RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 14 Nro. 1 2024
Uni-FACEF Centro Universitário Municipal de Franca

ChromaCheck

Miguel Rodrigues Gomes
Graduando em Engenharia de Software – Uni-FACEF
miguelrgomes07@gmail.br

Orientador: Prof. Me Carlos Alberto Lucas
Mestre em Educação – Uni-FACEF
carloslucas@facef.br

Resumo
Após uma série de pesquisas com empresas e prestadores de serviços na área de pintura predial / residencial, tornou-se aparente uma dor comum enfrentada por muitos: a dificuldade e o tempo necessários para criar orçamentos manualmente. Os profissionais dizem que o processo atual é muitas vezes demorado e sujeito a erros, afetando a eficiência operacional e a precisão das recomendações apresentadas aos clientes. Este problema demonstra a necessidade urgente de uma solução que não só simplifique o orçamento, mas também melhore a gestão de recursos e o atendimento ao cliente. A solução proposta é um sistema digital que automatiza cada etapa do processo, desde a coleta de informações até a geração e envio de orçamentos aos clientes. A motivação para essa iniciativa reside na necessidade de aumentar a eficiência operacional da empresa, garantindo a precisão e consistência dos orçamentos, e oferecendo uma experiência mais satisfatória aos clientes. Para alcançar esses objetivos, foram utilizadas práticas de desenvolvimento de software e técnicas de UX design. O sistema aliviará a carga de trabalho, eliminando algumas necessidades e proporcionando uma experiência mais eficiente para a empresa e os clientes, bem como apresentando uma plataforma que faz a junção entre usuário/cliente e usuário/empresa, com a principal atividade que visa simplificar e agilizar o processo de elaboração de orçamentos para serviços de pintura residencial, atualmente com alta demanda de tempo e recursos.

Palavras-chave: Automatização. Orçamentos. Pintura residencial. Plataforma.

Abstract
After a series of informal surveys with companies and service providers in the area of building/residential painting, a common pain faced by many became apparent: the difficulty and time required to create quotes manually. Professionals say the current process is often time-consuming and error-prone, affecting operational efficiency and the accuracy of recommendations presented to clients. This problem demonstrates the urgent need for a solution that not only simplifies budgeting, but also improves resource management and customer service. The proposed solution is a digital system that automates each step of the process, from collecting information to generating and sending quotes to customers. The motivation for this initiative lies in the need to increase the company's operational efficiency, ensuring the accuracy and consistency of budgets, and offering a more satisfactory experience to customers. To achieve these objectives, software development practices and UX design techniques were used. The system will alleviate the workload, eliminating some needs and providing a more efficient experience for the company and customers, as well as presenting a platform that

brings together user/client and user/company, with the main activity which aims to simplify and speed up the process of preparing quotes for residential painting services, currently with a high demand for time and resources.

Keywords: Automation. Estimates. Platform. Residential painting.

1 Introdução

A elaboração de orçamentos desempenha um papel fundamental na prestação de serviços de pintura residencial, abrangendo uma ampla gama de projetos que vão desde casas individuais até grandes edifícios comerciais. Pois, cada projeto apresenta desafios únicos em termos de escopo, materiais necessários e exigências específicas do cliente.

Atualmente, o processo de criação de orçamentos para serviços de pintura residencial é amplamente baseado em métodos tradicionais e manuais. Isso geralmente envolve uma série de etapas, desde a visita ao local para avaliar o trabalho, passando pela estimativa dos materiais e mão de obra necessários, até a elaboração final do orçamento e sua apresentação ao cliente.

No entanto, esse processo manual é frequentemente demorado e é necessário muitos recursos. Os profissionais da pintura precisam se deslocar até o local do projeto, realizar medições detalhadas, identificar os materiais adequados e, em seguida, transferir todas essas informações para um documento formal de orçamento. Além disso, a precisão e consistência dos orçamentos podem ser comprometidas por erros humanos ou falta de padronização nos processos.

Diante deste cenário apresentado, alguns questionamentos são necessários: Como a implementação de tecnologias digitais pode transformar esses processos? De que forma um sistema automatizado pode reduzir o tempo dedicado com tarefas repetitivas e minimizar erros humanos? E como a padronização oferecida pela tecnologia pode garantir orçamentos mais precisos e consistentes?

Ao longo deste trabalho, será explorado em detalhes como um software dedicado pode simplificar e agilizar o processo de elaboração de orçamentos para uma variedade de projetos de pintura residencial, abrangendo desde casas individuais até grandes empreendimentos comerciais. Analisaremos os benefícios e as vantagens que um sistema digital pode oferecer, tanto para os profissionais da pintura quanto para os clientes, e discutiremos as melhores práticas e estratégias para sua implementação eficaz.

2 Referencial Teórico

A solução proposta para resolver o problema de pesquisa selecionado envolve o desenvolvimento de um módulo de backend que dará suporte à operação de um sistema de orçamento, além do desenvolvimento de alta resolução do módulo de frontend de um aplicativo web e mobile, destinado a facilitar a criação de orçamentos e disponibilizado para os clientes. Observando que os clientes cada vez mais preferem orçamentos digitais, seja por motivos de facilidade na cotação de materiais e outros, este sistema também contará com a funcionalidade de cotação de materiais através de parcerias com lojas de tintas.

Neste referencial teórico, além de explorarmos a teoria dos termos relacionados ao problema em questão, também serão apresentadas as ferramentas utilizadas no desenvolvimento deste sistema, proporcionando uma compreensão abrangente do contexto teórico e prático envolvido no projeto.

2.1 Orçamento

O orçamento é um processo fundamental no contexto financeiro e de gestão, envolvendo a estimativa detalhada dos custos e recursos necessários para a realização de um projeto, atividade ou plano dentro de um período específico. Ele é utilizado em diversos contextos, como em projetos de construção, reforma, desenvolvimento de software, planejamento empresarial, entre outros.

O orçamento consiste em prever e planejar os gastos e receitas esperados durante a execução de um projeto ou para um determinado período. Isso envolve identificar todos os custos envolvidos, desde materiais e mão de obra até despesas administrativas e operacionais, e prever as possíveis fontes de receita, como vendas, investimentos ou financiamentos.

Para Garrison et al. (2007, p. 314) “Um orçamento é um plano detalhado de aquisição e uso de recursos financeiros e de outros tipos durante outros tipos determinados. Representa um plano para o futuro, expresso em termos quantitativos normais”

2.1.1 Objetivos do Orçamento

Os objetivos de um orçamento incluem o planejamento financeiro, estabelecimento de metas, controle de gastos, tomada de decisão informada, avaliação de desempenho e comunicação transparente das prioridades financeiras da organização. Em resumo, o orçamento ajuda a controlar e direcionar os recursos financeiros de forma eficiente, alinhando-se aos objetivos estratégicos da entidade.

Para Oliveira, Perez Jr., e Silva (2002, p. 123) os objetivos principais do orçamento se resumem em projetar de forma integrada e estruturada o resultado econômico, financeiro de um processo de planejamento e controlar o desempenho em face dos objetivos e metas definidas (acompanhamento orçamentário).

2.1.2 Etapas da elaboração do orçamento

Antes de criar o orçamento, há uma fase chamada planejamento, na qual as metas e o direcionamento são estabelecidos para serem incorporados ao orçamento.

O objetivo do planejamento é desenvolver processos, mecanismos e atitudes para tornar possível a avaliação das consequências futuras das decisões presentes em face aos objetivos da empresa, possibilitando assim a tomada de decisões no futuro, com mais agilidade e eficiência (PELEIAS, 2002).

Em um contexto globalizado, o planejamento se torna indispensável para as empresas, pois visa organizar as atividades e gerenciar os recursos disponíveis de forma a alcançar os objetivos estipulados. Planejar, nesse sentido, é antecipar decisões estratégicas.

Após definir o panorama econômico e os princípios fundamentais, a empresa inicia a elaboração de seu orçamento operacional, geralmente elaborado anualmente. Dessa forma, o orçamento operacional é uma ferramenta de curto prazo integrada ao planejamento estratégico, que, por sua vez, é de longo prazo. Diversos especialistas em orçamento recomendam um ciclo anual para o planejamento financeiro empresarial.

2.1.3 Pinturas em Geral

A pintura de ambientes residenciais e comerciais é um processo que envolve diversas etapas cuidadosamente planejadas para assegurar qualidade e durabilidade. Primeiro, ocorre a preparação da superfície, que pode incluir a remoção de imperfeições e a realização de reparos para garantir uma base uniforme. Em seguida, aplica-se um primer, essencial para melhorar a aderência da tinta e prolongar a durabilidade do acabamento. A escolha da tinta adequada depende de fatores como o tipo de superfície e a exposição ao ambiente; após essa seleção, a tinta é aplicada em camadas finas e uniformes para obter uma cobertura homogênea. Dependendo do projeto, podem ser necessários acabamentos adicionais, como vernizes ou protetores.

Por fim, a secagem e a cura da tinta completam o processo, permitindo que a camada de recobrimento atinja sua resistência ideal. Esse procedimento, cumpre não apenas a função estética de tornar os espaços atraentes, mas também a função protetora, garantindo que as superfícies, tanto internas quanto externas, mantenham sua integridade por mais tempo.

2.2 Ferramentas tecnológicas utilizadas neste projeto

Para o desenvolvimento de uma solução sistêmica, é preciso utilizar algumas ferramentas conhecidas como linguagens de programação. As mesmas são essenciais e possibilitam o relacionamento entre as telas e o banco de dados.

2.2.1 Linguagem - Java

Java é uma linguagem de programação desenvolvida pela *Sun MicroSystems*. É uma linguagem com suporte a orientação a objetos e muitos outros paradigmas, com ênfase na arquitetura cliente-servidor (Eduardo, 2022).

É uma linguagem de programação que permite criar programas que funcionam em diferentes sistemas operacionais. Você escreve código Java, que é compilado em um formato intermediário chamado bytecode. Esse bytecode é interpretado pela Máquina Virtual Java (JVM), que o traduz para o sistema operacional em que está sendo executado. Java é orientado a objetos, o que significa que você cria programas usando classes e objetos. É uma escolha popular para desenvolvimento devido à sua versatilidade e ampla gama de ferramentas disponíveis.

2.2.1.1 Benefícios da Utilização do Java

A linguagem Java oferece uma série de benefícios para projetos de software. Sua portabilidade, amplo suporte e arquitetura de segurança robusta fazem dela uma escolha confiável para desenvolvimento em diferentes plataformas. Além disso, sua

orientação a objetos promove modularidade e reutilização de código, facilitando o desenvolvimento e manutenção de sistemas. Java também se destaca pela escalabilidade em sistemas de grande porte e pelo desenvolvimento rápido, com um ecossistema maduro de ferramentas e frameworks disponíveis para acelerar o processo de desenvolvimento.

Em resumo, Java proporciona tanto versatilidade quanto confiabilidade para projetos de software. Com sua portabilidade, segurança e orientação a objetos, facilita o desenvolvimento de sistemas modulares e de fácil manutenção. Além disso, sua escalabilidade e ampla adoção garantem que ele seja uma escolha sólida para uma variedade de aplicativos, desde sistemas corporativos até aplicações distribuídas em nuvem.

2.2.2 Docker

O Docker é uma coleção de produtos oferecidos como Plataforma de Serviço (PAAS do termo em inglês Platform As a Service) que usam recursos de nível de sistema para entregar software em pacotes chamados contêineres. Os contêineres são isolados uns dos outros e contêm seus próprios softwares, bibliotecas e arquivos de configuração. Todos os contêineres são executados por um único kernel do sistema operacional e, usam menos recursos do que as máquinas virtuais (DOCKER, 2022).

Uma característica crucial do Docker é sua capacidade de compartilhar o kernel do sistema operacional do host com o software em execução, permitindo que ele combine o aplicativo e suas dependências em um contêiner virtual. Esse contêiner pode ser executado em qualquer distribuição Linux, proporcionando flexibilidade e portabilidade para o aplicativo, independentemente de ser executado localmente, na nuvem ou em uma nuvem privada.

Com o objetivo de estabelecer um ambiente de desenvolvimento robusto, que facilite a manipulação e centralização de ferramentas e tecnologias, todos os serviços utilizados no desenvolvimento e operacionalização de todo projeto foram baseados no Docker.

2.2.3 PostgreSQL

O PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional (ORDBMS) baseado no POSTGRES, Versão 4.21, desenvolvido no Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley. O POSTGRES foi pioneiro em muitos conceitos que só foram disponibilizados em alguns sistemas de banco de dados comerciais muito depois.

Armazena dados de forma eficiente em disco e oferece recursos avançados, como transações ACID, replicação de dados e suporte a dados geoespaciais. Essa combinação de recursos faz do PostgreSQL uma escolha robusta e confiável para armazenar e manipular dados em uma variedade de aplicativos e ambientes.

2.2.4 ReactJS

O ReactJS foi criado pela equipe do Facebook em 2011 para otimizar a atualização e sincronização de atividades nos feeds de notícias da rede social, incluindo chats, status, listas de contatos e muito mais. ReactJS é uma biblioteca JavaScript para criar interface de usuário (REACT, 2022).

O ReactJS simplificou essas descrições, facilitando a conexão entre HTML, CSS, JavaScript e todos os elementos da página. Graças à sua eficiência, o ReactJS foi integrado às interfaces de outras redes sociais do mesmo grupo, como o Instagram, nos anos subsequentes. Em 2013, o ReactJS tornou seu código acessível à comunidade, o que impulsionou sua popularidade. Hoje em dia, o ReactJS é reconhecido como uma das principais bibliotecas JavaScript do mercado. Sua flexibilidade e capacidade de integração com outras bibliotecas e frameworks são incentivos para seu uso. O ReactJS é, acima de tudo, JavaScript, e é considerado uma das melhores ferramentas para desenvolvimento nesta linguagem. Sua adoção como framework frontend foi impulsionada pelo uso de rotas de API para comunicação.

2.2.5 GitHub

GitHub é um recurso de hospedagem de código para controle de versão e cooperação. É um software majoritariamente usado por desenvolvedores, pois permite um alojamento prático de código-fonte e arquivos em nuvem (GITHUB, 2022).

Essa abordagem permite que múltiplos membros da equipe trabalhem simultaneamente em um projeto, cada um contribuindo com sua própria versão. Todos têm a mesma oportunidade de apresentar seu trabalho, sujeito à posterior aprovação do líder ou até mesmo de forma independente. Os arquivos resultantes estão acessíveis para aqueles com permissão, em qualquer lugar com acesso à internet, o que é crucial agora que o trabalho remoto se tornou predominante.

2.2.6 Trello

O Trello é uma ferramenta de colaboração que organiza projetos em quadros e insere listas de tarefas a serem feitas individualmente ou em equipe. Cada lista recebe um cartão com uma descrição, uma data de vencimento e uma meta a ser concluída (TRELLO, 2022).

Os painéis podem ser compartilhados com qualquer pessoa que tenha uma conta no Trello, e é possível adicionar vários usuários aos cartões. Ao marcar alguém em um cartão, não apenas atribui-se a responsabilidade da tarefa a esse membro da equipe, mas também permite que todos os outros membros do painel saibam o que estão fazendo no momento.

O Trello também apresenta um sistema de marcação com cores. Além de auxiliar na categorização das atividades, essa função facilita a filtragem de cartões durante as pesquisas na plataforma. No total, há dez opções de cores, que podem ter significados diferentes dependendo do contexto do painel.

Todas essas características, juntamente com a facilidade de uso e o fato de ser gratuito, foram levadas em consideração ao optar por utilizar essa ferramenta para auxiliar no gerenciamento do projeto conduzido pela equipe.

2.2.7 Figma

O Figma é um editor de gráficos vetoriais online que se concentra na prototipagem de interfaces gráficas e estruturas de design de experiência do usuário (GARRETT, 2021).

O maior diferencial do Figma é a capacidade de criar protótipos interativos que permitem testar visualmente os fluxos do usuário por meio de animações e transições. Isso facilita a visualização de como funcionará o produto, proporcionando uma experiência mais realista e ajudando a refinar seu design antes da implementação.

3 Proposta de solução

Para desenvolver o software foram criados artefatos recomendados pelas boas práticas de engenharia de software e projetado para melhorar a experiência do desenvolvedor e melhor implementação e manutenção código. Esses projetos são usados antes do desenvolvimento do código, durante o desenvolvimento, também será usado para manutenção do sistema.

3.1 Matriz SWOT

A matriz SWOT é uma técnica de planejamento estratégico usada para avaliar os pontos fortes, fracos, oportunidades e ameaças de uma organização.

- **Forças:** Aspectos internos positivos, como recursos profundos e uma marca forte, podem ser aproveitados para obter uma vantagem competitiva.

- Fraquezas: Fatores negativos internos, como falta de recursos ou processos desatualizados, que devem ser melhorados para se tornarem mais eficientes e competitivos.

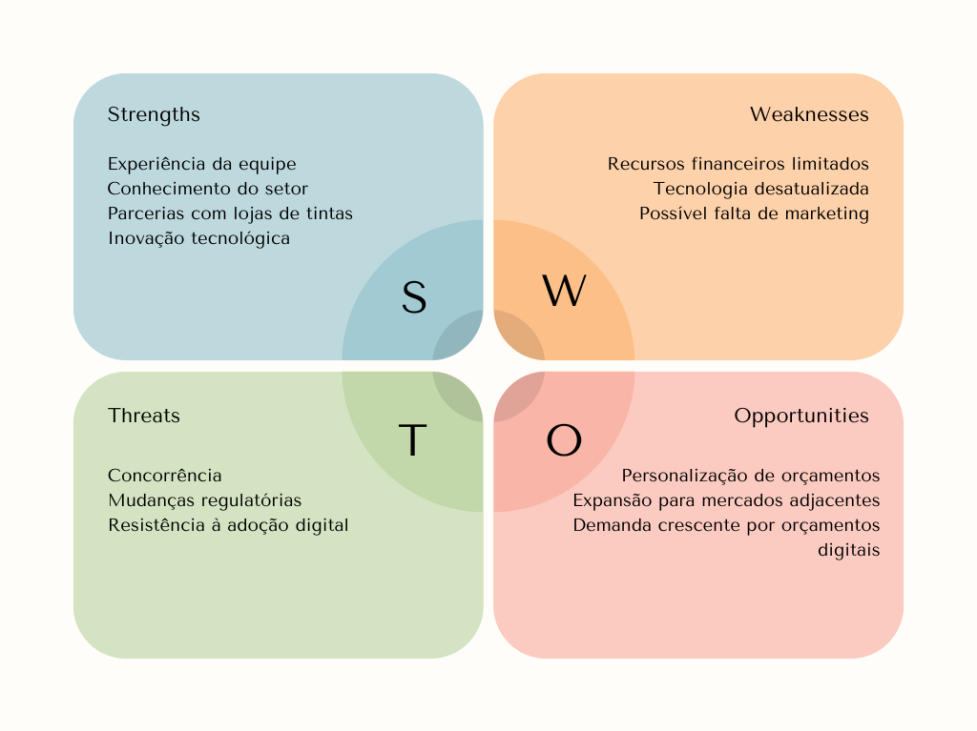
- Oportunidades: Fatores externos favoráveis que podem ser usados para alcançar crescimento e sucesso, como novas tendências ou avanços tecnológicos.

- Ameaças: Fatores externos negativos, como concorrência intensa ou mudanças regulatórias adversas, devem ser mitigados para proteger a organização.

3.2 Requisitos

A Elicitação de Requisitos, baseada em Pressman (2019) é a etapa de identificação do problema, coletando informações necessárias para propor uma solução (requisitos). Então, no projeto aqui apresentado, tudo que o sistema deveria ter foi mapeado conforme as regras, até os botões e instruções. Após a coleta, análise Requisitos, definindo a dificuldade e prioridade de cada item. A Tabela 1 contém requisitos específicos que foram feitos.

Figura 1: Matriz SWOT



Fonte: o autor

Tabela 1: Requisitos do sistema

Requisitos	
R0001F	Cadastrar Empresa
R0002F	Cadastrar Cliente
R0003F	Cadastrar Serviços
R0004F	Cadastrar Tabela de Preços
R0005F	Cadastrar Orçamento
R0006F	Cadastrar Materiais do Orçamento
R0007F	Cadastrar Serviços do Orçamento
R0008F	Enviar Orçamento para Cliente
R0009F	Aprovar Orçamento
R0010F	Enviar Orçamento para Lojas Parceiras

Fonte: o autor

3.3 Business Process Model and Notation (BPMN)

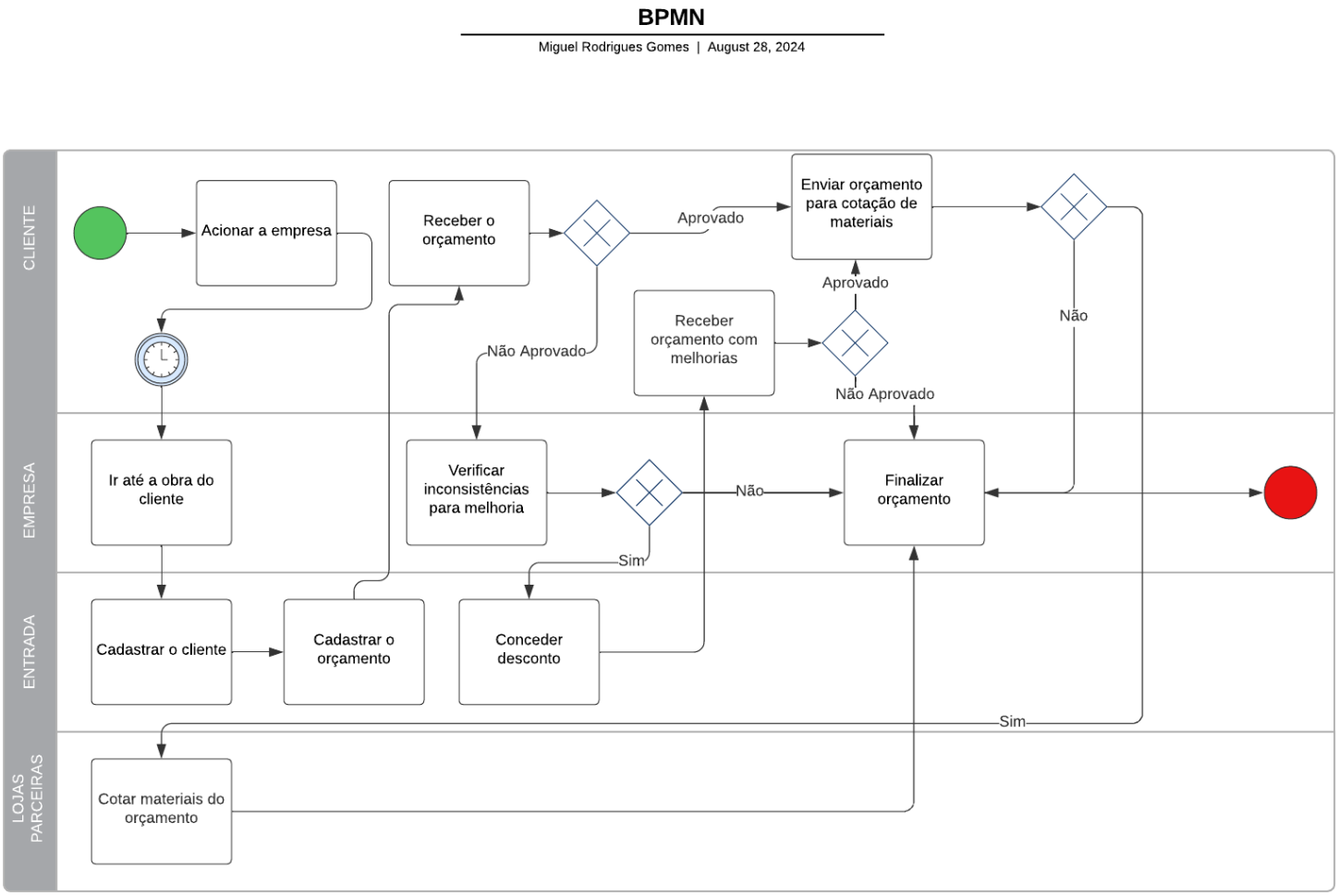
BPMN é usado para modelagem orientada a objetos, ajuda a organizar os processos de negócios para que nada se perca no caos diário e cada operação seja realizada da forma mais simples possível (TOTVS, 2022).

A modelagem de processos é uma solução de redução de custos para as empresas que podem operar de forma otimizada de acordo com seus objetivos e fornecer uma visão transparente de como o trabalho está progredindo.

É uma representação gráfica composta por ícones que simbolizam o fluxo do processo. Isto significa que os processos podem ser mapeados a partir deste símbolo. Cada ícone é uma etapa do processo de modelagem. Auxilia o processo de desenho usando símbolos padronizados em representações gráficas. Muito parecido com um fluxograma tradicional, mas com informações diferentes. Já o modo simbólico fornece informações mais completas.

BPMN é um método para tornar as operações empresariais mais eficientes, otimizando resultados e reduzindo assim o consumo de recursos. Aumento da transparência estrutural e estratégica, permitindo uma visão completa de como está o funcionamento da organização.

Figura 2: BPMN



Fonte: o autor

3.4 Plano 5W1H

O Plano 5W1H é uma ferramenta utilizada para levantamento e análise das características e do contexto do problema a ser resolvido. Por meio de tópicos que ajudam a esclarecer e aprofundar as questões envolvidas, ela oferece uma visão mais detalhada dos elementos presentes no problema. A seguir, são apresentadas as especificações de cada item para este projeto.

What: Desenvolver um software especializado na elaboração de orçamentos para pintura residencial. O sistema permitirá que profissionais da pintura criem orçamentos precisos e detalhados, gerenciando materiais, custos e prazos de forma eficiente.

Why: Para automatizar o processo de orçamentos, reduzindo o tempo necessário para a criação e aumentando a precisão das estimativas. Isso resulta em uma maior eficiência para os profissionais e uma melhor experiência para os clientes.

Where: O software será desenvolvido inicialmente para Web/Mobile e será necessário à internet.

Who: Desenvolvedores/estudantes responsáveis pelo projeto

When: O projeto começará em 2024, com previsão de conclusão e lançamento em 2025. Durante esse período, o software passará por fases de coleta de requisitos, desenvolvimento, testes e treinamento.

How:

- Coletando Requisitos: Entrevistas e sessões de feedback com profissionais da pintura e clientes para identificar necessidades e funcionalidades essenciais.

- Desenvolvendo: A equipe de programadores criará o software com base nos requisitos coletados, integrando funcionalidades como cálculo de custos, gerenciamento de materiais e geração de relatórios.

- Testando: Realização de testes rigorosos para garantir a funcionalidade do sistema e correção de possíveis bugs.

- Implantando: Disponibilização do software para os usuários finais, com suporte para instalação e configuração.

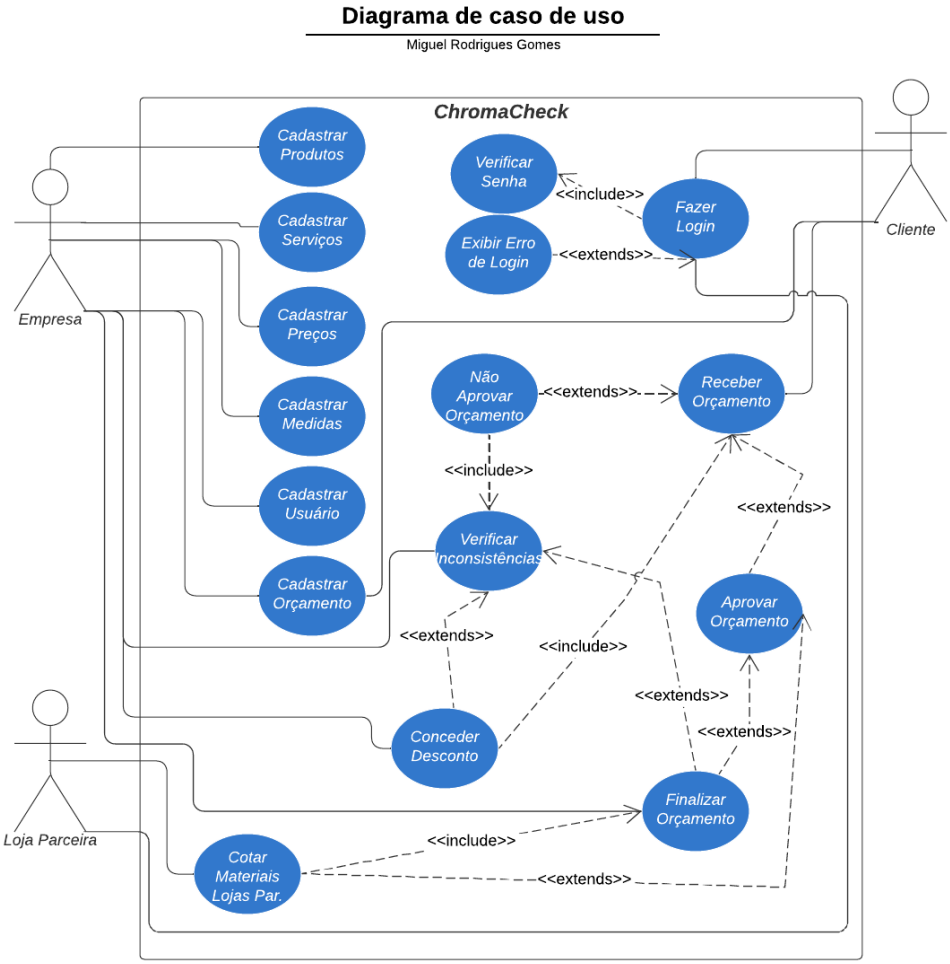
- Treinando: Capacitação dos usuários para garantir o uso eficaz do software e aproveitamento completo de suas funcionalidades.

- Suporte Contínuo: Oferecimento de manutenção e atualizações regulares para resolver problemas e melhorar o software conforme necessário.

3.5 Diagrama de Casos de Uso

De acordo com Booch, Rumbaugh e Jacobson (2022), um diagrama de casos de uso modela o comportamento de um sistema, subsistema ou classe, representando um conjunto de casos de uso, atores e seus relacionamentos. A Figura 3 mostra o diagrama de casos de uso do sistema.

Figura 3: Diagrama de Casos de Uso

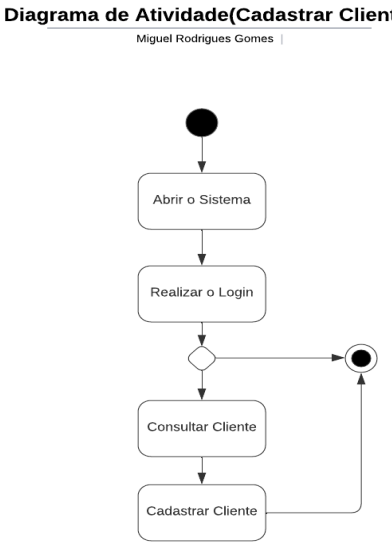


Fonte: o autor

3.6 Diagrama de Atividades do Sistema

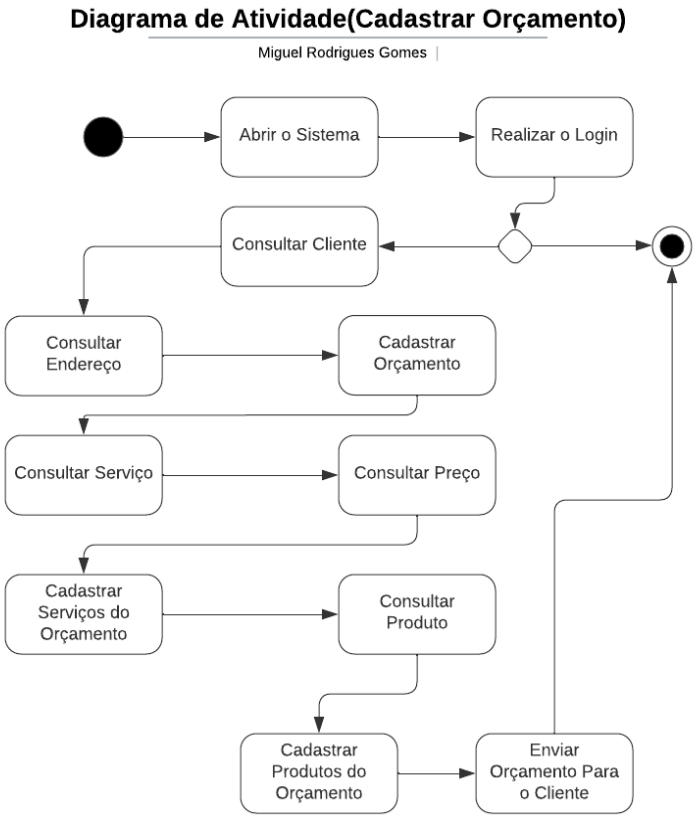
De acordo com Booch, Rumbaugh e Jacobson (2012, p. 401), um diagrama de atividades é “essencialmente um fluxograma que mostra o fluxo de controle de uma atividade para outra”. A Figura 4 apresenta o diagrama de atividades do sistema para o cadastro de cliente e a Figura 5 apresenta o diagrama de atividades do sistema para o cadastro de orçamento.

Figura 4: Diagrama de Atividades do Sistema(Cadastro de Cliente)



Fonte: o autor

Figura 5: Diagrama de Atividades do Sistema (Cadastro de Orçamento)

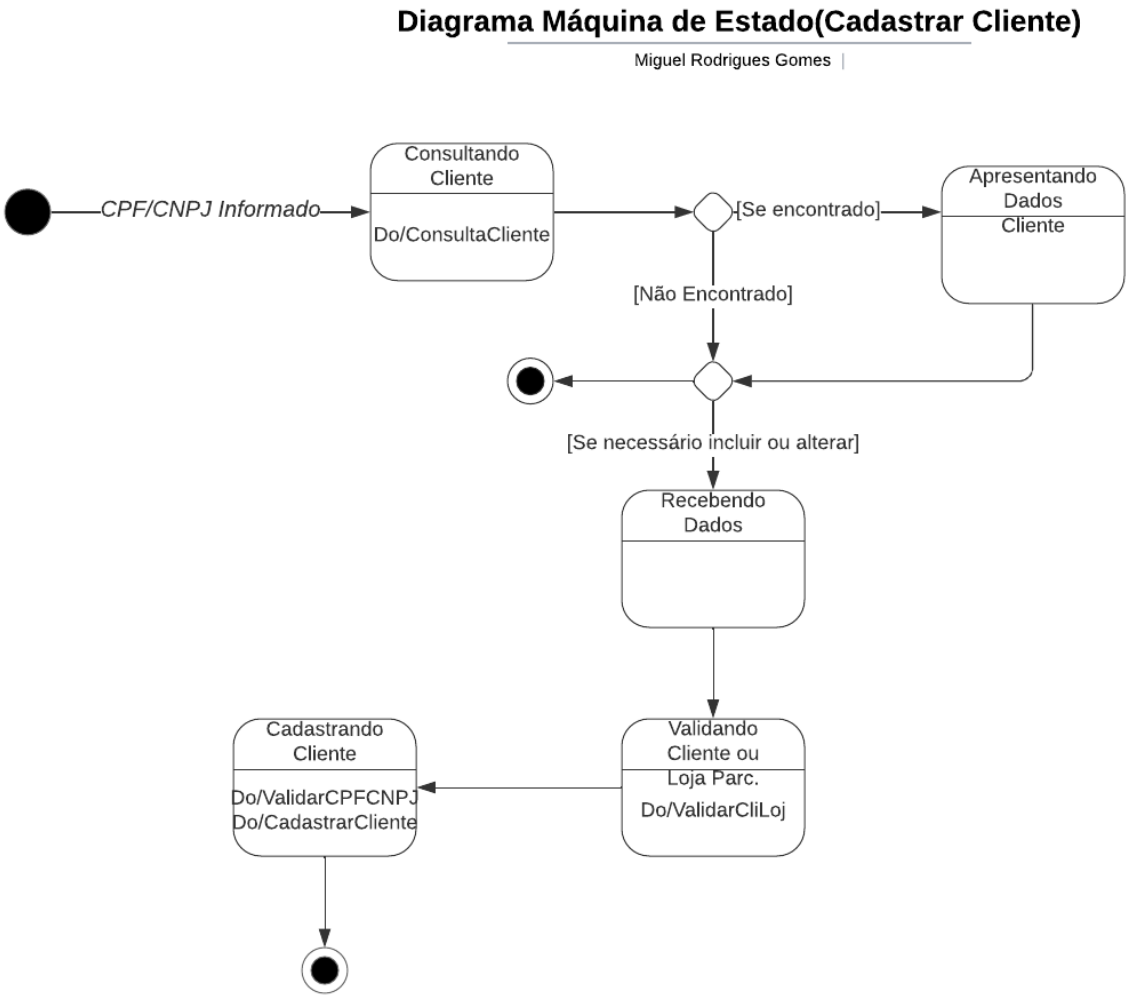


Fonte: o autor

3.7 Diagrama de Máquina de Estado

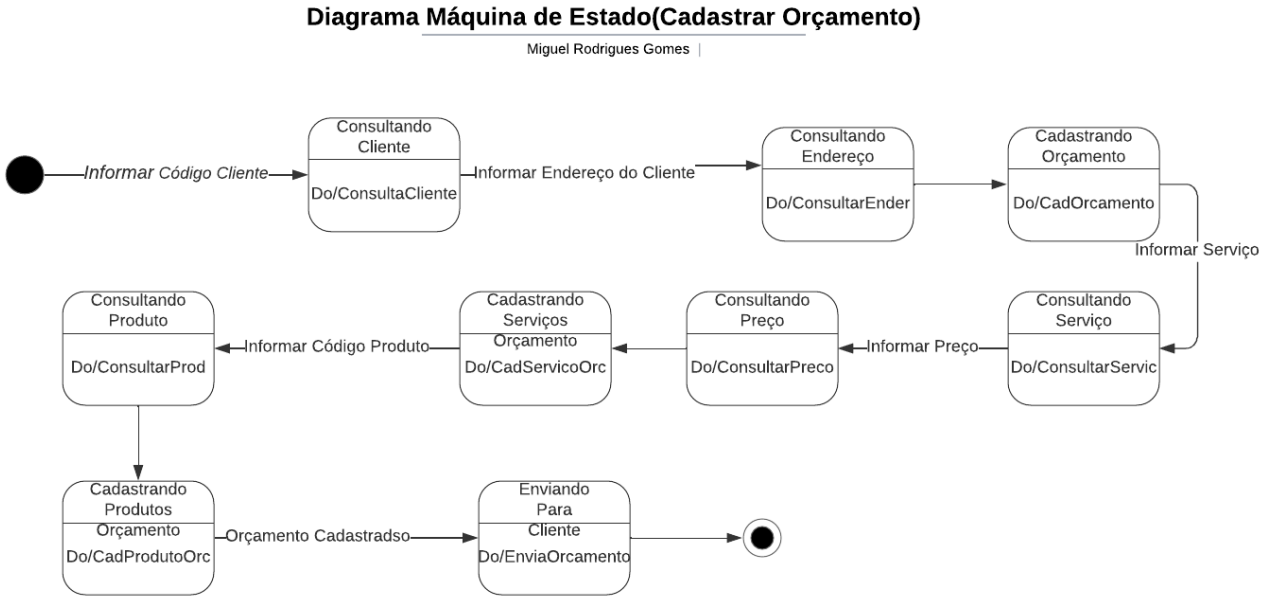
Segundo Booch, Rumbaugh e Jacobson (2012), os diagramas de máquinas de estados são fundamentais para entender como um objeto responde a eventos ao longo de seu ciclo de vida, ajudando a capturar a lógica de transições complexas e possíveis estados de objetos no sistema. Este diagrama é particularmente útil quando o comportamento de um sistema depende de uma sequência de eventos, permitindo uma visualização clara de como o sistema reage a diferentes estímulos.

Figura 6: Diagrama de Máquina de Estado (Cadastro de Cliente)



Fonte: o autor

Figura 7: Diagrama de Máquina de Estado (Cadastro de Orçamento)



Fonte: o autor

3.8 Modelagem do Banco de Dados

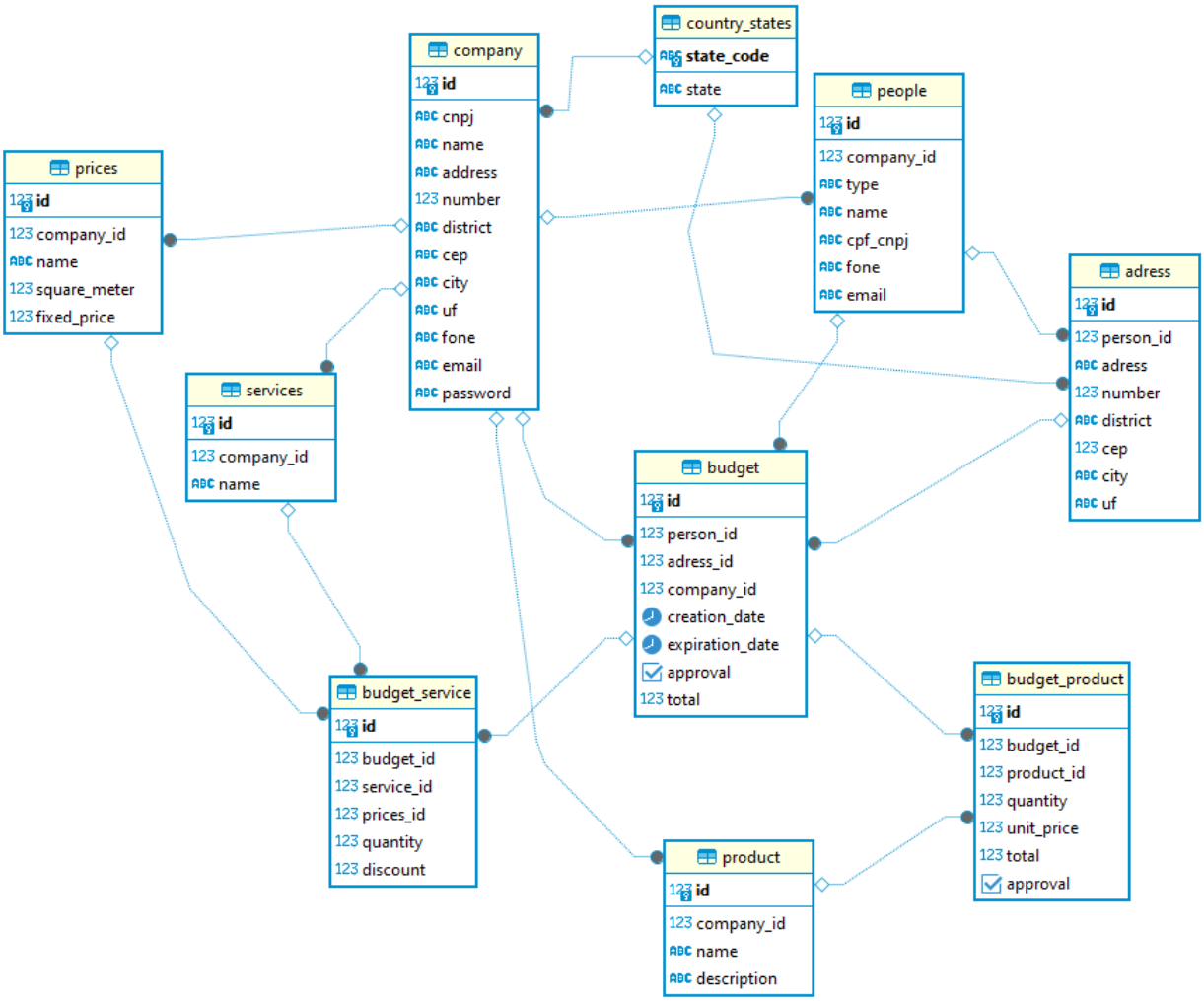
Conforme Elmasri e Navathe (2015), a modelagem de banco de dados é fundamental para garantir a integridade, consistência e desempenho de um sistema.

Ela começa com a modelagem conceitual, geralmente representada por diagramas de entidade-relacionamento (ER), que mapeiam as entidades, atributos e relacionamentos entre elas.

Em seguida, a modelagem lógica traduz esse modelo conceitual para um modelo específico de banco de dados, como o relacional.

Finalmente, a modelagem física aborda a implementação do banco de dados em um sistema de gerenciamento, considerando aspectos como indexação, particionamento e otimização de consultas.

Figura 8: Modelagem do Banco de Dados



Fonte: o autor

4 Desenvolvimento

Neste capítulo é mostrado como foi definida a comunicação do usuário com o sistema. Nesta seção são descritas técnicas de aplicação de desenvolvimento de protótipos de tela sobre como será a interface da aplicação.

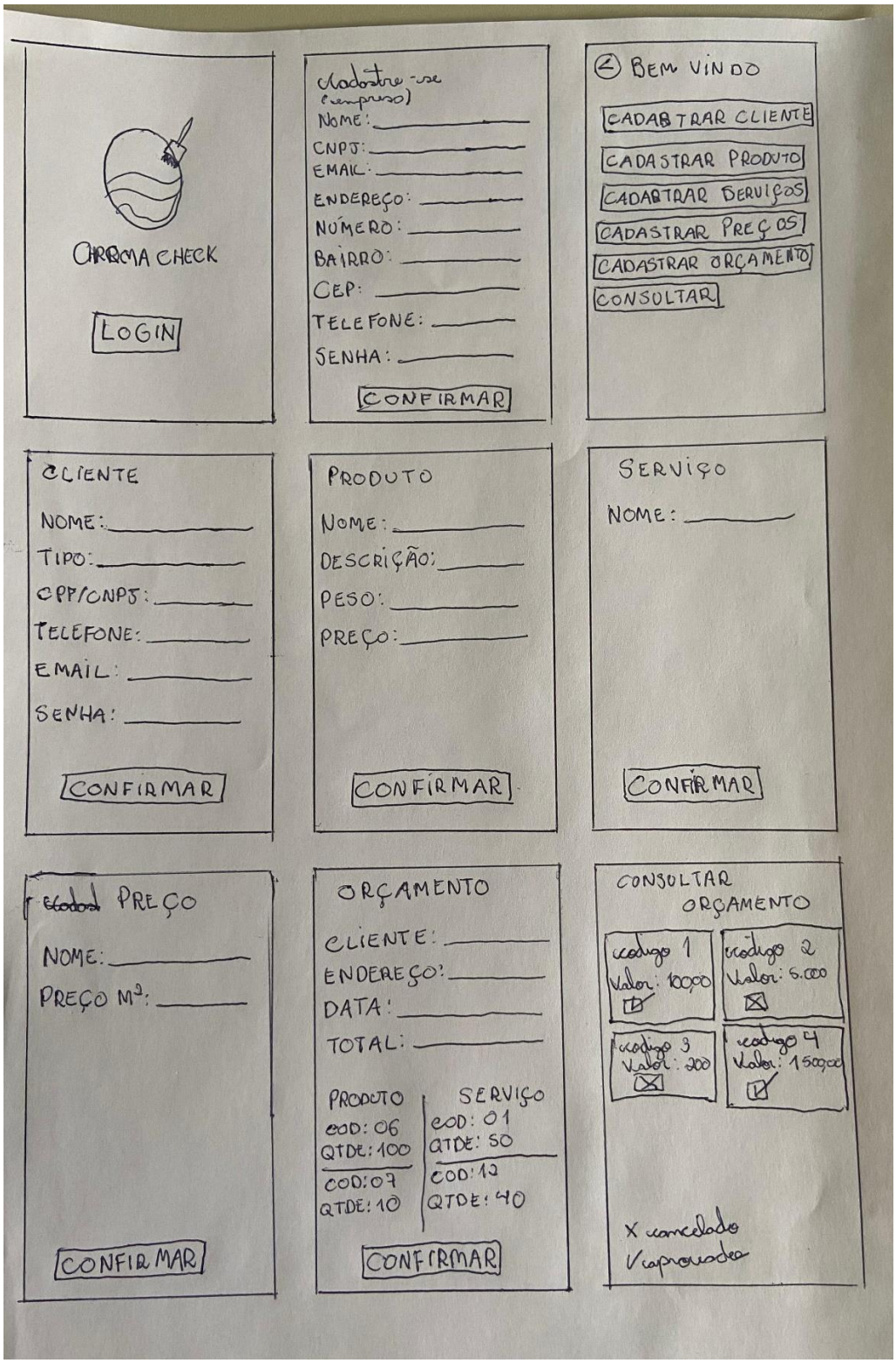
4.1 Cores

Para desenvolver a interface do sistema, foi escolhido uma paleta de cores que explora diferentes tons de azul, do azul escuro ao azul claro. Esta escolha não é apenas estética, mas também estratégica, visando transmitir sentimentos e ideias específicas ao usuário.

O azul é frequentemente associado a qualidades como harmonia, calma e tranquilidade. A utilização de diferentes tons de azul em toda a nossa interface visa criar um ambiente visual que promova sentimentos de calma e confiança.

4.2 Rabiscoframe

Figura 9: Rabiscoframe



Fonte: o autor

4.3 Arquitetura *backend*

Ao desenvolver o backend do software ChromaCheck, utilizamos uma arquitetura poderosa e moderna para garantir segurança, desempenho e escalabilidade. A escolha de técnicas e métodos é fundamental para atender aos requisitos de eficiência e precisão do sistema.

As tecnologias abaixo foram escolhidas para tornar o sistema de orçamentos de pintura residencial rápido, seguro e escalável. Java 17 garante alto desempenho e segurança para cálculos e processamento de orçamentos, enquanto PostgreSQL oferece armazenamento robusto e confiável para grandes volumes de dados. O modelo de Multi-Tenancy isola os dados de cada cliente, assegurando privacidade e proteção das informações. Swagger facilita a integração do sistema com outras ferramentas, permitindo fácil compartilhamento e atualização dos dados. Além disso, a estrutura MVC e as práticas de Clean Code mantêm o código organizado e fácil de expandir, garantindo a evolução do sistema conforme as necessidades dos clientes.

Logo abaixo, cada uma dessas tecnologias será apresentada detalhadamente.

4.3.1 Tecnologias e Arquitetura

Foi utilizado o Java 17 como linguagem principal para desenvolvimento backend. Java 17 oferece diversas melhorias de desempenho e segurança, bem como suporte para recursos de programação modernos, permitindo construir aplicativos confiáveis e eficientes.

Para armazenamento de dados, escolhemos o PostgreSQL, um banco de dados relacional avançado conhecido por sua robustez, confiabilidade e escalabilidade. A combinação de Java 17 e PostgreSQL garante que o sistema possa lidar com grandes quantidades de dados e transações complexas, mantendo alto desempenho e integridade das informações.

4.3.2 Modelo de Multi-Tenancy

A arquitetura do sistema é baseada em um modelo **Multi-Tenancy** onde cada cliente possui um banco de dados separado. Isso garante que os dados dos diferentes clientes sejam armazenados de forma isolada, proporcionando maior segurança e permitindo uma gestão eficiente da informação. Ao registar uma nova empresa, é criada uma base de dados separada para garantir que a integridade e a privacidade das informações sejam mantidas.

4.3.3 Integração e Documentação com Swagger

Para facilitar a integração e garantir uma comunicação eficiente entre os diferentes componentes do sistema, utilizamos Swagger para a documentação da API. O Swagger permite gerar documentação interativa e detalhada, facilitando a compreensão e o uso da API por outros desenvolvedores e integradores. A documentação gerada garante que a funcionalidade da API seja totalmente compreendida e utilizada corretamente, promovendo colaboração e integração eficaz.

4.3.4 Design Pattern e Melhores Práticas

No desenvolvimento da API, foi aplicado o padrão de design **MVC (Model, View, Controller)**, que separa a aplicação em camadas distintas. Essa abordagem organiza o código em hierarquias claras e define responsabilidades específicas para cada componente, o que resulta em um sistema mais modular, fácil de manter e expandir.

Além disso, seguimos práticas de **Clean Code** para garantir que o código seja limpo, legível e de alta qualidade. Essas práticas promovem uma manutenção mais fácil e segura do código, facilitando a identificação e correção de problemas e a implementação de novas funcionalidades.

A arquitetura do backend do sistema foi projetada para atender às necessidades específicas do software de orçamentos para serviços de pintura residencial, oferecendo segurança, eficiência e escalabilidade.

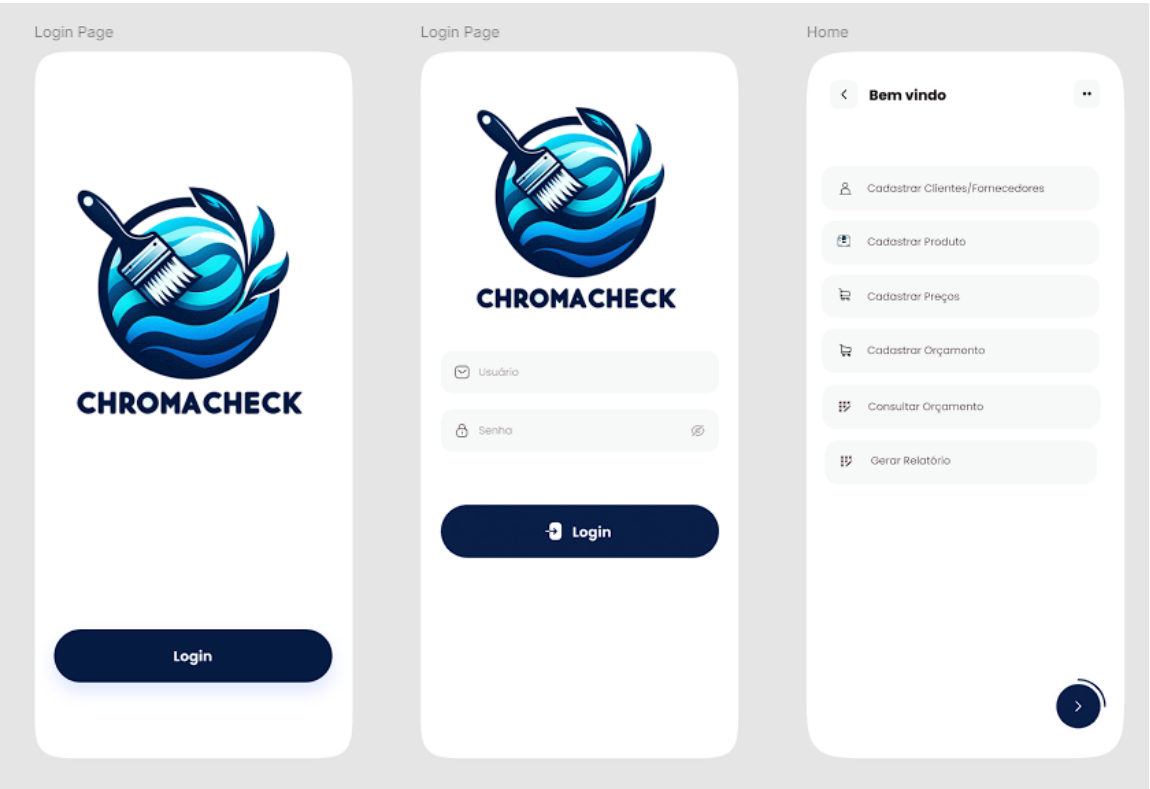
A combinação de Java 17, PostgreSQL, o modelo de Multi-Tenancy e as melhores práticas de desenvolvimento garante um sistema robusto e confiável, capaz de atender a uma ampla gama de projetos e requisitos dos clientes.

6 Resultados

6.1 Protótipo de Alta Resolução

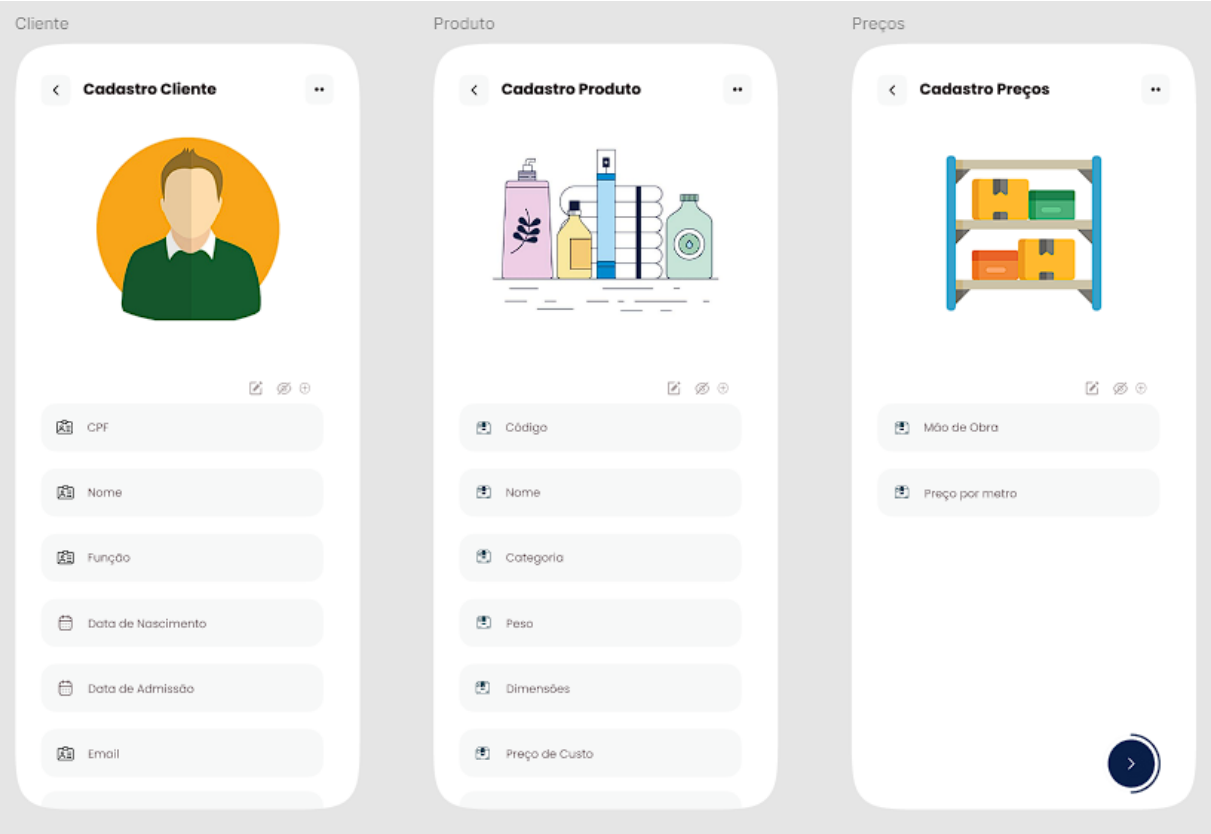
Protótipos com Alta Fidelidade têm mecanismos de respostas rápidas durante os testes. Estes modelos costumam se aproximar mais do software que os usuários terão contato. Significa que os usuários podem mexer nos protótipos como se realmente estivessem mexendo na aplicação de forma real (PERNICE, 2016).

Figura 10: Tela de Login e Principal



Fonte: o autor

Figura 11: Telas de cadastros



Fonte: o autor

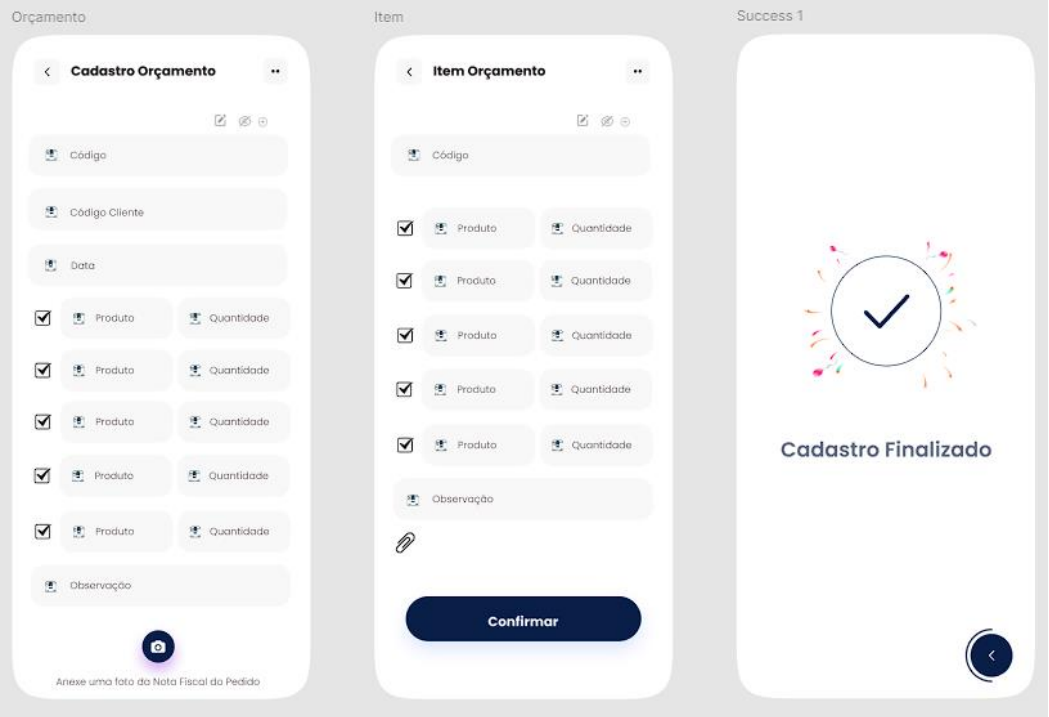
6.2 Desenvolvimento Backend

Logo abaixo será apresentado o código que define uma entidade CompanyEntity em uma aplicação Java com JPA (Java Persistence API) para mapeamento do banco de dados, representando uma tabela company. Cada instância de CompanyEntity representa uma empresa e inclui informações como CNPJ, nome, endereço, telefone e email, entre outros dados.

- **Anotações de JPA (@Entity, @Table, @Id, etc.):** Configuram o mapeamento da entidade para a tabela do banco de dados.
- **Atributos:** Cada campo corresponde a uma coluna da tabela company. Alguns campos possuem validações, como @NotBlank e @Positive, garantindo que sejam preenchidos e válidos.
- **Construtor com @Builder:** Facilita a criação de objetos CompanyEntity com todos os atributos.
- **Método setId:** Permite definir o Id da empresa, embora o acesso direto a esse campo esteja restrito (o @Setter(AccessLevel.NONE) no Id impede que seja modificado diretamente).

A entidade possui também anotações do Lombok (@Getter, @Setter, @NoArgsConstructor, etc.) para gerar automaticamente métodos getters e setters, construtores, e outros métodos auxiliares.

Figura 12: Telas de orçamentos



Fonte: o autor

Figura 13: CompanyEntity

```java
package br.com.java.tcc.application.company.persistence;

import ...

miguell-gomes@hotmail.com <miguell-gomes@hotmail.com> +1
@Entity
@Table(name = "company")
@NoArgsConstructor
@Getter
@Setter
public class CompanyEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false, unique = true)
    @Setter(AccessLevel.NONE)
    private Long Id;

    @Column(name = "cnpj", nullable = false, unique = true)
    private String cnpj;

    @NotBlank(message = "O nome é obrigatório e não pode estar vazio.")
    @Column(name = "name", nullable = false)
    private String name;

    @NotBlank(message = "O endereço é obrigatório e não pode estar vazio.")
    @Column(name = "address", nullable = false)
    private String address;

    @Positive(message = "O número deve ser maior que 0.")
    @Column(name = "number", nullable = false)
    private Integer number;

    @NotBlank(message = "O bairro é obrigatório e não pode estar vazio.")
    @Column(name = "district", nullable = false)
```

Fonte: o autor

O código abaixo define uma API REST para o gerenciamento de empresas, onde cada endpoint permite uma operação diferente sobre os dados de uma empresa. Ele utiliza o Spring Boot e Swagger para documentação.

- **findById:** Busca os dados de uma empresa específica com base no ID fornecido.
- **findByCnpjAndPassword:** Realiza uma busca de empresa com base em CNPJ e senha, retornando os dados se as credenciais forem válidas. Caso contrário, retorna uma resposta de não autorizado.
- **findAll:** Retorna uma lista com os dados de todas as empresas cadastradas.
- register: Cadastra uma nova empresa usando as informações recebidas no corpo da requisição e retorna o URI da empresa criada.
- **update:** Atualiza os dados de uma empresa específica com base no ID informado.
- **delete:** Exclui uma empresa com base no ID fornecido.

A classe utiliza anotações do Spring e Swagger para mapear os endpoints e descrever suas operações, facilitando a documentação e integração.

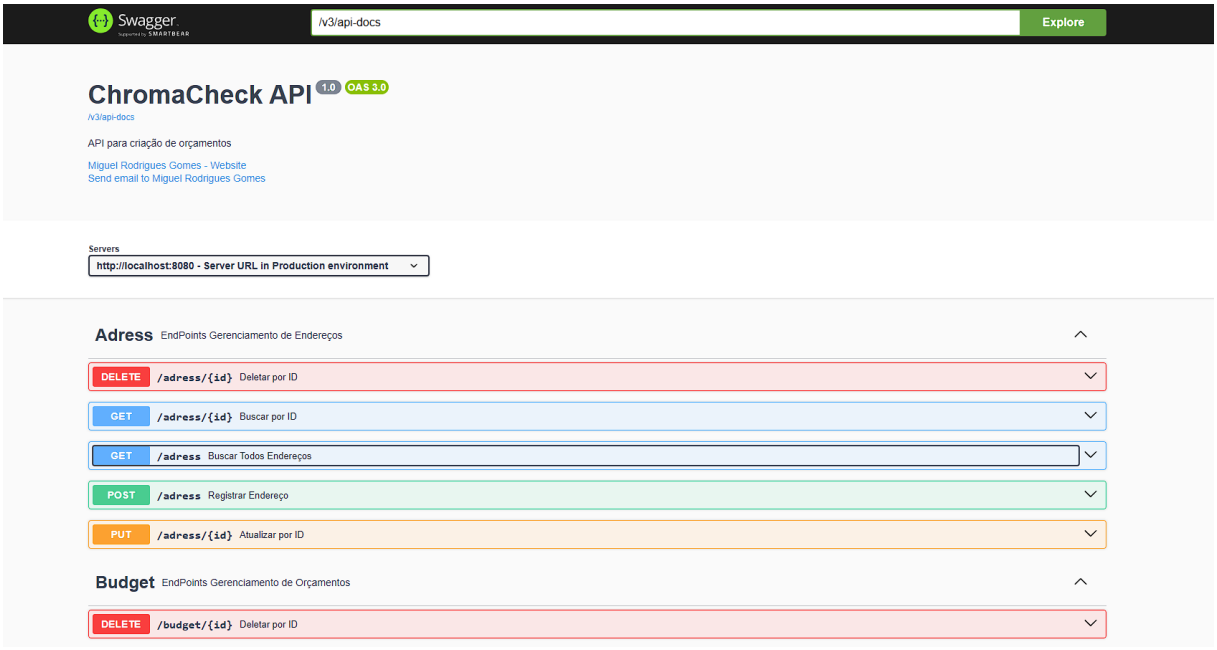
Figura 12: CompanyController

```
5 usages  Miguel Gomes +1
@RestController
@RequestMapping(value = "/company")
@Tag(name = "Company", description = "EndPoints Gerenciamento de Empresas")
@RequiredArgsConstructor
public class CompanyController {
    private final CompanyService companyService;

    miguell-gomes@hotmail.com <miguell-gomes@hotmail.com>
    @Operation(
            summary = "Buscar por ID",
            description = "Ao executar o endpoint irá retornar os dados de uma empresa respectivo ao id informado")
    @GetMapping(value = "/{id}")
    public ResponseEntity<CompanyResponse> findById(@PathVariable(name = "id") Long id){
        return ResponseEntity.ok().body(companyService.findById(id));
    }

    no usages  Miguel Gomes
    @Operation(
            summary = "Buscar empresa por CNPJ e senha",
            description = "Ao executar o endpoint irá retornar os dados de uma empresa com base no CNPJ e senha informados"
    )
    @GetMapping("/login")
    public ResponseEntity<CompanyResponse> findByCnpjAndPassword(
            @RequestParam(name = "cnpj") String cnpj,
            @RequestParam(name = "password") String password
    ) {
        // Implementar a lógica para buscar a empresa com base no CNPJ e na senha
        CompanyResponse companyResponse = companyService.findByCnpjAndPassword(cnpj, password);

        if (companyResponse != null) {
            return ResponseEntity.ok().body(companyResponse);
        } else {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(null);
        }
    }
```

Fonte: o autor

A seguir, está a implementação do Swagger, responsável por toda a documentação da API. Ele facilita o consumo dos endpoints ao fornecer uma interface interativa e detalhada, que auxilia desenvolvedores e integradores na compreensão e utilização dos serviços disponíveis.

Figura 13: Swagger



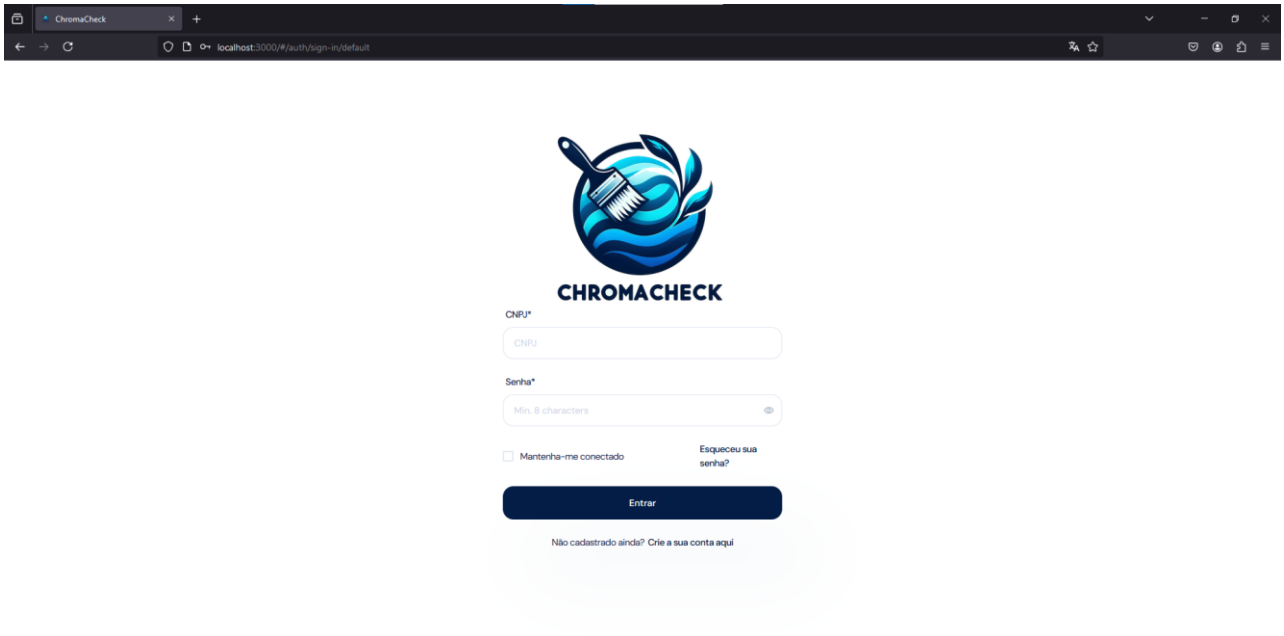
Fonte: o autor

6.2 Desenvolvimento Frontend

Nesta seção, será apresentado uma seleção de imagens que ilustram o desenvolvimento do front-end da aplicação, que foi realizado utilizando a biblioteca React. Durante o processo de criação, várias bibliotecas foram integradas para aprimorar a funcionalidade e a experiência do usuário. As bibliotecas utilizadas incluem:

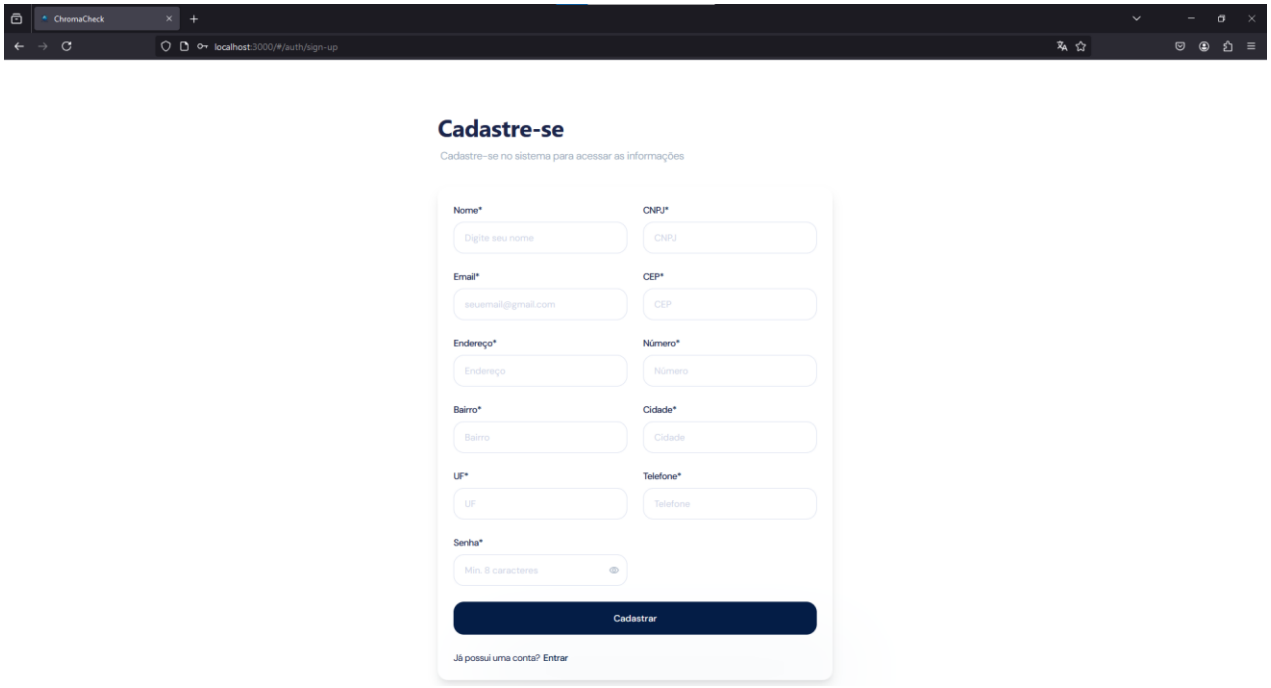
- **Chakra UI:** Para criar uma interface de usuário acessível e responsiva, oferecendo componentes prontos e estilos personalizáveis.
- **React Router:** Para gerenciar a navegação entre diferentes páginas e componentes da aplicação de forma eficiente.
- **Axios:** Para realizar requisições HTTP e facilitar a comunicação com APIs.
- **jsPDF** é uma biblioteca popular para geração de documentos PDF diretamente no navegador.

Figura 14: Login



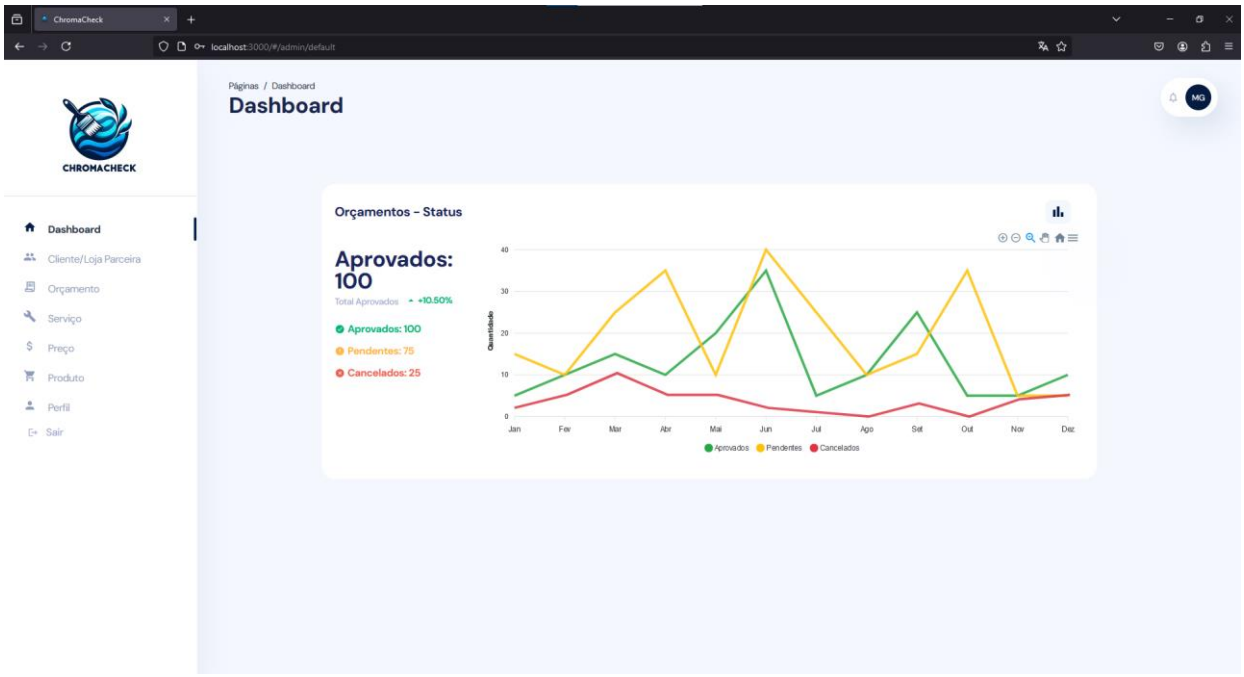
Fonte: o autor

Figura 15: Cadastro Empresa



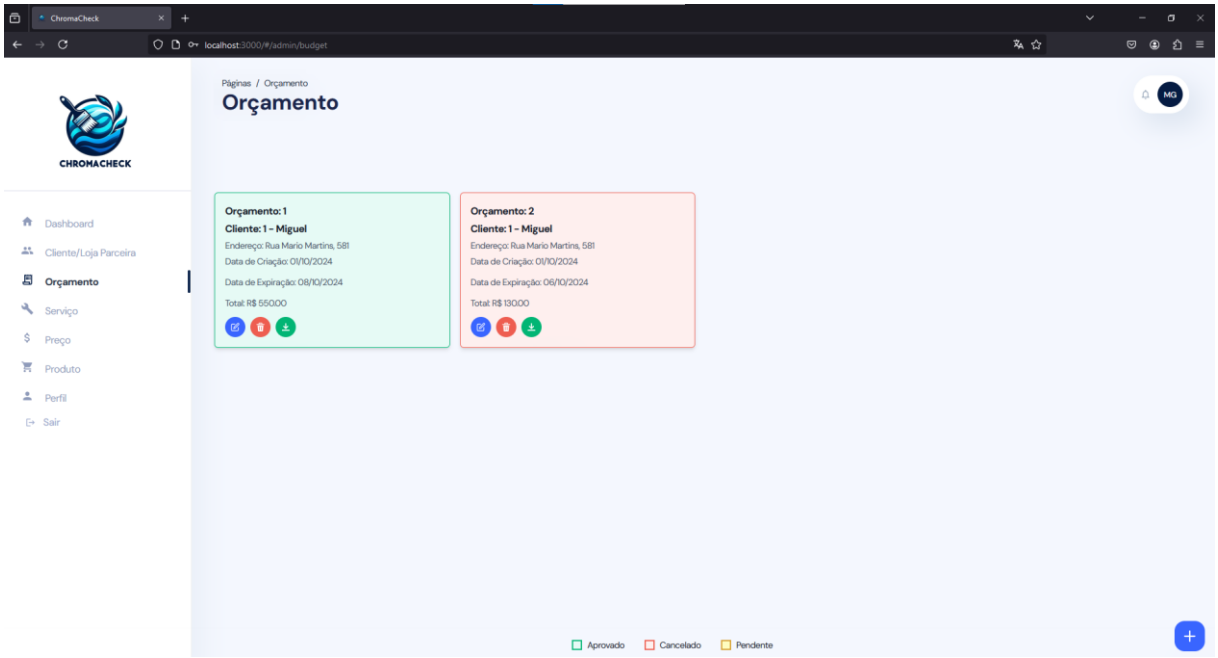
Fonte: o autor

Figura 16: Dashboard



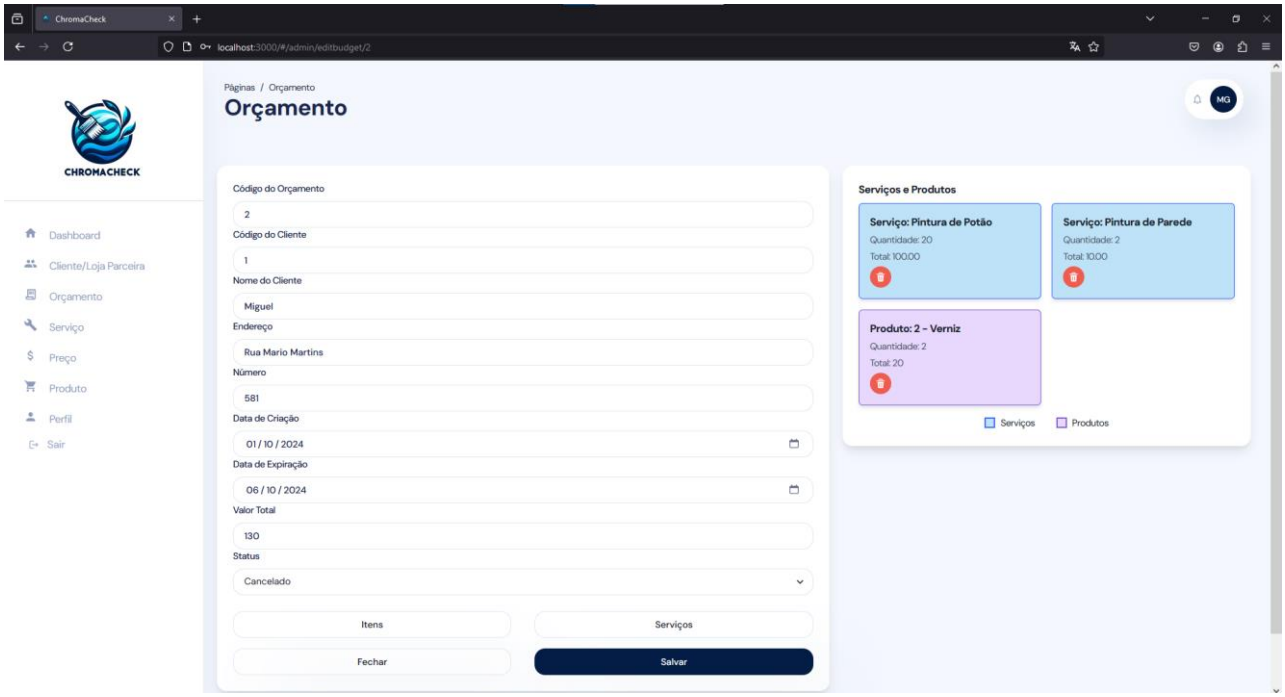
Fonte: o autor

Figura 17: Consultar Orçamentos



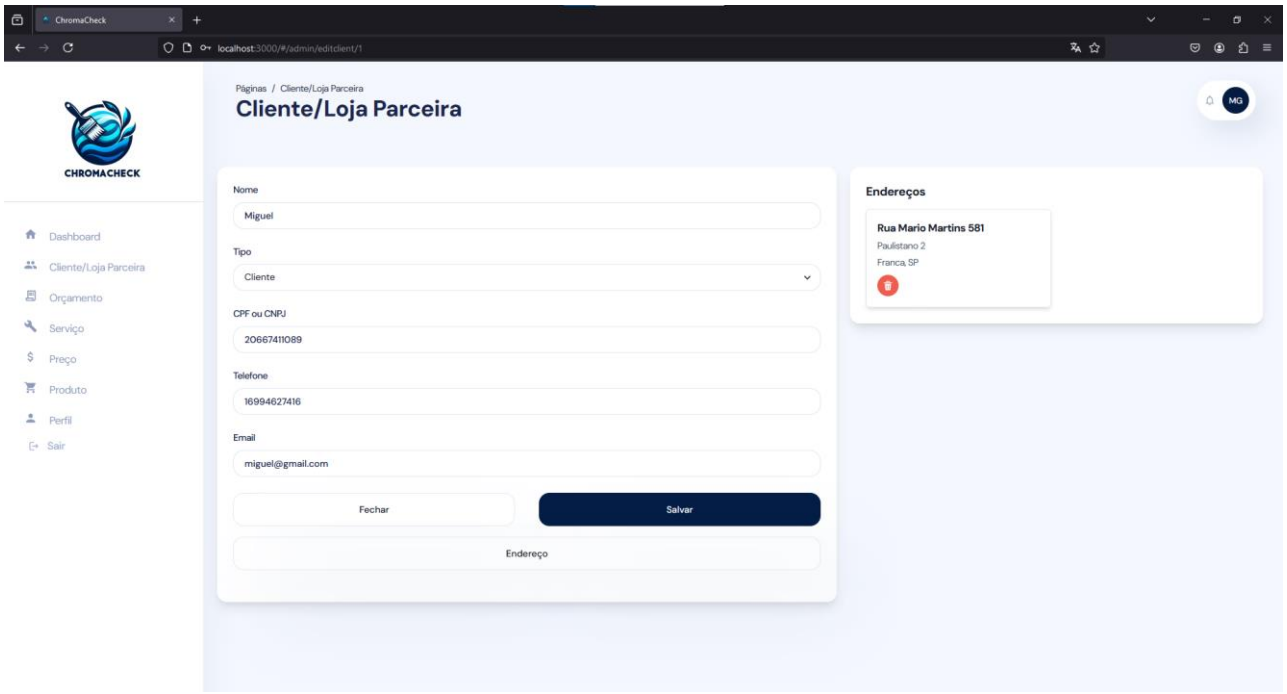
Fonte: o autor

Figura 18: Edição e Cadastro Orçamento



Fonte: o autor

Figura 19: Edição e Cadastro Orçamento



Fonte: o autor

7 Conclusão

Este projeto explorou o impacto da implementação de tecnologias digitais nos orçamentos de pintura residencial. Partindo de uma situação em que o tradicional processo manual de criação de orçamentos era demorado e sujeito a erros, com este software os prestadores poderão automatizar esta tarefa, que proporciona precisão e agilidade.

A implementação deste sistema trouxe uma mudança significativa no fluxo de trabalho dos profissionais de pintura, eliminando a necessidade de visitas demoradas ao local apenas para realizar medições e cálculos manuais. Com o software desenvolvido, esses profissionais podem gerar orçamentos de maneira ágil e padronizada, o que não só acelera o processo, mas também reduz a chance de erros humanos. Além disso, estão previstos testes de qualidade para garantir a precisão e a robustez do sistema, assegurando que ele atenda plenamente às necessidades dos usuários e funcione de forma confiável em diferentes cenários.

Os benefícios da adoção desta tecnologia incluem a otimização do tempo, a minimização dos recursos gastos em tarefas repetitivas e o fornecimento de orçamentos mais precisos para atender às expectativas dos clientes. Além disso, a padronização de processos traz maior confiança e transparência no relacionamento entre os profissionais de pintura e seus clientes.

Além disso, o software se adapta mais rapidamente às necessidades específicas de cada cliente e projeto, garantindo que os orçamentos não só sejam precisos, mas também personalizados. A integração de funcionalidades permite uma análise detalhada dos materiais e custos envolvidos, aumentando a transparência do processo, e assim a satisfação dos clientes e a confiança nos serviços prestados.

O projeto também destaca a importância da inovação contínua do setor de pintura residencial. A digitalização não deve ser vista apenas como uma tendência, mas como um pilar importante da competitividade e sustentabilidade a longo prazo. As futuras expansões do software, incluindo a incorporação de novas funcionalidades e expansão em diferentes geografias, trarão novas oportunidades para os profissionais de pintura, permitindo-lhes servir uma gama mais ampla de clientes de forma mais eficiente.

Por fim, a experiência adquirida com este projeto reforça a crença de que a tecnologia é uma poderosa aliada na transformação do setor de serviços, principalmente quando se trata de pintura residencial. Ser capaz de gerar orçamentos precisos em menos tempo e com maior transparência não só melhora a eficiência operacional, mas também fortalece o relacionamento com os clientes.

Referências

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. UML: guia do usuário. Tradução de Fábio Freitas da Silva e Cristina de Amorim Machado. – Rio de Janeiro: Elsevier, 2012. – 12ª reimpressão.

CARDOSO (2012). O ORÇAMENTO COMO SUPORTE A TOMADA DE DECISÃO. Disponível em:https://repositorio.ufmg.br/bitstream/1843/EMAE-98EG8U/1/monografia_alexsandro_22_06_definitiva.pdf. Acesso em 10 abr. 2024.

CUNHA. Pinturas. Disponível em:https://docente.ifrn.edu.br/valtencirgomes/disciplinas/construcao-civil-ii-1/pintura-apresentacao. Acesso em 10 abr. 2024.

ELMASRI, R.; NAVATHE, S. B. *Fundamentals of Database Systems*. 7ª edição. Pearson(2021)

Elmasri, R., & Navathe, S. B. (2015). FURTADO, Fernando. Clean Code: O que é, Casos de Uso, Exemplo de Código Limpo. 2019 Disponível em: https://www.alura.com.br/artigos/o-que-e-clean-code. Acesso em 21 ago. 2024.

GITHUB. GitHub Docs. Disponível em: https://docs.github.com/pt. Acesso em 10 abr. 2024.

GARRETT, F. O que é Figma?. (2021). Disponível em: https://www.techtudo.com.br/listas/2021/06/o-que-e-figma-quatro-perguntas-sobre-como-usar-o-site.ghtml. Acesso em 10 ago. 2024.

GRID. Swagger: entenda o que é e como usar. 2022. Disponível em: https://gr1d.io/2022/04/15/swagger/. Acesso em 20 ago. 2024.

HIGOR, Introdução ao Padrão MVC. 2013. Disponível em: https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308. Acesso em 18 ago. 2024.

IBM, Máquina de Estado. Disponível em: https://www.ibm.com/docs/pt-br/dmrt/9.5?topic=diagrams-state-machines. Acesso em 24 ago. 2024.

JOSÉ(2020). A História do Java. Disponível em: https://medium.com/@duduxss3/a-história-do-java-3bcb43f95f0b. Acesso em 10 abr. 2024.

LAURA. O que é pintura? Disponivel em: https://www.todamateria.com.br/o-que-e-pintura/. Acesso em 15 set. 2024.

PERNICE, K. UX Prototypes: Low Fidelity vs. High Fidelity. 2016. Disponível em: https://www.nngroup.com/articles/ux-prototype-hi-lo-fidelity/. Acesso em 10 ago. 2024.

POSTGRES, Documentation 2024. Disponível em: https://www.postgresql.org/docs/. Acesso em: 09 abr. 2024.

PRESSMAN, R.S. Engenharia de software: uma abordagem profissional; tradução Ariovaldo Griesi; revisão técnica Reginaldo Arakaki, Julio Arakaki, Renato Manzan de Andrade. – 7. ed. Porto Alegre : AMGH, 2011.

PRESSMAN, R. S. ; MAXIM, B.R.Engenharia de software: uma abordagem profissional. 8. ed. Porto Alegre : AMGH, 2016.

REACT .React: Uma biblioteca JavaScript para criar interfaces de usuário. 2022. Disponível em: https://pt-br.reactjs.org/. Acesso em 10 abr. 2024.

RED HAT. Arquitetura Multi Tenancy. 2020. Disponível em: https://www.redhat.com/pt-br/topics/cloud-computing/what-is-multitenancy. Acesso em 21 ago. 2024.

TRELLO (2022). O que é o Trello: conheça recursos, usos e muito mais | Trello. Disponível em: https://trello.com/pt-BR/tour. Acesso em 10 abr. 2024.

TOTVS, E. BPMN: entenda o que é a modelagem de processos de negócios, como fazer e sua importância! Disponível em: https://www.totvs.com/blog/gestaoindustrial/bpmn/. Acesso em: 10 ago. 2024.