

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
PUC Minas Virtual
Pós-graduação *Lato Sensu* em Engenharia de *Software*

Projeto Integrado

Relatório Técnico

OnClick Delivery Eats

Carlos Augusto Belli

Belo Horizonte
Outubro de 2023.

Projeto Integrado

Sumário

Projeto Integrado.....	2
1. Cronograma de Trabalho.....	3
2. Introdução.....	4
3. Definição Conceitual da Solução.....	5
3.1 Diagrama de Casos de Uso.....	6
3.2 Requisitos Funcionais.....	6
3.3 Requisitos Não-funcionais.....	8
4. Protótipo Navegável do Sistema.....	8
5. Diagrama de Classes de Domínio.....	9
6. Arquitetura da Solução.....	10
6.1 Padrão Arquitetural.....	10
6.2 C4 model - Diagrama de Contexto.....	11
7. Frameworks de Trabalho.....	13
8. Estrutura Base do Front End.....	15
9. Modelo Relacional ou Projeto de Banco de Dados NoSQL.....	19
10. Plano de Testes.....	20
11. Apropriação de Horas no Projeto.....	22
12. Código da Aplicação.....	23
13. Avaliação Retrospectiva.....	24
13.1 Objetivos Estimados.....	24
13.2 Objetivos Alcançados.....	24
13.2 Lições aprendidas.....	25
14. Referências.....	25

1. Cronograma de Trabalho

Datas		Atividade / Tarefa	Produto / Resultado
De	Até		
08 / 05 / 23	10 / 05 / 23	1. Formação dos objetivos do trabalho, apresentação do problema e descrição geral do software	Relatório técnico e descritivos
11 / 05 / 23	13 / 05 / 23	2. Levantamento dos casos de uso e desenvolvimento do diagrama de casos de uso	Diagrama de casos de uso
14 / 05 / 23	16 / 05 / 23	3. Levantamento dos requisitos funcionais e não funcionais do sistema	Requisitos funcionais e não funcionais do sistema
17 / 05 / 23	03 / 06 / 23	4. Desenvolvimento do protótipo navegável e interativo do sistema	Protótipo de interface navegável
04 / 06 / 23	06 / 06 / 23	5. Gravação do vídeo	Vídeo apresentado o protótipo e seus casos de uso
07 / 06 / 23	09 / 06 / 23	6. Desenvolvimento do diagrama de classes de domínio	Diagrama de classes de domínio
12 / 06 / 23	13 / 06 / 23	7. Definição do padrão arquitetural	Padrão do padrão arquitetural da aplicação
14 / 06 / 23	15 / 06 / 23	8. Desenvolvimento do diagrama de contexto	Diagrama de contexto – C4 model
16 / 06 / 23	17 / 06 / 23	9. Escolha dos frameworks de trabalho	Apresentação dos frameworks de trabalho
18 / 06 / 23	20 / 06 / 23	10. Estrutura base do front-end	Apresentação do layout mestre e menu do sistema
21 / 06 / 23	23 / 06 / 23	11. Desenvolvimento do modelo relacional do projeto	Modelo relacional do projeto
24 / 06 / 23	26 / 06 / 23	12. Elaboração do plano de testes	Plano de testes do sistema
27 / 06 / 23	19 / 07 / 23	13. Desenvolvimento do backend do sistema	Código fonte do backend
20 / 07 / 23	04 / 08 / 23	14. Desenvolvimento do front-end do sistema	Código fonte do front-end do sistema
05 / 08 / 23	06 / 08 / 23	15. Avaliação retrospectiva	Dados de aprendizagem adquirida durante o desenvolvimento do projeto
07 / 08 / 23	07 / 08 / 23	16. Objetivos estimados	Descrição dos objetivos estimados
08 / 08 / 23	08 / 08 / 23	17. Objetivos alcançados	Descrição dos objetivos alcançados
09 / 08 / 23	10 / 08 / 23	18. Retrospectiva e lições aprendidas	Descrição das lições aprendidas

2. Introdução

O setor de entrega de produtos vem experimentando um crescimento significativo nos últimos anos, impulsionado principalmente pela evolução das tecnologias e pela mudança nos hábitos de consumo. Nesse contexto, os sistemas de delivery têm se mostrado ferramentas essenciais para atender às demandas dos consumidores de forma eficiente e conveniente, e o setor de comida é um dos mais relevantes nesse cenário.

O sistema de delivery a ser desenvolvido visa atender às necessidades dos usuários, tanto dos clientes que desejem realizar pedidos de refeições de forma rápida e fácil, quanto dos restaurantes e estabelecimentos comerciais que desejem gerenciar seus pedidos de forma eficiente.

Espera-se que o resultado deste trabalho contribua para o avanço no desenvolvimento de sistemas de delivery de comida, oferecendo uma solução eficiente e inovadora para atender às demandas desse setor. Através da aplicação dos princípios da Engenharia de Software, espera-se obter um sistema confiável, escalável e de fácil manutenção, proporcionando benefícios tanto para os usuários quanto para os estabelecimentos comerciais envolvidos no processo de entrega de comida.

O presente trabalho tem como **objetivo geral** apresentar o desenvolvimento de um sistema de delivery voltado para estabelecimentos como restaurantes, pizzarias, lanchonetes e similares, aplicando conceitos e práticas de Engenharia de Software. O sistema proposto visa facilitar a comunicação entre estabelecimentos comerciais e clientes, permitindo a realização de pedidos e a entrega dos produtos de forma ágil e segura.

O estabelecimento poderá cadastrar seu cardápio com todas as variações e configurações, o cliente poderá acessar o cardápio e montar seu pedido, e o estabelecimento receberá esse pedido.

No decorrer deste trabalho serão abordadas diversas etapas do processo de engenharia de software, desde a análise de requisitos até a implementação e testes do sistema. Serão considerados aspectos como usabilidade, escalabilidade e segurança visando garantir uma solução completa e robusta.

Inicialmente a plataforma será desenvolvida apenas web, onde teremos uma única interface, que poderá ser acessada pelo prestador de serviço (estabelecimento) e usuário (cliente), essa interface fornecerá todos os recursos para ambos, e tendo a possibilidade de estender todas as funcionalidades para um app mobile.

Objetivos Específicos:

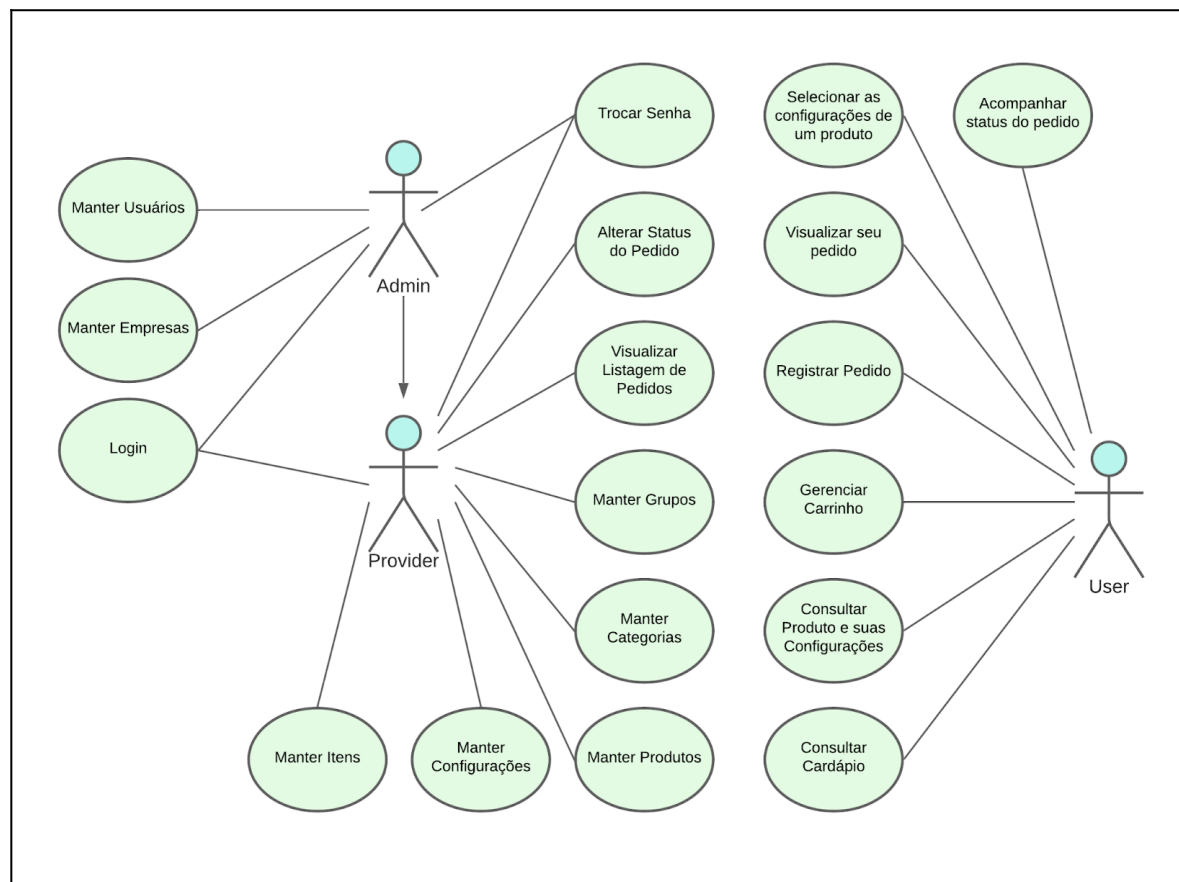
- **Levantamento de Arquitetura:** Será realizado um estudo detalhado para identificar a arquitetura que melhor atenderá às necessidades do projeto. Esse processo envolverá a análise das opções disponíveis, considerando fatores como escalabilidade, desempenho e segurança.
- **Desenvolvimento de Software:** Com base na arquitetura selecionada, procederemos ao desenvolvimento do software. Isso incluirá a criação de um sistema robusto, incorporando um banco de dados eficiente e uma interface de usuário intuitiva. Para isso, serão utilizadas as tecnologias mais adequadas e atualizadas disponíveis no mercado.
- **Implantação em Cloud:** O sistema resultante será implantado em uma plataforma de hospedagem em nuvem, também conhecida como cloud provider. Essa abordagem permitirá flexibilidade, escalabilidade e acessibilidade para os usuários, garantindo que o sistema esteja disponível de forma confiável e segura a partir de qualquer local com acesso à internet.

Essas metas específicas visam garantir que o projeto seja conduzido de forma eficiente, com uma arquitetura bem definida e o uso das melhores tecnologias disponíveis, culminando na implantação bem-sucedida do sistema em uma infraestrutura em nuvem.

3. Definição Conceitual da Solução

O sistema será simples e funcional, um usuário com perfil admin terá acesso para gerenciar empresas e usuários, o usuário com perfil provider terá acesso a gerenciar o cardápio e pedidos, e o usuários comuns poderão acessar o cardápio de um estabelecimento e fazer seus pedidos.

3.1 Diagrama de Casos de Uso



3.2 Requisitos Funcionais

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
Manter estabelecimentos - CRUD de estabelecimentos			
RF01	Um admin deve ser capaz de cadastrar uma nova empresa informando nome e URL	B	A
RF02	Um admin deve ser capaz de listar todos os estabelecimentos	B	M
RF03	Um admin deve ser capaz de alterar os dados de uma empresa	B	M
RF04	Um admin deve ser capaz de remover uma empresa e seus dados relacionados	B	M
Manter usuários - CRUD de usuarios			
RF05	Um admin deve ser capaz de cadastrar um usuário associado a uma empresa, e definir suas permissões	B	A
RF06	Um admin deve ser capaz de listar os usuários associados a uma empresa	B	M
RF07	Um admin deve ser capaz de alterar os dados de um usuário	B	M
RF08	Um admin deve ser capaz de remover um usuário	B	M

Projeto Integrado – Engenharia de *Software* - PMV

Login e senha			
RF09	Um usuário (admin e provider) pode efetuar login no sistema	M	A
RF10	Um usuário (admin e provider) deve ser capaz de trocar sua senha	M	B
Manter cardápio - CRUD de grupos, categorias e produtos			
RF11	Um provider deve ser capaz de consultar o cardápio da empresa ao qual está associado	M	A
RF12	Um provider deve ser capaz de cadastrar um grupo do cardápio	M	A
RF13	Um provider deve ser capaz de alterar um grupo do cardápio	M	A
RF14	Um provider deve ser capaz de remover um grupo do cardápio	M	A
RF15	Um provider deve ser capaz de cadastrar uma categoria associada a um grupo do cardápio	M	A
RF16	Um provider deve ser capaz de alterar uma categoria do cardápio	M	A
RF17	Um provider deve ser capaz de remover uma categoria do cardápio	M	A
RF18	Um provider deve ser capaz de cadastrar um produto associado a uma categoria do cardápio	M	A
RF19	Um provider deve ser capaz de alterar um produto do cardápio	M	A
RF20	Um provider deve ser capaz de remover um produto do cardápio	M	A
Manter configurações - CRUD de configurações e itens			
RF21	Um provider deve ser capaz de consultar as configurações associadas a uma categoria	M	A
RF22	Um provider deve ser capaz de cadastrar uma configuração associada a uma categoria	M	A
RF23	Um provider deve ser capaz de alterar uma configuração	M	A
RF24	Um provider deve ser capaz de remover uma configuração	M	A
RF25	Um provider deve ser capaz de cadastrar um item associado a uma configuração	M	A
RF26	Um provider deve ser capaz de alterar um item	M	A
RF27	Um provider deve ser capaz de remover um item	M	A
Gerenciar pedidos – Listar e alterar status			
RF28	Um provider deve ser capaz de consultar os pedidos do estabelecimento ao qual está associado	A	A
RF29	Um provider deve ser capaz de alterar o status de um pedido	M	A
Realizar pedidos			
RF30	Um usuário comum deve ser capaz de consultar o cardápio de um estabelecimento	M	A
RF31	Um usuário comum deve ser capaz de consultar um produto junto com as configurações e itens associados a categoria do produto	M	A
RF32	Um usuário comum deve ser capaz de selecionar as configurações disponíveis de um produto	M	M

RF33	Um usuário deve ser capaz de inserir um produto no carrinho	M	A
RF34	Um usuário deve ser capaz de alterar a quantidade de um produto no carrinho	M	A
RF35	Um usuário deve ser capaz de excluir um produto do carrinho	M	M
RF36	Um usuário deve ser capaz de realizar um pedido	A	A
RF37	Um usuário deve ser capaz de visualizar seu pedido realizado	B	M
RF38	Um usuário deve ser capaz de acompanhar o status do seu pedido	B	M

* B = Baixa, M = Média, A = Alta.

3.3 *Requisitos Não-funcionais*

ID	Descrição	Prioridade B/M/A
RNF01	O sistema deve apresentar tempo de resposta abaixo de 200 ms no processamento de 95% das operações de consulta.	M
RNF02	O sistema web deve ser responsivo de forma a proporcionar a utilização de qualquer uma de suas funcionalidades em qualquer resolução	A
RNF03	O sistema deve estar disponível em qualquer período em regime 24/7	A
RNF04	O sistema deve garantir a segurança da senha dos usuários criptografando-as ao serem inseridas no banco de dados	A
RNF05	O sistema deve ser hospedado em cloud provider para melhor disponibilidade	B
RNF06	O sistema deve cumprir as exigências de LGPD, dando possibilidade de exclusão dos dados do usuário que solicitar	A

4. *Protótipo Navegável do Sistema*

Para apresentar o funcionamento do projeto será disponibilizado dois links interativos onde será possível visualizar as telas e os casos de usos.

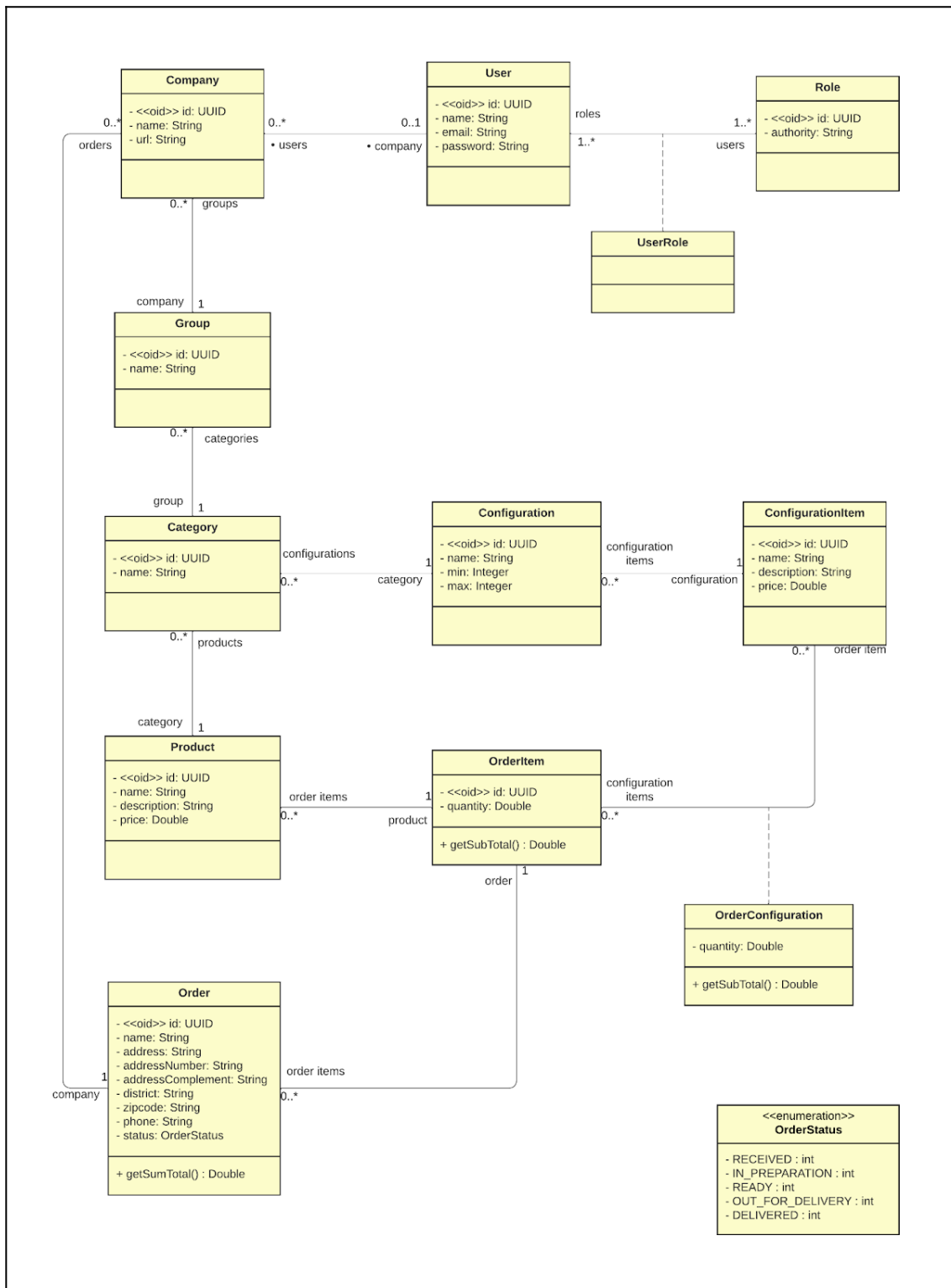
Link do protótipo navegável: <https://guto84.github.io/onclick-app-vite/>

O protótipo navegável trata-se do front-end da aplicação com as requests mockadas, o que torna a experiência mais fluida e real.

Link do vídeo apresentando os casos de uso:

<https://www.loom.com/share/6c543ac3906a46eab2ebdb1be5ce278b>

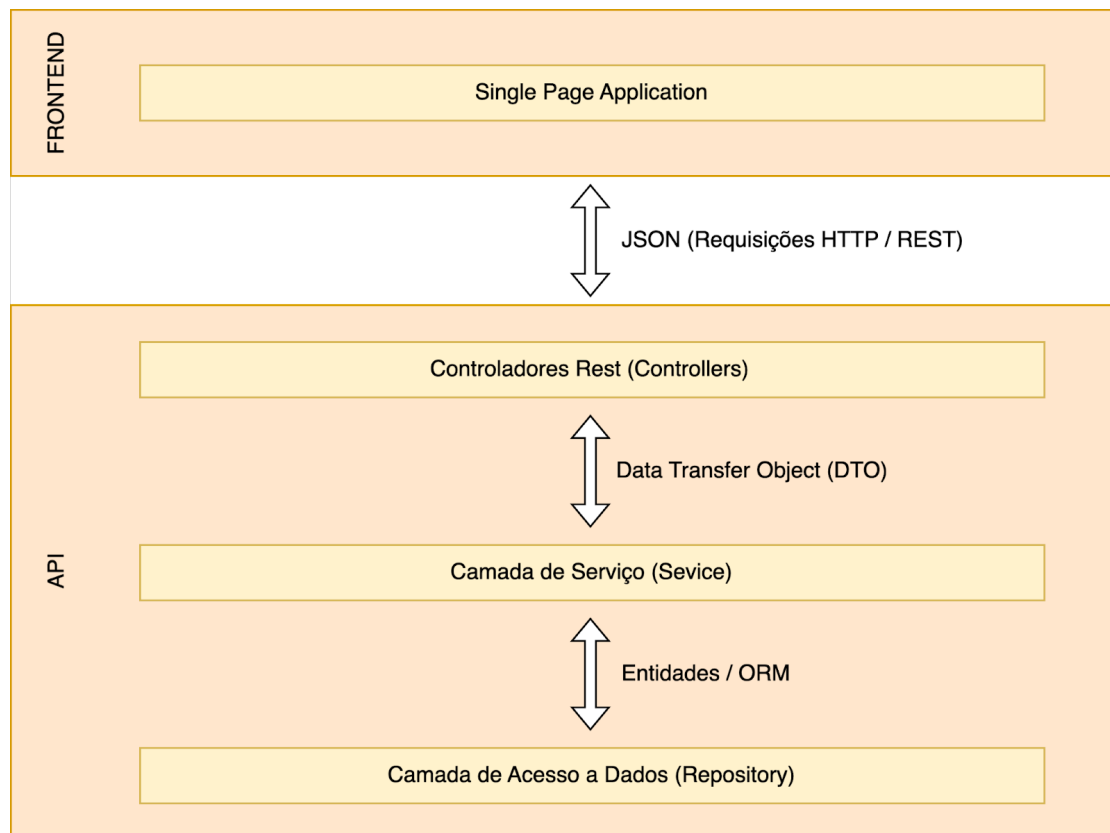
5. Diagrama de Classes de Domínio



6. Arquitetura da Solução

6.1 Padrão Arquitetural

Foi adotado o padrão arquitetural de camadas (MVC) para a organização da aplicação, garantindo que cada camada possui responsabilidades bem definidas. No âmbito desse padrão, os elementos de uma camada específica são permitidos a depender apenas dos componentes da mesma camada ou de camadas inferiores, como é evidenciado no diagrama apresentado a seguir:



Responsabilidades:

- **Controllers:** Os Controllers são encarregados de responder às interações do usuário. No contexto de uma API REST, essas interações correspondem às requisições recebidas. A responsabilidade dos Controllers é interpretar as requisições, direcionar as operações apropriadas da camada de serviços e retornar as respostas adequadas aos clientes.

- **Services:** A camada de Services é responsável por executar as operações de negócios. Cada método dentro da camada de Services deve ter um propósito relacionado ao negócio em questão. Por exemplo, o método "registrarPedido" pode abranger várias etapas, como verificar o estoque, salvar o pedido, atualizar o estoque e enviar notificações por e-mail. Essa camada concentra a lógica de negócios central e garante que as operações sejam realizadas de maneira coesa.
- **Repository:** A camada de Repository é encarregada de realizar operações individuais de acesso ao banco de dados. Ela encapsula as consultas e atualizações de banco de dados, fornecendo uma abstração que permite que as outras camadas operem com os dados sem se preocupar com os detalhes de persistência.
- **DTO (Data Transfer Object):** Os DTOs são objetos simples usados para transferir dados entre diferentes partes da aplicação. Eles fornecem uma maneira de estruturar e organizar os dados que são transmitidos entre as camadas, evitando o vazamento de informações sensíveis e melhorando a eficiência da comunicação.
- **Entidades:** As entidades representam as estruturas de dados que são armazenadas no banco de dados. Elas espelham as tabelas do banco de dados e encapsulam as informações relevantes. As entidades estão associadas às classes que definem os objetos no domínio da aplicação.

Esse design de arquitetura com base no padrão MVC permite uma separação clara de responsabilidades, promovendo a modularidade e facilitando o desenvolvimento, teste e manutenção da aplicação. Cada camada atua de forma independente, colaborando por meio de interfaces bem definidas e minimizando as dependências entre as diferentes partes da aplicação. Isso resulta em um sistema mais organizado, escalável e gerenciável.

6.2 C4 model - Diagrama de Contexto

A seguir, apresentamos um diagrama de contexto simples que ilustra a estrutura geral da aplicação:

Podemos ter diferentes tipos de usuários: Clientes, Prestadores de Serviço e Usuários Comuns. Esses usuários interagem com a Interface do Usuário, acessando-a por meio de um site dedicado.

A Interface do Usuário serve como a camada de interação, proporcionando uma maneira amigável e intuitiva para que os usuários se engajem com a aplicação. Essa interface se conecta à aplicação principal, desenvolvida usando o framework Spring Boot em linguagem Java. A API encapsula a lógica do negócio e oferece funcionalidades inteligentes aos usuários.

Uma parte crucial da Aplicação Spring Boot é a API de Comunicação, que atua como uma ponte entre a Interface do Usuário e os componentes internos da aplicação. Através desta API, os dados são transmitidos de e para a Interface do Usuário, garantindo uma comunicação eficiente e segura.

A camada de persistência é habilitada pela integração com uma base de dados. Todas as informações relevantes à aplicação são armazenadas e gerenciadas nesse banco de dados. Isso permite que os dados sejam retidos de forma confiável ao longo do tempo, garantindo consistência e integridade.

Em resumo, o fluxo de interação começa com os diferentes tipos de usuários acessando a Interface do Usuário por meio de um site. A Interface do Usuário se conecta à Aplicação Spring Boot, que é enriquecida com uma API de Comunicação. Essa API se comunica com a Base de Dados, onde todas as informações da aplicação são persistidas de maneira confiável. Esses elementos juntos formam um sistema coeso e eficaz, oferecendo uma experiência completa e integrada aos usuários.

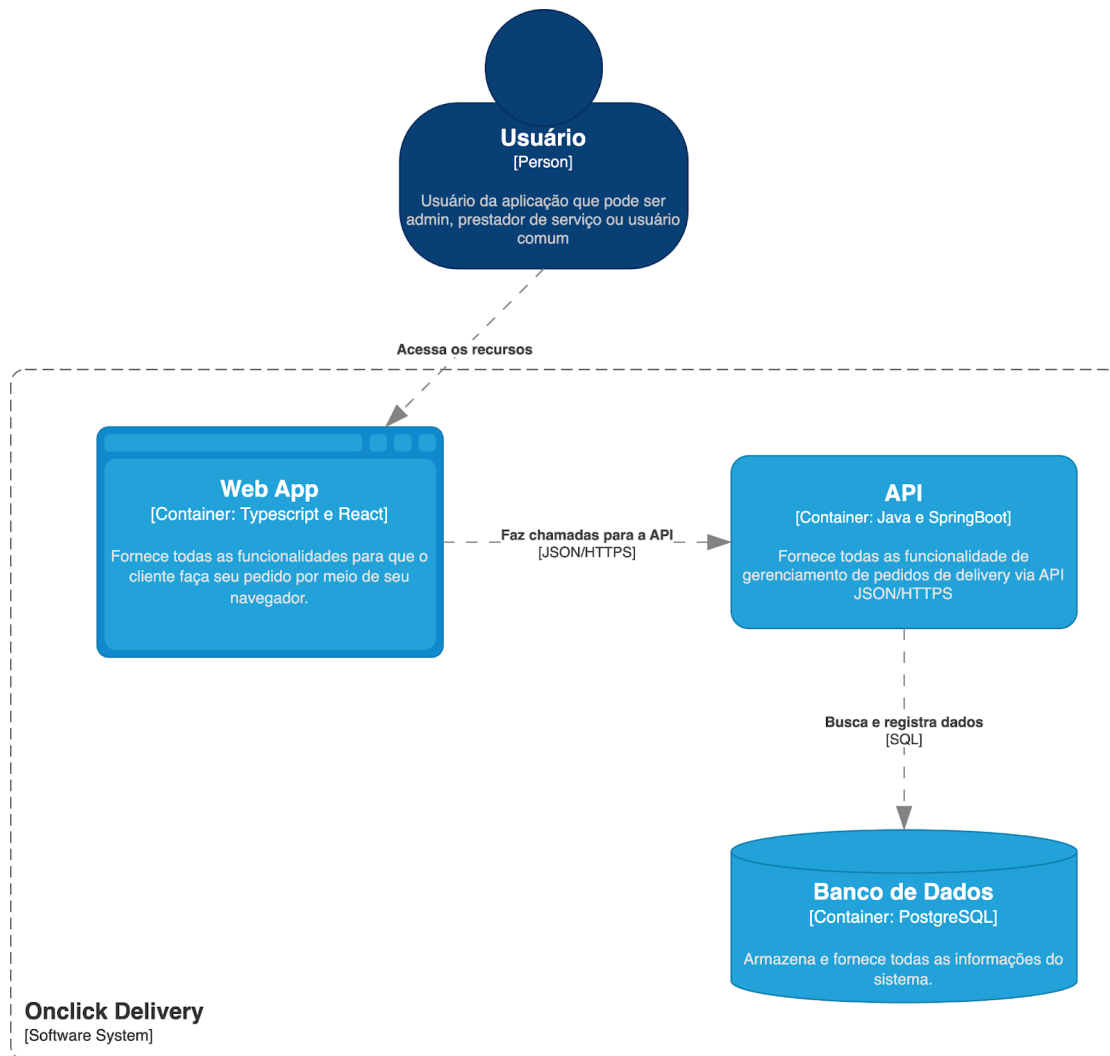


Figura 1 - Visão Geral da Solução

7. Frameworks de Trabalho

Para o **backend** foi utilizado o Spring Boot, um framework Java de código aberto que simplifica o processo de construção de aplicações para produção com uma configuração mínima, fornecendo um conjunto de convenções e padrões. Isso permite que os desenvolvedores foquem mais na escrita de código do aplicativo em vez de arquivos de configuração.

As aplicações Spring Boot podem ser executadas como aplicativos Java autônomos, o que significa que elas não requerem um servidor de aplicativos separado.

O Spring Boot oferece suporte ao Hibernate, um **ORM** amplamente utilizado que oferece recursos poderosos para mapear objetos Java para tabelas de banco de dados e vice-versa. O Spring Boot fornece integração direta com o Hibernate, permitindo que você use suas funcionalidades avançadas de forma fácil e configurável.

Juntamente com o Hibernate foi utilizado o Spring JPA, uma camada de abstração construída sobre o Hibernate que simplifica ainda mais o trabalho com banco de dados. Ele fornece um modelo de programação mais simples, reduzindo a quantidade de código boilerplate necessário para realizar operações de persistência. Além disso, o Spring Data JPA permite que você crie consultas personalizadas usando métodos de nomenclatura ou consultas baseadas em anotações.

Para o **banco de dados** foi escolhido o PostgreSQL, um sistema de gerenciamento de banco de dados relacional (RDBMS) de código aberto, que é amplamente reconhecido por sua robustez, escalabilidade e conformidade com os padrões SQL.

Para o **frontend** do sistema foi utilizado React com Typescript e Vite.

O React é uma biblioteca JavaScript de código aberto desenvolvida pelo Facebook. Ela é usada para criar interfaces de usuário (UI) interativas e reativas para aplicações web. O React permite que os desenvolvedores construam componentes reutilizáveis que representam partes específicas da interface do usuário, facilitando a criação de interfaces complexas e dinâmicas.

TypeScript é uma linguagem de programação desenvolvida pela Microsoft que estende o JavaScript adicionando recursos de tipagem estática. Ela é projetada para ser uma alternativa mais robusta e segura ao JavaScript padrão, especialmente para projetos de grande escala. O TypeScript compila para JavaScript, o que significa que o código TypeScript é transformado em código JavaScript antes de ser executado nos navegadores ou ambientes Node.js.

Vite.js é uma ferramenta de construção de aplicações web rápida e moderna, especialmente projetada para desenvolvimento eficiente e interativo, se destaca por sua velocidade e capacidade de oferecer um ambiente de desenvolvimento rápido e responsivo.

8. Estrutura Base do Front End

OnClick Delivery Eats







Empresas / Usuários

Empresa

NOME	URL
Lanchonete	lanchonete

Usuários

CADASTRAR

NOME	EMAIL	AÇÕES
Luke Skywalker	luke@starwars.com	 
Jimmy Page	page@lanchonete.com	 
Robert Plant	plant@lanchonete.com	 

Página de cadastro de usuários vinculados a uma empresa (Admin)

OnClick Delivery Eats

Cardápio







Cardápio

GRUPO







AÇÕES

^ Lanches

Lanches com Frango

NOME	DESCRIÇÃO	PREÇO	AÇÕES
Frango	Frango, Queijo e Tomate	R\$ 17,00	 
Frango-Salada	Frango, Queijo, Tomate e Alface	R\$ 18,00	 
Frango-Bacon	Frango, Queijo, Tomate e Bacon	R\$ 20,00	 

Lanches com Hamburguer

NOME	DESCRIÇÃO	PREÇO	AÇÕES
X-Bacon	Hamburguer, Queijo, Tomate e Bacon	R\$ 19,00	 
X-Burguer	Hamburguer, Queijo e Tomate	R\$ 16,00	 
X-Salada	Hamburguer, Queijo, Tomate e Alface	R\$ 17,00	 

Porções

Bebidas

Página de cadastro de cardápio (Provider)

OnClick Delivery Eats

OnClick Delivery Eats					☰
Pedidos					
Pedidos					
NOME	TELEFONE	DATA / HORA	STATUS	AÇÕES	
Luna Micaela Medina de Salgado	19995671234	01/09/23 - 20:46	Recebido		
Berenice Danielle Brito Góis	19995671234	01/09/23 - 20:46	Recebido		
Samuel Aécio Duarte Gomes	19995671234	01/09/23 - 20:46	Recebido		
Luciano Inácio Lira Lutero	19995671234	01/09/23 - 20:46	Recebido		
Giovane Leal	19995671234	01/09/23 - 20:46	Recebido		
Valentina Carrara Lira	19995671234	01/09/23 - 20:46	Recebido		
Maria Martines Rocha	19995671234	01/09/23 - 20:46	Recebido		
Pedro Dominato Gomes	19995671234	01/09/23 - 20:46	Recebido		
Giovanni Beltrão de Alencar	19995671234	01/09/23 - 20:46	Recebido		
Cirilo Matos de Cabral	19995671234	01/09/23 - 20:46	Recebido		
< 1 2 >					

Página de pedidos (Provider)

OnClick Delivery Eats			
Lanchonete			
▼ Bebidas			
▼ Porções			
^ Lanches			
Lanches com Frango			
NOME	DESCRIÇÃO	PREÇO	
Frango	Frango, Queijo e Tomate	R\$ 17,00	
Frango-Salada	Frango, Queijo, Tomate e Alface	R\$ 18,00	
Frango-Bacon	Frango, Queijo, Tomate e Bacon	R\$ 20,00	
Lanches com Hamburger			
NOME	DESCRIÇÃO	PREÇO	
X-Bacon	Hamburger, Queijo, Tomate e Bacon	R\$ 19,00	
X-Burguer	Hamburger, Queijo e Tomate	R\$ 16,00	
X-Salada	Hamburger, Queijo, Tomate e Alface	R\$ 17,00	

Página de Cardápio (Usuário)

OnClick Delivery Eats

X-Bacon

Hamburguer, Queijo, Tomate e Bacon

R\$ 19,00

Escolha seu Pão

Selecione 1 item

OBRIGATÓRIO

☐ Pão de Hamburguer

R\$ 0,00

☐ Pão Australiano

R\$ 5,00

☐ Pão Francês

R\$ 0,00

Adicionais

Selecione até 3 itens

☐ Alface

R\$ 1,00

☐ Maionese Caseira

50 gramas

R\$ 3,00

☐ Presunto

50 gramas

R\$ 2,00

☐ Tomate

R\$ 1,00

☐ Milho

R\$ 1,00

☐ Queijo

50 gramas

R\$ 2,00

ADICIONAR

Página de configuração do produto para pedido (Usuário)

OnClick Delivery Eats

← Fazer pedido

X-Burguer

Hamburguer, Queijo e Tomate

Pão de Hamburguer

R\$ 16,00

Coca-Cola

Lata 350 ml

R\$ 5,00

Total do Pedido: R\$ 20,00

Seus Dados

Nome

Logradouro

Numero

Complemento

Bairro

CEP

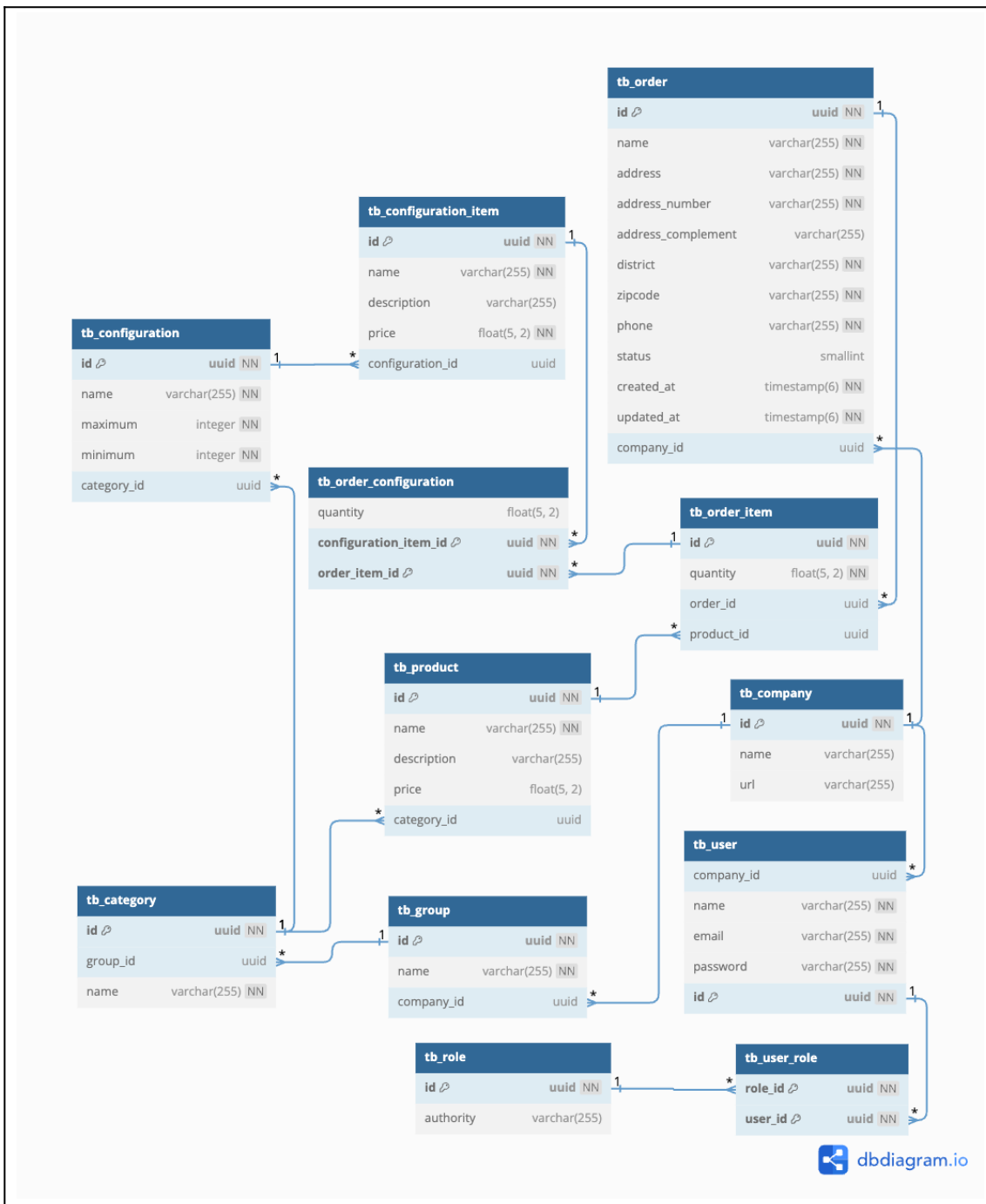
Telefone

CONTINUAR COMPRANDO

FINALIZAR

Página do carrinho e fechamento do pedido (Usuário)

9. Modelo Relacional ou Projeto de Banco de Dados NoSQL



10. Plano de Testes

Nº	Caso de uso	Objetivo do caso de teste	Entradas	Resultados esperados
1	Manter empresas - CRUD	Fazer o cadastro de uma nova empresa com sucesso	-Logar no sistema com usuário de perfil Admin (email: luke@starwars.com , senha: 123). -Acessar a tela de listagem de empresas e clicar no botão 'Cadastrar' para abrir o modal de cadastro. -Informar os seguintes dados: Nome: Temakeria Url: temakeria -Clicar no botão 'Cadastrar' do modal.	-O modal de cadastro se fecha. -O sistema apresenta uma mensagem de sucesso informando que o cadastro foi realizado. -A tabela é atualizada com a nova empresa cadastrada.
2	Manter empresas - CRUD	Verificar que não é possível inserir no sistema uma empresa com uma url já existente	-Realizar o processo de cadastro de empresas descrito no caso de teste nº 1 informando os seguintes dados: Nome: Nova Temakeria Url: temakeria -Clicar no botão 'Cadastrar' do modal.	-O modal de cadastro se fecha. -O sistema apresenta uma mensagem de erro.

3	Manter usuários CRUD	- Fazer o cadastro com sucesso de um usuário vinculado à uma empresa.	-Logar no sistema com usuário de perfil Admin (email: luke@starwars.com , senha: 123). -Acessar a tela de listagem da empresa e clicar no botão 'Usuários' (ícone verde). -Clicar no botão cadastrar da listagem de usuários para abrir o modal de cadastros. Informar os seguintes dados: Nome: Novo Usuário email: novousuario@email.com Senha e Confirmar senha: 123 -Selecione pelo menos uma permissão. -Clicar no botão 'Cadastrar' do modal.	-O modal de cadastro se fecha. -O sistema apresenta uma mensagem de sucesso informando que o cadastro foi realizado. -A tabela é atualizada com o novo usuário cadastrado.
4	Manter usuários CRUD	- Verificar que não é possível cadastrar um usuário sem informar os campos obrigatórios.	-Realizar o processo de cadastro de empresas descrito no caso de teste nº 3 informando os seguintes dados: Nome: Usuário Teste email: não preencher Senha e Confirmar senha: 123. -Não selecione nenhuma permissão. -Clicar no botão 'Cadastrar' do modal.	-Os inputs obrigatórios não preenchidos ficam vermelhos e exibem uma mensagem.

5	Alterar status do pedido	Alterar com sucesso o status de um pedido.	-Logar no sistema com usuário de perfil Provider (email: page@lanchonete.com , senha: 123). -Acessar a tela de listagem de pedidos. -Escolha um pedido e na coluna ações clique no botão 'Detalhes' (ícone azul) para abrir o modal com os detalhes do pedido. -No campo 'Status' selecione um novo status e clique no botão 'Alterar Status' do modal	-O modal de detalhes do pedido se fecha. -O sistema apresenta uma mensagem de sucesso informando que o status foi alterado. -A tabela é atualizada com o novo status.
---	--------------------------	--	--	---

11. Apropriação de Horas no Projeto

Histórico de apropriação de horas		
Data do registro	Atividade	Quantidade de horas
08 / 05 / 23	Contextualização e objetivos do trabalho	6 horas
11 / 05 / 23	Definição conceitual - diagrama de casos de uso	4 horas
12 / 05 / 23	Definição conceitual – diagrama de casos de uso	4 horas
13 / 05 / 23	Definição conceitual – requisitos funcionais	3 horas
14 / 05 / 23	Definição conceitual – requisitos funcionais	3 horas
15 / 05 / 23	Definição conceitual – requisitos funcionais	2 horas
16 / 05 / 23	Definição conceitual – requisitos não funcionais	2 horas
17 / 05 / 23	Diagrama de classes de domínio	3 horas
18 / 05 / 23	Diagrama de classes de domínio	3 horas
19 / 05 / 23	Diagrama de classes de domínio	4 horas
20 / 05 / 23	Protótipo navegável do sistema	4 horas
21 / 05 / 23	Protótipo navegável do sistema	4 horas
22 / 05 / 23	Protótipo navegável do sistema	5 horas
24 / 05 / 23	Protótipo navegável do sistema	3 horas
25 / 05 / 23	Protótipo navegável do sistema	2 horas
26 / 05 / 23	Protótipo navegável do sistema	3 horas

27 / 05 / 23	Protótipo navegável do sistema	8 horas
28 / 05 / 23	Protótipo navegável do sistema	6 horas
01 / 06 / 23	Protótipo navegável do sistema	3 horas
03 / 06 / 23	Protótipo navegável do sistema	3 horas
04 / 06 / 23	Protótipo navegável do sistema	3 horas
08 / 06 / 23	Vídeo de apresentação do protótipo	6 horas
10 / 06 / 23	Definição do padrão arquitetural	3 horas
11 / 06 / 23	Desenvolvimento do C4 model	4 horas
13 / 06 / 23	Escolha dos frameworks de trabalho	2 horas
14 / 06 / 23	Estrutura base do frontend	1 hora
15 / 06 / 23	Desenvolvimento do modelo relacional	3 horas
17 / 06 / 23	Elaboração do plano de testes	2 horas
19 / 06 / 23 a 10 / 07 / 23	Desenvolvimento do backend	52 horas
12 / 07 / 23 a 02 / 08 / 23	Desenvolvimento do frontend	67 horas
07 / 09 / 23	Código da aplicação	3 horas
07 / 09 / 23	Avaliação retrospectiva	2 horas

12. Código da Aplicação

Link repositório: <https://github.com/guto84/tcc-puc-eng-software>

Link aplicação:

- Cardápio lanchonete : <https://carlos2920.c34.integrator.host/lanchonete>
- Painel Admin:
 - Link: <https://carlos2920.c34.integrator.host/login>
 - Credenciais de acesso:
 - email: luke@starwars.com
 - senha: 123
- Painel Provider:
 - Link: <https://carlos2920.c34.integrator.host/login>
 - Credenciais de acesso:
 - email: page@lanchonete.com
 - senha: 123

Link Video:

https://drive.google.com/file/d/1MNoDe0p3tPIUAoMdpMCsajg56gCMCRIV/view?usp=drive_link

13. Avaliação Retrospectiva

O objetivo geral era apresentar o desenvolvimento de um sistema de delivery voltado para estabelecimentos como restaurantes, pizzarias, lanchonetes e similares, aplicando conceitos e práticas de Engenharia de Software.

13.1 Objetivos Estimados

1. **Levantamento de Arquitetura:** Realizar um estudo detalhado para identificar a arquitetura que melhor atenderia às necessidades do projeto, considerando escalabilidade, desempenho e segurança.
2. **Desenvolvimento de Software:** Desenvolver um sistema robusto com banco de dados eficiente e interface de usuário intuitiva, utilizando as tecnologias mais adequadas.
3. **Implantação em Cloud:** Implantar o sistema em uma plataforma de hospedagem em nuvem (cloud provider) para flexibilidade, escalabilidade e acessibilidade.

13.2 Objetivos Alcançados

Objetivo 1 - Levantamento de Arquitetura:

- Objetivo Estimado: Realizar um estudo detalhado para identificar a arquitetura ideal.
- Objetivo Alcançado: Concluimos a análise de arquitetura, escolhendo uma estrutura sólida para o desenvolvimento do sistema.

Objetivo 2 - Desenvolvimento de Software:

- Objetivo Estimado: Criar um sistema robusto e eficiente.
- Objetivo Alcançado: Desenvolvemos com sucesso um software que atendeu aos requisitos estabelecidos, incorporando um banco de dados sólido e uma interface de usuário amigável.

Objetivo 3 - Implantação em Cloud:

- Objetivo Estimado: Implantar o sistema em uma plataforma em nuvem.

- **Objetivo Alcançado:** Concluímos a implantação em uma plataforma em nuvem, garantindo flexibilidade e escalabilidade.

13.2 Lições aprendidas

A avaliação retrospectiva destaca que os objetivos estabelecidos foram alcançados com êxito, resultando em um sistema de delivery de comida eficiente e escalável. As lições aprendidas ao longo do processo contribuirão para melhorar futuros projetos de desenvolvimento de software e arquitetura de sistemas em nuvem. Este TCC foi uma oportunidade valiosa para aplicar os princípios da Engenharia de Software em um contexto prático e real.

Retrospectiva (Lições Aprendidas)		
	Descrição da Lição	Classificação
1	Planejamento e Pesquisa Sólidos: A fase inicial de pesquisa e planejamento foi fundamental para o sucesso do projeto. Aprofundar-se na análise de arquitetura permitiu tomar decisões informadas.	Positiva
2	Escolha de Tecnologia: A seleção cuidadosa das tecnologias utilizadas no desenvolvimento foi crucial. A escolha de tecnologias atualizadas contribuiu para a eficiência do sistema.	Positiva
3	Desafios de Implantação: A implantação em uma plataforma em nuvem apresentou desafios técnicos, mas também trouxe vantagens significativas. Aprendemos a superar esses desafios e a aproveitar ao máximo a infraestrutura em nuvem.	Positiva
4	Usabilidade Contínua: A usabilidade do sistema é um aspecto crítico. Ao longo do desenvolvimento, percebemos a importância de testar continuamente a interface do usuário e incorporar o feedback dos usuários.	Positiva
5	Adaptação Futura: A decisão de começar com uma plataforma web e a possibilidade de estender para um aplicativo móvel revelou-se estratégica. Isso nos permite expandir o alcance do sistema no futuro.	Positiva

14. Referências

Não se aplica.