

**PONTIFÍCIA UNIVERSIDADE CATÓLICA
DE MINAS GERAIS**

PUC Minas Virtual

**Pós-Graduação Lato Sensu em Arquitetura de
Software Distribuído**

Projeto Integrado

Relatório Técnico

Sistema de Gestão de Delivery

Julio Cezar da Silva

**Belo Horizonte
2023**

Julio Cezar da Silva

Sistema de Gestão de Delivery

Trabalho de Conclusão de Curso de Especialização em Arquitetura de Software
Distribuído como requisito parcial à obtenção do título de especialista.

Orientador: Prof. Luiz Alberto

**Belo Horizonte
2023**

Sumário

1	Introdução	3
2	Objetivo	4
2.1	Objetivos Específicos	4
3	Restrições Arquiteturais	5
4	Requisitos Funcionais	6
4.1	Módulo de produtos	6
4.1.1	RF-1: Cadastro de produtos	6
4.1.2	RF-2: Cadastro de categoria	6
4.1.3	RF-3: Cadastro de complementos	6
4.1.4	RF-4: Cadastro de combos	6
4.2	Módulo de clientes	6
4.2.1	RF-5: Cadastro de clientes	6
4.2.2	RF-6: Cadastro de endereços de clientes	7
4.3	Módulo de pedidos	7
4.3.1	RF-7: Cadastro de pedidos delivery	7
4.3.2	RF-8: Cadastro de pedidos mesa	7
4.3.3	RF-9: Cadastro de pedidos cliente	7
4.4	Módulo de estoque	7
4.4.1	RF-10: Cadastro de ingredientes	7
4.4.2	RF-11: Relatório de estoque	7
4.5	Módulo financeiro	8
4.5.1	RF-12: Formas de pagamento	8
4.5.2	RF-13: Pagamento Online	8
4.6	Módulo autenticação/autorização	8
4.6.1	RF-14: Níveis de acesso	8
4.6.2	RF-15: Autenticação	8
5	Requisitos Não Funcionais	9
5.1	Manutenabilidade	9
5.1.1	RNF-1: Desenvolvimento em camadas	9
5.2	Usabilidade	9
5.2.1	RNF-2: Padrão de interfaces	9
5.3	Segurança	9
5.3.1	RNF-3: Banco de dados	9
5.3.2	RNF-4: Autenticação/Autorização	9
5.4	Interoperabilidade	9
5.4.1	RNF-5: Comunicação com sistema de pagamento	9
5.5	Desempenho	10
5.5.1	RNF-6: Acesso após Autenticação	10
6	Mecanismos Arquiteturais	11

7	Modelagem Arquitetural	12
7.1	Diagrama de Contexto	12
7.2	Diagrama de Container	13
7.3	Diagrama de Componentes	14
8	Vídeo Apresentação 1	16
9	Avaliação Arquitetural ATAM	17
9.1	Análise das abordagens arquiteturais	17
9.1.1	Manutenabilidade	17
9.1.2	Usabilidade	17
9.1.3	Segurança	17
9.1.4	Interoperabilidade	17
9.1.5	Desempenho	18
9.2	Cenários	18
9.2.1	Cenário 1	18
9.2.2	Cenário 2	18
9.2.3	Cenário 3	18
9.2.4	Cenário 4	18
9.2.5	Cenário 5	18
9.2.6	Cenário 6	18
9.3	Evidências da Avaliação de Cenários	19
9.3.1	Cenário 1	19
9.3.2	Cenário 2	21
9.3.3	Cenário 3	23
9.3.4	Cenário 4	24
9.3.5	Cenário 5	27
9.3.6	Cenário 6	29
10	Avaliação crítica dos resultados	31
11	Conclusão	32
12	Repositórios	33
13	Vídeo Apresentação	34

1. Introdução

O mercado de delivery tem crescido de forma significativa no Brasil, o mesmo foi impulsionado ainda mais devido a pandemia de COVID-19.

Segundo a Associação brasileira de Bares e Restaurantes(Abrasel), o setor de alimentação movimentou cerca de R\$ 200 bilhões em 2019, sendo que destes, 11% foi representado pelo delivery. Em 2020 com a pandemia, o mercado cresceu para 20% do faturamento total do setor[1].

Durante a pandemia tivemos um grande aumento no uso de plataformas de delivery, principalmente do ifood[2] que hoje é a principal plataforma de delivery no Brasil. Essas plataformas possuem uma taxa altíssima, mesmo as altas taxas aplicadas, que variam de 12% a 27%, além de uma mensalidade em torno de R\$ 100,00[6].

De acordo com o Instituto FoodService Brasil(IFB), no Brasil espera-se que o delivery cresça em torno de 7,5% em 2023, fazendo com que os estabelecimentos invistam em ampliar e aprimorar este serviço de entrega de comida[5].

A transformação digital[3] está presente nas nossas atividades diárias, e não está limitada ao uso pessoal, o setor de delivery está cada vez mais preparado digitalmente, e os estabelecimentos que não adotarem as tecnologias que possam facilitar seus processos, controles e integração com os clientes ficarão para trás no mercado.

Com a utilização de tecnologias como sistemas gerenciais, o estabelecimento evita e minimiza erros e prejuízos nos seus processos, tem um melhor controle de estoque, financeiro e diversas rotinas do seu comércio. Além de otimizar o tempo tanto de preparação com as comandas chegando automaticamente via sistema, sem a movimentação de papel que pode se perder no processo, quanto a agilidade em atender o cliente e cadastrar seu pedido, algo que pode ser inclusive feito pelo próprio cliente direto de sua casa.

2. Objetivo

O objetivo do projeto é apresentar uma arquitetura de um sistema para estabelecimentos de alimentação, de modo que os mesmos possam ter um controle dos seus produtos, clientes e da sua rotina. Além de um sistema que permita aos clientes efetuarem pedidos de forma online.

Portanto, com objetivo de desenvolver a modelagem de um software no qual os estabelecimentos possam utilizar e oferecer através do mesmo diversos benefícios aos seus clientes, como promoções, fidelização do cliente, mensagens diretas ao smartphone do cliente, até mesmo preços mais baixos desde que o estabelecimento faça um bom marketing para atrair clientes que utilizem plataformas de pedidos como ifood.

O cliente poderá acessar o software através de um aplicativo para smartphone, seja uma progressive web page, ou mesmo um aplicativo nativo, o que será definido no decorrer das análises e projeto arquitetural.

2.1. Objetivos Específicos

Os objetivos específicos são:

- Permitir ao estabelecimento ter um controle dos seus cadastros de clientes, produtos, promoções, etc.
- Oferecer ao cliente do estabelecimento um sistema facilitador para que o mesmo possa efetuar pedidos diretamente da sua casa.
- Facilitar ao estabelecimento oferecer promoções diretamente ao cliente em casa, através do próprio sistema, isto é, enviar promoções para o cliente, mensagens de aniversário ou que mais convier ao estabelecimento.
- Facilitar o controle de estoque do estabelecimento.
- Facilitar o controle financeiro do estabelecimento.
- Permitir o fácil gerenciamento do estabelecimento através de diversos relatórios.
- Oferecer ao cliente a opção de pagar online, para isso deverá ser feito integrações com os principais players de pagamento, como pagseguro, mercadopago, etc.
- Permitir ao estabelecimento receber pedidos de plataformas de delivery como ifood.

3. Restrições Arquiteturais

Table 3.1: Restrições

ID	Descrição
RA001	O software deverá ser desenvolvido em Java utilizando o framework Spring
RA002	As APIs devem seguir o padrão RestFul
RA003	O software deverá integrar com plataformas de delivery, de forma a permitir os clientes pedirem por ela, como ifood.
RA004	O software deverá integrar com pelo menos um sistema de pagamento online, como pagseguro, mercadopago.
RA005	O software administrativo do estabelecimento deverá ter níveis de acesso para proteger dados sensíveis, como endereço e outras informações pessoais dos clientes

4. Requisitos Funcionais

Abaixo listamos os requisitos funcionais da aplicação[7]. Os requisitos terão níveis de dificuldade e prioridades definidos em B - Baixa, M - Média e A - Alta.

4.1. Módulo de produtos

4.1.1. RF-1: Cadastro de produtos

O sistema deve permitir que o colaborador cadastre produtos com nome, categoria, preço de custo por tamanho, preço de venda por tamanho, unidade de medida, descrição, foto, se será feito controle de estoque, ficha técnica, complementos.

Dificuldade: M - Prioridade: A

4.1.2. RF-2: Cadastro de categoria

O sistema deve permitir cadastrar a categoria de produtos, definindo possíveis tamanhos dos produtos nessa categoria ou tamanho única, por exemplo: Refrigerantes, definir os tamanhos 350ml, 500ml, 1l, 1.5l, 2l, etc.

Dificuldade: M - Prioridade: A

4.1.3. RF-3: Cadastro de complementos

O sistema deverá permitir cadastrar complementos e seus valores, os mesmos deverão ser ligados ao tamanho do produto no cadastro de produtos. Exemplo: Acrescimo de Bacon pizza M, Acrescimo de Bacon pizza G e assim por diante.

Dificuldade: B - Prioridade: M

4.1.4. RF-4: Cadastro de combos

O sistema deverá permitir o cadastro de combos, juntando vários produtos cadastrados em um combo e definindo o valor para o combo.

Dificuldade: A - Prioridade: M

4.2. Módulo de clientes

4.2.1. RF-5: Cadastro de clientes

O sistema deve permitir cadastrar os clientes com nome, email, data nascimento, telefone principal, cpf ou cnpj, identidade ou inscrição estadual, endereço principal(endereço, número, complemento, bairro, cidade, cep, valor do frete).

Dificuldade: B - Prioridade: A

4.2.2. RF-6: Cadastro de endereços de clientes

O sistema deverá permitir cadastrar mais endereços para o cliente, com nome do endereço e os campos definidos no requisito RF-004.

Dificuldade: B - Prioridade: M

4.3. Módulo de pedidos

4.3.1. RF-7: Cadastro de pedidos delivery

O sistema deve permitir cadastrar pedidos para os clientes, selecionando cliente, produtos, endereço de entrega, forma de pagamento, status do pedido (Aberto, Em preparo, Pronto/Para retirar, Saiu para Entregar, Finalizado/Entregue), lançamento de desconto ou taxa de serviço.

Dificuldade: A - Prioridade: A

4.3.2. RF-8: Cadastro de pedidos mesa

O sistema deve permitir cadastrar pedidos para os clientes no estabelecimento, selecionando cliente, produtos, forma de pagamento, status do pedido (Aberto, Em preparo, Pronto/Para retirar, Saiu para Entregar, Finalizado/Entregue), lançamento de desconto ou taxa de serviço, divisão de valores.

Dificuldade: A - Prioridade: A

4.3.3. RF-9: Cadastro de pedidos cliente

O sistema deve permitir que o próprio cliente faça seu pedido online, selecionando os produtos, endereço de entrega.

Dificuldade: A - Prioridade: A

4.4. Módulo de estoque

4.4.1. RF-10: Cadastro de ingredientes

O sistema deve permitir o gerenciamento do estoque através do cadastro de ingredientes, e nos produtos que controlam estoque ter a ficha técnica dos mesmos ligadas aos ingredientes. O cadastro deverá ter informações como nome do ingrediente, unidade de medida, data de vencimento, preço de custo.

Dificuldade: A - Prioridade: M

4.4.2. RF-11: Relatório de estoque

O sistema deve possuir relatório que mostre o estoque de produtos e os produtos com vencimento próximo, de modo a evitar perdas.

Dificuldade: A - Prioridade: M

4.5. Módulo financeiro

4.5.1. RF-12: Formas de pagamento

O sistema deve permitir o gerenciamento das formas de pagamento, para que possam ser selecionadas no lançamento do pedido. Exemplo: Cartão de Crédito, Cartão de Débito, Dinheiro, Pix, etc. Também as respectivas bandeiras quando necessário.

Dificuldade: A - Prioridade: M

4.5.2. RF-13: Pagamento Online

O sistema deve fazer integração com pelo menos 1 meio de pagamento online como pagseguro ou mercado pago.

Dificuldade: A - Prioridade: B

4.6. Módulo autenticação/autorização

4.6.1. RF-14: Níveis de acesso

O sistema deve permitir o cadastro/gerenciamento de níveis de acesso, a princípio dois níveis serão suficientes: Usuário e Administrador. Os usuários poderão fazer lançamento de pedidos, cadastros de clientes. Os administradores terão acesso irrestrito ao sistema.

Dificuldade: B - Prioridade: A

4.6.2. RF-15: Autenticação

O sistema deve ser fechado, não permitido o acesso a ele, somente para usuários cadastrados através de autenticação. Clientes externos deverão poder acessar um cardápio online e ao efetuar pedido fazer o devido cadastro.

Dificuldade: A - Prioridade: A

5. Requisitos Não Funcionais

Abaixo listamos os requisitos não funcionais da aplicação em termos arquiteturais[7].

5.1. Manutenabilidade

5.1.1. RNF-1: Desenvolvimento em camadas

O sistema deve ser desenvolvido utilizando uma arquitetura em camadas, de modo a facilitar a manutenção do mesmo. A princípio podemos utilizar o modelo MVC(Model View Controller).

Prioridade: M

5.2. Usabilidade

5.2.1. RNF-2: Padrão de interfaces

O sistema deve ser desenvolvido de modo a manter um padrão na interface com o usuário, de modo a facilitar a familiarização do usuário com as telas do sistema. O sistema deverá funcionar também de maneira responsiva, de modo a ser fácil de se utilizar em dispositivos móveis quanto em computadores.

Prioridade: A

5.3. Segurança

5.3.1. RNF-3: Banco de dados

O sistema deve proteger os dados sensíveis dos clientes e usuários através de criptografia de senhas no banco de dados, acesso restrito ao banco de dados, o mesmo não deverá ter acesso externo a internet.

Prioridade: A

5.3.2. RNF-4: Autenticação/Autorização

O sistema deverá ter mecanismos de autenticação e autorização para somente determinados níveis acessar informações mais sensíveis.

Prioridade: A

5.4. Interoperabilidade

5.4.1. RNF-5: Comunicação com sistema de pagamento

O sistema deve ser capaz de se comunicar/integrar com pelo menos um meio de pagamento externo como pagseguro através de uma API.

Prioridade: M

5.5. Desempenho

5.5.1. RNF-6: Acesso após Autenticação

O sistema deve ser capaz de exibir sua tela inicial após a inserção dos dados de acesso em menos de 5 segundos.

Prioridade: M

6. Mecanismos Arquiteturais

Abaixo listamos os mecanismos que irão compor a arquitetura do software proposto.

Table 6.1: Módulo Cadastros

Análise	Design	Implementação
Mapeamento objeto-relacional	Utilização do padrão de projeto Data Access Object (DAO)	Utilização do Hibernate como framework ORM
Gerenciamento de dados estruturados	Modelagem de banco de dados relacional	Utilização do PostgreSQL como SGBD
Autenticação de usuários	Utilização de algoritmo de criptografia unidirecional para armazenamento de senhas, como SHA-256 ou bcrypt, e implementação de um mecanismo de validação de login e senha no servidor.	Utilização de um framework de autenticação, como Spring Security
Integração entre diferentes sistemas	Utilização de protocolos e formatos de dados interoperáveis, como JSON	Utilização de APIs RESTful para comunicação entre os sistemas.
Registro de atividades do sistema (log)	Utilização de frameworks de logging	Log4j
Gerenciamento de versões de código	Utilização de um sistema de controle de versão	GitHub
Documentação das APIs	Utilização ferramentas de documentação	Swagger
Teste do Software	Utilização de testes unitários, trabalhar com a metodologia TDD	JUnit
Front-end	Interface de comunicação com o usuário do sistema	Angular
Back-end	Utilização de arquitetura MVC e injeção de dependências	Spring framework

7. Modelagem Arquitetural

Apresentamos abaixo a modelagem arquitetural do sistema de delivery, utilizaremos para nossa modelagem o modelo C4.

7.1. Diagrama de Contexto

Começaremos pelo diagrama de contexto, nele temos o nosso sistema de delivery Pop Food que é responsável por gerenciar todo o estabelecimento, e os atores como clientes e atendentes/administradores que fazem pedido de comida e são responsáveis pela administração dentro do software respectivamente. Podemos verificar ainda os sistemas externos como PagSeguro e Ifood para pagamento e pedidos online respectivamente.

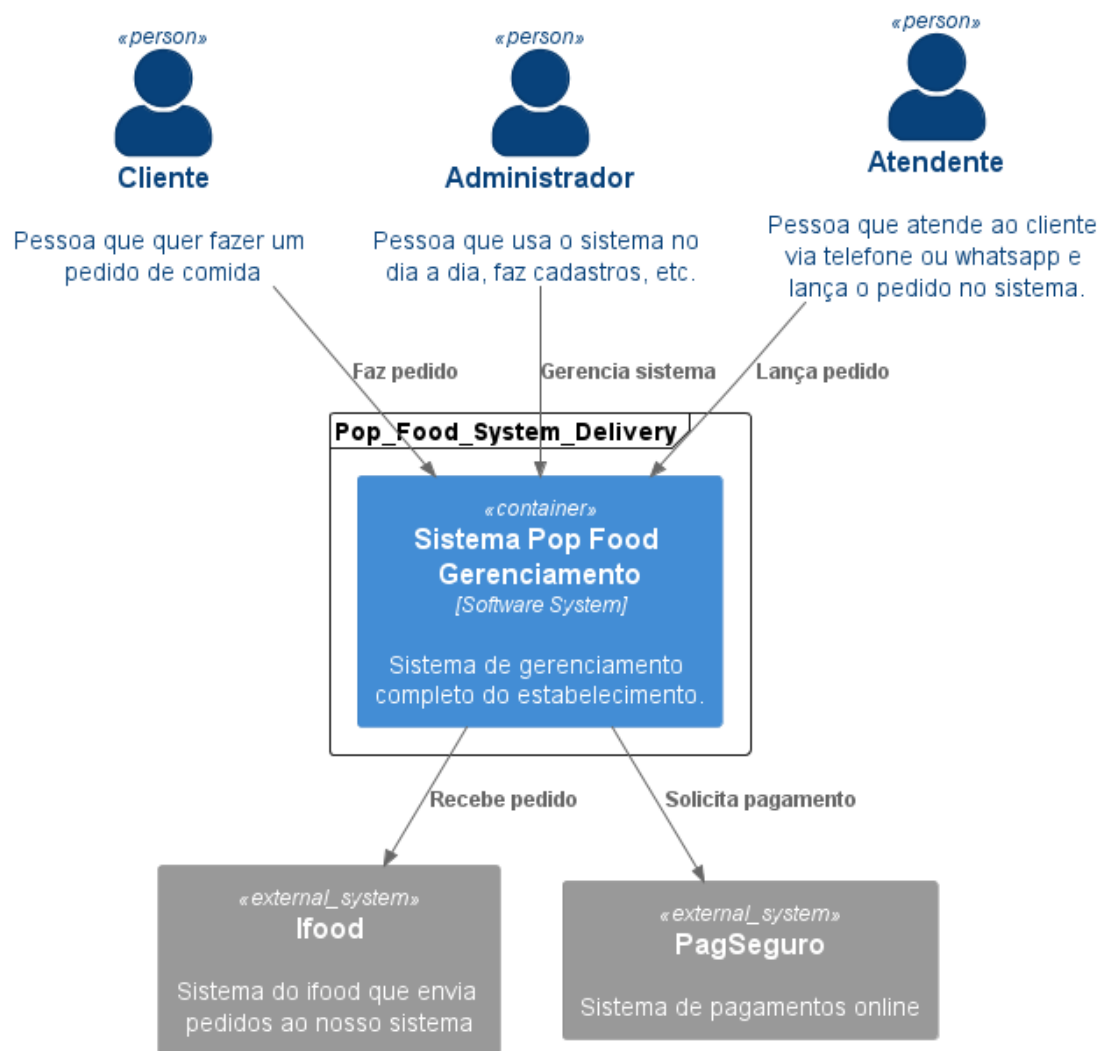


Figure 7.1: Diagrama de Contexto

Imagem resolução maior

7.2. Diagrama de Container

Mostramos agora o diagrama de container, nele podemos observar que teremos dois containers de frontend, ambos utilizando a tecnologia Angular, o frontend do Cliente é responsável por apresentar as telas do sistema que os clientes utilizarão para se cadastrar, ver o menu de opções, efetuar pedidos e pagamentos online.

No frontend de administração será disponibilizado as telas de login e administração do sistema, como cadastro de produtos, possibilidade de lançamento de pedidos, cadastro de clientes, etc.

Neste diagrama vemos o container do back end, que utilizará spring boot e será responsável pelas regras de negócio e integração com banco de dados e sistemas externos como iFood e PagSeguro.

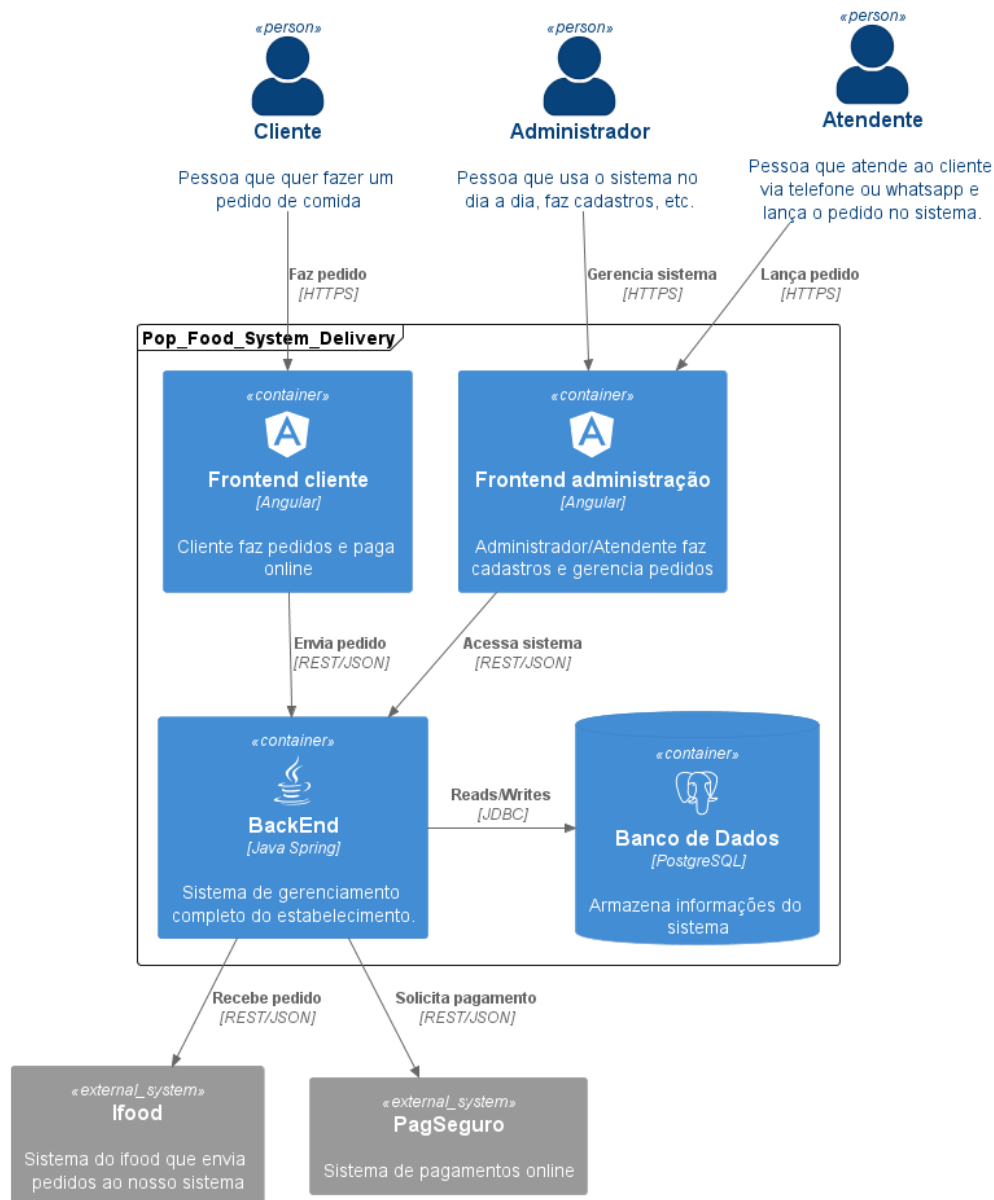


Figure 7.2: Diagrama de Container

Imagem resolução maior

7.3. Diagrama de Componentes

Entrando mais em detalhes, temos agora o diagrama de componentes, nele podemos ver as abstrações em caixa preta dos principais componentes do sistema.

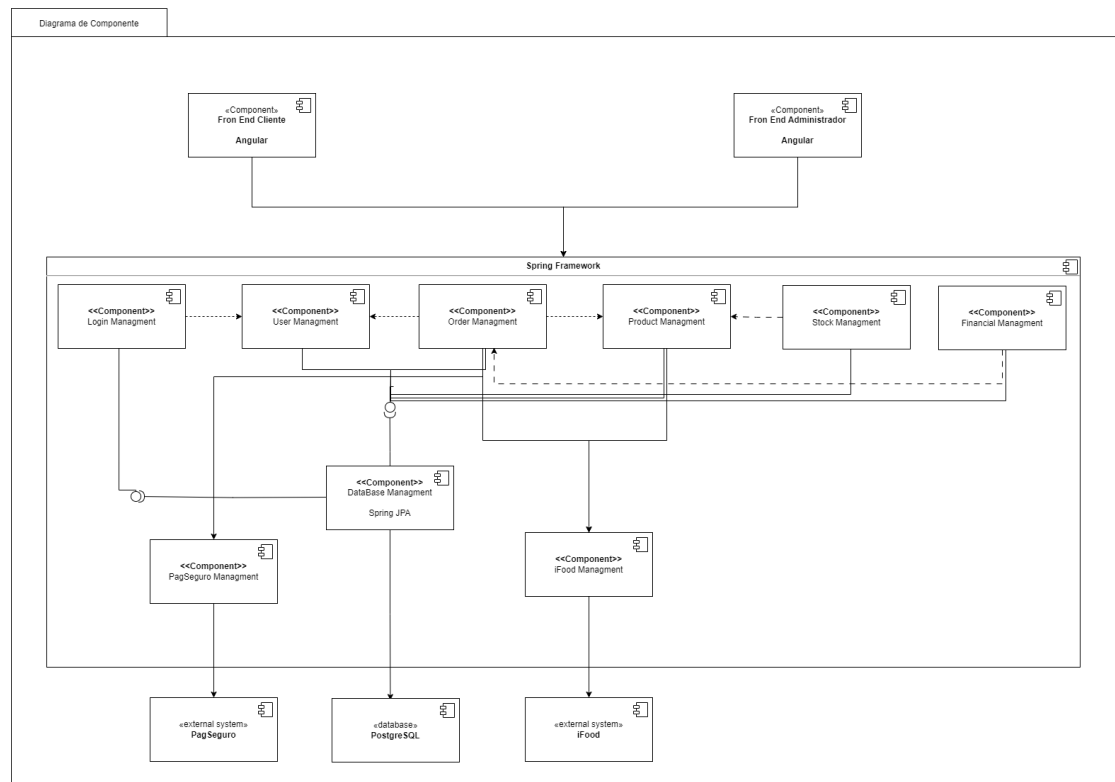


Figure 7.3: Diagrama de Componentes

Imagem resolução maior

Podemos observar no diagrama de componentes que temos os seguintes componentes:

- frontend Cliente: Componente que representa o código feito utilizando framework Angular para interação do cliente com o sistema delivery.
- frontend Administrador: Componente que representa o código feito utilizando framework Angular para interação do atendente/administrador com o sistema delivery. Este será somente um front end e teremos os níveis de acesso.
- Login Managment: Componente que representa o código que será responsável pelo login do usuário.
- User Managment: Componente que representa o código que será responsável pelo controle dos usuários do sistema.
- Order Managment: Componente que representa o código que será responsável pelo controle das vendas do sistema.
- Product Managment: Componente que representa o código que será responsável pelo controle dos produtos do sistema.

- Stock Managment: Componente que representa o código que será responsável pelo controle de estoque do sistema.
- Financial Managment: Componente que representa o código que será responsável pelo controle das informações financeiras do sistema.
- Database Managment: Componente que representa o código que será responsável pelo controle de banco de dados.
- iFood Managment: Componente que representa o código que será responsável pelo controle da integração com o ifood do sistema.
- PagSeguro Managment: Componente que representa o código que será responsável pelo controle da integração com o pagseguro do sistema.

8. Vídeo Apresentação 1

Vídeo Apresentação 1

9. Avaliação Arquitetural ATAM

A avaliação arquitetural ATAM (Architecture Tradeoff Analysis Method) tem como objetivo elicitar e refinar os requisitos de atributos de qualidade e as decisões de arquitetura no projeto de software.[4]

9.1. Análise das abordagens arquiteturais

Abaixo listamos as abordagens arquiteturais, seus cenários, importância e complexidade(definidas em B - baixa, M - média e A - Alta).

9.1.1. Manutenibilidade

Cenário 1: A arquitetura deve considerar a facilidade de manutenção do sistema ao longo do tempo, buscando criar uma estrutura que permita a evolução do sistema, isto pode ser atingido utilizando de padrões de projetos. Por exemplo na utilização do padrão MVC, separação das responsabilidades.

Importância: A - **Complexidade:** A

9.1.2. Usabilidade

Cenário 2: A arquitetura deve considerar a usabilidade do sistema, facilitando a utilização do mesmo por parte dos usuários, além de permitir uma fácil integração de novos usuários ao sistema. Para isto deve se adotar padrões de interface, tanto em dispositivos móveis como em telas maiores como monitores, utilizar de interfaces intuitivas e que o padrão seja mantido entre as telas do sistema.

Importância: A - **Complexidade:** M

9.1.3. Segurança

Cenário 3: A arquitetura deve ser pensada em tornar as informações protegidas e garantir acesso seguro ao sistema. Um usuário não autorizado não deve conseguir acessar informações confidenciais do sistema. Para isso pode se adotar criptografia nas senhas e controle de autenticação e autorização no sistema.

Importância: A - **Complexidade:** A

9.1.4. Interoperabilidade

Cenário 4: A arquitetura deve considerar a capacidade do sistema em se comunicar com outros sistemas de software em outras tecnologias, para isso deve-se adotar padrões de comunicação que são amplamente aceitos como RESTFUL, SOAP e na utilização de formatos de dados amplamente utilizados como JSON e XML.

Importância: A - **Complexidade:** M

9.1.5. Desempenho

Cenário 5: A arquitetura deve considerar a capacidade do sistema em se manter estável e atingir requisitos de tempo de respostas quando de alta carga de trabalho, como horários de pico. Para isso a arquitetura deve ser pensada de modo a oferecer o dimensionamento de recursos, seja horizontal ou vertical e caches para informações estáticas.

Importância: M - Complexidade: A

9.2. Cenários

Abaixo listamos os cenários para serem efetuados testes na aplicação, os mesmos demonstram como serão satisfeitos os atributos de qualidade do software(requisitos não funcionais).

9.2.1. Cenário 1

Manutenabilidade: Quando da necessidade de se atualizar um requisito de negócio do sistema, é necessário a modificação em alguma camada do mesmo, portanto é fundamental o software ser desenvolvido em camadas e com baixo acoplamento entre as mesmas, permitindo assim uma fácil manutenção e evolução do software. O atendimento deste cenário atende o RNF-1 - Desenvolvimento em Camadas.

9.2.2. Cenário 2

Usabilidade: Ao acessar o software de diferentes dispositivos o sistema deverá se adaptar ao tamanho das telas, além de ter um padrão visual entre telas e dispositivos de modo a facilitar o aprendizado do mesmo. O atendimento deste cenário atende o RNF-2 - Padrão de Interface.

9.2.3. Cenário 3

Segurança: O sistema deverá criptografar dados sensíveis do software, como a senha de acesso do usuário. O atendimento deste cenário atende o RNF-3 - Banco de dados.

9.2.4. Cenário 4

Segurança: Ao tentar acessar um recurso privado sem estar logado o sistema deverá exibir uma mensagem de não autorizado. O atendimento deste cenário atende o RNF-4 - Autenticação/Autorização.

9.2.5. Cenário 5

Interoperabilidade: Ao efetuar um pagamento através do software, o sistema deverá realizar uma comunicação com algum meio de pagamento para permitir a cobrança do cliente. O atendimento deste cenário atende o RNF-5 - Comunicação com sistema de pagamento.

9.2.6. Cenário 6

Desempenho: Ao efetuar login no sistema, o mesmo deverá validar e permitir ou negar o acesso em até 5 segundos, permitindo assim uma experiência fluida para o

usuário. O atendimento deste cenário atende o RNF-6 - Acesso após Autenticação.

9.3. Evidências da Avaliação de Cenários

9.3.1. Cenário 1

Atributo de Qualidade	Manutenabilidade
Requisito de Qualidade	A arquitetura deve considerar a facilidade de manutenção do sistema ao longo do tempo, buscando criar uma estrutura que permita a evolução do sistema.
Preocupação:	
O sistema deverá ser feito utilizando padrão MVC, sendo assim desenvolvido em camadas de modo a facilitar a manutenabilidade do mesmo.	
Cenário(s):	
Cenário 1	
Ambiente:	
Operação normal	
Estímulo:	
Ao sofrer uma mudança de requisitos, como adição de novos campos.	
Mecanismo	
Utilização de classes seguindo o modelo MVC e boas práticas de codificação de modo a separar as responsabilidades de cada classe	
Medida de Resposta	
Tempo gasto e facilidade para implementar mudanças de requisito	
Consideração sobre a arquitetura:	
Riscos	A adoção do padrão MVC pode resultar em uma arquitetura inicial mais complexa, exigindo uma correta implementação no decorrer do desenvolvimento.
Pontos de Sensibilidade	Caso o sistema seja frequentemente alterado poderá vir a ser necessário uma revisão mais próxima e constante na estrutura da arquitetura para continuar garantindo a manutenabilidade.
Tradeoff	No início do desenvolvimento poderá ocorrer um aumento na complexidade, o que será recompensado pela facilidade de manutenção a longo prazo.

9.3.1.1. Evidência do Cenário 1

O desenvolvimento é no padrão arquitetural MVC, para isso utilizamos a separação das classes seguindo as regras do padrão, colocando uma camada de controller para direcionar o fluxo, uma camada de modelo para representar as tabelas do banco, para isso utilizamos também o padrão repository e uma camada de serviço que contém as regras de negócio.

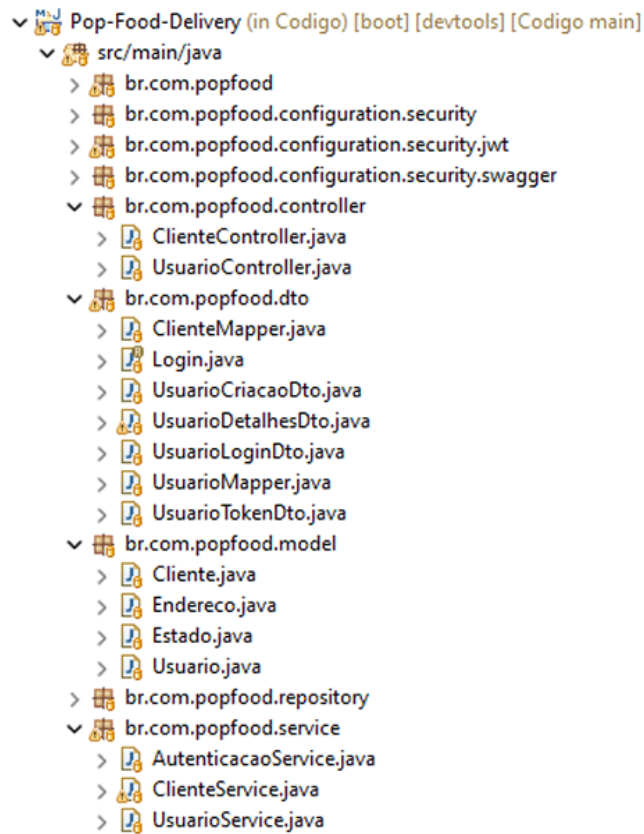


Figure 9.1: Padrão MVC

9.3.2. Cenário 2

Atributo de Qualidade	Usabilidade
Requisito de Qualidade	O software deve ter uma boa usabilidade mantendo o padrão e a adaptabilidade em diferentes telas.
Preocupação:	
Ao ser acessado de diferentes dispositivos o padrão deverá ser mantido e adaptado a resoluções diferentes.	
Cenário(s):	
Cenário 2	
Ambiente:	
Operação normal	
Estímulo:	
Ao acessar diferentes telas, o padrão deverá ser mantido e também adaptado a diferentes resoluções	
Mecanismo	
Utilização de templates que permitam padronizar as telas e mantém uma boa usabilidade entre diferentes resoluções de tela.	
Medida de Resposta	
Tempo gasto de aprendizado na utilização do sistema	
Consideração sobre a arquitetura:	
Riscos	Caso tenha falhas na estrutura dos templates o sistema poderá ter inconsistências visuais, principalmente se utilizar de dispositivos que sejam muito fora do padrão ou mais antigos.
Pontos de Sensibilidade	Dispositivos muito antigos ou com resoluções muito altas que fujam dos padrões existentes.
Tradeoff	O desenvolvimento dos templates pode vir a exigir um esforço adicional durante o desenvolvimento, isso será compensado no futuro com a facilidade de utilização do sistema e também para implementar novas funcionalidades, que seguirão um padrão já desenvolvido.

9.3.2.1. Evidência do Cenário 2

No quesito de usabilidade optou-se por utilizar no front end o Angular Material, ele é baseado no Material Design e fornece diversos componentes prontos e formas de se utilizar para se manter um padrão, tanto no quesito de aparência quanto na usabilidade do sistema.

Através da utilização do mesmo os componentes se ajustarão a diferentes resoluções e dispositivos, além de se manterem padronizados entre uma tela e outra de modo a facilitar o uso e aprendizado do sistema.

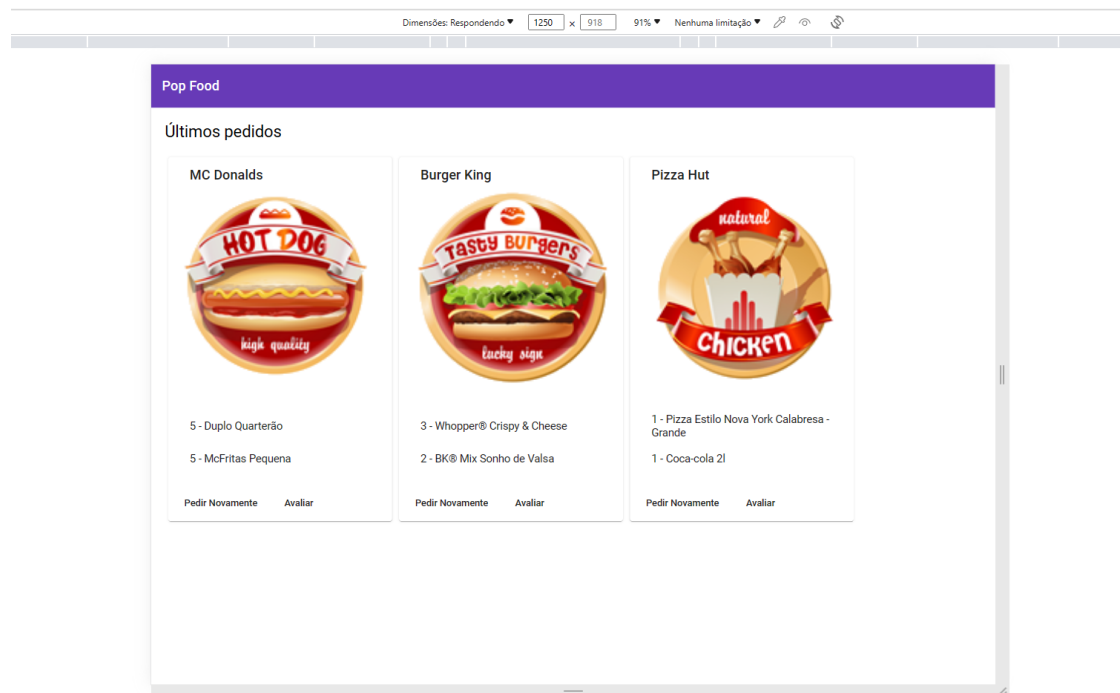


Figure 9.2: Utilização do Angular Material - Visualização Navegador

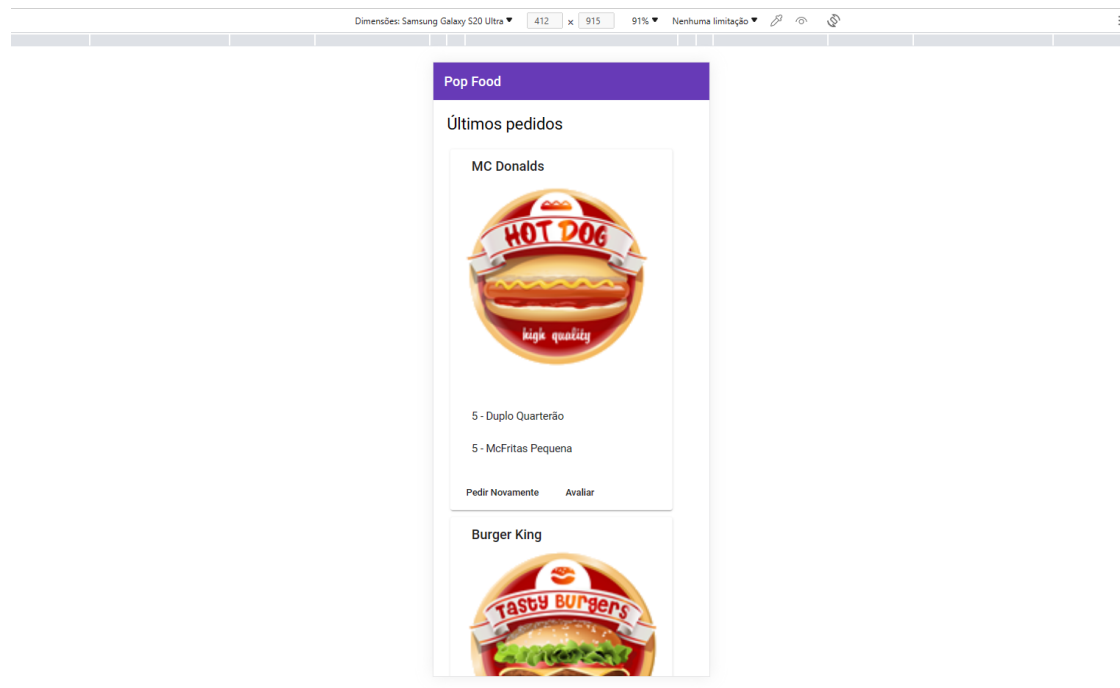


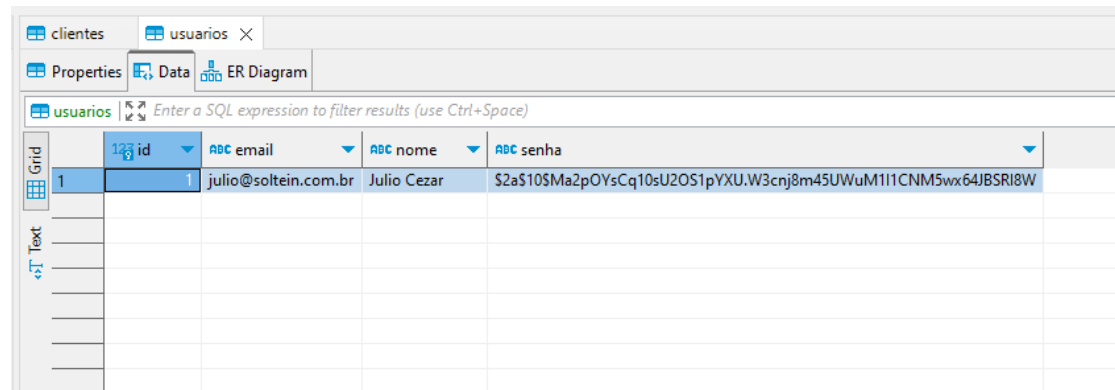
Figure 9.3: Utilização do Angular Material - Visualização Celular

9.3.3. Cenário 3

Atributo de Qualidade	Segurança
Requisito de Qualidade	O sistema deverá criptografar dados sensíveis do software, como a senha de acesso do usuário.
Preocupação:	
Proteger informações sensíveis por meio de criptografia para evitar acesso não autorizado aos dados.	
Cenário(s):	
Cenário 3	
Ambiente:	
Operação normal	
Estímulo:	
Tentativa não autorizada de acesso a senha do usuário	
Mecanismo	
Utilização de algoritmos de criptografia para proteger a senha do usuário.	
Medida de Resposta	
Dificultar descobrir a senha do usuário.	
Considreação sobre a arquitetura:	
Riscos	Não basta somente criptografar a senha, pois poderão pegar a criptografia e tentar comparar com senhas comuns e então encontrar a senha do usuário, portanto, torna-se necessário acrescentar alguma informação a senha, como um salt(técnica que consiste em adicionar alguma informação secreta a senha, e quando o usuário digitar a senha o sistema acrescenta a informação antes da comparação) que combinado a senha gere a criptografia.
Pontos de Sensibilidade	NA 23
Tradeoff	NA

9.3.3.1. Evidência do Cenário 3

No quesito de segurança optou por criptografar os dados sensíveis, a princípio foi identificado a senha como uma informação sensível, sendo necessário avaliar as demais informações para definir o que deve ser criptografado.



Grid	123 id	ABC email	ABC nome	ABC senha
1	1	julio@soltein.com.br	Julio Cezar	\$2a\$10\$Ma2pOYsCq10sU2OS1pYXU.W3cnj8m45UWuM111CNM5wx64JBSRl8W

Figure 9.4: Segurança criptografia senha

9.3.4. Cenário 4

Atributo de Qualidade	Segurança
Requisito de Qualidade	Ao tentar acessar um recurso privado sem estar logado, o sistema deverá exibir uma mensagem de não autorizado.
Preocupação:	
Garantir que somente usuários autorizados tenham acesso aos recursos privados.	
Cenário(s):	
Cenário 4	
Ambiente:	
Operação normal	
Estímulo:	
Tentativa de acesso a recurso privado sem estar logado	
Mecanismo	
Implementação de um controle de autenticação e autorização que verifica se o usuário está logado e possui as permissões necessárias para acessar o recurso.	
Medida de Resposta	
Não autorizar acesso a recursos privados do software.	
Consideração sobre a arquitetura:	
Riscos	Implementação de um controle de autenticação e autorização que verifica se o usuário está logado e possui as permissões necessárias para acessar o recurso.
Pontos de Sensibilidade	NA
Tradeoff	NA

9.3.4.1. Evidência do Cenário 4

Neste quesito foi desenvolvido um módulo de autenticação baseado em JWT(JSON Web Token), pois ele é utilizado amplamente em autenticação de sistemas, o back end ao receber os dados de acesso gera um token temporário que permite ao usuário acessar áreas privadas do sistema.

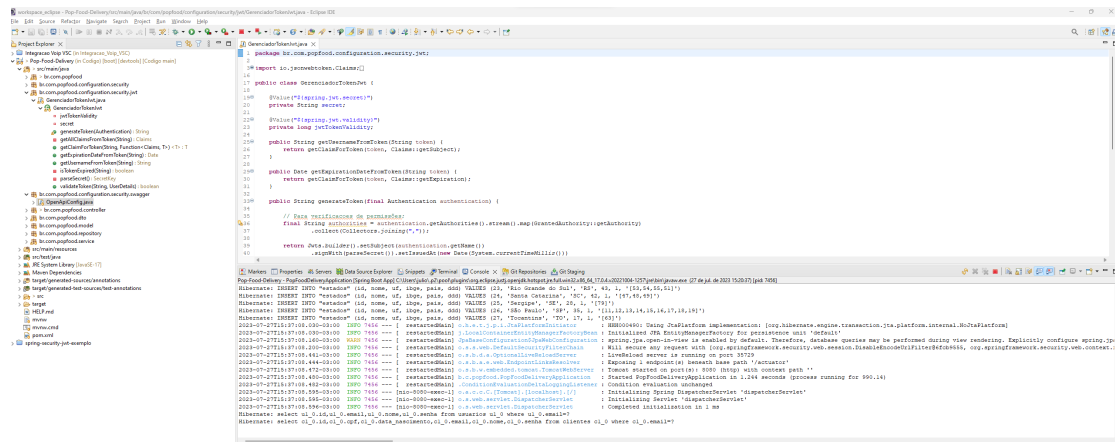


Figure 9.5: Parte do código responsável pelo JWT.

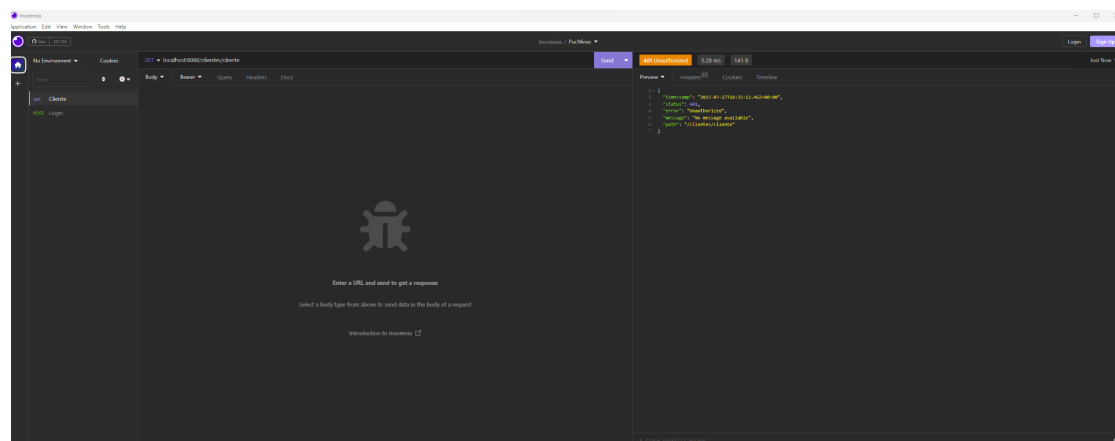


Figure 9.6: Tentativa de acesso sem estar logado.

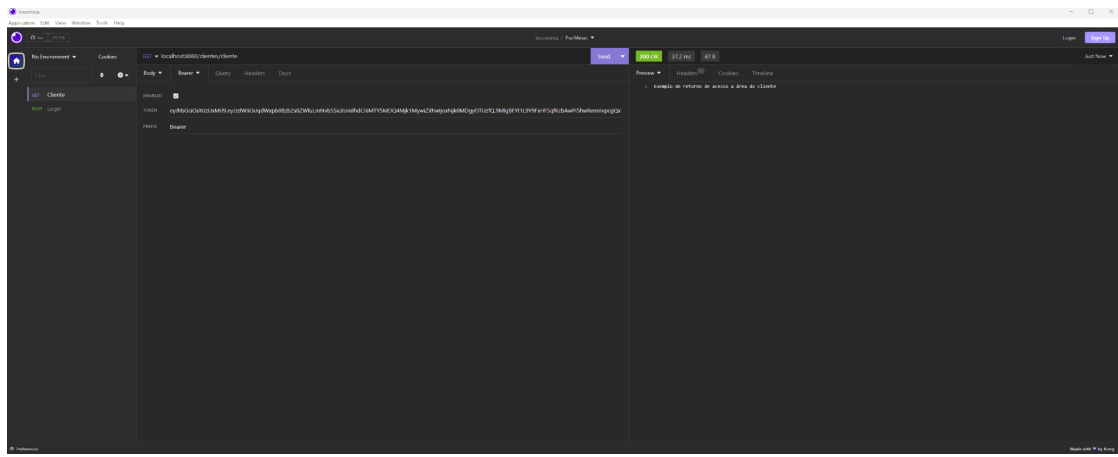


Figure 9.7: Tentativa de acesso logado ao sistema.

9.3.5. Cenário 5

Atributo de Qualidade	Interoperabilidade
Requisito de Qualidade	O sistema deve considerar a capacidade de se comunicar com outros sistemas de software em outras tecnologias, adotando padrões de comunicação amplamente aceitos, como RESTful, utilizando formatos de dados amplamente utilizados como JSON.
Preocupação:	
Garantir que o sistema possa integrar-se com outros sistemas de software, utilizando padrões amplamente adotados no mercado.	
Cenário(s):	
Cenário 5	
Ambiente:	
Operação normal	
Estímulo:	
Necessidade do cliente efetuar um pagamento online do pedido.	
Mecanismo	
Utilização de protocolos de comunicação amplamente aceitos, como RESTful, para integrar com o meio de pagamento externo.	
Medida de Resposta	
Sucesso na comunicação e integração com o meio de pagamento externo, evidenciado pela realização bem-sucedida de transações e processamentos de pagamento.	
Consideração sobre a arquitetura:	
Riscos	Os sistemas podem estar vulneráveis a ameaças de segurança, levando ao comprometimento de dados e roubos de informações sensíveis, principalmente por se tratar de dados de pagamento. Portanto é necessário verificar as diversas possibilidades, como a tela em que o usuário digita as informações de cartão já ser uma tela da operadora, dessa forma não teremos a responsabilidade sobre as informações.
Pontos de Sensibilidade	Um dos principais pontos de sensibilidade é o armazenamento das informações de cartões, deve-se decidir se essas informações ficarão armazenadas ou se encontrará uma operadora em que os dados sejam digitados em seu sistema. Caso opte por armazenar, é necessário uma evolução maior da arquitetura pois aí iremos tratar de implementações como certificação PCI o que aumenta a complexidade do sistema e infra-estrutura.
Tradeoff	É importante equilibrar a segurança com os desejos de funcionalidades do sistema, é de grande desejo ter a opção de pagamento online, mas devemos ter uma visão sobre a segurança desta funcionalidade e o riscos e prejuízos em casos de falha. Portanto é necessário encontrar um meio termo como informado anteriormente, uma plataforma que já tenha esse nível de segurança garantido e processamos somente enviar dados não sensíveis e esta plataforma cuidar das informações mais sensíveis como dados de cartão.

9.3.5.1. Evidência do Cenário 5

Foi desenvolvido um POC para simular a integração com a CIELO, utilizamos da documentação deles e desenvolvemos um exemplo fictício com o objetivo de testar a tecnologia utilizada, no caso RestTemplet do Spring.

Neste POC criamos uma chamada ao end point da CIELO passando as informações via JSON e obtemos o retorno de sucesso da geração do pedido de compra nos sistemas deles.

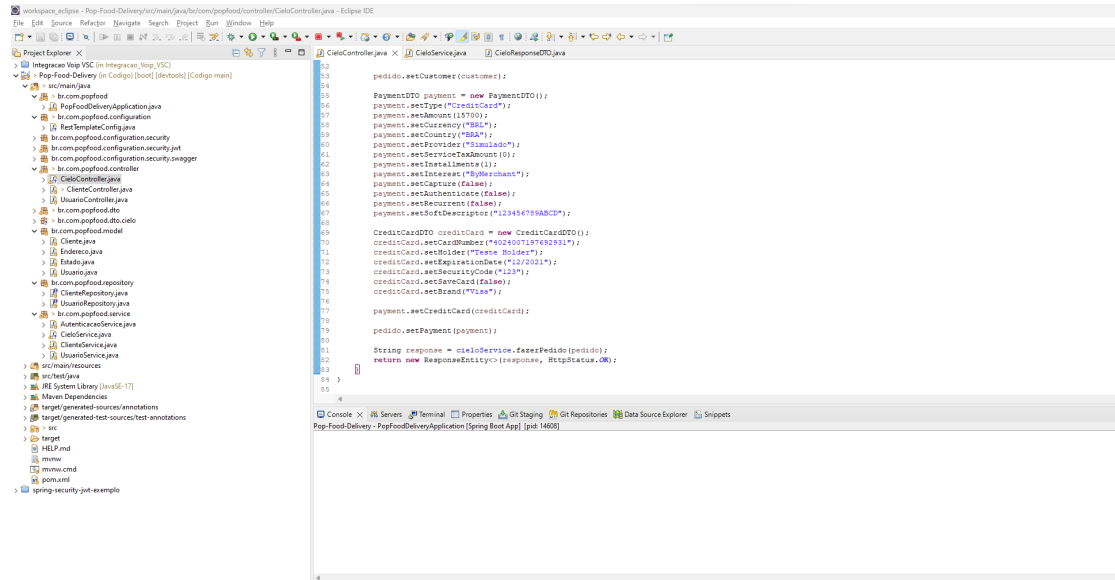


Figure 9.8: Código fonte controller POC - CIELO.

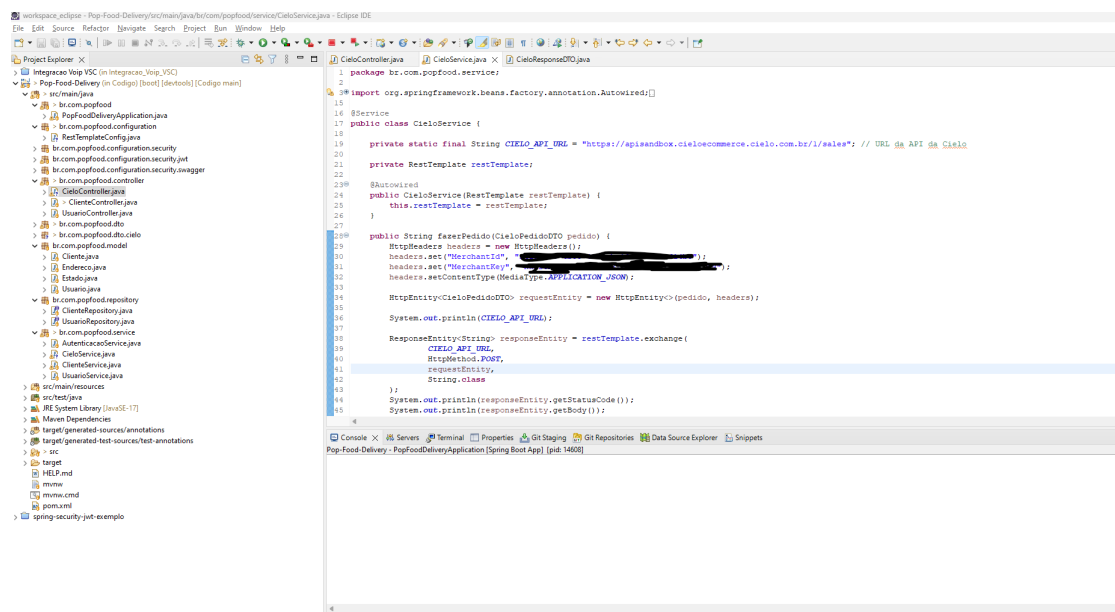


Figure 9.9: Código fonte service POC - CIELO.

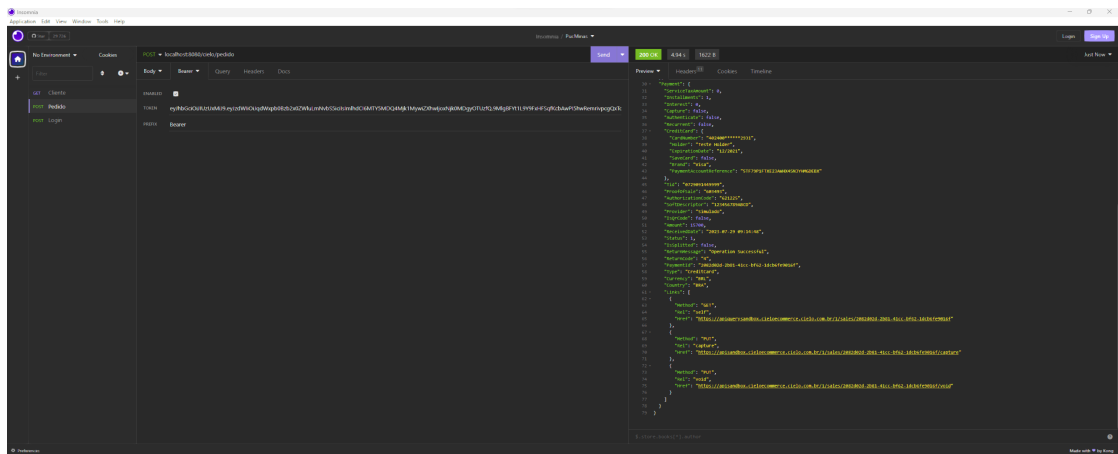


Figure 9.10: Teste de execução com INSOMINIA POC - CIELO.

9.3.6. Cenário 6

Atributo de Qualidade	Desempenho
Requisito de Qualidade	O sistema deve ser capaz de exibir sua tela inicial após a inserção dos dados de acesso em menos de 5 segundos.
Preocupação:	
Garantir que o sistema ofereça uma experiência fluida para o usuário ao efetuar o login, proporcionando tempo de resposta rápido e eficiente.	
Cenário(s):	
Cenário 6	
Ambiente:	
Operação normal	
Estímulo:	
Efetuar login no sistema.	
Mecanismo	
Implementação de estratégia de autenticação otimizada baseada em usuário e senha. Utilização de JWT para geração de token a ser validado nas rotas privadas da aplicação.	
Medida de Resposta	
Tempo de resposta na geração do TOKEN de acesso as demais telas do sistema.	
Consideração sobre a arquitetura:	
Riscos	Uma sobrecarga do servidor em caso de grande número de acessos simultâneos poderá acarretar em degradação do desempenho do sistema como um todo.
Pontos de Sensibilidade	NA
Tradeoff	NA

9.3.6.1. Evidência do Cenário 6

Foi efetuada a simulação de acesso ao sistema utilizando o INSOMINIA, nele foi verificado que o sistema em operação normal o acesso é muito rápido, ficando em 82.4 ms. É claro que no uso diário do sistema deverá ser acompanhado o crescimento do mesmo e a quantidade de usuários simultâneos, e caso necessário a infra-estrutura deverá crescer vertical e horizontalmente para atender a demanda.

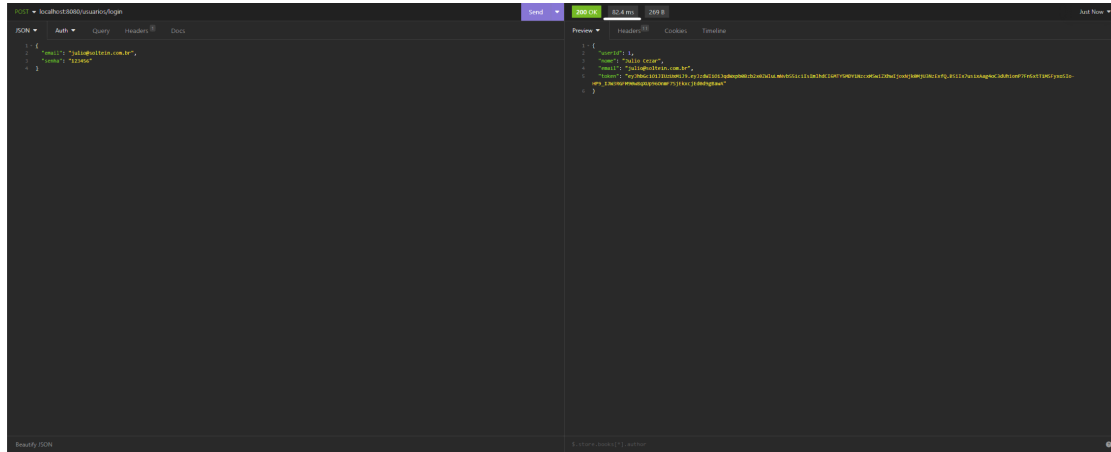


Figure 9.11: Tempo de resposta para acesso ao sistema. 82.4ms.

10. Avaliação crítica dos resultados

De acordo com a proposta do projeto, podemos verificar que os requisitos não funcionais foram atendidos e validados com sucesso, mas há margem para melhorias como por exemplo em relação a integração com operadora de pagamento online, deve-se focar na parte de segurança, fazendo um estudo mais aplicado nas opções de mercado antes de se optar por alguma.

Ponto Avaliado	Descrição
Arquitetura MVC	A utilização do padrão MVC permitiu que as responsabilidades do sistema fossem separadas de forma clara, isto irá facilitar a manutenção e evolução do mesmo. No entanto poderão surgir desafios em relação a manter a organização em camadas na medida que a equipe crescer, exigindo uma maior coordenação do desenvolvimento.
Usabilidade	Ao utilizar o Material Angular mantemos a padronização dos componentes de interface, e utilizamos recursos já utilizados em diversos aplicativos de software, isto permite manter uma facilidade de aprendizado para o usuário além de compatibilidade com diferentes resoluções. Como no desenvolvimento em camadas, a utilização do Material Angular requer que a equipe mantenha-se no propósito de utilizar o mesmo, sendo necessário novamente uma boa coordenação de equipe.
Segurança	No quesito de segurança garantimos a proteção de dados sensíveis através de criptografia e controle de acesso adequado. Mas ainda é necessário se atentar a outros quesitos como na questão de pagamentos online no que se refere a dados de cartão e também ao armazenamento do token de acesso.
Interoperabilidade	Ao utilizarmos o padrão RESTful que é amplamente aceito nos permitirá integra com a grande maioria de sistemas. Caso venha a ser necessário se conectar a sistemas que utilizam padrão SOAP, a tecnologia Spring fornece facilitadores, mas mesmo nessa condição a manutenção de diferentes padrões poderá ser um complicador para a equipe de desenvolvimento.
Desempenho	Ao atender o requisito de desempenho tornamos a experiência para o usuário mais fluida, entretanto é necessário manter um monitoramento do sistema e avaliar a necessidade futura de crescimento da infra-estrutura tanto vertical como horizontalmente.

11. Conclusão

Com a finalização deste trabalho podemos verificar o quão importante é a parte de definição arquitetural de um sistema. Através da avaliação podemos identificar pontos que requerem uma atenção maior e uma maior cautela ao escolher um fornecedor, como por exemplo no quesito de pagamento online e também no desempenho do sistema que requer tanto uma validação melhor por parte de segurança quanto um monitoramento das necessidades futuras.

A arquitetura adotada, baseada no padrão MVC, mostrou-se uma escolha sólida, proporcionando uma estrutura organizada e modular que facilitará a manutenção e a evolução do sistema. No entanto, enfrentaremos desafios em relação à integração entre as camadas, exigindo cuidados no gerenciamento de dependências.

A decisão de padronizar comunicações com sistemas externos utilizando RESTful e JSON contribuirá para a interoperabilidade do sistema.

Em resumo, a definição cuidadosa da arquitetura, aliada a uma avaliação crítica dos trade-offs, resultou em uma arquitetura robusta e eficiente. A busca por uma arquitetura bem equilibrada é um processo contínuo que impulsiona o sucesso do sistema, atendendo às necessidades dos usuários e permitindo sua evolução de forma sustentável.

12. Repositórios

Abaixo os repositórios utilizados para salvar as informações do projeto integrado.

Código LATEX para geração do artigo.

<https://github.com/soltein/TCC>

Código Back End Java.

https://github.com/soltein/Arquitetura_Software_PucMinas.

Código Front End Angular

<https://github.com/soltein/frontend-arquitetura-pucminas>

13. Vídeo Apresentação

Vídeo de apresentação final do projeto.

<https://youtu.be/hxmr67Iwvps>

Bibliography

- [1] Abrasel. Impacto da covid no setor de alimentação. <https://www.abrasel.com.br/noticias/5242-nota-abrasel-sobre-impacto-do-covid-19-no-setor-de-alimentacao-fora-do-lab.html>, 2023. [Acessado em 23 de janeiro de 2023].
- [2] FGV. O 'boom' das plataformas de delivery no brasil e suas consequências peculiares. <https://portal.fgv.br/artigos/boom-plataformas-delivery-brasil-e-suas-consequencias-peculiares>, 2023. [Acessado em 23 de janeiro de 2023].
- [3] Redação IFB. A transformação digital em bares e restaurantes. <https://foodbizbrasil.com/negocios/a-transformacao-digital-em-bares-e-restaurantes/>, 2023. [Acessado em 05 de fevereiro de 2023].
- [4] Antônio Mendes. *Arquitetura de software*. Campus, Rio de Janeiro, 1. ed. edition, 2002.
- [5] Lucas Muribeca. Delivery de alimentos deve aumentar 7,5 <https://redepara.com.br/Noticia/231697/delivery-de-alimentos-deve-aumentar-7-5-em-2023-no-brasil#:~:text=De%20acordo%20com%20o%20Instituto,restaurante%20Engenho%20Ded%C3%A9%2C%20em%20Bel%C3%A9m>, 2023. [Acessado em 05 de fevereiro de 2023].
- [6] Saipos. Quem paga a taxa ifood? <https://saipos.com/ifood/taxa-ifood>, 2023. [Acessado em 23 de janeiro de 2023].
- [7] Ian Sommerville. *Engenharia de Software*. Pearson Prentice Hall, São Paulo, 9. ed. edition, 2011.