**This Document consists of assignment description and solution details.**

1. **Assignment**

**Business Requirements**

Our client has several systems that require reference information to work. This information is delivered periodically in the format of CSV file and needs to be exposed via REST webservice.

Our assignment requires the creation of three modules:

1. A module responsible for loading a file from a directory to the MongoDB;
2. A module responsible for exposing the information on MongoDB with a web service;
3. Simple client consuming the webservice information to show how the webservice works – this client only has to log into the console the result.

**Technical Requirements**

The following requirements are mandatory: • Usage of MongoDB; • Usage of JEE 8; • Usage of Wildfly; • Usage of Maven to build the project. Even though other API’s may be used, the following stack should suffice to create a working solution.

**Not required but appreciated**

* Unit Testing
* Diagram of the solution – UML Architecture Diagram
* Instructions on usage (README)

**Expected**

1. The solution is delivered in a compressed file or made available on GitHub;
2. The solution will be usable in the following situations:
3. A file will be put in the import directory and the information will be uploaded to the database;
4. A HTTP GET to the webservice endpoint will reply with a JSON containing the information in the database:
5. A GET /countries will list all available countries;
6. A GET /country/{code} will show a single country. c. Using the simple client will log into the console of the result of the two available endpoints.

**File Data**

A sample file accompanies this document for the assignment with the name countries-20140629.csv.

1. **Solution**

**GitHub**

<https://github.com/GledsonOliveira/assignment>

**Prerequisites**

Install MongoDB database; <https://docs.mongodb.com/manual/installation/>



Install Wildfly Community Edition <https://www.wildfly.org/downloads/>

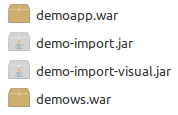


Install Java 11 or Superior <https://openjdk.java.net/projects/jdk/11/>



**Artifacts**

|  |  |
| --- | --- |
| **Artifact** | **Description** |
| demo-import.jar | Module responsible to read CSV file and import the data into the database. |
| demo-import- visual.jar | Visual Application responsible to read CSV file and import the data into the database. |
| demows.war | Module web service (HTTP/REST) to expose country information. |
| demoapp.war | Client Web Application for consulting web service (demows.war) |

**Image list:**

1. **Module Import CSV File**

You have two options for import csv file into database. First option is a simple java application. Second option is Java Swing Application.

Theses solutions attend business requirement number 1.

**Option 1 - Import CSV File**

**Description**

Simple java application (demo-import.jar) it will call *mongoimport* tool for import csv file into MongoDB.

This application will create one database called ***demoDB*** and one collection called ***countries***.

**Installation**

You must choose a directory to put the file demo-import.jar.

**Execute command with parameters**

$ java -jar demo-import.jar [server] [port] [file path]

[server] mongodb server information. Ex.: localhost



[port] mongodb port. Ex.: 27017



[file path] file path to csv file.



**Example:**

$ java -jar demo-import.jar localhost 27017 /home/user/countries-20140629.csv

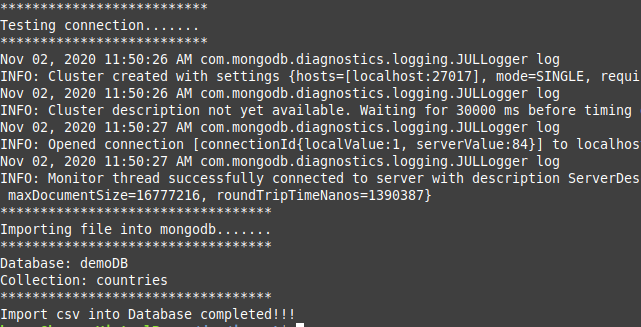
**Important!!!**

[This option use references mongoImport tool. https://docs.mongodb.com/database-tools/mongoimport/](https://docs.mongodb.com/database-tools/mongoimport/)

**Command:**



**Output:**



**Option 2 - Import CSV File**

**Description**

Visual java swing application using demo-import-visual.jar. This application will create one database called ***demoDB*** and one collection called ***countries*** after import data.

**Installation**

You must choose a directory to put the file demo-import-visual.jar.

**Execute command with parameters**

$ java -jar demo-import-visual.jar

**Example:**

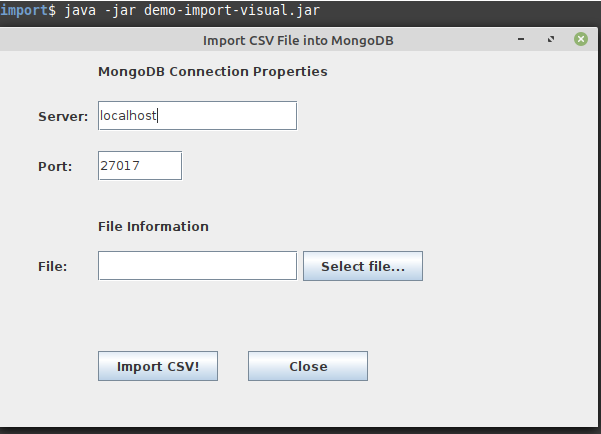
$ java -jar demo-import-visual.jar

You must inform the file path, server and port and click Import CSV button. After importing you can close the application.

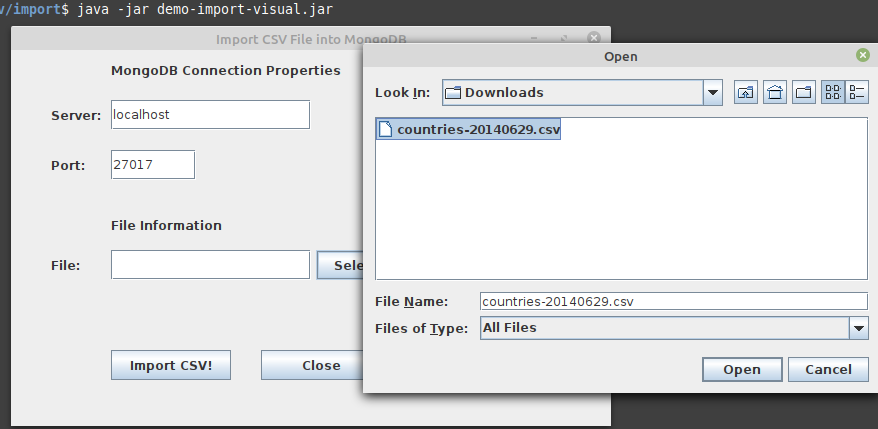
**Important!!!**

[This option use references mongoImport tool. https://docs.mongodb.com/database-tools/mongoimport/](https://docs.mongodb.com/database-tools/mongoimport/)

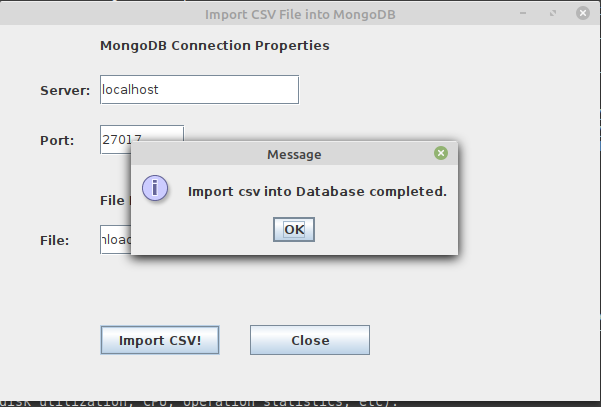
**Imagem demo-import-visual.jar**



**Select csv file…**



**After you choose the file click on the button Import CSV!**



**Message informing that import has finished.**

1. **Module Web Service Rest**

This solution attend business requirement number 2.

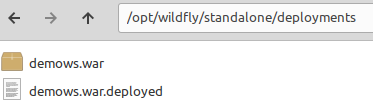
Simple java application using Jax-rs rest and expose 2 http entries for retrieve data from MongoDB.

Database: demoDB

Collection: countries

**Deployment [demows.war]**

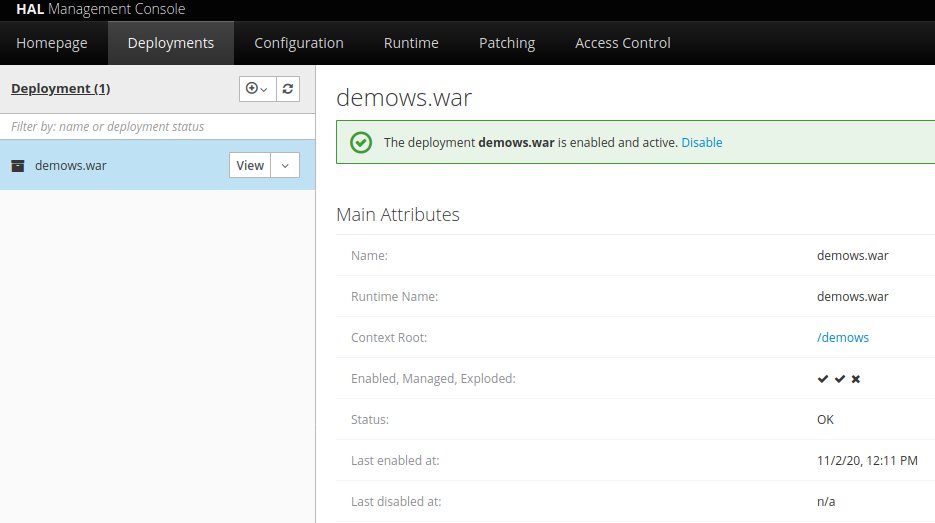
Deployment requires that you copy archived application [demows.war] into the deployments folder from Wildfly Server. Automatically it will be deployed.



If WildFly is stopped you need start. Start server:

*$ WildFly\_HOME/bin/standalone.sh -b=0.0.0.0 (for Linux) or \*.bat (for Windows)*

You can see or make deployments using Management Console from WildFly.



You can use any tools to consume this web service. Ex.: Postman or Curl (linux) or demoapp.war (next step).

Entries:

|  |  |
| --- | --- |
| HTTP | URL |
| GET | http://server:port/demows/ws/rest/country/{code} |
| GET | http://server:port/demows/ws/rest/countries |

Example:

$ curl -v http://localhost:8080/demows/ws/rest/country/PT

* HTTP/1.1 200 OK

{"code":"PT","englishName":"Portugal","frenchName":"Portugal"}

* If not found data the web service will return code http 404 (Not found)

1. **Module Client Web Application**

This solution attend business requirement number 3.

Demoapp use framework JavaServer Faces for building user interfaces on server-side applications.

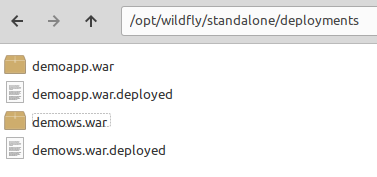
**Requirements**

Application demows.war have to be deployed and up.

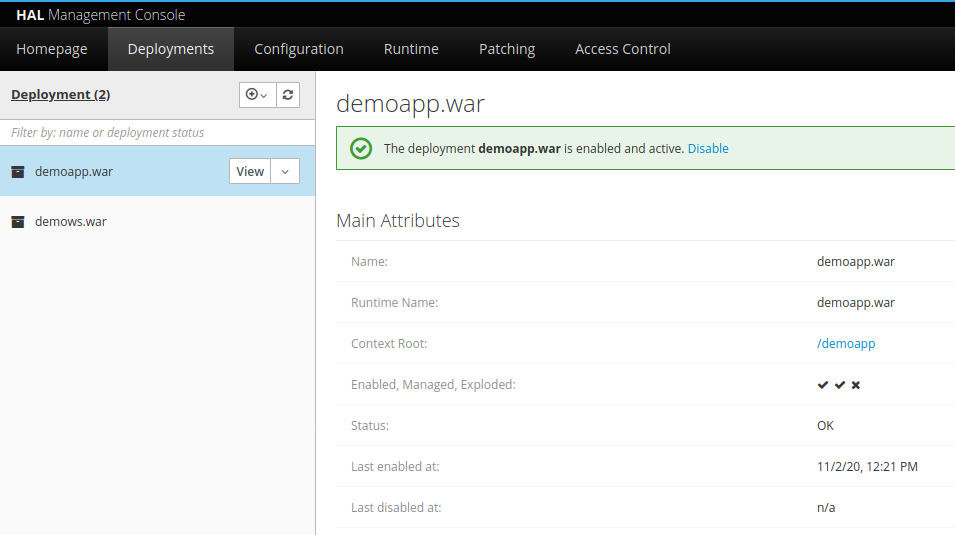
**Deployment [demoapp.war]**

Deployment requires that you copy archived application [demoapp.war] into the deployments folder from Wildfly Server.

Image: Deployments directory

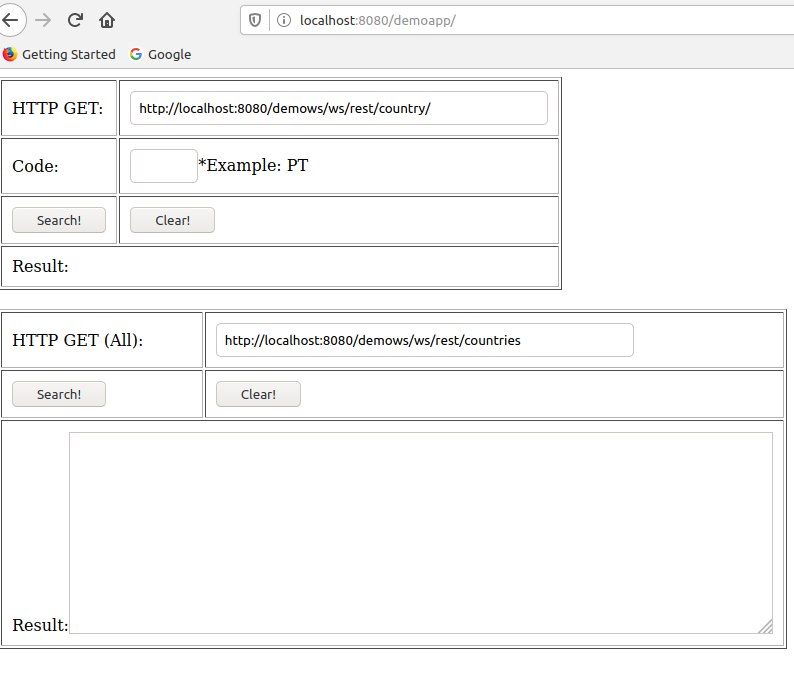


Automatically the war file will be deployed.



Wildfly will expose the application in the URL:

http://localhost:8080/demoapp/

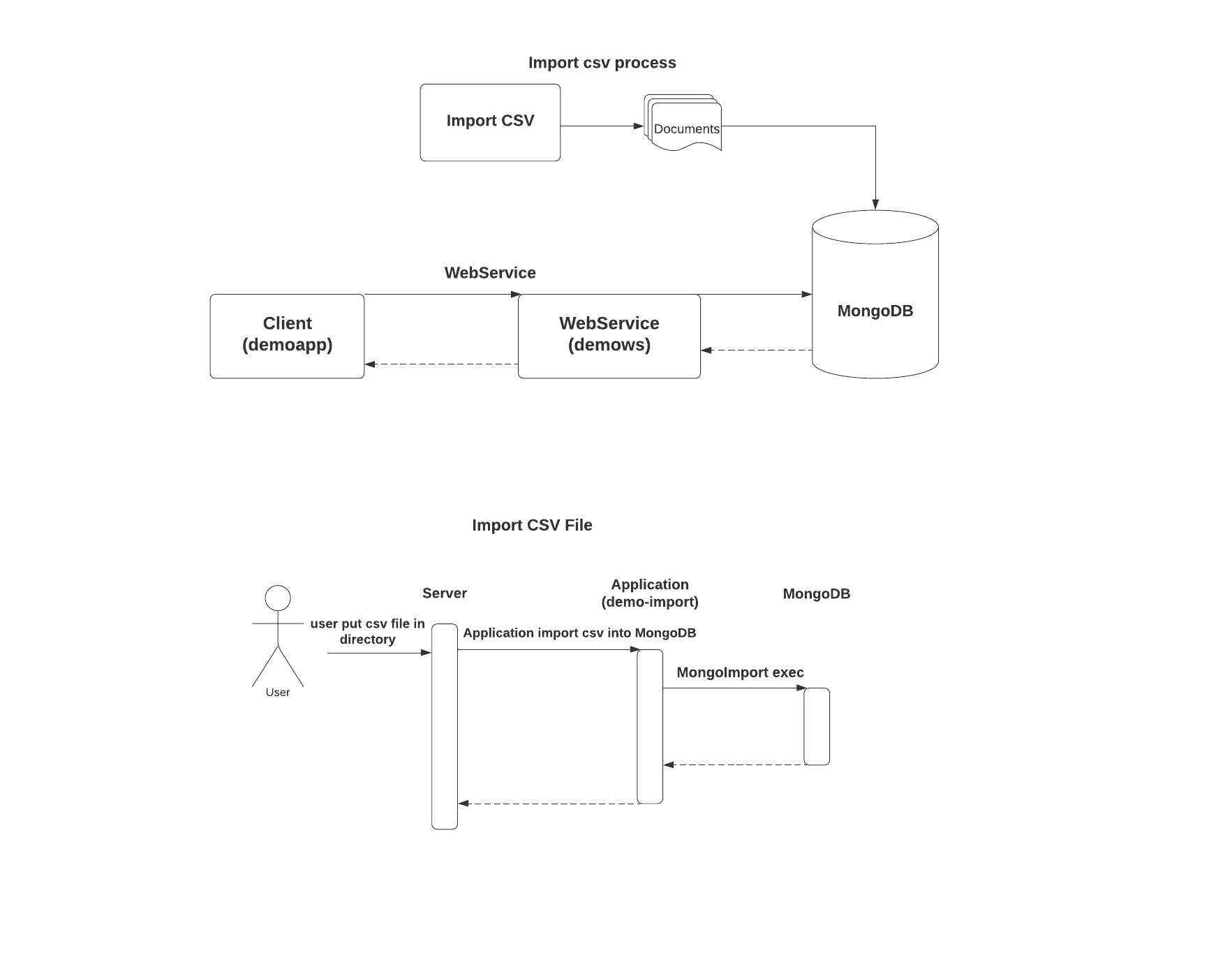


**First panel should bring information filtering by Code.**

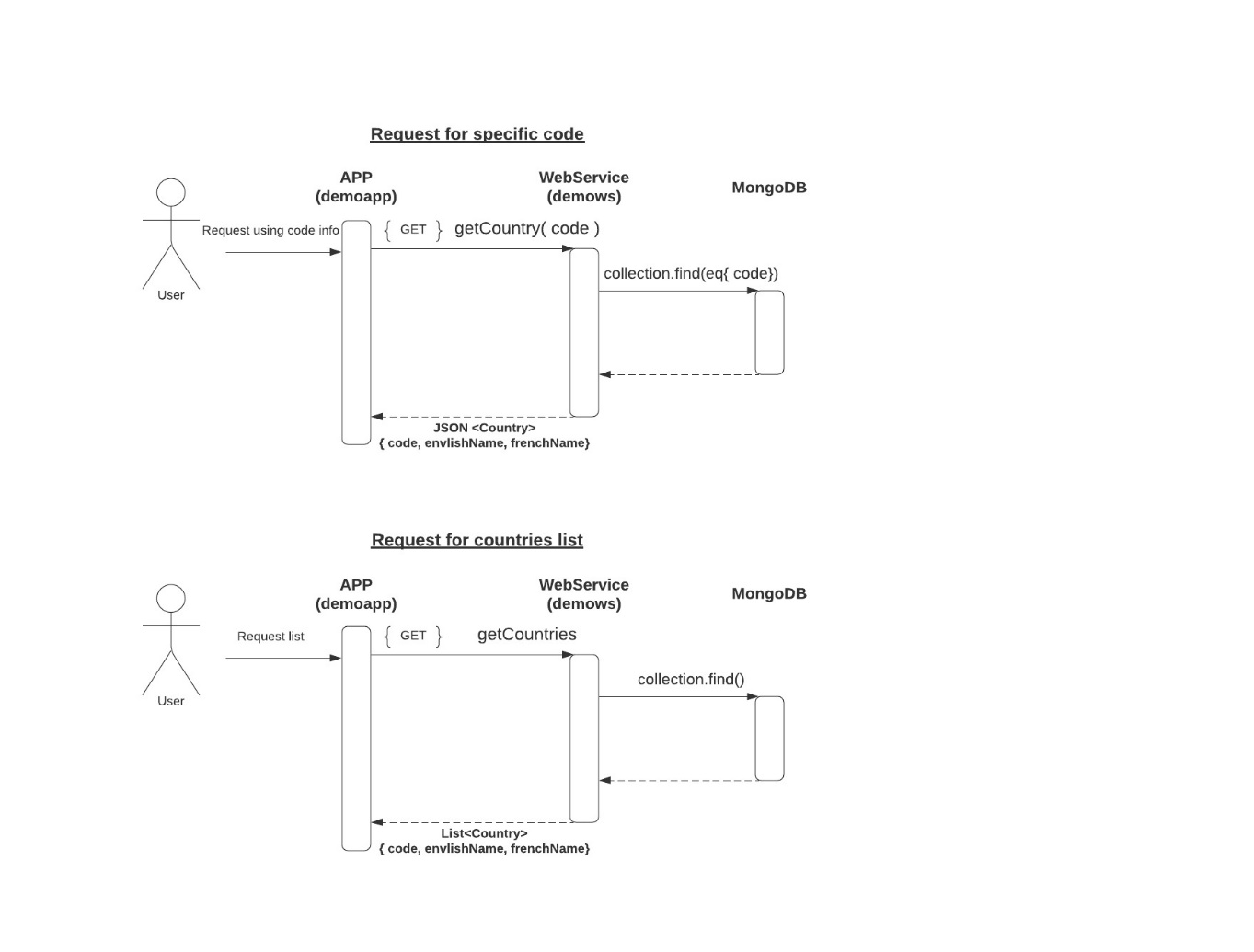
**Second panel should bring all countries information.**

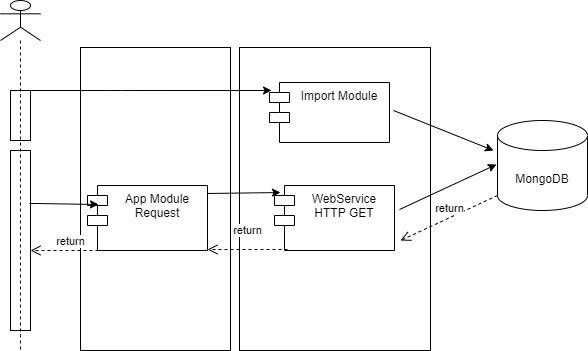
1. **Diagram UML**

**Process Diagram**



**Sequence Diagram**





1. **Junit Test**

Using in demows project.

