

# DESENVOLVIMENTO DE SOFTWARE PARA A WEB 1

---

Prof. Delano M. Beder (UFSCar)

---

## Atividade AA-3: Sistema para locação de bicicletas

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 <sup>1</sup> 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 locadoras
  - Cria uma nova locadora [Create - **CRUD**]  
POST <http://localhost:8080/locadoras>  
Body: raw/JSON (application/json)
  - Retorna a lista de locadoras [Read - **CRUD**]  
GET <http://localhost:8080/locadoras>
  - Retorna a locadora de id = {id} [Read - **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} [Update - **CRUD**]  
PUT <http://localhost:8080/locadoras/{id}>  
Body: raw/JSON (application/json)
  - Remove a locadora de id = {id} [Delete - **CRUD**]  
DELETE <http://localhost:8080/locadoras/{id}>

- REST API -- Retorna a lista de locações [Read - **CRUD**]  
GET <http://localhost:8080/locacoes>
- REST API -- Retorna a locação de id = {id} [Read - **CRUD**]  
GET <http://localhost:8080/locacoes/{id}>
- REST API -- Retorna a lista das locações do cliente de id = {id} [Read - **CRUD**]  
GET <http://localhost:8080/locacoes/clientes/{id}>
- REST API -- Retorna a lista de locações da locadora de id = {id} [Read - **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 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).