





DOCUMENTAÇÃO TÉCNICA DO PROJETO APLICADO: **SISTEMA WEB DE CONTROLE DE PEÇAS COM JAVASCRIPT, HTML, CSS + MULTI CLOUD**

Projeto Aplicado I

  Equipe do Projeto
Daniel Moreira Lourenço, Rafael Nascimento Araujo Silva, Tiago de Oliveira Souza, Augusto Giequelin Neto

 Professor(a) Responsável
 Iskailer Inaian Rodrigues

 Florianópolis - SC
 2025

SUMÁRIO

1. EQUIPE E PLANEJAMENTO DAS ATIVIDADES	PÁG. 3
1.1. Nome da Equipe	
1.2. Integrantes e Funções	
1.3. Cronograma do Projeto	
2. PROBLEMA ESCOLHIDO E DESCRIÇÃO	PÁG. 5
2.1. Contextualização	
2.2. Descrição do Problema	
2.3. Impacto	
2.4. Justificativa	
3. NECESSIDADES DO CLIENTE/USUÁRIO E VALIDAÇÃO	PÁG. 7
3.1. Perfil do Cliente/Usuário	
3.2. Necessidades Identificadas	
3.3. Método de Validação	
3.4. Conclusões Obtidas	
4. TECNOLOGIAS ESCOLHIDAS E JUSTIFICATIVA	PÁG. 9
4.1. Frontend	
4.2. Backend	
4.3. Banco de Dados	
4.4. Hospedagem/Deploy	
4.5. Outras Ferramentas	
5. DIAGRAMA E DESCRIÇÃO DA SOLUÇÃO PROPOSTA	PÁG. 11
5.1. Descrição Geral da Solução	
5.2. Diagrama de Arquitetura	
5.3. Protótipo da Interface	

1. Equipe e Planejamento das Atividades

✓ O que é esperado:

A equipe é composta por Daniel Lourenço, Rafael Nascimento e Tiago de Oliveira Souza e nossa proposta é desenvolver uma solução para controle de estoque de peças, onde o Daniel irá ficar responsável pelo desenvolvimento juntamente com o Rafael no frontend e backend + serviço de multicloud.

✚ Estrutura esperada:

- **Integrantes e Funções:**



- Daniel Lourenço – Responsável pelo planejamento e desenvolvimento fullstack da aplicação.
- Augusto Giequelin – Responsável pelo desenvolvimento front-end da aplicação.
- Rafael Nascimento – Responsável pela infraestrutura e implementação de serviços multicloud.
- Tiago Souza – Responsável pelo desenvolvimento backend da aplicação.

Integrantes e Funções		
👤 Nome	📌 Função	📌 Principais responsabilidades
Daniel Lourenço	Desenvolvimento	FullStack
Augusto Giequelin	Desenvolvimento	Front-end
Rafael Nascimento	Infraestrutura	Servidores e Cloud
Tiago Souza	Desenvolvimento	Backend

- **Cronograma do Projeto:**

- Semana 1: Planejamento da Arquitetura do Sistema
- Semana 2: Planejamento das Interfaces dos usuários
- Semana 3: Definição das tecnologias e início do desenvolvimento

Cronograma do Projeto

 Responsável	Tt Descrição	 Data Limite	Tt Finalizado
Daniel Lourenço	Planejamento do Sistema	10/03/2025	Sim
Augusto Giequelin	Planejamento das Interfaces dos usuários	21/03/2025	Sim
Tiago de Oliveira Souza, Rafael Nascimento	Definição das Tecnologias	01/05/2025	Sim
Daniel Lourenço, Augusto Giequelin	Início do Desenvolvimento FrontEnd	10/06/2025	Não
Tiago de Oliveira Souza	Início do Desenvolvimento Backend	10/07/2025	Não
Daniel Lourenço	Início do Banco de Dados	10/07/2025	Não
Daniel Lourenço	Containerização da Aplicação	10/08/2025	Não
Rafael Nascimento	Criação da Infraestrutura	10/09/2025	Não
Rafael Nascimento	Deploy da Aplicação	10/10/2025	Não

2. Problema Escolhido e Descrição

O que é esperado:

- A situação que ocorre é que há um controle de estoque de peças sendo realizada manualmente, através de planilhas de excel, e isso não é muito viável caso a demanda por cadastros e retiradas de produtos for grande, nisso resolvemos contornar essa situação com uma aplicação web.

Estrutura esperada:

- O controle de peças de uma empresa é feita manualmente, por conta desse processo, a empresa vem enfrentando dificuldades para realizar os cadastros, informando que é inviável realizar as entradas e saídas das peças, com isso, iremos construir um sistema web ao qual será realizado o cadastros de peças, onde será possível consultar as peças cadastradas e retiradas com horários e dias definidos, será realizado também o controle de restrição para que cada usuário tenha sua visão conforme sua permissão. Além disso será implementada uma solução multicloud, ao qual se um determinado servidor cair, será utilizado um failover(servidor de redundância), para que o sistema web continue funcionando.

3. Necessidades do Cliente/Usuário e Validação

O que é esperado:

Necessidade 1: Rastreabilidade das peças

- **Descrição:** A empresa precisa acompanhar o caminho completo de cada peça: entrada, movimentação e retirada.
- **Validação:** Entrevistas com responsáveis do setor de estoque revelaram a perda de controle nas movimentações e ausência de histórico confiável.

Necessidade 2: Controle de acesso por usuário

- **Descrição:** É necessário que o sistema limite o acesso conforme o perfil do usuário (ex: operador, supervisor, gerente), impedindo edições indevidas.
- **Validação:** Feedback da equipe de TI e dos gestores sobre o uso atual da planilha, onde todos têm acesso irrestrito, gerando insegurança e erros.

Necessidade 3: Cadastro individual de peças (em vez de kits)

- **Descrição:** As peças precisam ser cadastradas individualmente, não apenas como parte de kits, para um controle mais preciso do estoque.
- **Validação:** Observações feitas durante visitas ao local e relatos de operadores sobre a dificuldade em identificar itens específicos dentro de kits genéricos.

Necessidade 4: Redução de movimentações físicas desnecessárias


- **Descrição:** O sistema deve ajudar a localizar rapidamente as peças, diminuindo deslocamentos físicos e otimizando o tempo da equipe.
- **Validação:** Segundo o livro "*Administração de Materiais – Um Enfoque Prático*" de **Arnoldo Chiavenato**, a eficiência na gestão de estoques está diretamente ligada à **organização e rastreabilidade dos itens**, sendo que a **agilidade no acesso à informação** impacta diretamente no tempo de resposta da operação e no nível de serviço interno

Necessidade 5: Histórico de ações e fluxo de aprovação seguro

- **Descrição:** Todas as movimentações devem ser registradas com data, hora e responsável. Além disso, é necessário um fluxo de aprovação confiável para saídas e retiradas.

- **Validação:** A implementação de um **histórico de ações e fluxo de aprovação seguro** é essencial para garantir a **rastreabilidade, transparência e integridade das operações internas**. Em ambientes onde múltiplos usuários interagem com dados críticos, como controle de estoque, é indispensável registrar **quem fez o quê, quando e por qual motivo**. De acordo com a **ISO 9001:2015** (sistema de gestão da qualidade), é obrigatório manter **registros de ações que afetam a qualidade e rastreabilidade de processos**. A norma orienta que esses registros estejam organizados e acessíveis para consulta e auditoria.

4. Tecnologias Escolhidas e Justificativa

 Estrutura esperada:

- Frontend: ReactJs – Facilidade de reutilizar componentes e criar interfaces seguindo os princípios do SPA(Single Page Application) onde tudo é renderizado dinamicamente em um único arquivo .html
- Backend: NodeJs – Pela facilidade de enviar e receber requisições e integração com ReactJs, por se tratar de um projeto web foi optado para construção o Ecossistema JavaScript.
- Banco de Dados: PostgreSQL – Um SGBD moderno e robusto, open-source, com suporte avançado a transações e integridade referencial.
- Hospedagem/Deploy: AWS Ec2 – Facilidade de personalização do servidor Ec2, com implementação de um Sistema Operacional personalizado, onde podemos realizar diversas configurações como criação de firewall, implantação de CDN(Content Delivery Network), acesso e tunelamento SSH.
- Outras Ferramentas: Docker – O Docker é uma ferramenta que nos ajuda a empacotar o projeto inteiro em um container, o qual será executado em um processo isolado do Sistema Operacional, tornando a reutilização desse container independente em qualquer outra máquina, servidor ou Sistema Operacional.

5. Diagrama e Descrição da Solução Proposta

- Descrição Geral da Solução:

Atualmente, o controle das peças na empresa é feito de forma manual, e isso vem causando uma série de dificuldades no dia a dia. Cadastrar peças, registrar entradas e saídas... tudo isso tem tomado tempo demais e gerado confusões que poderiam ser evitadas com uma solução mais prática. Esse processo já se tornou inviável para acompanhar a movimentação real do estoque.

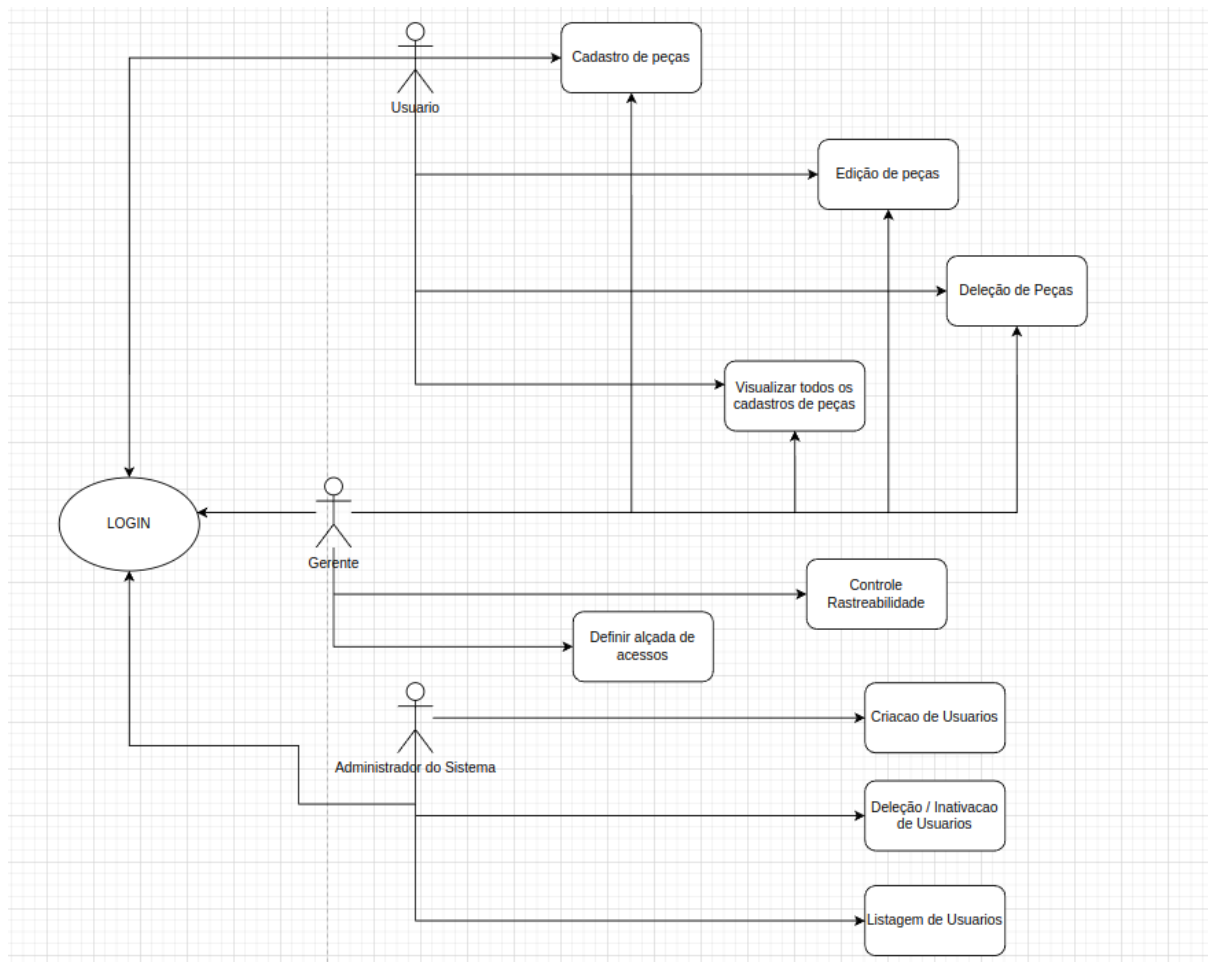
Pensando nisso, vamos desenvolver um sistema web que torne tudo mais simples e organizado. Nele, será possível cadastrar peças com facilidade, consultar o histórico de

retiradas e entradas com datas e horários precisos, e ainda controlar quem pode ver ou fazer o quê, de acordo com o nível de acesso de cada usuário.

Além disso, para garantir que o sistema esteja sempre disponível, mesmo em caso de falha de algum servidor, vamos adotar uma estrutura multicloud com failover. Isso significa que, se um servidor cair, outro entra em ação automaticamente, sem que o sistema pare de funcionar.

- Diagrama de Arquitetura do Sistema:

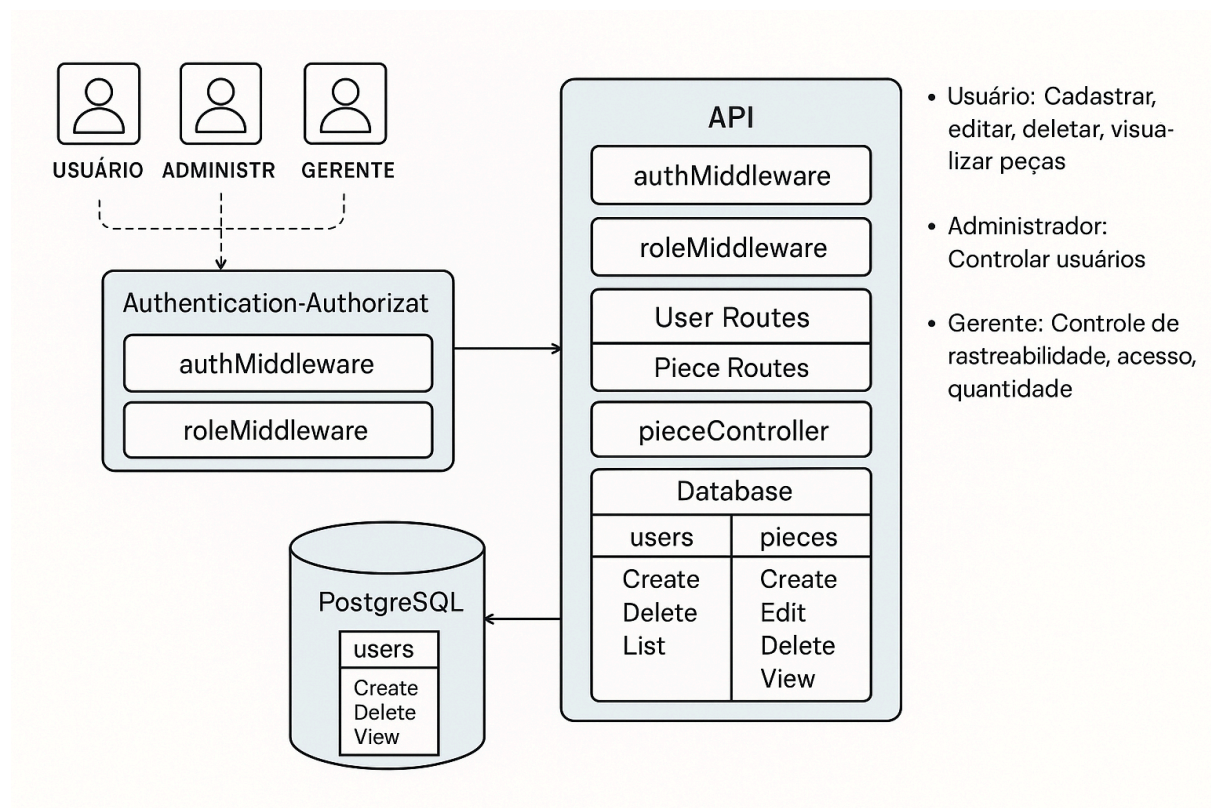
ARQUITETURA DO FRONTEND:



Descrição:

O usuário irá logar com seu email e senha, o sistema verificará se este usuário for do tipo 'usuário', 'administrador' ou 'gerente', para cada um dos tipos, haverá algumas funcionalidades específicas. O tipo 'usuário' poderá cadastrar peças, editar, deletar e visualizar as peças cadastradas. Somente o administrador irá controlar os usuários do sistema, realizando a criação, deleção e listagem de usuários. Já o Gerente, possui o controle total, além de ter o controle de rastreabilidade de peças, ele quem define quem tem acesso ao que, tendo controle também a quantidade de peças cadastradas.

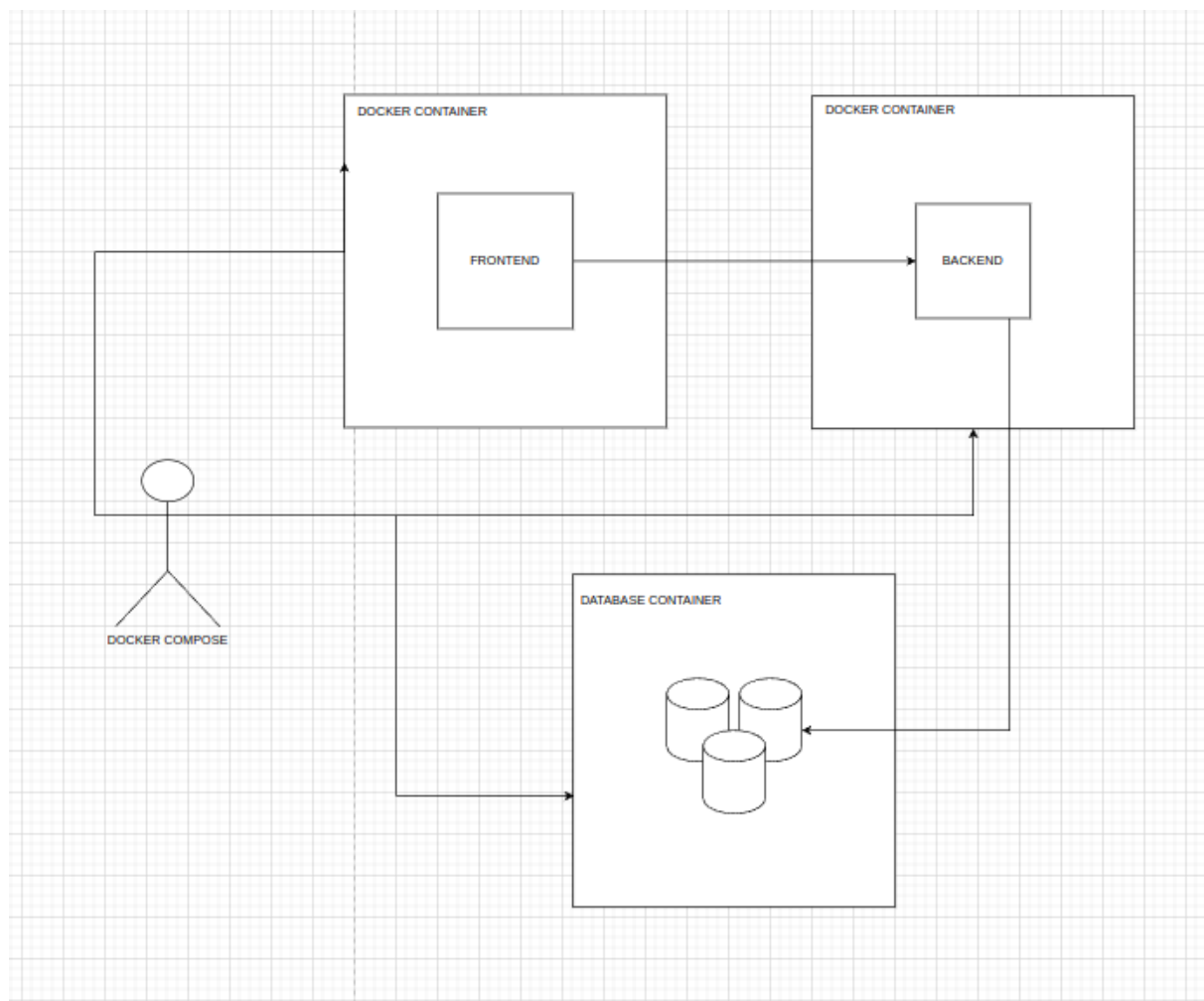
ARQUITETURA DO BACKEND:



Descrição:

arquitetura do backend com três camadas principais: Autenticação e Autorização, API, e Banco de Dados PostgreSQL. Usuários se autenticam e passam por middlewares (**authMiddleware** e **roleMiddleware**) que controlam o acesso conforme o tipo de usuário (usuário, administrador, gerente). A API é dividida em rotas e controladores que acessam as tabelas **users** e **pieces** no banco de dados. Cada tipo de usuário possui permissões específicas, destacadas ao lado direito do diagrama.

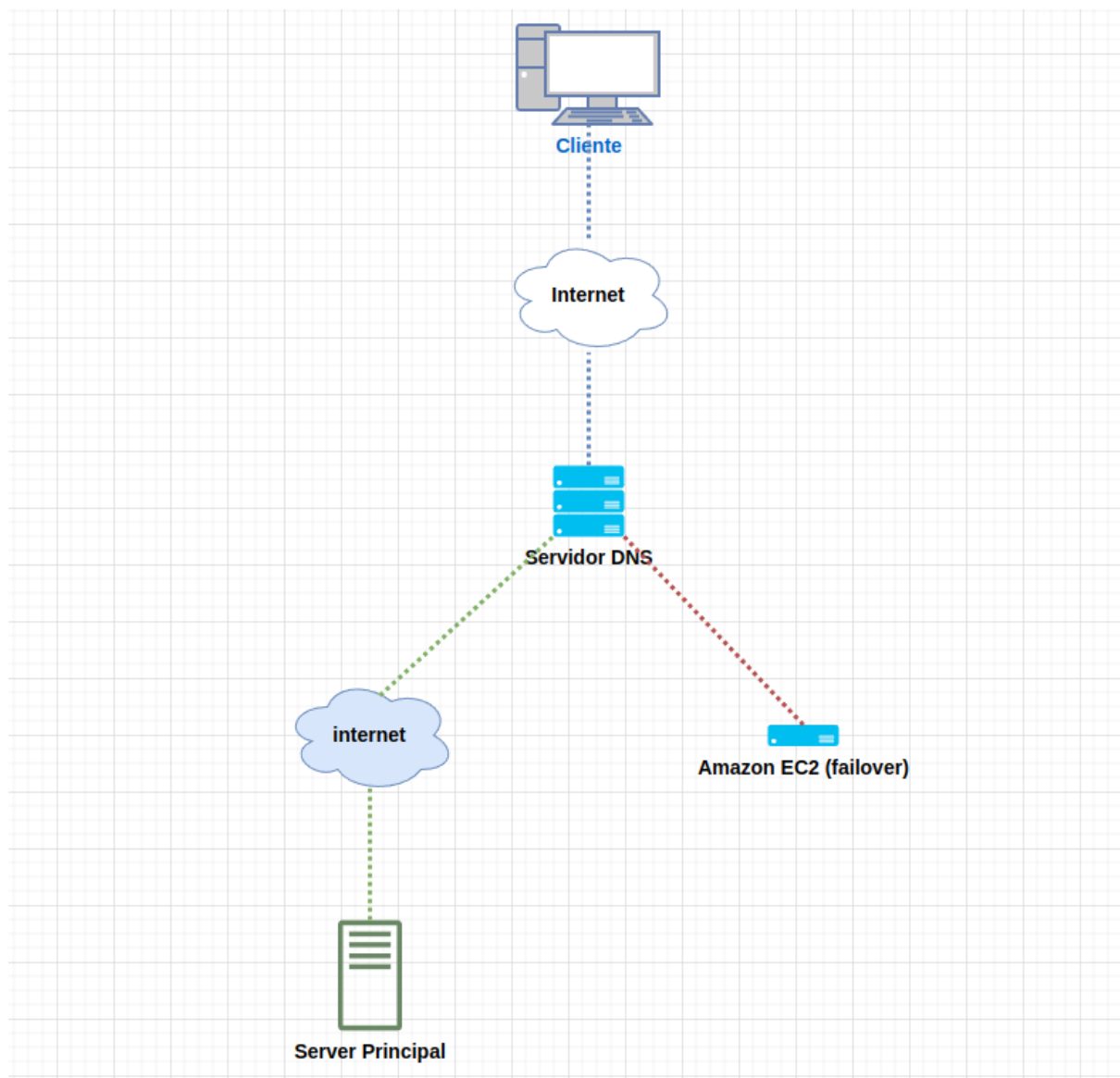
ARQUITETURA DA APLICAÇÃO COM DOCKER:



Descrição:

A aplicação será containerizada, tanto frontend, quanto backend e banco de dados, faremos uma imagem docker para cada um e iremos gerenciar essas imagens com o Docker compose, para que a aplicação venha ser executada em qualquer ambiente sem precisar se preocupar com drivers ou dependências que variam de sistema operacional para sistema operacional.

ARQUITETURA DA INFRAESTRUTURA:



Descrição:

O diagrama acima demonstra o funcionamento da infraestrutura com alta disponibilidade para garantir continuidade do serviço caso haja uma indisponibilidade no servidor on-premise, caso haja, automaticamente todas as solicitações serão redirecionadas para o servidor AWS.

Tecnologias Utilizadas:

➤ Cloud / Infraestrutura:

- AWS EC2: Servidor de contingência
- Amazon Route 53: Gerenciamento DNS com Failover

- VPN IPSec: Conexão segura entre local e nuvem

➤ **Componentes do Diagrama:**

1. Cliente o Acessa o serviço por meio de um domínio.
2. Balanceador de DNS (como Route 53) o Verifica a saúde do servidor local periodicamente (health check). o Redireciona o tráfego para o servidor AWS se o local estiver inacessível.
3. Servidor Local (On-Premises) o Servidor principal que fornece o serviço.
4. Instância EC2 (AWS) o Mantida em estado de standby. o Assume o tráfego automaticamente em caso de falha local.
5. S3 ou outro serviço de backup o Armazena dados replicados do servidor local para garantir sincronização.

PROTÓTIPO FIGMA:

<https://www.figma.com/design/gJXQtIUOBkxhTgrUjRR95L/Stock-Control?node-id=1-2&t=pvvUuoqdALy36rkA-1>

REPOSITÓRIO REMOTO:

https://github.com/lourenco-daniel/projeto_aplicado