Curso Superior de Desenvolvimento de Software Multiplataforma

Douglas Wenzel, 3011392413022
Fernando Chibli, 3011392323017
Gustavo Ferreira, 3011392413016
Isabel Maito, 3011392413045

Projeto Interdisciplinar III

Gestão Ágil de Projetos de Software

DelBicos - Delivery de Bicos

Orientadores
Profª Ma. Maria Janaína da Silva Ferreira

Votorantim - SP
Junho, 2025

RESUMO

O delivery de bicos, DelBicos é uma plataforma inovadora que conecta clientes a profissionais locais, garantindo qualidade, segurança e praticidade na contratação de serviços. Alinhada com os ODS da ONU, a empresa promove o crescimento econômico local e a inclusão social, ao mesmo tempo em que cria um modelo de negócios lucrativo e sustentável.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

1. DESCRIÇÃO DO PROJETO

DelBicos - Delivery de Bicos, é um projeto que propõe conectar clientes e trabalhadores informais na mesma vizinhança, garantindo demanda local, qualidade e segurança e será projetado através de um website.

O projeto consiste em uma aplicação web completa, desenvolvida com Node.js (Backend) e React/Next.js (Frontend), integrada a um banco de dados não relacional (MongoDB).

1.1. Objetivo Geral do Projeto

O backend é a estrutura responsável por toda a lógica de negócios, processamento de dados e integração com o banco de dados do sistema DelBicos. Desenvolvido como uma API RESTful, fornece os endpoints necessários para o funcionamento da aplicação frontend. Ele foi arquitetado para gerenciar todas as operações relacionadas aos serviços de agendamentos. Foi desenvolvido um sistema completo de autenticação de usuários, com registro, login e recuperação de senha. Para os serviços, foi criado endpoints que permitem cadastrar, listar, filtrar e gerenciar agendamentos disponíveis.

O sistema possui controle de agendamentos, permitindo que usuários reservem agenda em horários específicos. Foi implementado também um módulo de avaliações, onde os clientes podem classificar os serviços utilizados. Todas as rotas foram protegidas com middleware de autenticação para garantir a segurança dos dados.

A comunicação entre frontend e backend ocorre através de requisições HTTP, com respostas no formato JSON. Organizado adequadamente os erros e status codes para facilitar a integração. Os modelos de dados foram cuidadosamente planejados no Mongoose para representar usuários, serviços, agendamentos e avaliações.

Foi desenvolvido no Figma e no Penpot, a interface do usuário de forma intuitiva e com uma boa usabilidade, fácil de navegar com acessibilidade

1.2. Gestão Ágil de Projetos de Software

1.2.3. Entregas das *Sprints*

O projeto está estruturado em três Sprints, começando com a implementação de funcionalidades essenciais para o cliente, como login, acesso à página inicial e cadastro, enquanto garante desempenho e usabilidade. Em seguida, foca na alternância de perfis e na listagem de serviços por categoria, com prioridade para segurança e escalabilidade. Por fim, o desenvolvimento se completa com a listagem de serviços por profissional e o painel dedicado a eles, assegurando a compatibilidade e a documentação do sistema.

1.2.4. Apresentação Final

O projeto DelBicos já conta com a **Tela de feed deslogado**, que oferece uma visão inicial dos serviços ou profissionais disponíveis, incentivando o acesso ou o registro. A **Tela de Login** permite aos usuários autenticar-se no sistema, e, em caso de falha, a **Tela de erro ao logar** é exibida com uma mensagem informativa para o usuário. Uma vez autenticado, o usuário acessa a **Tela de feed logado**, que proporciona uma experiência personalizada com serviços relevantes e navegação completa pelas funcionalidades do aplicativo.

Figura 01 - Tela de Feed

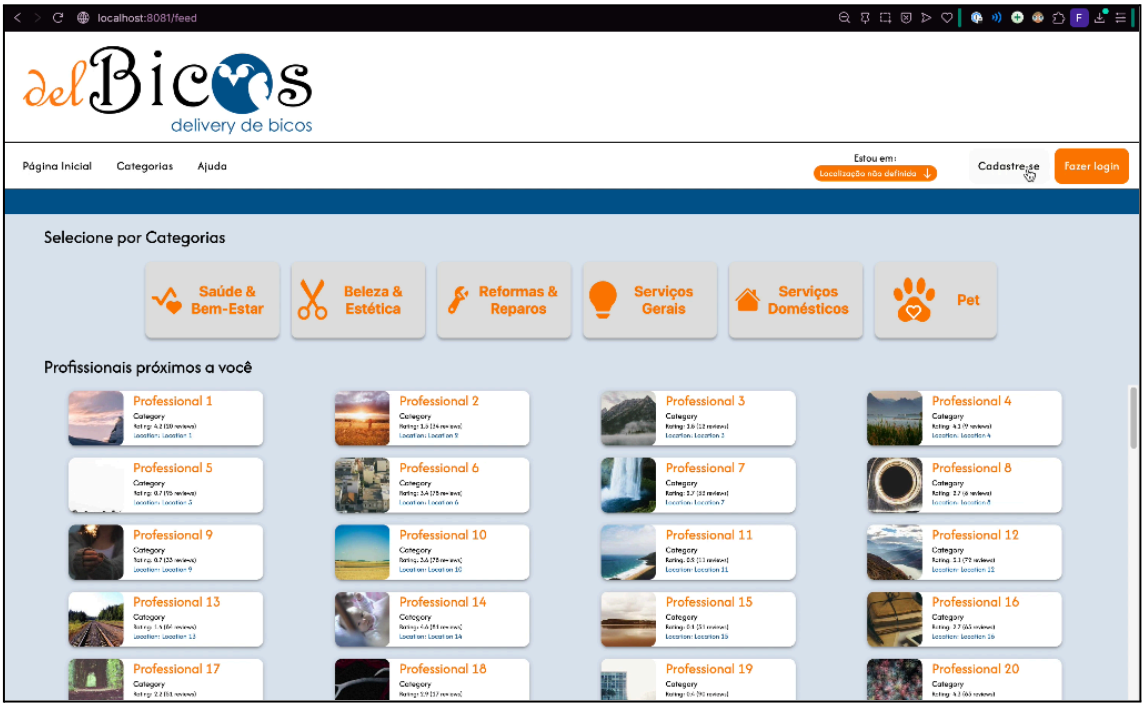


Figura 02 - Tela de Opções de Navegação

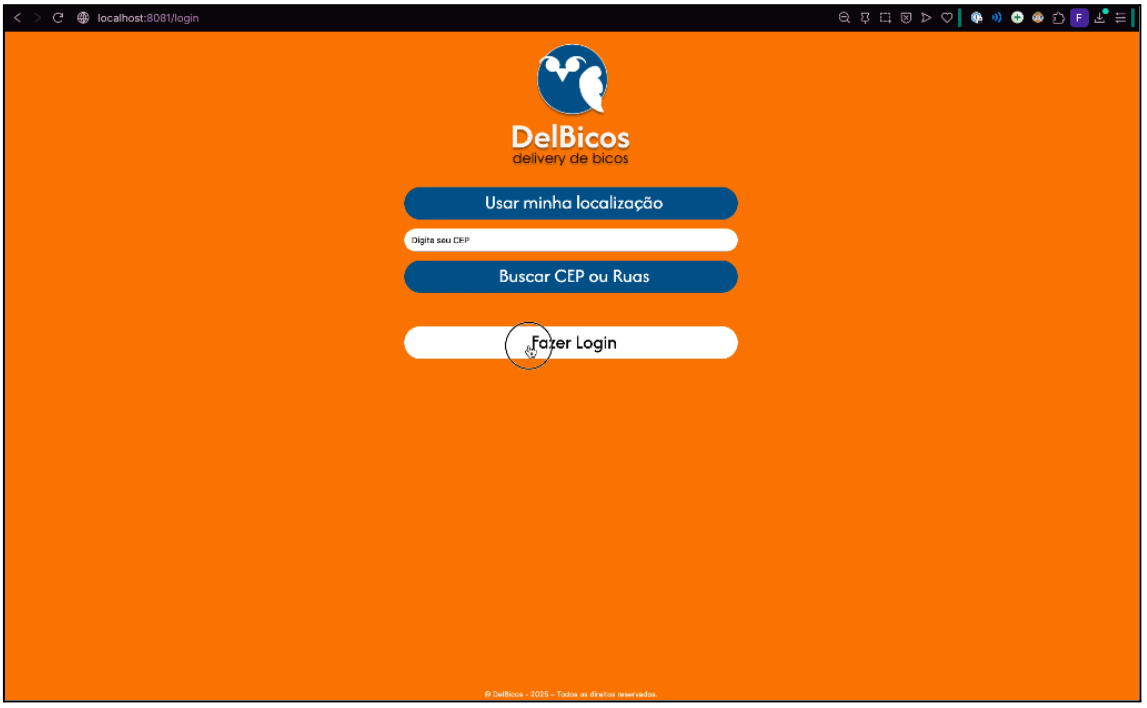


Figura 03 - Tela de Login

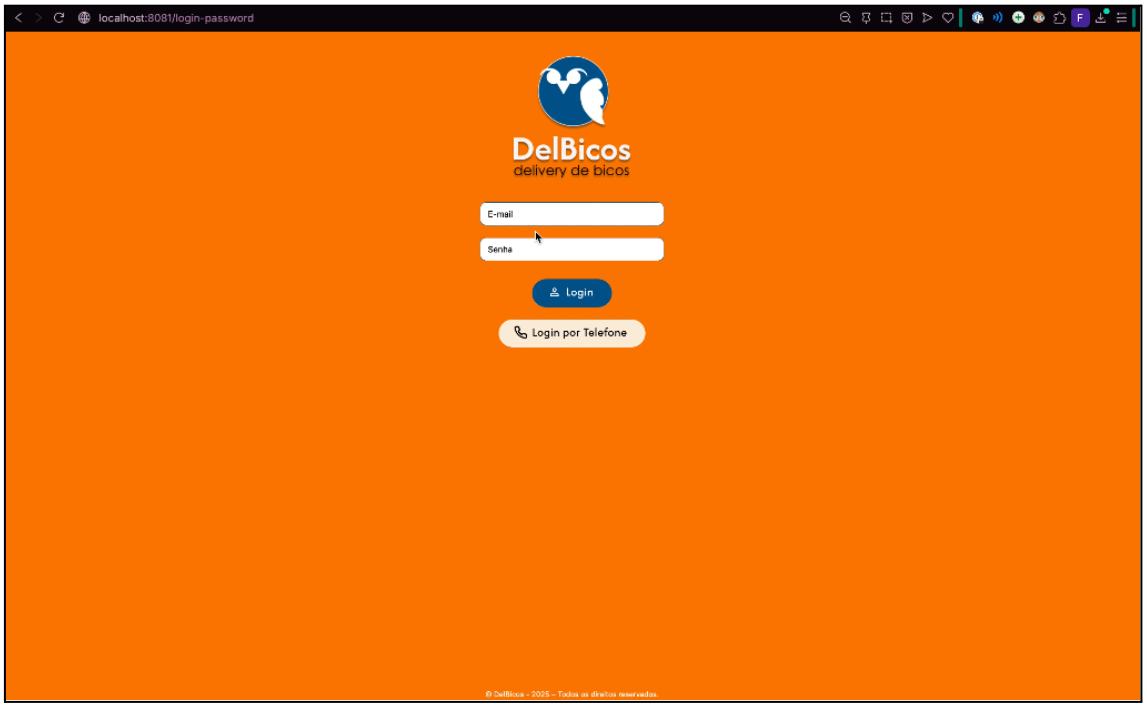


Figura 04 - Tela de Erro ao Logar sem dados

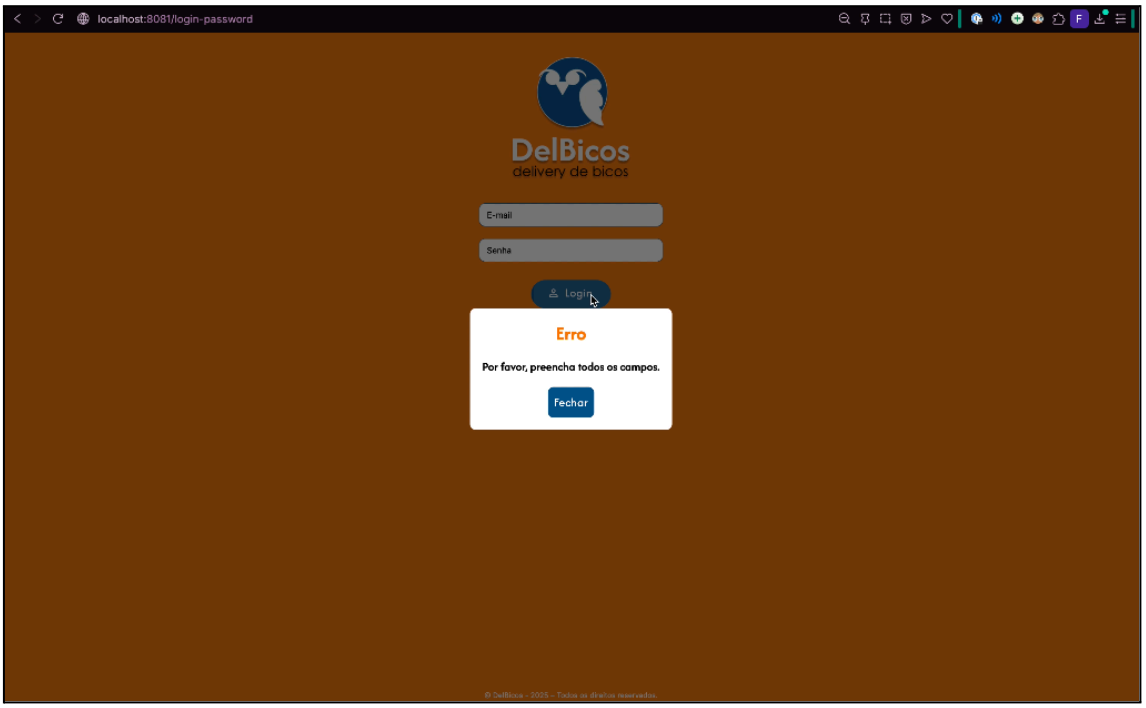


Figura 05 - Tela de Erro ao Logar com Dados Inválidos

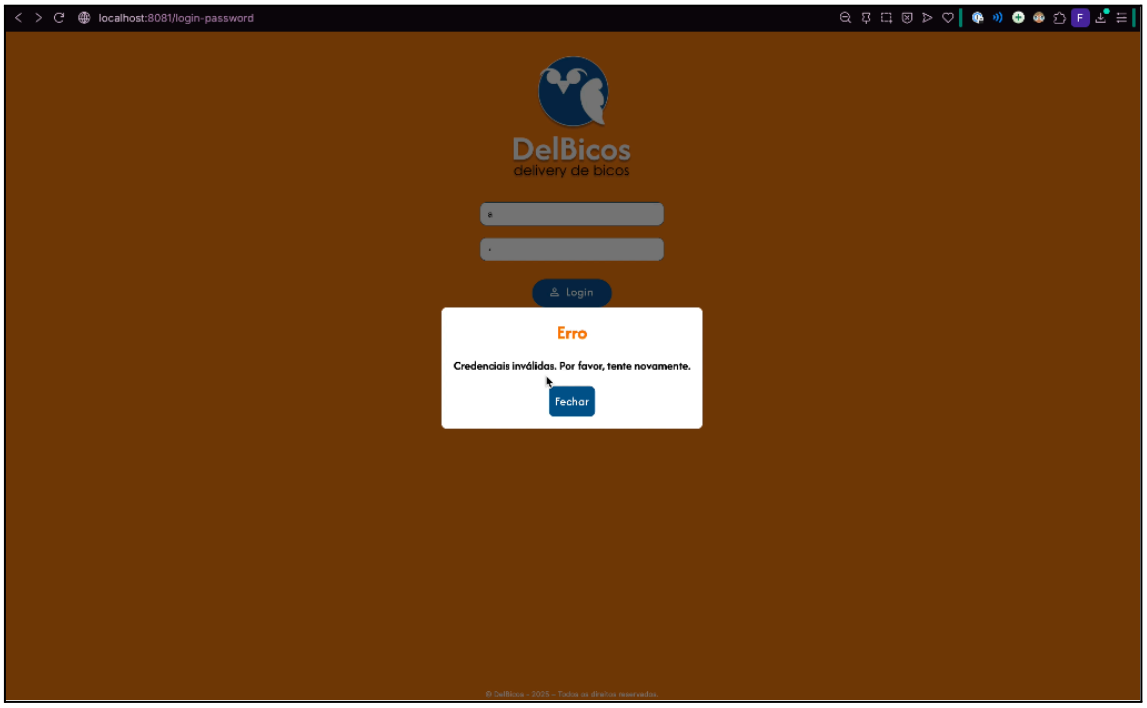
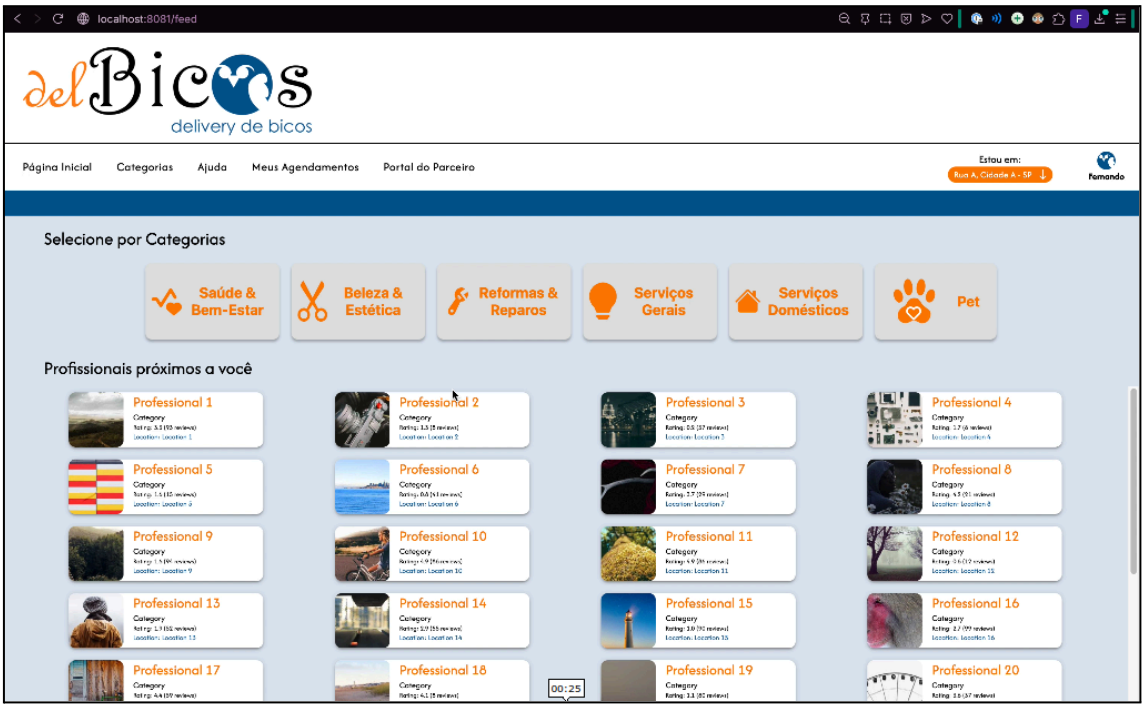


Figura 06 - Tela de Feed Logado



1.2.5. Backlogs

Este projeto é guiado por backlogs que detalham requisitos funcionais e não funcionais, organizados em três Sprints. As user stories desdobram esses requisitos, descrevendo funcionalidades da perspectiva do usuário, como login, cadastro e acesso a serviços específicos para clientes e profissionais. O desenvolvimento progride desde as funcionalidades básicas e críticas de desempenho, usabilidade e segurança na primeira fase, passando pela escalabilidade e gestão de perfis na segunda, até a listagem avançada de serviços e a compatibilidade na fase final, com a documentação permeando todo o processo.

Figura 07 - Backlog do Produto

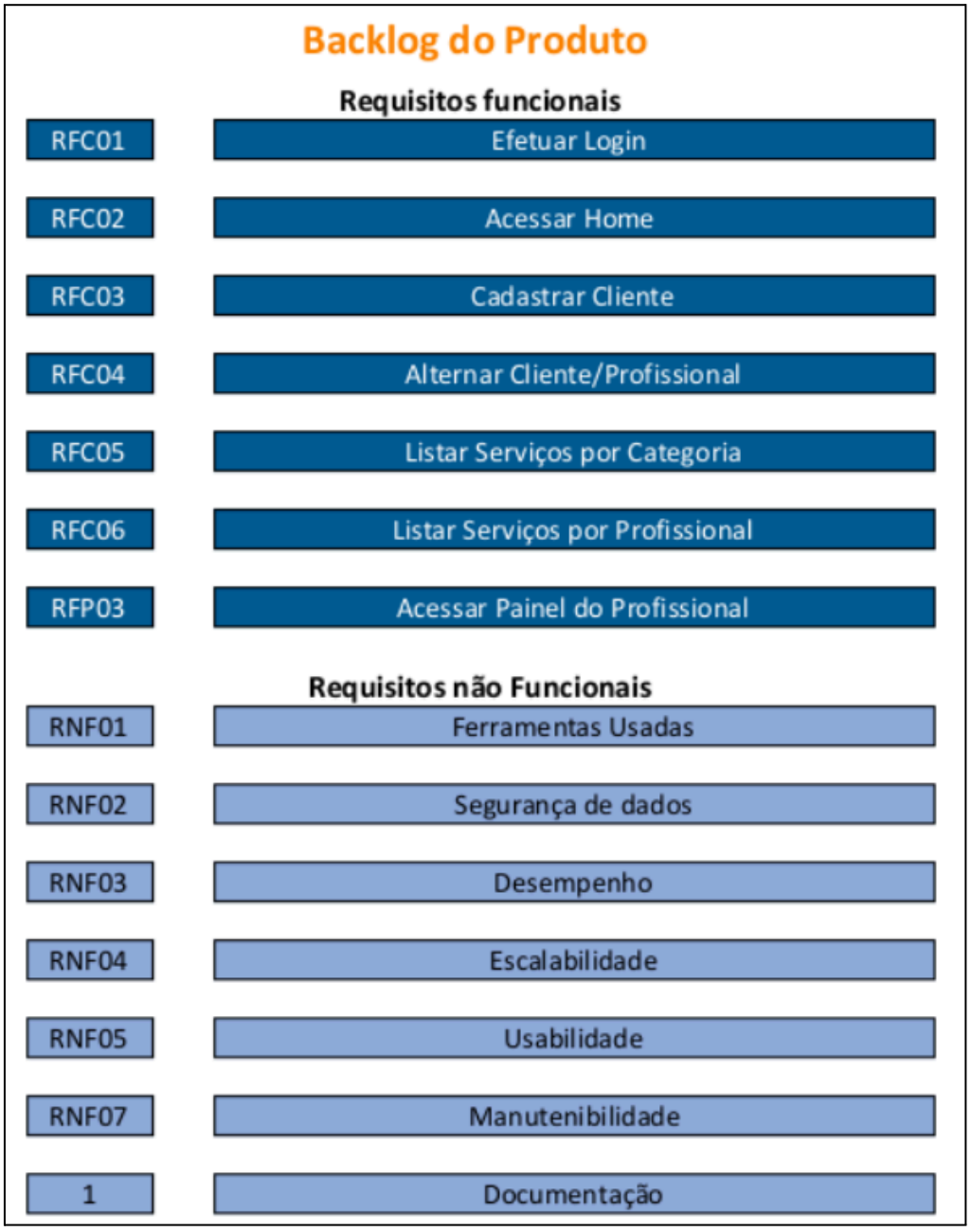
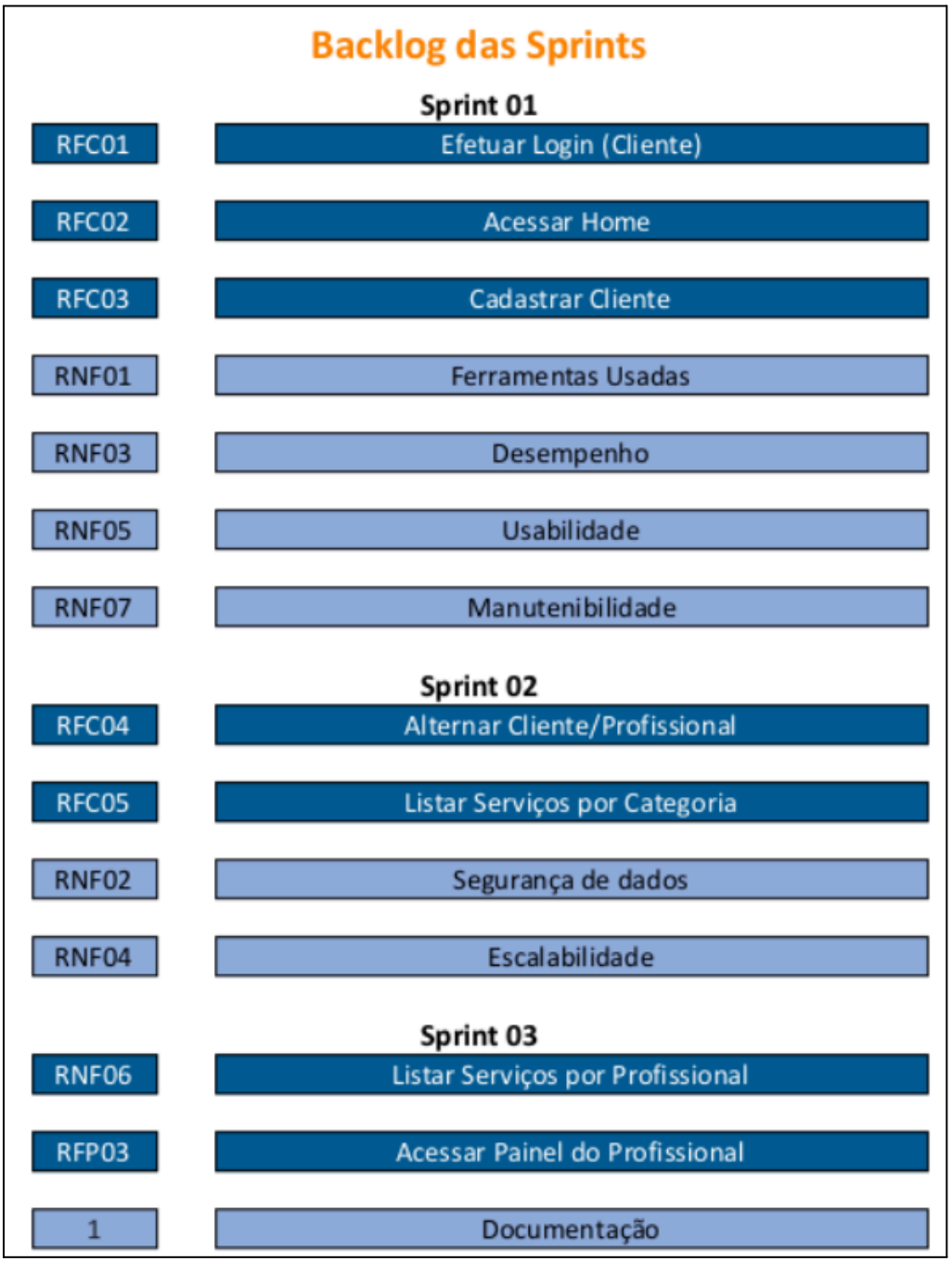


Figura 08 - Backlog das Sprints



1.2.6. User Stories

As user stories delineiam as funcionalidades do sistema sob a perspectiva de seus principais usuários. Para o cliente, o foco é na capacidade de se registrar, fazer login, navegar pela página inicial, e buscar, agendar e pagar por serviços, além de consultar seus compromissos. Já o profissional pode realizar login, cadastrar-se no sistema, acessar seu painel para verificar agendamentos e gerenciar tanto seus serviços quanto sua disponibilidade de horários. Há também uma user story dedicada ao desenvolvedor/mantenedor, que enfatiza a necessidade de uma documentação clara e acessível para facilitar a manutenção e futuras expansões do projeto.

Figura 09 - Sprints 01

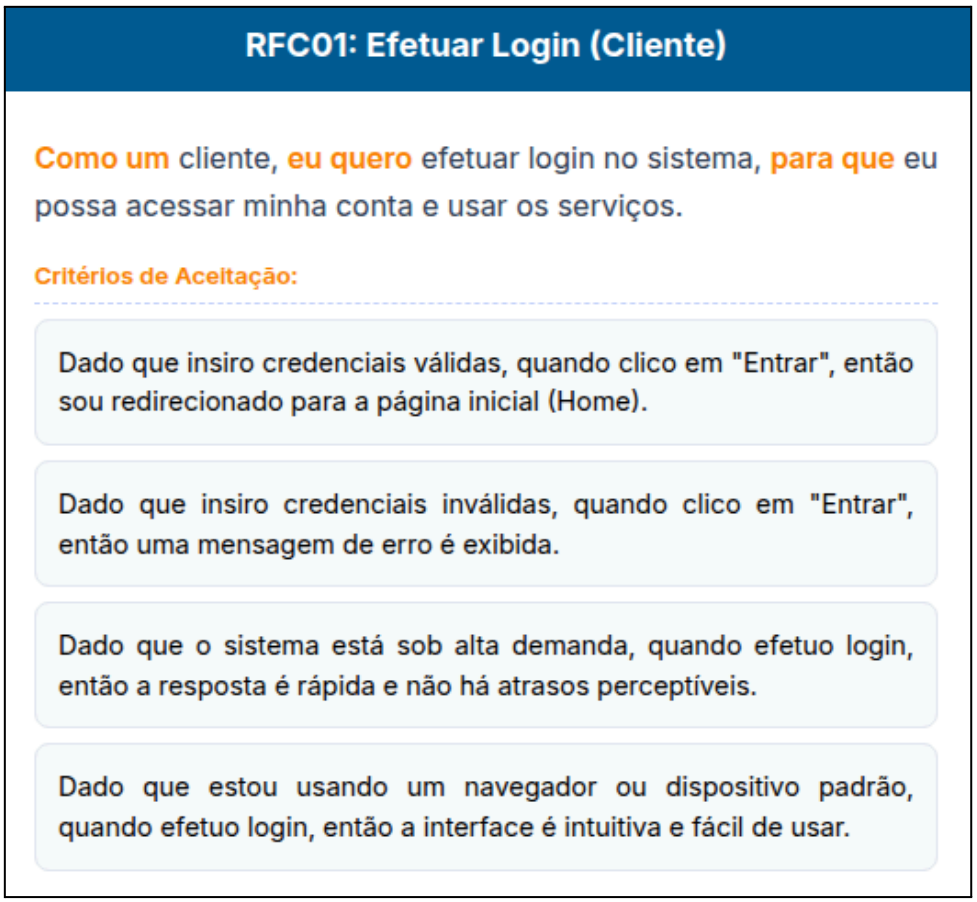


Figura 10 - Sprints 01-1

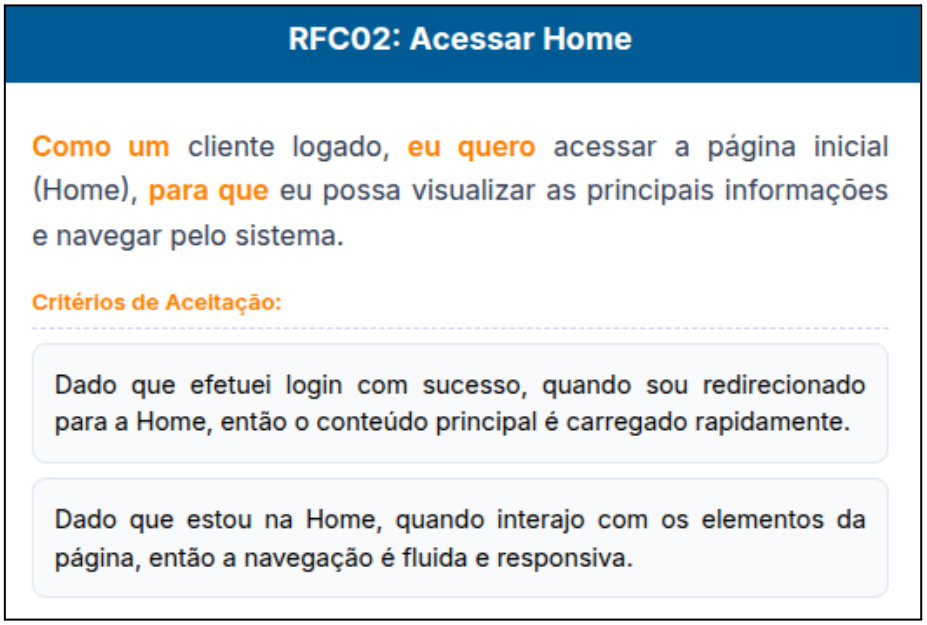


Figura 11 - Sprints 01-2

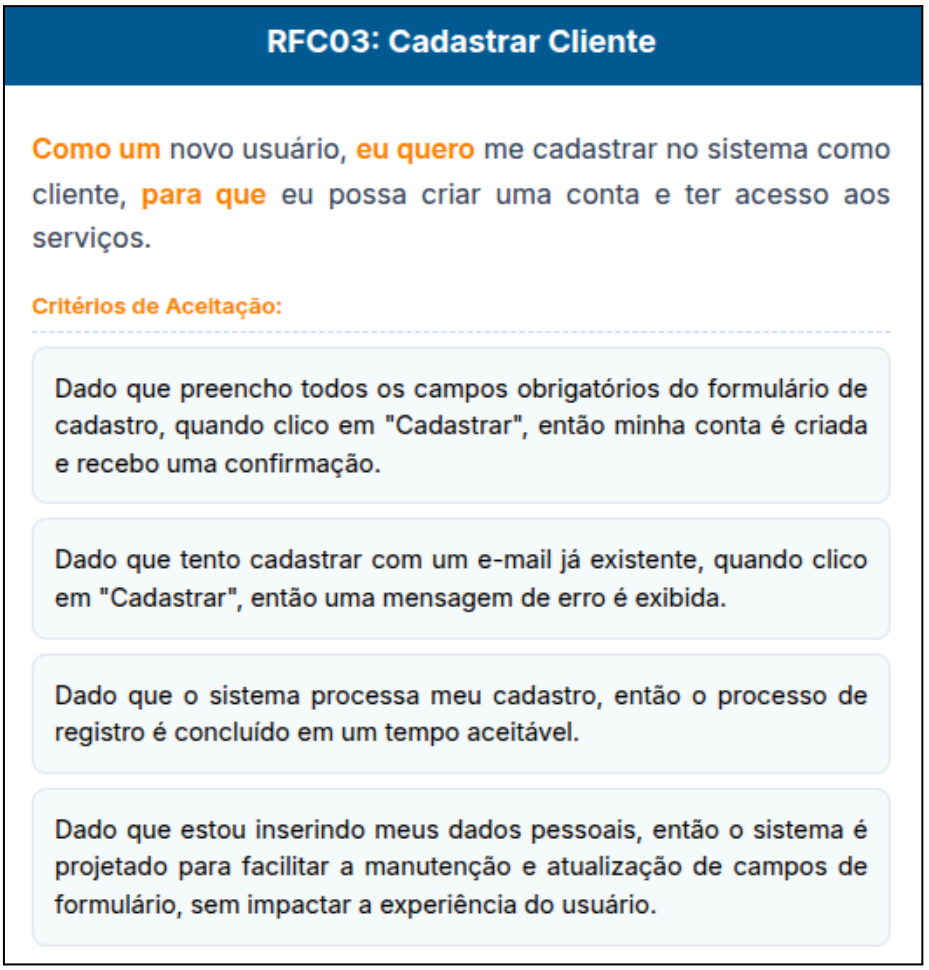


Figura 12 - Sprints 02

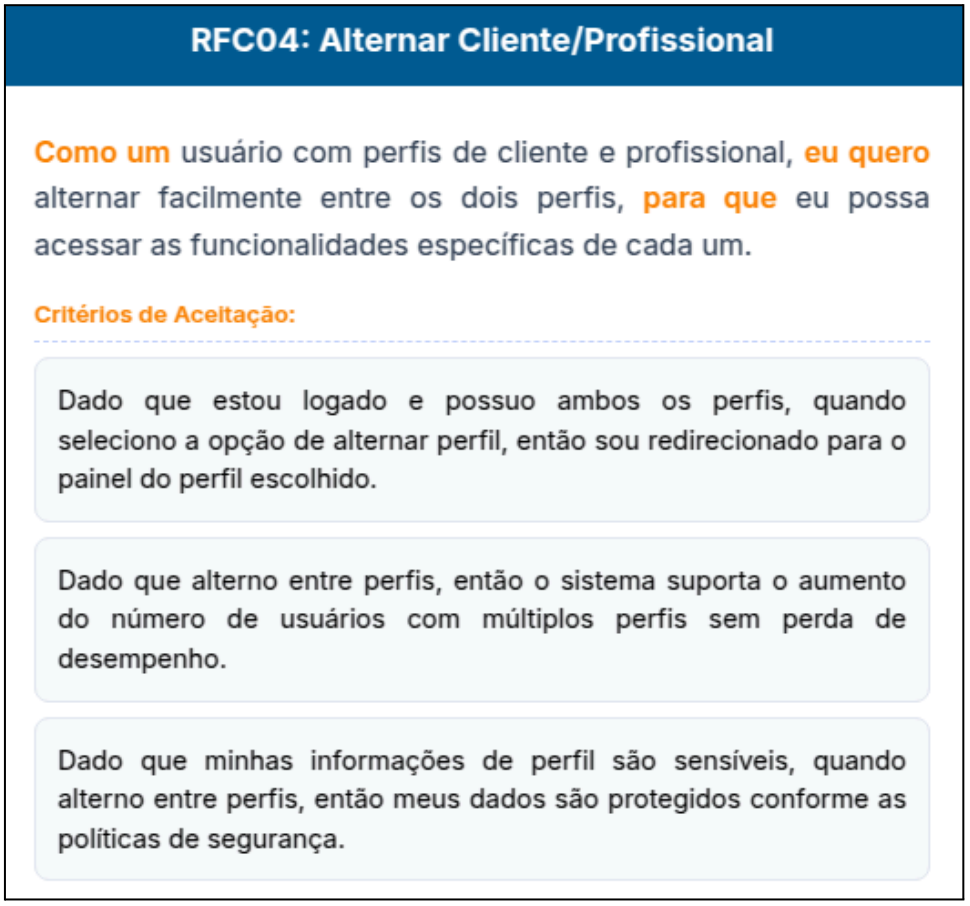


Figura 13 - Sprints 02-1

Figura 14 - Sprints 03

Figura 15 - Sprints 03-1

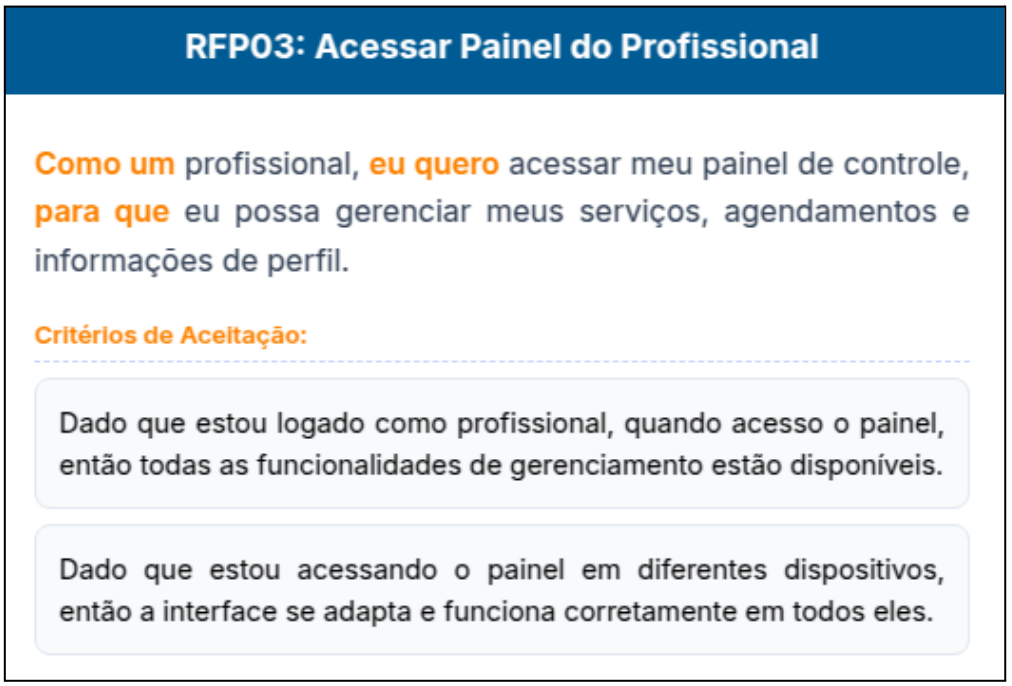
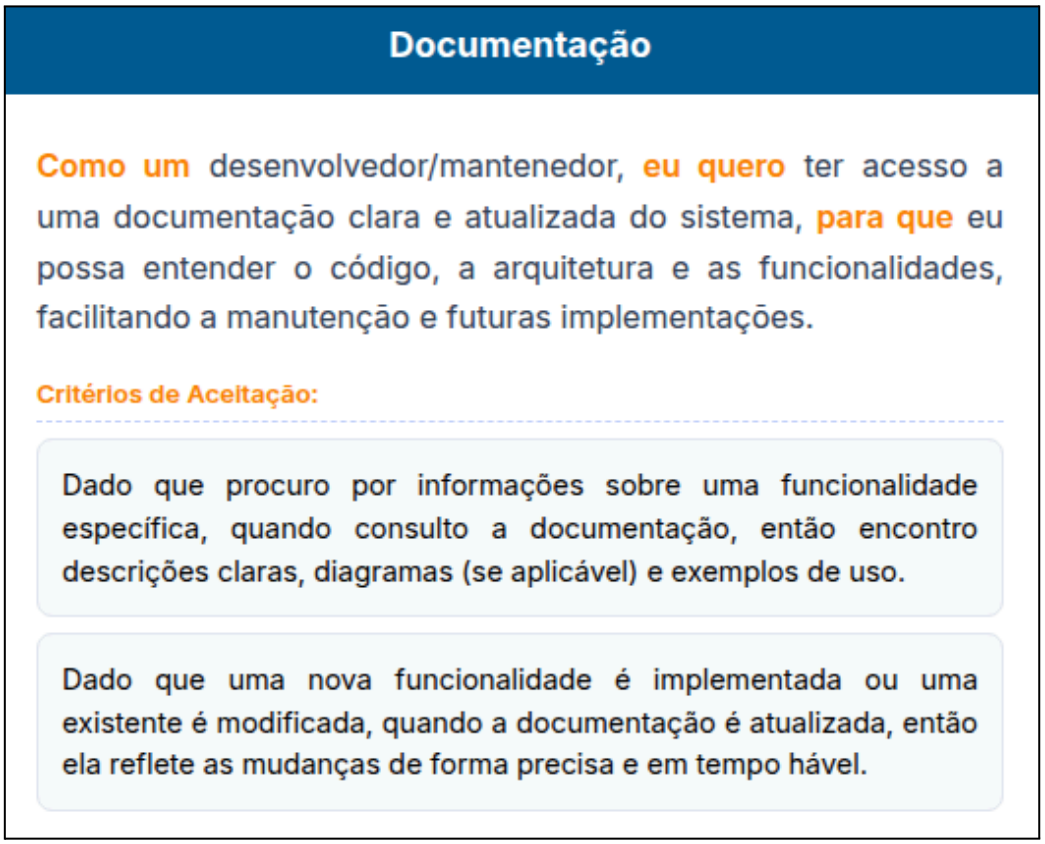


Figura 16 - Documentação



1.2.7. Protótipo

O protótipo do DelBicos no Figma serve como uma demonstração visual interativa das funcionalidades centrais do sistema. Ele apresenta a página inicial, a navegação para pesquisa de profissionais, os fluxos de cadastro e login de usuários, além das interações de agendamento de serviços e pagamentos. Embora a página do administrador esteja desenvolvida, as áreas dedicadas aos perfis de cliente e profissional ainda estão em fase de finalização.

https://www.figma.com/proto/1xZfYHmDwbr9aaqoh5ddxN/DelBicosV2?node-id=1364-969&t=tZpzceJg13szkrPk-1

1.2.8. Levantamento de Requisitos

Para realizar o levantamento dos requisitos, algumas informações foram empregadas para garantir que o projeto atinja todas as necessidades e expectativas da criação da mesma, tais como:

- **Pesquisas de Mercado** - Realização de estudos para entender o mercado atual de serviços prestados por profissionais autônomos;
- **Estudos da Concorrência** - Análise de empresas concorrentes para compreender seus serviços, estratégias de mercado, pontos fortes e fracos. Foram encontradas 4 empresas do ramo que proporcionam quase a mesma proposta deste projeto, das quais serviram para os estudos, dentre eles estão: GetNinjas[1], Beecos[2], Aqui Tem Bico[3] e Mary Help[4].

Utilizando uma combinação desses procedimentos, foi possível obter uma visão abrangente e detalhada dos requisitos necessários. Cada processo contribuiu de maneira única para entender melhor as necessidades do setor e identificar oportunidades e desafios, e desenvolver soluções eficazes e alinhadas com as expectativas da área.

1.2.9. Requisitos Funcionais

O sistema deve permitir que clientes realizem login, acessem a página inicial, se cadastrem, alternem para o perfil de profissional (se aplicável), listem e escolham serviços por categoria ou profissional, selecionem horários, efetuem pagamentos e consultem serviços agendados. Para os profissionais, as funcionalidades incluem login, cadastro, acesso ao painel, verificação de serviços agendados e alteração de suas tabelas de valores e disponibilidade. Já o administrador terá a capacidade de se autenticar e gerenciar clientes, profissionais, analisar chamados, controlar serviços realizados/pendentes e aprovar/desaprovar estornos.

---

[1] Disponível em: <https://www.getninjas.com.br/> Acesso em: Agosto, 2024
[2] Disponível em: <https://beecos.com.br/> Acesso em: Agosto, 2024
[3] Disponível em: <https://app.aquitembico.com.br/> Acesso em: Agosto, 2024
[4] Disponível em: <https://www.maryhelp.com.br/> Acesso em: Agosto, 2024

Quadro 1 - Requisitos Funcionais - Cliente

Requisitos Funcionais do Cliente		
Número de Requisito	Nome	Descrição
RFC01	Efetuar Login	O cliente pode fazer login no sistema.
RFC02	Acessar Home	O cliente pode acessar a página inicial após o login.
RFC03	Cadastrar Cliente	O cliente tem a opção de se cadastrar no sistema.
RFC04	Alternar Cliente/Profissional	O cliente pode alternar entre as visualizações de cliente e profissional, caso tenha múltiplos perfis.
RFC05	Listar Serviços por Categoria	O cliente pode listar serviços por categoria.
RFC06	Listar Serviços por Profissional	O cliente pode listar serviços oferecidos por um profissional específico.
RFC07	Escolher Horários e Datas do Profissional	O cliente pode escolher um horário e data para o serviço desejado com o profissional.
RFC08	Efetuar Pagamento	O cliente pode realizar o pagamento dos serviços.
RFC09	Consultar Serviços Agendados	O cliente pode consultar os serviços já agendados.

Quadro 2 - Requisitos Funcionais - Profissional

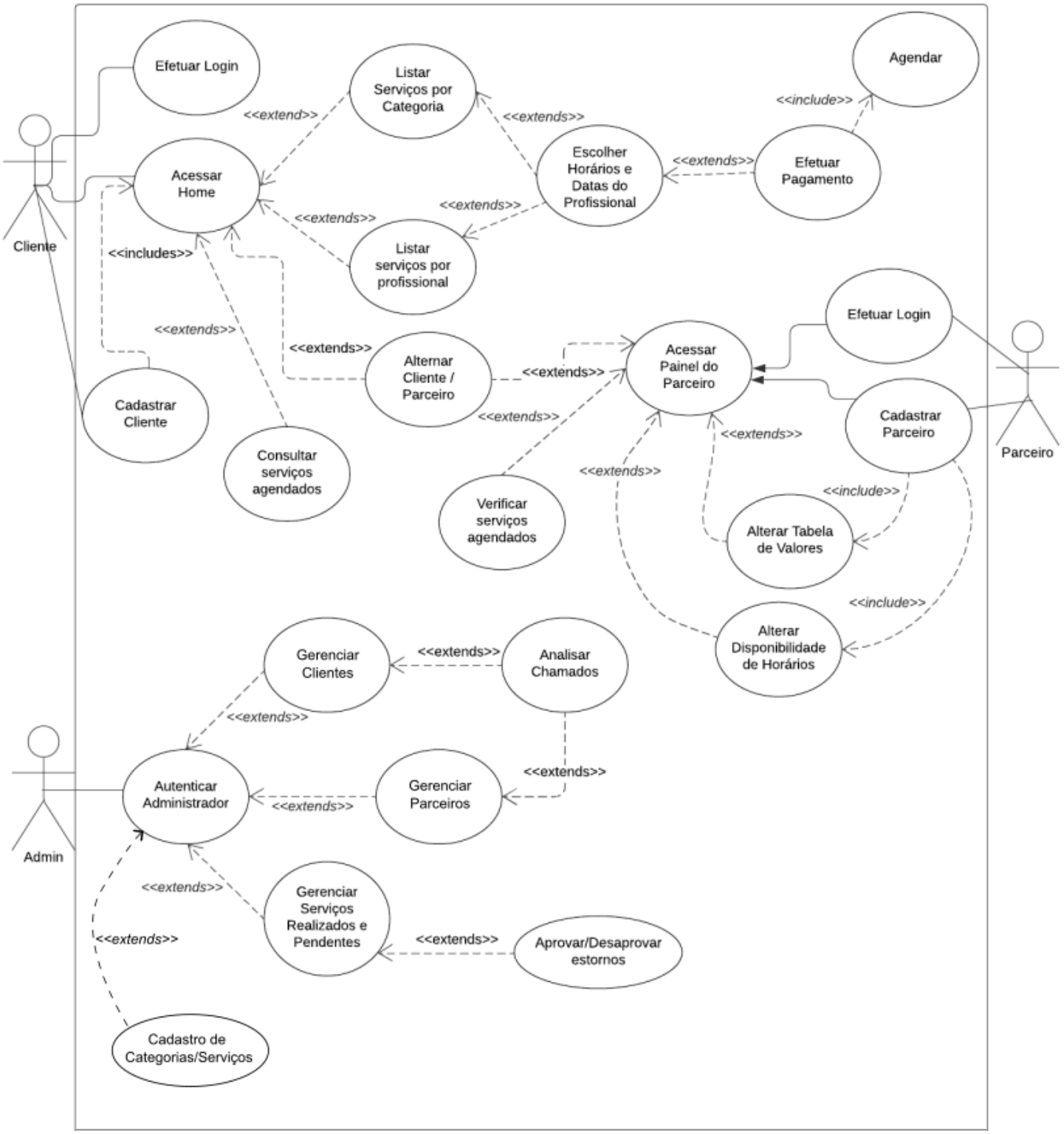
Requisitos Funcionais do Profissional		
Número de Requisito	Nome	Descrição
RFP01	Efetuar Login	O profissional pode fazer login no sistema.
RFP02	Cadastrar Profissional	O usuário tem a opção de se cadastrar no sistema como profissional.
RFP03	Acessar Painel do Profissional	O profissional pode acessar o painel com suas funcionalidades.
RFP04	Verificar Serviços Agendados	O profissional pode verificar os serviços já agendados por clientes.
RFP05	Alterar Tabela de Valores	O profissional pode alterar a tabela de valores de seus serviços.
RFP06	Alterar Disponibilidade de Horários	O profissional pode alterar sua disponibilidade de horários.

Quadro 3 - Requisitos Funcionais - Administrador

Requisitos Funcionais do Administrador		
Número de Requisito	Nome	Descrição
RFA01	Autenticar Administrador	O administrador pode se autenticar no sistema.
RFA02	Gerenciar Clientes	O administrador pode gerenciar o cadastro de clientes.
RFA03	Gerenciar Profissionais	O administrador pode gerenciar o cadastro de profissionais.
RFA04	Analisar chamados	O administrador pode analisar chamados de clientes e profissionais.
RFA05	Gerenciar Serviços Realizados e Pendentes	O administrador pode gerenciar os serviços realizados e os que ainda estão pendentes.
RFA06	Aprovar/Desaprovar Estornos	O administrador pode aprovar ou desaprovar pedidos de estorno.

1.2.10.Diagrama de Caso de Uso

Figura 17 - Diagrama de Caso de Uso



1.2.11. Requisitos Não Funcionais

O quadro abaixo descreve os requisitos de qualidade e restrições técnicas para um sistema. Os pontos chave incluem a segurança e o backup de dados, garantindo a proteção e recuperação de informações críticas. O sistema deve ser rápido, responsivo e escalável para suportar o crescimento de usuários e funcionalidades. A usabilidade é fundamental para uma experiência intuitiva e agradável, e o aplicativo precisa ser compatível com diversas plataformas. A manutenção do código é crucial para futuras atualizações e correções, enquanto a conformidade legal assegura a aderência às leis, especialmente em transações. Por fim, a eficiência energética é destacada para dispositivos móveis, e as ferramentas de desenvolvimento específicas são mencionadas como base tecnológica.

Quadro 4 - Requisitos Não Funcionais

Requisitos Não Funcionais		
Número de Requisitos	Nome	Descrição
RNF01	Ferramentas Usadas	Site desenvolvido em HTML5/CSS/SCRIPT/BOOTSTRAP
RNF02	Segurança de dados	Segurança de dados fornecidos pelos usuários
RNF03	Desempenho	O app deve ser rápido e responsivo, mesmo com um grande número de usuários simultâneos.
RNF04	Escalabilidade	A arquitetura do sistema deve suportar a expansão do número de usuários e a adição de novos recursos no futuro sem perda de desempenho.
RNF05	Usabilidade	A interface do usuário deve ser intuitiva e fácil de usar, proporcionando uma experiência agradável para todos os tipos de usuários, independentemente de sua familiaridade com a tecnologia.
RNF06	Compatibilidade	O aplicativo deve ser compatível com diferentes dispositivos e sistemas operacionais, como Android, iOS, e versões web.
RNF07	Manutenibilidade	O código e a infraestrutura do aplicativo devem ser projetados para facilitar a manutenção e a atualização, permitindo correções e melhorias sem impacto negativo na experiência do usuário.
RNF08	Conformidade Legal	O app deve estar em conformidade com as leis e regulamentações locais, especialmente no que diz respeito à proteção de dados e às transações financeiras.
RNF09	Eficiência Energética	O app deve ser otimizado para consumo mínimo de bateria, especialmente em dispositivos móveis.
RNF10	Backup e Recuperação	Implementar um sistema de backup regular dos dados para garantir que informações críticas possam ser recuperadas em caso de falha.

1.3. PROJETO DO SOFTWARE

1.3.1. Arquitetura da Aplicação

Para este projeto, o modelo arquitetural escolhido é o **Cliente-Servidor (Client-Server)**, complementado por uma **Arquitetura em Camadas (Layered Architecture)**. Esta combinação é ideal para sistemas web modernos, pois oferece modularidade, escalabilidade, fácil manutenção e clara separação de responsabilidades entre as diferentes partes do sistema.

Modelo Arquitetural Escolhido

➔ Cliente-Servidor

Neste modelo, a aplicação é dividida em duas partes principais: o **Cliente** (front-end) e o **Servidor** (back-end). Os clientes, que podem ser navegadores web ou aplicativos móveis, solicitam serviços ao servidor. O servidor processa essas requisições, interage com o banco de dados conforme necessário, e envia as respostas de volta ao cliente.

- **Cliente:** Responsável pela interface do usuário e pela interação direta com o utilizador (navegação, preenchimento de formulários, etc.).
- **Servidor:** Responsável por hospedar a lógica de negócio, gerenciar o acesso aos dados e processar as requisições dos clientes.

➔ Arquitetura em Camadas

Dentro do servidor, a arquitetura é organizada em camadas distintas, cada uma com um conjunto específico de responsabilidades, garantindo que as modificações em uma camada tenham impacto mínimo nas outras.

- **Rotas:** É a interface direta com o usuário. Inclui a lógica da interface do usuário (UI) e a renderização dos dados.
- **Middlewares & Controladores:** Contém as regras de negócio e a lógica central do sistema. É aqui que as operações como cadastro de

clientes/profissionais, agendamento de serviços, processamento de pagamentos e a gestão de usuários e chamados pelo administrador são realizadas. Esta camada interage com a Camada de Acesso a Dados para obter e persistir informações.

- **Models:** Atua como um intermediário entre a lógica de negócio e o banco de dados. É responsável por todas as operações de leitura, escrita, atualização e exclusão (CRUD) dos dados. Isso abstrai a lógica de negócio dos detalhes específicos do banco de dados.
- **Banco de Dados (Database Layer):** É onde todos os dados persistentes do sistema são armazenados. Isso inclui informações de clientes, profissionais, serviços, agendamentos, pagamentos, chamados e dados administrativos.

Justificativa

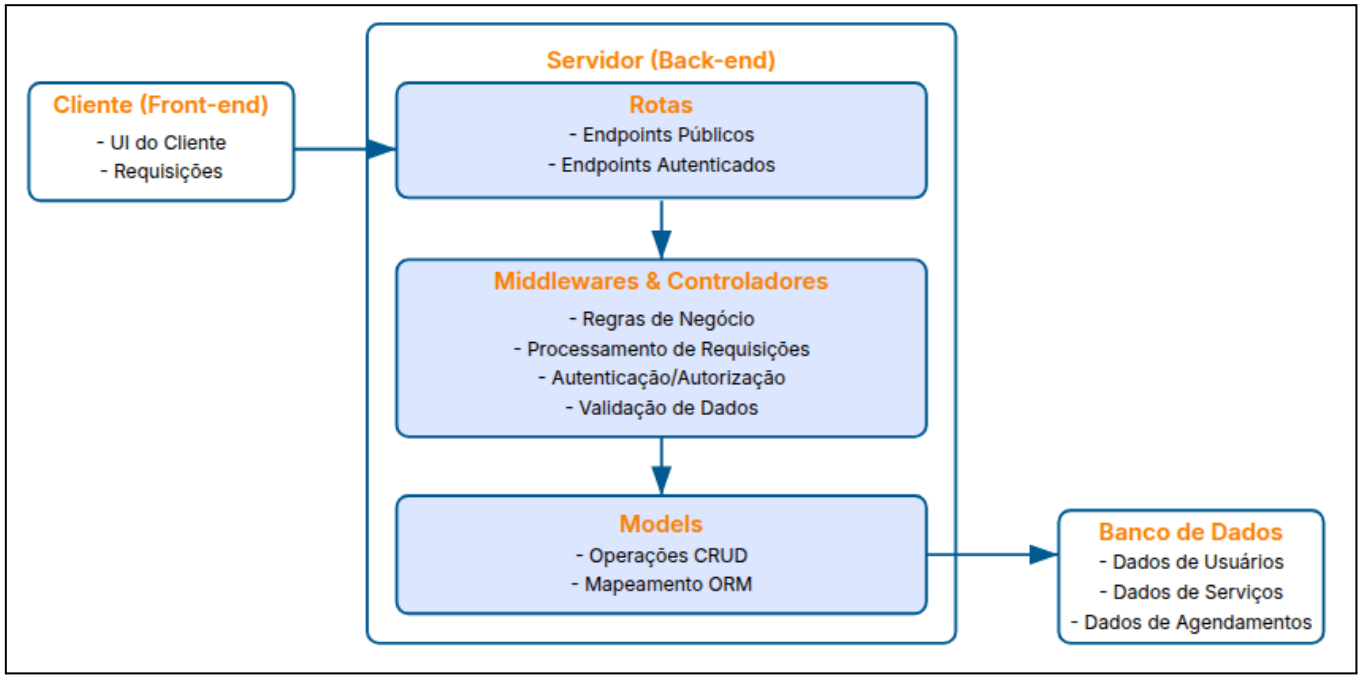
A escolha do modelo Cliente-Servidor com Arquitetura em Camadas oferece as seguintes vantagens:

- **Modularidade:** Cada camada tem uma responsabilidade bem definida, facilitando o desenvolvimento e a manutenção.
- **Escalabilidade:** Permite escalar diferentes partes do sistema independentemente (por exemplo, adicionar mais servidores de aplicação ou replicar o banco de dados).
- **Flexibilidade:** Facilita a substituição de tecnologias em uma camada sem afetar as outras.
- **Segurança:** Ajuda a isolar as camadas, tornando o sistema mais seguro ao controlar o fluxo de dados.

1.3.2. Diagrama Arquitetural Simplificado

Aqui está uma representação visual simplificada do modelo arquitetural:

Figura 18 - Diagrama Arquitetural Simplificado



No diagrama, as setas indicam o fluxo de comunicação e dependência entre as camadas, desde a requisição do cliente até a persistência dos dados e o retorno da resposta.

1.4. Tecnologias Utilizadas

Durante todo o semestre foram utilizadas várias tecnologias para o desenvolvimento do projeto por completo, desde os protótipos até o seguimento do site, todas as técnicas foram aprendidas em sala de aula e acompanhada por todos seus instrutores.

A implementação do site foi feita utilizando principalmente em HTML, CSS, Bootstrap e JavaScript permitindo a estruturação, estilização e adição de interatividade. Foi utilizado o framework Bootstrap, que possibilitou a criação de uma interface de usuário dinâmica e atrativa. Além disso, a ferramenta GitHub foi empregada para controle de versão e colaboração no código, assegurando

que todas as alterações fossem monitoradas e integradas de forma organizada. Combinando essas tecnologias, o site foi desenvolvido de maneira integrada, eficiente e alinhada com o projeto desenvolvido.

A lista completa das tecnologias utilizada segue abaixo:

- **Astah** é uma ferramenta de modelagem de software usada principalmente para criar diagramas UML (Unified Modeling Language) e outros tipos de diagramas que ajudam no planejamento e documentação de sistemas de software.[5]
- **Axios** é um cliente HTTP para comunicação a API backend[6]
- **Bootstrap** é uma framework de front-end que facilita o desenvolvimento de sites e aplicações web responsivas e modernas, fornecendo um conjunto de ferramentas pré-estilizadas em HTML, CSS e JavaScript.[7] https://getbootstrap.com/
- **BrModelo** é uma ferramenta de modelagem de banco de dados que facilita a criação de diagramas entidade-relacionamento (ER), auxiliando no design, documentação e análise de bancos de dados relacionais.[8]
- **Canvas** - O Modelo Canvas é uma ferramenta estratégica que permite visualizar, desenvolver e validar modelos de negócios de forma simplificada, usando um quadro dividido em nove blocos que representam os principais componentes de um negócio, como proposta de valor, segmentos de clientes, canais, e fontes de receita.
- **CSS** (Cascading Style Sheets) é uma linguagem usada para definir a aparência e o layout de páginas web, controlando elementos como cores, fontes, espaçamentos e posicionamentos.
- **Excel** é uma ferramenta de software da Microsoft usada para criar e manipular planilhas, realizar cálculos, analisar dados e gerar gráficos, amplamente utilizada em tarefas financeiras, contábeis e administrativas.[9]

---

[5] Disponível em: <https://astah.net/> Acesso em: Maio, 2025
[6] Disponível em: <https://axios-http.com/ptbr/docs/intro> Acesso em: Maio, 2025
[7] Disponível em: <https://getbootstrap.com/> Acesso em: Maio, 2025
[8] Disponível em: <http://www.sis4.com/> Acesso em: Maio, 2025
[9] Disponível em: <https://www.office.com/> Acesso em: Maio, 2025

- **Figma** é uma ferramenta de design colaborativa baseada na web, usada para criar interfaces de usuário, protótipos interativos e gráficos, permitindo a colaboração em tempo real entre designers e equipes.[10]
- **GIMP** é um software de edição de imagens gratuito e de código aberto, utilizado para retoque fotográfico, composição e criação de gráficos.[11]
- **GitHub** é uma plataforma de hospedagem e colaboração para desenvolvimento de software utilizando o controle de versão Git, facilitando o compartilhamento de código, colaboração em projetos e gerenciamento de código-fonte.[12]
- **Google Docs** é uma plataforma de processamento de texto baseada na web, oferecendo colaboração em tempo real, armazenamento na nuvem e acesso multiplataforma para edição de documentos, tudo de forma gratuita.[13]
- **Google Drive** é um serviço de armazenamento na nuvem que permite guardar arquivos online, acessá-los de qualquer lugar e compartilhá-los facilmente, integrado com outros serviços do Google.[14]
- **HTML** (Hypertext Markup Language) é a linguagem de marcação padrão para criação e estruturação de páginas web, utilizando tags para definir o conteúdo e a formatação dos elementos.
- **InkScape** é um software de código aberto para criação de gráficos vetoriais, oferecendo ferramentas robustas para desenho, manipulação de imagens e criação de ilustrações escaláveis.[15]
- **JavaScript** é uma linguagem de programação amplamente utilizada para desenvolvimento web, permitindo interatividade dinâmica em páginas, manipulação de conteúdo e comunicação assíncrona com servidores.
- **JWT** é um sistema de autenticação baseado em tokens para segurança de rotas.

---

[10] Disponível em: <https://www.figma.com/> Acesso em: Maio, 2025
[11] Disponível em: <https://www.gimp.org/> Acesso em: Maio, 2025
[12] Disponível em: <https://github.com/> Acesso em: Maio, 2025
[13] Disponível em: <https://docs.google.com/> Acesso em: Maio, 2025
[14] Disponível em: <https://drive.google.com/drive/u/0/my-drive> Acesso em: Maio, 2025
[15] Disponível em: <https://inkscape.org/> Acesso em: Maio, 2025

- **Material UI** é uma biblioteca de componentes React de código aberto que implementa o Material Design do Google. É abrangente e pode ser usada imediatamente em produção. [16]
- **MongoDB** é um banco de dados NoSQL utilizado para armazenamento das informações.[17]
- **Node.js** é um ambiente de execução JavaScript que permite o desenvolvimento do servidor. [18]
- **Penpot** é um editor de gráfico de vetor e prototipagem de projetos de design baseado principalmente no navegador web, dispõe de cliente desktop para GNU/Linux, macOS e Windows.[19]
- **Photoshop** é um software poderoso de edição de imagens e design gráfico, amplamente utilizado por profissionais para manipulação, retoque e criação de arte digital.[20]
- **React Native** é um framework de código aberto usado para desenvolver aplicativos para Android, Android TV, iOS, macOS, tvOS, Web, Windows, e UWP, permitindo que os desenvolvedores usem a estrutura React juntamente com os recursos nativos de cada plataforma.[21]
- **VsCode** (Visual Studio Code) é um editor de código-fonte leve e altamente extensível desenvolvido pela Microsoft, amplamente utilizado por desenvolvedores para programação e desenvolvimento de software.[22]
- **Word** é um software de processamento de texto da Microsoft, utilizado para criação, edição e formatação de documentos diversos, oferecendo ferramentas avançadas de escrita e design.[23]

[16] Disponível em: <https://mui.com/material-ui/> Acesso em: Maio, 2025
[17] Disponível em: <https://www.mongodb.com/> Acesso em: Maio, 2025
[18] Disponível em: <https://nodejs.org/pt> Acesso em: Maio, 2025
[19] Disponível em: <https://penpot.app/> Acesso em: Maio, 2025
[20] Disponível em: <https://www.adobe.com/br/products/photoshop.html> Acesso em: Maio, 2025
[21] Disponível em: <https://reactnative.dev/> Acesso em: Maio, 2025
[22] Disponível em: <https://code.visualstudio.com/> Acesso em: Maio, 2025
[23] Disponível em: <https://www.office.com/> Acessi em: Maio, 2025

1.5. Integração e Papéis da Equipe

Fernando Chibli, Product Owner e Fullstack, definiu a visão do produto, gerenciou o backlog do produto, priorizou as funcionalidades com base no valor de negócio e manteve a comunicação constante com as partes interessadas para garantir que o produto atendesse às suas necessidades, tomou decisões estratégicas que guiaram o desenvolvimento.

Douglas Wenzel, Scrum Master e Fullstack, facilitou as reuniões diárias (Daily Scrums), promoveu um ambiente de colaboração e auto-organização, manteve a equipe focada nos objetivos, ajudou a resolver conflitos e otimizou os fluxos de trabalho, resultando em um aumento significativo na produtividade.

Gustavo Lopes, Desenvolvedor Fullstack, projetou e implementou a arquitetura do servidor, bancos de dados e APIs, garantindo a segurança e a escalabilidade do sistema, desenvolveu a lógica de negócios robusta.

Isabel Maito, Desenvolvedora Fullstack, criou um design responsivo e intuitivo, otimizou o desempenho da interface, resultando em uma experiência de usuário aprimorada. Responsável pela implementação da interface do usuário, garantindo a usabilidade e a experiência do usuário.

Quadro 5 - Equipe Técnica

Nome	Função
Fernando Chibli	***Product Owner & FullStack***
Douglas Wenzel	***Scrum Master & FullStack***
Gustavo Ferreira	***Dev Team FullStack***
Isabel Maito	***Dev Team FullStack***

2. REFERÊNCIAS

ATLASSIAN. **User Stories no Agile**. Disponível em: <https://www.atlassian.com/br/agile/project-management/user-stories> Acesso em: Maio, 2025.

IBM. **What is client-server architecture?** Disponível em: <https://www.ibm.com/topics/client-server-architecture> Acesso em: Junho, 2025.

MICROSOFT AZURE. **N-tier architecture style**. Disponível em: <https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier> Acesso em: Junho, 2025.

MOUNTAIN GOAT SOFTWARE. **User Stories**. Disponível em: <https://www.mountaingoatsoftware.com/agile/user-stories> Acesso em: Maio, 2025.

SCRUM.ORG. **What is Scrum?** Disponível em: <https://www.scrum.org/resources/what-is-scrum> Acesso em: Maio, 2025.