcommerce%2C%207%20s%C3%A3o,R%24%2090%20bilh%C3%B5es%20em%202020>. Acesso em: 28 nov. 2020.

NANINI, Alessandro **Tendências para bares e restaurantes: Descubra como duplicar seu Faturamento em 2019**, [2019]. Disponível em: <https://blog.goomer.com.br/tendencias-para-restaurantes-2019>. Acesso em: 02 jan. 2019.

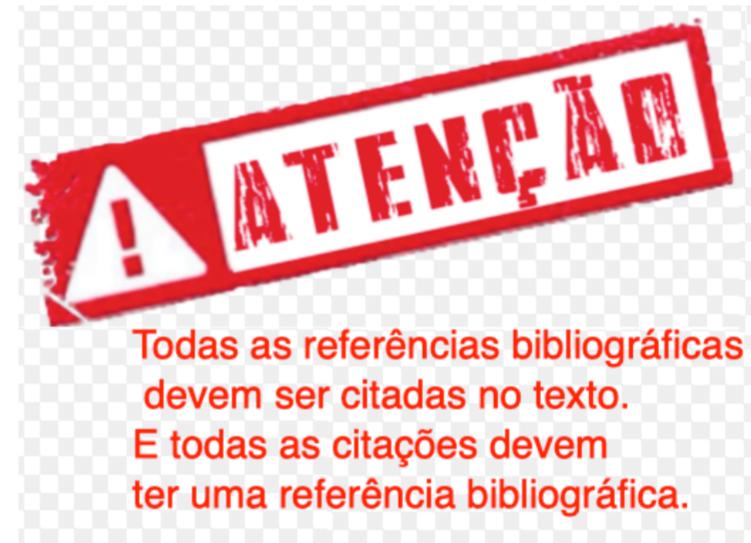
PUDENCE, Edson **Os Cinco maiores problemas no atendimento ao cliente em restaurantes**, [2018]. Disponível em:  
<https://cobizz.com.br/2018/08/09/os-cinco-maiores-problemas-no-atendimento-ao-cliente-em-restaurantes>. Acesso em: 09 ago. 2018.

SCHNAIDER, Amanda **E-commerce cresce 47%, maior alta em 20 anos**, [2020]. Disponível em:  
<https://www.meioemensagem.com.br/home/marketing/2020/08/27/e-commerce-cresce-47-maior-alta-em-20-anos.html>. Acesso em: 28 nov. 2020

SCUADRA, Transformação digital em foodservice: saiba como aplicar no seu restaurante, [2018]. Disponível em:  
<https://www.scuadra.com.br/blog/transformacao-digital-em-foodservice-saiba-como-aplicar-no-seu-restaurante/>. Acesso: 28 nov. 2020.

Referência não citada no texto.

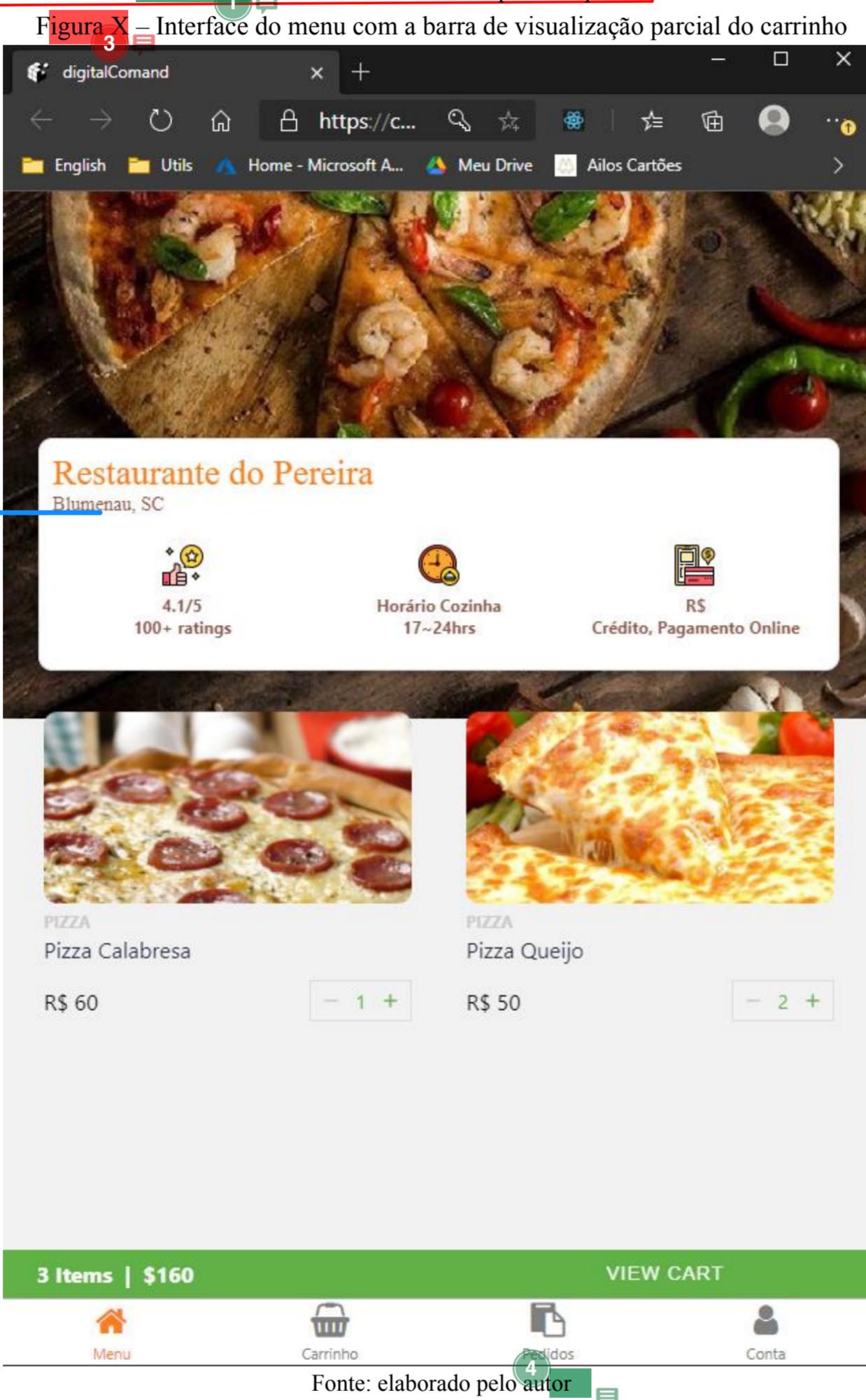
**ATENÇÃO:** Mencionei acima vários momentos para descrever mais .. estais com 15 páginas e podes chegar até 20 páginas.



Todas as referências bibliográficas devem ser citadas no texto.  
E todas as citações devem ter uma referência bibliográfica.

## APÊNDICE A – INTERFACE DO MENU COM ITENS ADICIONADOS NO CARRINHO

Após atualizado as variáveis de controle do carrinho, é apresentado uma barra próxima do menu inferior para o cliente visualizar quantos itens já foram adicionados ao seu carrinho e qual o valor total do pedido no momento, sem a necessidade de clicar no menu “Carrinho” e visualizar o carrinho por completo.



A figura deve ser citada no texto.

- 1 Remover aspas duplas.  
Fonte courier.
- 2 Frase longa.
- 3 Número.

Ponto final.

## APÊNDICE B – MÉTODO CONFIRMPAYMENT

Método de confirmação de pagamento, realiza o consumo do endpoint disponibilizado pela plataforma MundiPagg e recebe o retorno da cobrança gerada na operadora do cartão, além da atualização do status do pedido caso receba uma confirmação do pagamento.

Figura X - ConfirmPayment

```

public async Task<bool> ConfirmPayment(PaymentDto paymentDto)
{
    var order = unitOfWork.OrderRepository.GetOne(o => o.Id == paymentDto.Id);
    var totalOrder = 0.00;

    foreach (var item in order.ListOfItens)
    {
        totalOrder += (item.Quantity * item.Product.Value);
    }

    if (order != null)
    {
        var mundiPaggItems = new MundiPaggItemDto();
        mundiPaggItems.items = new List<MundiPaggItensDto>();
        mundiPaggItems.items.Add(new MundiPaggItensDto
        {
            Amount = 1, // (int)totalOrder,
            Description = "Pedido nº" + order.Id,
            Quantity = 1
        });

        var customer = new MundiPaggCustomerDto
        {
            Name = order.Client.Fullname,
            Email = order.Client.Email,
        };

        var payments = new List<MundiPaggPaymentsDto>();
        payments.Add(new MundiPaggPaymentsDto
        {
            payment_method = "credit_card",
            credit_card = new MundiPaggCrediCardDto
            {
                card = new MundiPaggCardDto
                {
                    holder_name = paymentDto.CardName,
                    number = paymentDto.CardNumber,
                    exp_month = Int32.Parse(paymentDto.ValidUntil.Substring(0, 2)),
                    exp_year = Int32.Parse(paymentDto.ValidUntil.Substring(3, 2)),
                    cvv = paymentDto.Cvv
                }
            }
        });
        var request = new
        {
            items = mundiPaggItems.items,
            customer = customer,
            payments = payments
        };

        HttpClient client = new HttpClient();
        client.BaseAddress = new Uri("https://api.mundipagg.com/core/");
        client.DefaultRequestHeaders.Accept.Clear();
        client.DefaultRequestHeaders.Accept.Add(
            new MediaTypeWithQualityHeaderValue("application/json"));
        client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic", "REDACTED");
        var requestJson = JsonConvert.SerializeObject(request);
        HttpResponseMessage response = await client.PostAsync(
            "v1/orders/", new StringContent(requestJson, Encoding.UTF8, "application/json"));

        var responseObject = JsonConvert.DeserializeObject<MundiPaggResponse>(response.Content.ReadAsStringAsync().Result);

        if(responseObject.status != "paid")
        {
            var error = responseObject.charges.First().last_transaction.acquirer_message;
            throw new Exception(error);
        }
        else
        {
            order.Status = Status.Payed;
            unitOfWork.Commit();
        }
    }

    return true;
}

```

Fonte: elaborado pelo autor

1  
Fonte courier.

2  
Frase longa.

3  
Número da figura.

4  
Fonte courier.

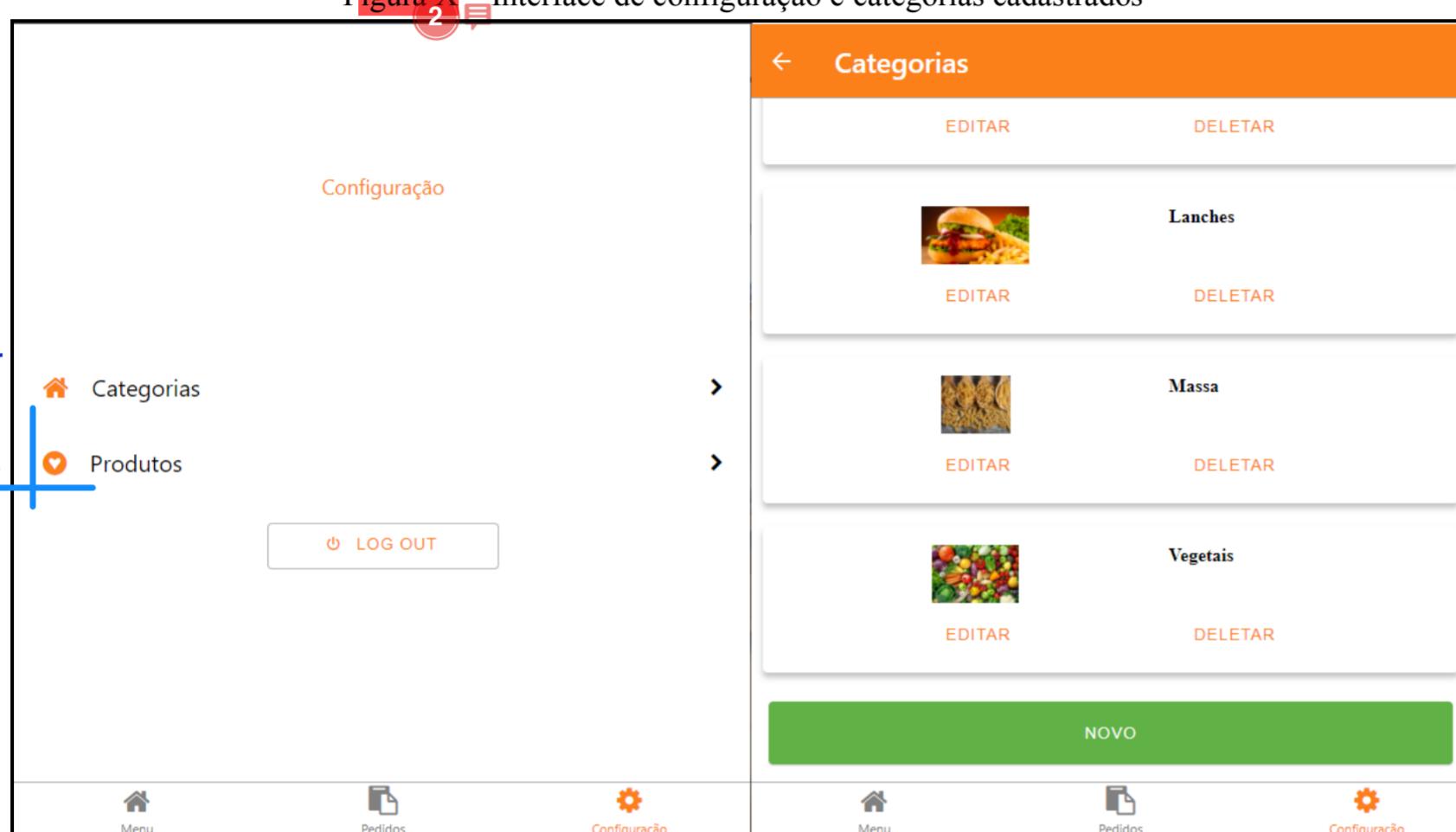
5  
Ponto final.

A figura  
deve ser  
citada e  
explicada  
no texto.

## APÊNDICE C – ORDEM DE INTERFACES PARA CADASTRO DAS CATEGORIAS

Ordem das interfaces para realizar o cadastro das categorias dos produtos, através do menu configurações do perfil de administrador.

Figura X – Interface de configuração e categorias cadastrados



Fonte: elaborado pelo autor  
Figura X – Interface de cadastro e edição da categoria

A interface de cadastro e edição da categoria é dividida em duas janelas. A janela esquerda, intitulada "Nova Categoria", contém campos para "Url" e "Nome", com um botão laranja "INSERIR" no final. A janela direita, intitulada "Atualizar Categoria", exibe os mesmos campos preenchidos com "https://blog.saipos.com/wp-content/uploads/2020/01/delivery-lanches-2.jpg" na url e "Lanches" no nome, com um botão laranja "ATUALIZAR" no final.

Fonte: elaborado pelo autor



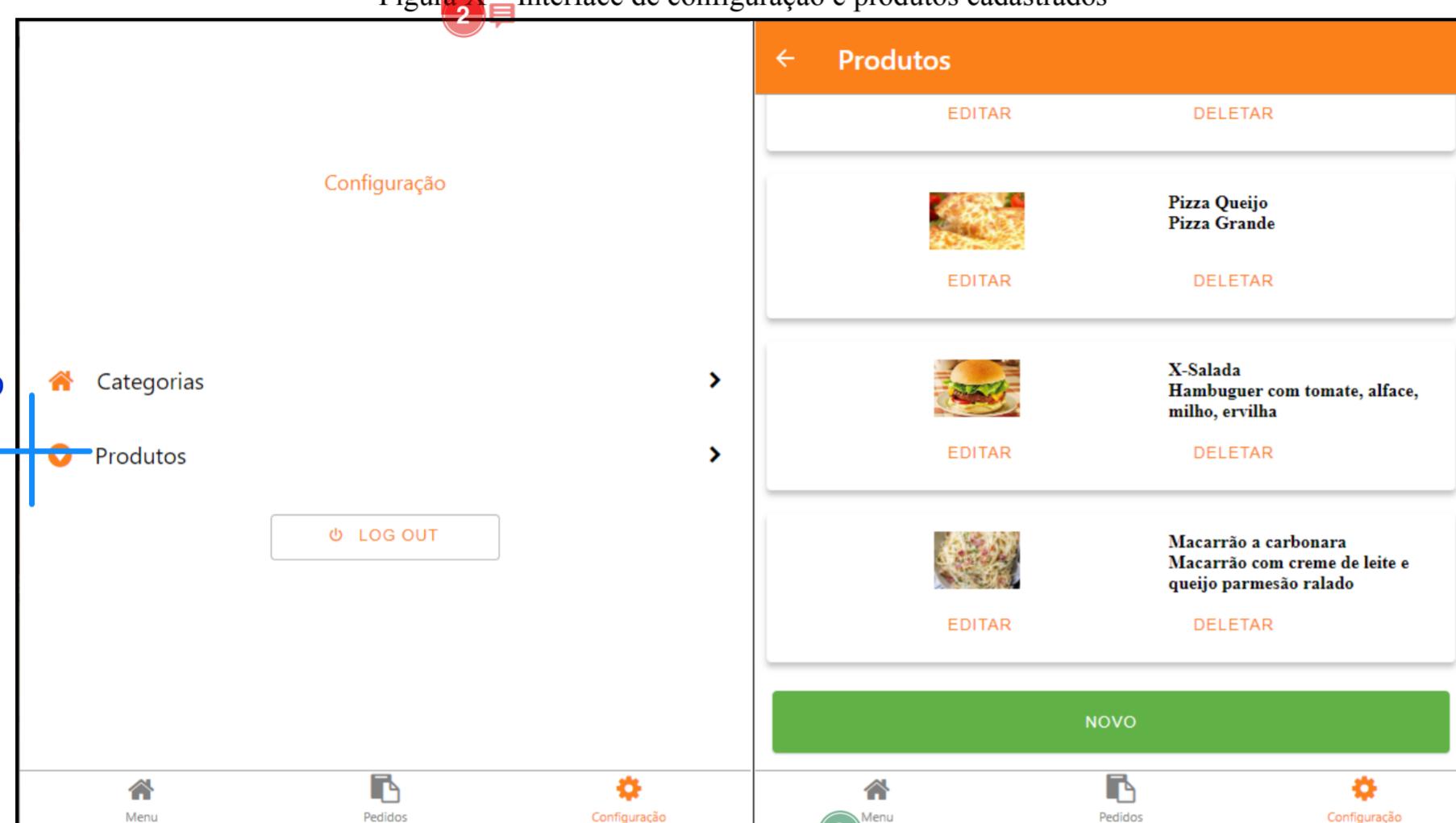
**A figura deve ser citada no texto.**

**A figura deve ser citada no texto.**

## APÊNDICE D – ORDEM DE INTERFACES PARA CADASTRO DOS PRODUTOS

Ordem das interfaces para realizar o cadastro dos produtos, através do menu configurações do perfil de administrador.

Figura X – Interface de configuração e produtos cadastrados



- 1 A ordem
- 2 Número.

Fonte: elaborado pelo autor

Figura X – Interface de cadastrar e editar produtos

A interface de cadastrar e editar produtos é dividida em duas seções: 'Novo Produto' (esquerda) e 'Atualizar Produto' (direita). Ambas as seções contêm campos para 'Nome', 'Descrição', 'Url Imagem' e 'Valor'. No lado esquerdo, há uma combobox com opções 'Pizza' e 'Lanches'. Um botão 'INserir' está na base da seção 'Novo Produto', e um botão 'ATUALIZAR' está na base da seção 'Atualizar Produto'. Os ícones 'Menu', 'Pedidos' e 'Configuração' estão no topo e na base de ambas as seções.

- 3 Ponto final.
- 4 Número.

Fonte: elaborado pelo autor

A figura deve ser citada no texto.

**REVISADO**

/Semestre: 2020/2

19

- 5 Ponto final.