**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC Minas Virtual**

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

Projeto Integrado

Relatório Técnico

Gestão Glamour

Wagner Martins Santana Vieira

Belo Horizonte

Outubro de 2023.

# Projeto Integrado

**Sumário**

[Projeto Integrado 3](#_Toc147859821)

[1. Cronograma de Trabalho 5](#_Toc147859822)

[2. Introdução 6](#_Toc147859823)

[3. Definição Conceitual da Solução 9](#_Toc147859824)

[3.1 Diagrama de Casos de Uso 10](#_Toc147859825)

[3.2 Requisitos Funcionais 11](#_Toc147859826)

[3.3 Requisitos Não-funcionais 12](#_Toc147859827)

[4. Protótipo Navegável do Sistema 13](#_Toc147859828)

[5. Diagrama de Classes de Domínio 14](#_Toc147859829)

[6. Arquitetura da Solução 15](#_Toc147859830)

[6.1 Padrão Arquitetural 15](#_Toc147859831)

[6.2 C4 model - Diagrama de Contexto 17](#_Toc147859832)

[7. Frameworks de Trabalho 18](#_Toc147859833)

[8. Estrutura Base do Front End 19](#_Toc147859834)

[9. Modelo Relacional ou Projeto de Banco de Dados NoSQL 19](#_Toc147859835)

[10. Plano de Testes 20](#_Toc147859836)

[11. Apropriação de Horas no Projeto 23](#_Toc147859837)

[12. Código da Aplicação 24](#_Toc147859838)

[13. Avaliação Retrospectiva 25](#_Toc147859839)

[13.1 Objetivos Estimados 25](#_Toc147859840)

[13.2 Objetivos Alcançados 26](#_Toc147859841)

[13.2 Lições aprendidas 27](#_Toc147859842)

[14. Agradecimentos 27](#_Toc147859843)

**Lista de Ilustrações**

[**Figura 1** - Diagrama casos de uso. 10](#_Toc147660354)

[**Figura 2** - Utilização figma. 13](#_Toc147660355)

[**Figura 3** - Diagrama de classes de domínio. 14](#_Toc147660356)

[**Figura 4** - Estrutura do projeto 16](#_Toc147660357)

[**Figura 5** - Diagrama de contexto. 18](#_Toc147660358)

[**Figura 6** - Relatório jacoco 23](#_Toc147660359)

## Cronograma de Trabalho

|  |  |  |  |
| --- | --- | --- | --- |
| **Datas** | | **Atividade / Tarefa** | **Produto / Resultado** |
| **De** | **Até** |
| 01 / 07 / 23 | 15 / 07 /23 | 1.Definição Conceitual | Criar o diagrama de casos de uso.  Listar os requisitos funcionais.  Identificar os requisitos não-funcionais. |
| 15 / 07 / 23 | 30 / 07 / 23 | 2.Protótipo Navegável do Sistema | Desenvolver um protótipo navegável da aplicação.  Criar telas básicas para a agenda e o financeiro. |
| 30 / 07 / 23 | 10 / 08 / 23 | 3.Diagrama de Classes de Domínio | Modelar o diagrama de classes de domínio para representar os principais objetos da aplicação. |
| 10 / 08 / 23 | 16 / 08 / 23 | 4.Arquitetura da Solução | Definir o padrão arquitetural a ser utilizado.  Criar o diagrama de contexto usando o C4 model. |
| 17 / 08 / 23 | 23 / 08 / 23 | 5.Frameworks de Trabalho | Escolher e configurar os frameworks necessários para o desenvolvimento |
| 24 / 08 / 23 | 30 / 08 / 23 | 6.Estrutura Base do Front End | Criar a estrutura base do front end da aplicação |
| 31 / 08 / 23 | 06 / 09 / 23 | 7.Modelo Relacional ou Projeto de Banco de Dados NoSQL | Definir o modelo de banco de dados relacional ou estrutura para banco de dados |
| 07 / 09 / 23 | 24 / 09 / 23 | 8.Desenvolvimento das Funcionalidades | Implementar a funcionalidade da agenda e financeiro e relatórios.  Conectar o sistema ao banco de dados. |
| 25 / 09 / 23 | 02 / 10 / 23 | 9.Plano de Testes | Elaborar um plano de testes para garantir a qualidade do software.  Realizar testes unitários para as principais funcionalidades. |
| 03 / 10 / 23 | 06 / 10 / 23 | 11.Avaliação Retrospectiva | Revisar o progresso do projeto.  Verificar se todos os objetivos foram alcançados. |
| 06 / 10 / 23 | 08 / 10 / 23 | 12.Ajustes e Finalização | Realizar os ajustes necessários na aplicação.  Preparar o sistema para o lançamento. |
| 09 / 10 / 23 | 10 / 10 / 23 | 13.Entrega | Disponibilizar todo o projeto. |

## Introdução

O segmento de salões de beleza desempenha um papel significativo na indústria de cuidados pessoais, atendendo a uma vasta clientela que busca serviços variados, desde cortes de cabelo e tratamentos estéticos até serviços de manicure e maquiagem. Com a crescente demanda por serviços de beleza e bem-estar, a eficiência operacional e a organização interna tornam-se fatores críticos para o sucesso e a competitividade desses estabelecimentos.

Este trabalho apresenta o desenvolvimento de um sistema de gestão específico para salões de beleza, visando otimizar suas operações e proporcionar uma experiência mais agradável tanto para os clientes como para os profissionais do salão. O objetivo principal do sistema é simplificar e centralizar o gerenciamento das atividades diárias, englobando áreas cruciais como agendamento de serviços e controle financeiro.

Um dos principais desafios enfrentados pelos salões de beleza é a gestão de agendamentos, que muitas vezes pode ser caótica e sujeita a conflitos de horários, levando à insatisfação dos clientes e à perda de receita. Além disso, o controle financeiro muitas vezes é complexo, dificultando a visualização do lucro real e a identificação de oportunidades de crescimento.

Para abordar essas questões, este sistema propõe uma solução abrangente que permitirá aos salões de beleza gerenciar sua agenda de forma eficiente, agendando e reagendando compromissos com facilidade, e possibilitando o controle detalhado das finanças, incluindo registros de receitas e despesas.

Ao fornecer ferramentas intuitivas e eficazes, o sistema visa liberar os proprietários e funcionários de tarefas administrativas repetitivas, permitindo que se concentrem no atendimento ao cliente e na oferta de serviços de qualidade, elevando, assim, a satisfação dos clientes e a reputação do salão.

Ao longo deste trabalho, serão apresentados os requisitos funcionais e não-funcionais do sistema, a arquitetura proposta, bem como o desenvolvimento do protótipo navegável do sistema, que permitirá visualizar como as funcionalidades serão implementadas na interface do usuário.

Dessa forma, o sistema de gestão para salão de beleza busca alavancar a tecnologia e a automação para aprimorar a eficiência operacional desses estabelecimentos, promovendo um ambiente organizado e profissional, capaz de atender às demandas do mercado competitivo de beleza e bem-estar.

Para contextualizar o presente trabalho, é essencial considerar o cenário em ascensão da indústria de beleza e cuidados pessoais, que tem sido impulsionada pela crescente demanda por serviços de beleza e estética. Os salões de beleza desempenham um papel crucial nesse contexto, oferecendo uma ampla gama de tratamentos e serviços para atender às necessidades dos clientes em busca de cuidados pessoais e bem-estar. No entanto, muitos desses estabelecimentos enfrentam desafios na organização interna, especialmente no gerenciamento de agendamentos e controle financeiro. A falta de uma solução tecnológica adequada pode resultar em conflitos de horários e perda de receita.

O problema que o projeto se propõe a resolver é a ineficiência na gestão dos salões de beleza, que muitas vezes enfrentam desafios na organização de agendamentos e no controle financeiro. A ausência de um sistema especializado pode levar a conflitos de horários, cancelamentos de última hora e dificuldades na visualização do fluxo de caixa, afetando diretamente a satisfação dos clientes e a rentabilidade do negócio. Além disso, a falta de uma visão clara do desempenho financeiro pode prejudicar a tomada de decisões estratégicas e dificultar o crescimento sustentável do salão.

A motivação para resolver o problema da gestão ineficiente dos salões de beleza é abrangente e justifica-se em diversas frentes. Sob o aspecto acadêmico, este projeto proporciona a oportunidade de desenvolver uma solução tecnológica inovadora, aplicando conceitos avançados de desenvolvimento de software ao setor em crescimento. Mercadologicamente, o sistema de gestão busca atender à demanda real do mercado, diferenciando os salões por meio de uma administração eficiente que pode levar à fidelização de clientes e aumento da rentabilidade. Socialmente, almeja-se elevar a qualidade dos serviços prestados, melhorando a experiência do cliente e proporcionando maior satisfação aos profissionais do salão. Com a expectativa de trazer economia de recursos, agilidade nas decisões e impactos positivos na qualidade de vida, a relevância deste trabalho é evidente, promovendo a excelência dos salões de beleza e contribuindo para o seu crescimento e sucesso no mercado competitivo de beleza e bem-estar.

O objetivo geral deste trabalho é desenvolver um sistema de gestão para salões de beleza que proporcione uma administração eficiente e organizada, visando otimizar o agendamento de serviços e o controle financeiro, a fim de aprimorar a experiência do cliente e fortalecer a competitividade desses estabelecimentos no mercado de beleza e bem-estar.

Os objetivos específicos são:

* Identificar os requisitos funcionais do sistema: Esta etapa envolve uma análise aprofundada das necessidades dos salões de beleza, com o objetivo de mapear e documentar os requisitos funcionais essenciais. Isso inclui entender as operações diárias, as interações com clientes, o fluxo de agendamentos, a gestão financeira e outros aspectos cruciais do negócio.
* Projetar a arquitetura do sistema: A arquitetura do sistema será elaborada de forma cuidadosa, definindo a estrutura geral, a disposição dos componentes e a interação entre eles. Serão estabelecidas as interfaces com o usuário, os mecanismos de comunicação e a lógica de negócios, garantindo uma base sólida para o desenvolvimento futuro.
* Desenvolver o protótipo navegável: A criação de um protótipo navegável permitirá visualizar antecipadamente a aparência e a usabilidade da aplicação. Isso auxiliará na validação das ideias, na coleta de feedback dos usuários e na identificação de melhorias necessárias, garantindo que a solução atenda às expectativas.
* Efetuar o desenvolvimento do sistema de gestão: Nesta fase, ocorrerá a implementação prática das funcionalidades principais, como o agendamento de serviços, o controle financeiro e a geração de relatórios. Serão aplicadas as melhores práticas de desenvolvimento, visando à eficiência, ao desempenho e à facilidade de manutenção do sistema.
* Realizar testes e validação do sistema: Uma bateria completa de testes será conduzida para assegurar a qualidade do sistema. Isso envolverá testes de funcionalidade, desempenho e segurança. Além disso, a validação será realizada em conjunto com os usuários reais, garantindo que o sistema atenda às suas necessidades e expectativas de maneira eficaz.

## Definição Conceitual da Solução

A abordagem proposta para esta solução busca uma transformação profunda na maneira como os salões de beleza operam e interagem com seus clientes. O desenvolvimento de uma aplicação web altamente sofisticada, construída com tecnologias de última geração, representa um avanço significativo na modernização desse setor.

A escolha deliberada do framework Spring Boot e da linguagem de programação Java para a implementação do sistema oferece inúmeras vantagens, desde a robustez até a facilidade de manutenção. Essa decisão estratégica visa garantir uma base sólida para a construção de uma aplicação que atenda às necessidades exclusivas dos salões de beleza, permitindo uma gestão eficiente e uma experiência excepcional para os clientes.

A arquitetura de micro serviços, amplamente reconhecida por sua flexibilidade e escalabilidade, desempenhará um papel crucial no desdobramento da solução. A modularidade inerente aos micros serviços permite a divisão dos componentes em unidades independentes, simplificando a complexidade e facilitando a manutenção contínua. Além disso, essa abordagem proporciona um alto grau de adaptabilidade, permitindo que a aplicação cresça e se adapte às mudanças dinâmicas do mercado.

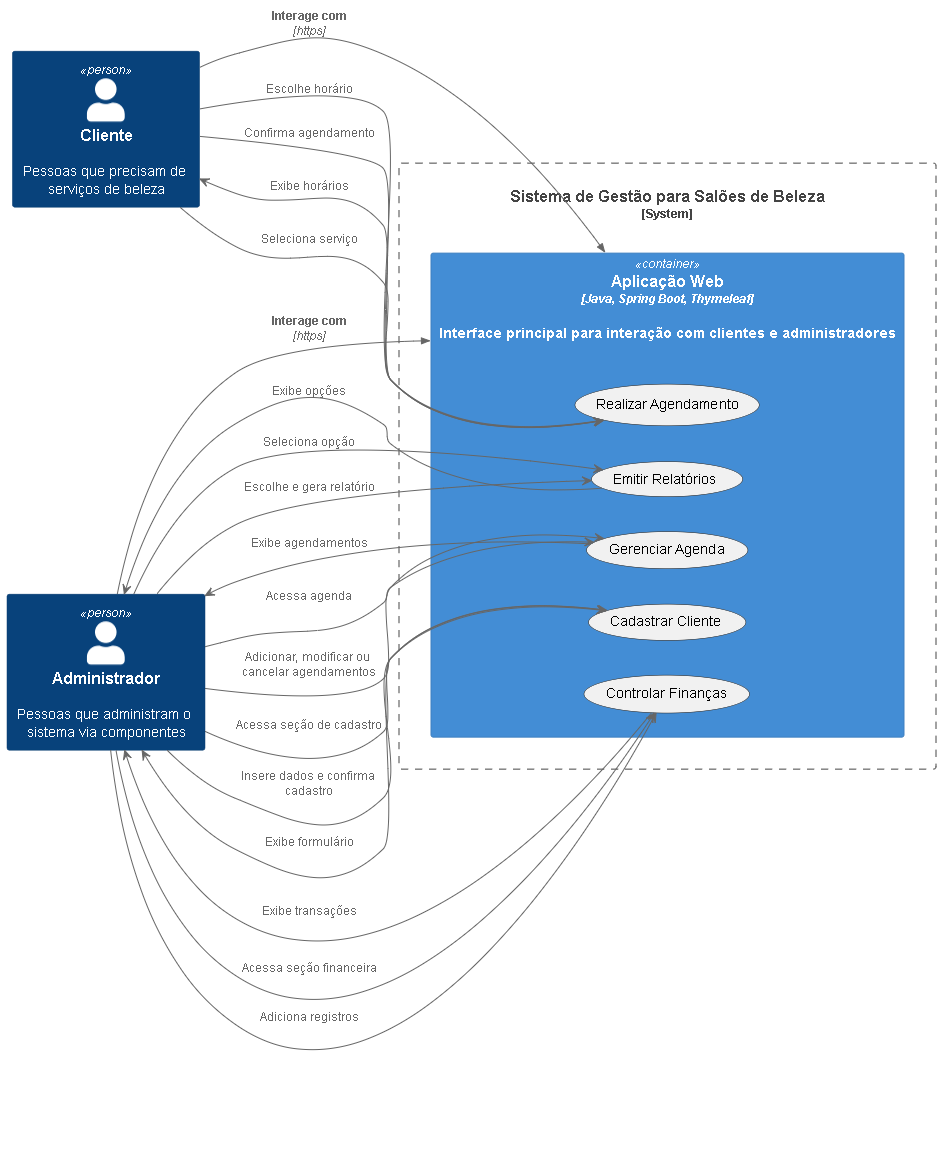
A elaboração cuidadosa de requisitos funcionais e não-funcionais reflete o compromisso em entregar um sistema de alta qualidade e desempenho. O diagrama de casos de uso fornecerá uma visão clara das interações entre os usuários e o sistema, garantindo uma compreensão abrangente dos fluxos de trabalho essenciais. Ao definir requisitos funcionais específicos, o sistema será capaz de lidar de forma eficaz com funções essenciais, como agendamento de serviços, cadastro de clientes, controle financeiro e geração de relatórios.

Os requisitos não-funcionais, por sua vez, assegurarão que a aplicação seja executada de maneira otimizada, mantendo altos padrões de desempenho, segurança e escalabilidade. Toda essa estrutura conceitual serve como alicerce para o desenvolvimento do sistema de gestão para salões de beleza, que tem como objetivo primordial elevar a eficiência, competitividade e satisfação geral do cliente nesse mercado dinâmico e em constante evolução.

## Diagrama de Casos de Uso

Diagrama do caso de uso realizado em UML disponibilizado no github:   
[ProjetoIntegrado-WagnerMartins-DigramaCasoUso](https://github.com/wagnermartinssantana/projetoIntegrado/blob/main/docs/C4model/diagrama-caso-uso.puml).

**Figura 1** - Diagrama casos de uso.



Fonte: Elaboração Própria.

## Requisitos Funcionais

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Descrição Resumida** | **Dificuldade (B/M/A)\*** | **Prioridade**  **(B/M/A)\*** |
| RF01 | O sistema deve permitir o cadastro e autenticação de administradores e clientes. | M | A |
| RF02 | Os administradores devem poder cadastrar, visualizar, editar e excluir informações de clientes, serviços e funcionários. | B | B |
| RF03 | Os clientes devem poder visualizar a disponibilidade de horários e agendar serviços. | A | M |
| RF04 | O sistema deve permitir o controle financeiro, registrando receitas e despesas e gerando relatórios. | B | A |
| RF05 | Deve haver um calendário interativo que permita ao administrador gerenciar os agendamentos. | A | M |
| RF06 | O sistema deve permitir que os clientes visualizem a lista de serviços oferecidos, incluindo descrição e preço. | B | A |
| RF07 | O sistema deve permitir que os administradores visualizem um calendário mensal com a agenda de todos os funcionários. | M | M |
| RF08 | O sistema deve calcular automaticamente o valor total a ser pago com base nos serviços agendados. | B | M |  |  |  |
| RF09 | Os administradores devem poder cadastrar novos serviços, incluindo nome, descrição e preço. | B | A |  |  |  |
| RF10 | O sistema deve permitir que os funcionários visualizem os detalhes dos serviços a serem realizados em cada agendamento. | B | A |  |  |  |
| RF11 | Os administradores devem poder gerar relatórios de desempenho de funcionários, incluindo número de atendimentos e serviços realizados. | M | M |  |  |  |
| RF12 | O sistema deve permitir que os clientes avaliem os serviços após o atendimento. | M | M |  |  |  |
| RF13 | Os administradores devem poder bloquear horários específicos no calendário, por exemplo, para feriados ou manutenções. | B | B |  |  |  |
| RF14 | Os clientes devem poder reagendar ou cancelar agendamentos com antecedência. | B | M |  |  |  |
| RF15 | Os administradores devem poder cadastrar informações detalhadas sobre cada funcionário como experiência e especializações. | B | B |  |  |  |
| RF16 | Os administradores devem poder cadastrar promoções especiais e descontos nos serviços. | B | M |  |  |  |
| RF17 | Os clientes devem poder visualizar as promoções ativas e aplicá-las ao agendar um serviço. | A | M |  |  |  |
| RF18 | O sistema deve permitir que os administradores visualizem um histórico completo de transações financeiras. | A | A |  |  |  |
| RF19 | O sistema deve gerar relatórios financeiros mensais com resumo de receitas e despesas. | B | M |  |  |  |
| RF20 | O sistema deve permitir que os administradores exportem relatórios financeiros em formato PDF ou planilha Excel. | B | M |  |  |  |
| RF21 | Os clientes devem poder visualizar os serviços oferecidos, incluindo descrição e preço. | M | A |  |  |  |
| RF22 | O sistema deve fornecer uma interface de busca para os clientes encontrarem facilmente os serviços desejados. | M | M |  |  |  |
| RF23 | O sistema deve permitir que os clientes visualizem as avaliações e comentários de outros clientes sobre os serviços e funcionários do salão. | A | M |  |  |  |
| RF24 | O sistema deve disponibilizar um histórico completo de transações financeiras, incluindo detalhes como data, valor, tipo de transação e agendamento associado. | A | M |  |  |  |
| RF25 | Os administradores devem ter a capacidade de atualizar as informações de contato do salão, como telefone e endereço. | B | A |  |  |  |

\* B = Baixa, M = Média, A = Alta.

## Requisitos Não-funcionais

|  |  |  |
| --- | --- | --- |
| **ID** | **Descrição** | **Prioridade**  **B/M/A** |
| RNF01 | O sistema deve ser seguro, protegendo os dados dos clientes e transações financeiras. | A |
| RNF02 | Deve ser de fácil utilização e possuir uma interface intuitiva. | M |
| RNF03 | Deve ser responsivo, adaptando-se a diferentes dispositivos (computador, tablet, smartphone). | M |
| RNF04 | O sistema deve ter alta disponibilidade, minimizando o tempo de inatividade. | A |
| RNF05 | Deve ser escalável, permitindo o aumento de usuários e funcionalidades sem perda de desempenho. | M |
| RNF06 | O sistema deve ter tempos de resposta rápidos, garantindo uma experiência ágil para os usuários durante a interação com a aplicação. | A |
| RNF07 | A aplicação deve ser desenvolvida seguindo boas práticas de programação e padronização de código. | M |
| RNF08 | A aplicação deve ser compatível com os principais navegadores web, como Chrome, Firefox, Safari e Edge. | M |

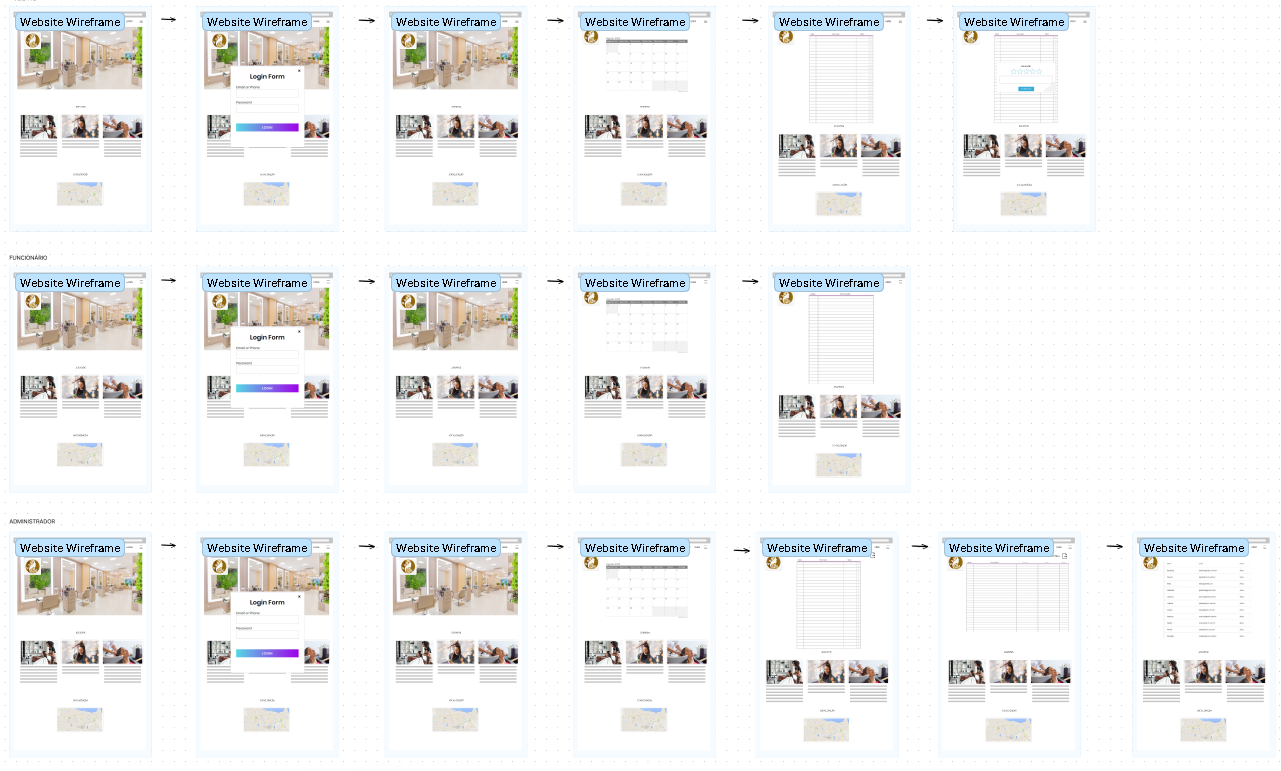
\* B = Baixa, M = Média, A = Alta.

## Protótipo Navegável do Sistema

Utilizado a ferramenta Figma, para a realização do protótipo que permitiu a criação de uma experiência interativa e a elaboração de wireframes precisos. O protótipo abrange a tela inicial da aplicação e demonstra três usuários distintos como cliente, funcionário e administrador e as suas principais funcionalidades.

Imagem utilizando o Figma:

**Figura 2** - Utilização figma.



Fonte: Elaboração Própria.

O documento em PDF contendo o protótipo navegável e os wireframes está disponível no seguinte link: [ProjetoIntegrado-WagnerMartins.pdf](https://github.com/wagnermartinssantana/projetoIntegrado/blob/main/docs/prototipo/ProjetoIntegrado-WagnerMartins.pdf)

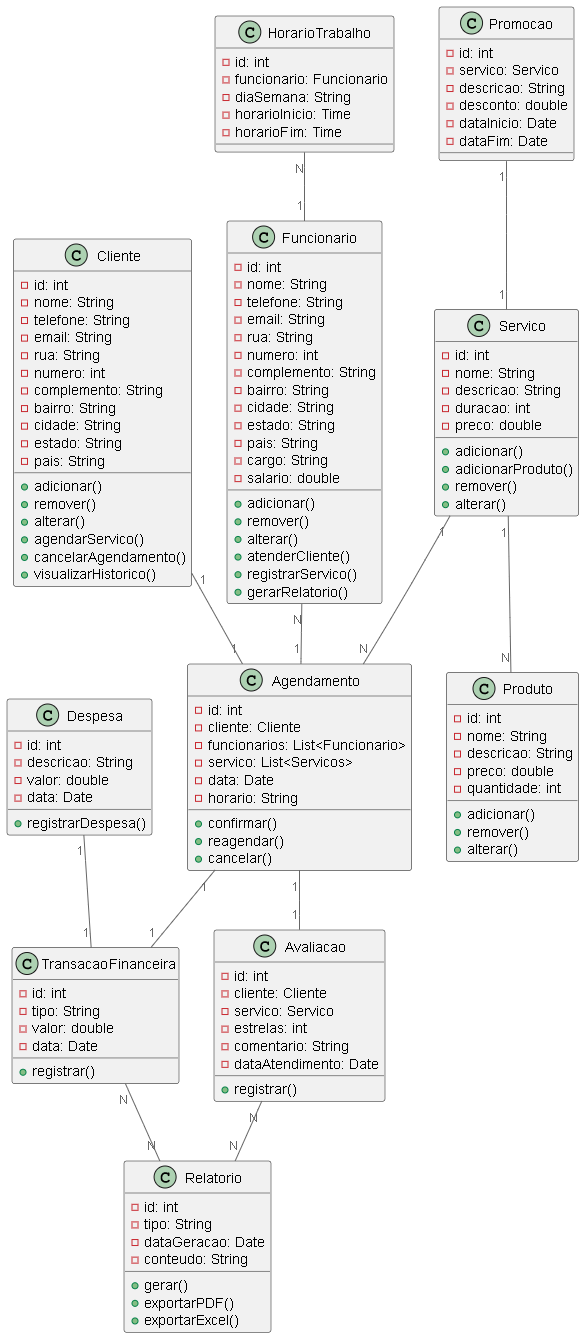
Foi realizado um vídeo de apresentação do protótipo navegável desenvolvido. O vídeo destaca os aspectos-chave do protótipo, demonstrando a experiência de navegação entre as telas e ressaltando os elementos interativos. O vídeo está disponível no formato MP4 e pode ser visto através do seguinte link no Google Drive: [ProjetoIntegrado-WagnerMartins.mp4](https://drive.google.com/file/d/1aePfFP--kkptOlBDxjcz_xpslPRryJO9/view?usp=sharing) ou no GitHub : [ProjetoIntegrado-WagnerMartins.mp4](https://github.com/wagnermartinssantana/projetoIntegrado/blob/main/docs/prototipo/ProjetoIntegrado-WagnerMartins.mp4) .

O meu repositório contendo o protótipo navegável na extensão do Figma para ser visualizado está disponível no GitHub: [ProjetoIntegrado-WagnerMartins.fig.](https://github.com/wagnermartinssantana/projetoIntegrado/blob/main/docs/prototipo/ProjetoIntegrado-WagnerMartins.fig)

## Diagrama de Classes de Domínio

Diagrama de classes de domínio em UML registrado no github: [ProjetoIntegrado-WagnerMartins-ClassesDominio](https://github.com/wagnermartinssantana/projetoIntegrado/blob/main/docs/C4model/diagrama-classe-dominio.puml) .

**Figura 3** - Diagrama de classes de domínio.



Fonte: Elaboração Própria.

## Arquitetura da Solução

## Padrão Arquitetural

A aplicação adota o padrão arquitetural MVC (Model-View-Controller). Ele ajuda a organizar e manter a aplicação de forma eficaz, promovendo a reutilização de código e a clareza na estrutura do projeto.

**Model (Modelo)**: A camada Model é uma parte fundamental no projeto e desempenha um papel central. Ela é responsável por representar os dados da aplicação e conter a lógica de negócios subjacente. No Gestão Glamour, a camada Model é composta pelos seguintes elementos:

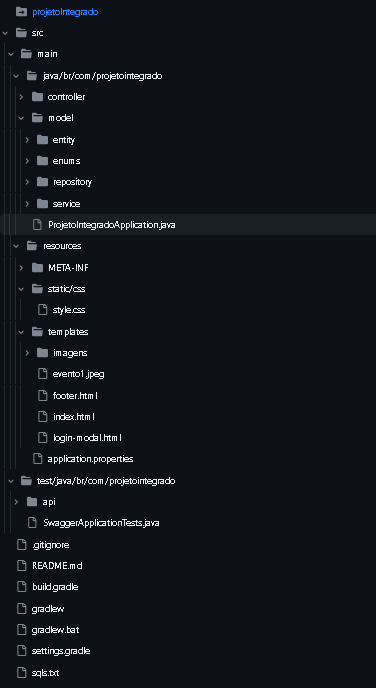
* Entidades (Entity): As entidades representam os principais objetos de negócio. No projeto, essas entidades incluem Cliente, Funcionario, Produto e outras. Cada entidade é mapeada para uma tabela no banco de dados e reflete a estrutura dos dados que você precisa armazenar e gerenciar. As entidades foram anotadas com o JPA (Java Persistence API) para definir como são mapeadas no banco de dados.
* Enums (Enumerations): Enums foram utilizados para representar conjuntos de valores fixos, como status, tipos, ou outras categorias que não mudam com frequência.
* Repositórios (Repository): Os repositórios são interfaces ou classes que fornecem métodos para realizar operações de leitura e escrita no banco de dados relacionado às entidades. Esses repositórios implementam interfaces fornecidas pelo Spring Data JPA.
* Serviços (Services): Os serviços são responsáveis por implementar a lógica de negócios da aplicação. Eles atuam como intermediários entre os controladores (parte da camada Controller) e os repositórios.

**View:** A camada View lida com a apresentação dos dados ao usuário final. Ela é responsável por exibir as informações aos usuários por meio de interfaces gráficas como Index, Login .... Toda parte visual.

**Controller:** Os controllers atuam como intermediários entre as camadas Model e View. Eles recebem solicitações do usuário, processam-nas usando os modelos apropriados e atualizam a visualização correspondente. Na aplicação, temos controllers para diferentes recursos, como ClienteController, FuncionarioController, AgendamentoController, etc.

A adoção do padrão MVC em nossa aplicação permite uma organização clara e modularização das funcionalidades, facilitando o desenvolvimento, a manutenção e a escalabilidade do sistema. A figura a seguir ilustra a estrutura do projeto com base nesse padrão arquitetural:

**Figura 4** - Estrutura do projeto



Fonte: Elaboração Própria.

A estrutura completa se encontra no [github](https://github.com/wagnermartinssantana/projetoIntegrado).

## C4 model - Diagrama de Contexto

**Atuadores (Atores):**

Cliente (User): Representa as pessoas que precisam de serviços de beleza, ou seja, os clientes do salão.

Administrador (Admin): São as pessoas responsáveis por administrar o sistema por meio de componentes, como funcionários do salão.

**Sistema (System):**

Sistema de Gestão para Salões de Beleza: É o sistema principal que engloba todas as funcionalidades e interações. Ele é o núcleo do software.

**Container (Retângulo):**

Gestão Glammour: É a aplicação web desenvolvida para fornecer a interface principal de interação com os clientes e administradores. Ela é construída usando tecnologias como Java, Spring Boot e Thymeleaf.

**Banco de Dados (Database):**

Banco de Dados: Armazena os dados essenciais do sistema, como informações dos clientes, agendamentos e transações financeiras. O banco de dados é composto por tabelas representadas no diagrama como "tb\_agendamento, “tb\_avaliacao”, “tb\_cliente”, “tb\_funcionario”, “tb\_horarioTrabalho”, “tb\_promocao".

**Conexão com Banco de Dados (Database Connection):**

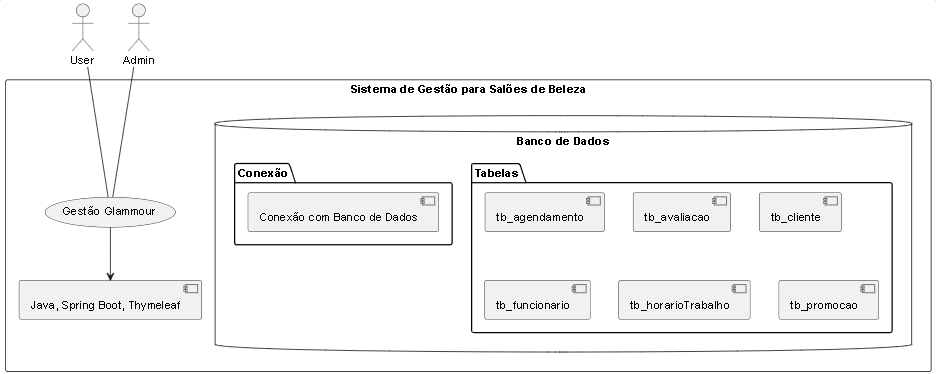
Representa a conexão que a aplicação web do Gestão Glammour estabelece com o banco de dados para recuperar e armazenar informações.

Em resumo, este Diagrama de Contexto mostra que os atores Cliente e Administrador interagem com o sistema por meio da aplicação web Gestão Glammour. Esta aplicação utiliza um banco de dados para armazenar e recuperar informações importantes. A conexão com o banco de dados é essencial para o funcionamento do sistema.

Esse diagrama fornece uma visão geral da arquitetura do sistema, destacando as principais partes e como elas se relacionam. Ele é uma representação valiosa para entender as interações de alto nível entre os atores e os componentes do sistema.

Diagrama em UML se encontra no git: [diagrama-contexto.puml](https://github.com/wagnermartinssantana/projetoIntegrado/blob/feature/docs/C4model/diagrama-contexto.puml) .

**Figura 5** - Diagrama de contexto.



Fonte: Elaboração Própria.

## Frameworks de Trabalho

Frameworks utilizados para o Gestão Glamour.

**Backend**

Spring Boot: É o principal framework de backend utilizado. Ele simplifica o desenvolvimento de aplicativos Java e fornece um ambiente de configuração, agilizando o desenvolvimento.

Spring Data JPA: Este módulo do Spring Framework facilita o acesso a bancos de dados relacionais, utilizando a API Java Persistence (JPA).

**Frontend**

Thymeleaf: É um mecanismo de modelo que integra bem com o Spring Boot. Ele é usado para criar páginas HTML dinâmicas.

Bootstrap 4: Um framework de front-end popular para criar interfaces de usuário responsivas e atraentes. Fornece componentes e estilos CSS pré-construídos.

Font Awesome: Uma biblioteca de ícones vetoriais escaláveis que melhora a experiência do usuário, fornecendo ícones visualmente atraentes.

**Documentação da API**

Swagger: É uma ferramenta que gera documentação interativa para APIs RESTful. Facilita a documentação e teste da API ProjetoIntegrado.

**Versionamento**

Git: O projeto Gestao Glamour utiliza um robusto sistema de controle de versão Git, permitindo um gerenciamento preciso do código-fonte e garantindo a integridade e colaboração contínua em todas as etapas do desenvolvimento.

Esses frameworks e tecnologias foram escolhidos para simplificar o desenvolvimento, criar uma interface de usuário amigável e documentar eficientemente a API. Eles desempenham papéis cruciais na construção do projeto do Gestão Glamour.

## Estrutura Base do Front End

< Nesta seção, você deve apresentar imagens legíveis do ***layout* mestre** e do ***menu* de opções do sistema**.>

## Modelo Relacional ou Projeto de Banco de Dados NoSQL

< Cole aqui uma imagem legível do modelo relacional ou do projeto de banco de dados NoSQL de todo o sistema.>

## Plano de Testes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Número** | **Caso de uso** | **Objetivo do caso de teste** | **Entradas** | **Resultados esperados** |
|  | Registro de Cliente | Verificar se um novo cliente pode ser registrado no sistema. | Dados do cliente, incluindo nome, telefone e e-mail. | O funcionário é cadastrado com sucesso no sistema e um ID é gerado. |
|  | Cadastro de Produto | Verificar se um novo produto pode ser cadastrado no sistema. | Dados do produto, incluindo nome, descrição, preço e quantidade em estoque. | O produto é cadastrado com sucesso no sistema e um ID é gerado. |
|  | Criação de Promoção | Verificar se uma nova promoção pode ser criada no sistema. | Serviço ao qual a promoção se aplica, datas de início e término da promoção, descrição e valor do desconto. | A promoção é criada com sucesso e associada ao serviço escolhido. |
|  | Registro de Transação Financeira | Verificar se uma transação financeira pode ser registrada no sistema. | Tipo de transação (receita ou despesa), valor e data da transação. | A transação é registrada com sucesso no sistema e um ID é gerado. |
|  | Atualização de Dados do Cliente | Verificar se os dados de um cliente existente podem ser atualizados corretamente. | Dados atualizados do cliente, incluindo nome, telefone e e-mail. | Os dados do cliente são atualizados com sucesso no sistema. |
|  | Agendamento de Serviço com Conflito de Horário | Verificar se o sistema impede o agendamento de serviços em horários conflitantes. | Cliente selecionado, serviço escolhido, data e horário que já tenha um agendamento. | O sistema deve impedir o agendamento e exibir uma mensagem de erro. |
|  | Consulta de Disponibilidade de Produto | Verificar se o sistema mostra a disponibilidade de um produto corretamente. | Produto selecionado e quantidade desejada. | O sistema deve mostrar se o produto está disponível em quantidade suficiente ou não. |
|  | Geração de Relatório Financeiro | Verificar se o sistema gera relatórios financeiros. | Período para o relatório (data inicial e data final). | O sistema deve gerar um relatório dentro do período especificado. |
|  | Consulta de Disponibilidade de Produto | Verificar a disponibilidade de um produto corretamente. | Produto selecionado e quantidade desejada. | O sistema deve mostrar se o produto está disponível em quantidade suficiente ou não. |
|  | Geração de Relatório Financeiro | Verificar se o sistema gera relatórios financeiros corretamente. | Período para o relatório (data inicial e data final). | O sistema deve gerar um relatório com as transações financeiras que ocorreram dentro do período especificado. |
|  | Aplicação de Promoção a um Serviço | Verificar se aplica o desconto de uma promoção a um serviço corretamente. | Serviço ao qual a promoção se aplica e promoção ativa. | O sistema deve aplicar o desconto ao serviço selecionado. |
|  | Exclusão de Produto | Verificar se o sistema permite a exclusão de produtos corretamente. | Produto a ser excluído. | O produto é removido com sucesso do sistema. |
|  | Avaliação de Serviço por um Cliente | Verificar se um cliente pode avaliar um serviço prestado. | Cliente, serviço e avaliação (por exemplo, nota e comentário). | A avaliação é registrada no sistema e associada ao cliente e serviço correspondentes. |
|  | Cadastro de Funcionário | Verificar se um novo funcionário pode ser cadastrado no sistema. | Dados do funcionário, incluindo nome, telefone, e-mail, cargo e salário. | O funcionário é cadastrado com sucesso no sistema e um ID é gerado. |
|  | Cancelamento de Agendamento | Verificar se o sistema permite que um cliente cancele um agendamento existente. | Agendamento a ser cancelado. | O agendamento é cancelado e a disponibilidade é atualizada. |
|  | Registro de Receita | Verificar se o sistema permite o registro de uma nova receita. | Tipo de receita (por exemplo, venda de serviço, venda de produto), valor e data. | A receita é registrada corretamente no sistema. |
|  | Atualização de Dados do Cliente | Verificar se o sistema atualiza dados do cliente. | nome, telefone e e-mail. | Os dados do cliente são atualizados com sucesso no sistema |
|  | Adição de Promoção | Verificar se o sistema permite a criação de uma nova promoção. | Dados da promoção, incluindo data de início, data de término, descrição e desconto. | A promoção é criada com sucesso e aplicada aos serviços relevantes. |
|  | Consulta de Avaliações de um Serviço | Verificar se o sistema mostra as avaliações de um serviço específico corretamente. | Serviço selecionado. | Uma lista de avaliações para o serviço escolhido é exibida. |
|  | Relatório de Despesas | Verificar se gera relatórios de despesas corretamente. | Período para o relatório (data inicial e data final). | O sistema deve gerar um relatório com as despesas dentro do período especificado. |
|  | Registro de Despesa | Verificar se o sistema permite o registro de uma nova despesa. | Tipo de despesa (por exemplo, compra de produtos, pagamento de funcionários), valor e data. | A despesa é registrada corretamente no sistema. |
|  | Pesquisa de Funcionário | Verificar se permite a pesquisa de funcionários. | Critérios de pesquisa, como nome, cargo ou cidade. | Uma lista de funcionários que correspondem aos critérios de pesquisa é exibida. |
|  | Remoção de Promoção Expirada | Verificar se o sistema remove automaticamente as promoções expiradas. | Data atual. | Promoções que atingiram a data de término são removidas da lista de promoções. |
|  | Exclusão de Serviço | Verificar se permite a exclusão de serviços existentes. | Serviço a ser excluído. | O serviço é removido do sistema. |
|  | Registro de Compra de Produtos | Verificar se registro de uma compra de produtos. | Produtos comprados, quantidade, valor total e data. | O registro da compra é adicionado ao histórico de compras da empresa. |
|  | Promoção Expirada | Verificar se o sistema remove automaticamente | Data atual | Promoções que atingiram a data de término são removidas. |

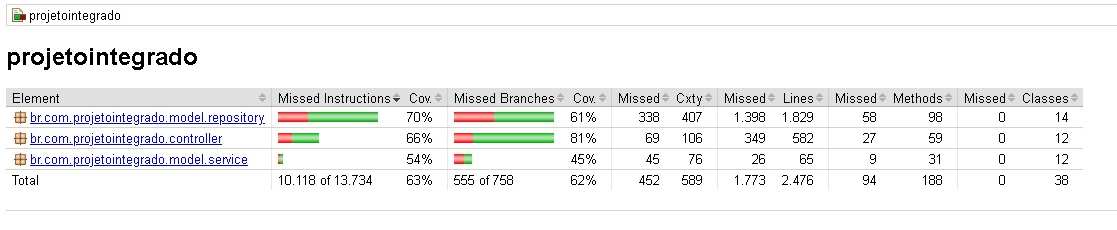
Além dos planos de testes o projeto Gestão Glamour inclui um conjunto abrangente de testes unitários para cada controller e suas funcionalidades. Esses testes são projetados para verificar a funcionalidade adequada das operações CRUD (Create, Read, Update, Delete) e outras funcionalidades essenciais.

O projeto Gestão Glamour também faz uso do relatório Jacoco para avaliar a abrangência dos testes.

Em resumo, os testes unitários e o relatório Jacoco são parte integrante do processo de desenvolvimento do projeto Gestão Glamour, garantindo a qualidade do código e a funcionalidade confiável do sistema.

Veja uma imagem do relatório jacoco gerado no projeto.

**Figura 6** - Relatório jacoco.



Fonte: Elaboração Própria.

## Apropriação de Horas no Projeto

|  |  |  |
| --- | --- | --- |
| **Histórico de apropriação de horas** | | |
| **Data do registro** | **Atividade** | **Quantidade de horas** |
| 01/07/2023 | Iniciado leitura do manual. | 3 horas |
| 02/07/2023 | Formulando o projeto. | 4horas |
| 08/07/2023 | Pesquisa e Coleta de Informações | 7 horas |
| 09/07/2023 | Finalização da Pesquisa | 6 horas |
| 15/07/2023 | Início da Introdução | 3 horas |
| 16/07/2023 | Continuação da Introdução | 6 horas |
| 17/07/2023 | Finalização Introdução | 2 horas |
| 22/07/2023 | Definição Conceitual da Solução | 5 horas |
| 23/07/2023 | Continuação da Definição Conceitual da Solução | 6 horas |
| 24/07/2023 | Continuação da Definição Conceitual da Solução | 3 horas |
| 25/07/2023 | Finalização da Definição Conceitual da Solução | 2 horas |
| 29/07/2023 | Diagrama de Classes de Domínio | 5 horas |
| 30/07/2023 | Continuação do Diagrama de Classes de Domínio | 4 horas |
| 05/08/2023 | Protótipo Navegável do Sistema | 6 horas |
| 06/08/2023 | Continuação do Protótipo Navegável do Sistema | 3 horas |
| 07/08/2023 | Continuação do Protótipo Navegável do Sistema | 5 horas |
| 11/08/2023 | Realizado o vídeo de apresentação e atualizado o projeto no GitHub | 5 horas |
| 19/08/2023 | Iniciado o parte 2 do projeto leitura e entendimento. | 3 horas |
| 26/08/2023 | Realizado a integração com o SWAGGER. | 5 horas |
| 27/08/2023 | Iniciado as entidades JAVA | 5 horas |
| 28/08/2023 | Continuado as entidades JAVA | 3 horas |
| 02/09/2023 | Realização das classes controllers, services e repositors. | 3 horas |
| 03/08/2023 | Continuação das classes controllers, services e repositors. | 2 horas |
| 24/08/2023 | Continuação das classes controllers, services e repositors. | 2 horas |
| 28/08/2023 | Realização testes Junit. | 2 horas |
| 29/08/2023 | Continuação testes Junit. | 3 horas |
| 30/08/2023 | Continuação testes Junit. | 2 horas |
| 02/09/2023 | Realização do front index. | 4 horas |
| 03/09/2023 | Continuação do front index. | 4 horas |
| 09/09/2023 | Realização do front cliente. | 3 horas |
| 10/09/2023 | Continuação do front cliente. | 4 horas |
| 16/09/2023 | Realização do front administrador. | 3 horas |
| 17/09/2023 | Continuação do front administrador. | 4 horas |
| 23/09/2023 | Realização Documentação Padrão Arquitetural. | 2 horas |
| 24/09/2023 | Realização Documentação Diagrama de contexto. | 2 horas |
| 30/09/2023 | Realização Documentação Frameworks de Trabalho. | 1 hora |
| 01/10/2023 | Realização Documentação Estrutura Base do FrontEnd. | 2 horas |
| 07/10/2023 | Realização Modelo Relacional. |  |
| 08/10/2023 | Realização do Plano de Testes. | 4 horas |
| 09/10/2023 | Realização Avaliação e Retrospectiva. | 3 horas |
| 10/10/2023 | Realização do Vídeo Final. | 3 horas |
| 11/10/2023 | Ajustes Finais no projeto. | 4 horas |
| 12/10/2023 | Disponibilização do Projeto e verificações | 2 horas |

## Código da Aplicação

< Informe aqui o ***link*** para seu **repositório público de código**.

Informe também o **endereço do *site* de sua aplicação**. É importante observar que, no ambiente fornecido para a avaliação, a base de dados deverá apresentar exemplos de teste previamente cadastrados que permitam visualizar o correto funcionamento do sistema. Indique também as **credenciais de acesso**, para **todos os perfis de usuários** da aplicação, que devem ser **empregadas pelos avaliadores**.

Indique o ***link*** para acesso ao **vídeo de apresentação de seu projeto**. Espera-se a produção de um vídeo sintético de, **no máximo**, **5 minutos**, no formato **MP4**, apresentando o projeto e a solução desenvolvida.>

## Avaliação Retrospectiva

Nesta seção, é apresentada uma avaliação do processo de desenvolvimento do projeto, incluindo os objetivos estimados e os objetivos alcançados.

## Objetivos Estimados

Os objetivos estimados para a execução deste projeto integrado abrangeram diversas áreas do conhecimento em Engenharia de Software, refletindo a natureza multidisciplinar da pós-graduação. Esses objetivos incluíram:

* Identificação de Requisitos Funcionais do Sistema: Esta fase envolveu uma análise profunda das necessidades dos salões de beleza. O objetivo era mapear e documentar os requisitos funcionais essenciais. Isso incluiu uma compreensão abrangente das operações diárias, interações com os clientes, o fluxo de agendamentos, a gestão financeira e outros aspectos críticos do negócio.
* Projeto da Arquitetura do Sistema: A arquitetura do sistema foi elaborada com atenção aos detalhes, definindo a estrutura geral, a disposição dos componentes e a interação entre eles. As interfaces com o usuário, mecanismos de comunicação e a lógica de negócios foram estabelecidos, proporcionando uma base sólida para o desenvolvimento futuro.
* Desenvolvimento do Protótipo Navegável: A criação de um protótipo navegável permitiu a visualização antecipada da aparência e usabilidade da aplicação. Isso facilitou a validação das ideias, a coleta de feedback dos usuários e a identificação de melhorias necessárias, garantindo que a solução estivesse alinhada com as expectativas.
* Implementação do Sistema de Gestão: Nesta fase, ocorreu a implementação prática das funcionalidades principais, como o agendamento de serviços, o controle financeiro e a geração de relatórios. Foram aplicadas as melhores práticas de desenvolvimento, focando na eficiência, desempenho e facilidade de manutenção do sistema.
* Testes e Validação do Sistema: Realizamos uma bateria completa de testes para assegurar a qualidade do sistema. Isso envolveu testes de funcionalidade, desempenho e segurança. Além disso, a validação foi conduzida com a participação de usuários reais, garantindo que o sistema atendesse às suas necessidades e expectativas de maneira eficaz.

## 13.2 Objetivos Alcançados

A maior parte dos objetivos do projeto foi alcançada com sucesso. O desenvolvimento abrangente do sistema de gestão para salões de beleza integrou os conhecimentos adquiridos nas disciplinas de Design, Requisitos, Arquitetura Back e Front, APIs e WebServices, e Testes. Essa integração eficaz permitiu a execução do projeto de acordo com os padrões estabelecidos.

O projeto demonstrou a aplicação prática de diversas disciplinas de Engenharia de Software, resultando em um sistema funcional e alinhado com os requisitos iniciais. A não conclusão de um dos objetivos enfatiza a importância de um gerenciamento eficaz do tempo e recursos em projetos complexos. Essa lição aprendida pode ser valiosa para projetos futuros.

Essa retrospectiva revela que, ao aplicar conhecimentos interdisciplinares e seguir uma metodologia eficiente, é possível atingir a maioria dos objetivos estabelecidos, mesmo em projetos desafiadores.

## Lições aprendidas

|  |  |  |
| --- | --- | --- |
|  | **Retrospectiva (Lições Aprendidas)** | |
|  | **Descrição da Lição** | **Classificação** |
| 1 | Experiência trabalho sobre pressão de tempo e avaliação. | Negativa |
| 2 | A necessidade de um gerenciamento rigoroso do tempo | Negativa |
| 3 | A relevância de definir claramente os requisitos | Positiva |
| 4 | A eficácia de realizar testes abrangentes | Positiva |
| 5 | A necessidade de documentação detalhada | Positiva |
| 6 | A necessidade de revisões regulares do progresso | Positiva |
| 7 | A relevância de definir prioridades claras | Positiva |
| 8 | A lição de que, em projetos complexos, desafios e erros surgirão | Negativa |
| 9 | Para a profissão de desenvolvedor é necessário estudar mais as tecnologias. | Positiva |
| 10 | Flexibilidade e adaptação as mudanças e imprevistos e muito importante. | Positiva |
| 11 | Reconhecer as realizações das etapas. | Positiva |
| 12 | Trabalho solo é uma jornada solitária com uma equipe deixa tudo mais fácil. | Negativa |

A maior lição é reconhecer que cada membro da equipe, seja um desenvolvedor, analista de qualidade, Scrum Master, analista de requisitos ou qualquer outro papel, desempenha um papel crucial e contribui para o sucesso do projeto. Todos colaboram de maneira única, e juntos, entregamos um trabalho completo e de qualidade. A possibilidade de enxergar e exercer papeis diferentes foi muito positivo para valorizar ainda mais todos os colegas de uma equipe de trabalho!

## Agradecimentos

Primeiramente, gostaria de expressar minha profunda gratidão a Deus por me proporcionar a oportunidade de realizar este projeto.

À minha amada esposa, Dayane, e à minha família, quero expressar meu mais sincero agradecimento. Sua compreensão, apoio e paciência foram inestimáveis enquanto eu me dedicava a este projeto.

Também desejo estender minha gratidão aos meus amigos, que proporcionaram momentos de distração e alegria ao longo desse desafio.

Aos professores da PUC Minas, quero expressar minha sincera apreciação. Seus ensinamentos e orientações desempenharam um papel significativo na minha formação e na realização deste projeto.

A todos, muito obrigado!

1 Samuel 7:12

... Até aqui nos ajudou o Senhor.