PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

PUC Minas Virtual

Pós-graduação *Lato Sensu* em Engenharia de *Software*

Projeto Integrado

Relatório Técnico

Cantina Fácil

Nilson Pereira Durval Junior
Daniel Dutra Melo

Belo Horizonte
Abril, 2023.

Projeto Integrado

Sumário

1. Cronograma de Trabalho

Datas		Atividade / Tarefa	Produto / Resultado
De	**Até**		
23/01/2023	24/01/2023	1. Desenvolver Etapa 1 - Elaborar cronograma	Cronograma
25/01/2023	27/01/2023	2. Desenvolver Etapa 1 - Elaborar contextualização do projeto.	Parágrafo de contextualização do projeto
30/01/2023	03/02/2023	3. Desenvolver Etapa 1 - Elaboração de casos de uso	Casos de uso
06/02/2023	10/02/2023	4. Desenvolver Etapa 1 - Elaborar requisitos funcionais	Parágrafo de Requisitos funcionais
06/02/2023	10/02/2023	5. Desenvolver Etapa 1 - Elaborar requisitos não funcionais	Parágrafo de Requisitos não funcionais
13/02/2023	17/02/2023	6. Desenvolver Etapa 1 - Elaborar protótipo navegável	Protótipo navegável do sistema
20/02/2023	24/02/2023	7. Desenvolver Etapa 1 - Elaborar Diagrama de classes de domínio.	Diagrama de classes
27/02/2023	28/02/2023	8. Desenvolver Etapa 1 - Preencher apropriação de horas	Apropriação de horas
03/04/2023	07/04/2023	9. Desenvolver Etapa 2 - Descrever padrão arquitetural.	Padrão arquitetural do sistema
10/04/2023	14/04/2023	10. Desenvolver Etapa 2 - Definir arquitetura da solução	Arquitetura da solução

17/04/2023	21/04/2023	11. Desenvolver Etapa 2 - Desenvolver modelo relacional do projeto	Modelo relacional do projeto
24/04/2023	28/04/2023	12. Desenvolver Etapa 2 - Elaborar casos de teste	Casos de teste
01/05/2023	05/05/2023	13. Desenvolver Etapa 2 - Atualizar histórico de apropriação de horas	Histórico de apropriação de horas

2. Introdução

É sabido que se passa boa parte do tempo de vida nas escolas, principalmente na infância e adolescência, onde ainda se está amadurecendo e se precisa do controle dos seus tutores. Sendo assim, um aspecto bem importante dessa fase, é a alimentação. Como as crianças e adolescentes passam boa parte do seu tempo nas escolas elas precisam se alimentar. Para isso, existe a cantina, onde os estudantes compram suas refeições. Porém, as crianças e adolescentes não possuem renda, por isso, quem paga pelas suas refeições são seus tutores, normalmente, se dá dinheiro ou existe uma conta, onde o estudante pega a refeição e é anotado, para que os tutores efetuem o pagamento posteriormente. Entretanto, essa abordagem pode ser melhorada, unindo controle de alimentação e financeiro, em um sistema. Esta é a proposta desse projeto, um sistema de gerenciamento de cantinas de escolas, que possuirá também um aplicativo para o estudante, e para o seu tutor. Nele, as cantinas se cadastram e possuem acesso a inúmeras funcionalidades, como cadastro de produtos, pedidos feitos, limites de compra do aluno etc. O aluno tem acesso ao cardápio e a uma carteira virtual que poderá ser alimentada pelo seu tutor, poderá fazer pedidos pelo app e retirar na cantina, evitando filas, e também poderá fazer o pagamento online pelo app. Os tutores poderão fazer o gerenciamento da alimentação dos seus filhos, excluindo alimentos e incentivando outros, receberão mensagem das compras no app dos seus filhos e realizarão o pagamento com o cartão de crédito.

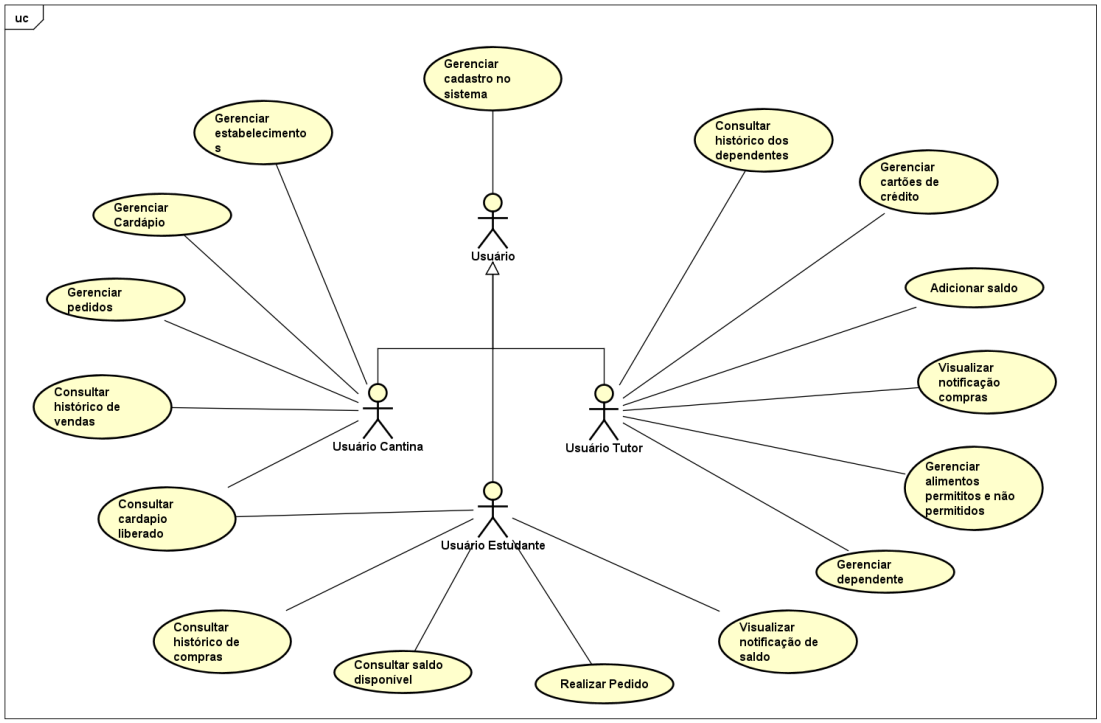
O objetivo deste trabalho é apresentar a descrição do projeto de uma aplicação de gerenciamento de cantinas escolares, sendo esse dividido em 3 módulos, cantinas, estudantes e tutores.

Os objetivos específicos da aplicação são:

- Facilitar o controle financeiro e de pedidos das cantinas;
- Dar o controle da alimentação dos estudantes, na escola, aos tutores;
- Evitar a perda de tempo dos estudantes em filas;

3. Definição Conceitual da Solução

3.1. Diagrama de Casos de Uso



3.2. Requisitos Funcionais

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF01	O usuário deve poder gerenciar seu cadastro no sistema.	B	A
RF02	O sistema deve permitir ao usuário cantina gerenciar o cardápio. Adicionando, consultando, removendo e atualizando produtos.	B	A
RF03	O usuário cantina deve ser capaz de gerenciar seus estabelecimentos.	B	A
RF04	O sistema deve permitir aos usuários, cantina e estudante, consultar o cardápio liberado do aluno.	B	A
RF05	O sistema deverá disponibilizar um histórico de vendas, para o usuário cantina.	B	M

RF06	O sistema deverá permitir ao usuário tutor, gerenciar seus cartões de crédito.	A	A
RF07	O sistema deverá exibir notificações de compras para o usuário tutor.	M	B
RF08	O sistema deverá validar o cartão de crédito do usuário tutor.	A	A
RF09	O usuário tutor será capaz de liberar/bloquear alimentos dos seus dependentes.	M	A
RF10	O usuário tutor será capaz de gerenciar seus dependentes.	B	A
RF11	O sistema deverá emitir notificação de saldo adicionado para o usuário estudante.	M	B
RF12	O usuário estudante poderá realizar pedidos de lanches.	M	A
RF13	O sistema deverá disponibilizar histórico de compras, para os usuários estudante e tutor.	B	B
RF14	O sistema deverá integrar com cartão de crédito no pagamento	A	A
RF15	O sistema deverá prover relatórios estatísticos sobre as vendas de produtos	B	B
RF16	Os pedidos poderão ser avaliados pelos usuários do tipo estudante.	B	B

* 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.	A
RNF02	O sistema deve ser seguro para evitar o acesso não autorizado ou a divulgação de informações	A

	confidenciais, como informações financeiras e pessoais dos usuários.	
RNF03	O sistema deve ter alta disponibilidade, com 99,98% em funcionamento.	A
RNF04	O sistema deve ser escalável, sendo possível aumentar e diminuir o número de instâncias de acordo com o número de acessos.	M
RNF05	O sistema deve ser simples e intuitivo.	A
RNF06	o sistema deve garantir a integridade dos dados, garantindo que os dados sejam mantidos precisos e consistentes em todas as transações.	A
RNF07	o sistema deve ser flexível o suficiente para permitir atualizações e melhorias contínuas	M
RNF08	O sistema deverá ser responsivo.	A

4. Protótipo Navegável do Sistema

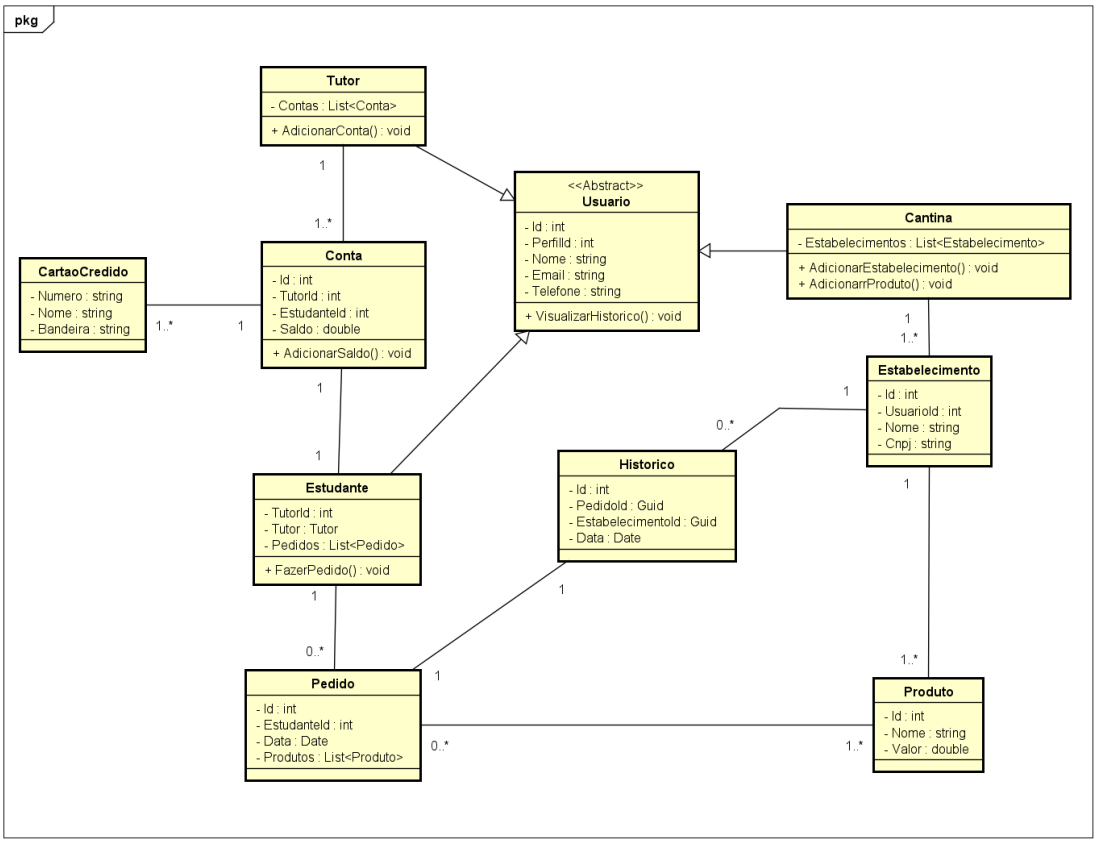
Link protótipo navegável:

https://www.figma.com/community/file/1258493753151285352

Vídeo do Protótipo Navegável:

https://github.com/nilsondurval/cantina-facil/tree/main/Docs/Apresentacao%20Etapa%201%20-%20Prototipo%20Navegavel

5. Diagrama de Classes de Domínio



6. Arquitetura da Solução

6.1. Padrão Arquitetural

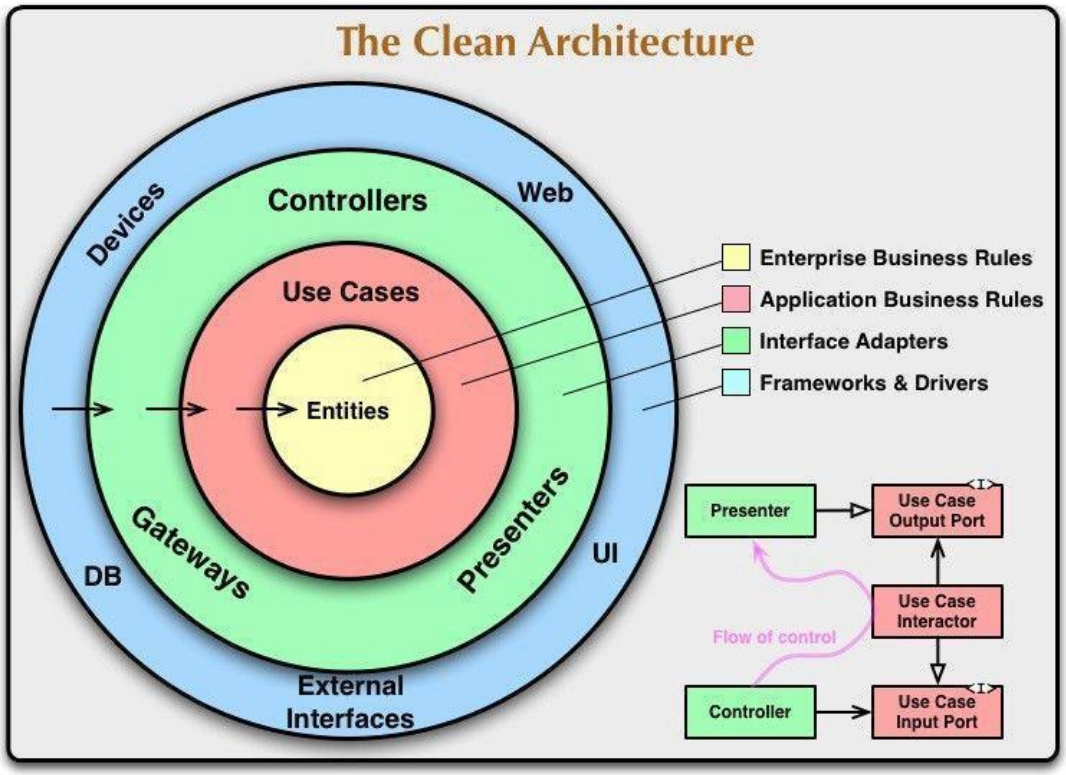
O padrão arquitetural utilizado para implementação da aplicação foi a arquitetura em camadas, utilizando-se dos conceitos da arquitetura limpa. Proposto pelo Robert Cecil Martin, o Uncle Bob, como é conhecido no meio, esse padrão arquitetural visa o de organizar as responsabilidades de partes de um software, normalmente criando um isolamento e dando um propósito bem definido a cada camada de forma que a mesma possa ser reutilizável por um nível mais alto ou até substituível.

Mas na prática, principalmente em aplicações corporativas, essas características se perdem em meio a códigos desorganizados e entendimento confuso sobre tais conceitos.

De acordo com o Uncle Bob, a conformidade com essas regras simples não é difícil e evitará muitas dores de cabeça no futuro. Ao separar o software em camadas e em conformidade com a regra de dependência, você criará um sistema intrinsecamente testável, com todos os benefícios que isso implica. Quando qualquer uma das partes externas do sistema se torna obsoleta, como o banco de dados ou a estrutura da Web, você pode substituir esses elementos obsoletos com o mínimo de confusão. (MARTIN, 2012, tradução nossa)

Na imagem abaixo pode-se visualizar como é dividida as camadas da arquitetura limpa.

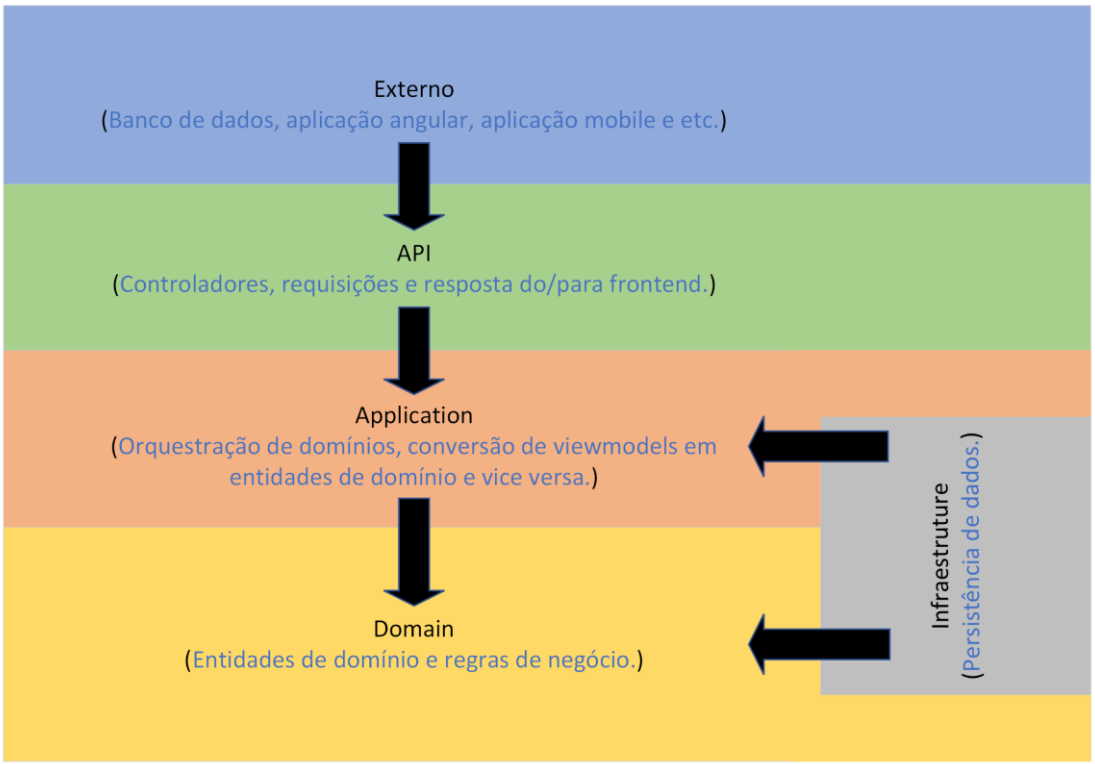
Figura 1- Arquitetura limpa original



Fonte: The Clean Code Blog, 2012.

Na arquitetura proposta, as camadas foram definidas e segregadas da seguinte forma, da mais externa para a mais interna. No equivalente a camada mais externa da arquitetura limpa, a de frameworks e drivers, estão o banco de dados, a aplicação web em angular e o aplicativo mobile. Na próxima camada, equivalente a interface adapters, na aplicação chamada de API, estão os controladores, que fazem a comunicação com as aplicações front-end. Já na próxima camada, que seria o equivalente a camada de application business rules, chamada na aplicação de Application, temos a camada de orquestração das regras de negócio, e tradução das viewmodels do frontend em entidades de negócio e vice-versa. Logo após, fica a camada mais interna, a camada de domínio, ou na aplicação Domain, equivalente a enterprise business rules da arquitetura limpa, é lá onde se encontram as agregações e entidades de domínio, junto com toda regra de negócio que as envolve. Além disso, na arquitetura proposta, existe a camada infraestructure, nela está o framework de persistência de dados. Para satisfazer um dos conceitos pregados pela arquitetura limpa, a camada de infraestrutura (acesso a dados) é que depende do domínio e não o inverso, dessa forma podemos substituí-la sem afetar as outras camadas.

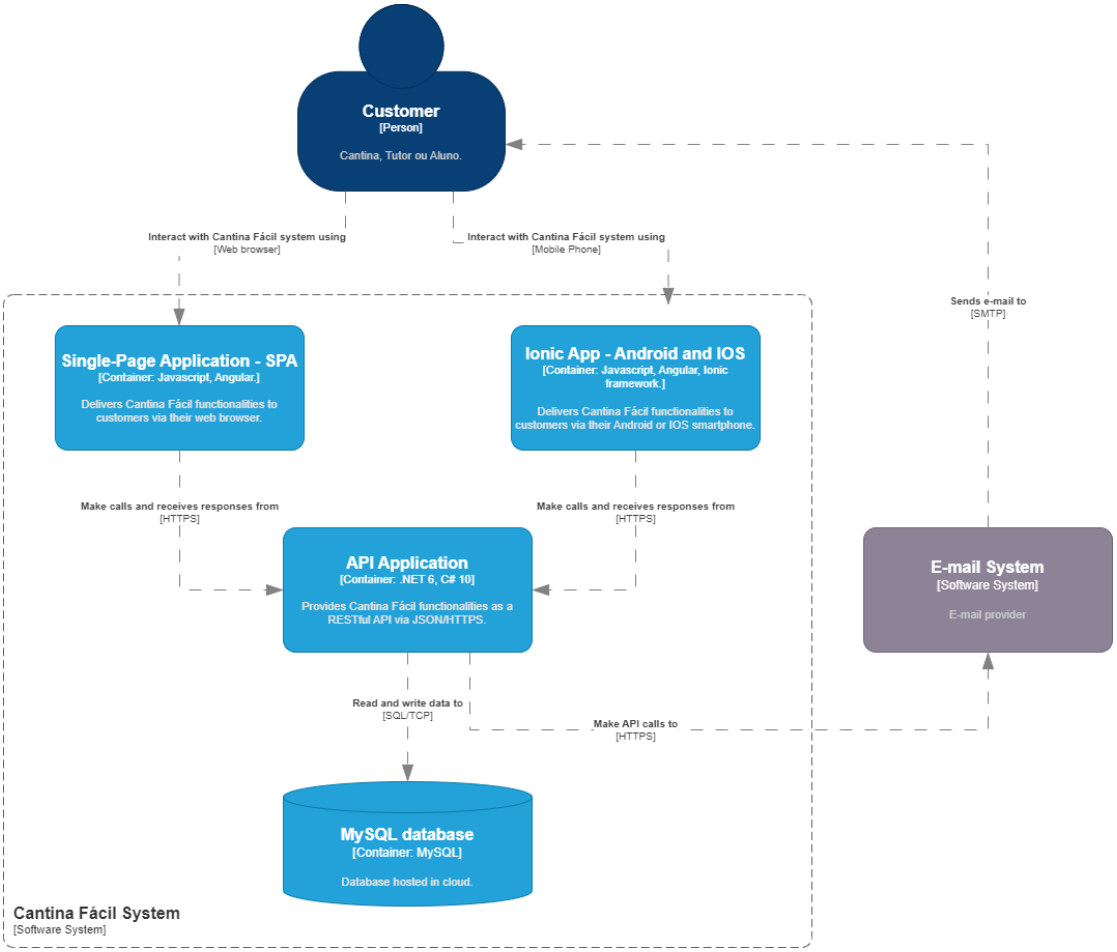
Abaixo pode-se ver uma ilustração da arquitetura proposta, mostrando o fluxo da dependência entre as camadas. A camada API, recebe requisições externas, repassa para a camada Application, essa por sua vez, pode chamar direto a camada de acesso a dados, no caso de consultas simples, ou chamar um domínio para processar alguma regra de negócio antes de chamar o acesso a dados.



Fonte: MARTIN, 2011, p. 122.

6.2. C4 model – Diagrama de Contexto

Na visão macro da arquitetura do sistema, como se pode ver na imagem a seguir. Temos um usuário que pode ser do tipo cantina, tutor ou estudante. Esses usuários interagem com o sistema utilizando um web browser ou telefone celular. No acesso via browser as requisições são recebidas pela SPA – Single-Page Application, no caso do acesso via telefone celular as requisições são tratadas pelo aplicativo híbrido que será escrito utilizando o framework Ionic. Para prover as funcionalidades para o usuário, a SPA e o aplicativo se comunicam com a API. Que por sua vez consulta e grava dados no banco de dados MySQL.



Fonte: Elaborada pelo autor.

7. *Frameworks de Trabalho*

Os Frameworks utilizados nessa aplicação foram os seguintes:

Backend:

1. **.NET 6,** é uma plataforma de desenvolvimento multiplataforma de código aberto gratuita para criar muitos tipos diferentes de aplicativos. Com o .NET, você pode usar vários idiomas, editores e bibliotecas para criar para Web, dispositivos móveis, desktop, jogos, IoT e muito mais.

   Site oficial: https://dotnet.microsoft.com/pt-br/

2. **C#,** é uma linguagem de programação orientada a objetos e orientada a componentes. C# fornece construções de linguagem para dar suporte diretamente a esses conceitos, tornando C# uma linguagem natural para criação e uso de componentes de software.

   Site oficial: https://learn.microsoft.com/pt-br

3. **Swagger,** é um framework composto por diversas ferramentas que, independente da linguagem, auxilia a descrição, consumo e visualização de serviços de uma API REST.
   Site oficial: https://swagger.io/

Frontend:

1. **Angular 15,** é um framework baseado em JavaScript utilizado para criação de Single Page Applications (SPA).
   Site oficial: https://angular.io/

2. **Typescript,** uma linguagem de programação de código aberto que estende o JavaScript, adicionando tipos estáticos opcionais. Ela oferece recursos avançados de orientação a objetos, interfaces, módulos e compila para JavaScript puro.
   Site oficial: https://www.typescriptlang.org/

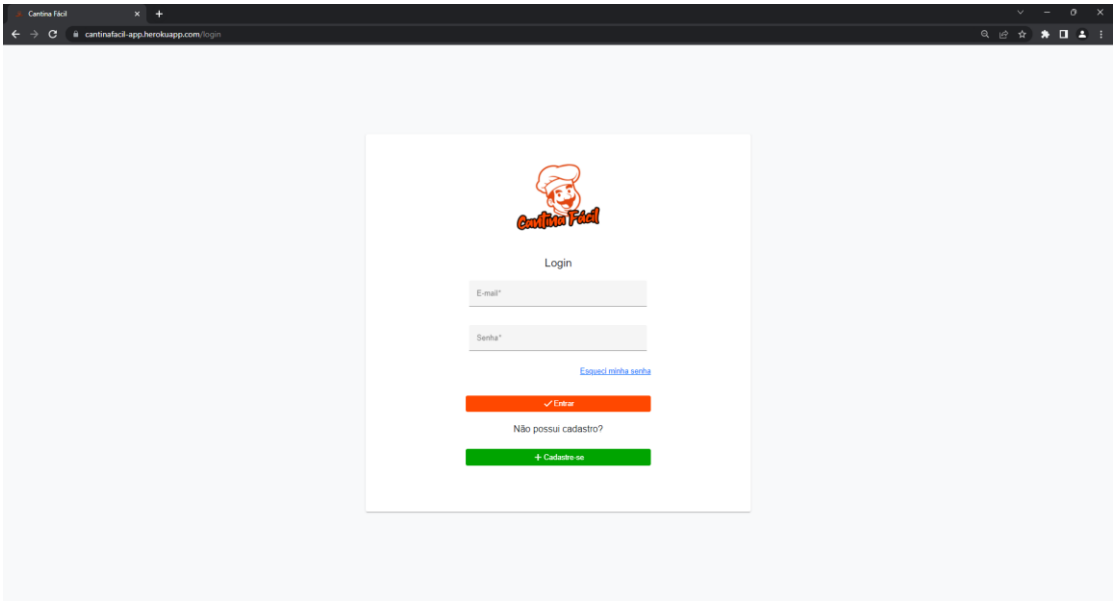
3. **Angular Material,** é a implementação oficial, para o Angular, do Material Design, a especificação de design para interfaces interativas do Google.
   Site oficial: https://material.angular.io/

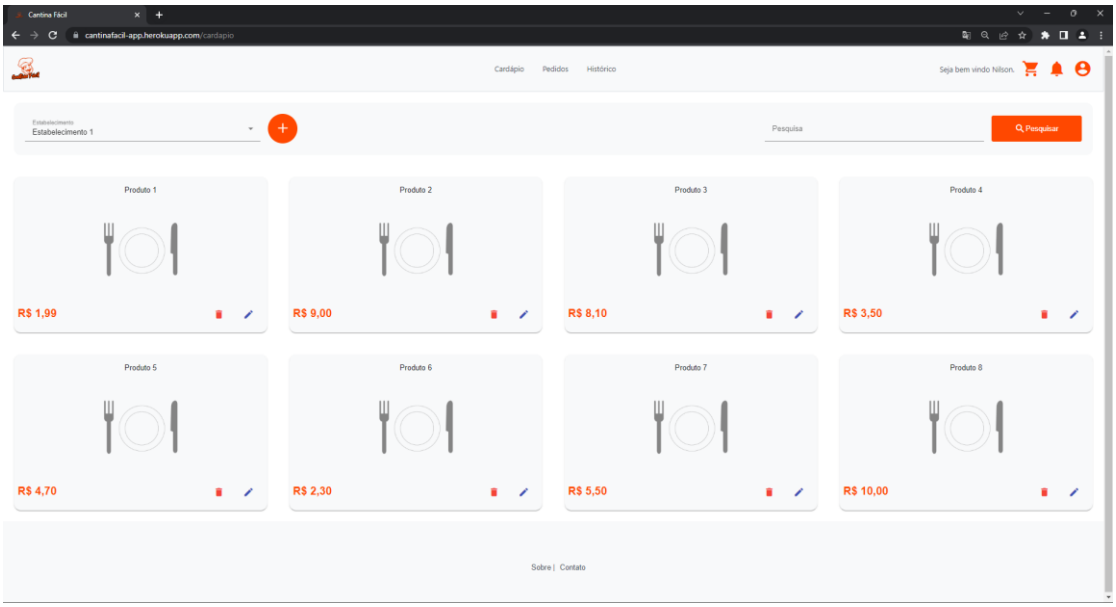
4. **PrimeNG,** é uma biblioteca de componentes rica e altamente personalizável que pode ser facilmente integrada em projetos Angular existentes.
   Site oficial: https://primeng.org/

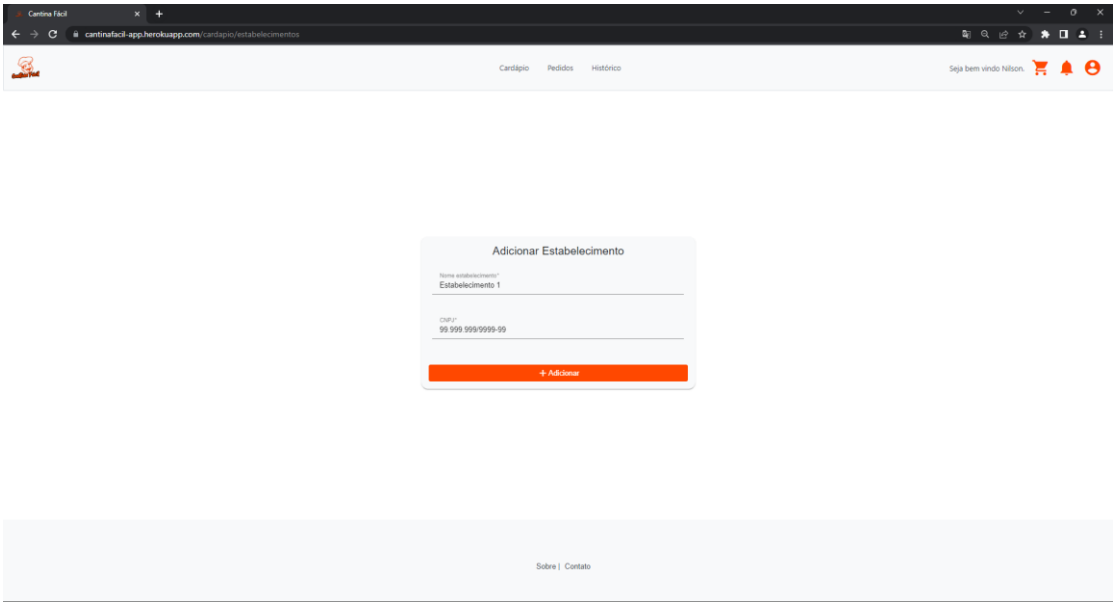
Database:

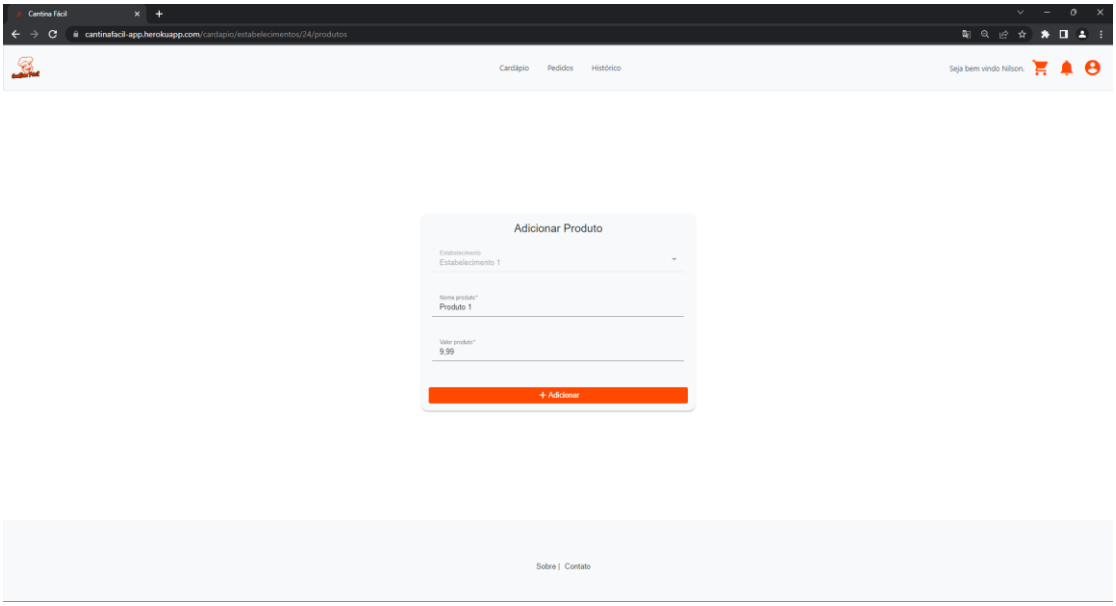
1. **MySQL,** é um sistema de gerenciamento de banco de dados, que utiliza a linguagem SQL como interface. É atualmente um dos sistemas de gerenciamento de bancos de dados mais populares
   Site oficial: https://mysql.com/

8. Estrutura Base do Front End

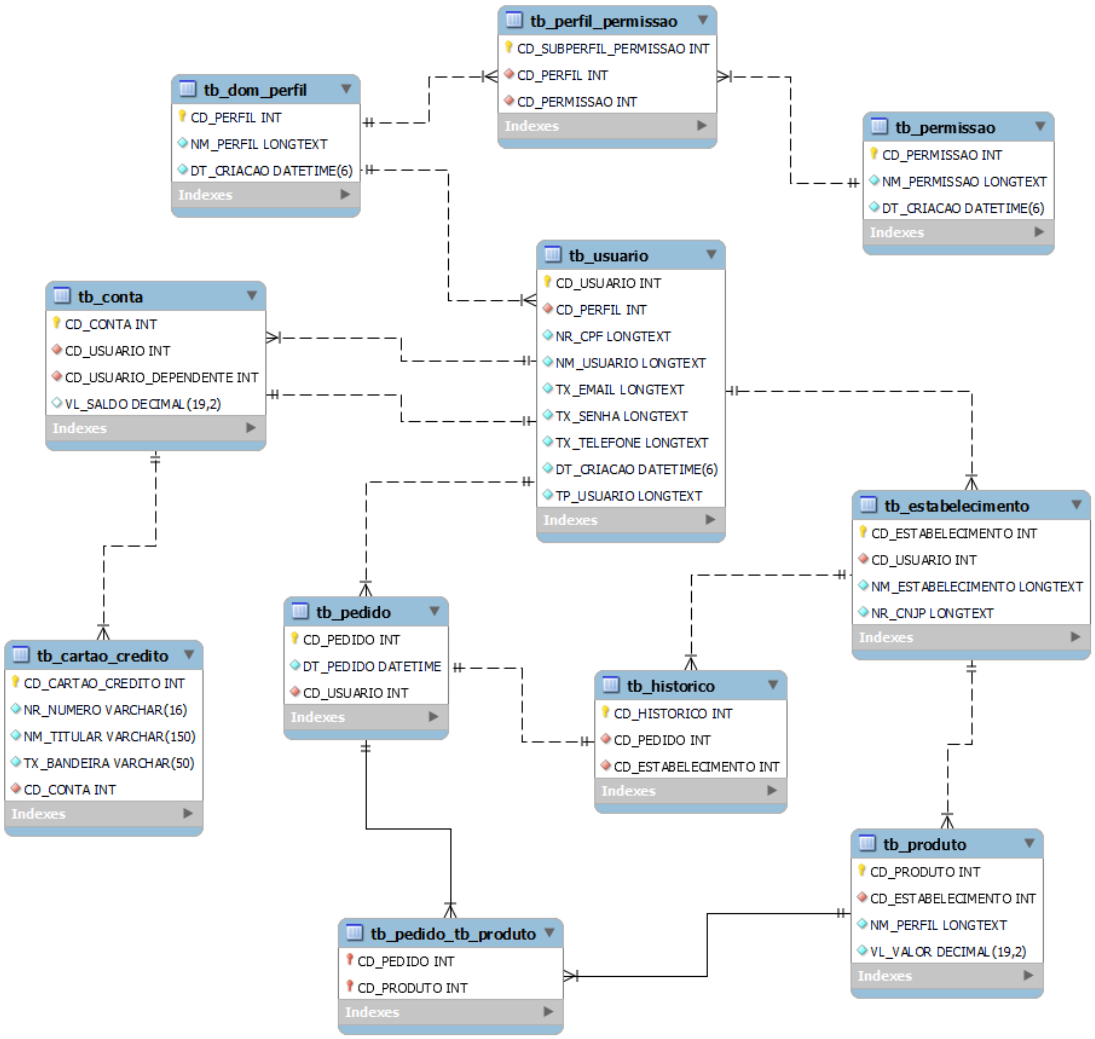





Cantina Fácil
cantinafacil-app.herokuapp.com/cardapio/estabelecimentos
Cardápio
Pedidos
Histórico
Seja bem vindo Nilson.
Adicionar Estabelecimento
Nome estabelecimento*
Estabelecimento 1
CNPJ*
99.999.999/9999-99
+ Adicionar
Sobre | Contato


Cantina Fácil
cantinafacil-app.herokuapp.com/cardapio/estabelecimentos/24/produtos
Cardápio
Pedidos
Histórico
Seja bem vindo Nilson.
Adicionar Produto
Estabelecimento
Estabelecimento 1
Nome produto*
Produto 1
Valor produto*
9.99
+ Adicionar
Sobre | Contato

9. Modelo Relacional



10. Plano de Testes

Número	**Caso de uso**	**Objetivo do caso de teste**	**Entradas**	**Resultados esperados**
1.	Fazer cadastro do usuário	Cadastrar os dados de um usuário na base de dados	Clicar em “Cadastre-se”, preencher as informações corretamente e clicar em “Cadastrar”	O sistema apresenta uma mensagem de “Cadastrado com sucesso” e retorna a tela de login.
2.		Mostrar mensagem de erro quando um dado obrigatório não for preenchido	Clicar em “Cadastrar-se” e clicar em “Cadastrar”	O sistema apresenta a mensagem “ Campo obrigatório!”
3.	Logar no sistema	Conseguir que o usuário efetue o login	Preencher o e-mail, senha e clicar em “Entrar”	O sistema apresenta a mensagem “Bem vindo ao cantina fácil” e a tela de “boas vindas”
4.		Mostrar mensagem de erro quando a informações forem incorretas	Preencher o e-mail, senha e clicar em “Entrar”	O sistema apresenta a mensagem “Usuário e/ou senha invalido(s)”
5.	Adicionar um estabelecimento	Conseguir cadastrar um novo estabelecimento	Preencher o nome e CNPJ do estabelecimento	O sistema retorna a tela de cardápio
6.		Mostrar mensagem caso o usuário não preencha os campos	Clicar em “Adicionar estabelecimento” e clicar em “Adicionar”	O sistema apresenta a mensagem “Campo obrigatório!”

7	Editar um estabelecimento	Conseguir editar um estabelecimento existente	Preencher o nome e CNPJ do estabelecimento	O sistema apresenta a mensagem “Estabelecimento atualizado com sucesso!” e retorna a tela de cardápio
8		Mostrar mensagem caso o usuário não preencha os campos	Clicar em “Editar estabelecimento” e clicar em “Atualizar”	O sistema apresenta a mensagem “Campo obrigatório!”
9	Adicionar um produto	Conseguir cadastrar um novo produto	Preencher o nome e valor do produto	O sistema apresenta a mensagem “Produto cadastrado com sucesso!” e retorna a tela de cardápio
10		Mostrar mensagem caso o usuário não preencha os campos	Clicar em “Adicionar produto” e clicar em “Adicionar”	O sistema apresenta a mensagem “Campo obrigatório!”
11	Editar um produto	Conseguir editar um produto existente	Preencher o nome e valor do produto	O sistema apresenta a mensagem “Produto atualizado com sucesso!” e retorna a tela de cardápio
12		Mostrar mensagem caso o usuário não preencha os campos	Clicar em “Editar produto” e clicar em “Atualizar”	O sistema apresenta a mensagem “Campo obrigatório!”

11. Apropriação de Horas no Projeto

Histórico de apropriação de horas		
Data do registro	**Atividade**	**Quantidade de horas**
20/01/2023	Rever toda matéria referente ao Projeto Integrado, textos e vídeos.	16
27/01/2023	Relatório Técnico: Objetivos do Trabalho, Apresentação do Problema, Descrição Geral do Software.	12
03/02/2023	Relatório Técnico: Identificação da ferramenta e desenvolvimento do diagrama de Caso de Uso.	18
10/02/2023	Relatório Técnico: Identificação de atores, requisitos funcionais e não funcionais.	20
10/02/2023	Relatório Técnico: Descrição casos de usos.	10
24/02/2022	Relatório Técnico: Criação do Protótipo de Interface, navegável e seus itens relacionados.	20
24/02/2022	Relatório Técnico: Criação do Diagrama de Classes de domínio.	20
10/05/2023	Adequação de acordo com o feedback do Professor orientador	12
15/05/2023	Definição do padrão arquitetural do projeto e tecnologias	8
16/05/2023	Definição da solução	8
17/05/2023	Definição de frameworks	6
20/05/2023	Definição da estrutura front-end	10
21/05/2023	Definição da estrutura back-end	10
22/05/2023	Projeto do banco de dados	10
05/06/2023	Desenvolvimento do back-end	25
15/06/2023	Desenvolvimento do front-end	25
25/06/2023	Revisão do projeto	20

12. Código da Aplicação

As funcionalidades escolhidas para desenvolvimento e validação da arquitetura da aplicação, foram o cadastro de usuários, sendo somente do perfil de cantina, e o gerenciamento de estabelecimentos e produtos. São dados a seguir, detalhes de como se pode acessar o código fonte do projeto e também acessar a aplicação, que foi publicada no Heroku. Existe um usuário pré-cadastrado para testes, porém é possível cadastrar outros usuários se for necessário.

Código da aplicação

https://github.com/nilsondurval/cantina-facil

Endereço da documentação Swagger da API:

https://cantinafacil-api.herokuapp.com/swagger/index.html

Endereço da aplicação

https://cantinafacil-app.herokuapp.com

Usuário pré-cadastrado

- E-mail: teste1@teste.com
- Senha: pucMINAS@2023

Vídeo de navegação na aplicação:

https://github.com/nilsondurval/cantina-facil/tree/main/Docs/Apresentacao%20Etapa%202%20-%20Arquitetura%20da%20Solucao

13. Avaliação Retrospectiva

Os objetivos sempre foram muito claros durante esse projeto, buscando um acompanhamento do cronograma e executar de fato o que foi proposto. As lições aprendidas durante todo o período do curso agregaram para o resultado, mesmo em um projeto com tantas dificuldades, foi possível concluir de maneira satisfatória.

13.1. Objetivos Estimados

A distribuição do tempo e os detalhes das interfaces trouxeram bastante desafios, mas pouco a pouco as funcionalidades foram atendidas.

13.2. Objetivos Alcançados

Alguns são os pontos que podemos levantar como retrospectivo positivo, como a aplicação dos conceitos teóricos na prática, trazendo maior vivência e entendimento do assunto, divisão das tarefas em dupla, ideias para se aplicar em projetos reais, que de fato podem funcionar e a validação em cada etapa do processo.

13.3. Lições Aprendidas

	Retrospectiva (Lições Aprendidas)	
	Descrição da Lição	**Classificação**
1	Aplicação dos conceitos teóricos na prática	Positiva
2	Divisão de tarefas em dupla	Positiva
3	Ideias a se aplicar em projetos reais	Positiva
4	Validação das etapas do processo	Positiva
5	Conhecimento adquirido no projeto	Positiva
6	Acompanhamento de cronograma	Positiva

14. Referências

MARTIN, Robert CECIL. Código Limpo: Habilidades Práticas do Agile Software. Rio de Janeiro: Alta Books, 2011.

MARTIN, Robert CECIL. Arquitetura limpa: O guia do artesão para estrutura e design de software. Rio de Janeiro: Alta Books, 2019.

Coding Blocks, Atlanta, 18 de fevereiro de 2018. Disponível em: https://www.codingblocks.net/podcast/clean-architecture-make-your-architecture-scream/. Acesso em: 05 de Junho de 2023

KUROSE, Jim; ROSS, Keith. Rede de computadores e a internet: uma abordagem top-down. São Paulo: Pearson Education do Brasil, 2013.

Guia.Dev, São Paulo, 01 de fevereiro de 2021. Disponível em: https://guia.dev/pt/pillars/software-architecture/layers.html#:~:text=Arquitetura%20em%20camadas%20%C3%A9%20um,mais%20alto%20ou%20at%C3%A9%20substitu%C3%ADvel. Acesso em: 25 de Junho de 2023

MARTIN, Robert CECIL. The Clean Code Blog, Califórnia, 13 de agosto de 2012. Disponível em: https://blog.cleancoder.com/ .Acesso em: 27 de Junho de 2023