UNIVERSIDADE PAULISTA

INSTITUTO DE CIÊNCIAS EXATAS E TECNOLOGIA

D2538H2 - JONATHAN SILVA SALES DE OLIVEIRA

N2984J6 - VINICIUS DA SILVA PATO

D654JJ4 - CARLOS HENRIQUE G LEMES ALVES

N272BH2 - ALEX WONG DE FREITAS

ATIVIDADES PRÁTICAS SUPERVISIONADAS

Aplicativo de Prevenção de Incêndios

SÃO PAULO

2021

D2538H2 - JONATHAN SILVA SALES DE OLIVEIRA

N2984J6 - VINICIUS DA SILVA PATO

D654JJ4 - CARLOS HENRIQUE G LEMES ALVES

N272BH2 - ALEX WONG DE FREITAS

ATIVIDADES PRÁTICAS SUPERVISIONADAS

Aplicativo de Prevenção de Incêndios

Trabalho de Desenvolvimento de Sistemas Distribuídos feito durante o oitavo semestre da graduação em Ciência da Computação.

Orientador: Prof. Lauro Tomiatti

SÃO PAULO

2021

SUMÁRIO

1 OBJETIVO E MOTIVAÇÃO DO TRABALHO 5
2 INTRODUÇÃO 7
3 FUNDAMENTOS E TECNOLOGIAS DE SISTEMAS DISTRIBUÍDOS 9
4 PLANO DE DESENVOLVIMENTO DA APLICAÇÃO 11
4.1 Banco de Dados 11
4.2 Sistema Distribuído – Back-end 12
4.2.1 Spring Boot 12
4.2.2 Spring Framework 12
4.2.3 Spring Data JPA 13
4.2.4 H2 Database 13
4.2.5 Spring Security 14
4.2.6 Spring Web 14
4.2.7 JJWT, Validation API, Lombok 15
4.2.8 Maven 15
4.3 Sistema Distribuído – Front-end 15
4.3.1 HTML, CSS e JavaScript 15
4.3.2 NPM 15
4.3.3 Node.js 16
4.3.4 React 16
4.3.5 Create React App 16
5 PLANO DE DESENVOLVIMENTO DA APLICAÇÃO 17
5.1 Repositório 17
5.2 Entidades 17
5.3 Autenticação 17
5.4 Banco de Dados 17
5.5 Front-end 18
5.6 Back-end 18
6 RELATÓRIO COM LINHAS DE CÓDIGO 19
6.1 Entidade User 19
6.2 ReportController 21
6.3 ReportDTO 23
6.4 JWT Utils 25

6.5 JwtResponse 27
7 FICHAS DA APS 29
8 REFERÊNCIAS 32

1 OBJETIVO E MOTIVAÇÃO DO TRABALHO

A proposta da atividade prática supervisionada é o desenvolvimento de uma aplicação de sistema distribuído para dispositivo móvel, no qual a aplicação deve ser relativa ao gerenciamento de informações urbanas sendo elas alguns problemas recorrentes do meio urbano como enchentes, poluição, desmatamento, invasão térmica entre outros problemas no qual a tecnologia seria de grande ajuda para agilizar a resolução desses problemas com objetivo de amenizar os impactos, até coleta de dados para futuros estudos estatísticos, análise de dados e ciência de dados para implementação de algoritmo de machine learning para análise preditiva.

Sendo assim, o objetivo principal desta atividade que foi proposta a nós é desenvolver esta aplicação mobile para diminuir os impactos urbanos que são de extrema recorrência a ambientes urbanos, o objetivo de forma prática será benéfico tanto para órgãos públicos, os habitantes dos centros urbanos e os agentes dos centros urbanos como policiais, bombeiros, garis, lixeiros, carteiros entre outras membros que fazem todos os centros urbanos funcionar adequadamente, pois os mesmos são de certa forma a base do funcionamento desta grande cadeia.

2 INTRODUÇÃO

De acordo com que foi proposto e apresentado anteriormente o nosso grupo entrou em acordo e possuíra o foco principal nos incêndios nos centros urbanos cujo é bem recorrente porém não possuem tanta visibilidade por habitantes e órgãos públicos, realizando análises de dados fornecidos pelo corpo de bombeiros do Estado de São Paulo[1] podemos concluir que de 2016 até 2021 houve uma queda considerável no número de incêndios mas infelizmente o número de incêndios em vegetação ainda permanecesse muito alto, notícias recentes mostram que acidentes domésticos estão relacionado a incêndios, devido à grande crise em que o Brasil está passando no momento algumas famílias estão recorrendo a não usar gás natural como fonte de combustível para cozinhas mas sim lenhas devido ao alto custo do gás, consequentemente com a manipulação deste método o número de acidentes aumentou segundo notícias do G1, Yahoo Notícias, R7 e BBC.

Já os incêndios nas grandes vegetações são devido à grande expansão dos centros urbanos, para que grandes empresas, empresários ou até mesmo moradores destes centros urbanos consigam expandir suas casas, empresas para centros comerciais.

De acordo com que foi sobreposto, a alta demanda e o pouco investimento de tecnologia para resolução deste grande problema nos centros urbanos, nossa solução tem o intuito de auxiliar estes profissionais para diminuir o impacto trazido pelos incêndios e fazer com que o resgate e controle do acidente seja o mais breve possível com o auxílio de moradores dos centros urbanos, isso mesmo, cidadãos terá a praticidade de reportar incêndios ao corpo de bombeiros para que os mesmos cheguem e resolvam o problema, para evitar trote deste recorrências nosso aplicativo exigirá dados pessoais do usuário para que o ator do trote seja punido de acordo com a lei de acordo com o artigo 266 do código penal diz que interromper ou perturbar o serviço telefônico poderá incorrer pena de detenção de um a seis meses ou multa.

Com isto, este documento apresenta uma solução web para auxiliar na incidência desses incidentes de incêndios, facilitando na denúncia de transeuntes ao visualizar qualquer

---

[1] SECRETARIA DE SEGURANÇA PÚBLICA DE SÃO PAULO, **Dados Estatísticos do Corpo de Bombeiros do Estado de São Paulo**, disponível em: <http://www.ssp.sp.gov.br/Estatistica/CorpoBombeiro.aspx>, acesso em: 10 nov. 2021.

ocorrência. Foi pensado que facilitando o acesso a denúncia, o estado pode atuar de maneira mais ágil reduzindo assim os impactos causado.

A princípio existem diversos órgãos a se recorrer ao avistar um foco de incêndio ou queimada, de acordo com o portal federal do INPE, caso se depare com uma ocorrência pode se acionar uma destas opções: Bombeiros, Secretaria Estadual do Meio Ambiente, IBAMA, Prefeitura, e Instituto Florestal.

Este projeto traz um método rápido onde o usuário de maneira fácil acessará um site web, clicará em um simples botão de denúncia onde será captado os dados necessários para a denúncia.

Será apresentado as tecnologias utilizadas para o desenvolvimento deste projeto, onde será implementado conceitos de sistemas distribuídos, aplicação web, API, responsividade, Spring Boot e Spring Data.

Com este projeto é esperado surgir impactos positivos sobre a estatística de desastres causados por incêndios. Aplicando de forma correta todos os conceitos, será de extremo benefício para a sociedade, auxiliando indivíduos comuns a denunciarem com mais facilidade e auxiliando os órgãos responsáveis a receberem as denúncias de forma mais ágil.

3 FUNDAMENTOS E TECNOLOGIAS DE SISTEMAS DISTRIBUÍDOS

Há diversos fundamentos de tecnologia de sistemas distribuídos, dentre elas a monolítica e a de microsserviços, os sistemas monolíticos como os sistema está inteiro em um único bloco, seu desenvolvimento é mais ágil porém são muitas vezes são bastante robustos, único que apresenta um único processo em que diferentes componentes ligados a um único programa de uma única plataforma, mas há algumas desvantagens como a manutenção, dentre elas a manutenibilidade, conforme uma aplicação monolítica cresce, diversas funções são adicionadas a um mesmo código e processo, o que pode acarretar em quedas em cascatas da aplicação como um todo, o código se torna complexo e difícil de dar manutenção, as entregas por sua vez acabam se tornando mais críticas, menos frequentes e até estáveis, a escabilidade, por se tratar de um único código, todas as funcionalidades precisam ser escalada como um todo, normalmente escalada verticalmente, adicionando mais máquina (processador, memória, …) para aplicação, ou horizontalmente por modelos de balanceador de carga a linha de código robusta e a falta de flexibilidade de linguagens de programação.

Sendo assim foi decidido em nossa aplicação desenvolver a arquitetura em microsserviços, uma arquitetura de microsserviços, os serviços são refinados e os protocolos são leves, o objetivo é que os serviços possam consigam fornecer suas funcionalidades independentemente de fatores externos.

O acoplamento fraco reduz todos os tipos de dependências e as complexidades em torno delas, já que os desenvolvedores de serviço não precisam se preocupar com os usuários do serviço, eles não forçam suas mudanças para os usuários do serviço.

Portanto, permite que as organizações que desenvolvem software cresçam rapidamente e se tornem grandes, além de usar os serviços prontos para uso com mais facilidade juntamente apresenta mais vantagens como a facilidade e rapidez na atualização e implementação dos serviços, pois os códigos são divididos em pequenas aplicações, aumento da flexibilidade da infraestrutura.

Atualmente, quando temos a necessidade de entrega de um serviço as APIs são essenciais para a entrada de um serviço cada vez mais rico trazendo uma gama de funcionalidades para aplicação satisfazendo nossas necessidades e até mesmo na agilidade de desenvolvimento e de entrega do serviço proposto.

Com a necessidade da utilização de APIs, como todos sabemos o formato de dados mais comum do retorno de dados dessas APIs é o JSON, adotamos este formato também para representar os dados a serem transmitidos, foi acordado utilizar este formato de dado pois JSON

é um notação que permite estruturar dados em formato de texto para serem utilizados em diferentes tipos de sistemas, pois isto já traz uma valorização no conceito de interação entre sistemas para sistemas distribuídos, pois como vamos utilizar APIs de terceiros e de grande maioria das APIs e outros sistemas que permitem interação com terceiros adotam o formato JSON para o retorno dos dados.

4 PLANO DE DESENVOLVIMENTO DA APLICAÇÃO

Nosso plano de desenvolvimento é fazer back-end feito em Spring Boot com Spring Data usando Java para fazer um servidor de back-end que fornece dados através de APIs.

Sobre Spring Boot é um framework Java Open Source que tem como objetivo facilitar esse processo em aplicações Java, consequentemente, ele traz mais agilidade para o processo de desenvolvimento, uma vez que os desenvolvedores conseguem reduzir o tempo gasto com as configurações iniciais, com o Spring Boot conseguirmos abstrair e facilitar a configuração de servidores, gerenciamento de dependências, configuração de bibliotecas.

O Spring Data é um projeto Spring com proposta de unificar e facilitar acesso a diferentes tecnologias de armazenamento de dados, devido a este fato foi o que nos motivou a escolher o mesmo pois facilitaria o armazenamento de dados de nosso serviço, as principais vantagens do Spring Data além do benefício de conseguir interagir com diferentes tipos de banco de dados de forma padronizada, assim podemos escrever muito menos código repetido, pois as consultas podem ser definidas junto aos métodos de pesquisa e suas documentações, além disso as consultas JPQL são compiladas assim que o contexto Spring é criado e na primeira vez que a consulta é utilizada o que facilita a detecção de erros de sintaxe.

Para o Front End foi decidido utilizar a ferramenta Create React App devido a facilidade de desenvolver em React, React Router Dom para gerenciamento de rotas e Bootstrap, basicamente o front-end vai pegar os dados através de requisições nas APIs do back-end estabelecendo um sistema distribuído, todo esse desenvolvimento será feito através da IDE Visual Studio Code.

4.1 Banco de Dados

Foi utilizado um banco de dados relacional para armazenamento dos dados do sistema. Seu desenho lógico foi feito a partir da definição das entidades e seus relacionamentos.

Manipulações diretas ao banco serão realizadas apenas pelos desenvolvedores, o impacto dos demais usuários será apenas de maneira indireta em funções da aplicação.

Com o propósito de demonstração, a equipe vai utilizar um banco de dados em memória para facilitar a inicialização do mesmo e apresentação de algo, junto com scripts de inicialização dos dados.

Caso fosse ser utilizado em um ambiente de produção, a equipe optaria por utilizar o banco de dados Microsoft SQL Server por ter mais familiaridade com seu uso.

4.2 Sistema Distribuído – Back-end

A aplicação é representada por um sistema distribuído em duas partes, sendo elas o front-end e o back-end.

O back-end é feito em linguagem Java e é composto por uma feita em Spring Boot através de um projeto gerado pelo Spring Initializr com Java na versão 11, Maven e diversas dependências relacionadas ao Spring Boot.

4.2.1 Spring Boot

Spring Boot[2] é um framework que facilita a geração de aplicativos baseados em Spring de forma fácil, podendo ser configurado com diversas dependências e fazendo autoconfiguração e inclusão de diversos projetos do Spring que podem ser encaixados na aplicação de forma rápida e eficiente.

4.2.2 Spring Framework

O Spring Framework[3] possibilita o desenvolvimento e configuração de aplicações Java de forma mais eficiente.

Algumas das tecnologias fornecidas pelo Spring são gerenciamento e instanciação de componente, injeção de dependências, conversão de tipos, ORM, Serialização de JSON, transações, fornecimento de repositórios e controllers, template engines como thymeleaf, validação, anotações de código, eventos entre outras coisas através de seus diversos projetos de fácil acoplamento.

---

[2] VMWARE, **Spring.io**, disponível em: <https://spring.io/>, acesso em: 5 set. 2021.

[3] *Ibid*.

4.2.3 Spring Data JPA

Spring Data JPA[4] possibilita que o desenvolvedor consiga lidar com diversos tipos de bancos de dados de uma forma padronizada para manipulação e acesso.

Além disso existem outros projetos internos que fornecem capacidade de mapear os objetos e classes do programa em Java com as tabelas, linhas, e colunas de um banco de dados, criando um tipo de Object-Relational Mapping.

Com o Spring Data JPA, você ganha acesso ao Hibernate que seria uma implementação do Java Persistence API que facilita o mapeamento de objetos e classes para serem tratados como entidades com as tabelas de um banco de dados. E com a possibilidade de consultar e alterar os dados do banco de dados através das classes e objetos.

Além disso, o Spring Data JPA permite a criação de Repositórios para realizar as alterações no banco de dados de acordo com as entidades definidas dentro da aplicação.

O Spring Data JPA tem a capacidade de gerar automaticamente as tabelas de um banco de dados baseados nas entidades definidas através de um conceito chamado Data Definition Language e isso é utilizado no projeto para inicializar as tabelas para depois preencher com o conteúdo do script de inicialização de dados.

4.2.4 H2 Database

H2 Database é um banco de dados que fica em memória que foi utilizado nessa aplicação. Devido ao fato de ser em memória, fica fácil de fazer demonstrações uma vez que se tenha um script de inicialização de tabela e um script de inserção de dados.

Esse banco de dados pode ser facilmente trocado dentro do framework Spring por outros como MySQL, MongoDB, Sqlite ou SQL Server caso seja necessário, como por exemplo em um ambiente de produção.

---

[4] *Ibid*.

4.2.5 Spring Security

O Spring Security[5] é um dos projetos do Spring Framework que fornece ferramentas de autenticação, autorização e criptografia para serem utilizados no desenvolvimento de aplicações que precisam criptografar seus dados e senhas, uso de conceitos de autenticação e autorização de usuários baseados em cargos.

No caso do nosso projeto, ele é utilizado para gerar chaves de JSON Web Tokens, criptografar senhas e para autenticação e autorização para acesso a APIs e partes do sistema tanto no front-end quanto no back-end.

Esse framework pode ser utilizado para criptografar os dados armazenados para proteger os dados dos usuários, mas no projeto desenvolvido essa funcionalidade ainda não foi implementada.

4.2.6 Spring Web

Spring Web[6] é um framework que é utilizado na aplicação para gerar Controllers do tipo REST para fornecer APIs que podem ser usadas pelo front-end para fazer alterações no banco de dados através de Repositórios, podendo também para fazer a autenticação de usuários retornando JSON Web Tokens após o usuário ser autenticado e ao mesmo tempo sendo capaz de validar um JSON Web Token.

Além disso através do Spring Web é possível configurar funcionalidades como Cross-Origin Resource Sharing para permitir o compartilhamento de serviços e recursos com requisições de origens externas possibilitando que o front-end se comunique com as APIs do back-end.

---

[5] *Ibid*.

[6] *Ibid*.

4.2.7 JJWT, Validation API, Lombok

São dependências extras adicionadas ao projeto do Spring Boot para fornecer ferramentas de manipulação de JSON Web Tokens, validação de dados recebidos na API.

Lombok é um plugin que fornece anotações para implementação automática de construtores, getters e setters de classes, além de poder gerar alguns métodos que são usados com frequência.

4.2.8 Maven

Apache Maven é uma ferramenta para gerenciamento de projetos que facilita o gerenciamento de dependências, plugins, scripts de builds, ciclos entre diversas outras coisas de um projeto desenvolvido com ele.

É utilizado dentro do nosso projeto para gerenciar as dependências e plugins, além de cuidar dos scripts relacionados a build e run do projeto.

4.3 Sistema Distribuído – Front-end

4.3.1 HTML, CSS e JavaScript

As páginas web da aplicação que representam sua View foram feitas com HTML para montar a estrutura, CSS para estilos e JavaScript para lógica com auxílio de outros frameworks e bibliotecas como Bootstrap, React e Create React App.

4.3.2 NPM

NPM é um gerenciador de pacotes que é utilizado para gerenciar as dependências de uma aplicação que utilize Node.js.

4.3.3 Node.js

Node.js é um ambiente de execução de JavaScript que é utilizado para rodar código JavaScript fora do Browser utilizando o motor de JavaScript V8.

Ele é utilizado pela nossa aplicação em diversos momentos devido aos módulos adicionados e ao fato da nossa aplicação rodar através do Create React App que utiliza bibliotecas que precisam do Node.js como Webpack.

4.3.4 React

No desenvolvimento da aplicação foi escolhido o uso da Biblioteca de JavaScript React[7] para gerar componentes que podem ser usados para montar a interface do usuário, além de utilizar o conceito de estado para atualizar o conteúdo dos componentes de acordo com o que o usuário faz dentro da aplicação.

4.3.5 Create React App

Create React App [8] é um projeto do Facebook que permite criar uma aplicação web de página única, facilitando o desenvolvimento de uma aplicação, nessa aplicação é possível usar React para adicionar componentes e renderizar eles através do uso de React e JSX para criar uma aplicação que consegue modificar seu conteúdo dentro de uma única página.

Esse projeto foi utilizado para criar o molde inicial do projeto na parte do front-end e a partir disso foi desenvolvido o resto.

---

[7] **React A JavaScript library for building user interfaces**, React - Facebook, disponível em: <https://reactjs.org/>, acesso em: 9 set. 2021.

[8] **Create React App**, Facebook, disponível em: <https://create-react-app.dev/>, acesso em: 5 set. 2021.

5 PLANO DE DESENVOLVIMENTO DA APLICAÇÃO

5.1 Repositório

O repositório do projeto se encontra na seguinte URL:
https://github.com/AlexWFreitas/APS8

5.2 Entidades

O sistema vai possuir três entidades, sendo elas, User, Role e Report.

A entidade User representa um usuário, um usuário tem diversas Roles e Reports.

Roles representam as autorizações que o usuário tem dentro do sistema.

Um usuário com cargo de admin pode deletar usuários e reports por exemplo.

Um report representa um relato de incêndio, onde ele pode conter um título, descrição e localização do incêndio. Além disso ele possui campos usados para identificar quem criou o Report, além da data de criação do report.

5.3 Autenticação

Existe uma API para fazer tarefas de registro e login de usuários, essa ferramenta também gera JWT para serem usados como identidade dentro do sistema.

Além disso existe uma outra API que serve para validar a autenticação em relação as autorizações que o usuário tem, podendo retornar um erro caso não esteja autorizado ou uma mensagem indicando que o usuário está autorizado, que pode ser usada para fazer uma renderização da página condicional.

5.4 Banco de Dados

Como estamos utilizando um banco de dados do tipo H2 Database, nós optamos por incluir um arquivo Data.sql para inicializar os dados da aplicação para demonstração e debugging.

As tabelas são geradas automaticamente durante a inicialização através do Data Definition Language relacionando as classes definidas como entidade e as relações entre as entidades.

5.5 Front-end

O Front-end utiliza do Cliente HTTP Axios para fazer requisições com o back-end para requisitar dados que serão utilizados para gerar o conteúdo das páginas ou interagir com o sistema, além de poder ser utilizado para fazer autenticação e autorização de usuários.

As APIs também são utilizadas para fazer alterações no banco de dados, podendo então interagir com as partes internas do sistema através da interface disponibilizada para os usuários através do front.

Além disso é utilizado bibliotecas e frameworks como Bootstrap, React e Redux para gerar os componentes do site e gerenciamento da capacidade de store e state que o Redux e o React disponibilizam.

5.6 Back-end

O Back-end é feito em uma aplicação Spring Boot com a inclusão de diversos projetos do Spring que são utilizados para gerar Controllers, Autenticação, Criptografia, Repositórios, Entidades e interação com Banco de Dados através de ferramentas de ORM como o JPA / Hibernate.

O sistema utiliza entidades que estão armazenadas em uma parte relacionada aos Models. Dentro do models existem classes que são utilizadas para receber dados e enviar dados pelas APIs, chamados de DTO ou classes de Payload.

Existem vários Controllers do tipo RestController que são utilizados para disponibilizar APIs para interação com o sistema do back-end.

Foi implementada uma configuração de Spring Security com CORS para fazer criptografia de senhas, autenticação e autorização de usuários e autorização de rotas dentro do sistema.

Existem Repositórios que são usados para interagir com o banco de dados através dos Controllers.

Foi montado um Serviço no back-end que é utilizado para gerar tokens do tipo JWT, além de fazer a validação, sendo então disponibilizados para uso através de rotas de API disponibilizadas através de Controllers.

Houve bastante dificuldades em como fazer a conexão do back-end com o front-end no desenvolvimento desse projeto, mas que foram muito interessantes para perceber a evolução como um desenvolvedor.

6 RELATÓRIO COM LINHAS DE CÓDIGO

6.1 Entidade User

Entidade representando o Usuário, possui id, username, password, email e nome completo.

Além disso possui uma coleção de Roles através de uma relação de muitos para muitos.

Também possui uma relação de um para muitos com Reports, onde um usuário pode ter criado vários reports.

```java
@Entity
@Table( name = "users",
        uniqueConstraints = {
            @UniqueConstraint(columnNames = "username"),
            @UniqueConstraint(columnNames = "email")
        })
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotBlank
    @Size(max = 20)
    private String username;

    @NotBlank
    @Size(max = 50)
    @Email
    private String email;

    @NotBlank
    @Size(max = 100)
    private String fullName;

    @NotBlank
    @Size(max = 120)
    private String password;

    @ManyToMany(fetch = FetchType.LAZY)
    @JoinTable( name = "user_roles",
                joinColumns = @JoinColumn(name = "user_id"),
                inverseJoinColumns = @JoinColumn(name = "role_id"))
    private Set<Role> roles = new HashSet<>();
```

```
    @OneToMany(mappedBy="creator")
    private Set<Report> reports;

    public User() {
    }

    public User(String username, String email, String password, String
fullName) {
        this.username = username;
        this.email = email;
        this.password = password;
        this.fullName = fullName;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public Set<Role> getRoles() {
        return roles;
    }
```

```
    public void setRoles(Set<Role> roles) {
        this.roles = roles;
    }

    public String getFullName() {
        return this.fullName;
    }

    public void setFullName(String fullName) {
        this.fullName = fullName;
    }

}
```

6.2 ReportController

Controller utilizado para interagir com o Repositório de Reports, pode ser configurado para fornecer APIs para operações de CRUD sobre a Tabela Reports.

Utiliza da anotação de CrossOrigin para poder ser acessada por outros locais.

Possui uma anotação de RestController para configurar esse controller como um Controller do tipo REST estabelecendo o Controller como um fornecedor de APIs.

Possui injeção de repositórios do tipo User e Report através de injeção de dependências feita pelo Spring com o construtor do Controller.

```
@CrossOrigin(origins = "*")
@RestController
@RequestMapping("api/reports/")
public class ReportController {

    private final ReportRepository reportRepository;
    private final UserRepository userRepository;

    public ReportController(ReportRepository reportRepository, UserRepository
userRepository) {
        this.reportRepository = reportRepository;
        this.userRepository = userRepository;
    }

    @GetMapping
    @PreAuthorize("hasRole('USER') or hasRole('ADMIN')")
    public List<ReportDTO> GetAllReports() {
```

```
        var reports = reportRepository.findAll();

        List<ReportDTO> listReportDTO = new ArrayList<>();

        reports.forEach(report -> {
            var reportDTO = new ReportDTO(report.getId(),
report.getReportTitle(), report.getReportMessage(), report.getLocation(),
report.getCreateDate(), report.getCreator().getId(),
report.getCreator().getFullName());
            listReportDTO.add(reportDTO);
        });

        return listReportDTO;
    }

    @PostMapping
    @PreAuthorize("hasRole('USER') or hasRole('ADMIN')")
    public ResponseEntity<?> createReport(@Valid @RequestBody
CreateReportRequest createReportRequest) {

        // Retrieve User [ Creator ]
        var creator =
userRepository.findById(createReportRequest.getIdUser());

        // Create new Report
        Report report = new Report(
                                    createReportRequest.getReportTitle(),
                                    createReportRequest.getReportContent(),
                                    createReportRequest.getLocation(),
                                    creator.get());

        reportRepository.save(report);

        return ResponseEntity.ok(new MessageResponse("Report registered
successfully!"));

    }
}
```

6.3 ReportDTO

Representa um Data Transfer Object que é utilizado para mapear os dados a serem enviados pela API sem expor informações sensíveis do banco de dados.

Quando nós tentamos popular um objeto do Report através dos métodos do ReportRepository, ele estava trazendo dados do User junto incluindo sua senha.

Então a solução que pensamos foi utilizar DTOs para popular apenas com os dados que queremos e montar o DTO após usar o repositório para então só expor dados que queiramos intencionalmente.

```
package com.aps.webapp.models.DTO;

import java.time.LocalDateTime;

public class ReportDTO {

    private Long id;

    private String reportTitle;

    private String reportMessage;

    private String location;

    private LocalDateTime createdDate;

    private Long idUser;

    private String creatorName;

    public ReportDTO (Long id, String reportTitle, String reportMessage,
String location, LocalDateTime createdDate, Long idUser, String creatorName) {
        this.id = id;
        this.reportTitle = reportTitle;
        this.reportMessage = reportMessage;
        this.location = location;
        this.createdDate = createdDate;
        this.idUser = idUser;
        this.creatorName = creatorName;
    }

    public Long getId() {
        return this.id;
    }
```

```
public void setId(Long id) {
    this.id = id;
}

public String getReportTitle() {
    return this.reportTitle;
}

public void setReportTitle(String reportTitle) {
    this.reportTitle = reportTitle;
}

public String getReportMessage() {
    return this.reportMessage;
}

public void setReportMessage(String reportMessage) {
    this.reportMessage = reportMessage;
}

public String getLocation() {
    return this.location;
}

public void setLocation(String location) {
    this.location = location;
}

public LocalDateTime getCreatedDate() {
    return this.createdDate;
}

public void setCreatedDate(LocalDateTime createdDate) {
    this.createdDate = createdDate;
}

public Long getIdUser() {
    return this.idUser;
}

public void setIdUser(Long idUser) {
    this.idUser = idUser;
}

public String getCreatorName() {
    return this.creatorName;
}
```

```
    public void setCreatorName(String creatorName) {
        this.creatorName = creatorName;
    }

}
```

6.4 JWT Utils

Classe responsável por gerar tokens do tipo JWT, processar eles e também fazer a validação de Tokens.

Utilizada para tarefas relacionadas a autenticação e autorização de acesso.

```
package com.aps.webapp.security.jwt;

import java.util.Date;

import com.aps.webapp.security.services.UserDetailsImpl;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Component;

import io.jsonwebtoken.ExpiredJwtException;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.MalformedJwtException;
import io.jsonwebtoken.SignatureAlgorithm;
import io.jsonwebtoken.SignatureException;
import io.jsonwebtoken.UnsupportedJwtException;

@Component
public class JwtUtils {
    private static final Logger logger =
LoggerFactory.getLogger(JwtUtils.class);

    @Value("${aps.app.jwtSecret}")
    private String jwtSecret;

    @Value("${aps.app.jwtExpirationMs}")
    private int jwtExpirationMs;

    public String generateJwtToken(Authentication authentication) {
```

```
        UserDetailsImpl userPrincipal = (UserDetailsImpl)
authentication.getPrincipal();

        return Jwts.builder()
                .setSubject((userPrincipal.getUsername()))
                .setIssuedAt(new Date())
                .setExpiration(new Date((new Date()).getTime() +
jwtExpirationMs))
                .signWith(SignatureAlgorithm.HS512, jwtSecret)
                .compact();
    }

    public String getUserNameFromJwtToken(String token) {
        return
Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(token).getBody().getSubj
ect();
    }

    public boolean validateJwtToken(String authToken) {
        try {
            Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(authToken);
            return true;
        } catch (SignatureException e) {
            logger.error("Invalid JWT signature: {}", e.getMessage());
        } catch (MalformedJwtException e) {
            logger.error("Invalid JWT token: {}", e.getMessage());
        } catch (ExpiredJwtException e) {
            logger.error("JWT token is expired: {}", e.getMessage());
        } catch (UnsupportedJwtException e) {
            logger.error("JWT token is unsupported: {}", e.getMessage());
        } catch (IllegalArgumentException e) {
            logger.error("JWT claims string is empty: {}", e.getMessage());
        }

        return false;
    }
}
```

6.5 JwtResponse

Classe usada para popular um objeto de resposta para ser enviado como um JSON pela API, o front vai receber essa resposta e então montar um token no Local Storage para armazenar o JWT para propósitos de autenticação.

```
package com.aps.webapp.payload.response;

import java.util.List;

public class JwtResponse {
    private String token;
    private String type = "Bearer";
    private Long id;
    private String username;
    private String email;
    private List<String> roles;
    private String fullName;

    public JwtResponse(String accessToken, Long id, String username, String
email, String fullName, List<String> roles) {
        this.token = accessToken;
        this.id = id;
        this.username = username;
        this.email = email;
        this.fullName = fullName;
        this.roles = roles;
    }

    public String getAccessToken() {
        return token;
    }

    public void setAccessToken(String accessToken) {
        this.token = accessToken;
    }

    public String getTokenType() {
        return type;
    }

    public void setTokenType(String tokenType) {
        this.type = tokenType;
    }

    public Long getId() {
```

```
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public List<String> getRoles() {
        return roles;
    }

    public String getFullName() {
        return this.fullName;
    }

    public void setFullName(String fullName) {
        this.fullName = fullName;
    }


}
```

7 FICHAS DA APS

UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: ___Alex Wong de Freitas_____TURMA: _CC8P22_____RA:__N272BH2_____

CURSO:____Ciência da Computação_____CAMPUS:___Pinheiros_____SEMESTRE:___8º_____TURNO:___Noturno_____

CÓDIGO DA ATIVIDADE:_____53A6_____SEMESTRE:___2021/2_____ANO GRADE:_____2018/1_____

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
11/6/2021	Pesquisa sobre Sistemas Distribuídos	6	Alex Wong de Freitas		
11/7/2021	Escrita do Documento	6	Alex Wong de Freitas		
11/8/2021	Desenvolvimento do Back-End	6	Alex Wong de Freitas		
11/9/2021	Desenvolvimento do Front-End	6	Alex Wong de Freitas		
11/10/2021	Escrita do Documento	6	Alex Wong de Freitas		
11/11/2021	Desenvolvimento do Back-End	6	Alex Wong de Freitas		
11/12/2021	Desenvolvimento do Front-End	6	Alex Wong de Freitas		
11/13/2021	Escrita do Documento	6	Alex Wong de Freitas		
11/14/2021	Desenvolvimento do Back-End	6	Alex Wong de Freitas		
11/15/2021	Desenvolvimento do Front-End	6	Alex Wong de Freitas		
11/16/2021	Escrita do Documento	6	Alex Wong de Freitas		
11/17/2021	Desenvolvimento do Back-End	6	Alex Wong de Freitas		
11/18/2021	Desenvolvimento do Front-End	6	Alex Wong de Freitas		
11/19/2021	Finalização da APS	8	Alex Wong de Freitas		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS:_____

AVALIAÇÃO:_____
Aprovado ou Reprovado

NOTA:_____

DATA:____/_____/_____

_____

UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: CARLOS HENRIQUE GONÇALVES LEMES ALVES TURMA: CC8P22 RA: D654JJ4

CURSO: CIÊNCIA DA COMPUTAÇÃO CAMPUS: PINHEIROS SEMESTRE: 8 SEM TURNO: NOTURNO

CÓDIGO DA ATIVIDADE: SEMESTRE: 2021/2 ANO GRADE: 2021/2

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
5-Sep	Pesquisa de Temas	10	Carlos Alves		
15-Sep	Definição de Tema	4	Carlos Alves		
25-Sep	Definição de Aplicação Sobre Tema	6	Carlos Alves		
5-Oct	Definição de Técnologias	5	Carlos Alves		
15-Oct	Definição de Requisitos	4	Carlos Alves		
25-Oct	Definição de UI	5	Carlos Alves		
5-Nov	Pesquisa e Desenvolvimento de Documentação	10	Carlos Alves		
15-Nov	Desenvolvimento de FrontEnd	10	Carlos Alves		
18-Nov	Verificação Final para entrega	6	Carlos Alves		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 60

AVALIAÇÃO:_____
Aprovado ou Reprovado

NOTA:_____

DATA:____/_____/_____

_____
CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

8 REFERÊNCIAS

SECRETARIA DE SEGURANÇA PÚBLICA DE SÃO PAULO. **Dados Estatísticos do Corpo de Bombeiros do Estado de São Paulo**. Disponível em: <http://www.ssp.sp.gov.br/Estatistica/CorpoBombeiro.aspx>. Acesso em: 10 nov. 2021.

VMWARE. **Spring.io**. Disponível em: <https://spring.io/>. Acesso em: 5 set. 2021.

Create React App. Facebook. Disponível em: <https://create-react-app.dev/>. Acesso em: 5 set. 2021.

React A JavaScript library for building user interfaces. React - Facebook. Disponível em: <https://reactjs.org/>. Acesso em: 9 set. 2021.