

DESENVOLVIMENTO DE SOFTWARE PARA A WEB 1

Prof. Delano M. Beder (UFSCar)

Atividade AA-3: Sistema para oferta de vagas de estágios/empregos

Obs 1: Essa atividade deve ser baseada na atividade AA-2. Ou seja, deve-se apenas implementar os novos requisitos (funcionalidades providas em uma REST API) aqui mencionados -- levando em consideração o que já foi desenvolvido na atividade AA-2.

O sistema deve incorporar os seguintes requisitos:

- REST API -- CRUD ¹ de profissionais
 - Cria um novo profissional [Create - **CRUD**]
POST <http://localhost:8080/api/profissionais>
Body: raw/JSON (application/json)
 - Retorna a lista de profissionais [Read - **CRUD**]
GET <http://localhost:8080/api/profissionais>
 - Retorna o profissional de id = {id} [Read - **CRUD**]
GET <http://localhost:8080/api/profissionais/{id}>
 - Atualiza o profissional de id = {id} [Update - **CRUD**]
PUT <http://localhost:8080/api/profissionais/{id}>
Body: raw/JSON (application/json)
 - Remove o profissional de id = {id} [Delete - **CRUD**]
DELETE <http://localhost:8080/api/profissionais/{id}>
- REST API -- CRUD de empresas
 - Cria uma nova empresa [Create - **CRUD**]
POST <http://localhost:8080/api/empresas>
Body: raw/JSON (application/json)
 - Retorna a lista de empresas [Read - **CRUD**]
GET <http://localhost:8080/api/empresas>
 - Retorna a empresa de id = {id} [Read - **CRUD**]
GET <http://localhost:8080/api/empresas/{id}>
 - Retorna a lista de todas as empresas da cidade de nome = {nome}
GET <http://localhost:8080/api/empresas/cidades/{nome}>
 - Atualiza a empresa de id = {id} [Update - **CRUD**]
PUT <http://localhost:8080/api/empresas/{id}>
Body: raw/JSON (application/json)
 - Remove a empresa de id = {id} [Delete - **CRUD**]
DELETE <http://localhost:8080/api/empresas/{id}>

- REST API -- Retorna a lista de vagas [Read - **CRUD**]
GET <http://localhost:8080/api/vagas>
- REST API -- Retorna a vaga de id = {id} [Read - **CRUD**]
GET <http://localhost:8080/api/vagas/{id}>
- REST API -- Retorna a lista de vagas (em aberto) da empresa de id = {id} [Read - **CRUD**]
GET <http://localhost:8080/api/vagas/empresas/{id}>

Obs 2: Em todas as funcionalidades mencionadas acima, não há necessidade de autenticação (login)

Dica: Na configuração do *Spring Security* utilize algo semelhante ao apresentado no código abaixo:

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http.authorizeHttpRequests((authz) -> authz
        // linhas anteriores
        .requestMatchers("/api/**").permitAll()
        .anyRequest().authenticated())
        .csrf(AbstractHttpConfigurer::disable)
        .formLogin((form) -> form
            .loginPage("/login")
            .permitAll())
        .logout((logout) -> logout
            .logoutSuccessUrl("/").permitAll());

    return http.build();
}
```

Arquitetura: Modelo-Visão-Controlador

Tecnologias

- Spring MVC (Controladores REST), Spring Data JPA, Spring Security & Thymeleaf (Lado Servidor)

Ambiente de Desenvolvimento

- A compilação e o *deployment* deve ser obrigatoriamente ser realizado via *maven*.
- Os arquivos fonte do sistema devem estar hospedados obrigatoriamente em um repositório (preferencialmente github).