

RESTful Webservices (parte 2 - Prática)

Workshop TQI – JULHO/2014

Alexandre Fidélis Vieira Bitencourt

<https://github.com/alexandrefvb/restful-webservices-workshop>



Agenda

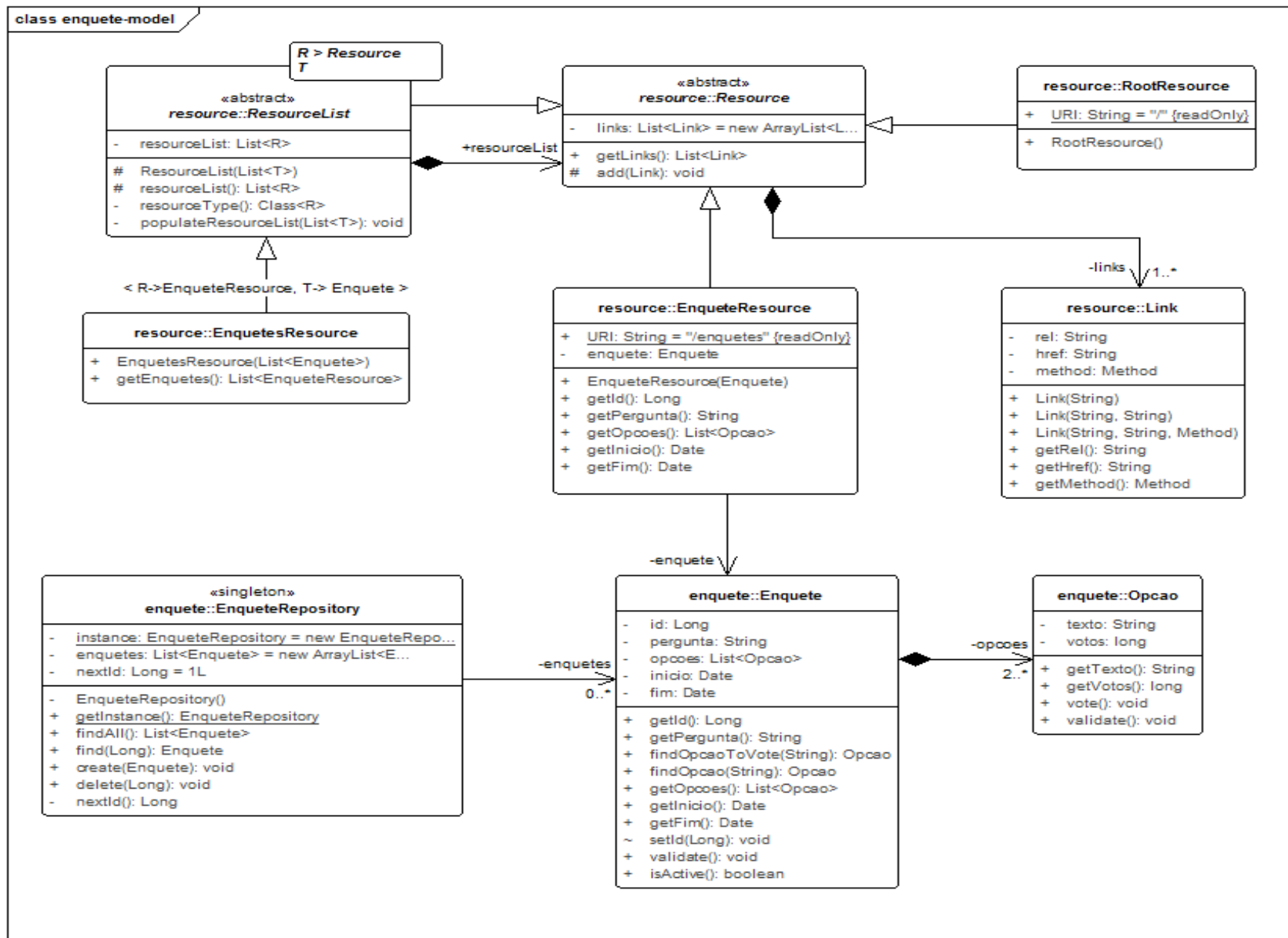
- Problema proposto: Enquetes
- Solução do problema (model)
- Especificação API REST
- Implementação
 - JAX-RS
 - Spring MVC



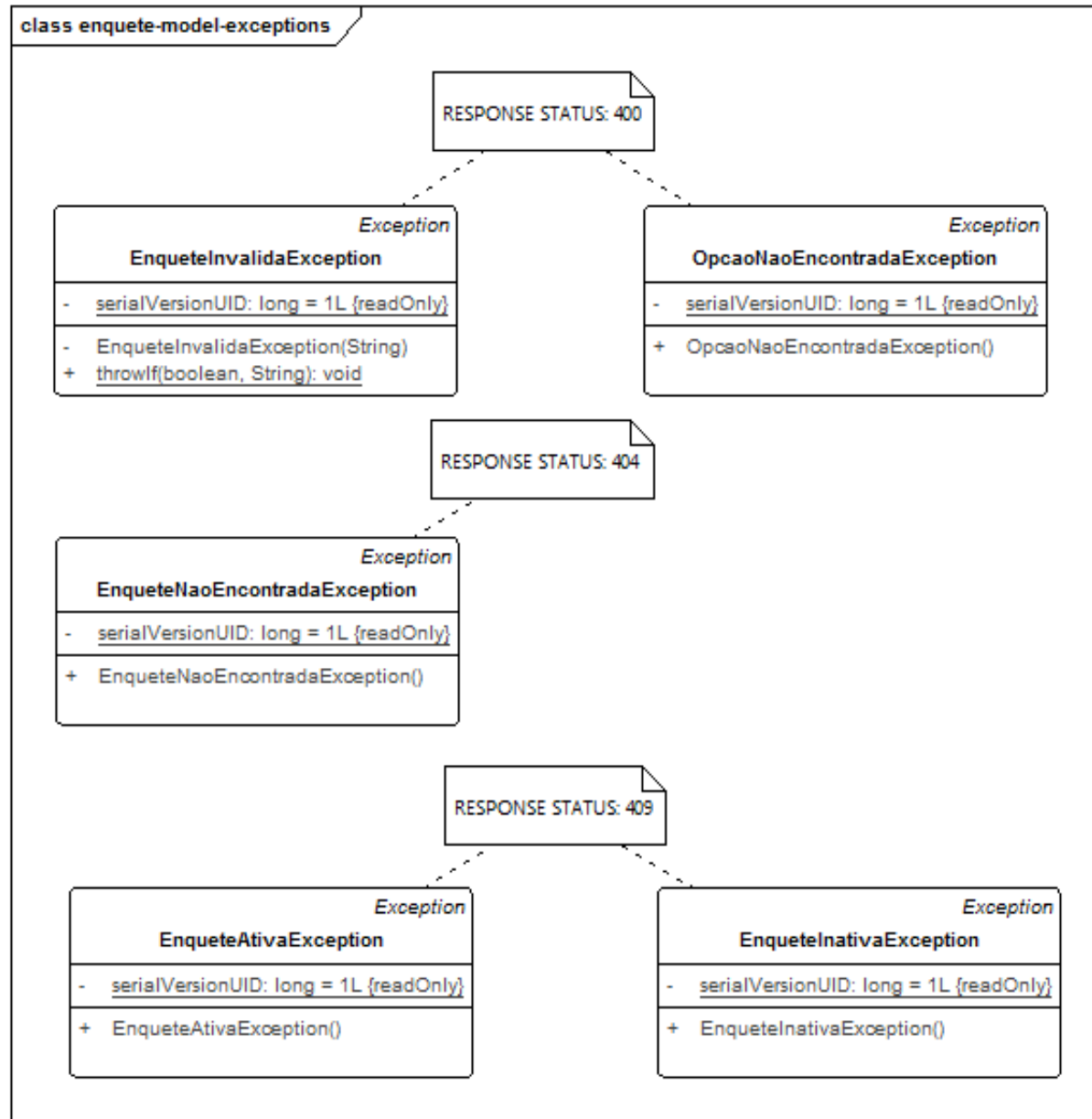
Problema proposto: Enquetes

- Para ilustrar a criação de uma API RESTful vamos criar serviços responsáveis por expor a funcionalidade de enquetes para os clientes.
- Uma enquete consiste em uma pergunta com um conjunto de opções que podem ser votadas pelos clientes. Uma enquete possui uma data de início e uma data de fim.
- A API deve permitir as seguintes operações:
 - Criar uma nova enquete.
 - Listar as enquetes existentes.
 - Apagar uma enquete (caso não esteja ativa).
 - Consultar o resultado de uma enquete.
 - Votar em uma opção da enquete (caso esteja ativa).

Solução do problema (model)



Solução do problema (model)



Especificação API REST

- A primeira coisa a ser feita é identificar os resources, suas URIs e operações aceitas:
- Temos os seguintes resources e operações:
 - Raiz (URI /)
 - GET – Obtém a representação da raiz com os links para as operações disponíveis. (HATEOAS)
 - Enquetes (URI /enquetes)
 - GET – Obtém a representação das enquetes cadastradas.
 - POST – Cria uma nova enquete enviando uma representação da enquete a ser criada.
 - Enquete (URI /enquetes/{id})
 - GET – Obtém a representação da enquete identificada pela URI.
 - DELETE – Remove a enquete identificada pela URI.
 - Voto em uma enquete (URI /enquetes/{id}/voto)
 - POST – Vota na opção cuja representação é enviada no corpo da requisição.

Especificação API REST

- Status e retorno das requisições:

- GET /enquetes

- Sempre retorna status 200 com a representação da lista de enquetes disponíveis no atributo enquetes.

- POST /enquetes

- Retorna 201 com a representação da enquete criada incluída no corpo da resposta.
 - Retorna 400 caso algum dado da requisição esteja inválido.

- GET /enquetes/{id}

- Retorna 200 com a representação da enquete correspondente ao id informado na URI.
 - Retorna 404 caso não exista uma enquete com o id informado.

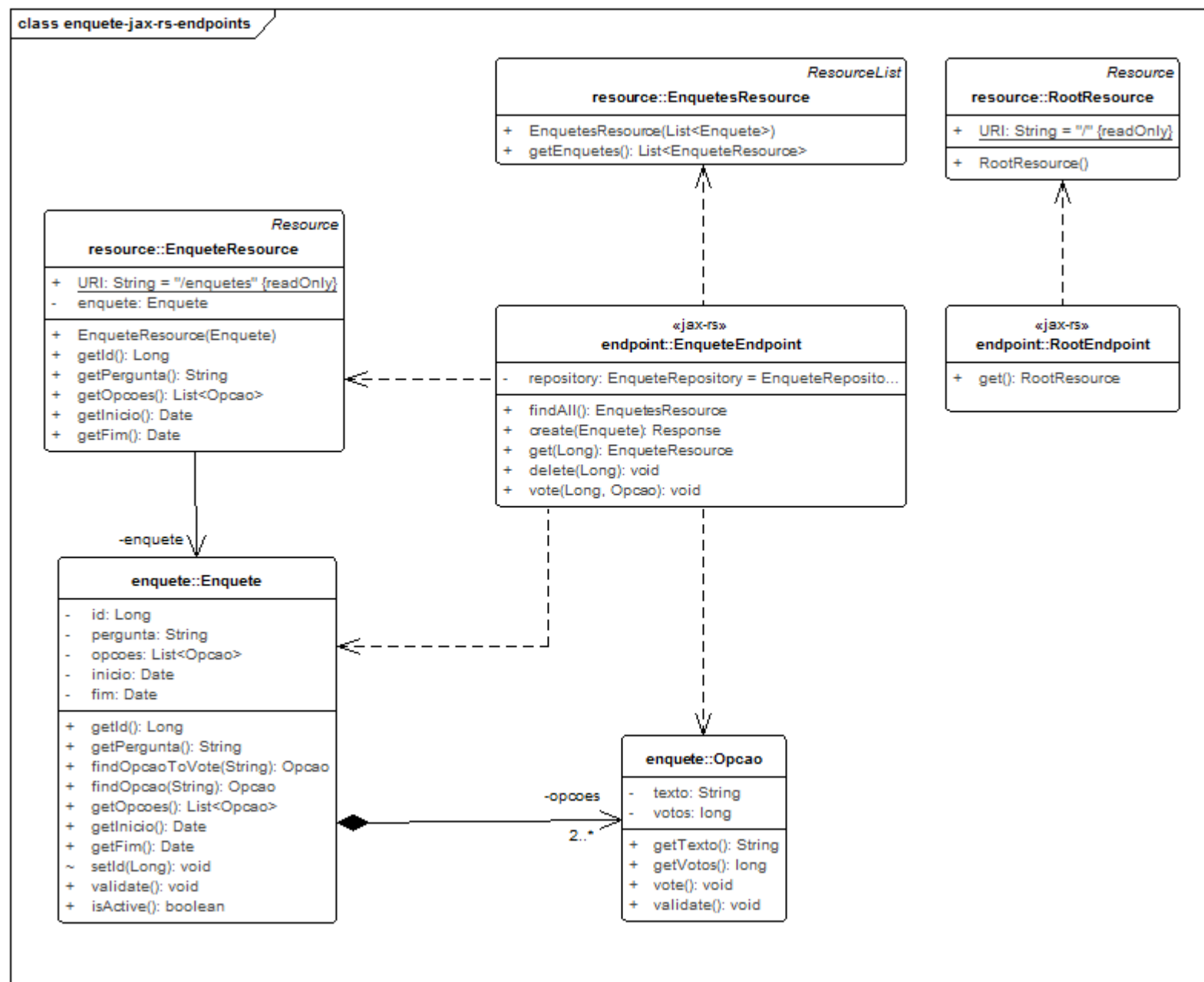
- DELETE /enquetes/{id}

- Retorna 204 caso a enquete seja removida com sucesso.
 - Retorna 404 caso a enquete com o id informado não exista.
 - Retorna 409 caso a enquete esteja ativa.

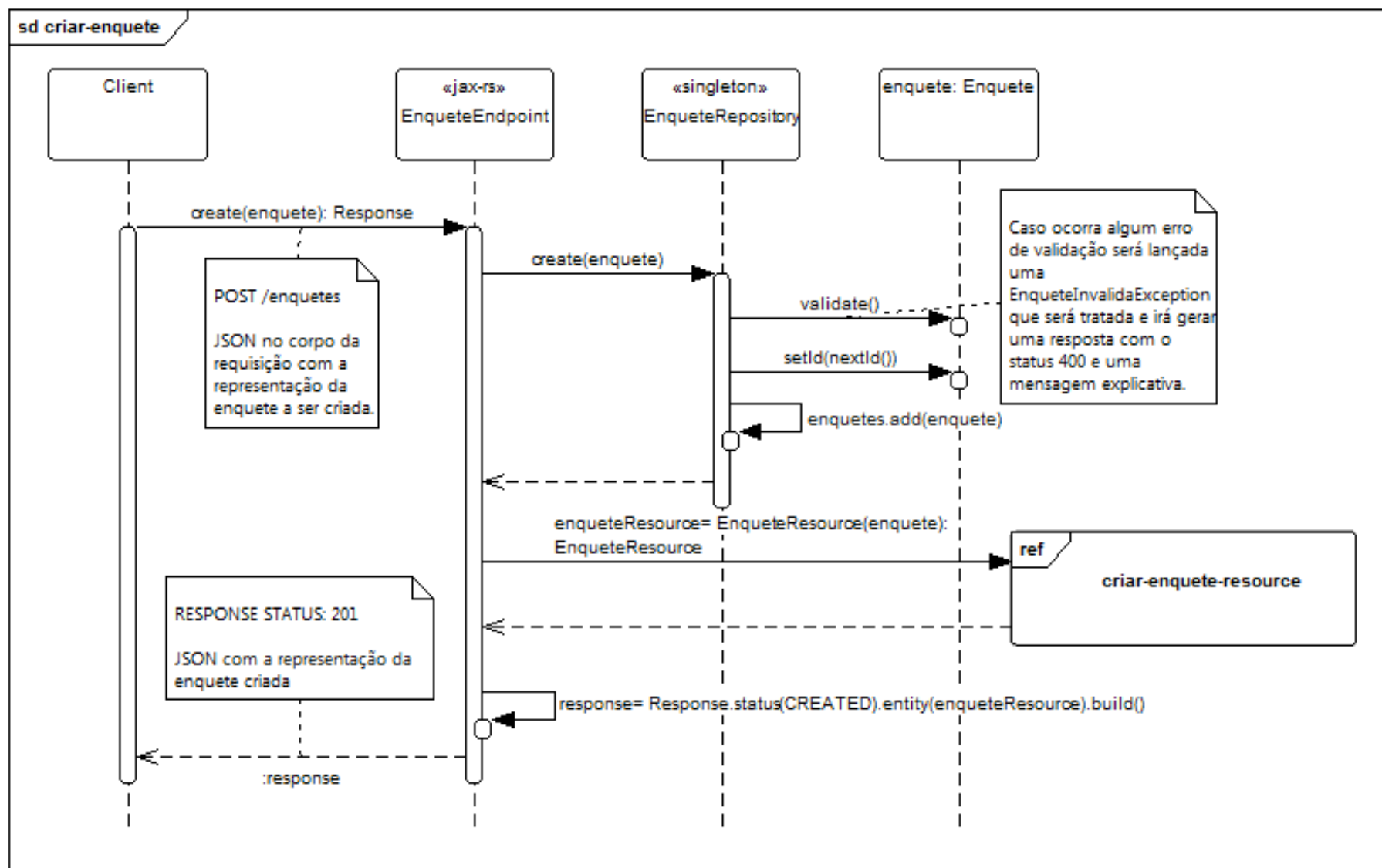
- POST /enquetes/{id}/voto

- Retorna 204 caso o voto na opção informada seja contabilizado.
 - Retorna 404 caso não seja encontrada enquete com o id informado.
 - Retorna 400 caso não seja encontrada a opção informada na enquete com o id informado.
 - Retorna 409 caso a enquete não esteja ativa.

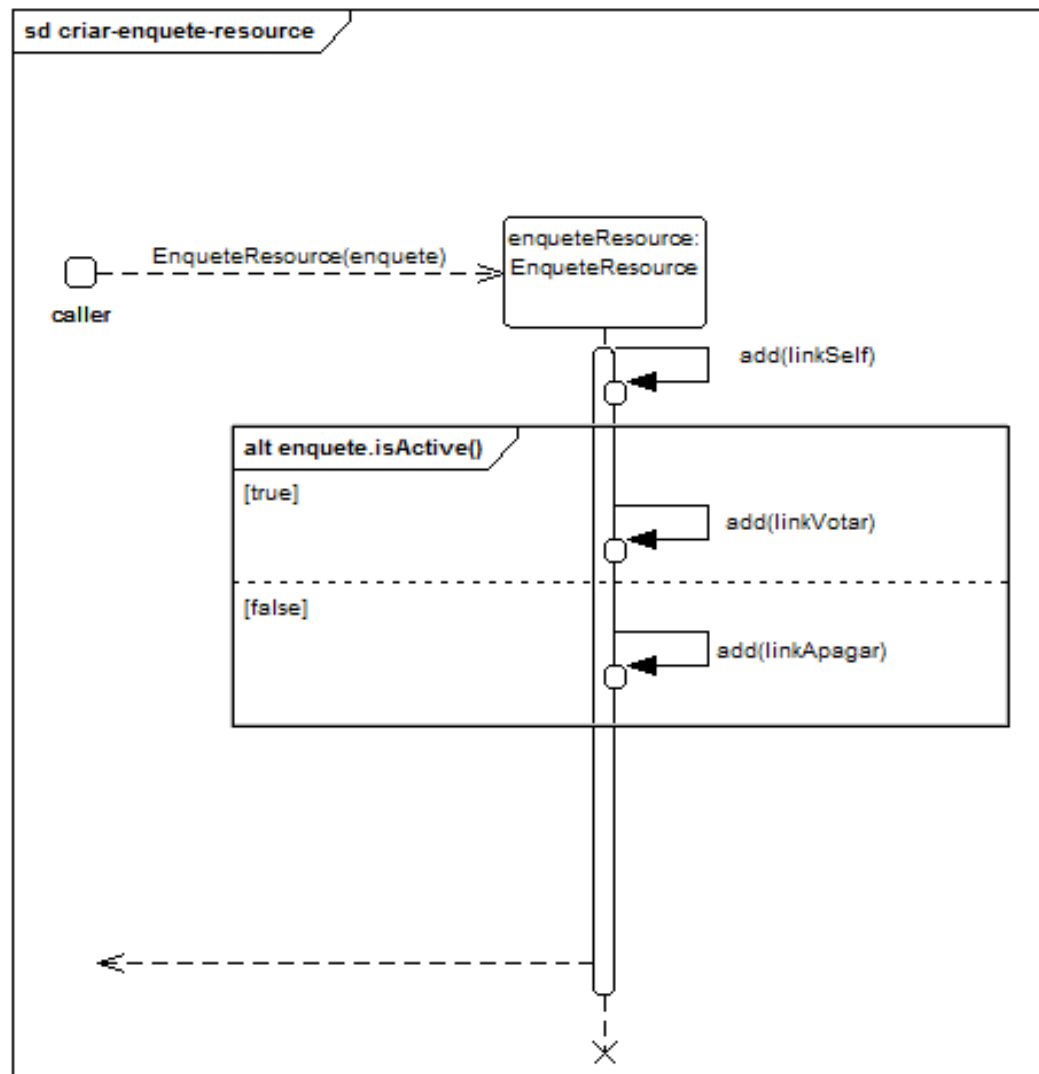
Implementação - classes



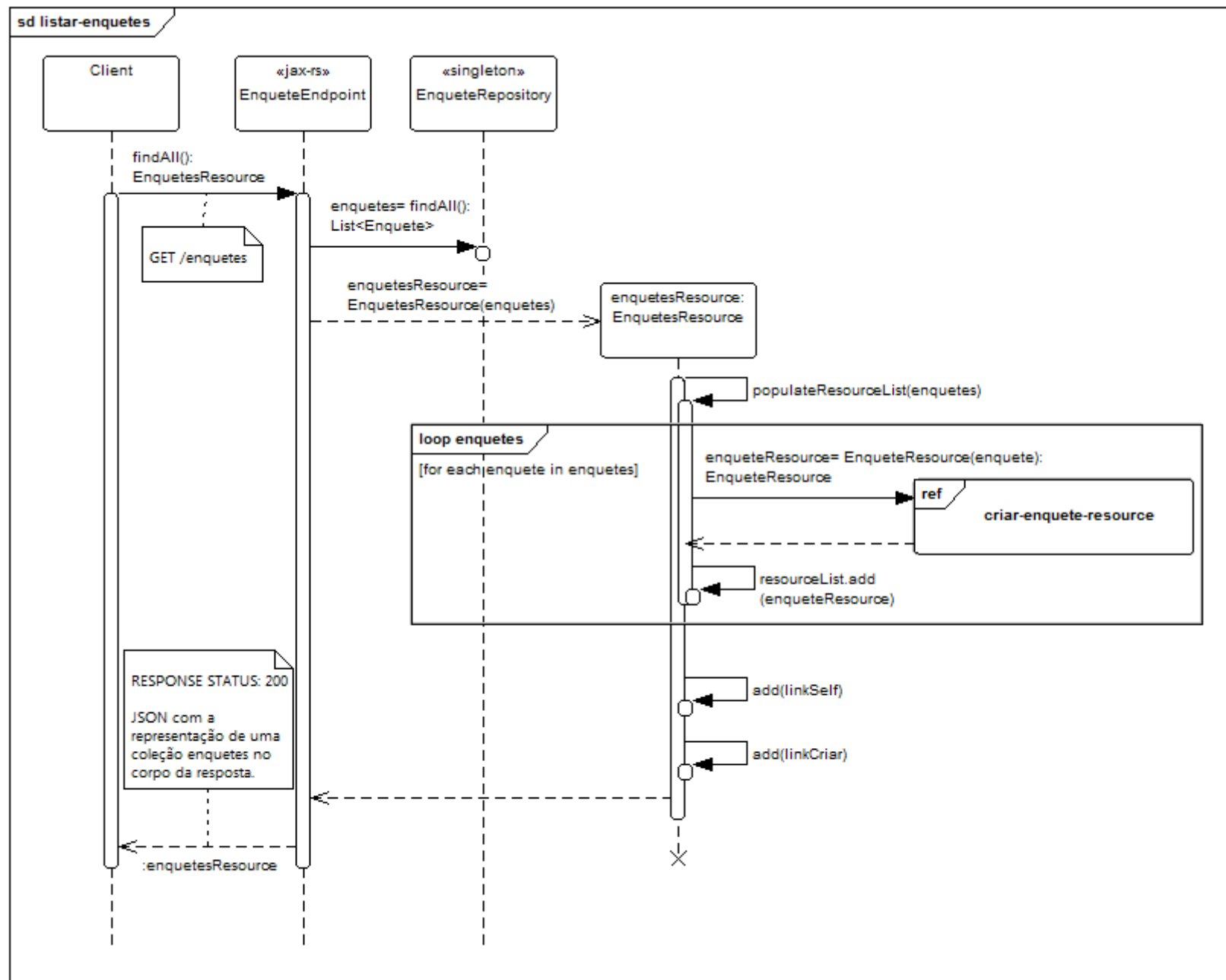
Implementação – criar enquete



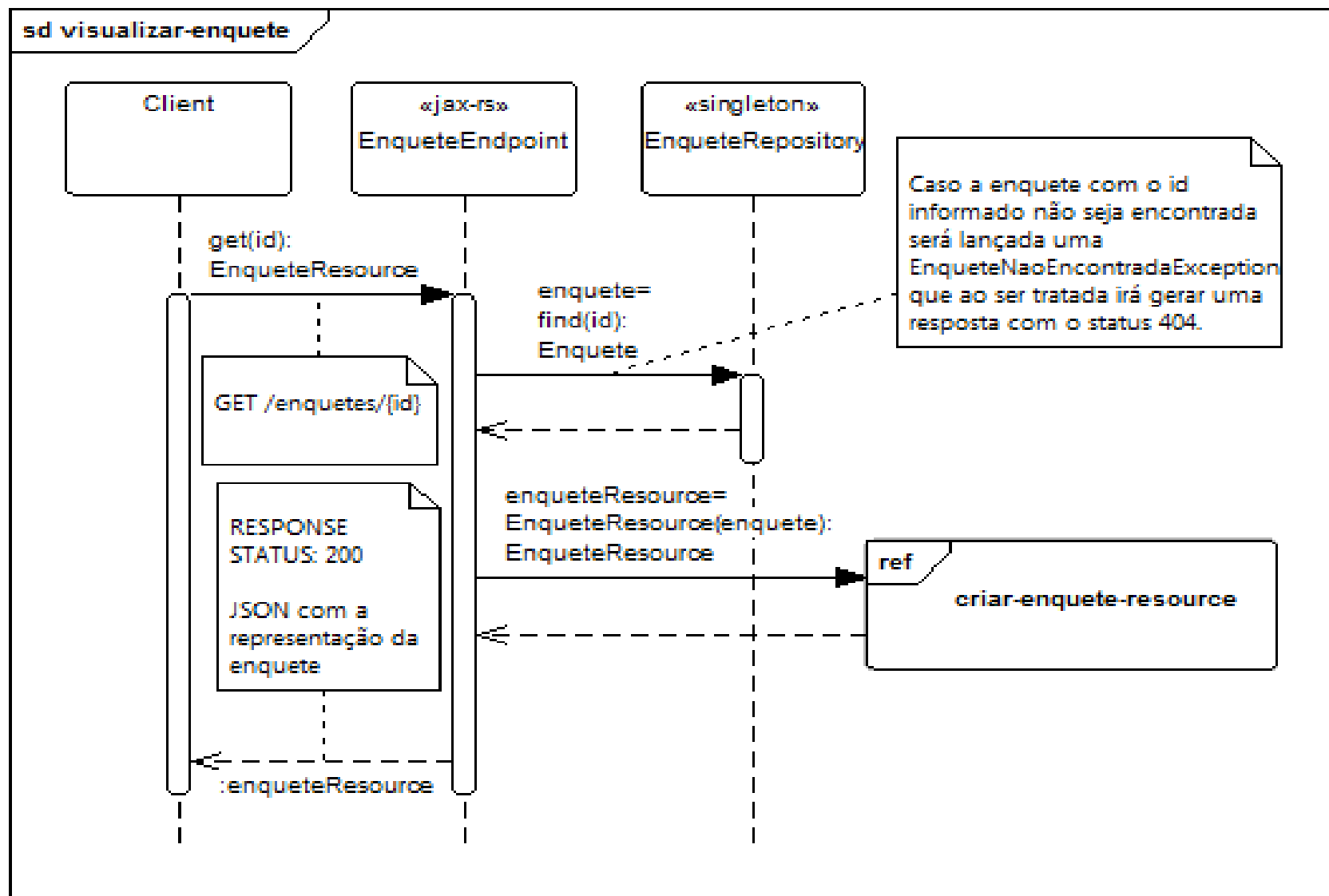
Implementação – criar enquete resource



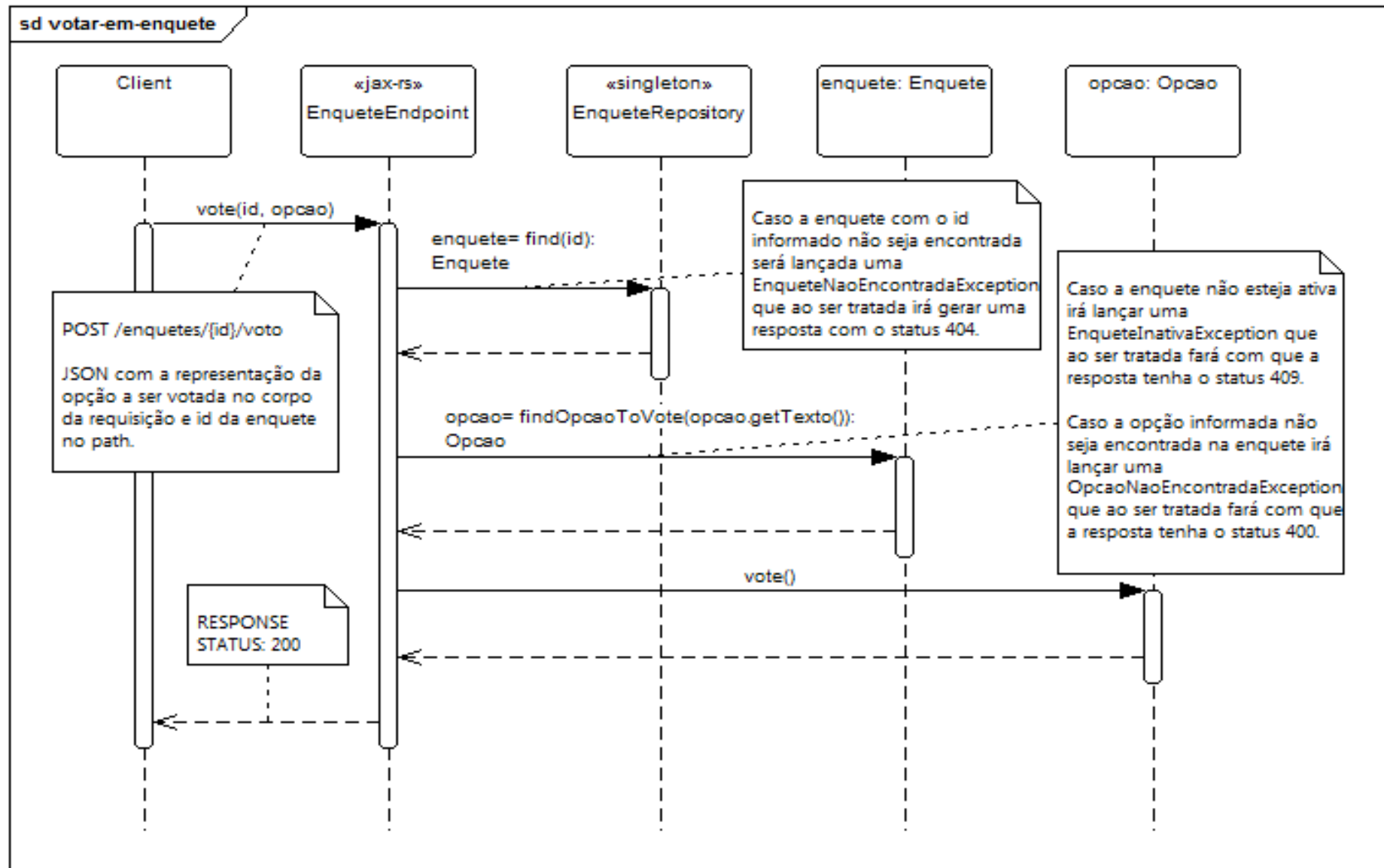
Implementação - listar enquetes



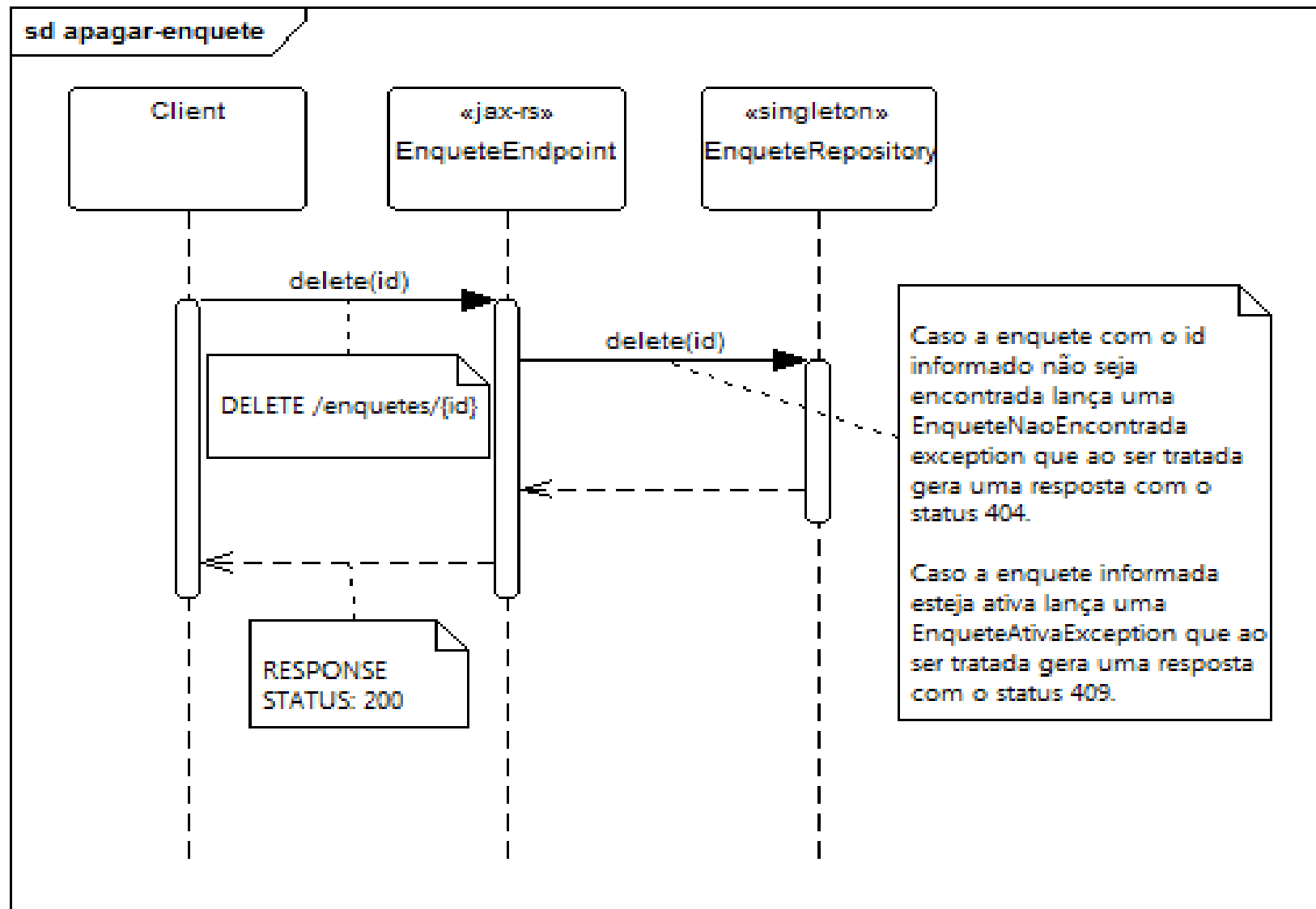
Implementação – visualizar enquete



Implementação – votar em enquete



Implementação – apagar enquete



Implementação JAX-RS

- O JAX-RS é a especificação padrão do Java EE para a criação de webservices REST.
- Baseado em annotations permite que um Objeto seja mapeado para receber em seus métodos as requisições HTTP provenientes dos clientes.
- Possui suporte a JSON, XML e outros formatos através de Providers. Para a nossa implementação faremos uso do provider Jackson que cuida da transformação de objetos java em json e vice-versa.
- A implementação de referência do JAX-RS (Jersey) foi utilizada no exemplo.
- O módulo enquete-jax-rs contém a implementação JAX-RS dos serviços especificados.
- O módulo enquete-model contém o domain model da aplicação que será reutilizado também na implementação Spring MVC.

Implementação JAX-RS - maven

...

```
<dependency>
```

```
  <groupId>org.glassfish.jersey.containers</groupId>
```

```
  <artifactId>jersey-container-servlet</artifactId>
```

```
  <version>2.10.1</version>
```

```
</dependency>
```

```
<dependency>
```

```
  <groupId>org.glassfish.jersey.media</groupId>
```

```
  <artifactId>jersey-media-json-jackson</artifactId>
```

```
  <version>2.10.1</version>
```

```
</dependency>
```

...

Implementação JAX-RS – web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
  <servlet>
    <servlet-name>jersey</servlet-name>
    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
    <init-param>
      <param-name>jersey.config.server.provider.packages</param-name>
      <param-value>br.com.tqi.enquete</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>jersey</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Implementação JAX-RS – configuração jackson

@Provider

```
public class ObjectMapperContextResolver implements
    ContextResolver<ObjectMapper> {
    private ObjectMapper objectMapper;
    public ObjectMapperContextResolver() {
        this.objectMapper = new ObjectMapper();
        this.objectMapper.configure(
            SerializationFeature.WRITE_DATES_AS_TIMESTAMPS, false);
        this.objectMapper.setDateFormat(new SimpleDateFormat(
            "dd/MM/yyyy HH:mm:ss"));
    }
    @Override
    public ObjectMapper getContext(Class<?> objectType) {
        return this.objectMapper;
    }
}
```

Implementação JAX-RS – criar enquete

```
@Path(EnqueteResource.URI)  
public class EnqueteEndpoint {  
    ...  
    @POST  
    @Consumes(MediaType.APPLICATION_JSON)  
    @Produces(MediaType.APPLICATION_JSON)  
    public Response create(Enquete enquete) throws EnqueteInvalidaException {  
        this.repository.create(enquete);  
        return Response.status(Status.CREATED).entity(new EnqueteResource(enquete)).build();  
    }  
    ...  
}
```

Implementação JAX-RS – listar enquetes

```
@Path(EnqueteResource.URI)  
public class EnqueteEndpoint {  
    ...  
    @GET  
    @Produces(MediaType.APPLICATION_JSON)  
    public EnquetesResource findAll() {  
        return new EnquetesResource(this.repository.findAll());  
    }  
    ...  
}
```

Implementação JAX-RS – visualizar enquete


```
@Path(EnqueteResource.URI)
public class EnqueteEndpoint {
    ...
    @GET
    @Path("{id}")
    @Produces(MediaType.APPLICATION_JSON)
    public EnqueteResource get(@PathParam("id") Long id)
        throws EnqueteNaoEncontradaException {
        return new EnqueteResource(this.repository.find(id));
    }
    ...
}
```

Implementação JAX-RS – apagar enquete

```
@Path(EnqueteResource.URI)  
public class EnqueteEndpoint {  
    ...  
    @DELETE  
    @Path("{id}")  
    public void delete(@PathParam("{id") Long id)  
        throws EnqueteNaoEncontradaException, EnqueteAtivaException {  
        this.repository.delete(id);  
    }  
    ...  
}
```

Implementação JAX-RS – votar em enquete

```
@Path(EnqueteResource.URI)
public class EnqueteEndpoint {
    ...
    @POST
    @Path("{id}/voto")
    @Consumes(MediaType.APPLICATION_JSON)
    public void vote(@PathParam("id") Long id, Opcao opcao)
        throws EnqueteNaoEncontradaException, EnqueteInativaException,
            OpcaoNaoEncontradaException {
        this.repository.find(id).findOpcaoToVote(opcao.getTexto()).vote();
    }
    ...
}
```



Implementação JAX-RS – Tratamento de Exceptions

- As condições de erro são tratadas através de exceptions.
- A classe `br.com.tqi.enquete.provider.EnqueteExceptionHandler` é responsável por tratar as exceptions e converter o status resposta no status esperado.
- O tratador de exceções deve implementar a interface `javax.ws.rs.ext.ExceptionMapper<T>` especificando o tipo da exception a ser tratada.
- No caso de enquetes todas as exceptions são tratadas pelo mesmo tratador que cria uma resposta contendo a mensagem de erro da Exception e o status correspondente.

Implementação Spring MVC

- O Spring MVC é um módulo do Spring Framework responsável pela implementação do padrão MVC para aplicações web.
- Assim como o JAX-RS possui annotations que mapeiam as requisições web para métodos de um controller (Endpoint).
- Possui suporte a JSON, XML e outros formatos através de MessageConverters responsáveis por converter objetos java em representações. Assim como no JAX-RS vamos utilizar o Jackson para criação das representações.
- O módulo enquete-spring-mvc contém a implementação Spring MVC dos serviços especificados.
- O módulo enquete-model é o mesmo utilizado no exemplo JAX-RS.

Implementação Spring MVC - maven

...

```
<dependency>
```

```
    <groupId>org.springframework</groupId>
```

```
    <artifactId>spring-webmvc</artifactId>
```

```
    <version>4.0.5.RELEASE</version>
```

```
</dependency>
```

```
<dependency>
```

```
    <groupId>com.fasterxml.jackson.core</groupId>
```

```
    <artifactId>jackson-databind</artifactId>
```

```
    <version>2.4.1.1</version>
```

```
</dependency>
```

...

Implementação Spring MVC – web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
<servlet>
    <servlet-name>springmvc</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet
</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>springmvc</servlet-name>
    <url-pattern>/*</url-pattern>
</servlet-mapping>

</web-app>
```

Implementação Spring MVC – spring-servlet.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-
4.0.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-
context-4.0.xsd">

<context:component-scan base-package="br.com.tqi.enquete" />

</beans>
```

Implementação Spring MVC – configuração jackson

@Configuration

@EnableWebMvc

```
public class SpringMVConfiguration extends WebMvcConfigurerAdapter {  
  
    public void configureMessageConverters(List<HttpMessageConverter<?>> converters) {  
  
        MappingJackson2HttpMessageConverter e = new MappingJackson2HttpMessageConverter();  
  
        converters.add(e);  
  
        e.setObjectMapper(createObjectMapper());  
  
    }  
  
    private ObjectMapper createObjectMapper() {  
  
        ObjectMapper objectMapper = new ObjectMapper();  
  
        objectMapper.configure(SerializationFeature.WRITE_DATES_AS_TIMESTAMPS, false);  
  
        objectMapper.setDateFormat(new SimpleDateFormat("dd/MM/yyyy HH:mm:ss"));  
  
        return objectMapper;  
  
    }  
  
}
```

Implementação Spring MVC – criar enquete

```
@RestController
@RequestMapping(EnqueteResource.URI)
public class EnqueteEndpoint {
    ...
    @RequestMapping(method = RequestMethod.POST, consumes = { MediaType.APPLICATION_JSON_VALUE },
        produces = { MediaType.APPLICATION_JSON_VALUE })
    @ResponseStatus(HttpStatus.CREATED)
    public EnqueteResource create(@RequestBody Enquete enquete)
        throws EnqueteInvalidaException {
        this.repository.create(enquete);
        return new EnqueteResource(enquete);
    }
    ...
}
```

Implementação Spring MVC – listar enquetes

```
@RestController
```

```
@RequestMapping(EnqueteResource.URI)
```

```
public class EnqueteEndpoint {
```

```
...
```

```
    @RequestMapping(method = RequestMethod.GET,
```

```
        produces = { MediaType.APPLICATION_JSON_VALUE })
```

```
    public EnquetesResource findAll() {
```

```
        return new EnquetesResource(this.repository.findAll());
```

```
    }
```

```
...
```

Implementação Spring MVC – visualizar enquete

```
@RestController

@RequestMapping(EnqueteResource.URI)

public class EnqueteEndpoint {

    ...

    @RequestMapping(value = "{id}", method = RequestMethod.GET,

        produces = { MediaType.APPLICATION_JSON_VALUE })

    public EnqueteResource get(@PathVariable("id") Long id) throws

    EnqueteNaoEncontradaException {

        return new EnqueteResource(this.repository.find(id));

    }

    ...
}
```


Implementação Spring MVC – apagar enquete

```
@RestController

@RequestMapping(EnqueteResource.URI)

public class EnqueteEndpoint {

    ...

    @RequestMapping(value = "{id}", method = RequestMethod.DELETE)
    @ResponseStatus(HttpStatus.NO_CONTENT)

    public void delete(@PathVariable("id") Long id)

        throws EnqueteNaoEncontradaException, EnqueteAtivaException {

        this.repository.delete(id);

    }

    ...
}
```

Implementação Spring MVC – votar em enquete

```
@RestController

@RequestMapping(EnqueteResource.URI)

public class EnqueteEndpoint {

    ...

    @RequestMapping(value = "{id}/voto", method = RequestMethod.POST,
        consumes = { MediaType.APPLICATION_JSON_VALUE })

    @ResponseStatus(HttpStatus.NO_CONTENT)

    public void vote(@PathVariable("id") Long id, @RequestBody Opcao opcao)

        throws EnqueteNaoEncontradaException, EnqueteInativaException,

            OpcaoNaoEncontradaException {

        this.repository.find(id).findOpcaoToVote(opcao.getTexto()).vote();

    }

    ...
}
```

Implementação Spring MVC – Tratamento de Exceptions

```
...
@ExceptionHandler({ EnqueteInvalidaException.class,
    OpcaoNaoEncontradaException.class })
@ResponseStatus(HttpStatus.BAD_REQUEST)
public String badRequestExceptionHandler(Exception e) {
    return e.getMessage();
}

@ExceptionHandler({ EnqueteNaoEncontradaException.class })
@ResponseStatus(HttpStatus.NOT_FOUND)
public String notFoundExceptionHandler(Exception e) {
    return e.getMessage();
}

@ExceptionHandler({ EnqueteAtivaException.class,
    EnqueteInativaException.class })
@ResponseStatus(HttpStatus.CONFLICT)
public String conflictExceptionHandler(Exception e) {
    return e.getMessage();
}
...
```



Referências

- <https://jersey.java.net/>
- <http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html>
- <http://wiki.fasterxml.com/JacksonHome>

Dúvidas?

