Problem:

Implement a simple Web application with following characteristics:

- Be able to read data from a xml file

- Respond to below query from those data:
  + when requesting URL **http://localhost/YOURAPPLICATIONNAME/average** return in JSON format the average age of people grouped by age range (0-19, 20-39, 40-59, 60-79 and so on)

Design:

By analyzing the needs of the problem, I designed a web service as below:

- RESTful web service

- POST method

- Input data in plain text. This is the absolute path or the URL of a file which contains a XML data in the predefined format. For example: "**D:\Coding\amadeus-app\data.xml**" or "**http://hcmc1.vn/data.xml**".

- Output data in the JSON format. For example:

```
[
  {"range": "0-19", "average": 17.666666},
  {"range": "20-39", "average": 25.333334},
  {"range": "40-59", "average": 47.0},
  {"range": "60-79", "average": 63.0}
]
```

Technology:

- Java EE
- Spring Web
- JAXB
- Junit (for testing)
- Spring test (for testing)
- Maven
- Github
- Tomcat server

Command line:

- Typing the following command to build a Java web application from scratch, by using the Maven "*maven-archetype-webapp*" template:

```
$ mvn archetype:generate -DgroupId=com.example -DartifactId=amadeus-app
-DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false
```

(If you download the source code from https://github.com/luongthenhan/amadeus-app, ignore this command).

- Enter the project folder

```
$ cd amadeus-app
```

- Testing the project (internet connection required):

```
$ mvn clean test
```

- Packaging the project to build a war file:

```
$ mvn clean package
```

Deploy:

- Install Apache Tomcat (I used the version 8.5.4). In general case, the Tomcat server is installed on the port 8080.

- To automatically deploy a war application on Tomcat, you have to do some configurations:

+ Add a user with roles **manager-gui** and **manager-script** in the **%TOMCAT_PATH%/conf/tomcat-users.xml** file.

```
<tomcat-users >
     <role rolename="manager-gui"/>
     <role rolename="manager-script"/>
     <user username="admin" password="password" roles="admin-gui,
manager-gui, manager-script" />
</tomcat-users>
```

+ Define the Tomcat server in the **%MAVEN_PATH%/conf/settings.xml** file.

```
<settings>
   <servers>
     <server>
          <id>TomcatServer</id>
          <username>admin</username>
          <password>password</password>
     </server>
  </servers>
</settings>
```

+ In the **pom.xml** file of the project, add the Maven plugin for Tomcat:

```
<plugin>
   <groupId>org.codehaus.mojo</groupId>
   <artifactId>tomcat-maven-plugin</artifactId>
   <version>1.1</version>
   <configuration>
        <url>http://localhost:8080/manager/text</url>
        <server>TomcatServer</server>
        <path>/amadeus-app</path>
    </configuration>
</plugin>
```

- Type the following command to deploy the WAR file on Tomcat. In general, you can call the web service at the URL: **http://localhost:8080/amadeus-app/average**

```
$ mvn tomcat:deploy
```

- The Tomcat server is installed on port 8080 and your machine may have also an Apache HTTP server on port 80. In order to call the web service at the URL **http://localhost/amadeus-app/average**, you have to transfer the request from port 80 to port 8080. To do that, you should configure two files **httpd.conf** and **httpd-vhosts.conf** by using a proxy pass.

+ In the **httpd.conf** file, remove the # character from the beginning of two lines:

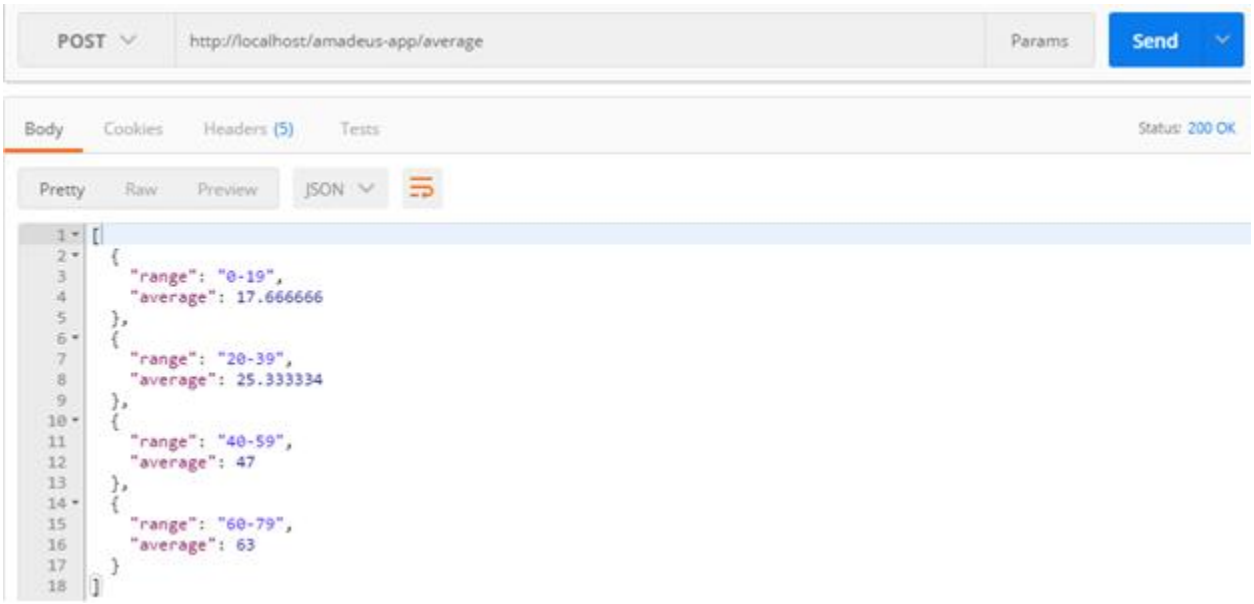
```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

+ In the **httpd-vhosts.conf** file, add a virtual host:

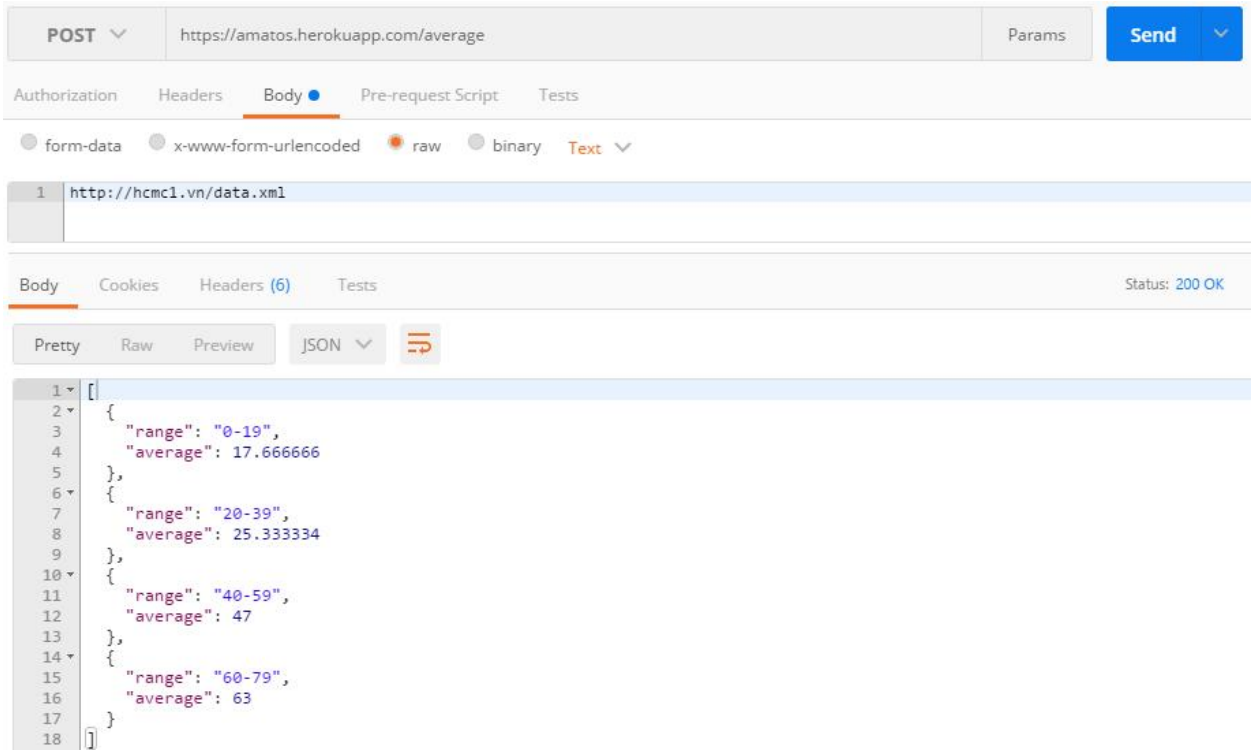
```
<VirtualHost *:80>
     ProxyPreserveHost On
     ProxyRequests Off
     ServerName localhost
     ProxyPass /amadeus-app/ http://localhost:8080/amadeus-app/
     ProxyPassReverse /amadeus-app/ http://localhost:8080/amadeus-app/
</VirtualHost>
```

Testing:

- Using Postman (a Chrome app):



- The web service can be deployed on the Cloud. For example:
Web service URL: https://amatos.herokuapp.com/average
Input String: http://hcmc1.vn/data.xml



For further:

1. When you call the web service by source code, for example by jQuery:

```
$.ajax({
    method: "POST",
    data: "D:\Coding\amadeus-app2\data2.xml",
    url: "http://localhost:8080/amadeus-app/average"
}).done(function (data) {
    console.log(data);
});
```

You may have this error:

`XMLHttpRequest cannot load http://localhost:8080/amadeus-app/average. No` **`'Access-Control-Allow-Origin'`** `header is present on the requested resource.`

- To resolve this problem, you just have to add a **`@CrossOrigin`** annotation to the handler method. The value of **`origins`** is domains of the client application.

```
@CrossOrigin(origins = "http://localhost:80")
@RequestMapping(value = "/average", method = RequestMethod.POST, produces = "application/json")
public @ResponseBody
List<AgeRange> calculateAverage(@RequestBody String filePath, final HttpServletResponse response) {

}
```

2. You may want to secure your web service. To do that, you can use a **Basic Auth** method.

- In the client application:

```
$.ajax({
    method: "POST",
    data: "D:\Coding\amadeus-app2\data2.xml",
    url: "http://localhost:8080/amadeus-app/average",
    xhrFields: { withCredentials: true},
    beforeSend: function (xhr) {
       xhr.setRequestHeader("Authorization", "Basic " + btoa("username:password"));
    }
}).done(function (data) {
    console.log(data);

});
```

- Add the **`@Context HttpHeaders httpHeaders`** parameter in the handler method of the web service:

```
List<AgeRange> calculateAverage(@RequestBody String filePath, final HttpServletResponse response, @Context HttpHeaders httpHeaders) {
}
```

The **`httpHeaders`** parameter contains the pair **`username:password`** encrypted. Then, you can decrypt it and check something.