

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Thiago Vasconcelos Braga

**Desenvolvimento do protótipo de uma API para  
integração dos pedidos de aplicativos de  
*delivery* e sistemas próprios em restaurantes**

**Uberlândia, Brasil**

**2023**

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Thiago Vasconcelos Braga

**Desenvolvimento do protótipo de uma API para  
integração dos pedidos de aplicativos de *delivery* e  
sistemas próprios em restaurantes**

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Sistemas de Informação.

Orientador: Paulo Henrique Ribeiro Gabriel

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2023



# UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Faculdade de Computação

Av. João Naves de Ávila, nº 2121, Bloco 1A - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902

Telefone: (34) 3239-4144 - <http://www.portal.facom.ufu.br/> facom@ufu.br



## ATA DE DEFESA - GRADUAÇÃO

Curso de Graduação em:	Bacharelado em Sistemas de Informação				
Defesa de:	Trabalho de Conclusão de Curso II				
Data:	04/12/2023	Hora de início:	09:35	Hora de encerramento:	10:30
Matrícula do Discente:	11921BSI225				
Nome do Discente:	Thiago Vasconcelos Braga				
Título do Trabalho:	Desenvolvimento do protótipo de uma API para integração dos pedidos de aplicativos de delivery e sistemas próprios em restaurantes				
A carga horária curricular foi cumprida integralmente?	<input checked="" type="checkbox"/> Sim <input type="checkbox"/> Não				

Reuniu-se remotamente por meio da Plataforma MS Teams, da Universidade Federal de Uberlândia, a Banca Examinadora, designada pelo Colegiado do Curso de Graduação em Bacharelado em Sistemas de Informação, assim composta: Professores: Dra. Maria Adriana Vidigal de Lima - FACOM/UFU; Dr. Rodrigo Sanches Miani - FACOM/UFU; e Dr. Paulo Henrique Ribeiro Gabriel - FACOM/UFU, orientador do candidato.

Iniciando os trabalhos, o presidente da mesa, Dr. Paulo Henrique Ribeiro Gabriel, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao discente a palavra, para a exposição do seu trabalho. A duração da apresentação do discente e o tempo de arguição e resposta foram conforme as normas do curso.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado Nota 90 (Somente números inteiros)

OU

Aprovado(a) sem nota.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.



Documento assinado eletronicamente por **Maria Adriana Vidigal de Lima, Professor(a) do Magistério Superior**, em 05/12/2023, às 09:37, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Rodrigo Sanches Miani, Professor(a) do Magistério Superior**, em 05/12/2023, às 11:10, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Paulo Henrique Ribeiro Gabriel, Professor(a) do Magistério Superior**, em 05/12/2023, às 13:36, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



A autenticidade deste documento pode ser conferida no site [https://www.sei.ufu.br/sei/controlador\\_externo.php?acao=documento\\_conferir&id\\_orgao\\_acesso\\_externo=0](https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0), informando o código verificador **5017669** e o código CRC **3B1483F1**.

# Resumo

Este trabalho de conclusão de curso tem como objetivo a utilização de processos de software para desenvolver um protótipo de uma aplicação *back-end* que disponibilize uma interface de programação (API, do inglês *Application Programming Interface*) para desenvolvedores, capaz de receber e gerenciar pedidos de restaurantes, sejam eles provenientes dos principais aplicativos de entrega disponíveis ou de sites e aplicações próprias do estabelecimento. Assim, é possível integrar todas essas informações em uma única ferramenta que pode ser consumida por outras aplicações. Os processos de software foram aplicados na etapa de desenvolvimento, os resultados obtidos foram um protótipo com algumas funcionalidades, e o levantamento das possíveis funcionalidades futuras.

**Palavras-chave:** API, Processos de software, Aplicativos.

# Lista de ilustrações

Figura 1 – Restaurante sem integração de pedidos . . . . .	8
Figura 2 – Restaurante com integração de pedidos . . . . .	9
Figura 3 – Arquitetura Limpa. Fonte: Martin (2020). . . . .	13
Figura 4 – Diagrama de Pacotes da arquitetura geral . . . . .	17
Figura 5 – Diagrama de Pacotes visão arquitetural da API . . . . .	18
Figura 6 – Diagrama de pacotes - Arquitetura do serviço de busca de pedidos externos . . . . .	19
Figura 7 – Diagrama de Classes da Aplicação . . . . .	21
Figura 8 – Diagrama ER do Banco de Dados. . . . .	23
Figura 9 – Caso de uso Cadastro de Estabelecimento . . . . .	24
Figura 10 – Caso de uso Cadastro de Aplicativo para o Estabelecimento . . . . .	25
Figura 11 – Casos de uso buscar detalhes do estabelecimento . . . . .	25
Figura 12 – Casos de uso abertura do estabelecimento . . . . .	26
Figura 13 – Diagrama de caso de uso cadastro de novo pedido . . . . .	27
Figura 14 – Diagrama de Caso de Uso busca pedidos do estabelecimento . . . . .	27
Figura 15 – Casos de uso busca pedidos externos . . . . .	28
Figura 16 – Diagrama de Caso de Uso cadastro de item . . . . .	28
Figura 17 – Diagrama de Caso de uso buscar itens do estabelecimento . . . . .	29
Figura 18 – Diagrama de Caso de uso alteração de dados do pedido . . . . .	30
Figura 19 – Arquitetura simplificada do serviço de busca de pedidos externos . . . . .	32
Figura 20 – Fluxograma de busca de pedidos externos . . . . .	34
Figura 21 – Chamada POST para cadastro de estabelecimento . . . . .	36
Figura 22 – Chamada POST para cadastro de aplicativo para o estabelecimento . . . . .	37
Figura 23 – Chamada GET para buscar informações do estabelecimento . . . . .	37
Figura 24 – Chamada POST para abertura do estabelecimento . . . . .	38
Figura 25 – Chamada POST para cadastro de pedido para o estabelecimento . . . . .	38
Figura 26 – Chamada Postman para busca de pedidos do estabelecimento . . . . .	39
Figura 27 – Chamada POST para cadastro de item em um estabelecimento . . . . .	39
Figura 28 – Chamada PUT para alteração de pedido do estabelecimento . . . . .	40
Figura 29 – geração de pedido Externo pelo aplicativo Ifood . . . . .	41
Figura 30 – Pedido externo do aplicativo Ifood cadastrado no banco de dados . . . . .	41
Figura 31 – Chamada GET para buscar os itens cadastrados por um estabelecimento . . . . .	42

# Lista de abreviaturas e siglas

API	<i>Application Programming Interface</i> (Interface de Programação de Aplicação)
IoT	<i>Internet of Things</i> (Internet das Coisas)
REST	<i>Representational State Transfer</i> (Transferência de Estado Representacional)
HTTP	<i>Hypertext Transfer Protocol</i> (Protocolo de Transferência de Hipertexto)

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>8</b>
<b>2</b>	<b>REFERENCIAL TEÓRICO</b>	<b>11</b>
<b>2.1</b>	<b>Processo de Software</b>	<b>11</b>
<b>2.2</b>	<b>Arquitetura de Software e Código Limpo</b>	<b>12</b>
2.2.1	Arquitetura Limpa	13
2.2.2	Serviços Web e API's	14
<b>2.3</b>	<b>Trabalhos Correlatos</b>	<b>15</b>
<b>3</b>	<b>DESENVOLVIMENTO</b>	<b>16</b>
<b>3.1</b>	<b>Planejamento, Análise de Requisitos e Modelagem</b>	<b>16</b>
3.1.1	Análise de Requisitos e Modelagem	16
<b>3.2</b>	<b>Preparação do ambiente de desenvolvimento e codificação</b>	<b>29</b>
3.2.1	Preparação do ambiente de desenvolvimento	29
3.2.2	Codificação	30
<b>4</b>	<b>AVALIAÇÃO DO PROTÓTIPO</b>	<b>36</b>
<b>4.1</b>	<b>Cadastro de Estabelecimentos</b>	<b>36</b>
<b>4.2</b>	<b>Cadastro de Aplicativo para o Estabelecimento</b>	<b>36</b>
<b>4.3</b>	<b>Busca informações do Estabelecimento</b>	<b>37</b>
<b>4.4</b>	<b>Abertura do Estabelecimento</b>	<b>37</b>
<b>4.5</b>	<b>Cadastrar Pedido para o Estabelecimento</b>	<b>37</b>
<b>4.6</b>	<b>Buscar Pedidos do Estabelecimento</b>	<b>38</b>
<b>4.7</b>	<b>Cadastrar Item para o Estabelecimento</b>	<b>39</b>
<b>4.8</b>	<b>Alterar Pedido para o Estabelecimento</b>	<b>40</b>
<b>4.9</b>	<b>Buscar Pedidos Externos para o Estabelecimento</b>	<b>40</b>
<b>4.10</b>	<b>Buscar Itens do Estabelecimento</b>	<b>42</b>
<b>5</b>	<b>CONCLUSÃO</b>	<b>43</b>
	<b>REFERÊNCIAS</b>	<b>45</b>



# 1 Introdução

A Internet e a tecnologia de informação mudaram a forma como as pessoas compram, vendem e se relacionam. Uma das inovações trazidas por essa revolução tecnológica é a maneira de pedir alimentos em restaurantes e demais estabelecimentos alimentícios. Nesse contexto, os aplicativos de *food-delivery* se popularizaram (JUNIOR, 2021): um pedido que antes exigiria alguns minutos de ligação telefônica e pagamento para o entregador em dinheiro vivo, pode ser feito, atualmente, com alguns cliques e pagamento online em segundos.

Além de benefícios para os consumidores, os estabelecimentos conseguem atender um número maior de requisições e ampliar seus canais de vendas (BONFANTI, 2020; JUNIOR, 2021). Porém, para esses estabelecimentos, surgiram desafios inerentes à nova tecnologia. Entre esses problemas, destaca-se a dificuldade de gerenciar pedidos que são recebidos de diferentes fontes, visto que existem diversos aplicativos disponíveis no mercado, além de sistemas de venda próprios. O processo de intercalar o gerenciamento entre as diversas origens de pedidos torna complexo, e até mesmo inviável, trabalhar em larga escala, além de dificultar o levantamento de informações gerenciais e estatísticas importantes para o sucesso do negócio. (BONFANTI, 2020). Para ilustrar o problema, pode-se observar na FIGURA 1 o exemplo de um restaurante que gerencia os pedidos de diversas fontes sem nenhum tipo de integração, para trabalhar conforme ilustrado na figura, são necessários muitos recursos e o acompanhamento de cada painel de administração dos aplicativos que o estabelecimento trabalha.

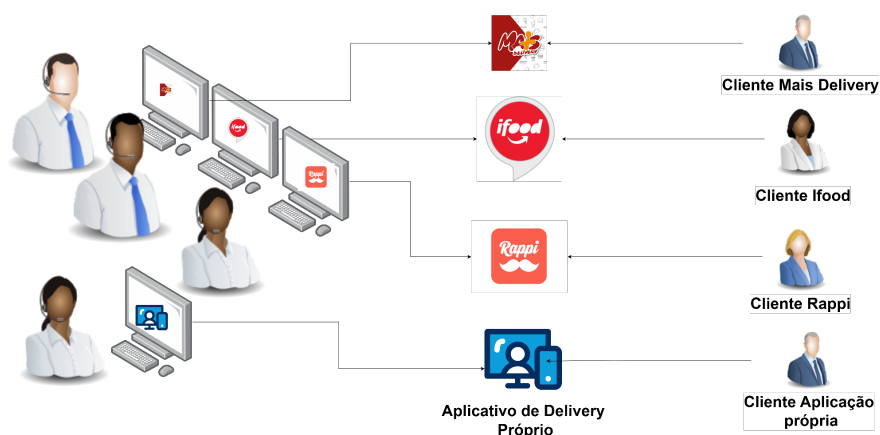


Figura 1 – Restaurante sem integração de pedidos

Para tratar esse problema, é necessário buscar formas de integrar o gerenciamento de pedidos de diversos aplicativos e das ferramentas próprias dos estabelecimentos em “um único lugar”. Assim, é importante o desenvolvimento de uma ferramenta única capaz de

receber solicitações de várias aplicações, gerenciar fluxo de caixa e de clientes e produzir relatórios com informações administrativas do negócio de maneira integrada (BONFANTI, 2020). Abaixo, na FIGURA 2 pode-se visualizar um exemplo de restaurante que trabalha utilizando uma aplicação integradora, que é capaz de reduzir de maneira significativa os recursos necessários para gerenciar os pedidos das diversas origens, pois dessa maneira teremos todas as informações devidamente integradas em um único sistema.

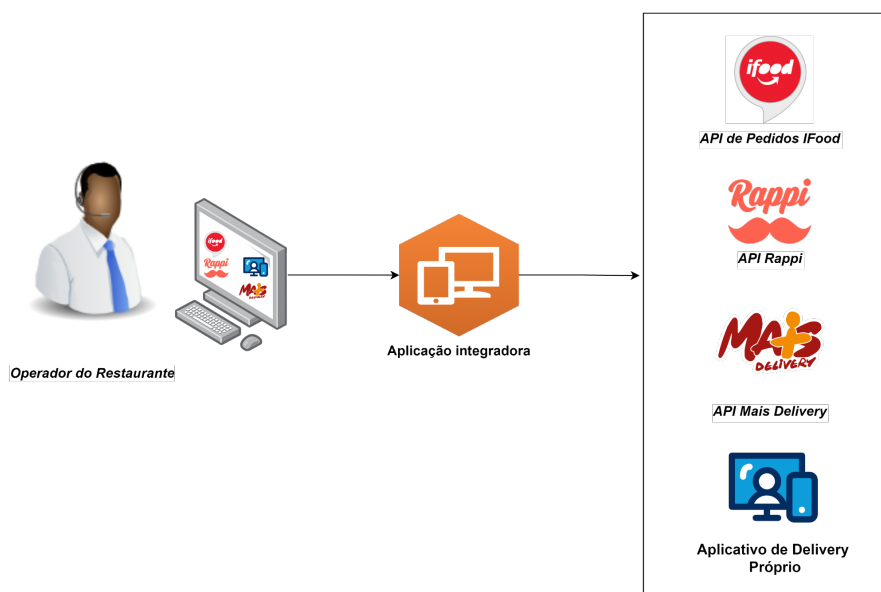


Figura 2 – Restaurante com integração de pedidos

Diante do exposto, esse trabalho de conclusão de curso tem como objetivo a utilização de processos de software para desenvolver um protótipo de uma aplicação *back-end* que disponibilize uma interface de programação (API, do inglês *Application Programming Interface*) para desenvolvedores, capaz de receber e gerenciar pedidos de restaurantes, sejam eles provenientes dos principais aplicativos de entrega disponíveis ou de sites e aplicações próprias do estabelecimento. Assim, é possível integrar todas essas informações em uma única ferramenta que pode ser consumida por aplicações web e móveis, ou de internet das coisas (IoT, do inglês *Internet of Things*).

Com a alta demanda e os diversos meios de venda disponíveis, restaurantes tendem a consumir muitos recursos para o gerenciamento de todos os canais onde realizam vendas. Por isso, uma ferramenta capaz de fazer a abertura e o fechamento do estabelecimento nos aplicativos, receber pedidos de todos eles, orquestrar essa dinâmica e gerar informações relevantes do negócio de maneira integrada é de grande valia para economia de capital. Além da economia proporcionada por essa ferramenta, será possível colaborar com outros desenvolvedores que criem sistemas de integração de pedidos, fazendo o reaproveitamento de códigos e implementando de fato apenas o *front-end* de suas aplicações.

Para atingir os objetivos, foi utilizada a linguagem de programação *C#* com o

---

kit de desenvolvimento *.NET* e Banco de dados *Sql Server*. Com isso, foi desenvolvido o protótipo de uma API que atende aos requisitos estabelecidos. O restante desta monografia é organizado da seguinte maneira: no Capítulo 2 são apresentados os principais conceitos e processos empregados no desenvolvimento do protótipo. A etapa de desenvolvimento é detalhada no Capítulo 3 e a avaliação do protótipo é descrita no Capítulo 4. Finalmente, no Capítulo 5, são levantadas as possíveis funcionalidades futuras, bem como uma analogia com os conteúdos estudados durante o curso de Bacharelado em Sistemas de Informação.

## 2 Referencial Teórico

Neste capítulo, são descritos conceitos necessários para o correto entendimento dos processos e métodos utilizados ao decorrer desta monografia. Além disso, são apresentados trabalhos relacionados à proposta deste trabalho.

### 2.1 Processo de Software

Um dos principais pilares deste trabalho de conclusão de curso é o processo de desenvolvimento de software, que, segundo [Pressman e Maxim \(2016\)](#), é uma metodologia para as atividades, ações e tarefas necessárias para desenvolver um software de alta qualidade. As etapas da metodologia variam de acordo com as especificidades do projeto. [Pressman e Maxim \(2016\)](#) afirmam que, o processo adotado depende do software a ser desenvolvido e citam, como exemplo, que um determinado processo pode ser apropriado para o sistema de uma aeronave, enquanto um outro totalmente diferente pode ser indicado para a criação de um site. Apesar de cada caso exigir definições específicas, os autores definem cinco etapas de um processo de software genérico.

1. **Comunicação:** Consiste em compreender os objetivos das partes interessadas para com o projeto e fazer o levantamento das necessidades que ajudarão a definir as funções e características do software.
2. **Planejamento:** A atividade de planejamento cria um “mapa” que ajuda a guiar a equipe na sua jornada. O mapa, denominado plano de projeto de software, define o trabalho de engenharia de software, descrevendo as tarefas técnicas a serem conduzidas, os riscos prováveis, os recursos que serão necessários, os produtos resultantes a ser produzidos e um cronograma de trabalho.
3. **Modelagem:** Trata-se da criação de modelos para melhor entender as necessidades do software e o projeto que irá atender a essas necessidades.
4. **Construção:** Essa atividade combina geração de código (manual ou automatizada) e testes necessários para revelar erros na codificação.
5. **Entrega:** O software (como uma entidade completa ou como um incremento parcialmente efetivado) é entregue ao cliente, que avalia o produto entregue e fornece **feedback**, baseado na avaliação.

Como o processo deve ser adaptado de acordo com as necessidades do projeto, as etapas deste trabalho, citadas na introdução, serão baseadas na definição genérica de

Pressman e Maxim (2016), as fases de Planejamento e Modelagem, serão englobadas na etapa de Análise de Requisitos e Modelagem, enquanto a fase de Construção será tratada na etapa de Codificação e Preparação do Ambiente de Desenvolvimento.

## 2.2 Arquitetura de Software e Código Limpo

Arquitetura de software pode ser definida como a organização fundamental de um sistema, seus componentes, as relações entre eles e o ambiente que guia seu design e evolução (RICHARDS; FORD, 2020). Segundo Martin (2020), uma boa arquitetura tem como objetivo principal minimizar os recursos humanos necessários para construir e manter um projeto. Em sua obra, ele descreve diversas práticas de Arquitetura que tornam o sistema de fácil manutenção, impactando diretamente nos recursos humanos e financeiros demandados.

Um dos componentes de uma boa arquitetura de software são os chamados códigos limpos. Martin (2009) descreve a definição de alguns autores importantes sobre esse termo, dentre eles Grady Booch, autor do livro *Object Oriented Analysis and Design with Applications*, que cita que “o código limpo jamais torna confuso o objetivo do desenvolvedor, está repleto de abstrações claras e linhas de controle objetivas”. Para definir Código Limpo, Martin (2020) cita cinco princípios de projeto de software que formam um importante passo para se escrever um bom código, cujas definições serão explicitadas nos itens seguintes, com algumas adaptações.

1. Princípio da Responsabilidade Única (*Single Responsibility Principle*):  
Os objetivos de cada módulo do sistema devem ser definidos para que não haja partes com responsabilidades demais, dessa maneira sempre que for necessário uma alteração, os programadores saberão exatamente qual parte do sistema é responsável pelo que será necessário alterar, caso um módulo tenha muitas responsabilidades essa identificação se torna mais difícil.
2. Princípio Aberto Fechado (*Open Closed Principle*):  
Um artefato de software deve ser aberto para extensão, mas fechado para modificação. O módulos do sistema devem estar abertos para ganhar novas funcionalidades e fechados para alterações em funcionalidades já definidas. Isso implica que uma boa arquitetura de software deve reduzir a quantidade de código a ser mudado para o mínimo possível.
3. Princípio da Substituição de Liskov (*Liskov Substitution Principle*):  
O Princípio da Suibstituição de Liskov é aplicado quando o comportamento de uma aplicação não depende, de maneira alguma, da utilização de qualquer dos subtipos de um tipo, ou seja, quaisquer subtipos de uma classe mãe são substituíveis por ela.

4. Princípio da Segregação de Interface (*Interface Segregation Principle*):  
Esse princípio define que os objetos implementados não devem ser forçados a depender de uma mesma interface que um deles não utiliza. Quando esse princípio é aplicado, a classe e suas dependências se comunicam usando interfaces segregadas.
5. Princípio da Inversão de Dependências (*Dependence Inversion Principle*):  
Os sistemas mais flexíveis são aqueles em que as dependências de código-fonte se referem apenas a abstrações e não a itens concretos. As declarações *use*, *import* e *include* devem se referir apenas a módulos-fonte que contenham interfaces, classes abstratas ou outro tipo de declaração abstrata, ou seja, não devem depender de nada que seja concreto.

### 2.2.1 Arquitetura Limpa

Conforme escrito por [Martin \(2020\)](#), existe uma série de ideias de diferentes autores relacionadas à arquitetura de sistemas, apesar de suas diferenças nos detalhes, todas têm o objetivo de separar as obrigações, o que é feito por meio da divisão em camadas. Considerando essa necessidade de separar responsabilidades, ele definiu uma arquitetura que engloba diversos conceitos de design e chamou de Arquitetura Limpa. Essa arquitetura descreve quatro camadas de software, conforme ilustrado na FIGURA 3.

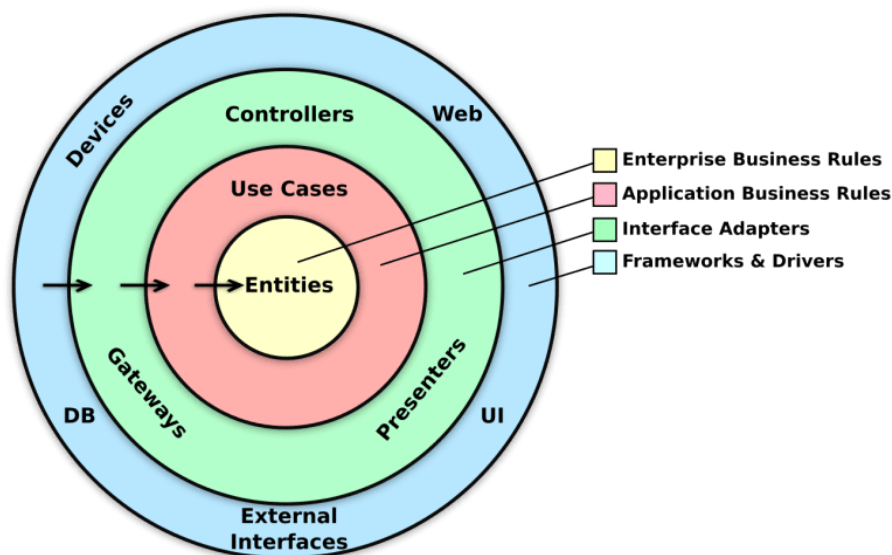


Figura 3 – Arquitetura Limpa. Fonte: [Martin \(2020\)](#).

Quanto mais interna a camada, maior é o nível do código em termos de negócios. Assim, na FIGURA 3, os círculos mais externos são mecanismos, os mais internos são

políticas ou regras de negócios. A seguir, essas camadas são descritas brevemente, segundo as definições de [Martin \(2020\)](#):

1. Camada de Entidades:

As Entidades reúnem as Regras Cruciais de Negócios da empresa inteira. Uma entidade pode ser um objeto com métodos ou um conjunto de estruturas de dados e funções.

2. Camada de Casos de Uso:

O software da camada de casos de uso contém as regras de negócio específicas da aplicação. Ele reúne e implementa todos os casos de uso do sistema. Esses casos de uso orquestram o fluxo de dados para e a partir das entidades e orientam essas entidades na aplicação das Regras Cruciais de Negócios a fim de atingir os objetivos do caso de uso.

3. Camada de Controladores ou Adaptadores de Interface:

O software da camada de adaptadores de interface consiste em um conjunto de adaptadores que convertem dados no formato que é mais conveniente para os casos de uso. Os apresentadores, visualizações e controladores pertencem à camada de adaptadores de interface. Os modelos são apenas estruturas de dados transmitidas dos controladores para os casos de uso e, então, dos casos de uso para os controladores.

4. Camada de Interfaces Externas:

A camada mais externa do modelo mostrado na FIGURA 3 é geralmente composta de *frameworks* e ferramentas como a base de dados e o *framework* web. Em geral, você não programa muita coisa nessa camada além do código de associação que estabelece uma comunicação com o círculo interno seguinte.

### 2.2.2 Serviços Web e API's

Um serviço web é qualquer serviço disponível na rede, que utiliza mensagens para comunicação e não se importa com qual sistema operacional ou linguagem está sendo utilizada. ([CERAMI, 2009](#))

API é um tipo de web service com um conjunto de definições e protocolos usados no desenvolvimento e na integração de aplicações. Enquanto API's REST (Application Programming Interface Representational State Transfer) são API's que seguem um conjunto de regras e convenções para construir e interagir com aplicações que expõem recursos e operações através de uma interface baseada em protocolo HTTP. ([REDHAT, 2023](#)).

## 2.3 Trabalhos Correlatos

Como trabalhos correlatos, existem alguns softwares no mercado que fazem a integração de pedidos, porém, não disponibilizam API's públicas para integrar quaisquer aplicações de *front-end* de controle de pedidos. O primeiro deles, é o Sistema de Gestão KCMS<sup>1</sup>, que é um sistema com *front-end* e *back-end* que permite integrar todas as áreas de um restaurante, inclusive o recebimento de pedidos por meio de diversos aplicativos de *delivery*. Esse, no entanto, é uma sistema fechado, que não disponibiliza uma API de integração para que outros desenvolvedores consumam.

Outro trabalho relevante é o Sistema de Integração de pedidos SAIPOS<sup>2</sup>, que é capaz de integrar pedidos de alguns aplicativos de *delivery*, contando também com um *front-end* e *back-end*. Porém, também é uma aplicação fechada e não disponibiliza API para utilização com demais sistemas de *front-end*, portanto o restaurante ou o desenvolvedor depende de uma sistema proprietário.

---

<sup>1</sup> <<https://www.kcms.com.br/>>

<sup>2</sup> <<https://saipos.com/sistema-para-restaurante/central-de-pedidos-para-restaurant>>



## 3 Desenvolvimento

O desenvolvimento deste trabalho foi organizado em etapas, sendo que a primeira englobou as atividades de planejamento, análise de Requisitos e modelagem. Já na segunda etapa, foi realizada a preparação do ambiente de desenvolvimento e codificação. Nas seções seguintes, discorre-se sobre a execução das etapas definidas, bem como sobre as experiências e resultados.

### 3.1 Planejamento, Análise de Requisitos e Modelagem

A execução se iniciou com o planejamento das etapas que envolviam o processo de construção de software, que foi essencial para estabelecer as metas a serem atingidas nas etapas seguintes e o prazo para concluir cada uma delas, bem como, recursos necessários, tempo de desenvolvimento, recursos externos, dentre outras necessidades importantes.

#### 3.1.1 Análise de Requisitos e Modelagem

Logo em seguida ao planejamento, foram iniciados os trabalhos de Análise de Requisitos e Modelagem das estruturas de software necessárias. O primeiro passo desta etapa foi o levantamento de requisitos, que resultou na definição dos casos de uso da aplicação, que são relatados com mais detalhes nas subseções seguintes.

Definidos os casos de uso, o próximo passo dado foi desenhar o modelo arquitetural dos componentes que integram o sistema, seguindo os padrões de arquitetura limpa. A seguir, são descritos os diagramas e desenhos arquiteturais obtidos por meio da análise dos requisitos e dos casos de uso definidos.

#### Diagrama de Arquitetura Geral

Antes de construir os diagramas relacionados às regras de negócio, foi desenvolvido um diagrama de pacotes que representa a arquitetura de software geral, que foi utilizado como referência para construção da aplicação dentro dos padrões de arquiteturas definidos. O resultado pode ser visualizado abaixo na FIGURA 4.

#### Diagrama Arquitetural da API

Adentrando mais especificamente na arquitetura da API, que é o primeiro pacote mostrado no diagrama da FIGURA 4, pode-se visualizar que essa parte da solução conta com 4 camadas, a Camada de Controladores, que é o ponto de entrada das requisições externas, a Camada de Casos de Uso, onde as requisições são direcionadas de acordo com

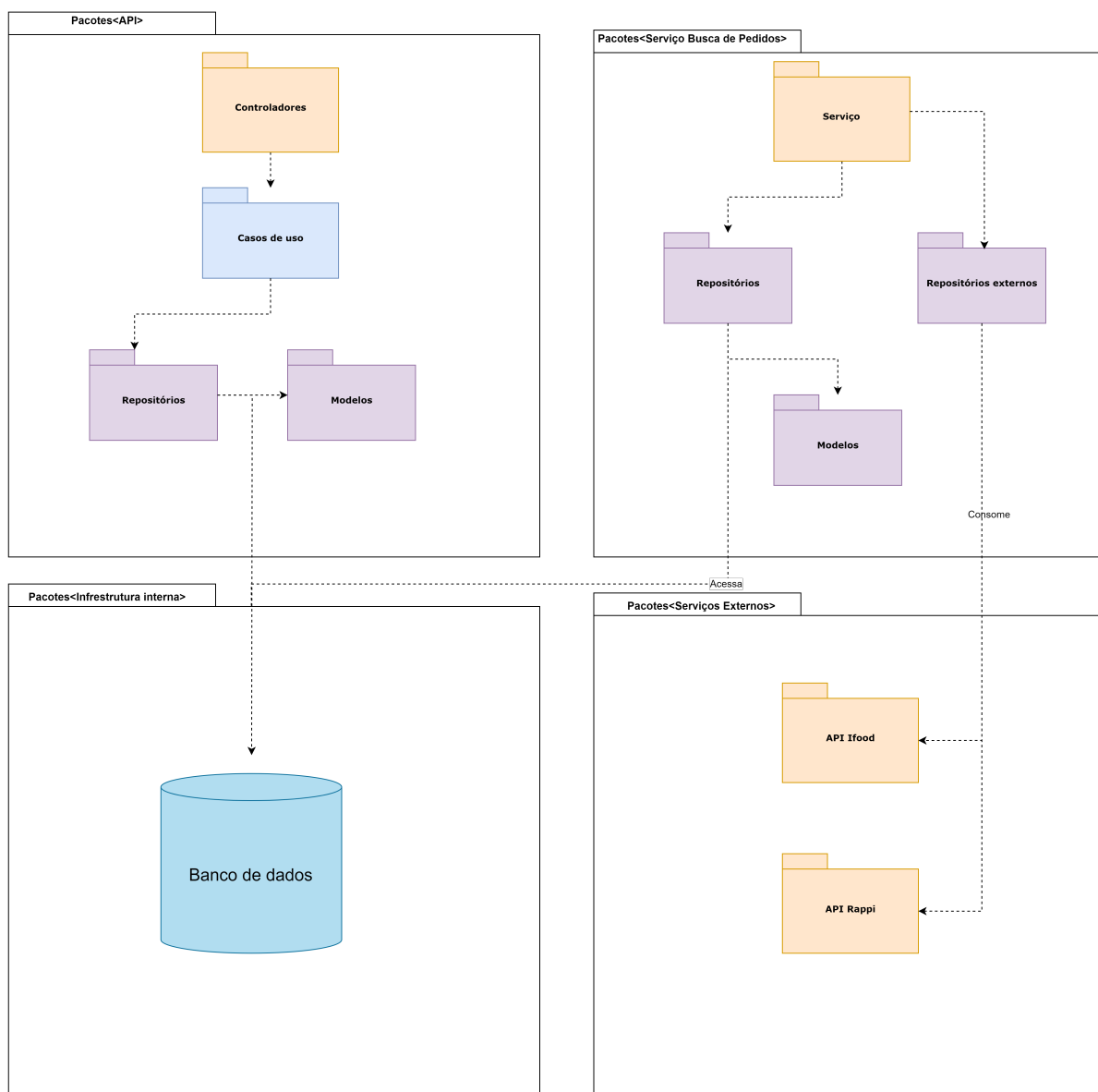


Figura 4 – Diagrama de Pacotes da arquitetura geral

cada caso de uso, a Camada de Modelos, na qual os tipos e entidades são definidos e a camada de Repositórios, a qual tem a responsabilidade de acessar os modelos e o banco de dados. O pacote referente à API poder visualizado detalhadamente abaixo na FIGURA 5.

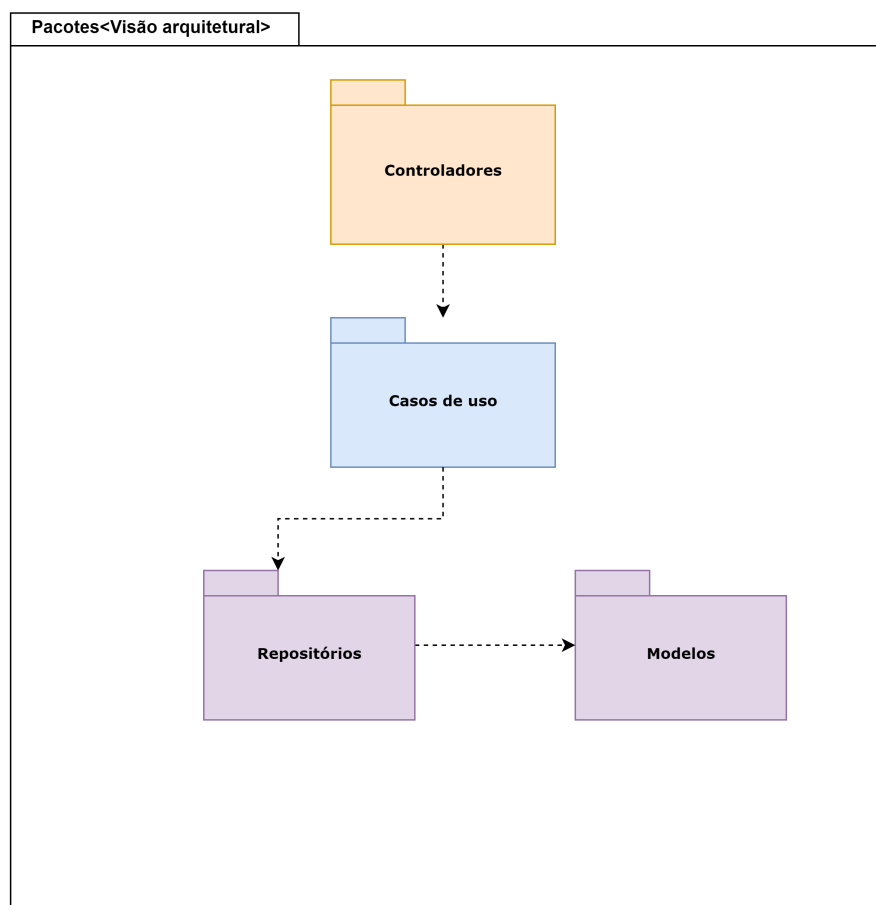


Figura 5 – Diagrama de Pacotes visão arquitetural da API

### Diagrama Arquitetural do Serviço de Pedidos Externos

Para buscar os pedidos externos dos estabelecimentos cadastrados, foi desenvolvido um serviço que consome as API's REST dos aplicativos externos, esse serviço segue a arquitetura disposta na FIGURA 6.

### Diagramas de Classes de Domínio

As classes de domínio do projeto representam as entidades do mundo real que fazem parte do software e são abstraídas por meio de classes, para representá-las foi desenvolvido o diagrama de classes que pode ser visualizar abaixo na FIGURA 7. Lembrando que, como as classes de domínio tem os mesmos nomes das tabelas de banco de dados, na etapa de

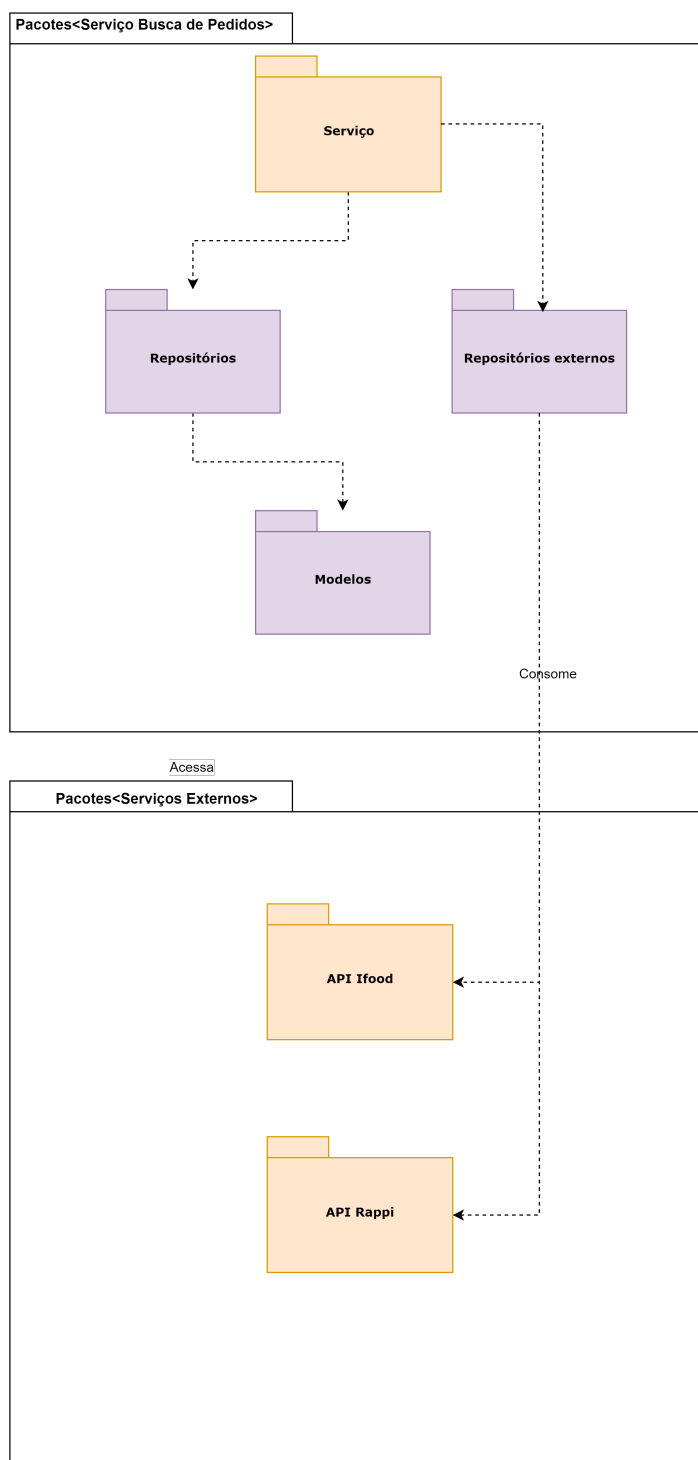


Figura 6 – Diagrama de pacotes - Arquitetura do serviço de busca de pedidos externos

Desenvolvimento será explanado com detalhes o nome de cada tabela e a entidade que a representa no diagrama de classes.

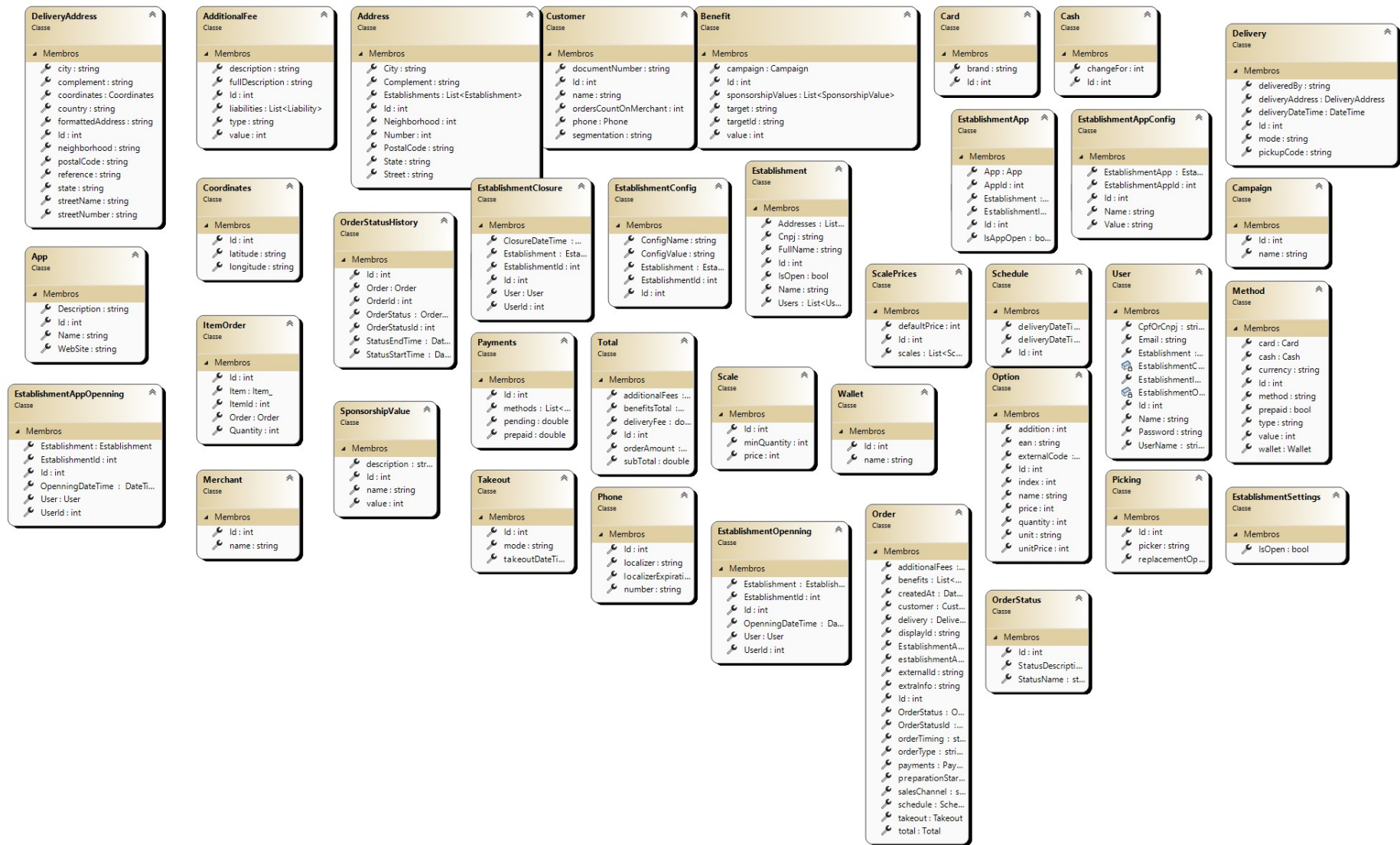


Figura 7 – Diagrama de Classes da Aplicação

## Diagrama de Banco de Dados

Foi necessária a utilização de uma boa tecnologia e arquitetura de armazenamento de dados, levando em consideração essa premissa, foi desenvolvido o diagrama entidade relacionamento exposto na FIGURA 8, no qual podemos visualizar as tabelas utilizadas pela aplicação e o relacionamento definido entre elas para que atenda às necessidades das regras de negócio definidas. Na Seção 3.2, será relatado detalhadamente o significado e os dados que cada uma das tabelas armazena.

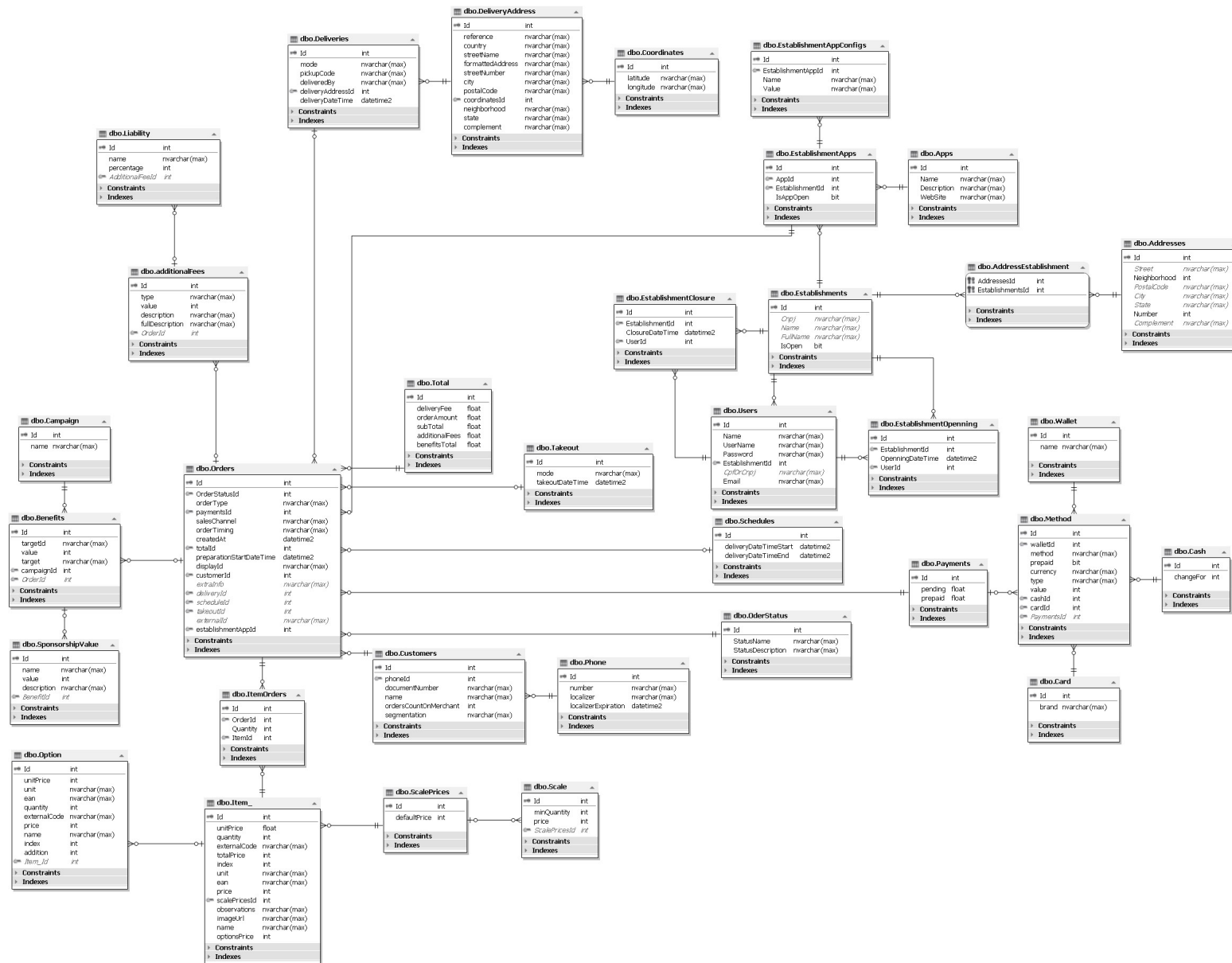


Figura 8 – Diagrama ER do Banco de Dados.



## Diagramas de Casos de Uso

Nesta seção apresentaremos os diagramas de casos de uso desenvolvidos para a aplicação, que são resultado do processo de Modelagem e serão utilizados como referência para o desenvolvimento do restante do Processo de Software.

**Cadastro de Estabelecimento.** Este caso de uso trata-se do primeiro contato para que um estabelecimento utilize a API, nele é realizado um cadastro por meio de um *end-point*. Todo estabelecimento deve ter pelo menos um usuário, portanto nesse momento também ocorre o cadastro do primeiro usuário do novo estabelecimento. O Diagrama de Caso de Uso deste tópico pode ser visualizado abaixo na FIGURA 9.

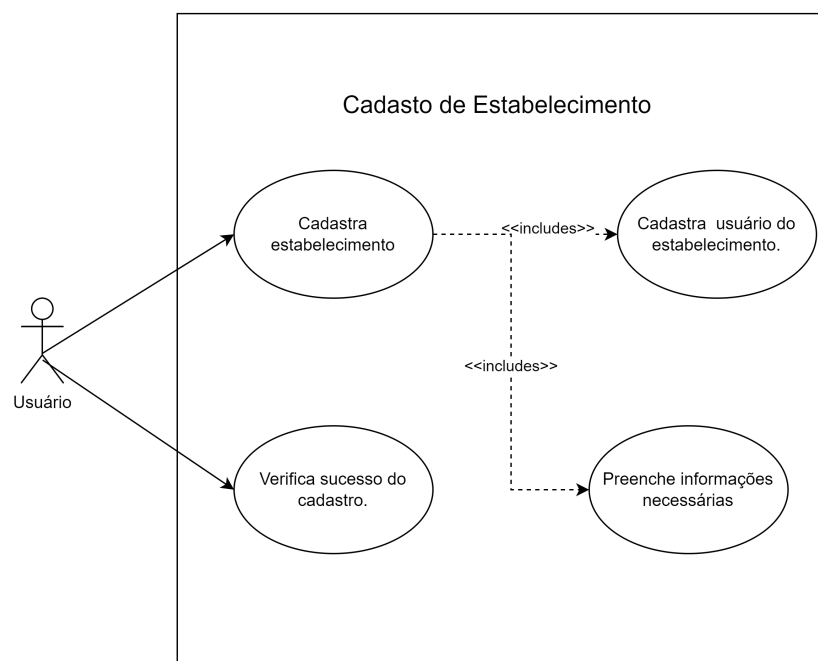


Figura 9 – Diagrama de caso de uso Cadastro de Estabelecimento

**Cadastro de Aplicativo para o Estabelecimento.** Neste caso de uso, o estabelecimento cadastra um aplicativo externo para que possa integrar os pedidos no banco de dados da API, ou seja, o primeiro passo para integração de pedido externos é o cadastro de um novo aplicativo. O Diagrama de Caso de Uso pode ser visualizado abaixo na FIGURA 10.

**Busca Informações do Estabelecimento.** Trata-se de um *end-point* disponibilizado para busca de detalhes sobre o estabelecimento que está logado no momento, como nome, endereço, entre outros dados. O Diagrama de Caso de Uso referente a esta funcionalidade pode ser visualizado na FIGURA 11.

**Abertura do Estabelecimento.** A abertura significa que o estabelecimento está apto para receber pedidos internos e externos naquele dia, os estabelecimentos são abertos

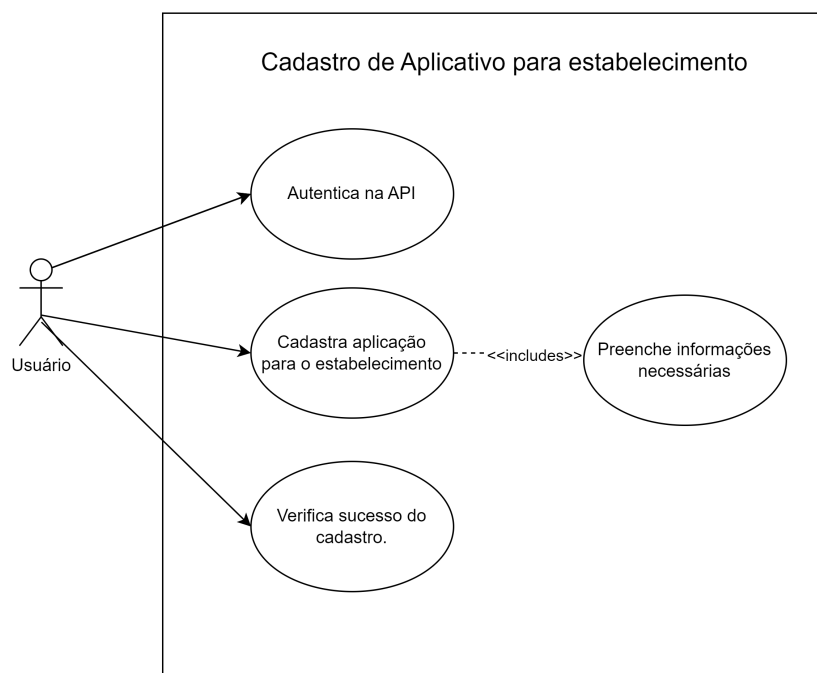


Figura 10 – Diagrama de caso de uso Cadastro de Aplicativo para Estabelecimento

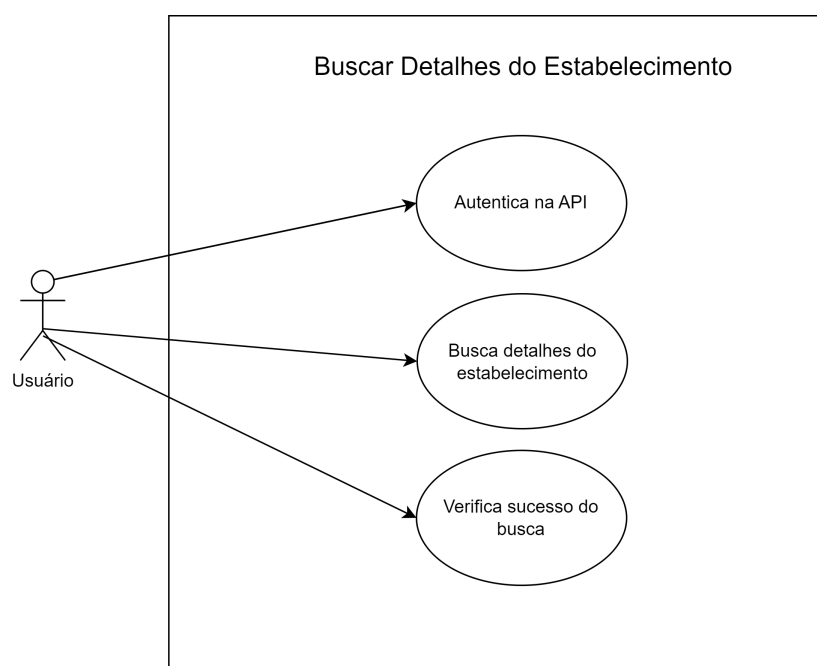


Figura 11 – Diagrama de caso de uso buscar detalhes do estabelecimento

por meio de uma chamada ao *end-point* de abertura. O Diagrama de Caso de Uso desta funcionalidade pode ser visualizado abaixo na FIGURA 12.

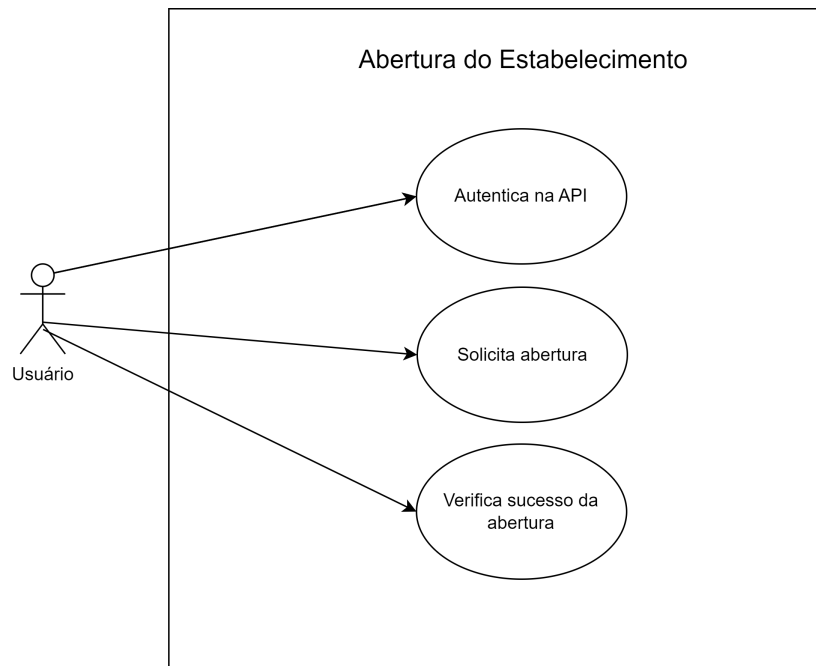


Figura 12 – Diagrama de caso de uso abertura do estabelecimento

**Cadastrar Pedido para o Estabelecimento.** Representa a funcionalidade de cadastro de pedidos por meio de aplicações próprias do estabelecimento através da API REST. O Diagrama de Caso de Uso que representa esta funcionalidade pode ser visualizado na FIGURA 13.

**Buscar Pedidos do Estabelecimento.** Busca no banco de dados da aplicação, todos os pedidos do estabelecimento com filtros desejados. O Diagrama de Caso de Uso pode ser visualizado abaixo na FIGURA 14.

**Buscar Pedidos Externos do Estabelecimento.** Trata-se do serviço que realiza chamadas nas APIS externas para encontrar os pedidos nas aplicações terceiras e inserí-los no banco de dados. Esse caso de uso envolve apenas serviços de sistema e serviços externos. O Diagrama de Caso de Uso desta funcionalidade pode ser visualizado abaixo na FIGURA 15.

**Cadastrar Item para o Estabelecimento.** Este caso de uso representa o cadastro de um novo item para o estabelecimento. O Diagrama de Caso de Uso pode ser visualizado abaixo, na FIGURA 16.

**Buscar Itens de um estabelecimento.** Retorna todos os itens cadastrados para um estabelecimento. O Diagrama de Caso de Uso pode ser visualizado abaixo, na Figura 17.

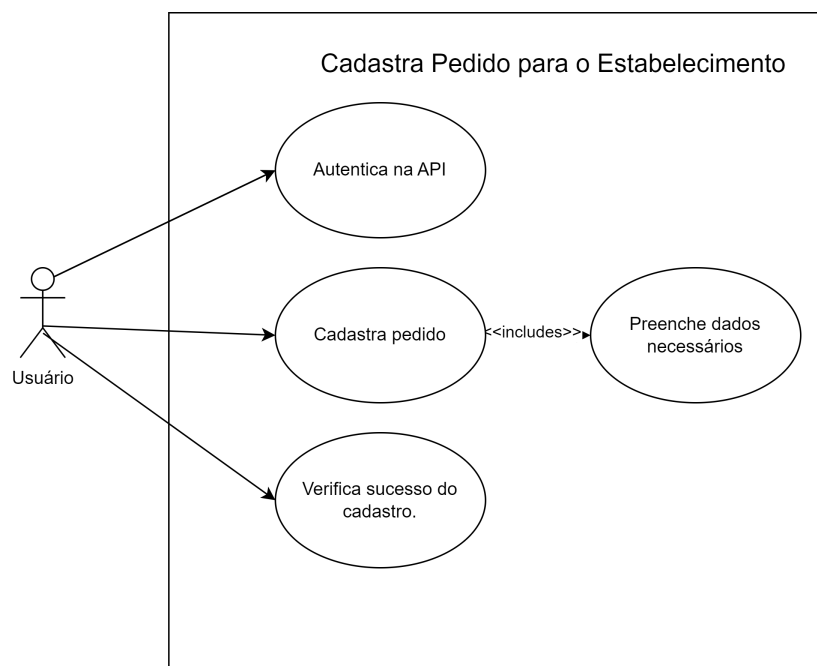


Figura 13 – Diagrama de caso de uso cadastro de novo pedido

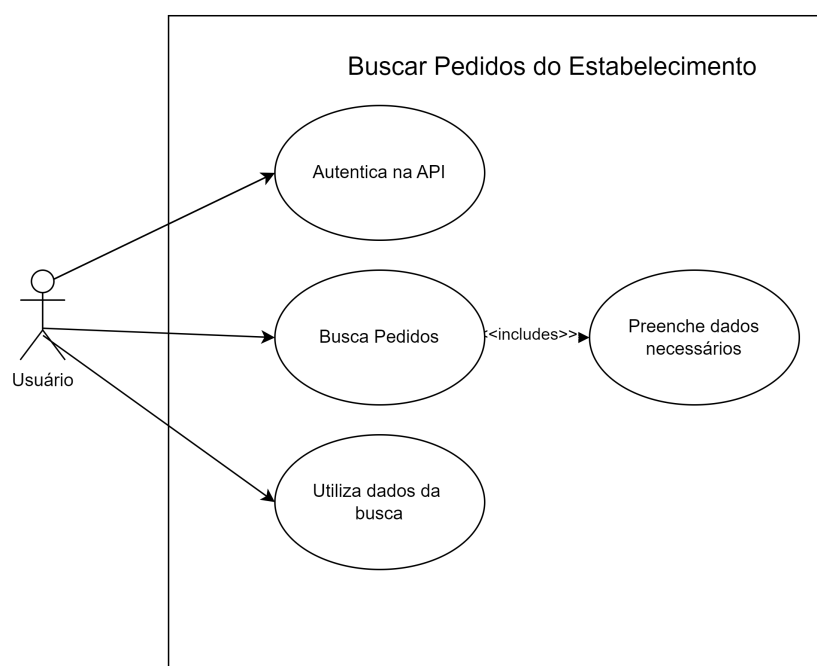


Figura 14 – Diagrama de Caso de Uso busca pedidos do estabelecimento

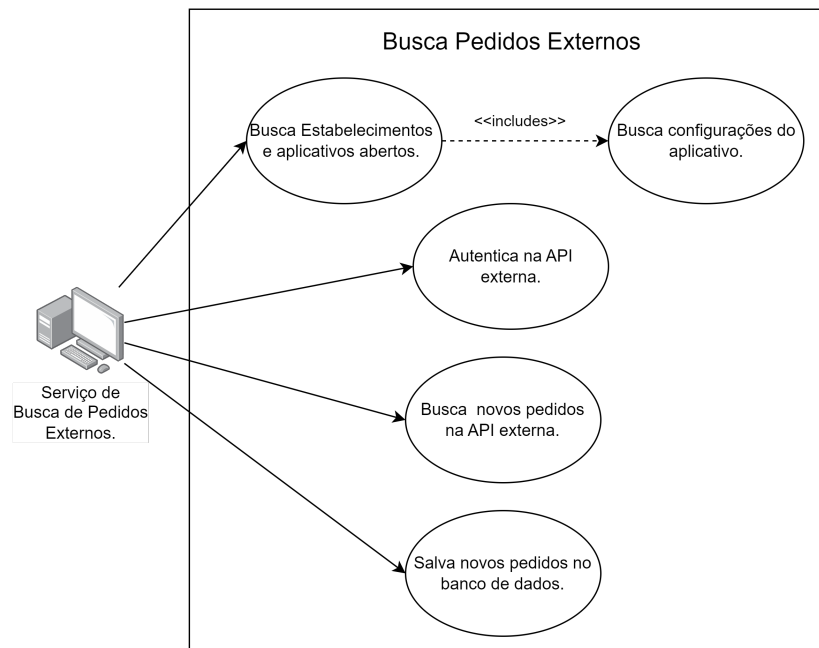


Figura 15 – Diagrama de caso de uso busca pedidos externos do estabelecimento

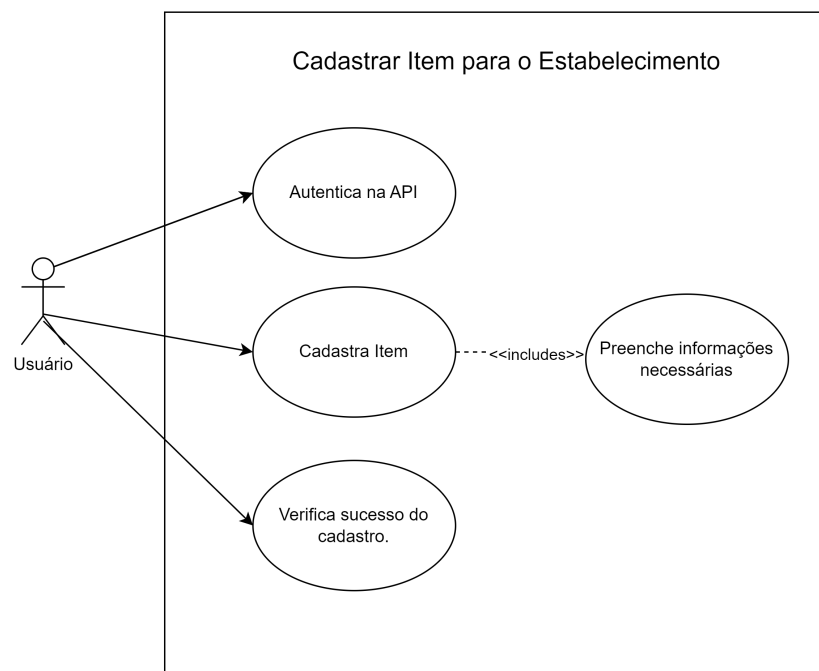


Figura 16 – Diagrama de Caso de Uso cadastro de item

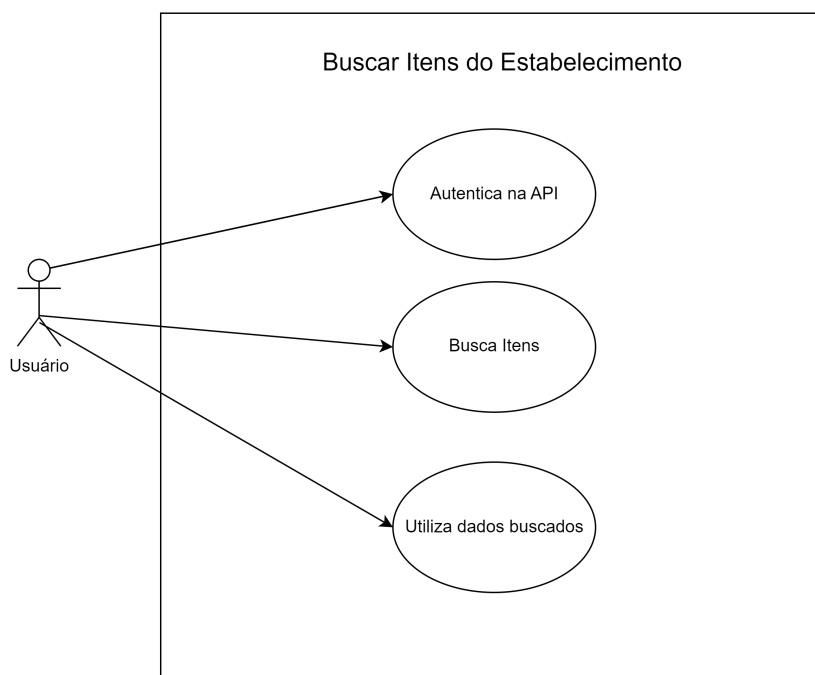


Figura 17 – Diagrama de Caso de uso buscar itens do estabelecimento

**Alterar Pedido para o Estabelecimento.** Caso de uso para alteração de dados de um pedido do estabelecimento. O Diagrama de Caso de Uso para esta funcionalidade pode ser visualizado na FIGURA 18.

## 3.2 Preparação do ambiente de desenvolvimento e codificação

A segunda fase do processo de desenvolvimento envolveu a preparação do ambiente e a codificação da solução, etapas essas que são relatadas com detalhes nos tópicos seguintes.

### 3.2.1 Preparação do ambiente de desenvolvimento

Nesta etapa, foram instaladas as ferramentas necessárias para tornar o ambiente de desenvolvimento apto para codificar na plataforma *.NET* e *C#*. A [Microsoft \(2023a\)](#), empresa proprietária do *.NET*, define a ferramenta como uma plataforma de desenvolvimento multiplataforma de código aberto gratuita para criar diversos tipos de aplicação.

Já o *C#*, segundo definição disponível na documentação oficial ([MICROSOFT, 2023c](#)), é uma linguagem de programação, orientada a objetos e fortemente tipada, permite que os desenvolvedores criem muitos tipos de aplicativos seguros e robustos que são executados no *.NET*. O *C#* tem suas raízes na família de linguagens C. Além disso, fornece construções de linguagem para dar suporte à Orientação à Objetos, tornando *C#* uma linguagem natural para criação e uso de componentes de software. Desde sua ori-

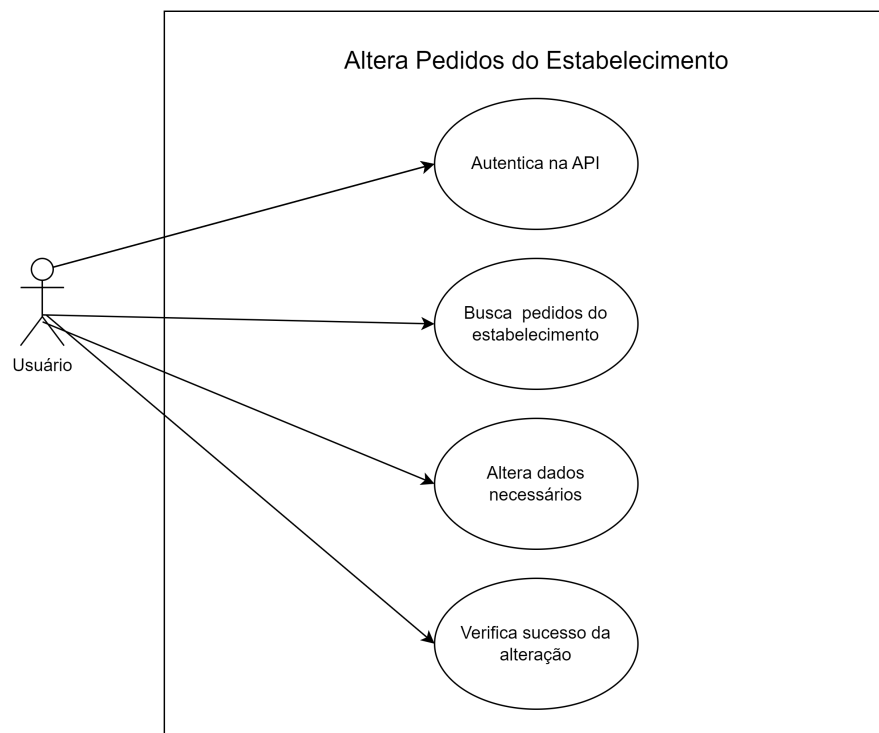


Figura 18 – Diagrama de Caso de uso alteração de dados do pedido

gem, o *C#* adicionou recursos para dar suporte a novas cargas de trabalho e práticas emergentes de design de software.

Finalmente, foi instalada localmente uma instância de banco de dados do *SQL Server Developer* (MICROSOFT, 2023b), executando na porta TCP 1334, de modo a implementar todas as funcionalidades de banco de dados deste projeto.

### 3.2.2 Codificação

Na etapa de codificação, as regras de negócio e os casos de uso são transformados em código de programação e o banco de dados é utilizado para armazenar as operações que ocorrem por meio destas regras. A seguir, discorre-se sobre esta etapa do processo de desenvolvimento de software.

#### Desenvolvimento do Banco de dados

Na etapa de desenvolvimento do banco de dados foram criadas as estruturas necessárias para que a aplicação seja capaz de armazenar as transações realizadas. A seguir, são descritos os dados armazenados em cada tabela mostrada na FIGURA 8:

1. Addresses - Tabela que armazena os endereços relacionados aos pedidos da aplicação.
2. AddressEstablishment - Armazena os endereços relacionados aos estabelecimentos.

3. Apps - Armazena os aplicativos que estão aptos a integrar com a aplicação.
4. Benefits - Armazena os benefícios relacionados a um pedido, que pode ser um cupom de desconto, por exemplo.
5. Campaign - Armazena as campanhas criadas dentro de aplicativos que podem gerar cupons de desconto.
6. Card - Armazena dados não sigilosos sobre cartões utilizados para pagar os pedidos.
7. Cash - Armazena informações sobre pagamentos em dinheiro relacionados aos pedidos.
8. Coordinates - Armazena as coordenadas relacionadas a um endereço.
9. Customers - Armazena dados dos clientes que realizaram pedidos.
10. Deliveries - Dados relacionados aos deliveries, que estão associados a um pedido.
11. DeliveryAddress - Endereços relacionados às entregas.
12. EstablishmentApps - Armazena a relação entre estabelecimentos e os aplicativos que utilizam.
13. EstablishmentAppsConfigs - Configurações dos aplicativos de cada estabelecimento.
14. EstablishmentClosure - Armazena informações sobre fechamentos diários dos estabelecimentos.
15. EstablishmentOpenning - Armazena informações sobre a abertura diária de estabelecimentos.
16. Establishment - Estabelecimentos cadastrados na plataforma.
17. Indoors - Armazena detalhes de retiradas em estabelecimento.
18. Item - Itens disponíveis de cada estabelecimento.
19. ItemOrders - Relacionamento entre Item e pedidos, armazena quais itens um pedido tem.
20. Merchants - Armazena qual estabelecimento do ifood um pedido está relacionado.
21. Method - Armazena os métodos de pagamento que cada pedido pode ter
22. OrderStatus - Tabela que guarda os status que um pedido pode ter.
23. Option - Opcionais relacionados com cada item de um pedido.



24. Orders - Tabela que armazena os pedidos.
25. Payments - Tabela que armazena os pagamentos associados a um pedido.
26. Phone- Tabela para armazenamento de telefones.
27. Schedules - Agendamentos de entrega relacionados a um pedido.
28. TakeOut - Retiradas associadas a um pedido.
29. Total - Armazena dados relacionados ao valor total de um pedido.
30. Users - Usuários de cada estabelecimento.
31. Wallet - Carteiras associadas a um pedido.

### Desenvolvimento do Serviço de Busca de Pedidos Externos

Para que seja possível a busca de pedidos em aplicativos terceiros de mercado, foi necessário desenvolver um serviço que é capaz de consumir APIs REST disponibilizadas pelas principais aplicações. A principal funcionalidade do serviço é buscar a todo instante atualizações sobre os pedidos de aplicações terceiras e integrar essas atualizações na base de dados local.

Dessa maneira, além dos pedidos recebidos pela própria aplicação, integramos pedidos dos aplicativos de mercado em um único banco de dados, possibilitando uma maior integração e controle por parte do estabelecimento. Na FIGURA 19, podemos visualizar uma representação do funcionamento básico do serviço de busca de pedidos externos.

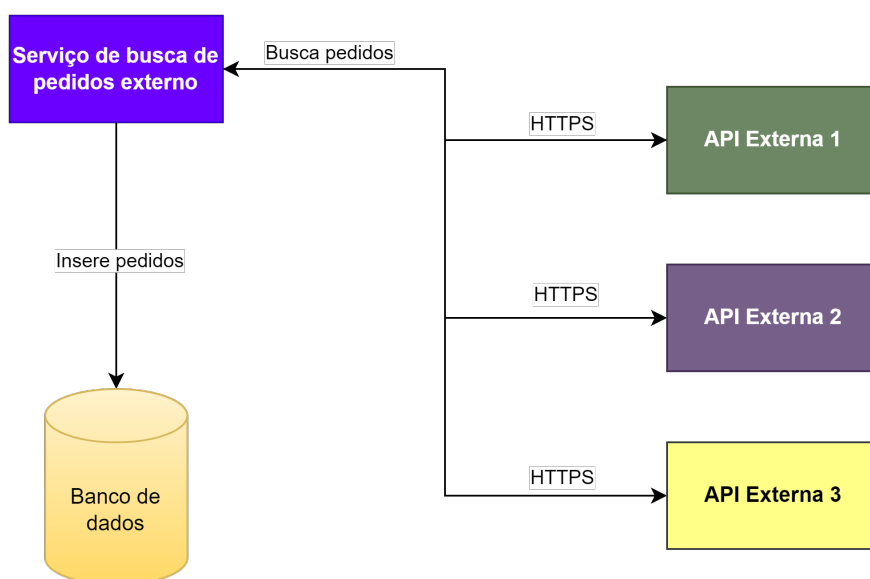


Figura 19 – Arquitetura simplificada do serviço de busca de pedidos externos

Para que o Serviço de Busca de Pedidos cumpra o seu papel, são seguidas algumas etapas, que envolve desde a verificação se o estabelecimento utiliza um aplicativo específico até autenticação e a busca efetiva na API externa. Utilizando, como exemplo, a integração com o aplicativo Ifood, os passos a serem seguidos são:

1. Busca de aplicativos abertos no banco de dados:

Nesta etapa são buscados quais estabelecimento utilizam cada um dos aplicativos e quais estão abertos, para que dessa maneira o serviço obtenha as credenciais e consuma os serviços externos de cada aplicativo do estabelecimento.

2. Autenticação no Serviço:

No momento da consulta dos aplicativos abertos, o serviço obtêm as credenciais necessárias para consumir os serviços externos.

3. Busca de eventos pendentes:

Consome os *end-points* do aplicativo externo para verificar se há algum novo evento.

4. Insere eventos pendentes na base de dados:

Caso exista algum evento ainda não integrado, o serviço insere no banco de dados para que seja consumido pela API.

Na FIGURA 20 podemos visualizar um fluxograma simplificado que representa cada uma das etapas que são responsabilidade do Serviço de Busca de Pedidos Externos.

## Desenvolvimento dos casos de uso da API

Para que as informações e regras de negócio definidas nos casos de uso da aplicação sejam devidamente persistidas, consultadas ou alteradas no banco de dados, a etapa de desenvolvimento da API envolveu a exposição de alguns *end-points* no padrão REST para que sejam consumidos por possíveis clientes. Para tal, foram desenvolvidas os seguintes casos de uso.

### Cadastro de Estabelecimento

Neste caso de uso, foi exposto o *end-point*, `"/api/Establishment"` que é o ponto de entrada da requisição de possíveis clientes da API. Sempre que um novo estabelecimento for se cadastrar na aplicação, será feita uma chamada HTTPS para este *end-point* via método *POST*. Essa chamada aciona a camada de casos de uso, que por sua vez aciona a camada de domínio. Logo em seguida, é chamada a camada de infraestrutura e persistido o novo estabelecimento no banco de dados.

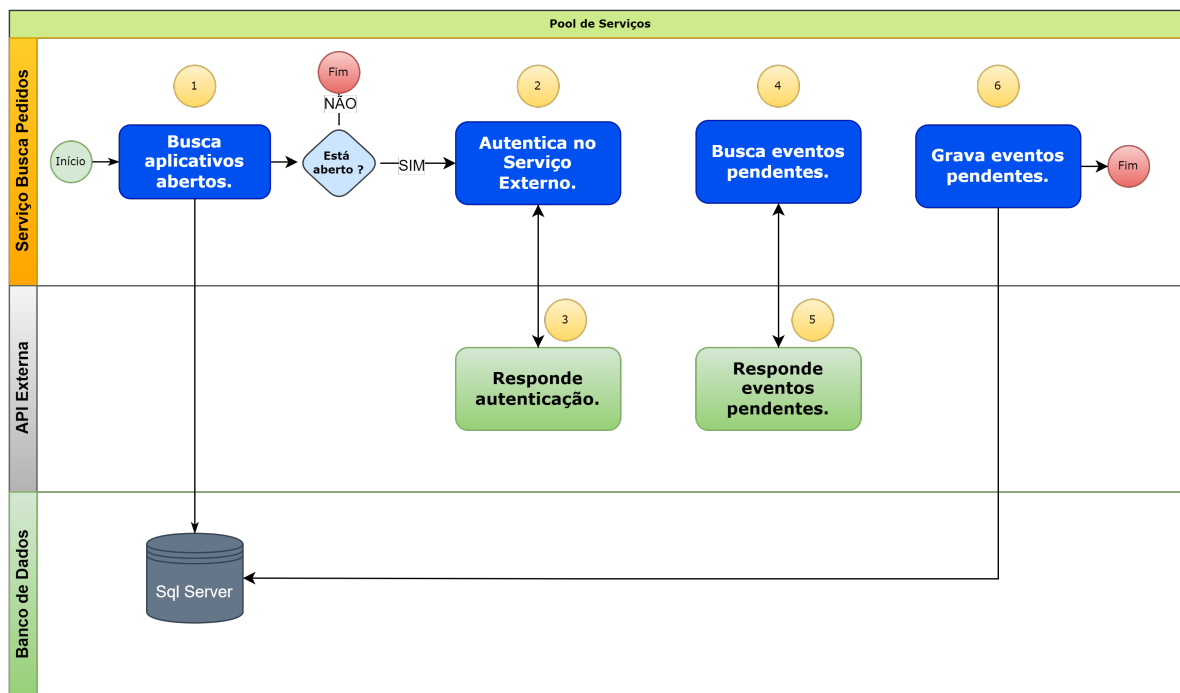


Figura 20 – Fluxograma de busca de pedidos externos

### Cadastro de Aplicativo para o Estabelecimento

Esse caso de uso foi efetivado com a exposição do *end-point* de cadastro de aplicativos para um estabelecimento. Sempre que um estabelecimento deseja incluir um novo aplicativo externo, esse *end-point* é utilizado, e é necessário enviar as credenciais de acesso do estabelecimento nas APIS dos aplicativos externos.

### Busca informações do Estabelecimento

Esse caso de uso é acionado quando um usuário logado, solicita, por meio do *end-point* `/api/Establishment` por meio do método GET as informações do estabelecimento.

### Abertura do Estabelecimento

Neste caso de uso foi desenvolvido o *end-point* `"/api/OpeningAndClosureEstablishment"` com o método *Post*, que efetua a abertura diária de do estabelecimento, permitindo assim que a aplicação passe a buscar pedidos externos deste aplicativo e receber pedidos internos.

### Cadastrar Pedido para o Estabelecimento.

Foi desenvolvido um *end-point* para receber pedidos de quaisquer aplicações próprias do estabelecimento capazes de consumir APIS REST's. Para inserir o novo pedido

basta consumir o *end-point* "api/Order" com o método *POST*.

#### Buscar Pedidos do Estabelecimento

Através do *end-point* "api/Order" por meio do método *GET* é possível buscar os pedidos pendentes do estabelecimento.

#### Cadastrar Item para o Estabelecimento

Foi desenvolvido o *end-point* "api/item", no qual é possível cadastrar um item para um estabelecimento através do método *POST*.

#### Alterar Pedido para o Estabelecimento.

Possibilita alterar dados de um pedido por meio de uma requisição *PUT* no *end-point* "api/Order".

#### Buscar Itens do Estabelecimento

Foi desenvolvido o *end-point* para buscar todos os itens cadastrados para o estabelecimento, que pode ser consumido através do método *GET*.

## 4 Avaliação do Protótipo

Para avaliar os *end-points* do protótipo da API foram efetuadas chamadas por meio do software *Postman* que é um cliente HTTPS capaz de consumir API's REST. Neste capítulo são mostradas as chamadas e as respostas obtidas para alguns testes realizados para os casos de uso da aplicação.

### 4.1 Cadastro de Estabelecimentos

Por meio do cliente *Postman*, foi feita uma chamada para o *end-point* descrito no caso de uso **Cadastro de Estabelecimento**. Assim, foram preenchidos os dados necessários, conforme descrito no capítulo anterior. Como pode ser visualizado no lado direito da FIGURA 21, recebeu-se um código de resposta *201(created)*, o que significa que se obteve sucesso no cadastro.

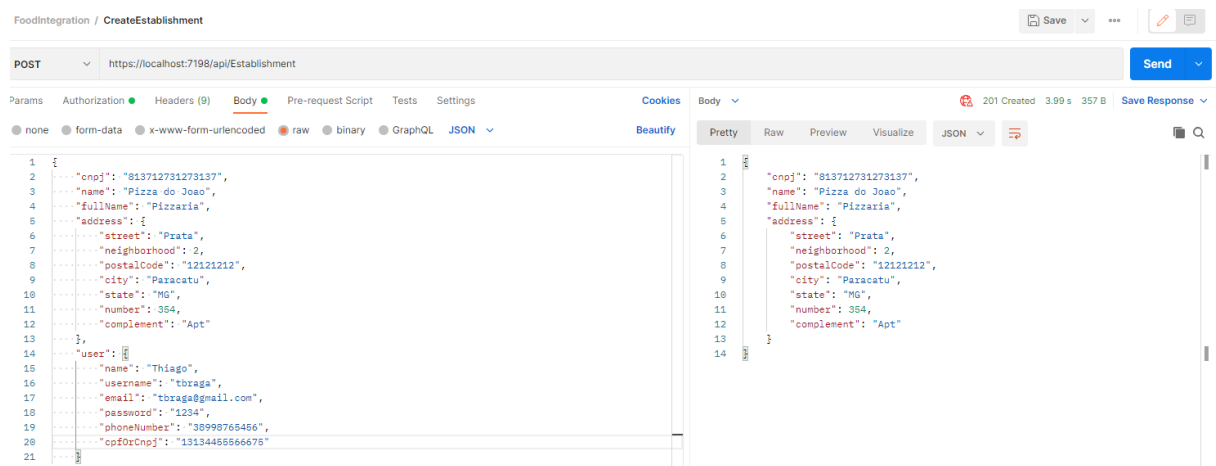


Figura 21 – Chamada POST para cadastro de estabelecimento

### 4.2 Cadastro de Aplicativo para o Estabelecimento

Também por meio do cliente *Postman*, foi realizada uma chamada ao *end-point* descrito no caso de uso **Cadastro de Aplicativo para o Estabelecimento**. Na FIGURA 22, do lado esquerdo, pode ser visualizado o corpo da requisição, conforme descrito no caso de uso. Já do lado direito pode ser visualizada a resposta da API com código *201 Created*, que representa o sucesso na requisição.

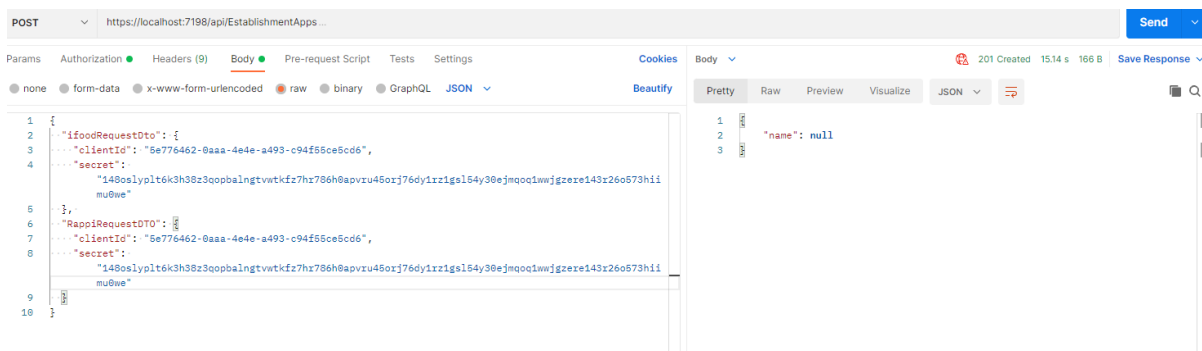


Figura 22 – Chamada POST para cadastro de aplicativo para o estabelecimento

### 4.3 Busca informações do Estabelecimento

Foi feito um teste também no *end-point* descrito no caso de uso **Busca Informações do Estabelecimento**. A requisição deste teste foi executada por meio do método *GET*, que obteve o corpo de resposta que pode ser visualizado no lado direito da FIGURA 23. O código de resposta obtido foi *200(OK)*.

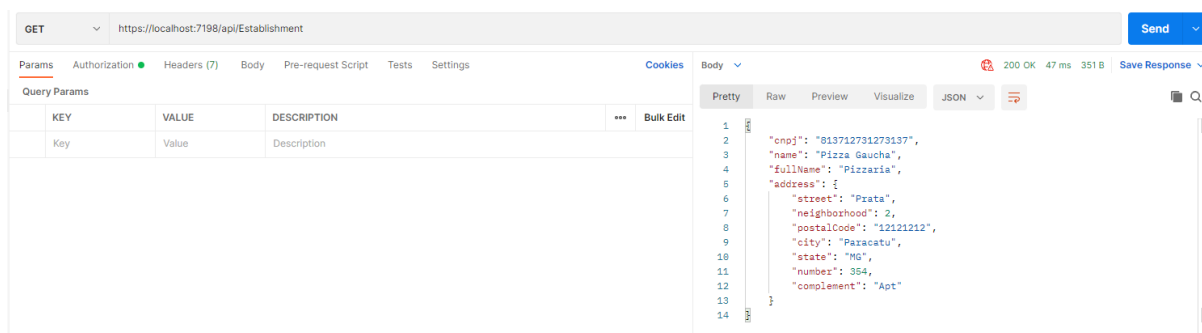


Figura 23 – Chamada GET para buscar informações do estabelecimento

### 4.4 Abertura do Estabelecimento

Foi realizado um teste chamando o *end-point* descrito no caso de uso **Abertura do Estabelecimento**, por meio do método *POST*. Conforme pode ser visualizado ao lado direito da FIGURA 24, obteve-se o código de resposta *201(created)*, que representa o sucesso na abertura.

### 4.5 Cadastrar Pedido para o Estabelecimento

Também se executou o teste de consumir o *end-point* descrito no caso de uso **Cadastrar Pedido Para o Estabelecimento**. Para tal, foi feita uma requisição seguindo

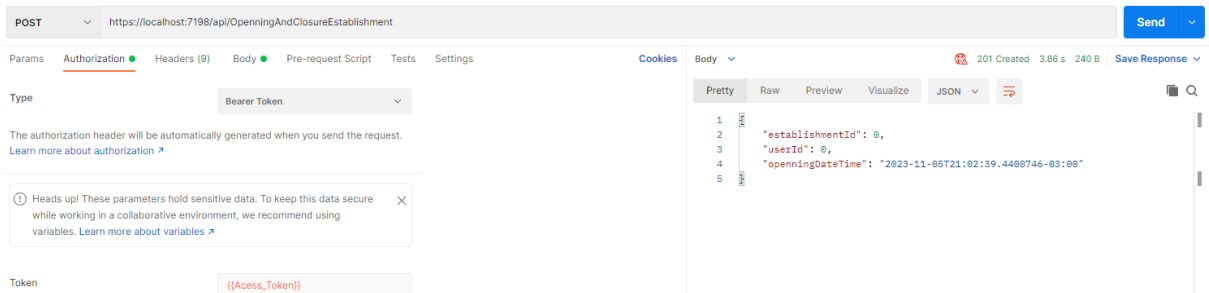


Figura 24 – Chamada POST para abertura do estabelecimento

o padrão de corpo estabelecido e foi obtida a resposta `201 Created`, ou seja, sucesso na criação do pedido. Na FIGURA 25, é possível visualizar o corpo da requisição e a resposta obtida.

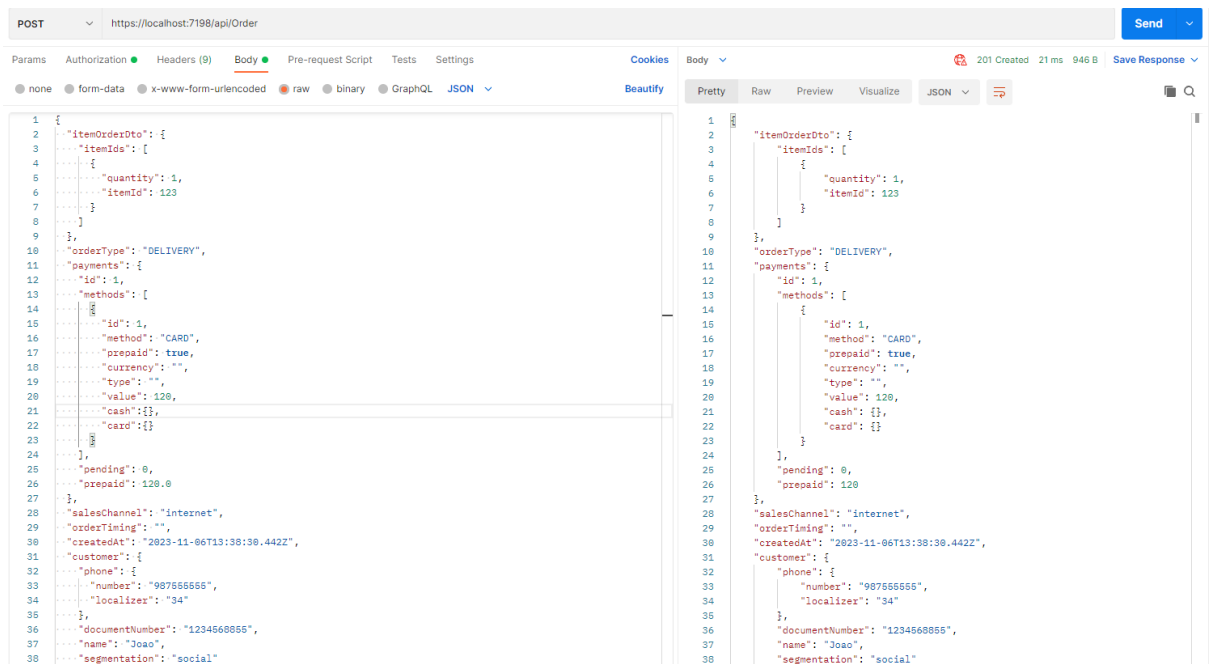


Figura 25 – Chamada POST para cadastro de pedido para o estabelecimento

## 4.6 Buscar Pedidos do Estabelecimento

Foi executada uma chamada no *end-point* descrito no caso de uso **Buscar Pedidos do Estabelecimento**. Conforme pode ser visualizado na FIGURA 26, foi retornado um código `200(OK)`. Também foi obtida uma lista com os pedidos do estabelecimento, visível no lado direito na figura.

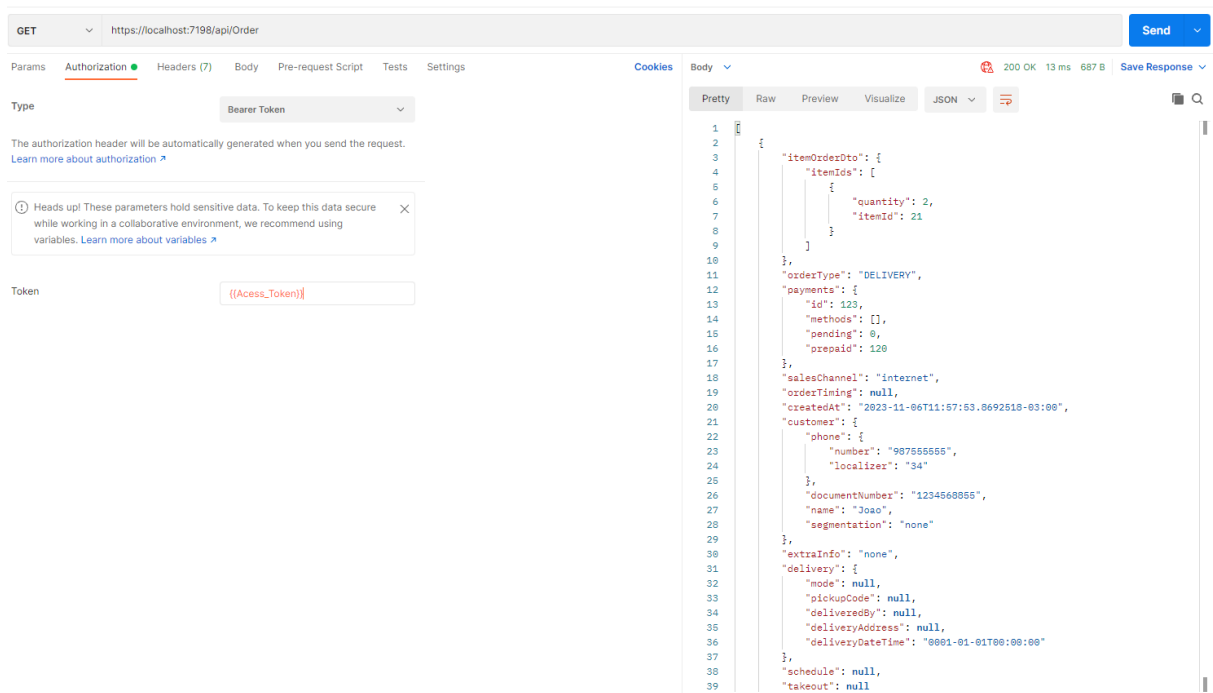


Figura 26 – Chamada Postman para busca de pedidos do estabelecimento

## 4.7 Cadastrar Item para o Estabelecimento

Para testar o *end-point* descrito no caso de uso **Cadastrar Item para o Estabelecimento**, foi efetuado um cadastro seguindo o corpo de requisição estabelecido. Como pode ser visualizado na FIGURA 27, obtemos a resposta (201 Created), o que representa o sucesso na criação do novo item.

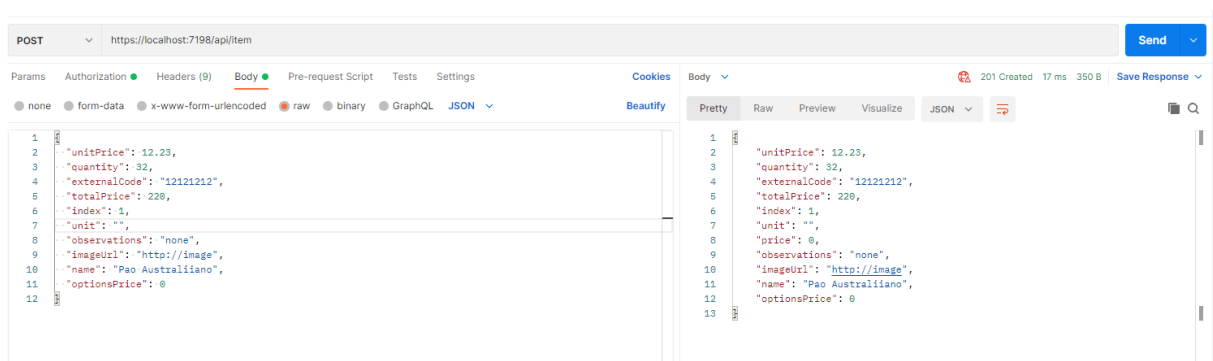
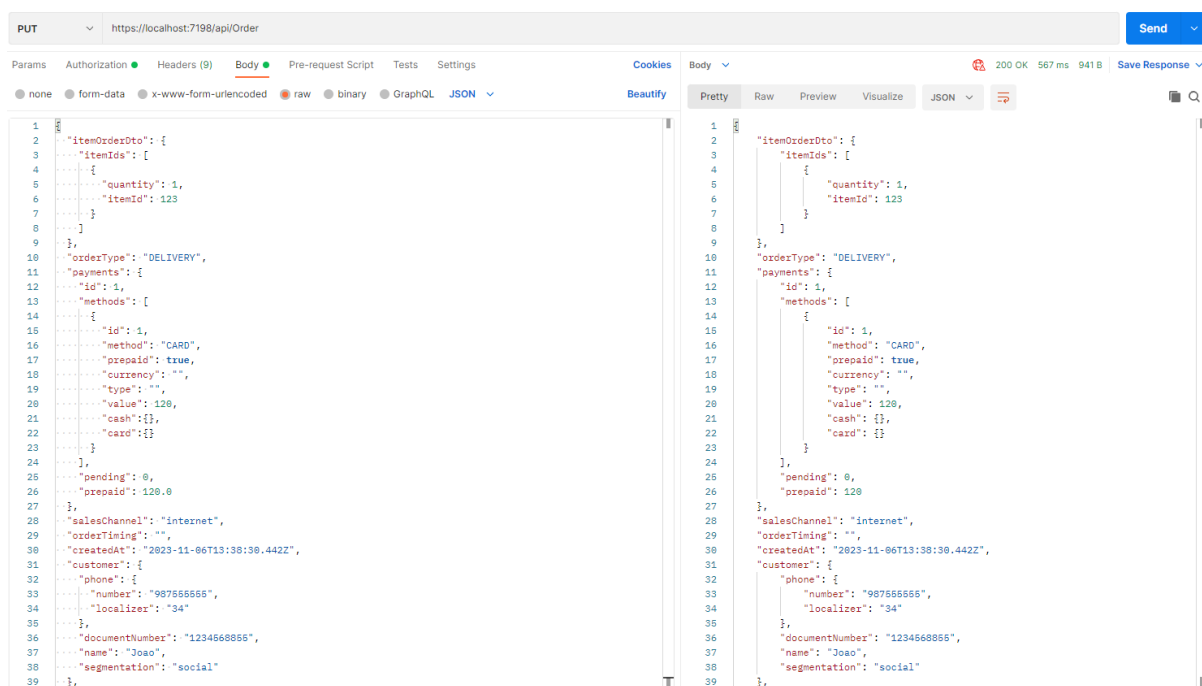


Figura 27 – Chamada POST para cadastro de item em um estabelecimento



## 4.8 Alterar Pedido para o Estabelecimento

Para o caso de uso **Alterar Pedido Para o Estabelecimento**, foi executada uma chamada no *end-point* descrito com alterações nos dados de um pedido. Essa chamada retornou *Sucesso* e os novos dados do pedido, como pode ser visualizado abaixo na FIGURA 28.



The screenshot shows a REST client interface with a PUT request to `https://localhost:7198/api/Order`. The request body is a JSON object with the following structure:

```
1 {
2   "itemOrderDto": {
3     "itemIds": [
4       {
5         "quantity": 1,
6         "itemId": 123
7       }
8     ]
9   },
10  "orderType": "DELIVERY",
11  "payments": {
12    "id": 1,
13    "methods": [
14      {
15        "id": 1,
16        "method": "CARD",
17        "prepaid": true,
18        "currency": "",
19        "type": "",
20        "value": 120,
21        "cash": {},
22        "card": {}
23      }
24    ]
25  },
26  "pending": 0,
27  "prepaid": 120.0,
28  "salesChannel": "internet",
29  "orderTiming": "",
30  "createdAt": "2023-11-06T13:38:38.442Z",
31  "customer": {
32    "phone": {
33      "number": "987565555",
34      "localizer": "34"
35    }
36  },
37  "documentNumber": "1234568855",
38  "name": "Joao",
39  "segmentation": "social"
40 }
```

The response body is a JSON object with the following structure:

```
1 {
2   "itemOrderDto": {
3     "itemIds": [
4       {
5         "quantity": 1,
6         "itemId": 123
7       }
8     ]
9   },
10  "orderType": "DELIVERY",
11  "payments": {
12    "id": 1,
13    "methods": [
14      {
15        "id": 1,
16        "method": "CARD",
17        "prepaid": true,
18        "currency": "",
19        "type": "",
20        "value": 120,
21        "cash": {},
22        "card": {}
23      }
24    ]
25  },
26  "pending": 0,
27  "prepaid": 120
28 },
29 "salesChannel": "internet",
30 "orderTiming": "",
31 "createdAt": "2023-11-06T13:38:38.442Z",
32 "customer": {
33   "phone": {
34     "number": "987565555",
35     "localizer": "34"
36   }
37 },
38 "documentNumber": "1234568855",
39 "name": "Joao",
40 "segmentation": "social"
41 }
```

Figura 28 – Chamada PUT para alteração de pedido do estabelecimento

## 4.9 Buscar Pedidos Externos para o Estabelecimento

Esse caso de uso consiste em um serviço que executa automaticamente a busca de pedidos em aplicativos externos e salva na base de dados própria da aplicação desenvolvida. Para testar essa funcionalidade, conforme visualizado na FIGURA 29, foi gerado um pedido no aplicativo externo Ifood<sup>1</sup> para um dos estabelecimentos cadastrados na base da aplicação. Já na FIGURA 30, é possível visualizar o pedido do Ifood armazenado na base de dados interna, para que possa ser integrado com os demais pedidos do estabelecimento.

<sup>1</sup> <<https://www.ifood.com.br/>>

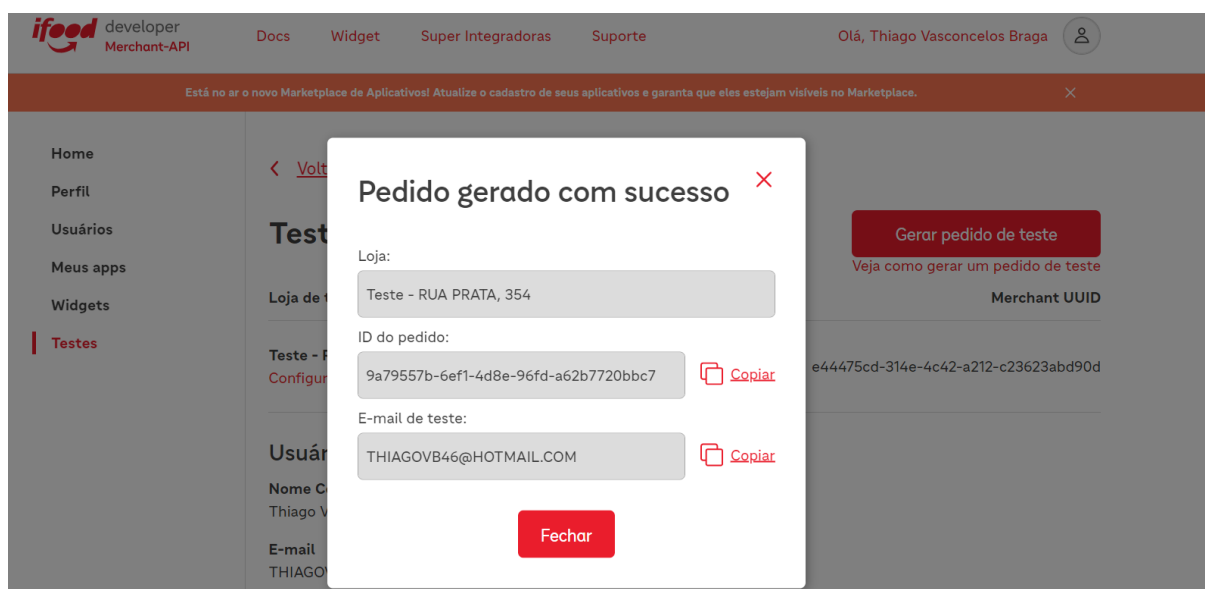


Figura 29 – geração de pedido Externo pelo aplicativo Ifood

SQL Server query: `SELECT externalId, salesChannel, createdAt FROM dbo.Orders`

externalId nvarchar(max)	salesChannel nvarchar(max)	createdAt datetime2
483cad83-cd5a-46f8-b841-3bad011ecde7	DEVELOPER-PORTAL	30/09/2023 16:37:52.4233333
9f694713-8921-44aa-a094-800fbb3afc2b	DEVELOPER-PORTAL	30/09/2023 19:16:29.6366667
483cad83-cd5a-46f8-b841-3bad011ecde7	DEVELOPER-PORTAL	30/09/2023 16:37:52.4233333
9f694713-8921-44aa-a094-800fbb3afc2b	DEVELOPER-PORTAL	30/09/2023 19:16:29.6366667
9a79557b-6ef1-4d8e-96fd-a62b7720bbc7	DEVELOPER-PORTAL	06/11/2023 14:13:28.1433333

Figura 30 – Pedido externo do aplicativo Ifood cadastrado no banco de dados

## 4.10 Buscar Itens do Estabelecimento

Finalmente, para testar o *end-point* descrito no caso de **Uso Buscar Itens do Estabelecimento**, foi realizada uma chamada *GET*, que retornou uma lista com os pedidos do estabelecimento com o código *200(OK)*, como pode ser visualizado na FIGURA 31.

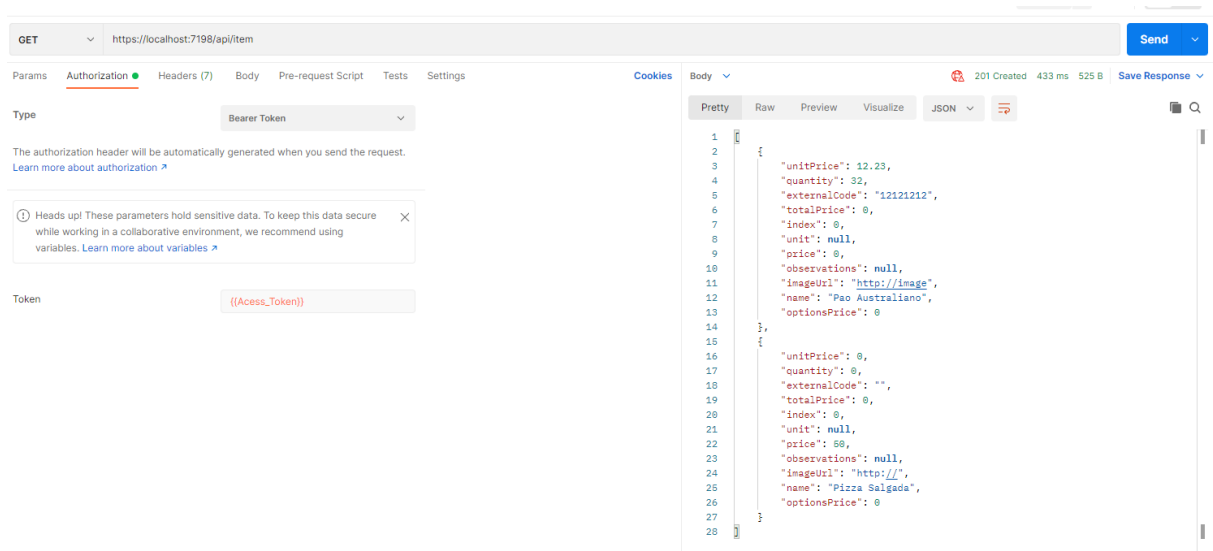


Figura 31 – Chamada GET para buscar os itens cadastrados por um estabelecimento

## 5 Conclusão

Este trabalho foi desenvolvido com objetivo de utilizar processos de software para modelagem e desenvolvimento de um protótipo para uma aplicação *back-end* que disponibilize uma interface de programação (API, do inglês *Application Programming Interface*) para desenvolvedores, capaz de receber e gerenciar pedidos de restaurantes, sejam eles provenientes dos principais aplicativos de entrega disponíveis ou de sites e aplicações próprias do estabelecimento. Isso tornou possível integrar todas essas informações em uma única ferramenta que pode ser consumida por aplicações web, mobile, de internet das coisas (IoT, do inglês *Internet of Things*), entre outras.

Um dos objetivos definidos foi a utilização de metodologias de processo de software, algumas das etapas utilizadas para o desenvolvimento do trabalho foram:

1. Planejamento: O Desenvolvimento dessa etapa envolveu definir cronogramas de realização de tarefas, definição de quais casos de uso a aplicação envolveria e quais diagramas seriam necessários para iniciar a etapa seguinte.
2. Modelagem Nessa etapa foi desenvolvida toda a arquitetura do software e processos que envolvem o seu funcionamento, por meio da arquitetura definida foram desenvolvidos diagramas no intuito de deixar muito bem definido o desenho arquitetural da aplicação.
3. Construção Na etapa de construção, todo o planejamento realizado e os diagramas de arquitetura e funcionamento definidos foram utilizados para a codificação do protótipo da aplicação.

Através das metodologias de processo de software utilizadas, foram implementadas as seguintes funcionalidades em protótipo da aplicação:

1. Cadastro de Estabelecimento
2. Cadastro de Aplicativo para o Estabelecimento
3. Busca Informações do Estabelecimento
4. Abertura do Estabelecimento
5. Cadastrar Pedido para o Estabelecimento
6. Buscar Pedidos do Estabelecimento
7. Cadastrar Item para o Estabelecimento

8. Alterar Pedido para o Estabelecimento
9. Buscar Pedidos Externos para o Estabelecimento
10. Buscar Itens do Estabelecimento

Por meio do desenvolvimento deste trabalho de conclusão de curso, foi possível notar uma associação direta dos conhecimentos necessários para sua execução, com o conteúdo ministrado ao decorrer do curso de Bacharelado em Sistemas de Informação. Entre os principais conhecimentos consolidados podemos citar: Bancos de Dados, Programação Orientada à Objetos, Linguagem de Programação, Mecanismos de Autenticação, Protocolo de Comunicação HTTPS, Modelagem e Engenharia de Software, dentre outros.

Como sugestões para continuação do trabalho, é possível incluir algumas funcionalidades não implementadas que podem ser de muita relevância para um possível produto final, como geração de relatórios de compras e vendas e integração de outros aplicativos. Outra sugestão para continuação ou reutilização do trabalho é o desenvolvimento de uma aplicação de *front-end* que consuma a *API* desenvolvida.

Para possíveis consultas futuras, os códigos fontes desenvolvidos neste trabalho estão disponíveis em repositório. <sup>1</sup>.

---

<sup>1</sup> <[https://github.com/thiagovb46/Food\\_Integration/](https://github.com/thiagovb46/Food_Integration/)>

# Referências

BONFANTI, D. Por que é importante centralizar todos os seus pedidos delivery em uma única plataforma? **blog Linx**, ago. 2020. Disponível em: <<https://www.linx.com.br/blog/por-que-e-importante-centralizar-todos-os-seus-pedidos-delivery-em-uma-unica-plataforma/>>. Acesso em: 17 nov. 2023. Citado 2 vezes nas páginas 8 e 9.

CERAMI, E. **Web Services Essentials**: Distributed applications with XML-RPC, SOAP, UDDI & WSDL. EUA: O'Reilly Media, 2009. 304 p. ISBN 978-0596002244. Citado na página 14.

JUNIOR, F. Delivery transformou tendência em necessidade e continua em crescimento. **Jornla da USP**, abr. 2021. Disponível em: <<https://jornal.usp.br/?p=395377>>. Acesso em: 17 nov. 2023. Citado na página 8.

MARTIN, R. C. **Código Limpo**: Habilidades práticas do agile software. Rio de Janeiro, RJ: Alta Books, 2009. 425 p. ISBN 978-8550816005. Citado na página 12.

\_\_\_\_\_. **Arquitetura Limpa**: O guia do artesão para estrutura e design de software. Rio de Janeiro, RJ: Alta Books, 2020. 780 p. ISBN 978-8550816005. Citado 4 vezes nas páginas 5, 12, 13 e 14.

MICROSOFT. **O que é o .NET? Introdução e visão geral**. 2023. Disponível em: <<https://learn.microsoft.com/pt-br/dotnet/core/introduction>>. Acesso em: 22 jan. 2023. Citado na página 29.

\_\_\_\_\_. **O que é o SQL Server?** 2023. Disponível em: <<https://learn.microsoft.com/pt-br/sql/sql-server/what-is-sql-server?view=sql-server-ver16>>. Acesso em: 22 jan. 2023. Citado na página 30.

\_\_\_\_\_. **Um tour pela linguagem C#**. 2023. Disponível em: <<https://learn.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/>>. Acesso em: 12 jan. 2023. Citado na página 29.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software**: Uma abordagem profissional. 8. ed. Porto Alegre, RS: AMGH, 2016. 968 p. ISBN 978-8580555332. Citado 2 vezes nas páginas 11 e 12.

REDHAT. **API REST**. 2023. Disponível em: <<https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>>. Acesso em: 12 jan. 2023. Citado na página 14.

RICHARDS, M.; FORD, N. **Fundamentals of Software Architecture**: An engineering approach. EUA: O'Reilly Media, 2020. 419 p. ISBN 978-1492043454. Citado na página 12.