Processo Admin

Gerenciador de Processos

# Objetivo

Sistema de gerenciamento de processos, com controle de usuários e perfil de acesso, bem como fluxo de parecer.

# Necessidades

* Visão de administrador
  + Incluir, excluir, atualizar e visualizar usuários,
* Visão de usuário-triador
  + Incluir e visualizar processos;
  + Atribuir um ou mais usuários a realizar um parecer sobre um processo;
* Visão de usuário-finalizador
  + Visualizar processos pendentes de parecer;
  + Incluir o parecer sobre o processo;

# Escopo técnico

* O sistema foi desenvolvido utilizando o AngularJS no Frontend e Java, com apoio do SpringBoot no Backend.

# O que veremos neste documento

* Informações sobre a construção do backend e frontend;
* Informações técnicas sobre o desenvolvimento;
* Manual do processo para rodar a aplicação.

# Backend

A API do sistema Gerenciador de Processos foi desenvolvida na IDE Eclipse Photon.

1. **Versões e Dependências**

Segue a lista das dependências do projeto:

* Java: 1.8;
* Spring Tools Suite: 3.9.5
* MongoDB: 4.0;
* Spring Boot: 1.5.15 [Spring Data, Spring Rest, Spring Security];
* JWT (Segurança): 0.9.5;

1. **Arquitetura**

Explicando a arquitetura do backend de maneira simplificada, segue este modelo:

Controllers => Services => Repository => Entity

|=> MongoDB

Desenvolvida no padrão REST, sobre uma camada de segurança para gerenciar o acesso aos serviços do sistema.

1. **Testes da API**

Os testes dos serviços da API foram realizados com o software Postman, acessando a cada serviço desenvolvido;

1. **Configurando o projeto**

Os testes dos serviços da API foram realizados com o software Postman, acessando a cada serviço desenvolvido;

1. **Executando a aplicação localmente**

Pré-requisito:

* Possuir Java 1.8, Spring Tools Suite e o MongoDB;

Segue o passo a passo para executar a API localmente:

* Abrir a IDE do Eclipse;
* Importar o projeto na opção “Existing Maven Projects” e selecionar a pasta da API;
* Executar a opção “Maven > Update Project...” para baixar as dependências do projeto (POM.xml)
* O arquivo de configuração application.properties, deverá ser ajustado, caso necessário:

spring.data.mongodb.host=localhost

spring.data.mongodb.database=process\_db

spring.data.mongodb.port=27017

jwt.secret=processAdmin\_XPT\*

jwt.expiration=604800

* Caso o serviço do mongoDB não esteja ativo, subir o serviço:

>> mongod --dbpath <DIRETORIO>\softplan-desafio-fullstack\data

1. **Gerando o jar da API**

Foi executado a opção de “Maven Install” o qual foi responsável por gerar o arquivo:

softplan-desafio-fullstack\process-api\target\**process-api-0.0.1-SNAPSHOT**

# Frontend

A aplicação frontend foi desenvolvida em AngularJS 5. Segue as versões de cada pacote instalado:

* Angular CLI: 6.1.5;
* Node: 8.11.4 LTS
* OS: win32 x64

1. **Criando o e subindo o projeto**

Executado o comando:

> ng new process-admin

Após a criação:

> ng serve

1. **Criando o e subindo o projeto**

AdminLTE foi o template escolhido.

Foi retirado muitos componentes e só adicionado o que iria ser utilizado.

1. **Criando os componentes header**

Serão criado os componentes existentes no sistema:

* Header, Menu, Footer, Home, Login

> ng g component ./components/header

> ng g component ./components/menu

> ng g component ./components/footer

> ng g component ./components/security/login

> ng g component ./components/home

1. **Criando os serviços para comunicação com a API Rest**

Users:

> ng g service ./services/user

Process:

> ng g service ./services/process

1. **Criando as rotas**

Criado as rotas para a página principal e a tela de login.

1. **Implementando o serviço de login**

**-- Realizar as operações de**