**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC Minas Virtual**

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

Trabalho de Conclusão de Curso

PMaster – Sistema de Gerenciamento de Projetos

GABRIEL HENRIQUE DE OLIVEIRA LEITE

Belo Horizonte

Abril, 2022.

# Trabalho de Conclusão de Curso

**Sumário**

Trabalho de Conclusão de Curso 3

1. Cronograma de trabalho 4

2. Diagrama de casos de uso 5

3. Requisitos não-funcionais 5

4. Protótipo navegável do sistema 5

5. Diagrama de classes de domínio 6

6. Modelo de componentes 7

6.1. Padrão arquitetural 7

6.2. Diagrama de componentes 8

6.3. Descrição dos componentes 9

7. Diagrama de implantação 10

8. Plano de Testes 11

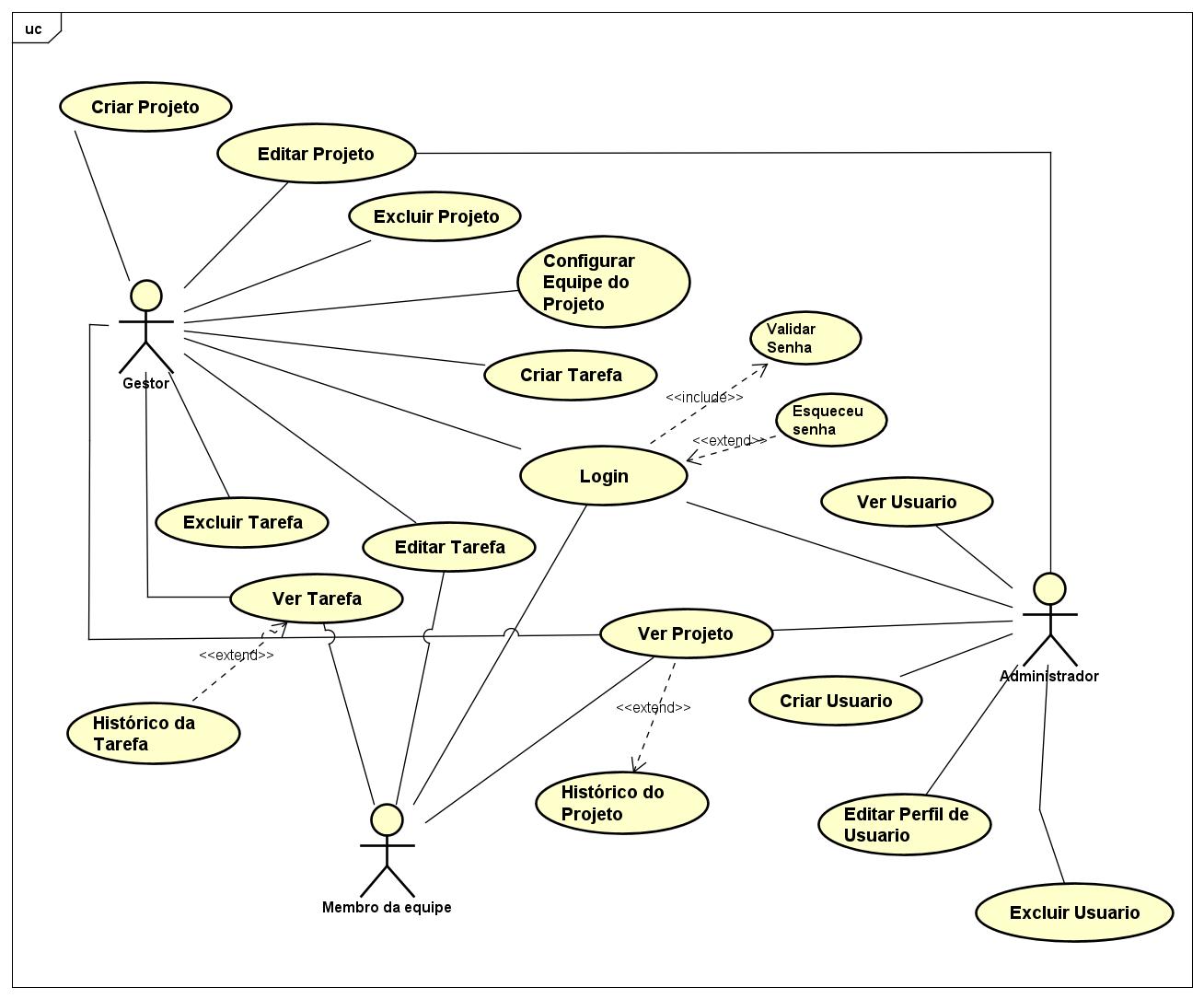
9. Estimativa de pontos de função 12

10. Referências 12

## Cronograma de trabalho

|  |  |  |  |
| --- | --- | --- | --- |
| **Datas** | | **Atividade / Tarefa** | **Produto / Resultado** |
| **De** | **Até** |
| 29 / 11 / 21 | 03 / 12 / 21 | 1. Diagrama de casos de uso | Produzir o diagrama de casos de uso com os atores e casos do sistema. |
| 06 / 12 / 21 | 10 / 12 / 21 | 2. Requisitos não-funcionais | Listar e especificar todos os requisitos não-funcionais essencial para o sistema. |
| 13 / 12 / 21 | 07 / 01 / 22 | 3. Protótipo navegável do sistema | Fazer um protótipo do sistema mostrando as telas e os fluxos do sistema. |
| 10 / 01 / 22 | 21 / 01 / 22 | 4. Modelo de componentes | Diagrama contendo os componentes do sistema e também uma especificação de cada um deles. |
| 24 / 01 / 22 | 28 / 01 / 22 | 5. Diagrama de implantação | Fazer o diagrama de como será a implantação do sistema. |
| 31 / 01 / 22 | 04 / 02 / 22 | 6. Plano de testes | Listagem e descrição dos testes que deve ser feito no sistema para uma melhor qualidade do software. |
| 07 / 02 / 22 | 25 / 02 / 22 | 7. Planilha de pontos de função | Preenchimento da planilha da contagem de pontos de função |
| 28 / 02 / 22 | 04 / 03 / 22 | 8. Organização dos materiais | Separar os diagramas feitos e o trabalho para compartilhar com a banca |
| 07 / 03 / 22 | 11 / 03 / 22 | 9. Gravar o protótipo | Fazer um vídeo com o protótipo navegável e seu funcionamento. |
| 14 / 03 / 22 | 18 / 03 / 22 | 10. Revisão do trabalho | Revisão do trabalho, tanto na parte linguística quanto na parte lógica |
| 17 / 04 / 22 | 17 / 04 / 22 | 11. Entrega do trabalho | Anexo do trabalho na plataforma da PUC |

## Diagrama de casos de uso



## Requisitos não-funcionais

* O sistema deve rodar em qualquer dispositivo móvel.
* O sistema deve ser responsivo nas interfaces gráficas.
* O sistema só poderá gravar a senha dos usuários de forma criptografada no banco de dados.
* O sistema deve manter a sessão de login do usuário por no máximo 10 minutos de inatividade.
* O sistema deverá ter interoperabilidade com a plataforma da Google para login e autenticação via Gmail.
* O sistema deverá processar pelo menos 10 requisições por segundo.
* O sistema deve ter uma disponibilidade de 99.0% ou mais.
* O sistema deve tolerar uma taxa de falha de 1/1.000 segundos.
* O sistema deve utilizar Java na versão 8 ou superior para desenvolvimento.
* O sistema deve ser feito em microsserviços.
* O sistema deverá fazer cópias de backup com pelo menos 24 horas de intervalo entre as cópias.

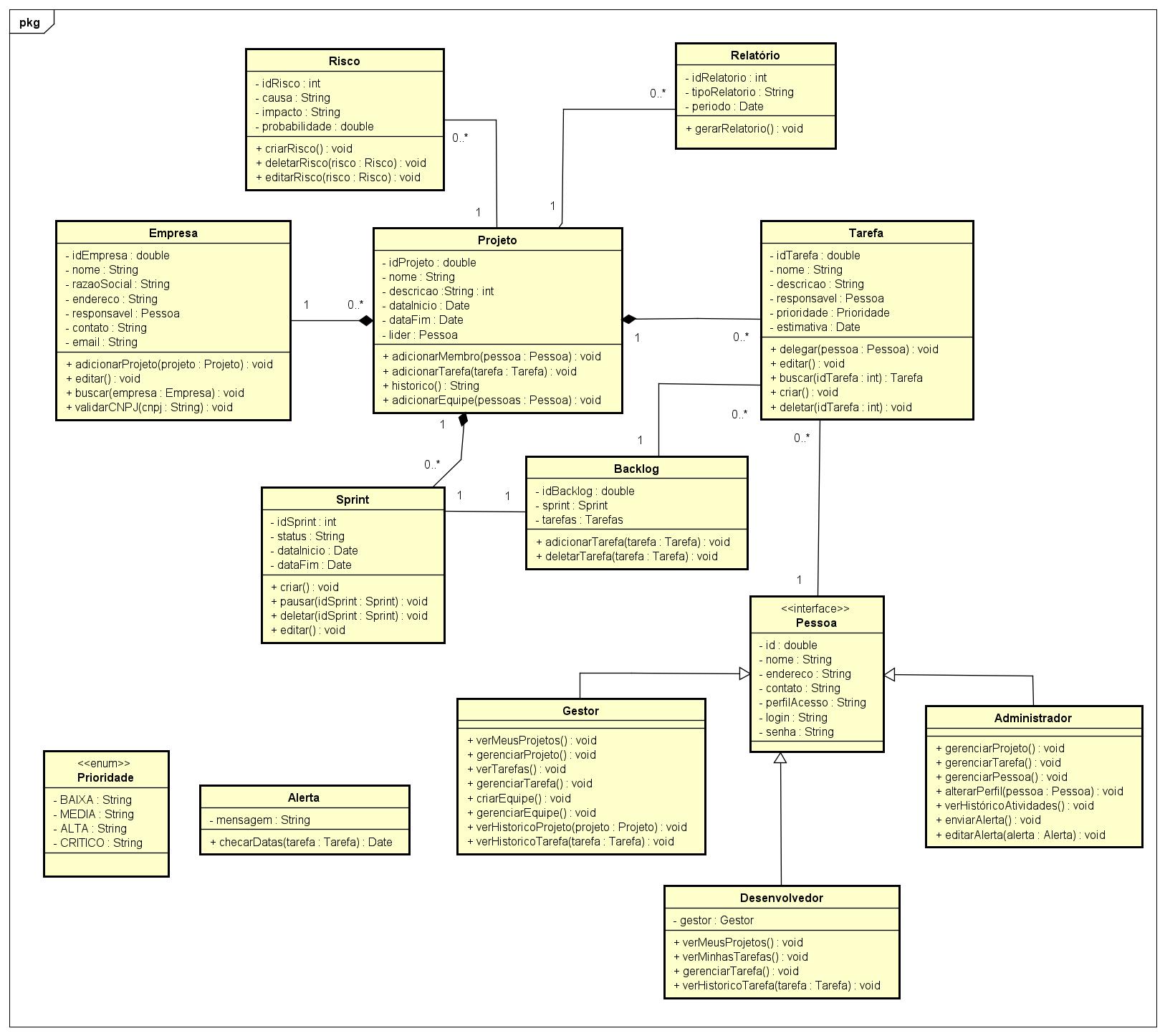
## Protótipo navegável do sistema

<Faça um protótipo navegável do sistema mostrando a tela inicial da aplicação e as de três casos de uso principais. A navegação entre as telas também precisa ser apresentada. Deve-se utilizar alguma ferramenta para a criação dos *wireframes* (como [Figma](http://figma.com/) ou [Balsamiq](https://balsamiq.com/wireframes/), por exemplo).

Faça um vídeo de apresentação do protótipo navegável desenvolvido e disponibilize-o de forma que a banca de avaliação do TCC possa visualizá-lo. Esse vídeo deve ter duração de, no máximo, 3 minutos. Utilize, preferencialmente, o formato AVI.

Nesta seção, indique o *link* desse vídeo e do repositório (como o [GitHub](http://github.com/), [Bitbucket](https://bitbucket.org/product/), etc) onde seu protótipo navegável está disponível.>

## Diagrama de classes de domínio



## Modelo de componentes

### Padrão arquitetural

Para o Sistema de Gerenciamento de Projetos **PMaster** foi escolhido uma arquitetura de microsserviços para se ter uma maior independência entre serviços e, com isso, os desenvolvedores terem mais autonomia para tomarem decisões importantes em serviços específicos, essa arquitetura é também utilizada para ter uma maior escalabilidade da aplicação e uma implantação contínua e automatizada.

O Spring Boot vai ser utilizado no sistema por se apresentar um framework completo voltado para a Web e com várias vantagens como: ter servidores embutidos (Tomcat), aumento da produtividade, fácil configuração para ambientes de produção, oferece diversos plugins para melhor desenvolvimento e oferece também um padrão de testes unitários e de integração.

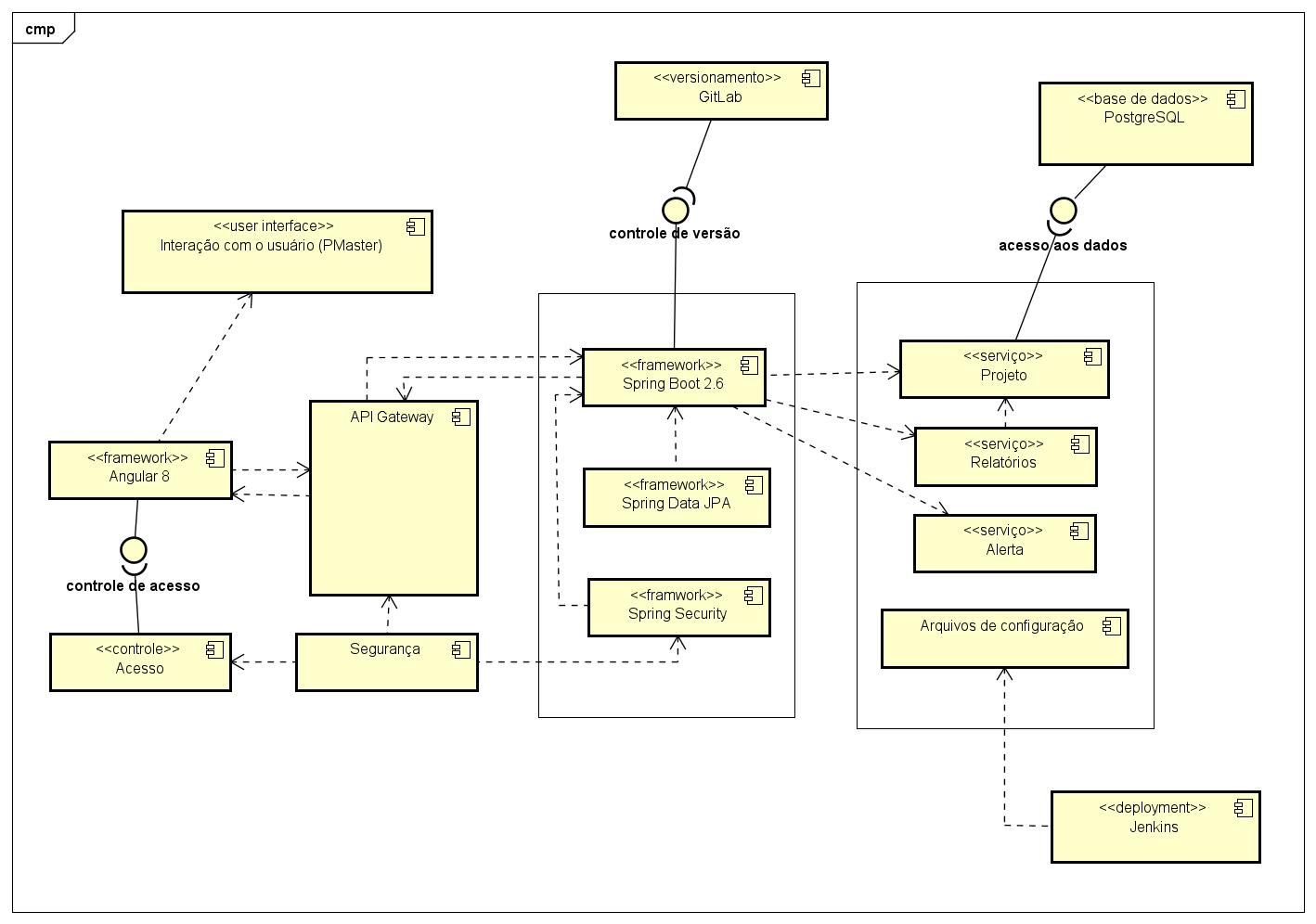
Além do Spring Boot será utilizado o Spring Data para a integração com o banco de dados podendo fazer consultas dinâmicas e ter muitas possibilidades para tratar a persistência de dados. Ainda no universo Spring será utilizado o Spring Security para autenticação e autorização com OAuth 2.0.

Para o banco de dados foi decidido pelo PostgreSQL por ter uma melhor forma de escalabilidade e também uma ótima contribuição da comunidade, tornando-o mais confiável e seguro.

Na questão do versionamento será utilizado o Gitlab como forma de trabalho em equipe e colaboração com o projeto, no lado da implantação será utilizado o Jenkins com todo o seu poder de entrega contínua e configuração de scripts.

Por último o Angular 8 vai fazer todo o processo de interação com usuário ao criar as páginas e tratar os dados. É uma ótima ferramenta por trabalhar com Typescript que tem maior escalabilidade e manutenabilidade, além de ser mantido pela Google e ter uma vasta comunidade por trás do projeto.

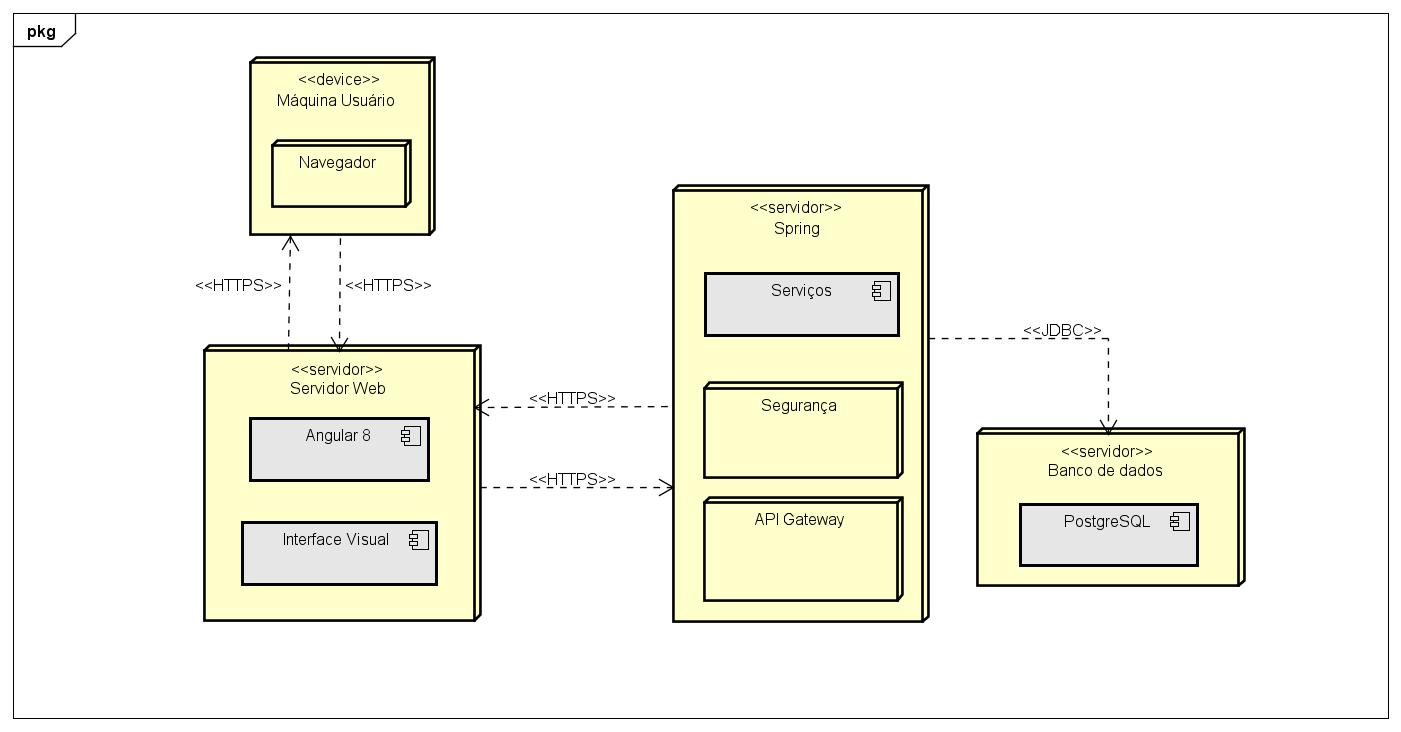
### Diagrama de componentes



### Descrição dos componentes

|  |  |  |
| --- | --- | --- |
| **Número** | **Componente** | **Descrição** |
|  | Spring Boot 2.6 | Framework voltado para Java que permite maior produtividade dentro de sistemas Web. |
|  | Spring Data | Parte do framework Spring que faz todo o gerenciamento e comunicação com o banco de dados |
|  | Spring Security | Parte do framework Spring responsável pela segurança da aplicação como as funções de autenticação e autorização |
|  | GitLab | Gerenciador de repositório baseado em Git para versionamento do projeto |
|  | Jenkins | Ferramenta para automatizar a implantação do sistema |
|  | PostgreSQL | Banco de dados de código aberto para persistência de dados |
|  | Angular 8 | Framework para construção de aplicações web |
|  | Projeto | Componente para gerenciar as funcionalidades principais do projeto |
|  | Relatórios | Componente focado em gerar relatórios e disponibilizá-los diariamente para os usuários |
|  | Alerta | Componente responsável por fazer leitura de logs do sistema e gerar alertas se necessário |
|  | Arquivos de configuração | Componente para toda configuração embutida no sistema |
|  | API Gateway | Componente de integração do sistema para gerenciamento de APIs e entradas/saídas do sistema |
|  | Segurança | Componente de gerenciamento de todos os pontos de segurança do sistema, seja para o mundo externo seja entre perfis de usuário |
|  | Acesso | Componente que gerencia acessos dos usuários e seus respectivos perfis |
|  | Interação com o Usuário | Componente do projeto que possibilita a interação do usuário com o sistema. |

## Diagrama de implantação



## Plano de Testes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Número** | **Caso de uso** | **Objetivo do caso de teste** | **Entradas** | **Resultados esperados** |
|  | Login |  |  |  |
|  | Login |  |  |  |
|  | Criar Projeto |  |  |  |
|  | Criar Projeto |  |  |  |
|  | Adicionar Tarefa |  |  |  |
|  | Adicionar Tarefa |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Estimativa de pontos de função

A contagem de pontos de função foi compartilhada no meu Github como público pelo link:

-<https://github.com/ghleite/TCC-PUCMinas/tree/main/Analise de PF>

Para visualizarem mais sobre os itens deste trabalho como os diagramas basta acessarem a raiz da pasta:

-<https://github.com/ghleite/TCC-PUCMinas>

## Referências

PRESSMAN, Roger. **Engenharia de Software:** Uma abordagem Profissional. 8. ed. McGraw-Hill/Bookman. 2014.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. Pearson Universidades. 2011.

MEDEIROS, Ernani. **Desenvolvendo Software com UML Definitivo 2.0**. Pearson Makron Books, 2004