**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC Minas Virtual**

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

Trabalho de Conclusão de Curso

SGP

Sistema de Gerenciamento de Projetos

Jordhan Pimentel Carvalho

Belo Horizonte

Abril 2022.

# Trabalho de Conclusão de Curso

**Sumário**

[Trabalho de Conclusão de Curso 3](#_Toc101096589)

[1. Cronograma de trabalho 4](#_Toc101096590)

[2. Diagrama de casos de uso 5](#_Toc101096591)

[3. Requisitos não-funcionais 5](#_Toc101096592)

[4. Protótipo navegável do sistema 6](#_Toc101096593)

[5. Diagrama de classes de domínio 7](#_Toc101096594)

[6. Modelo de componentes 8](#_Toc101096595)

[6.1. Padrão arquitetural 8](#_Toc101096596)

[6.2. Diagrama de componentes 9](#_Toc101096597)

[6.3. Descrição dos componentes 10](#_Toc101096598)

[7. Diagrama de implantação 11](#_Toc101096599)

[8. Plano de Testes 12](#_Toc101096600)

[9. Estimativa de pontos de função 14](#_Toc101096601)

## Cronograma de trabalho

|  |  |  |  |
| --- | --- | --- | --- |
| **Datas** | | **Atividade / Tarefa** | **Produto / Resultado** |
| **De** | **Até** |
| 07 / 01 / 22 | 12 / 01 / 22 | 1. Leitura do roteiro e escolha do tema. | Definição do tema do projeto. |
| 13 / 01 / 22 | 15 / 01 / 22 | 1. Criação do cronograma das atividades. | Cronograma de atividades documentado. |
| 16 / 01 / 22 | 25 / 01 / 22 | 1. Desenvolvimento do Diagrama de casos de uso. | Diagrama de Caso de Uso. |
| 26 / 01 / 22 | 31 / 01 / 22 | 1. Especificação dos Requisitos não-funcionais. | Listagem dos Requisitos-não funcionais. |
| 01 / 02 / 22 | 10 / 02 / 22 | 1. Criação do Protótipo navegável do sistema. | Protótipo navegável criado no Figma. |
| 11 / 02 / 22 | 17 / 02 / 22 | 1. Estudo do Diagrama de classes de domínio. | Entendimento do Diagrama de classes de domínio. |
| 18 / 02 / 22 | 22 / 02 / 22 | 1. Desenvolvimento do Diagrama de classes de domínio. | Diagrama de classes de domínio definido. |
| 23 / 02 / 22 | 27 / 02 / 22 | 1. Análise e escolha do padrão arquitetural. | Definição do padrão arquitetural. |
| 28 / 02 / 22 | 06 / 03 / 22 | 1. Desenvolvimento do Diagrama de componentes. | Diagrama de componentes elaborado. |
| 07 / 03 / 22 | 11 / 03 / 22 | 1. Descrição dos componentes. | Descrição dos componentes feita. |
| 12 / 03 / 22 | 16 / 03 / 22 | 1. Elaboração do Diagrama de implantação. | Diagrama de implantação elaborado. |
| 17 / 03 / 22 | 23 / 03 / 22 | 1. Criação do Plano de Testes. | Documentação do plano de testes. |
| 24 / 03 / 22 | 28 / 03 / 22 | 1. Estimar os Pontos de função. | Documentação dos pontos de função. |
| 29 / 03 / 22 | 05 / 04 / 22 | 1. Revisão do TCC. | Trabalho revisado. |
| 06 / 04 / 22 | 15 / 04 / 22 | 1. Criar vídeo apresentação do protótipo navegável. | Vídeo de apresentação do protótipo gravado. |
| 15 / 04 / 22 | 17 / 04 / 22 | 1. Entrega do trabalho. | Envio do TCC na plataforma AVA. |

## Diagrama de casos de uso

|  |
| --- |
|  |

## Requisitos não-funcionais

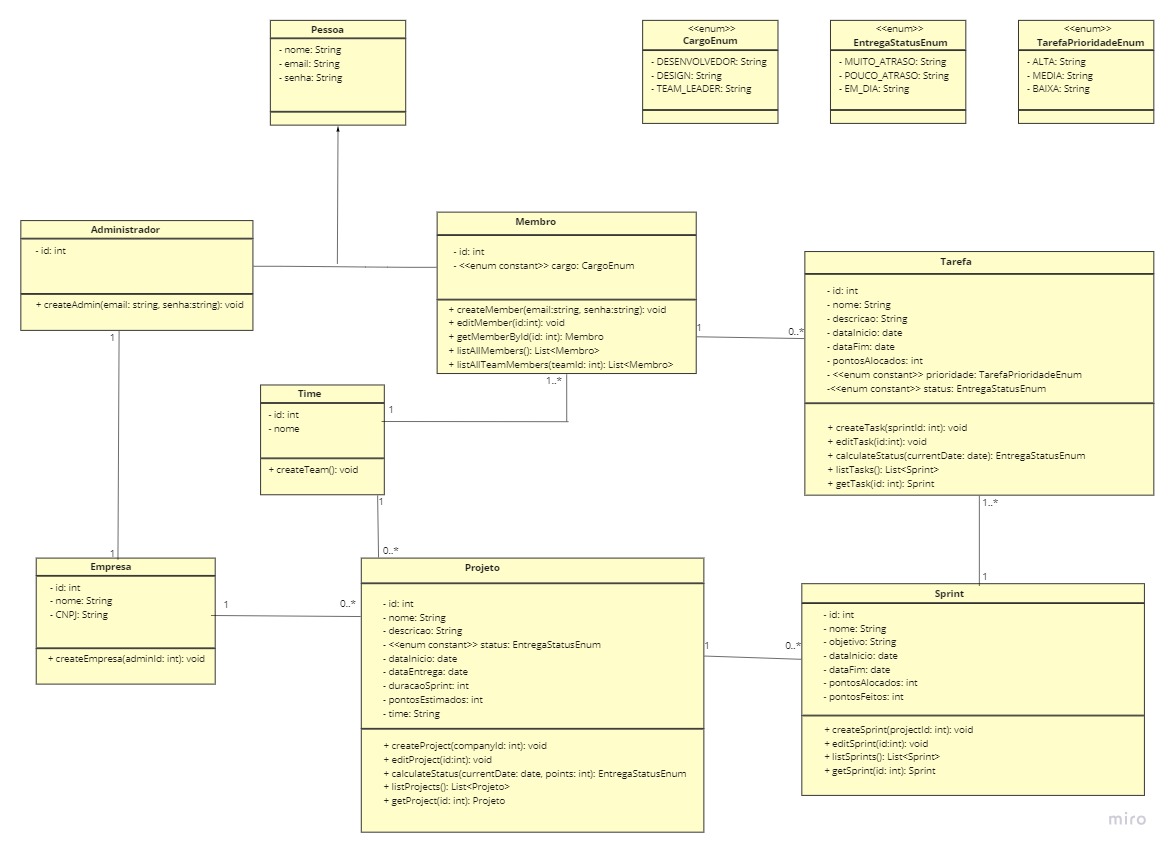
|  |  |
| --- | --- |
| RN-01 | O sistema deverá ser responsivo, possibilitando a navegação de variados tamanhos de tela. |
| RN-02 | O sistema estará acessível por pelo menos 99% do tempo. |
| RN-03 | O sistema deverá ser compatível com os sistemas operacionais Windows, macOS e Linux. |
| RN-04 | O sistema deverá utiliza um banco de dados relacional. |
| RN-05 | A infraestrutura do sistema deverá ser implementada na AWS. |
| RN-06 | A senha do usuário deve ser criptografada antes de ser salva no banco. |
| RN-07 | A API do sistema funcionará de acordo com o protocolo HTTP, seguindo o padrão REST. |
| RN-08 | O sistema deve ser escalável o suficiente para suportar grandes picos de usuário. |

## Protótipo navegável do sistema

Abaixo se encontram três links distintos, o primeiro para o repositório contendo o arquivo figma, um pdf das telas e imagens de cada tela de navegação, o segundo é um link para o vídeo no youtube e o terceiro é um link para o protótipo navegável interativo no figma:

* Repositório: <https://github.com/Jordhan-Carvalho/puc-minas>
* YouTube: <https://youtu.be/DW55nTVjqHo>
* Figma: <https://www.figma.com/proto/1ImzlsvmkKbnStk8MPpVjQ/SGP?node-id=10%3A2&scaling=scale-down&page-id=0%3A1&starting-point-node-id=10%3A2>

## Diagrama de classes de domínio



## Modelo de componentes

### Padrão arquitetural

O padrão arquitetural escolhido para o desenvolvimento da aplicação foi o MVC, por ser considerado um padrão já provado em diversas aplicações de alto nível além de ser bem popular, sendo assim, de fácil adesão para novos engenheiros no projeto.

A linguagem escolhida para o desenvolvimento foi a Typescript, assim aproveitamos o benefício de ter somente uma linguagem para toda a aplicação e com a vantagem da tipagem.

No front-end iremos utilizar a biblioteca React para facilitar a manipulação do HTML e Redux para centralizar e controlar o estado da aplicação baseado nos dados vindos em formato json do back-end que seguirá o padrão RESTful API. O deploy será feito através de um bucket no AWS S3 e usará o CDN da Amazon, o AWS Cloudfront para distribuição.

O back-end se utilizará do NodeJs e do framework NestJS, que facilitará a injeção de dependências, fazendo com que possamos realizar os testes unitários de maneira mais prática. Ele estará contido em um container Docker e disponibilizado na AWS ECR, o deploy será realizado através do serviço de containers manejado pela Amazon, conhecido como AWS Fargate, encarregado de abstrair a parte operacional de escalabilidade, patches, proteção e gerenciamento de servidores.

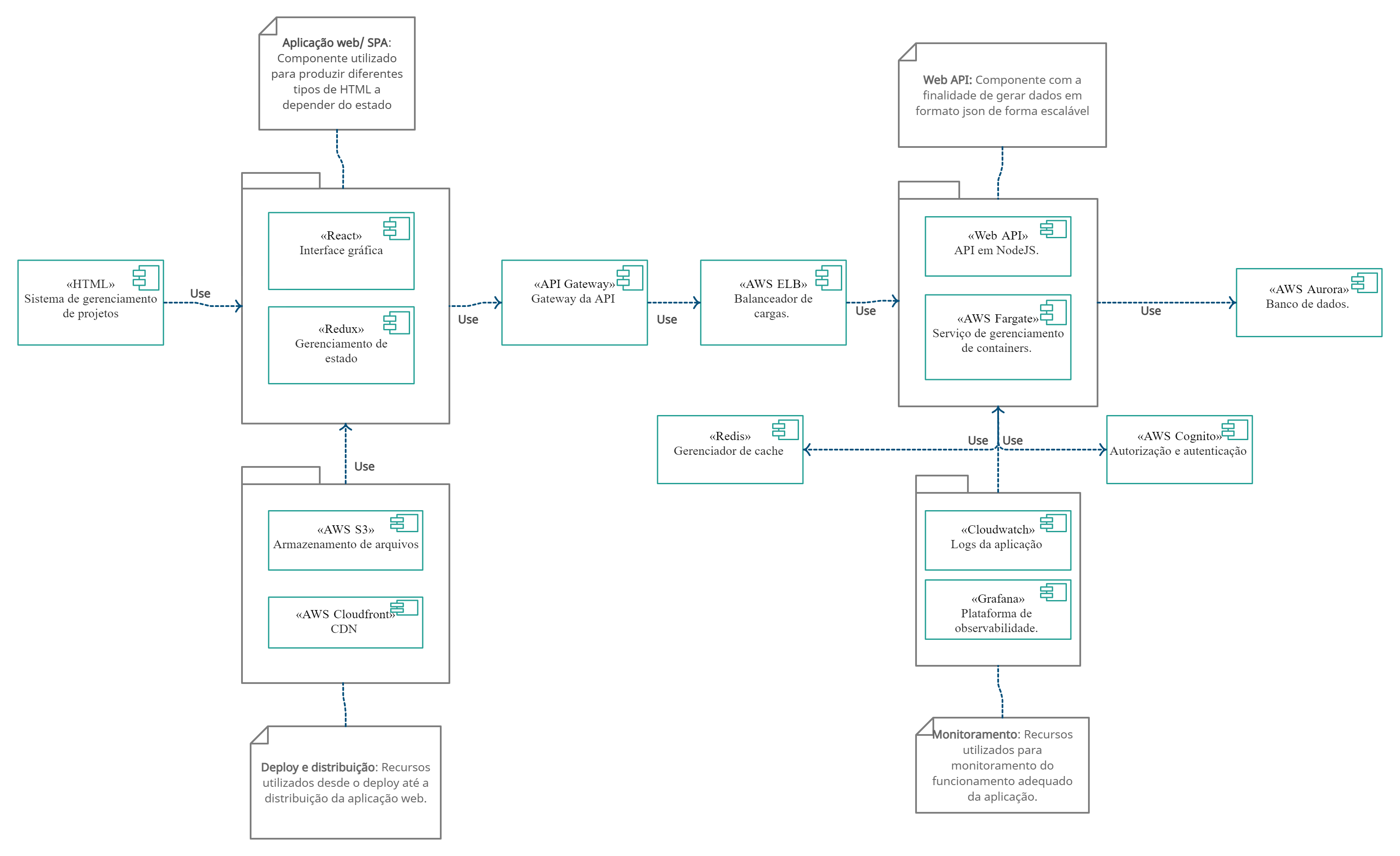
Utilizaremos também o AWS API Gateway para expor os endpoints e comunicar com a rede (VPC) interna, passando a chamada para os balanceadores de cargas, AWS ELB, que por sua vez, faz a comunicação direta com os containers alocados no AWS Fargate.

Para o banco de dados iremos utilizar o Amazon Aurora compatível com o postgreSQL, por conta de sua alta performance, tolerância a falhas e escalabilidade quase que instantânea.

Todos os serviços de infraestrutura serão mantidos como IaC (Infrastructure as Code), se utilizando do Amazon CDK em Typescript, assim temos um versionamento e controle mais apurado da infraestrutura.

O monitoramento da aplicação ficará por conta da plataforma open source, grafana, que será integrada ao sistema de logs da AWS conhecido como Cloudwatch.

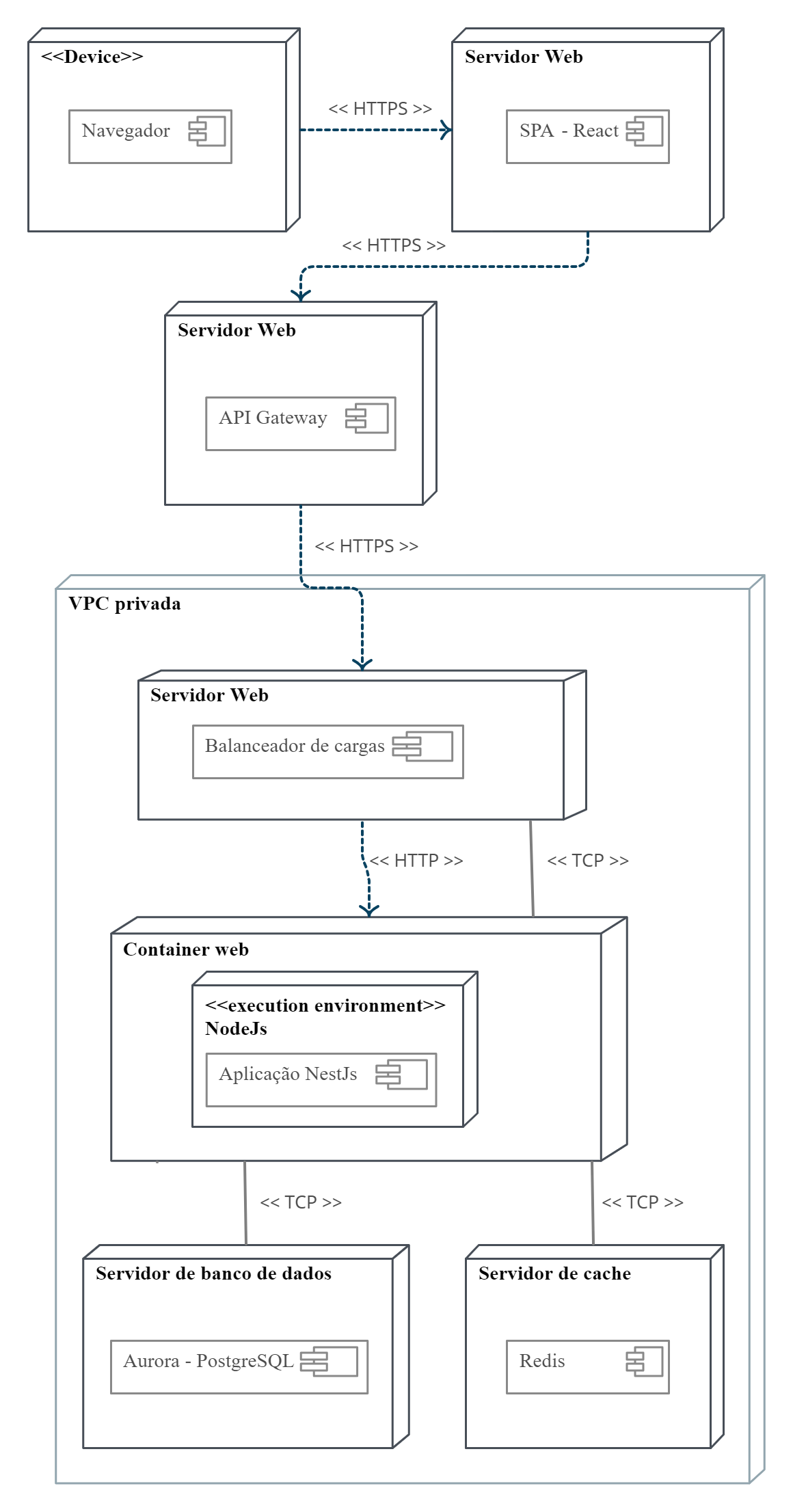
### Diagrama de componentes



### Descrição dos componentes

|  |  |  |
| --- | --- | --- |
| **Número** | **Componente** | **Descrição** |
|  | ReactJS | Componente da camada de apresentação responsável pela interface gráfica da aplicação. |
|  | Redux | Biblioteca responsável pelo gerenciamento do estado da aplicação no front-end. |
|  | AWS S3 | Serviço fornecido pela Amazon para armazenamento de objetos. |
|  | AWS Cloudfront | Serviço de rede de entrega de conteúdo (CDN) fornecido pela Amazon. |
|  | API Gateway | Gateway fornecido pela Amazon para APIs REST e WebSocket em qualquer escala. |
|  | AWS ELB | Balanceador de carga automático fornecido pela Amazon. |
|  | AWS Fargate | Sistema de gerenciamento de containers fornecido pela Amazon. |
|  | Web API | Componente responsável pelo tratamento dos dados recebido pelo gateway e por retornar os valores em formato JSON para consumo. |
|  | Redis | Banco de dados in-memory responsável pelo cache. |
|  | AWS Cognito | Serviço de autenticação/autorização fornecido pela Amazon. |
|  | AWS Cloudwatch | Serviço fornecido pela Amazon responsável por registrar e disponibilizar os registros de log. |
|  | Grafana | Plataforma open source responsável por observar as métricas e a saúde da aplicação. |
|  | AWS Aurora | É um banco de dados relacional compatível com PostgreSQL fornecido pela Amazon. |

## Diagrama de implantação



## Plano de Testes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Número** | **Caso de uso** | **Objetivo do caso de teste** | **Entradas** | **Resultados esperados** |
| 1 | Adicionar/remover membro | Validar o cadastro com sucesso de um novo membro na empresa. | - Entrar no sistema com uma conta com permissão de Administrador na empresa; | - Sistema volta para tela de Membros com o novo membro listado. |
| - Acessar a tela Membros; |
| - Clicar no ícone “+” no canto superior direito; |
| - Informar os seguintes dados nos respectivos campos: |
| [E-mail: joao@hotmail.com](mailto:joao@hotmail.com) |
| **Nome**: João Pereira |
| **Cargo**: Desenvolvedor |
| **Time**: A |
| **Senha provisória**: trocar123 |
| - Clicar em “Confirmar” |
| 2 | Adicionar/remover membro | Verificar que não é possível cadastrar um membro quando o email já está em uso por outro usuário. | - Realizar o cadastro de um membro com os dados informados no caso de teste 1. | - O sistema apresenta uma mensagem de erro informando o e-mail está em uso; - O sistema permanece na tela de adicionar membro. |
| - Acessar a tela Membros; |
| - Clicar no ícone “+” no canto superior direito; |
| - Informar os seguintes dados nos respectivos campos: |
| [E-mail: joao@hotmail.com](mailto:joao@hotmail.com) |
| **Nome**: João Madureira |
| **Cargo**: Desenvolvedor |
| **Time**: B |
| **Senha provisória**: trocar123 |
| - Clicar em “Confirmar” |
| 3 | Adicionar/remover membro | Validar a exclusão de um membro existente. | - Realizar o cadastro de um membro com os dados informados no caso de teste 1; | - O sistema atualiza a lista de membros sem o membro excluído. |
| - Como usuário Administrador acessar a tela Membros; |
| - Achar o membro criado “João Pereira”; |
| - Clicar no ícone de subtração “-”; |
| 4 | Criar e gerir sprint | Verificar que é possível criar uma nova sprint sendo líder do time. | - Entrar no sistema com uma conta com permissões de Lider de time; | - O sistema volta para tela de Sprints, com a nova sprint listada. |
| - Acessar a tela de Projetos; |
| - Clicar no Projeto em que é responsável; |
| - Clicar na opção Sprints no menu; |
| - Clicar no botão +; |
| - Informar os seguintes dados nos respectivos campos: |
| **Objetivo da sprint**: teste |
| **Data de início**: 01/01/2022 |
| - Clicar em “Confirmar” |
| 5 | Criar e gerir sprint | Verificar que o botão de adicionar sprint não está disponível para usuários que não sejam o líder do time responsável pelo projeto. | - Entrar no sistema com uma conta com permissões de Lider de time; | - O botão de adição “+” não deve estar disponível. |
| - Acessar a tela de Projetos; |
| - Clicar no Projeto em que é responsável; |
| - Clicar na opção Sprints no menu; |
| - Clicar no botão +; |
| 6 | Visualizar Métricas do projeto | Verificar que as métricas de desempenho do projeto são exibidas corretamente. | - Entrar no sistema com uma conta com permissões de Administrador na empresa; | - Sistema exibe informações de desempenho do projeto e time/membros vinculados. |
| - Acessar a tela Desempenho de projetos; |
| - Clicar em um projeto existente; |
| 7 | Visualizar Métricas do projeto | Verificar que as métricas não são exibidas caso o usuário não seja líder do projeto ou administrador. | - Entrar no sistema com uma conta com permissões de membro; | - Sistema exibe mensagem de erro de permissão. |
| - Acessar a tela de Desempenho de projetos; |
| - Clicar em um projeto existente; |

## Estimativa de pontos de função

Link do repositório: <https://github.com/Jordhan-Carvalho/puc-minas>

Previa da planilha:

