-

**Eduardo Lopes Fonseca Gonzales**

**Gabriel Machado Dos Santos**

**Guilherme Angelo Silva**

**João Pereira Neto**

**SISTEMA GERENCIADOR DE DOCUMENTOS**

**Projeto Integrador**

Disciplinas Envolvidas: Engenharia de Software, Técnicas Avançadas de Banco de Dados Relacional e não Relacional, Técnicas Avançadas de Programação, Técnicas Avançadas de Programação Web e Mobile, Gestão Ágil de Projetos de Software.

Jales

2023

SUMÁRIO

[1 INTRODUÇÃO 3](#_Toc144809989)

[2 LEVANTAMENTO DE REQUISITOS DE SOFTWARE 4](#_Toc144809990)

[2.1 Descrição dos objetivos do sistema 4](#_Toc144809991)

[2.2 Descrição do sistema atual 4](#_Toc144809992)

[2.3 Descrição dos principais problemas 4](#_Toc144809993)

[2.4 Descrição dos requisitos funcionais 4](#_Toc144809994)

[2.5 Descrição dos requisitos não funcionais 4](#_Toc144809995)

[3 VISÃO DE CASO DE USO – UML 5](#_Toc144809996)

[3.1 Diagrama de Classes 5](#_Toc144809997)

[3.2 Dicionário de classes 5](#_Toc144809998)

[3.3 Definição dos Atores 5](#_Toc144809999)

[3.4 Lista de Casos de Uso 5](#_Toc144810000)

[3.4. Diagrama de Casos de Uso 5](#_Toc144810001)

[3.5. Diagrama de Casos de uso individuais 6](#_Toc144810002)

[3.6. Diagrama de Sequência 7](#_Toc144810003)

[3.7. Diagrama de Comunicação 7](#_Toc144810004)

[3.8. Diagrama de Atividade 7](#_Toc144810005)

[4 DEFINIÇÃO DA INTERFACE COM O USUÁRIO (UX) (3º semestre) 8](#_Toc144810006)

[4.1 Descrição de cenário 8](#_Toc144810007)

[4.2 Descrição de personas 8](#_Toc144810008)

[4.3 Esboços de tela (wireframes) 8](#_Toc144810009)

[4.4 Protótipos de tela 8](#_Toc144810010)

[5 BANCO DE DADOS 9](#_Toc144810011)

[5.1 Modelo Entidade Relacionamento 9](#_Toc144810012)

[5.2 Script das tabelas 9](#_Toc144810013)

[6 ARQUITETURA DE SOFTWARE 10](#_Toc144810014)

[6.1 Arquitetura de desenvolvimento 10](#_Toc144810015)

[6.2 Telas do sistema 10](#_Toc144810016)

[7 CONCLUSÃO 11](#_Toc144810017)

[8 REFERÊNCIAS 12](#_Toc144810018)

# INTRODUÇÃO

A geração de documentos de obra se torna fundamental para qualquer construção, sendo registros oficiais das etapas, decisões, normas e regulamentos que estão relacionados a uma obra.

Segundo Costa (2020), "O licenciamento para obras é imprescindível ao construir um imóvel. Quando um projeto para construção de um imóvel é aprovado pela prefeitura, significa que o mesmo atendeu à legislação e a construção pode ser iniciada após a liberação do alvará, documento autorizando o início dos serviços".

A documentação é essencial para garantir que os padrões de qualidade sejam preservados em todos os aspectos da obra. Sendo crítico para a segurança pública e durabilidade das infraestruturas que foram construídas, de forma que, tornando-se necessário para evitar multas, atraso no projeto e litígios legais.

Com isto, a gestão destes documentos e o atendimento de novas atualizações destes dados se tornam essenciais. Segundo MOBUSS (2018), “A gestão de documentos que conta com ferramentas inovadoras assegura a disseminação de dados rápida, eficiente e uniforme.”. Desta forma, uma gestão adequada permite que as informações podem ser localizadas rapidamente quando necessário, assegurando que o controle de documentos sensíveis e informações confidenciais não sejam divulgadas a pessoas não autorizadas. Muitas organizações estão sujeitas à uma rigorosidade em relação a documentos, portanto, gerir permite conformidade com regulamentações, diminuindo riscos legais e multas.

A incorporação de tecnologias modernas torna o trabalho de gestão mais eficiente e seguro. Com a digitalização dos documentos físicos e convertendo em formatos eletrônicos, reduz dependência a papéis físicos, facilita o armazenamento e a recuperação de informações. Por meio dela, possibilita medidas de segurança assertivas, com criptografia de dados e controle de acesso baseada em funções de pessoas dentro do sistema, restringindo que somente pessoas autorizadas possam ter acesso a tais informações.

Atualmente, na Prefeitura de Jales, a gestão dos documentos de aprovação de construções é realizada por um sistema que necessita de atualizações, resultando em dificuldades e prolongamento no processo de busca por esses documentos.

Nesse cenário, este projeto propõe o desenvolvimento de um software destinado ao gerenciamento de obras que tenham documentos aprovados, incluindo aqueles que estejam relacionados a esses registros. Este sistema oferecerá uma pesquisa mais robusta, com o objetivo de otimizar o tempo de resposta e minimizar o esforço necessário para localizar os documentos, ao mesmo tempo em que assegura o controle de acesso e a segurança desses arquivos.

# LEVANTAMENTO DE REQUISITOS DE SOFTWARE

|  |  |  |
| --- | --- | --- |
|  | Requisitos   Funcionais | Descrição |
| 1 | Cadastro de usuários | O sistema deve ser capaz de cadastrar usuários para utilização do sistema, desde atendente até administrador. |
| 2 | Autenticação de Usuários | O sistema deve verificar as credenciais (E-mail e senha) para realizar o login. |
| 3 | Controle de Acessos | O sistema deve verificar as credenciais apresentadas e apenas realizar o login se o usuário estiver cadastrado. |
| 4 | Alteração de usuários | O sistema deve ser capaz de realizar a alteração de dados de usuários se for necessário. |
| 5 | Listagem de usuários | O sistema deve listar os usuários para que o administrador possa manter o controle dos usuários. |
| 6 | Exclusão de usuários | O sistema deve ser capaz de excluir usuários via Administrador. |
| 7 | Desativação de usuários | O sistema deve ser capaz de desativar usuários via administrador dessa forma evitando que possam fazer login, preservando arquivos que dependem do mesmo. |
| 8 | Cadastro de Imóveis | O sistema deve permitir que os usuários autorizados cadastrem imóveis. |
| 9 | Alteração de Imóveis | O sistema deve permitir que os usuários autorizados alterem imóveis cadastrados. |
| 10 | Exclusão de imóveis | O sistema deve permitir que os usuários autorizados possam excluir imóveis do sistema |
| 11 | Desativação de imóveis | O sistema deve permitir que os usuários autorizados desativem imóveis, dessa forma não irá interagir no banco de dados apenas existindo para que arquivos dependentes do mesmo não sejam perdidos. |
| 12 | Cadastro de processos | O sistema deve permitir que os usuários autorizados façam o cadastro de processos. |
| 13 | Alteração de processos | O sistema deve permitir que os usuários autorizados façam alterações nos processos cadastrados. |
| 14 | Listagem de processos | O sistema deve listar os processos presentes nele. |
| 15 | Exclusão de processos | O sistema deve permitir que usuários autorizados façam a exclusão de processos |
| 16 | Desativação de processos | O sistema deve permitir a desativação de processos cadastrados por usuários autorizados, dessa forma arquivos dependentes não serão perdidos. |
| 17 | Cadastro de tipos de processo | O sistema deve permitir o cadastro do tipo do processo. |
| 18 | Alteração de tipos de processo | O sistema deve permitir alteração por usuários autorizados dos tipos de processos cadastrados. |
| 19 | Listagem de tipos de processo | O sistema deve listar todos os tipos de processos cadastrados. |
| 20 | Exclusão de tipos de processo | O sistema deve permitir usuários capazes excluir tipos de processos cadastrados. |
| 21 | Desativação de tipos de processo | O sistema deve permitir usuários autorizados desativar tipos de processos dessa forma arquivos dependentes não serão perdidos. |
| 22 | Cadastro de etapa | O sistema deve permitir o cadastro de etapas por usuários autorizados. |
| 23 | Alteração de etapa | O sistema deve permitir a alteração de etapas cadastradas por usuários autorizados. |
| 24 | Listagem de etapa | O sistema deve listar todas as etapas cadastradas. |
| 25 | Exclusão de etapa | O sistema deve permitir a exclusão de etapas cadastradas por usuários autorizados. |
| 26 | Desativação de etapa | O sistema deve permitir a desativação de etapas cadastradas por usuários autorizados, dessa forma arquivos dependentes não serão afetados. |
| 27 | Cadastro de tipos de etapa | O sistema deve permitir o cadastro de tipos de etapas por usuários autorizados |
| 28 | Alteração de tipos de etapa | O sistema deve permitir alterações para tipos de etapas por usuários autorizados. |
| 29 | Exclusão de tipos de etapa | O sistema deve permitir a exclusão de tipos de etapa cadastrados por usuários autorizados |
| 30 | Desativação de tipos de etapa | O sistema deve permitir a desativação de tipos de etapa cadastrados por usuários autorizados. |
| 31 | Listagem de tipos de etapa | O sistema deve listar todos os tipos de etapas cadastradas. |
| 32 | Cadastro de documentos | O sistema deve cadastrar documentos por meio de usuários autorizados. |
| 33 | Alteração de documentos | O sistema deve permitir alteração de documentos cadastrados no sistema por meio de usuários autorizados. |
| 34 | Exclusão de documentos | O sistema deve permitir a exclusão de documentos cadastrados no sistema por meio de usuários autorizados. |
| 35 | Desativação de documentos | O sistema deve permitir a desativação de documentos cadastrados no sistema por usuários autorizados, dessa forma arquivos dependentes não serão afetados. |
| 36 | Listagem de documentos | O sistema deve listar todos os documentos cadastrados. |
| 37 | Cadastro de tipos de documento | O sistema deve permitir o cadastro de tipos de documentos por usuários autorizados. |
| 38 | Alteração de tipos de documento | O sistema deve permitir a alteração de tipos de documentos por usuários autorizados |
| 39 | Exclusão de tipos de documento | O sistema deve permitir a exclusão de tipos de documentos por usuários autorizados. |
| 40 | Listagem de tipos de documento | O sistema deve listar todos os tipos de documentos cadastrados. |
| 41 | Desativação de tipos de documento | O sistema deve permitir a desativação de tipos de documento por usuários autorizados, assim arquivos dependentes não serão afetados. |

## Descrição dos objetivos do sistema

O sistema a ser desenvolvido tem como objetivo principal simplificar e aprimorar o acesso e busca aos arquivos e documentos relacionados aos projetos de obras da Prefeitura de Jales, centralizando e organizando de forma mais estruturada.

## Descrição do sistema atual

O sistema atual utilizado pela Secretaria de Obras revela-se inadequado para suprir as crescentes necessidades e demandas cotidianas do órgão, principalmente devido à sua limitação quanto à capacidade de armazenamento de informações. Esta escassez de espaço para registros se traduz em obstáculos consideráveis na tarefa de localizar e gerenciar os dados essenciais.

Essa deficiência prejudica significativamente a eficiência administrativa e a capacidade de busca e resposta da Secretaria de Obras aos arquivos e documentos, sendo assim urgente a busca por uma solução mais robusta e eficaz para gerenciamento de informações.

## Descrição dos principais problemas

A equipe do projeto identificou um problema relacionado ao desenvolvimento do banco de dados. Ele envolve a necessidade de armazenar uma vasta quantidade de dados, bem como informações sensíveis. Portanto, não será permitido a negligência da segurança do banco de dados.

## Descrição dos requisitos funcionais

|  |  |  |
| --- | --- | --- |
|  | Requisitos não funcionais | Descrição |
| 1 | Atualização do banco de dados | Aprimorar a agilidade na atualização do banco de dados do cliente é essencial para evitar a presença de informações desatualizadas no sistema. |
| 2 | Máquina compatível com navegador | É fundamental que o dispositivo suporte um navegador para acessar e utilizar o sistema. |
| 3 | Auxiliar na atualização do sistema | Vamos fornecer suporte aos usuários, orientando-os a utilizar o software de maneira mais produtiva. |

### Login Administrador

O sistema deve permitir que o usuário consiga efetuar login como administrador, validando se os dados informados estão corretos.

### Alterar Administrador

O sistema deve permitir o administrador alterar seus dados, caso precise.

### Logout Administrador

O sistema deve permitir que o administrador efetue logout de sua sessão, retornando a página inicial do sistema.

### Cadastrar Atendente

O sistema deve permitir que o administrador cadastre um atendente no sistema.

### Listar Atendente

O sistema deve permitir que o administrador liste todos os atendentes cadastrados no sistema.

### Desativar Atendente

O sistema deve permitir que o administrador desative um atendente cadastrado no sistema, colocando seu status como “Desativado”, impossibilitando o usuário efetuar login com a conta afetada, existindo somente para que outros registros dependentes dele não sejam removidos ou corrompidos.

### Excluir Atendente

O sistema deve permitir que o administrador exclua um atendente cadastrado no sistema.

Caso o atendente tenha participado de algum registro no banco de dados, efetua Desativar Atendente.

Caso não, ele o removerá do banco de dados, efetuando o logout da sessão aberta para a conta afetada, caso exista.

### Login Atendente

O sistema deve permitir que o usuário consiga efetuar login como atendente, validando se os dados informados estão corretos.

### Alterar Atendente

O sistema deve permitir que o administrador e o atendente alterem seus dados, caso precise.

### Logout Atendente

O sistema deve permitir que o atendente efetue logout de sua sessão, retornando a página inicial do sistema.

### Cadastrar Usuário Público

O sistema deve permitir que o administrador e o atendente cadastrem um usuário público no sistema.

### Listar Usuário Público

O sistema deve permitir que o administrador e o atendente listem todos os usuários públicos cadastrados no sistema.

### Desativar Usuário Público

O sistema deve permitir que o administrador e o atendente desativem um usuário público cadastrado no sistema, colocando seu status como “Desativado”, impossibilitando o usuário efetuar login com a conta afetada, existindo somente para que outros registros dependentes dele não sejam removidos ou corrompidos.

### Excluir Usuário Público

O sistema deve permitir que o administrador e o atendente excluam um usuário público cadastrado no sistema.

Caso o Usuário público tenha participado de algum registro no banco de dados, efetua Desativar Usuário Público.

Caso não, ele o removerá do banco de dados, efetuando o logout da sessão aberta para a conta afetada, caso exista.

### Login Usuário Público

O sistema deve permitir que o usuário consiga efetuar login como usuário público, validando se os dados informados estão corretos.

### Alterar Usuário Público

O sistema deve permitir que o administrador, atendente e o usuário público alterem seus dados, caso precise.

### Logout Usuário Público

O sistema deve permitir que o usuário público efetue logout de sua sessão, retornando a página inicial do sistema.

### Cadastrar Munícipe

O sistema deve permitir que o administrador cadastre um munícipe no sistema.

### Listar Munícipe

O sistema deve permitir que o administrador, atendente e usuários públicos listem todos os munícipes cadastrados no sistema.

### Alterar Munícipe

O sistema deve permitir que o administrador e o atendente alterem os dados do munícipe, caso precise.

### Desativar Munícipe

O sistema deve permitir que o administrador e o atendente desativem um munícipe cadastrado no sistema, colocando seu status como “Desativado”, assim, o registro não interage mais no banco de dados, existindo somente para que outros registros dependentes dele não sejam removidos ou corrompidos.

### Excluir Munícipe

O sistema deve permitir que o administrador e o atendente excluam um munícipe cadastrado no sistema.

Caso o engenheiro tenha participado de algum registro no banco de dados, efetua Desativar Munícipe.

Caso não, ele o removerá do banco de dados.

### Cadastrar Engenheiro

O sistema deve permitir que o administrador cadastre um engenheiro no sistema.

### Listar Engenheiro

O sistema deve permitir que o administrador, atendente e usuários públicos listem todos os engenheiros cadastrados no sistema.

### Alterar Engenheiro

O sistema deve permitir que o administrador e o atendente alterem os dados do engenheiro, caso precise.

### Desativar Engenheiro

O sistema deve permitir que o administrador e o atendente desativem um engenheiro cadastrado no sistema, colocando seu status como “Desativado”, assim, o registro não interage mais no banco de dados, existindo somente para que outros registros dependentes dele não sejam removidos ou corrompidos.

### Excluir Engenheiro

O sistema deve permitir que o administrador e o atendente excluam um engenheiro cadastrado no sistema.

Caso o engenheiro tenha participado de algum registro no banco de dados, efetua Desativar Engenheiro.

Caso não, ele o removerá do banco de dados.

### Cadastrar Fiscal

O sistema deve permitir que o administrador cadastre um fiscal no sistema.

### Listar Fiscal

O sistema deve permitir que o administrador, atendente e usuários públicos listem todos os fiscais cadastrados no sistema.

### Alterar Fiscal

O sistema deve permitir que o administrador e o atendente alterem os dados do fiscal, caso precise.

### Desativar Fiscal

O sistema deve permitir que o administrador e o atendente desativem um fiscal cadastrado no sistema, colocando seu status como “Desativado”, assim, o registro não interage mais no banco de dados, existindo somente para que outros registros dependentes dele não sejam removidos ou corrompidos.

### Excluir Fiscal

O sistema deve permitir que o administrador e o atendente excluam um fiscal cadastrado no sistema.

Caso o fiscal tenha participado de algum registro no banco de dados, efetua Desativar Fiscal.

Caso não, ele o removerá do banco de dados.

### Cadastrar Endereço

O sistema deve permitir que o administrador cadastre um endereço no sistema.

### Listar Endereço

O sistema deve permitir que o administrador, atendente e usuários públicos listem todos os endereços cadastrados no sistema.

### Alterar Endereço

O sistema deve permitir que o administrador e o atendente alterem os dados do endereço, caso precise.

### Desativar Endereço

O sistema deve permitir que o administrador e o atendente desativem um endereço cadastrado no sistema, colocando seu status como “Desativado”, assim, o registro não interage mais no banco de dados, existindo somente para que outros registros dependentes dele não sejam removidos ou corrompidos.

### Excluir Endereço

O sistema deve permitir que o administrador e o atendente excluam um endereço cadastrado no sistema.

Caso o endereço tenha participado de algum registro no banco de dados, efetua Desativar Endereço.

Caso não, ele o removerá do banco de dados.

### Cadastrar Imóvel

O sistema deve permitir que o administrador cadastre um imóvel no sistema.

### Listar Imóvel

O sistema deve permitir que o administrador, atendente e usuários públicos listem todos os imóveis cadastrados no sistema.

### Alterar Imóvel

O sistema deve permitir que o administrador e o atendente alterem os dados do imóvel, caso precise.

### Desativar Imóvel

O sistema deve permitir que o administrador e o atendente desativem um imóvel cadastrado no sistema, colocando seu status como “Desativado”, assim, o registro não interage mais no banco de dados, existindo somente para que outros registros dependentes dele não sejam removidos ou corrompidos.

### Excluir Imóvel

O sistema deve permitir que o administrador e o atendente excluam um imóvel cadastrado no sistema.

Caso o imóvel tenha participado de algum registro no banco de dados, efetua Desativar Imóvel.

Caso não, ele o removerá do banco de dados.

### Cadastrar TipoProcesso

O sistema deve permitir que o administrador cadastre um tipo processo no sistema.

### Listar TipoProcesso

O sistema deve permitir que o administrador, atendente e usuários públicos listem todos os tipo de processos cadastrados no sistema.

### Alterar TipoProcesso

O sistema deve permitir que o administrador e o atendente alterem os dados do tipo de processo, caso precise.

### Desativar TipoProcesso

O sistema deve permitir que o administrador e o atendente desativem um tipo de processo cadastrado no sistema, colocando seu status como “Desativado”, assim, o registro não interage mais no banco de dados, existindo somente para que outros registros dependentes dele não sejam removidos ou corrompidos.

### Excluir TipoProcesso

O sistema deve permitir que o administrador e o atendente excluam um tipo de processo cadastrado no sistema.

Caso o tipo de processo tenha participado de algum registro no banco de dados, efetua Desativar TipoProcesso.

Caso não, ele o removerá do banco de dados.

### Cadastrar Processo

O sistema deve permitir que o administrador cadastre um processo no sistema.

### Listar Processo

O sistema deve permitir que o administrador, atendente e usuários públicos listem todos os processos cadastrados no sistema.

### Alterar Processo

O sistema deve permitir que o administrador e o atendente alterem os dados do processo, caso precise.

### Desativar Processo

O sistema deve permitir que o administrador e o atendente desativem um processo cadastrado no sistema, colocando seu status como “Desativado”, assim, o registro não interage mais no banco de dados, existindo somente para que outros registros dependentes dele não sejam removidos ou corrompidos.

### Excluir Processo

O sistema deve permitir que o administrador e o atendente excluam um processo cadastrado no sistema.

Caso o processo tenha participado de algum registro no banco de dados, efetua Desativar Processo.

Caso não, ele o removerá do banco de dados.

### Cadastrar TipoEtapa

O sistema deve permitir que o administrador cadastre um tipo de etapa no sistema.

### Listar TipoEtapa

O sistema deve permitir que o administrador, atendente e usuários públicos listem todos os tipo de etapas cadastrados no sistema.

### Alterar TipoEtapa

O sistema deve permitir que o administrador e o atendente alterem os dados do tipo de etapa, caso precise.

### Desativar TipoEtapa

O sistema deve permitir que o administrador e o atendente desativem um tipo de etapa cadastrado no sistema, colocando seu status como “Desativado”, assim, o registro não interage mais no banco de dados, existindo somente para que outros registros dependentes dele não sejam removidos ou corrompidos.

### Excluir TipoEtapa

O sistema deve permitir que o administrador e o atendente excluam um tipo de etapa cadastrado no sistema.

Caso o tipo de etapa tenha participado de algum registro no banco de dados, efetua Desativar TipoEtapa.

Caso não, ele o removerá do banco de dados.

### Cadastrar Etapa

O sistema deve permitir que o administrador cadastre uma etapa no sistema.

### Listar Etapa

O sistema deve permitir que o administrador, atendente e usuários públicos listem todos as etapas cadastrados no sistema.

### Alterar Etapa

O sistema deve permitir que o administrador e o atendente alterem os dados da etapa, caso precise.

### Desativar Etapa

O sistema deve permitir que o administrador e o atendente desativem uma etapa cadastrado no sistema, colocando seu status como “Desativado”, assim, o registro não interage mais no banco de dados, existindo somente para que outros registros dependentes dele não sejam removidos ou corrompidos.

### Excluir Etapa

O sistema deve permitir que o administrador e o atendente excluam uma etapa cadastrado no sistema.

Caso a etapa tenha participado de algum registro no banco de dados, efetua Desativar Etapa.

Caso não, ele o removerá do banco de dados.

### Cadastrar TipoDocumento

O sistema deve permitir que o administrador cadastre um tipo de documento no sistema.

### Listar TipoDocumento

O sistema deve permitir que o administrador, atendente e usuários públicos listem todos os tipos de documentos cadastrados no sistema.

### Alterar TipoDocumento

O sistema deve permitir que o administrador e o atendente alterem os dados do tipo de documento, caso precise.

### Desativar TipoDocumento

O sistema deve permitir que o administrador e o atendente desativem um tipo de documento cadastrado no sistema, colocando seu status como “Desativado”, assim, o registro não interage mais no banco de dados, existindo somente para que outros registros dependentes dele não sejam removidos ou corrompidos.

### Excluir TipoDocumento

O sistema deve permitir que o administrador e o atendente excluam um tipo de documento cadastrado no sistema.

Caso o tipo de documento tenha participado de algum registro no banco de dados, efetua Desativar TipoDocumento.

Caso não, ele o removerá do banco de dados.

### Cadastrar Documento

O sistema deve permitir que o administrador cadastre um documento no sistema.

### Listar Documento

O sistema deve permitir que o administrador, atendente e usuários públicos listem todos os documentos cadastrados no sistema.

### Alterar Documento

O sistema deve permitir que o administrador e o atendente alterem os dados do documento, caso precise.

### Desativar Documento

O sistema deve permitir que o administrador e o atendente desativem um documento cadastrado no sistema, colocando seu status como “Desativado”, assim, o registro não interage mais no banco de dados, existindo somente para que outros registros dependentes dele não sejam removidos ou corrompidos.

### Excluir Documento

O sistema deve permitir que o administrador e o atendente excluam um documento cadastrado no sistema.

Caso o documento tenha participado de algum registro no banco de dados, efetua Desativar Documento.

Caso não, ele o removerá do banco de dados.

## Descrição dos requisitos não funcionais

2.5.1 Atualização do banco de dados.

Atualizar o banco de dados do cliente de maneira rápida, para que não haja dados desatualizados no sistema.

2.5.2 A máquina precisa suportar um navegador de internet.

Necessário que a máquina possua um navegador para utilização do sistema.

2.5.3 Auxiliar usuários na utilização do software.

Iremos auxiliar os usuários para que possam utilizar o software da maneira mais produtiva possível.

# VISÃO DE CASO DE USO – UML

## 3.1 Diagrama de Classes

*Fazer um texto explicativo sobre o diagrama e inserir abaixo. As imagens devem conter o nome da figura e abaixo a fonte*

*Colocar em outra cor as classes que foram implementadas nas aulas de programação*

## 3.2 Dicionário de classes

Descrever cada uma das classes do diagrama e seus atributos. Exemplo:

Administrador: tem como objetivo controlar o fluxo de acesso de cliente e produtor e medir o nível de satisfação do cliente/produtor.

**Quadro 1** – Descrição Classe Administrador

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idAdministrador | Integer | Código que irá identificar o administrador. |

Fonte: Elaborado pelos autores

## 3.3 Definição dos Atores

*Inserir a figura dos atores e abaixo fazer uma descrição sobre o ator (es).*

## 3.4 Lista de Casos de Uso

*(Descrever sobre a Lista de Casos de Uso)*

Quadro 1 – Lista de Casos de Uso

| **Nº** | **Descrição do Caso de Uso** | **Entrada** | **Caso de Uso** | **Resposta** |
| --- | --- | --- | --- | --- |
| 01 | Funcionário Cadastra Livro | dados livro | Cadastrar Livro | Msg01 |
| 02 | Funcionário Cadastra Autor | dados autor | Cadastrar Autor | Msg02 |

## 3.4. Diagrama de Casos de Uso

*Fazer um texto explicativo sobre o diagrama e inserir-lo abaixo. As imagens deve conter o nome da figura e abaixo a fonte.*

**Figura 01** — Diagrama de Contexto – Visão do Paciente.



Fonte: Silva, 2016.

## 3.5. Diagrama de Casos de uso individuais

*(2 principais: cadastro, consulta, relatório, agendamento, controle, etc..)*

*Inserir o diagrama e abaixo colocar a documentação do mesmo.*

3.5.1 – Caso de uso: Cadastrar Atendimento

**Figura 02** — Diagrama de Contexto – Visão do Paciente

**

Fonte: Silva, 2016.

**Fluxo Normal**

1-AtorAdm solicita exclusão do registro da pessoa

2-Sistema envia mensagem 06: "Deseja excluir o registro?"

3-AtorAdm confirma exclusão do registro

4-Sistema exclui registro

5-Sistema envia msg07: "Registro excluído com sucesso!"

6-Sistema atualiza banco de dados e retorna ao Caso de Uso "Alterar Pessoa"

**Fluxo Exceção**

2-Sistema envia mensagem 06: "Deseja excluir o registro?"

2.1-AtorAdm não confirma exclusão do registro

2.2-Sistema cancela ação

2.3-Sistema retorna ao caso de Uso "Alterar Pessoa"

Fluxo Alternativo

4-Sistema exclui registro

4.1-Sistema identifica erros de conexão com banco de dados ao excluir registro

4.2-Sistema envia msg07:"Erro s de conexão de BD"

4.3-Sistema retorna ao item 1

## 3.6. Diagrama de Sequência

*(2 principais: cadastro, consulta, relatório, agendamento, controle, etc..)*

*Inserir o diagrama e abaixo colocar a documentação do mesmo.*

## 3.7. Diagrama de Comunicação

*(1 diagrama relevante do projeto)*

## 3.8. Diagrama de Atividade

*(1 diagrama de uma ação específica do sistema)*

# DEFINIÇÃO DA INTERFACE COM O USUÁRIO (UX) (3º semeestre)

## Descrição de cenário

(citar referências sobre o assunto e descrever pelo menos 2 cenários de uso relevantes para o sistema)

## Descrição de personas

(citar referências sobre o assunto e descrever pelo menos 2 personas relevantes para o sistema)

## Esboços de tela (wireframes)

(fazer o wireframe da tela de login, tela principal e dois exemplos de cadastros relevantes para o sistema e descrever sobre a imagem que foi colocada)

## Protótipos de tela

(fazer o protótipo da tela de login, tela principal e dos dois exemplos de cadastros relevantes para o sistema, preferencialmente em Html e css ou em bootstrap, se não for possível fazer o protótipo em algum aplicativo de edição pertinente)

(descrever sobre as telas demonstradas)

# BANCO DE DADOS

## Modelo Entidade Relacionamento

Um banco de dados é uma coleção organizada de dados armazenados eletronicamente em um sistema de computador, geralmente controlado por um Sistema de Gerenciamento de Banco de Dados, a maioria dos bancos de dados usa a linguagem de consulta estruturada (SQL) para escrever e consultar dados.

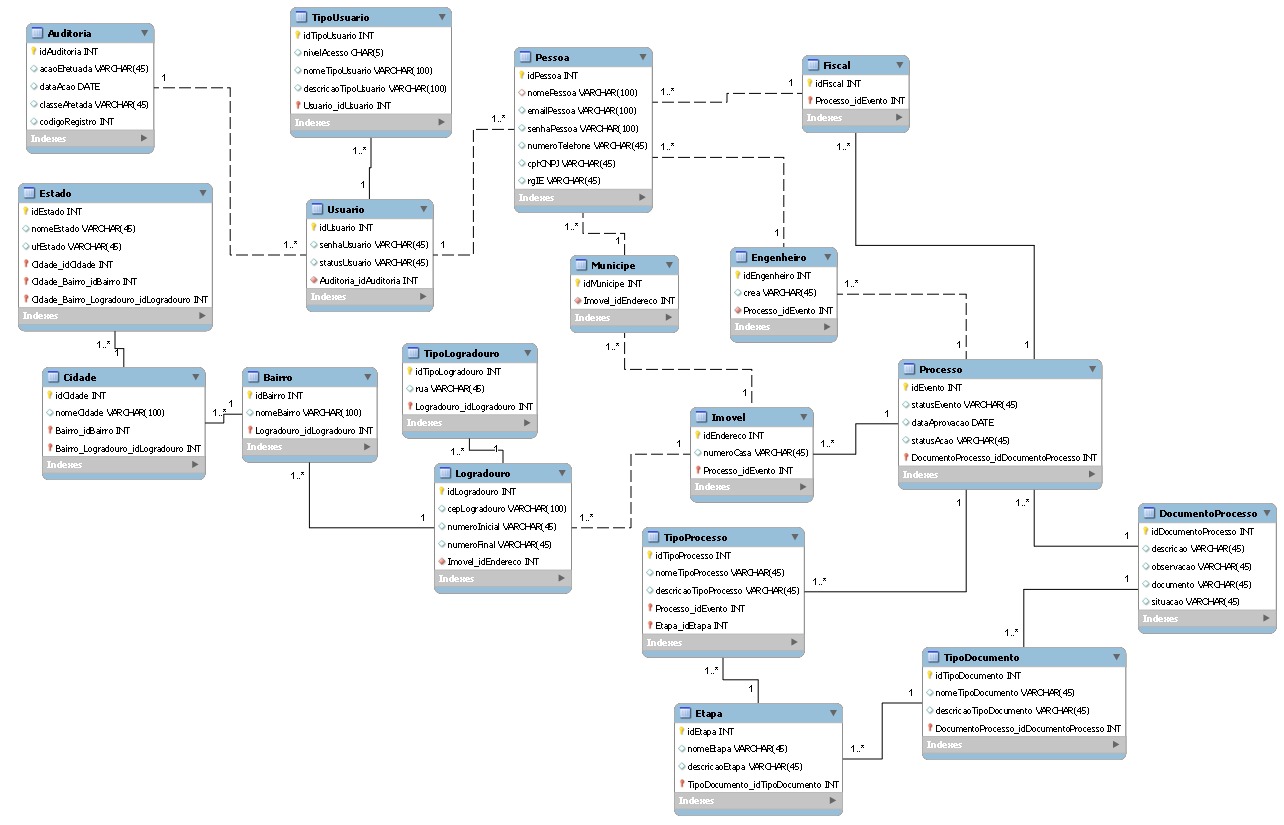
Segundo o site Oracle “SQL é uma linguagem de programação usada por quase todos os bancos de dados relacionais para consultar, manipular e definir dados e fornecer controle de acesso. O SQL foi desenvolvido pela primeira vez na IBM nos anos 1970, com a Oracle como principal contribuinte, o que levou à implementação do padrão SQL ANSI; o SQL estimulou muitas extensões de empresas como IBM, Oracle e Microsoft. Embora o SQL ainda seja amplamente usado hoje em dia, novas linguagens de programação estão começando a aparecer.”

O Modelo Entidade-Relacionamento (MER) é uma ferramenta de modelagem de dados que representa os objetos do mundo real (entidade).

As entidades são representações de objetos do mundo real, como pessoas, produtos ou lugares. Os atributos são as características de uma entidade, como nome, idade ou endereço. Os relacionamentos são as conexões entre diferentes entidades, como a relação entre um cliente e um pedido.

Segundo o site Alura o MER é uma ferramenta essencial para o projeto e implementação de bancos de dados. Ele ajuda a capturar relações complexas entre os dados e a estruturar os dados de maneira compreensível.

Figura 9 – Mapeamento do Objeto Relacional



## Script das tabelas

Segundo o livro “Banco de dados: Projeto e Implementação”, os scripts são as instruções ou comandos escritos em uma linguagem de consulta estruturada (SQL). Elas são utilizados para definir a estrutura de um banco de dados relacional.

**Quadro 1 – Script SQL – Tabela Estado**

|  |
| --- |
| CREATE TABLE estado (  idestado integer GENERATED BY DEFAULT AS IDENTITY,  nomeestado character varying(50) NOT NULL,  ufestado character varying(2) NOT NULL,  CONSTRAINT "PK\_estado" PRIMARY KEY (idestado)  ); |

Fonte: Elaborado pelos autores.

**Quadro 2 – Script SQL – Tabela Municipe**

|  |
| --- |
| CREATE TABLE municipe (  idmunicipe integer GENERATED BY DEFAULT AS IDENTITY,  nomepessoa character varying(70) NOT NULL,  emailpessoa text NOT NULL,  telefonepessoa character varying(19) NOT NULL,  cpfcnpjpessoa character varying(18) NOT NULL,  rgiepessoa character varying(15) NOT NULL,  CONSTRAINT "PK\_municipe" PRIMARY KEY (idmunicipe),  CONSTRAINT "FK\_municipe\_pessoa\_IdPessoa" FOREIGN KEY ("IdPessoa") REFERENCES pessoa (idpessoa) ON DELETE CASCADE  ); |

Fonte: Elaborado pelos autores.

**Quadro 3 – Script SQL – Tabela TipoUsuario**

|  |
| --- |
| CREATE TABLE tipousuario (  idtipousuario integer GENERATED BY DEFAULT AS IDENTITY,  nivelacesso character varying(1) NOT NULL,  nometipousuario character varying(20) NOT NULL,  descricaotipousuario character varying(300) NOT NULL,  CONSTRAINT "PK\_tipousuario" PRIMARY KEY (idtipousuario)  ); |

Fonte: Elaborado pelos autores.

**Quadro 4 – Script SQL – Tabela Cidade**

|  |
| --- |
| CREATE TABLE cidade (  idcidade integer GENERATED BY DEFAULT AS IDENTITY,  nomecidade character varying(100) NOT NULL,  "IdEstado" integer NOT NULL,  CONSTRAINT "PK\_cidade" PRIMARY KEY (idcidade),  CONSTRAINT "FK\_cidade\_estado\_IdEstado" FOREIGN KEY ("IdEstado") REFERENCES estado (idestado) ON DELETE CASCADE,  CONSTRAINT "FK\_cidade\_bairro\_IdBairro" FOREIGN KEY ("IdBairro") REFERENCES bairro (idbairro) ON DELETE CASCADE  ); |

Fonte: Elaborado pelos autores.

**Quadro 5 – Script SQL – Tabela Usuario**

|  |
| --- |
| CREATE TABLE usuario (  idusuario integer GENERATED BY DEFAULT AS IDENTITY,  senhausuario character varying(50) NOT NULL,  cargousuario character varying(50) NOT NULL,  statususuario boolean NOT NULL,  "IdTipoUsuario" integer NOT NULL,  nomepessoa character varying(70) NOT NULL,  emailpessoa text NOT NULL,  telefonepessoa character varying(19) NOT NULL,  cpfcnpjpessoa character varying(18) NOT NULL,  rgiepessoa character varying(15) NOT NULL,  CONSTRAINT "PK\_usuario" PRIMARY KEY (idusuario),  CONSTRAINT "FK\_usuario\_tipousuario\_IdTipoUsuario" FOREIGN KEY ("IdTipoUsuario") REFERENCES tipousuario (idtipousuario) ON DELETE CASCADE,  CONSTRAINT "FK\_usuario\_pessoa\_IdPessoa" FOREIGN KEY ("IdPessoa") REFERENCES pessoa (idpessoa) ON DELETE CASCADE  ); |

Fonte: Elaborado pelos autores.

**Quadro 6 – Script SQL – Tabela Auditoria**

|  |
| --- |
| CREATE TABLE auditoria (  idauditoria integer GENERATED BY DEFAULT AS IDENTITY,  acaoefetuada character varying(50) NOT NULL,  dataacao date NOT NULL,  classeafetada character varying(50) NOT NULL,  "IdUsuario" integer NOT NULL,  idregistro integer NOT NULL,  CONSTRAINT "FK\_auditoria\_usuario\_IdUsuario" FOREIGN KEY ("IdUsuario") REFERENCES usuario (idusuario) ON DELETE CASCADE  ); |

Fonte: Elaborado pelos autores.

**Quadro 7 – Script SQL – Tabela Engenheiro**

|  |
| --- |
| CREATE TABLE engenheiro (  idengenheiro integer GENERATED BY DEFAULT AS IDENTITY,  crea character varying(50) NOT NULL,  CONSTRAINT "PK\_engenheiro" PRIMARY KEY (idengenheiro),  CONSTRAINT "FK\_engenheiro\_pessoa\_IdPessoa" FOREIGN KEY ("IdPessoa") REFERENCES pessoa (idpessoa) ON DELETE CASCADE  ); |

Fonte: Elaborado pelos autores.

**Quadro 8 – Script SQL – Tabela Fiscal**

|  |
| --- |
| CREATE TABLE fiscal (  idfiscal integer GENERATED BY DEFAULT AS IDENTITY,  CONSTRAINT "PK\_fiscal" PRIMARY KEY (idfiscal),  CONSTRAINT "FK\_fiscal\_pessoa\_IdPessoa" FOREIGN KEY ("IdPessoa") REFERENCES pessoa (idpessoa) ON DELETE CASCADE  ); |

Fonte: Elaborado pelos autores.

**Quadro 9 – Script SQL – Tabela Processo**

|  |
| --- |
| CREATE TABLE processo (  idprocesso integer GENERATED BY DEFAULT AS IDENTITY,  statusevento character varying(50) NOT NULL,  dataaprovacao date NOT NULL,  situacao character varying(50) NOT NULL,  CONSTRAINT "PK\_processo" PRIMARY KEY (idprocesso),  CONSTRAINT "FK\_processo\_engenheiro\_IdEngenheiro" FOREIGN KEY ("IdEngenheiro") REFERENCES engenheiro (idengenheiro) ON DELETE CASCADE,  CONSTRAINT "FK\_processo\_fiscal\_IdFiscal" FOREIGN KEY ("IdFiscal") REFERENCES fiscal (idfiscal) ON DELETE CASCADE,  CONSTRAINT "FK\_processo\_imovel\_IdImovel" FOREIGN KEY ("IdImovel") REFERENCES imovel (idimovel) ON DELETE CASCADE,  CONSTRAINT "FK\_processo\_tipoprocesso\_IdTipoProcesso" FOREIGN KEY ("IdTipoProcesso") REFERENCES tipoprocesso (idtipoprocesso) ON DELETE CASCADE,  CONSTRAINT "FK\_processo\_documentoprocesso\_IdDocumentoProcesso" FOREIGN KEY ("IdDocumentoProcesso") REFERENCES documentoprocesso (iddocumentoprocesso) ON DELETE CASCADE  ); |

Fonte: Elaborado pelos autores.

**Quadro 10 – Script SQL – Tabela Imóvel**

|  |
| --- |
| CREATE TABLE imovel (  idimovel integer GENERATED BY DEFAULT AS IDENTITY,  idendereco integer NOT NULL,  numerocasa character varying(50) NOT NULL,  CONSTRAINT "PK\_imovel" PRIMARY KEY (idimovel),  CONSTRAINT "FK\_imovel\_logradouro\_IdLogradouro" FOREIGN KEY ("IdLogradouro") REFERENCES logradouro (idlogradouro) ON DELETE CASCADE,  CONSTRAINT "FK\_imovel\_municipe\_IdMunicipe" FOREIGN KEY ("IdMunicipe") REFERENCES municipe (idmunicipe) ON DELETE CASCADE,  CONSTRAINT "FK\_imovel\_logradouro\_IdLogradouro" FOREIGN KEY ("IdMunicipe") REFERENCES logradouro (idlogradouro) ON DELETE CASCADE,  CONSTRAINT "FK\_imovel\_processo\_IdProcesso" FOREIGN KEY ("IdProcesso") REFERENCES processo (idprocesso) ON DELETE CASCADE  ); |

Fonte: Elaborado pelos autores.

**Quadro 11 – Script SQL – Tabela Logradouro**

|  |
| --- |
| CREATE TABLE logradouro (  idlogradouro integer GENERATED BY DEFAULT AS IDENTITY,  ceplogradouro character varying(50) NOT NULL,  numeroinicial character varying(50) NOT NULL,  numerofinal character varying(50) NOT NULL,  CONSTRAINT "PK\_logradouro" PRIMARY KEY (idlogradouro),  CONSTRAINT "FK\_logradouro\_tipologradouro\_IdTipoLogradouro" FOREIGN KEY ("IdTipoLogradouro") REFERENCES tipologradouro (idtipologradouro) ON DELETE CASCADE,  CONSTRAINT "FK\_logradouro\_bairro\_IdBairro" FOREIGN KEY ("IdBairro") REFERENCES bairro (idbairro) ON DELETE CASCADE  ); |

Fonte: Elaborado pelos autores.

**Quadro 12 – Script SQL – Tabela TipoLogradouro**

|  |
| --- |
| CREATE TABLE tipologradouro (  idtipologradouro integer GENERATED BY DEFAULT AS IDENTITY,  rua character varying(50) NOT NULL,  CONSTRAINT "PK\_tipologradouro" PRIMARY KEY (idtipologradouro),  CONSTRAINT "FK\_tipologradouro\_logradouro\_IdLogradouro" FOREIGN KEY ("IdLogradouro") REFERENCES logradouro (idlogradouro) ON DELETE CASCADE  ); |

Fonte: Elaborado pelos autores.

**Quadro 13 – Script SQL – Tabela Bairro**

|  |
| --- |
| CREATE TABLE bairro (  idbairro integer GENERATED BY DEFAULT AS IDENTITY,  nomebairro character varying(50) NOT NULL,  CONSTRAINT "PK\_bairro" PRIMARY KEY (idbairro),  CONSTRAINT "FK\_bairro\_logradouro\_IdLogradouro" FOREIGN KEY ("IdLogradouro") REFERENCES logradouro (idlogradouro) ON DELETE CASCADE,  CONSTRAINT "FK\_bairro\_cidade\_IdCidade" FOREIGN KEY ("IdCidade") REFERENCES cidade (idcidade) ON DELETE CASCADE  ); |

Fonte: Elaborado pelos autores.

**Quadro 14 – Script SQL – Tabela TipoProcesso**

|  |
| --- |
| CREATE TABLE tipoprocesso (  idtipoprocesso integer GENERATED BY DEFAULT AS IDENTITY,  nometipoprocesso character varying(50) NOT NULL,  descricaotipoprocesso character varying(50) NOT NULL,  CONSTRAINT "PK\_tipoprocesso" PRIMARY KEY (idtipoprocesso),  CONSTRAINT "FK\_tipoprocesso\_etapa\_IdEtapa" FOREIGN KEY ("IdEtapa") REFERENCES etapa (idetapa) ON DELETE CASCADE  ); |

Fonte: Elaborado pelos autores.

**Quadro 15 – Script SQL – Tabela Etapa**

|  |
| --- |
| CREATE TABLE etapa (  idetapa integer GENERATED BY DEFAULT AS IDENTITY,  nomeetapa character varying(50) NOT NULL,  descricaoetapa character varying(50) NOT NULL,  CONSTRAINT "PK\_etapa" PRIMARY KEY (idetapa),  CONSTRAINT "FK\_etapa\_tipoprocesso\_IdTipoProcesso" FOREIGN KEY ("IdTipoProcesso") REFERENCES tipoprocesso (idtipoprocesso) ON DELETE CASCADE,  CONSTRAINT "FK\_etapa\_tipodocumento\_IdTipoDocumento" FOREIGN KEY ("IdTipoDocumento") REFERENCES tipodocumento (idtipodocumento) ON DELETE CASCADE  ); |

Fonte: Elaborado pelos autores.

**Quadro 16 – Script SQL – Tabela TipoDocumento**

|  |
| --- |
| CREATE TABLE tipodocumento (  idtipodocumento integer GENERATED BY DEFAULT AS IDENTITY,  nometipodocumento character varying(50) NOT NULL,  descricaotipodocumento character varying(50) NOT NULL,  CONSTRAINT "PK\_tipodocumento" PRIMARY KEY (idtipodocumento),  CONSTRAINT "FK\_tipodocumento\_etapa\_IdEtapa" FOREIGN KEY ("IdEtapa") REFERENCES etapa (idetapa) ON DELETE CASCADE  ); |

Fonte: Elaborado pelos autores.

**Quadro 17 – Script SQL – Tabela DocumentoProcesso**

|  |
| --- |
| CREATE TABLE tipodocumento (  iddocumentoprocesso integer GENERATED BY DEFAULT AS IDENTITY,  descricao character varying(50) NOT NULL,  observacao character varying(50) NOT NULL,  documento character varying(50) NOT NULL,  situacao character varying(50) NOT NULL,  CONSTRAINT "PK\_documentoprocesso" PRIMARY KEY (iddocumentoprocesso),  CONSTRAINT "FK\_documentoprocesso\_tipodocumento\_IdTipoDocumento" FOREIGN KEY ("IdTipoDocumento") REFERENCES tipodocumento (idtipodocumento) ON DELETE CASCADE,  CONSTRAINT "FK\_documentoprocesso\_processo\_IdProcesso" FOREIGN KEY ("IdProcesso") REFERENCES processo (idprocesso) ON DELETE CASCADE,  CONSTRAINT "FK\_documentoprocesso\_fiscal\_IdFiscal" FOREIGN KEY ("IdFiscal") REFERENCES fiscal (idfiscal) ON DELETE CASCADE  ); |

Fonte: Elaborado pelos autores.

**Quadro 18 – Script SQL – Tabela Pessoa**

|  |
| --- |
| CREATE TABLE pessoa (  idpessoa integer GENERATED BY DEFAULT AS IDENTITY,  nomepessoa character varying(50) NOT NULL,  emailpessoa character varying(50) NOT NULL,  senhapessoa character varying(50) NOT NULL,  numerotelefone character varying(50) NOT NULL,  cpfcnpj character varying(50) NOT NULL,  rgie character varying(50) NOT NULL,  CONSTRAINT "PK\_pessoa" PRIMARY KEY (idpessoa)  ); |

Fonte: Elaborado pelos autores.

# ARQUITETURA DE SOFTWARE

A arquitetura de software refere-se à estrutura fundamental de um sistema de software e à organização de seus componentes ou módulos, bem como às relações entre elas. Essa estrutura fornece uma visão de alto nível do sistema, ajudando a guiar o design e a implementação para atender aos requisitos funcionais e não funcionais do software.

Segundo GONÇALVES (2021), existem diversos tipos de padrões arquiteturais, que são formulados como soluções abrangentes e reutilizáveis de componentes de aplicação, com o objetivo de resolver problemas comuns que surgem dentro de um contexto específico. Portanto, os padrões representam uma abordagem consistente e reutilizável para desafios que ocorrem repetidamente.

A introdução de uma arquitetura de software eficiente no sistema de gerenciamento de documentos de obra da Prefeitura de Jales traz consigo uma série de benefícios significativos e adaptados às demandas específicas desse cenário. Esses benefícios abrangem desde a organização eficaz dos documentos até a melhoria na comunicação interna, garantindo uma gestão integrada e eficiente das operações municipais.

## 6.1 Arquitetura de desenvolvimento

IDE utilizada (Back-End): O Visual Studio, desenvolvido pela Microsoft, é um ambiente de desenvolvimento integrado (IDE), que oferece suporte a diversas linguagens de programação, como C#, C++, Visual Basic e Python. Com um editor de código avançado, ferramentas de depuração integradas e designers visuais para interfaces gráficas, o Visual Studio simplifica o processo de desenvolvimento de software. Ele possui integração com sistemas de controle de versão, facilita a compilação e implantação de aplicativos e é altamente extensível, permitindo que desenvolvedores personalizem o ambiente conforme suas necessidades.

Linguagem utilizada (Back-End): O C# é uma linguagem de programação moderna e orientada a objetos desenvolvida pela Microsoft como parte da plataforma .NET. Lançada no início dos anos 2000, destaca-se pela integração com a plataforma .NET, oferecendo interoperabilidade entre diferentes linguagens e sistemas operacionais. Caracterizada por uma sintaxe limpa e expressiva.

IDE utilizada (Front-End): O Visual Studio Code é um IDE leve, mas poderoso. Ele vem com suporte para JavaScript, TypeScript e Node.js e tem um rico ecossistema de extensões para outras linguagens e tempos de execução (como C++, C#, Java, Python, PHP, Go, .NET).

Linguagem utiliza (Front-End): O React é uma biblioteca JavaScript de código aberto mantida pelo Facebook (atual Meta), projetada para facilitar o desenvolvimento de interfaces de usuário (UI) interativas e eficientes. Lançado em 2013, o React introduz o conceito de "componentes", permitindo a construção de UIs reutilizáveis e modularizadas.

## 6.1.1 Arquitetura de software utilizada

A estrutura do código no back-end é composta por diversos elementos cruciais que colaboram para uma arquitetura robusta e eficiente. Os DTOs, ou Data Transfer Objects, são peças essenciais nesse contexto, desempenhando um papel vital na transferência eficiente de dados entre diferentes partes da aplicação. Essas estruturas encapsulam informações específicas, frequentemente representando entidades de negócios, otimizando a comunicação entre camadas da aplicação. Além dos DTOs, outros componentes fundamentais incluem services, controllers e repositories. Os services representam a camada de lógica de negócios, encapsulando operações e regras específicas do domínio, facilitando a modularização e organização do código. Já os controllers atuam como intermediários entre as requisições do usuário e os services, gerenciando o fluxo de dados e interações. Esses componentes recebem as requisições, acionam operações nos services correspondentes e respondem ao cliente com os resultados apropriados. Quanto aos repositories, são responsáveis pela interação com o armazenamento de dados, isolando as operações de persistência e permitindo que o restante do sistema interaja com os dados sem se preocupar com os detalhes específicos do armazenamento. Também foi utiliza APIs, como o Swagger, desempenhando um papel crucial na comunicação eficiente entre diferentes partes do sistema ou sistemas distintos. Essas APIs funcionam como pontes padronizadas para a exposição de funcionalidades e dados, permitindo integrações eficientes de serviços e facilitando a construção de aplicações modulares. Essa combinação de DTOs, services, controllers, repositories e APIs forma uma arquitetura coesa que promove modularidade, clareza e escalabilidade no desenvolvimento de software.

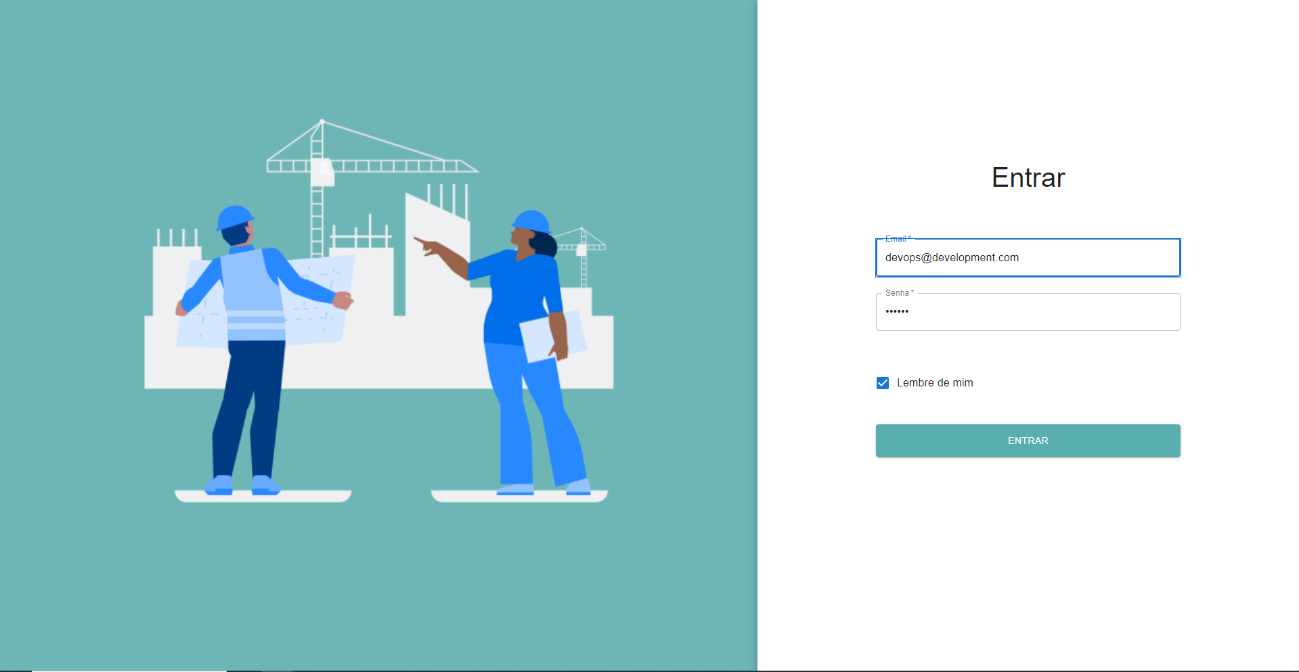
## 6.2 Telas do sistema

As telas em um sistema desempenham um papel crucial na experiência do usuário e na eficácia geral na plataforma. Elas servem como interface principal entre os usuários e as funcionalidades oferecidas pelo sistema.

A usabilidade é a primeira consideração, tendo em vista que telas bem projetadas e intuitivas facilitam a navegação e utilização do sistema, contribuindo para uma interação mais eficiente e agradável. A Experiência do Usuário (UX) também é impactada diretamente pela qualidade das telas, onde uma interface bem elaborada promove uma experiência positiva, enquanto uma interface confusa pode gerar frustração.

Além disso, uma interface intuitiva facilita o aprendizado do sistema, permitindo que novos usuários se familiarizem rapidamente com as funcionalidades e navegação. A consistência visual entre as telas é essencial para uma experiência coesa, enquanto a acessibilidade é uma consideração crucial para garantir que o sistema seja utilizável por uma variedade de usuários.

**Figura 14 - Tela de Login**



Fonte: Elaborado pelos autores.

A interface de login representa a principal porta de entrada para acessar o sistema, sendo concebida com opções e campos intuitivos para proporcionar ao usuário uma experiência de acesso rápida e descomplicada. A obtenção das credenciais de login é atribuída ao administrador do sistema. Ao preencher os campos e pressionar o botão de entrada, o sistema realiza uma validação junto ao banco de dados para verificar a autenticidade das informações fornecidas. Se as credenciais forem válidas, o usuário é redirecionado à tela inicial. Em contrapartida, caso não haja correspondência nos dados do banco, o sistema emite uma mensagem informando sobre a incorreção dos dados.

**Figura 15 - Tela de Início**

Interface gráfica do usuário, Site

Descrição gerada automaticamente

Fonte: Elaborado pelos autores.

Na página inicial do sistema, a barra de navegação superior exibe a identidade visual da Prefeitura de Jales, incluindo a logo, o nome do usuário, notificações e detalhes do perfil. À esquerda, uma segunda barra de navegação focaliza a interação do usuário com as diversas opções oferecidas pelo software.

No centro da página, são apresentados o status e andamento das atividades e arquivos cadastrados. Essa seção proporciona uma visão abrangente, destacando novos cadastros, processos em andamento, pendências, itens em atraso e aqueles com prazo para o dia atual. Adicionalmente, é possível visualizar, em formato de tabela, os últimos andamentos e arquivos registrados.

A função principal da tela inicial é oferecer ao usuário uma visão detalhada e informativa. As informações são apresentadas de maneira clara e objetiva, possibilitando ao usuário uma verificação rápida e precisa de documentos quando necessário, contribuindo para uma experiência eficiente e assertiva.

**Figura 16 - Tela Principal de Cadastros**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente com confiança média

Fonte: Elaborado pelos autores.

Na página principal de cadastros, apresentam-se cartões que oferecem ao usuário a escolha da função de cadastro e visualização de dados desejada. Este design foi cuidadosamente planejado para proporcionar uma experiência intuitiva, permitindo ao usuário acessar as informações de maneira ágil e eficiente.

O propósito central dessa tela é distribuir as diversas funcionalidades de cadastro de forma organizada, garantindo que o usuário possa acessar as diferentes opções disponíveis no sistema de maneira fácil e sem dificuldades.

**Figura 17 - Tela de Cadastro de Estado**

Interface gráfica do usuário

Descrição gerada automaticamente

Fonte: Elaborado pelos autores.

A interface destinada ao cadastro de estados foi concebida com o objetivo de proporcionar facilidade de compreensão e simplicidade na execução das funções. Nela, encontramos uma barra de pesquisa que possibilita a busca por informações específicas, como o nome do estado. No canto superior direito, encontra-se um botão para a adição de um novo estado.

O centro da tela apresenta uma tabela contendo as informações referentes aos estados cadastrados, permitindo a realização de ações como edição e remoção. O propósito dessa tela reside em oferecer uma ferramenta intuitiva e eficaz para o gerenciamento dos estados, facilitando o processo de cadastro e manipulação desses dados de forma ágil e direta.

**Figura 18 - Tela de Cadastro de Cidade**

Interface gráfica do usuário

Descrição gerada automaticamente

Fonte: Elaborado pelos autores.

A tela de cadastro de cidade segue uma abordagem semelhante à tela de cadastro de estado, visando proporcionar uma experiência consistente e intuitiva para o usuário.

Nessa interface, é possível encontrar uma disposição amigável e compreensível dos campos de preenchimento, com uma barra de pesquisa na parte superior que permite a busca por informações específicas, como o nome da cidade. Na lateral direita superior, o botão para adicionar uma nova cidade facilita a inclusão de dados.

O elemento central da tela consiste em uma tabela que organiza as informações relativas às cidades cadastradas. Dentro dessa tabela, o usuário pode executar ações de edição e remoção conforme necessário.

O propósito essencial da tela de cadastro de cidade é oferecer uma ferramenta eficaz para a administração de informações relacionadas às cidades. Sua estrutura intuitiva e funcionalidades simplificadas buscam facilitar o processo de cadastramento e gerenciamento de dados, contribuindo para uma experiência fluida e eficiente.

**Figura 19 - Tela de Cadastro de Usuário**

Interface gráfica do usuário

Descrição gerada automaticamente

Fonte: Elaborado pelos autores.

Com certeza. A tela de cadastro de usuário segue uma estrutura semelhante às telas anteriores, porém, incorpora um novo elemento significativo: o status do usuário. Essa adição proporciona uma visão adicional sobre o estado ou permissões associadas a cada usuário no sistema.

Assim como nas telas anteriores, a interface é projetada com clareza e simplicidade, apresentando campos de preenchimento e uma barra de pesquisa na parte superior para facilitar a localização de informações específicas. A presença de um botão na lateral direita superior permite a adição de novos usuários, tornando o processo de cadastro eficiente.

O ponto central da tela consiste em uma tabela que exibe as informações dos usuários cadastrados, agora incluindo o status correspondente a cada um. Além das ações de edição e remoção, o usuário pode gerenciar o status de outros usuários conforme necessário.

O propósito fundamental da tela de cadastro de usuário é oferecer uma ferramenta robusta para o gerenciamento de informações de usuários no sistema, proporcionando não apenas os dados tradicionais de cadastro, mas também um controle claro sobre o status associado a cada usuário, contribuindo para uma administração eficaz e personalizada de permissões e acessos.

**Figura 20 - Tela de Cadastro de Tipo de Usuário**

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Fonte: Elaborado pelos autores.

A tela de cadastro de tipo de usuário apresenta uma estrutura semelhante às telas anteriores, no entanto, diferencia-se ao já incluir, por padrão, alguns tipos de usuários predefinidos, cada um associado a configurações específicas de permissões dentro do sistema.

Assim como nas interfaces anteriores, essa tela é projetada com campos de preenchimento e uma barra de pesquisa, oferecendo praticidade na localização de informações específicas. A presença de um botão na lateral direita superior permite adicionar novos tipos de usuários, personalizando ainda mais as permissões dentro do sistema.

O componente central da tela é uma tabela que exibe os tipos de usuários cadastrados, incluindo aqueles predefinidos. Além das ações de edição e remoção, o usuário pode gerenciar as permissões associadas a cada tipo de usuário, adaptando-as conforme necessário.

O propósito essencial da tela de cadastro de tipo de usuário é fornecer uma ferramenta eficiente para a gestão das permissões dentro do sistema. Ao oferecer tipos de usuários predefinidos, ela simplifica o processo de atribuição de permissões, contribuindo para uma administração mais estruturada e segura das funcionalidades disponíveis para diferentes usuários no ambiente do sistema.

# CONCLUSÃO

Fazer uma conclusão se é viável o desenvolvimento do software.

# REFERÊNCIAS

ASFHAL, C. R. **Gestão de segurança do trabalho e de saúde ocupacional**. São Paulo: Reichmann & Autores, 2005.

BEAIRD, J. Princípios do Web Design Maravilhoso. Rio de Janeiro: Altabooks, 2008.

CHIAVENATO, I. **Gestão de pessoas**: o novo papel dos recursos humanos nas organizações. 3. ed. Rio de Janeiro: Elsevier, 2008.

COSTA, H. J. Acidentes do trabalho: teremos nova lei acidentária?. **Jus Navigandi**, Teresina, v. 9, n. 664, 1 maio 2005. Disponível em: <http://jus2.uol.com.br/doutrina/texto.asp?id=6662>. Acesso em: 1 jul. 2010.

DELIBERATO, P. C. P. **Fisioterapia preventiva**: fundamentos e aplicações. São Paulo: Manole, 2002.

DIAS, E. C.; MELO, E. M. de. Políticas públicas em saúde e segurança no trabalho. In: MENDES, R. (Org.). **Patologia do trabalho**. 2. ed. atual. e ampl. São Paulo: Atheneu, 2007. v. 2. p. 1683-1720.

WATRALL, E; SIARTO, J. Use A Cabeça! **Web Design.** Alta Books, 2009.   
  
OLIVEIRA, D. **MER e DER:** Definições, Banco de Dados e Exemplos. 18 setembro 2023. Disponível:<https://www.alura.com.br/artigos/mer-e-der-funcoes>. Acesso em: 21 nov. 2023.

*GUEDES, Gilleanes T. A.* ***UML Uma Abordagem prática****, 3 ed. São Paulo: Novatec, 2008.*

*GUEDES, Gilleanes T. A.* ***UML 2 Uma Abordagem prática****, São Paulo: Novatec, 2009.*

ORACLE. **O que é um Banco de Dados?**. 2023. Disponível: <https://www.oracle.com/br/database/what-is-database/>. Acesso em: 21 nov. 2023.

MICROSOFT. **O que é o Visual Studio?**. 28 outubro 2023.

Disponível: <https://learn.microsoft.com/pt-br/visualstudio/get-started/visual-studio-ide?view=vs-2022>. Acesso em: 03 dez. 2023.

MICROSOFT. **Um tour pela linguagem C#**. 15 fevereiro 2023. Disponível: <https://learn.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/>. Acesso em: 03 dez. 2023.

NEVES, V. **React:** o que é, como funciona e um Guia dessa popular ferramenta JS. 17 janeiro 2023. Disponível: < https://www.alura.com.br/artigos/react-js>. Acesso em: 03 dez. 2023.

MICROSOFT. **Visual Studio Code.** 2023. Disponível: **<**https://visualstudio.microsoft.com/pt-br/#vscode-section**>**. Acesso em: 03 dez. 2023.

MICROSOFT**. Criar DTOs (objetos de transferência de dados).** 17 julho 2023. Disponível: <https://learn.microsoft.com/pt-br/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5>. Acesso em: 03 dez. 2023.