DESENVOLVIMENTO DE SOFTWARE PARA A WEB 1

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

Atividade AA-3: Sistema para compra/venda de veículos

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 [**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 lojas
  - Cria uma nova loja [**C**reate - **CRUD**]

    POST http://localhost:8080/lojas

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

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

    GET http://localhost:8080/lojas/{id}
  - Atualiza a loja de id = {id} [**U**pdate - **CRUD**]

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

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

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

- REST API -- Retorna a lista de propostas de compra do veículo de id = {id} [**R**ead - **CRUD**]

  GET http://localhost:8080/propostas/veiculos/{id}
- REST API -- Retorna a lista das propostas do cliente de id = {id} [**R**ead - **CRUD**]

  GET http://localhost:8080/propostas/clientes/{id}
- REST API -- Cria um novo veículo na loja de id = {id} [**C**reate - **CRUD**]

  POST http://localhost:8080/veiculos/lojas/{id}

  Body: raw/JSON (application/json)
- REST API -- Retorna a lista de veículos da loja de id = {id} [**R**ead - **CRUD**]

  GET http://localhost:8080/veiculos/lojas/{id}
- REST API -- Retorna a lista de veículos de modelo cujo nome = {nome} [**R**ead - **CRUD**]

  GET http://localhost:8080/veiculos/modelos/{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", "/lojas").permitAll()
    .antMatchers("/clientes/{\\d+}", "/lojas/{\\d+}").permitAll()
    .antMatchers("/propostas/veiculos/{\\d+}").permitAll()
    .antMatchers("/propostas/clientes/{\\d+}").permitAll()
    .antMatchers("/veiculos/lojas/{\\d+}").permitAll()
    .antMatchers("/veiculos/modelos/{\\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 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. ↩