

DESENVOLVIMENTO DE SOFTWARE PARA A WEB 1

Profs. Alan D.B. Valejo & Delano M. Beder (UFSCar)

Atividade AA-3: Sistema para compra/venda de pacotes turísticos

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 clientes
 - Cria um novo cliente [Create - **CRUD**]
POST <http://localhost:8080/clientes>
Body: raw/JSON (application/json)
 - Retorna a lista de clientes [Read - **CRUD**]
GET <http://localhost:8080/clientes>
 - Retorna o cliente de id = {id} [Read - **CRUD**]
GET <http://localhost:8080/clientes/{id}>
 - Atualiza o cliente de id = {id} [Update - **CRUD**]
PUT <http://localhost:8080/clientes/{id}>
Body: raw/JSON (application/json)
 - Remove o cliente de id = {id} [Delete - **CRUD**]
DELETE <http://localhost:8080/clientes/{id}>
- REST API -- CRUD de agências de turismo
 - Cria uma nova agência de turismo [Create - **CRUD**]
POST <http://localhost:8080/agencias>
Body: raw/JSON (application/json)
 - Retorna a lista de agências de turismo [Read - **CRUD**]
GET <http://localhost:8080/agencias>
 - Retorna a agência de turismo de id = {id} [Read - **CRUD**]
GET <http://localhost:8080/agencias/{id}>
 - Atualiza a agência de turismo de id = {id} [Update - **CRUD**]
PUT <http://localhost:8080/agencias/{id}>
Body: raw/JSON (application/json)
 - Remove a agência de turismo de id = {id} [Delete - **CRUD**]
DELETE <http://localhost:8080/agencias/{id}>

- REST API -- Retorna a lista de pacotes turísticos [Read - **CRUD**]
GET <http://localhost:8080/pacotes>
- REST API -- Retorna a lista de pacotes turísticos do cliente de id = {id} [Read - **CRUD**]
GET <http://localhost:8080/pacotes/clientes/{id}>
- REST API -- Cria um novo pacote turístico na agência de id = {id} [Create - **CRUD**]
POST <http://localhost:8080/pacotes/agencias/{id}>
Body: raw/JSON (application/json)
- REST API -- Retorna a lista de pacotes turísticos da agência de id = {id} [Read - **CRUD**]
GET <http://localhost:8080/pacotes/agencias/{id}>
- REST API -- Retorna a lista de pacotes turísticos cujo destino = {nome} [Read - **CRUD**]
GET <http://localhost:8080/pacotes/destinos/{nome}>

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:

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.csrf().disable().authorizeRequests()
        // Controladores REST
        .antMatchers("/clientes", "/agencias", "/pacotes").permitAll()
        .antMatchers("/clientes/{\\d+}", "/agencias/{\\d+}").permitAll()
        .antMatchers("/pacotes/agencias/{\\d+}").permitAll()
        .antMatchers("/pacotes/clientes/{\\d+}").permitAll()
        .antMatchers("/pacotes/destinos/{\\w+}").permitAll()
        // Demais linhas
        .anyRequest().authenticated()
        .and().formLogin().loginPage("/login").permitAll()
        .and().logout().logoutSuccessUrl("/").permitAll();
}
```

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).