# Apresentação Geral do Projeto.

O projeto consiste no desenvolvimento de um sistema para gerir o meu próprio negócio. A ideia surgiu a partir de sugestões apresentadas em aula pelo professor. Eu possuo uma empresa de assessoria, montagem e manutenção de computadores.

O sistema de gestão me permitirá realizar a gestão dos meus clientes e seus respectivos equipamentos, assim como, dos meus fornecedores, estoque, ordens de serviço, emissão de relatórios, e tudo isso de forma ágil, que possa ser acessada de qualquer local, seja dentro da empresa ou em um serviço externo. Esse sistema permitirá que os clientes verifiquem suas Ordens de Serviço e assistam aos unboxings das peças usadas.

# Objetivos

* 1. **Objetivo Principal**

O objetivo principal do projeto integrador é desenvolver um software para gerenciar de forma eficiente minha empresa de Assessoria, Montagem e Manutenção de computadores.

* 1. **Objetivos Específicos**
* Automação de Processos Fundamentais como:
* Acompanhamento do status de cada projeto;
* Aumento da Eficiência:
* Minimizar erros operacionais e retrabalho através de uma interface intuitiva e integrada que permita acesso rápido às informações necessárias para cada fase do projeto;
* Emissão de relatórios.

# **Metodologia**

* 1. **Preparação Inicial**

Definir os objetivos, escopo, cronograma e recursos necessários para o projeto.

Atividades:

* Primeira reunião com as partes interessadas para buscar entender as necessidades e expectativas.
* Definição do escopo do projeto.
* Elaboração de um cronograma macro do projeto.
* Identificação dos recursos humanos e materiais necessários.
  1. **Requisitos**

Coletar e documentar todos os requisitos funcionais e não funcionais do sistema.

Atividades:

* Realização de workshops com as partes interessadas para coletar os requisitos do projeto.
* Documentação dos requisitos funcionais e não funcionais.
* Validação dos requisitos com as partes interessadas.
  1. **Design do Sistema**

Criar a arquitetura do sistema e o modelo detalhado das funcionalidades.

Atividades:

* Definição da arquitetura do sistema (frontend, backend, banco de dados, etc.).
* Criação de wireframes e protótipos de baixa fidelidade.
* Design detalhado de cada funcionalidade.
* Revisão e aprovação do design com as partes interessadas.
  1. **Desenvolvimento**

Implementar o sistema de acordo com os requisitos e design definidos.

Atividades:

* Desenvolvimento do frontend utilizando tecnologias apropriadas.
* Desenvolvimento do backend utilizando uma stack adequada.
* Implementação do banco de dados.
* Integração das diferentes partes do sistema.
* Testes unitários e de integração contínuos.

Iterações Ágeis:

* O desenvolvimento será feito em sprints (ciclos de 2 a 4 semanas).
* Revisões de sprint com as partes interessadas para demonstração e feedback.
  1. **Testes**

Garantir que o sistema funcione conforme esperado e esteja livre de bugs.

Atividades:

* Testes de sistema e testes de aceitação do usuário.
* Testes de desempenho e carga.
* Correção de bugs identificados durante os testes.
* Revisão final e aprovação para lançamento.
  1. **Implementação**

Lançar o sistema em um ambiente de produção.

Atividades:

* Planejamento da implantação (data, recursos necessários, etc.).
* Treinamento dos usuários finais.
* Migração de dados, se necessário.
* Implantação do sistema no ambiente de produção.
* Monitoramento inicial para identificar e corrigir quaisquer problemas pós-implantação.
  1. **Manutenção e Suporte**

Assegurar que o sistema continue a funcionar conforme esperado e realizar melhorias contínuas.

Atividades:

* Suporte técnico para resolver problemas dos usuários.
* Atualizações regulares para corrigir bugs e melhorar funcionalidades.
* Coleta de feedback dos usuários para melhorias futuras.
* Planejamento e desenvolvimento de novos recursos conforme necessário.

Ferramentas e Técnicas Utilizadas:

* Gestão de Projeto: Trello.
* Controle de Versão: GitHub.
* Desenvolvimento Frontend: “CSS, php, html, Javascript, React”.
* Desenvolvimento Backend: Java.
* Banco de Dados: MariaDB.
* Testes: Sikuli.

# **Arquitetura do Sistema (Modelagem)**

* 1. **Regras de Negócio**

Cliente:

* Assistir aos Unboxings;
* Criar e editar contas;

Técnico de Informática:

* Visualizar e gerenciar Ordens de Serviço;
* Verificar status de cada ordem de serviço desde a criação até a conclusão;
* Verificar histórico de serviços prestados para cada cliente;

Gerente/Dono:

* Cadastrar e atualizar projetos;
* Gerar relatórios detalhados sobre as atividades da empresa.
* Verificar relatórios dos projetos e de ordens de serviço.
  1. **Requisitos**
     1. **Requisitos Funcionais**
* Verificação do Status do Pedido
  + Funcionalidade para os clientes verificarem o status de seus pedidos.
  + Interface amigável onde os clientes podem acompanhar o progresso dos serviços solicitados.
* Emissão de ordem de serviço:
  + Criação e emissão de ordens de serviço detalhadas.
  + Rastreamento do status de cada ordem de serviço desde a criação até a conclusão.
* Acompanhamento de status de cada projeto:
  + Dashboard para monitorar o status dos projetos em andamento.
* Análise de Desempenho:
  + Relatórios de desempenho dos técnicos e do uso de recursos.
* Emissão de Relatórios:
  + Geração de relatórios detalhados sobre as atividades da empresa.
  + Relatórios de Ordens de Serviço, de Técnicos e Projetos.
  + Exportação de relatórios em formatos comuns (e.g., PDF, Excel).
    1. **Requisitos Não Funcionais**
* Desempenho:
  + O sistema deve ser capaz de processar pedidos e atualizações em tempo real.
  + Tempos de resposta rápidos para operações críticas (e.g., criação de ordens de serviço).
* Escalabilidade:
  + Capacidade de suportar um número crescente de clientes, ordens de serviço e dados de estoque sem perda de desempenho.
  + Possibilidade de adicionar novos módulos ou funcionalidades conforme a empresa cresce.
* Segurança:
  + Proteção contra acesso não autorizado aos dados dos clientes e da empresa.
* Usabilidade:
  + Interface intuitiva e fácil de usar para todos os tipos de usuários (técnicos, gerentes, clientes).
* Confiabilidade:
  + O sistema deve ter alta disponibilidade e ser capaz de operar sem interrupções significativas.
  + Testes rigorosos para garantir que todas as funcionalidades operem conforme o esperado.
* Manutenibilidade:
  + Código bem documentado e modular para facilitar futuras manutenções e atualizações.
* Compatibilidade:
  + Compatível com diferentes sistemas operacionais.
  + Integração com outros sistemas de software utilizados pela empresa (e.g., sistemas de contabilidade).
    1. **Modelagem Funcional**
    2. **Diagrama Casos de Uso**

Diagrama

Descrição gerada automaticamente

* + 1. **Detalhamento Casos de Uso**
* Cliente:
  + CU-01 - Verificar Status da Ordem de Serviço: O cliente pode acessar o sistema para verificar o status atual de suas ordens de serviço.
  + CU-02 - Assistir aos Unboxings: O cliente pode assistir a vídeos de unboxing das peças recebidas para seu pedido.
  + CU-03 – Criar/Editar Contas: O cliente consegue criar/editar a conta do mesmo.
* Técnico:
  + CU-04 - Gerenciar Ordens de Serviço: O técnico pode criar, atualizar e fechar ordens de serviço.
  + CU-05 - Acessar Histórico do Cliente: O técnico pode acessar o histórico de serviços prestados ao cliente.
* Administrador:
  + CU-06 - Gerenciar Contas de Usuário: O administrador pode gerenciar contas de usuários do sistema.
  + CU-07 - Emitir Relatórios: O administrador pode gerar relatórios sobre diversas atividades da empresa.
  1. **Diagrama de Classe**

Uma imagem contendo Diagrama

Descrição gerada automaticamente

# Banco

* 1. **Modelo Conceitual**



* 1. **Modelo Lógico**



* 1. **Modelo Físico**

CREATE TABLE Administrador (

IDAdmin INTEGER PRIMARY KEY AUTO\_INCREMENT,

Senha VARCHAR(50),

UsuarioAdmin VARCHAR(50)

);

CREATE TABLE Cliente (

IDUsuario INTEGER PRIMARY KEY AUTO\_INCREMENT,

Nome VARCHAR(50),

Email VARCHAR(100),

Endereco VARCHAR(255),

CPF CHAR(14) NOT NULL,

Telefone CHAR(14),

UsuarioCliente VARCHAR(50),

Senha VARCHAR(50)

);

CREATE TABLE Tecnico (

IDTecnico INTEGER PRIMARY KEY AUTO\_INCREMENT,

Nome VARCHAR(50),

Telefone CHAR(14),

Email VARCHAR(100),

CPF CHAR(14) NOT NULL,

UsuarioTec VARCHAR(50),

Senha VARCHAR(50)

);

CREATE TABLE Projeto\_OrdemdeServico (

IDOs INTEGER PRIMARY KEY AUTO\_INCREMENT,

Condicao VARCHAR(255),

Descricao VARCHAR(255),

LinkUnboxing VARCHAR(255),

DataInicio DATE,

DataFim DATE,

fk\_Administrador\_IDAdmin INTEGER,

fk\_Cliente\_IDUsuario INTEGER,

fk\_Tecnico\_IDTecnico INTEGER

);

ALTER TABLE Projeto\_OrdemdeServico ADD CONSTRAINT FK\_Projeto\_OrdemdeServico\_2

FOREIGN KEY (fk\_Administrador\_IDAdmin)

REFERENCES Administrador (IDAdmin)

ON DELETE RESTRICT;

ALTER TABLE Projeto\_OrdemdeServico ADD CONSTRAINT FK\_Projeto\_OrdemdeServico\_3

FOREIGN KEY (fk\_Cliente\_IDUsuario)

REFERENCES Cliente (IDUsuario)

ON DELETE RESTRICT;

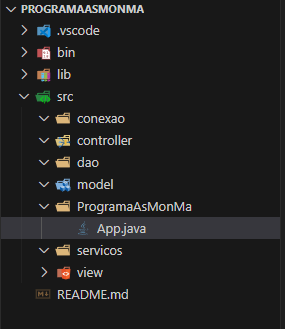
ALTER TABLE Projeto\_OrdemdeServico ADD CONSTRAINT FK\_Projeto\_OrdemdeServico\_4

FOREIGN KEY (fk\_Tecnico\_IDTecnico)

REFERENCES Tecnico (IDTecnico)

ON DELETE RESTRICT;

1. **Parte Desktop**
   1. **Telas – protótipo na sequência, com pequena descrição**



* 1. **Códigos – ( os mais relaevantes para o sistema )**
  2. **Plano de teste**

Contas criadas:

admin - 123456

cliente – 123456

tecnico – 123456

projeto – montagem – Computador Teste – youtube.com/teste – 20241015 – 20241031 – 1 – 1

abrir criar conta – Bruno – [bruno@teste.com](mailto:bruno@teste.com) – 88899900022 – Rua dos Tolos, 0 – 01140028922 – bruno – 123456

abrir conta – cliente – 123456

mostrar editar conta – voltar para pedido e sair –

área funcionário – logar com tecnico – 123456

mostrar criar/editar – criar os nova – modificar e excluir ela

voltar no tecnico e clicar atualizar

sair e entrar no administrador – admin – 123456

mostrar tabelas

criar e deletar tecnico – criar tecnico novo – giorno – [giorno@teste.com](mailto:giorno@teste.com) – 22233344411 – 0519988990 – giorno – 123456 – voltar para administrador e atualizar

abrir pasta local C – criar relatório dos clientes – mostrar relatório