Challenge: SkipTheDishes - Skip the Dishes

Name: Sylvio Antonio Pedroza Neto

Position: Java Developer

Email: sylvio.pedroza@gmail.com / Phone**:** +55 11 94252 5313

Linkedin**:** https://www.linkedin.com/in/sylvio-antonio-pedroza-neto-5a0b8150/

Github: https://github.com/sylvioneto/vanhack

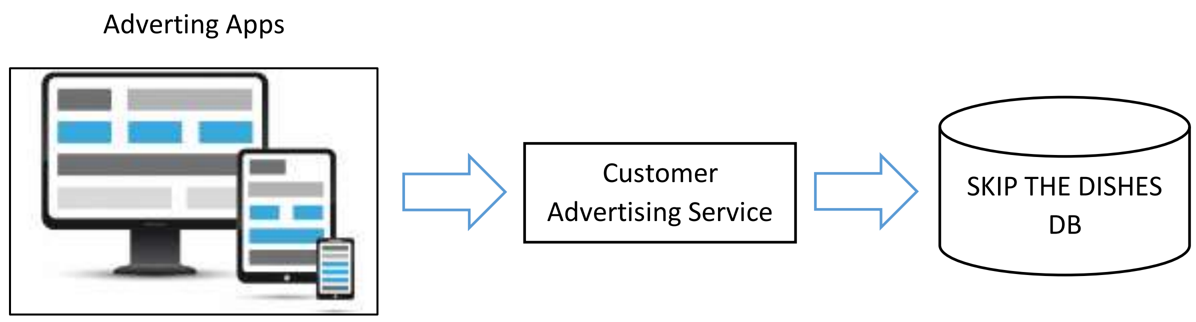
Contents

Introduction

I created a micro service that can be used for mobile and desktop clients in order to get customer information.

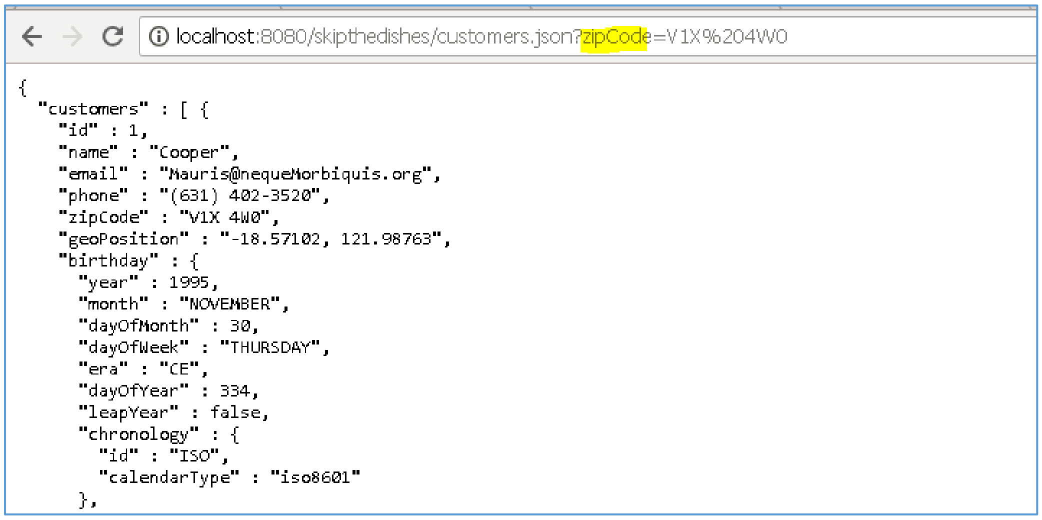
Advertising apps can invoke these APIs to find potential customers based on zip code or geo position, and send them promotions or other advertising goals. For instance, a restaurant would like to see all customers placed in a specific zip code where people don’t order food very often in order to offer them a discount. Or, a new restaurant want to know who leaves in the neighborhood and offer a promotional price in its opening week.

Architecture



Examples

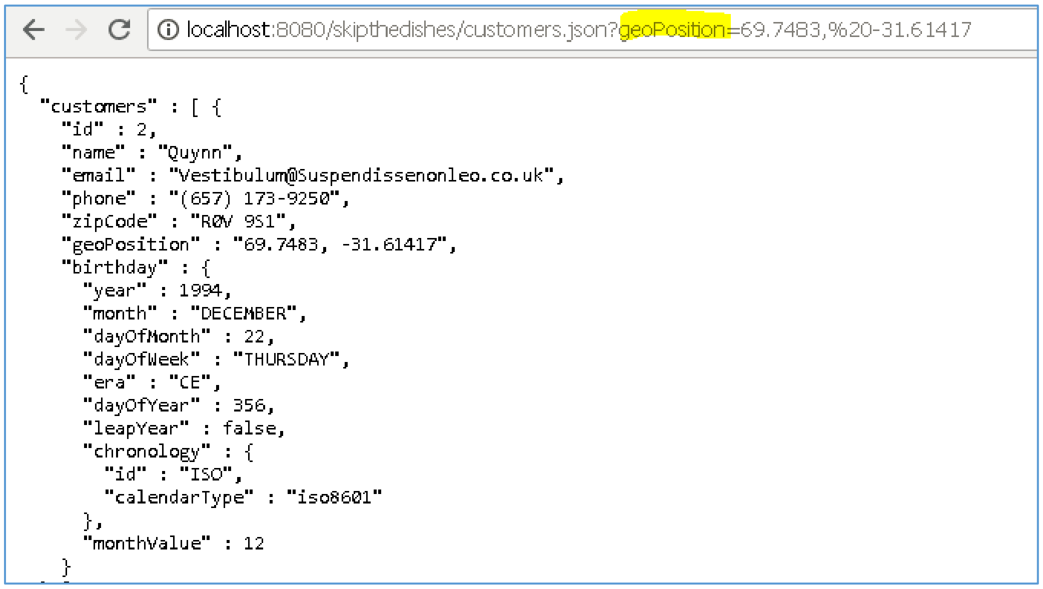
1: Using zip code



localhost:8080/skipthedishes/customers.json?zipCode=V1X%204W0

```
{
  "customers" : [ {
    "id" : 1,
    "name" : "Cooper",
    "email" : "Mauris@nequeMorbiquis.org",
    "phone" : "(631) 402-3520",
    "zipCode" : "V1X 4W0",
    "geoPosition" : "-18.57102, 121.98763",
    "birthday" : {
      "year" : 1995,
      "month" : "NOVEMBER",
      "dayOfMonth" : 30,
      "dayOfWeek" : "THURSDAY",
      "era" : "CE",
      "dayOfYear" : 334,
      "leapYear" : false,
      "chronology" : {
        "id" : "ISO",
        "calendarType" : "iso8601"
      },
```

Example 2: Using geo position



localhost:8080/skipthedishes/customers.json?geoPosition=69.7483,%20-31.61417

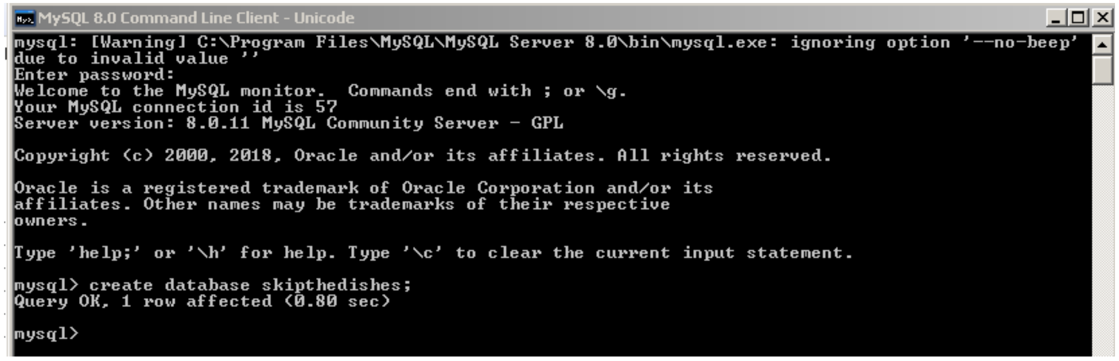
```
{
  "customers" : [ {
    "id" : 2,
    "name" : "Quynn",
    "email" : "Vestibulum@Suspendissenonleo.co.uk",
    "phone" : "(657) 173-9250",
    "zipCode" : "R0V 9S1",
    "geoPosition" : "69.7483, -31.61417",
    "birthday" : {
      "year" : 1994,
      "month" : "DECEMBER",
      "dayOfMonth" : 22,
      "dayOfWeek" : "THURSDAY",
      "era" : "CE",
      "dayOfYear" : 356,
      "leapYear" : false,
      "chronology" : {
        "id" : "ISO",
        "calendarType" : "iso8601"
      },
      "monthValue" : 12
    }
```

Database

MYSQL 8

To create the database, table and insert the sample data, please do the following:

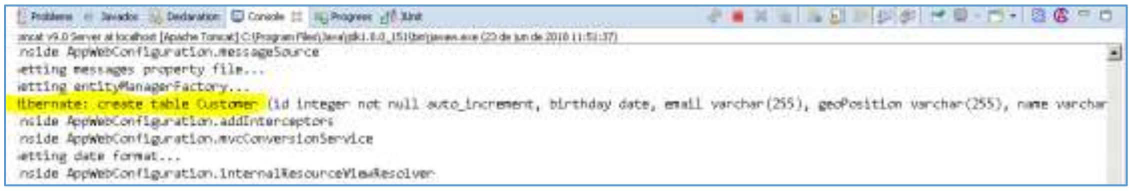
1. Login in MYSQL
2. Type “create database skipthedishes”



MySQL 8.0 Command Line Client - Unicode

```
mysql: [Warning] C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql.exe: ignoring option '--no-beep'
due to invalid value ''
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 57
Server version: 8.0.11 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database skipthedishes;
Query OK, 1 row affected (0.80 sec)

mysql>
```

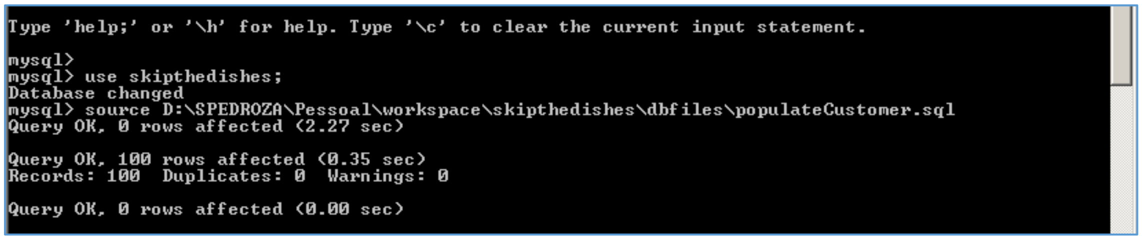
3. The application will create the tables automatically on start. You can check that in console:



```
mcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jdk1.8.0_151\bin\javaw.exe (23 de jun de 2018 11:51:37)
nside AppWebConfiguration.messageSource
etting messages property file...
etting entityManagerFactory...
Hibernate: create table Customer (id integer not null auto_increment, birthday date, email varchar(255), geoPosition varchar(255), name varchar
nside AppWebConfiguration.addInterceptors
nside AppWebConfiguration.mvcConversionService
etting date format...
nside AppWebConfiguration.internalResourceViewResolver
```

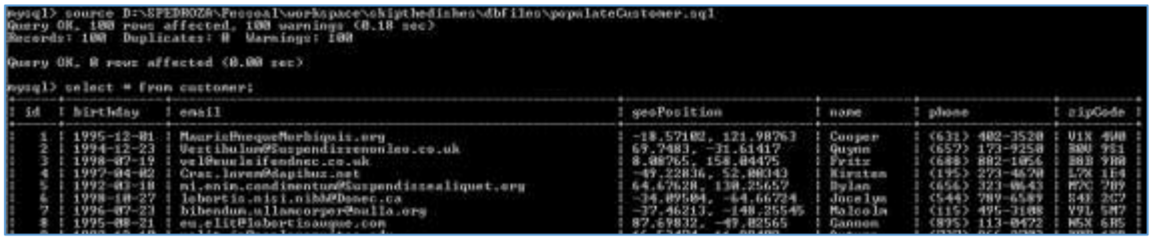
4. Run the *populateCustomer.sql* located in the dbfiles folder into the project. This script will create the customer table and load customer sample data:
   To run the script, type
   “source D:\SPEDROZA\Pessoal\workspace\skipthedishes\dbfiles\populateCustomer.sql”

   Replace “D:\SPEDROZA\Pessoal\workspace\skipthedishes\dbfiles\” according to your machine.



```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql> use skipthedishes;
Database changed
mysql> source D:\SPEDROZA\Pessoal\workspace\skipthedishes\dbfiles\populateCustomer.sql
Query OK, 0 rows affected (2.27 sec)

Query OK, 100 rows affected (0.35 sec)
Records: 100  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.00 sec)
```

   To check inserted data, just type “select * from customer”:



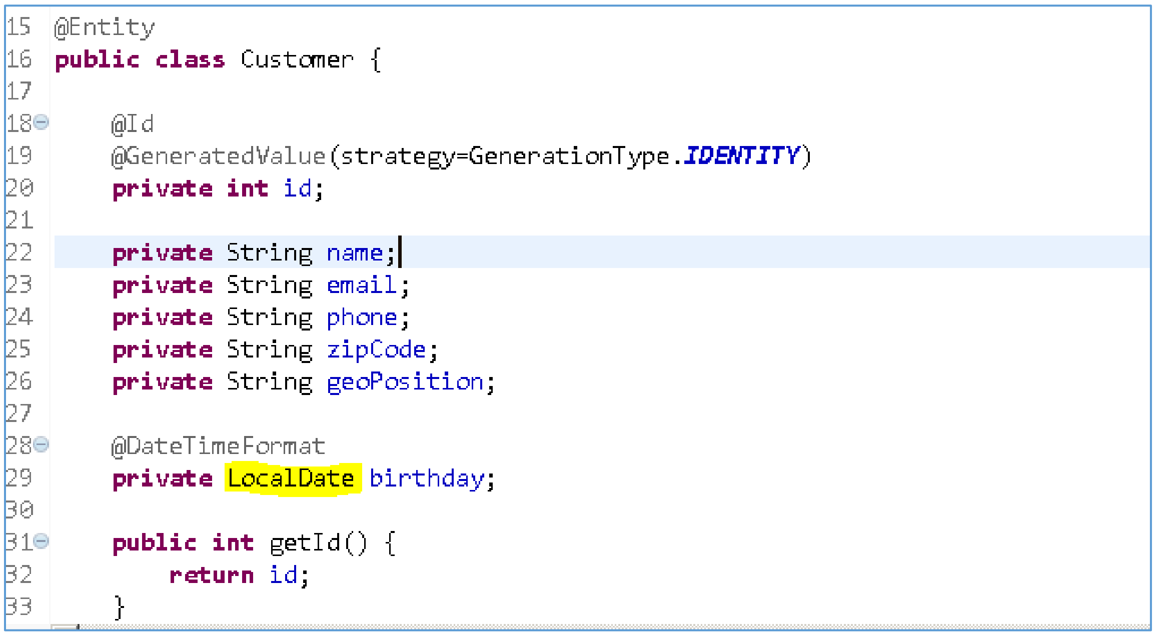
```
mysql> source D:\SPEDROZA\Pessoal\workspace\skipthedishes\dbfiles\populateCustomer.sql
Query OK, 100 rows affected, 100 warnings (0.18 sec)
Records: 100  Duplicates: 0  Warnings: 100

Query OK, 0 rows affected (0.00 sec)

mysql> select * from customer;
+-----+------------+---------------------------------------------+-------------------------+----------+----------------+---------+
| id  | birthday   | email                                       | geoPosition             | name     | phone          | zipCode |
+-----+------------+---------------------------------------------+-------------------------+----------+----------------+---------+
|   1 | 1995-12-01 | MaurisPhequeMorbiquis.org                   | -18.57102, 121.98763    | Cooper   | (631) 402-3520 | V1X 4N0 |
|   2 | 1994-12-23 | Vestibulum@Suspendissenonleo.co.uk          | 69.7483, -31.61417      | Quynn    | (657) 173-9258 | B0W 9S1 |
|   3 | 1998-07-19 | vel@nuelaifendnec.co.uk                     | 8.08765, 158.04475      | Fritz    | (688) 882-1056 | B8B 9R0 |
|   4 | 1997-04-02 | Cras.lorem@dapibus.net                      | -49.22836, 52.00343     | Kirsten  | (195) 273-4670 | L7X 1E4 |
|   5 | 1992-03-18 | mi.enim.condimentum@Suspendissealiquet.org  | 64.67628, 130.25657     | Dylan    | (656) 323-0643 | M7C 7B9 |
|   6 | 1998-10-27 | lobortis.nisi.nibh@Donec.ca                 | -34.09504, -64.66724    | Jocelyn  | (544) 789-6589 | S4E 2C7 |
|   7 | 1996-07-23 | bibendum.ullamcorper@nulla.org              | -37.46213, -148.25545   | Malcolm  | (115) 495-3108 | V9L 5M7 |
|   8 | 1995-08-21 | eu.elit@lobortisaugue.com                   | 87.69832, -47.02565     | Gannon   | (895) 113-0472 | N5X 6R5 |
```

Application

I used JAVA8 to compile my application, Spring MVC, Hibernate and JUNIT.

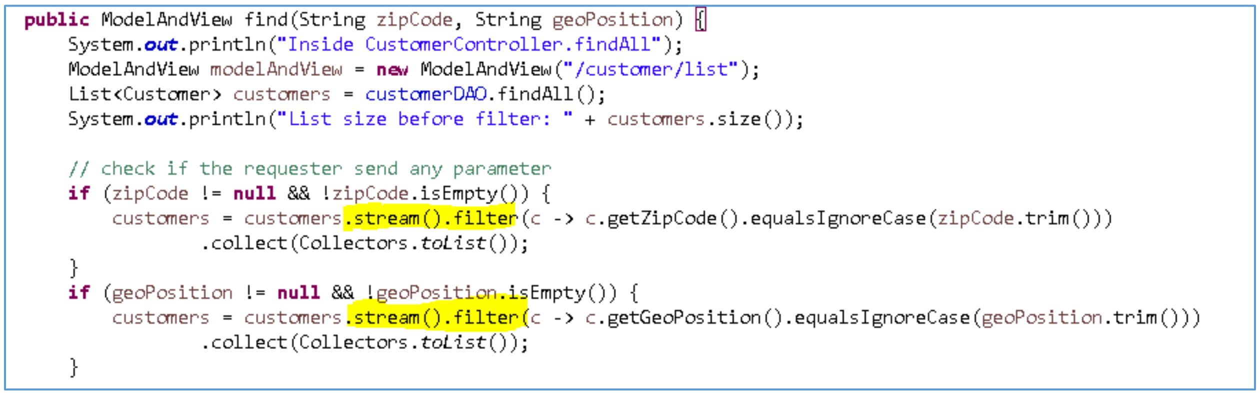
So, you will find some J8 features, for instance:

New java date api LocalDate



```
@Entity
public class Customer {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private int id;

    private String name;
    private String email;
    private String phone;
    private String zipCode;
    private String geoPosition;

    @DateTimeFormat
    private LocalDate birthday;

    public int getId() {
        return id;
    }
```

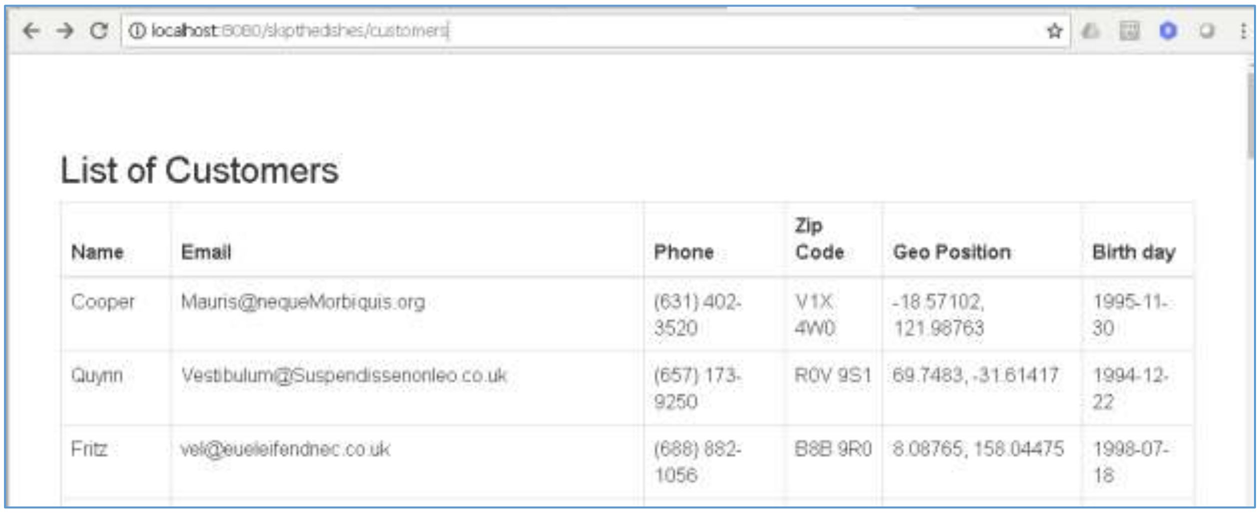
Streams and lambda



```
public ModelAndView find(String zipCode, String geoPosition) {
    System.out.println("Inside CustomerController.findAll");
    ModelAndView modelAndView = new ModelAndView("/customer/list");
    List<Customer> customers = customerDAO.findAll();
    System.out.println("List size before filter: " + customers.size());

    // check if the requester send any parameter
    if (zipCode != null && !zipCode.isEmpty()) {
        customers = customers.stream().filter(c -> c.getZipCode().equalsIgnoreCase(zipCode.trim()))
                .collect(Collectors.toList());
    }
    if (geoPosition != null && !geoPosition.isEmpty()) {
        customers = customers.stream().filter(c -> c.getGeoPosition().equalsIgnoreCase(geoPosition.trim()))
                .collect(Collectors.toList());
    }
```

I also used **internal resolvers** to return the requested response format. So, the requester can use the same URL to get html or json. Also, the developer doesn't have to maintain two different methods. It makes the application easy to maintain. Example testing both returns in a browser:

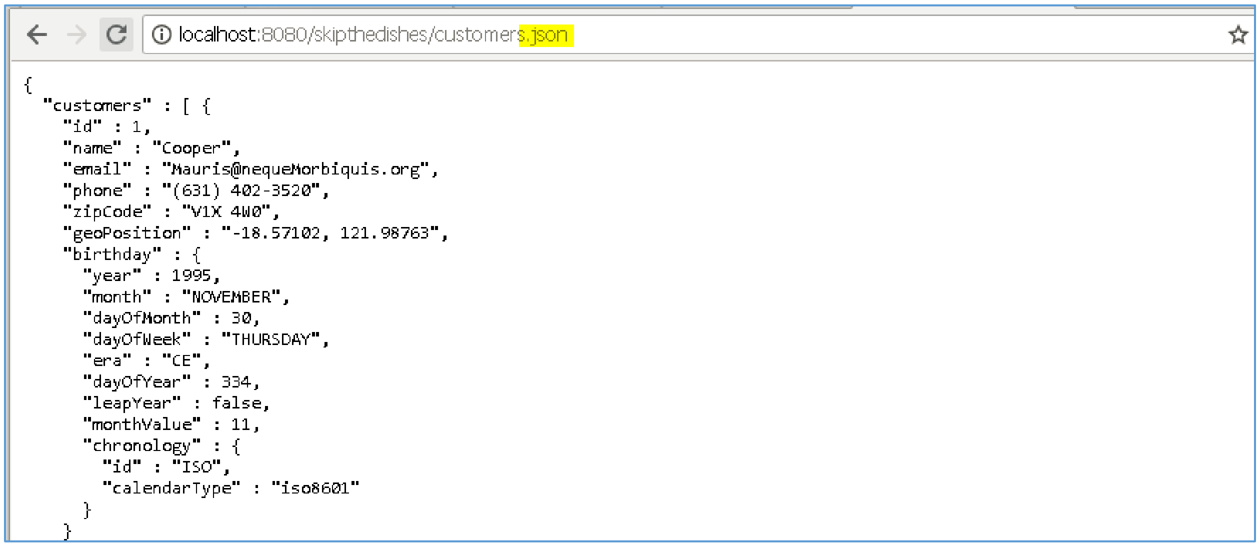
HTML



List of Customers

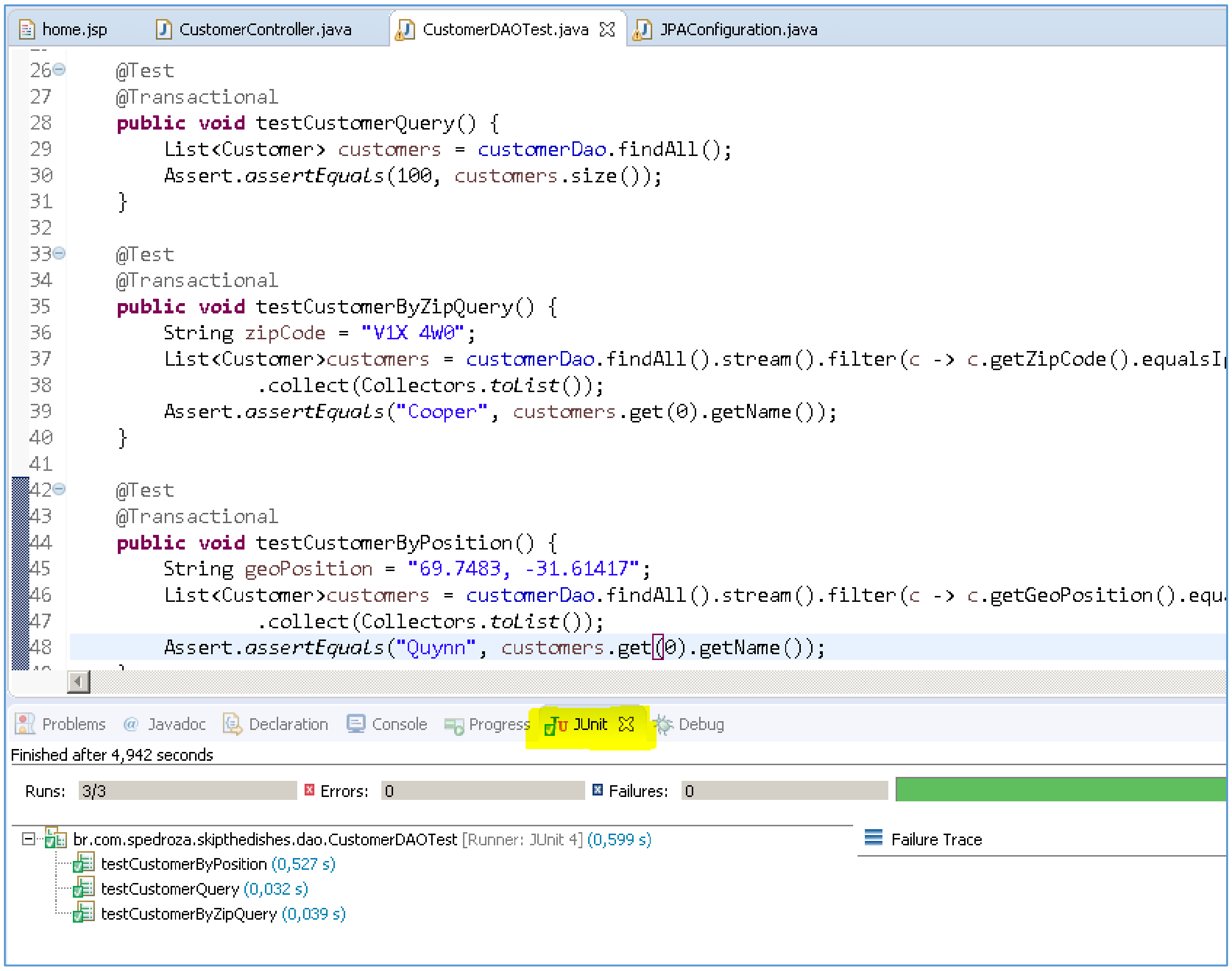
Name	Email	Phone	Zip Code	Geo Position	Birth day
Cooper	Mauris@nequeMorbiquis.org	(631) 402-3520	V1X 4W0	-18.57102, 121.98763	1995-11-30
Quynn	Vestibulum@Suspendissenonleo.co.uk	(657) 173-9250	R0V 9S1	69.7483, -31.61417	1994-12-22
Fritz	vel@eueleifendnec.co.uk	(688) 882-1056	B8B 9R0	8.08765, 158.04475	1998-07-18

JSON



```
{
  "customers" : [ {
    "id" : 1,
    "name" : "Cooper",
    "email" : "Mauris@nequeMorbiquis.org",
    "phone" : "(631) 402-3520",
    "zipCode" : "V1X 4W0",
    "geoPosition" : "-18.57102, 121.98763",
    "birthday" : {
      "year" : 1995,
      "month" : "NOVEMBER",
      "dayOfMonth" : 30,
      "dayOfWeek" : "THURSDAY",
      "era" : "CE",
      "dayOfYear" : 334,
      "leapYear" : false,
      "monthValue" : 11,
      "chronology" : {
        "id" : "ISO",
        "calendarType" : "iso8601"
      }
    }
```

Finally, I wrote JUNIT classes to test the application. So, it can be used in Continuous Delivery workflow.

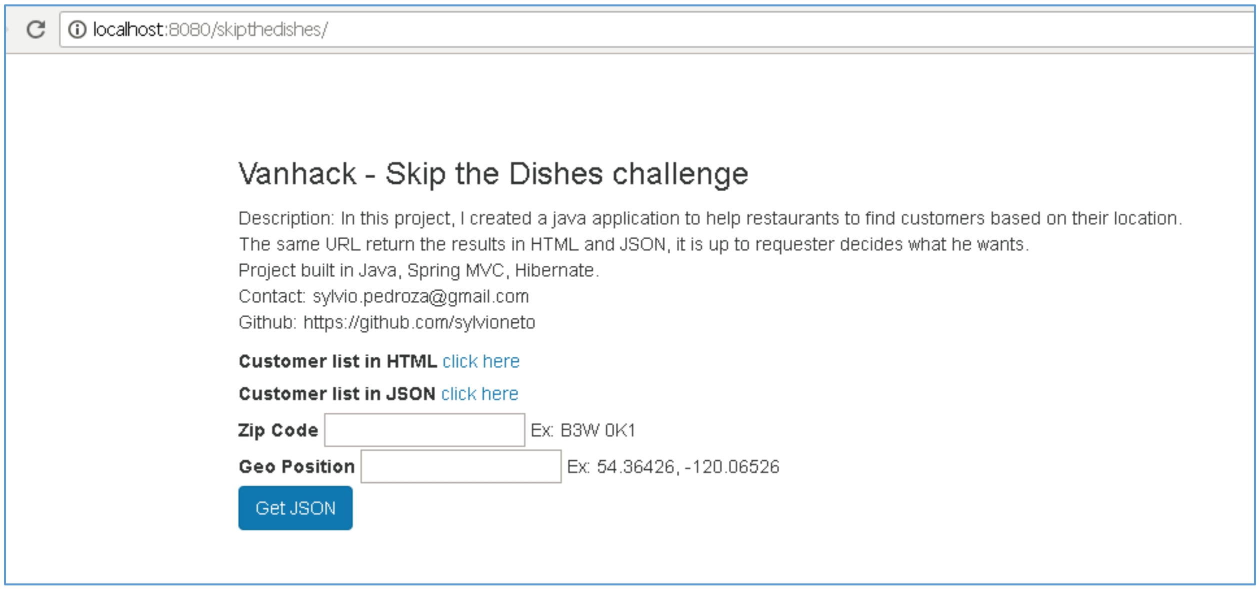


```
    @Test
    @Transactional
    public void testCustomerQuery() {
        List<Customer> customers = customerDao.findAll();
        Assert.assertEquals(100, customers.size());
    }

    @Test
    @Transactional
    public void testCustomerByZipQuery() {
        String zipCode = "V1X 4W0";
        List<Customer>customers = customerDao.findAll().stream().filter(c -> c.getZipCode().equalsI
                .collect(Collectors.toList());
        Assert.assertEquals("Cooper", customers.get(0).getName());
    }

    @Test
    @Transactional
    public void testCustomerByPosition() {
        String geoPosition = "69.7483, -31.61417";
        List<Customer>customers = customerDao.findAll().stream().filter(c -> c.getGeoPosition().equ
                .collect(Collectors.toList());
        Assert.assertEquals("Quynn", customers.get(0).getName());
```

```
Finished after 4,942 seconds
Runs: 3/3    Errors: 0    Failures: 0

br.com.spedroza.skipthedishes.dao.CustomerDAOTest [Runner: JUnit 4] (0,599 s)
    testCustomerByPosition (0,527 s)
    testCustomerQuery (0,032 s)
    testCustomerByZipQuery (0,039 s)
```

Testing

You can use a browser and the home page to test the application:



localhost:8080/skipthedishes/

Vanhack - Skip the Dishes challenge

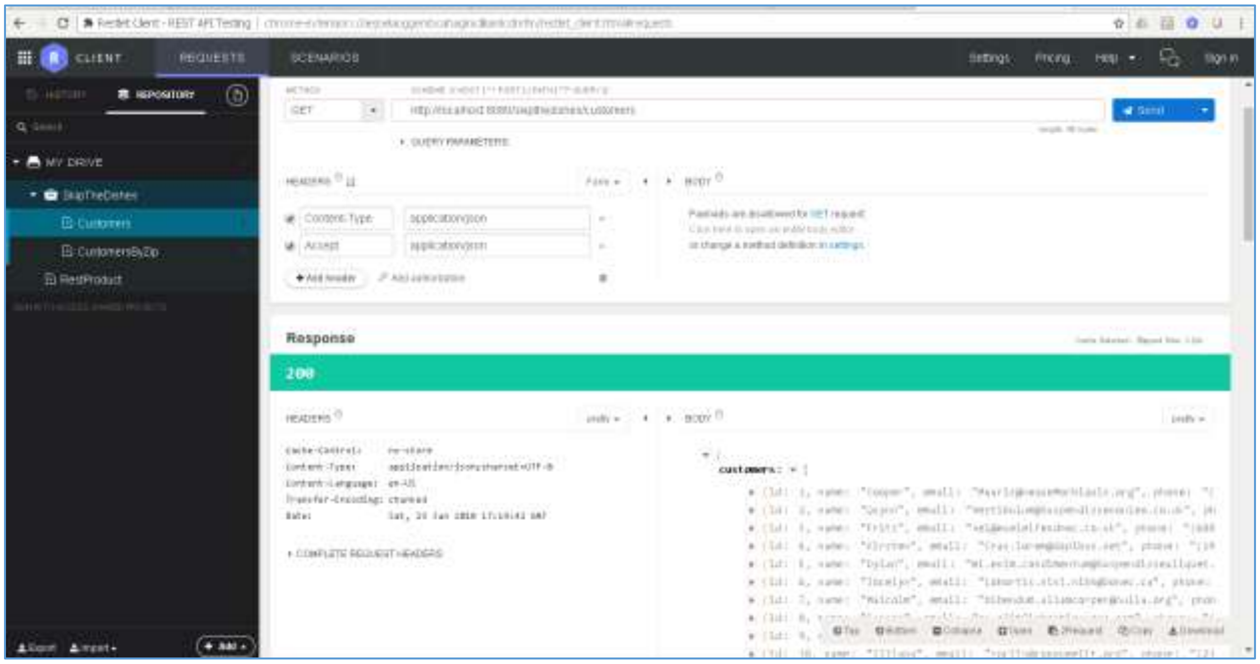
Description: In this project, I created a java application to help restaurants to find customers based on their location.
The same URL return the results in HTML and JSON, it is up to requester decides what he wants.
Project built in Java, Spring MVC, Hibernate.
Contact: sylvio.pedroza@gmail.com
Github: https://github.com/sylvioneto

Customer list in HTML click here

Customer list in JSON click here

Zip Code	Ex: B3W 0K1
Geo Position	Ex: 54.36426, -120.06526

Or any tool you want:



METHOD: GET

http://localhost:8080/skipthedishes/customers

HEADERS

Content-Type	application/json
Accept	application/json

BODY

Payloads are disallowed for GET request.

200

```
Cache-Control:      no-store
Content-Type:       application/json;charset=UTF-8
Content-Language:   en-US
Transfer-Encoding:  chunked
Date:               Sat, 20 Jan 2018 17:19:42 GMT
```

COMPLETE REQUEST HEADERS

```
{
  customers: [
    {id: 1, name: "Cooper", email: [illegible]
    {id: 2, name: [illegible]
    {id: 3, name: "Fritz", email: [illegible]
    {id: 4, name: [illegible]
    {id: 5, name: "Dylan", email: [illegible]
    {id: 6, name: [illegible]
    {id: 7, name: "Malcolm", email: [illegible]
    {id: 8, [illegible]
    {id: 9, [illegible]
    {id: 10, name: [illegible]
```