DESENVOLVIMENTO DE SOFTWARE PARA A WEB 1 **Profs. Alan D. B. Valejo & Delano M. Beder (UFSCar)**

Atividade A-3: Sistema para locação de bicicletas

Obs 1: Essa atividade deve ser baseada na atividade A-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 A-2.

O sistema deve incorporar os seguintes requisitos:

- REST API -- CRUD[1] de clientes
    - Cria um novo cliente [**C**reate - **CRUD**]

      POST http://localhost:8080/clientes

      Body: raw/JSON (application/json)
    - Retorna a lista de clientes [**R**ead - **CRUD**]

      GET http://localhost:8080/clientes
    - Retorna o cliente de id = {id} [**R**ead - **CRUD**]

      GET http://localhost:8080/clientes/{id}
    - Atualiza o cliente de id = {id} [**U**pdate - **CRUD**]

      PUT http://localhost:8080/clientes/{id}

      Body: raw/JSON (application/json)
    - Remove o cliente de id = {id} [**D**elete - **CRUD**]

      DELETE http://localhost:8080/clientes/{id}
- REST API -- CRUD de locadoras
    - Cria uma nova locadora [**C**reate - **CRUD**]

      POST http://localhost:8080/locadoras

      Body: raw/JSON (application/json)
    - Retorna a lista de locadoras [**R**ead - **CRUD**]

      GET http://localhost:8080/locadoras
    - Retorna a locadora de id = {id} [**R**ead - **CRUD**]

GET http://localhost:8080/locadoras/{id}

- Retorna a lista de todas as locadoras da cidade de nome = {nome}

  GET http://localhost:8080/locadoras/cidades/{nome}

- Atualiza a locadora de id = {id} [**U**pdate - **CRUD**]

  PUT http://localhost:8080/locadoras/{id}

  Body: raw/JSON (application/json)

- Remove a locadora de id = {id} [**D**elete - **CRUD**]

  DELETE http://localhost:8080/locadoras/{id}

- REST API -- Retorna a lista de locações [**R**ead - **CRUD**]

  GET http://localhost:8080/locacoes

- REST API -- Retorna a locação de id = {id} [**R**ead - **CRUD**]

  GET http://localhost:8080/locacoes/{id}

- REST API -- Retorna a lista das locações do cliente de id = {id} [**R**ead - **CRUD**]

  GET http://localhost:8080/locacoes/clientes/{id}

- REST API -- Retorna a lista de locações da locadora de id = {id} [**R**ead - **CRUD**]

  GET http://localhost:8080/locacoes/locadoras/{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:

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.csrf().disable().authorizeRequests()
    // Controladores REST
    .antMatchers("/clientes", "/locadoras", "/locacoes").permitAll()
    .antMatchers("/clientes/{\\d+}", "/locadoras/{\\d+}").permitAll()
    .antMatchers("/locacoes/{\\d+}").permitAll()
    .antMatchers("/locadoras/cidades/{\\w+}").permitAll()
    .antMatchers("/locacoes/clientes/{\\d+}").permitAll()
    .antMatchers("/locacoes/locadoras/{\\d+}").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 compilaçao e o *deployment* deve ser obrigatoriamente ser realizado via *maven*.

- Os arquivos fonte do sistema devem estar hospedados obrigatoriamente em um repositório (preferencialmente github).

1. CRUD: **C**reate, **R**ead, **U**pdate & **D**elete. ↩