Bruno Stuelp Filho – bstuelp@gmail.com

Project GitHub: https://github.com/bstuelp/ANDDigital

1. Technologies
   a. Java + JAX-RS + Jersey
   b. Gson
   c. Tomcat 9
   d. Postman

2. Project Structure

   package com.andDigital.entity;
           Entities Customer and Phones

   package com.andDigital.services;
           Rest Services Methods and Interface

   package com.andDigital.test;
           JUnit tests

3. API – URLs e Spec

   - GET - http://localhost:8080/ANDDigital/rest/customer/getAll
   Responses
   - 200 – get all phone numbers
     Response body: application/json: The registers

   - GET - http://localhost:8080/ANDDigital/rest/customer/getPhone/{id}
   Responses
   - 200 – get all phone numbers of a single customer
     Response body: application/json: The register
   - 400 – Customer not found
     Response body: application/json: Error message

   - PUT - http://localhost:8080/ANDDigital/rest/customer/activate
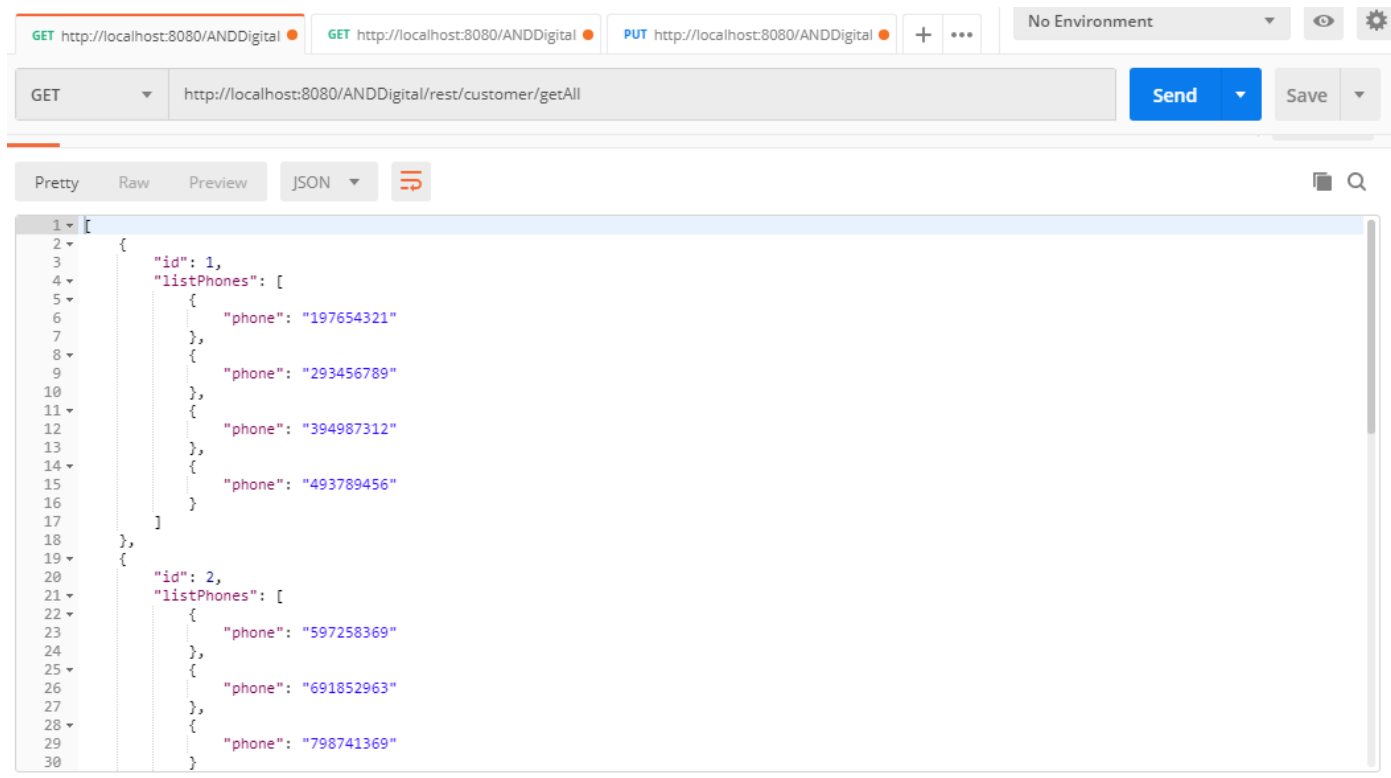   Responses
   - 200 – activate a phone number
     Response body: application/json: Success message
   - 400 – phone not found
     Response body: application/json: Error message
   Params – Json
           {
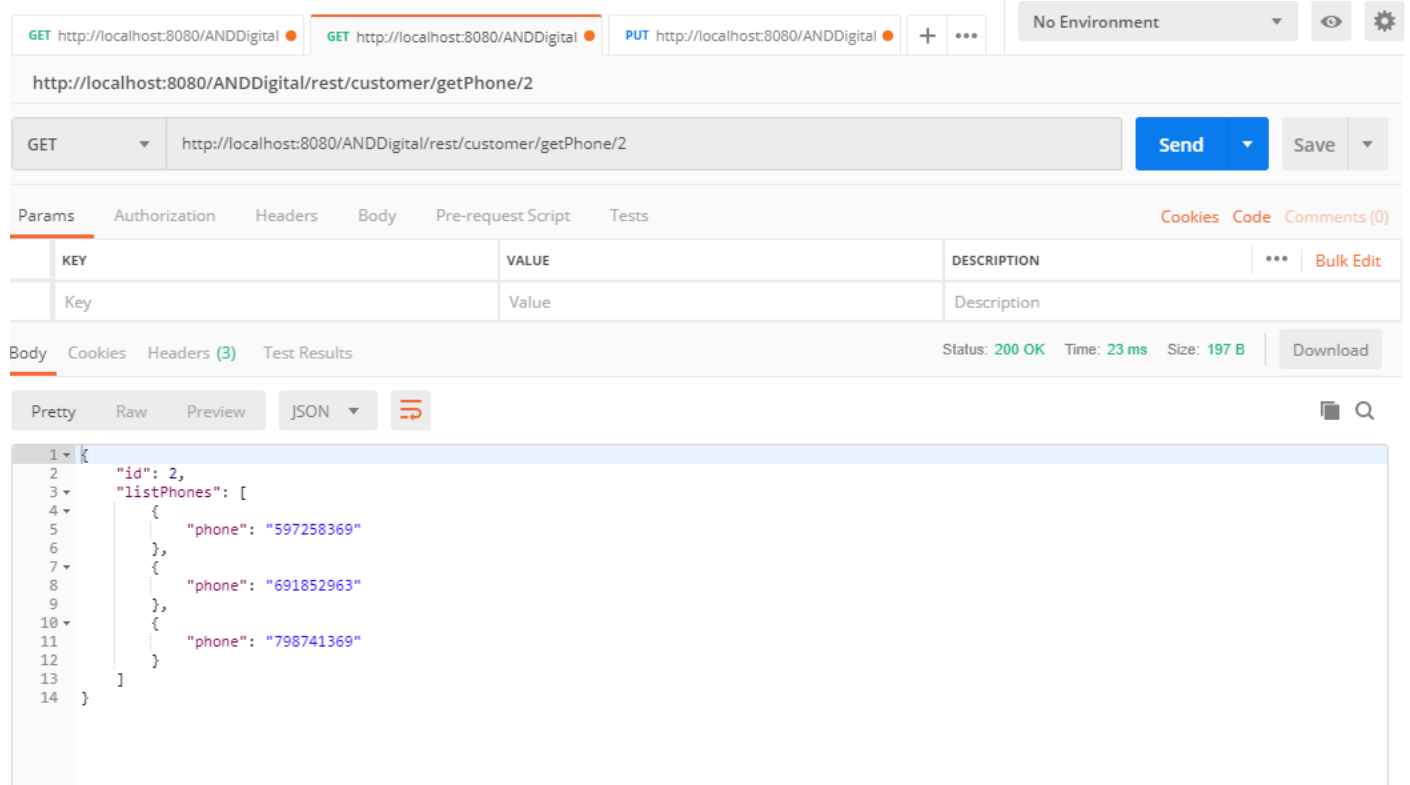   "number": "664512235"
           }

4. Tests case

- GET - http://localhost:8080/ANDDigital/rest/customer/getAll

http://localhost:8080/ANDDigital/rest/customer/getPhone/2 -> 200

http://localhost:8080/ANDDigital/rest/customer/getPhone/22 - > Return 400

http://localhost:8080/ANDDigital/rest/customer/activate - 200 - Number actvated
{
        "number": "664512235"
}

http://localhost:8080/ANDDigital/rest/customer/activate - 400 - Number not found.

```
{
        "number": "0011223344"
}
```
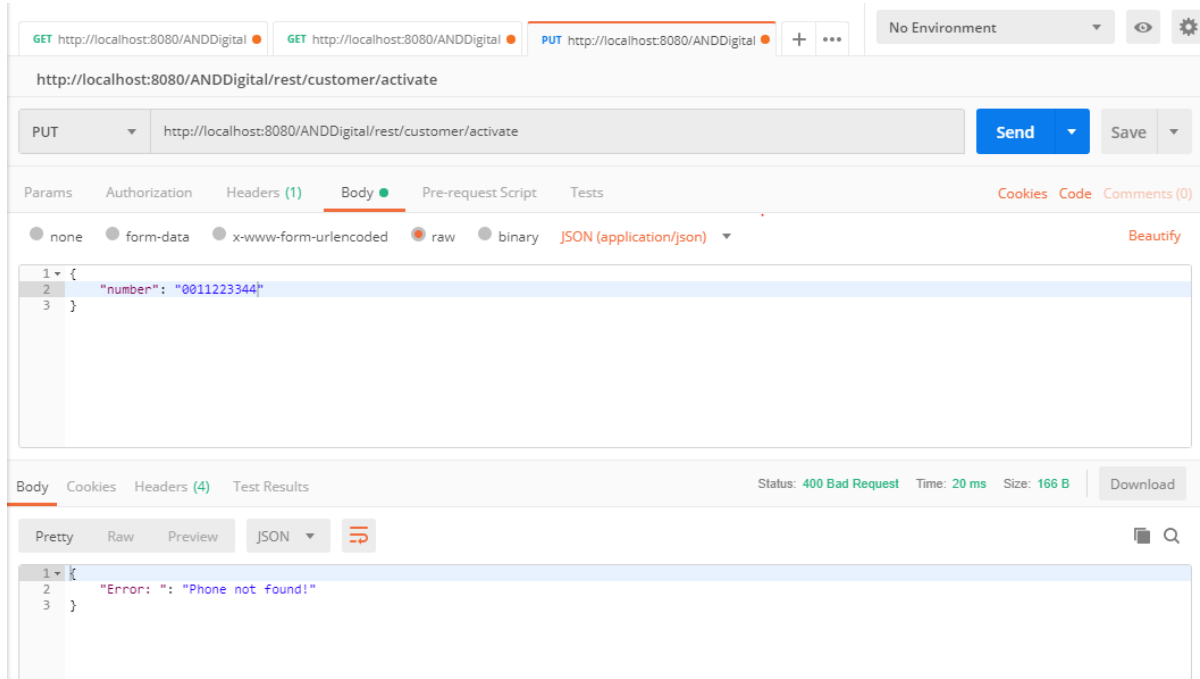

5. Code coverage
   a. Start

   b. After implementation

```java
@Test
public void testGetAllPhones() {
        assertEquals(200, classUnderTest.getAllPhones().getStatus());
}
```

```java
package com.andDigital.services;

import java.io.Serializable;

@Path("/customer")
public class Services implements Serializable, Iservices {
    private static final long serialVersionUID = 1L;
    private List<Customer> listCustomer = new ArrayList<Customer>();
    boolean activated = false;

    @GET
    @Path("/getAll")
    @Produces(MediaType.APPLICATION_JSON)
    @Override
    public Response getAllPhones() {
        listCustomer = this.createObject();

        return (Response.status(Response.Status.OK).entity(listCustomer).build());
    }

    @GET
    @Path("/getPhone/{id}")
    @Produces(MediaType.APPLICATION_JSON)
    @Override
    public Response getPhoneById(@PathParam("id") int id) {
        listCustomer = this.createObject();
```

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
ANDDigita	72,6 %	320	121	441
src	72,6 %	320	121	441
com.andDigital.services	73,7 %	283	101	384
com.andDigital.entity	50,0 %	20	20	40
com.andDigital.test	100,0 %	17	0	17

c. After implementation

```java
@Test
public void testGetPhoneById() {
    assertEquals(200, classUnderTest.getPhoneById(2).getStatus());

    assertEquals(400, classUnderTest.getPhoneById(22).getStatus());
}
```



```java
        return (Response.status(Response.Status.OK).entity(listCustomer).build());
    }

    @GET
    @Path("/getPhone/{id}")
    @Produces(MediaType.APPLICATION_JSON)
    @Override
    public Response getPhoneById(@PathParam("id") int id) {
        listCustomer = this.createObject();

        Customer data = listCustomer.stream().filter(customer -> id == customer.getId()).findAny().orElse(null);

        if (data == null) {
            Map<String, String> mapError = new HashMap<String, String>();
            mapError.put("Error: ", "Customer not found!");

            return (Response.status(Response.Status.BAD_REQUEST).entity(mapError).build());
        }

        return (Response.status(Response.Status.OK).entity(data).build());
    }

    @PUT
    @Path("/activate/{number}")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
```

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
ANDDigita	80,7 %	363	87	450
src	80,7 %	363	87	450
com.andDigital.services	81,8 %	314	70	384
com.andDigital.entity	57,5 %	23	17	40
com.andDigital.test	100,0 %	26	0	26

```java
34        return (Response.status(Response.Status.OK).entity(listCustomer).build());
35    }
36
37⊝    @GET
38     @Path("/getPhone/{id}")
39     @Produces(MediaType.APPLICATION_JSON)
40     @Override
41     public Response getPhoneById(@PathParam("id") int id) {
42        listCustomer = this.createObject();
43
44        Customer data = listCustomer.stream().filter(customer -> id == customer.getId()).findAny().orElse(null);
45
46        if (data == null) {
47            Map<String, String> mapError = new HashMap<String, String>();
48            mapError.put("Error: ", "Customer not found!");
49
50            return (Response.status(Response.Status.BAD_REQUEST).entity(mapError).build());
51        }
52
53        return (Response.status(Response.Status.OK).entity(data).build());
54    }
55
56⊝    @PUT
57     @Path("/activate/{number}")
58     @Consumes(MediaType.APPLICATION_JSON)
59     @Produces(MediaType.APPLICATION_JSON)
```

AndDigitalTest (20/02/2019 21:36:51)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
ANDDigita	84,3 %	386	72	458
src	84,3 %	386	72	458
com.andDigital.services	85,7 %	329	55	384
com.andDigital.entity	57,5 %	23	17	40
com.andDigital.test	100,0 %	34	0	34

d.   After implementation

```java
@Test
public void testActivatedNumber() {
        assertEquals(200, classUnderTest.activatedNumber("493789456").getStatus());
        assertEquals(400, classUnderTest.activatedNumber("000000000").getStatus());
}
```



```java
60     @Override
61     public Response activatedNumber(@PathParam("number") String number) {
62     {
63        listCustomer = this.createObject();
64
65        listCustomer.forEach((Customer c) -> {
66            List<Phones> data = c.getListPhones();
67            data.forEach((Phones p) -> {
68                if (number.equals(p.getPhone()))
69                {
70                    activated = true;
71                    p.setActivated(true);
72                }
73            });
74        });
75
76        if (!activated)
77        {
78            Map<String, String> mapError = new HashMap<String, String>();
79            mapError.put("Error: ", "Phone not found!");
80
81            return (Response.status(Response.Status.BAD_REQUEST).entity(mapError).build());
82        }
83
84        return (Response.status(Response.Status.OK).entity("Phone activated.").build());
85    }
```

AndDigitalTest (20/02/2019 21:38:59)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
ANDDigital	95,3 %	445	22	467

```
65
66        listCustomer.forEach((Customer c) -> {
67            List<Phones> data = c.getListPhones();
68            data.forEach((Phones p) -> {
69                if (number.equals(p.getPhone()))
70                {
71                    activated = true;
72                    p.setActivated(true);
73                }
74            });
75        });
76
77        if (!activated)
78        {
79            Map<String, String> mapError = new HashMap<String, String>();
80            mapError.put("Error: ", "Phone not found!");
81
82            return (Response.status(Response.Status.BAD_REQUEST).entity(mapError).build());
83        }
84
85        return (Response.status(Response.Status.OK).entity("Phone activated.").build());
86    }
87
88⊖   @Override
89    public List<Customer> createObject() {
90        /* Customer 1 */
```

Problems Search Console Servers Coverage ✕

AndDigitalTest (20/02/2019 21:47:33)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
ANDDigita	98,6 %	480	7	487
src	98,6 %	480	7	487
com.andDigital.entity	82,5 %	33	7	40
com.andDigital.services	100,0 %	384	0	384
com.andDigital.test	100,0 %	63	0	63

```
12
13⊖   @Before
14    public void setUp() throws Exception
15    {
16        classUnderTest = new Services();
17    }
18
19⊖   @Test
20    public void testGetAllPhones() {
21        assertEquals(200, classUnderTest.getAllPhones().getStatus());
22    }
23
24⊖   @Test
25    public void testGetPhoneById() {
26        assertEquals(200, classUnderTest.getPhoneById(2).getStatus());
27
28        assertEquals(400, classUnderTest.getPhoneById(22).getStatus());
29    }
30
31⊖   @Test
32    public void testActivatedNumber() {
33        assertEquals(200, classUnderTest.activatedNumber("493789456").getStatus());
34        assertEquals(400, classUnderTest.activatedNumber("000000000").getStatus());
35    }
36 }
37
```

Problems Search Console Servers Coverage ✕

AndDigitalTest (20/02/2019 21:52:03)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
ANDDigita	100,0 %	477	0	477
src	100,0 %	477	0	477
com.andDigital.entity	100,0 %	33	0	33
com.andDigital.services	100,0 %	393	0	393
com.andDigital.test	100,0 %	51	0	51

6. Project decisions
   - I have created two entities as if they represent the database;
   - I populate these objects in the "**public** List<Customer> createObject()" method;
   - In "**public** Response getAllPhones()" I just return the Object;
   - In "**public** Response getPhoneById(@PathParam("id") int id)" I used Lambda function to filter customer data in object;
   - In "**public** Response activatedNumber(String json)" I used again lambda functions and nested loops to set activated a phone number;