Spring Data Exam – 02 April 2022

Real Estate Agency

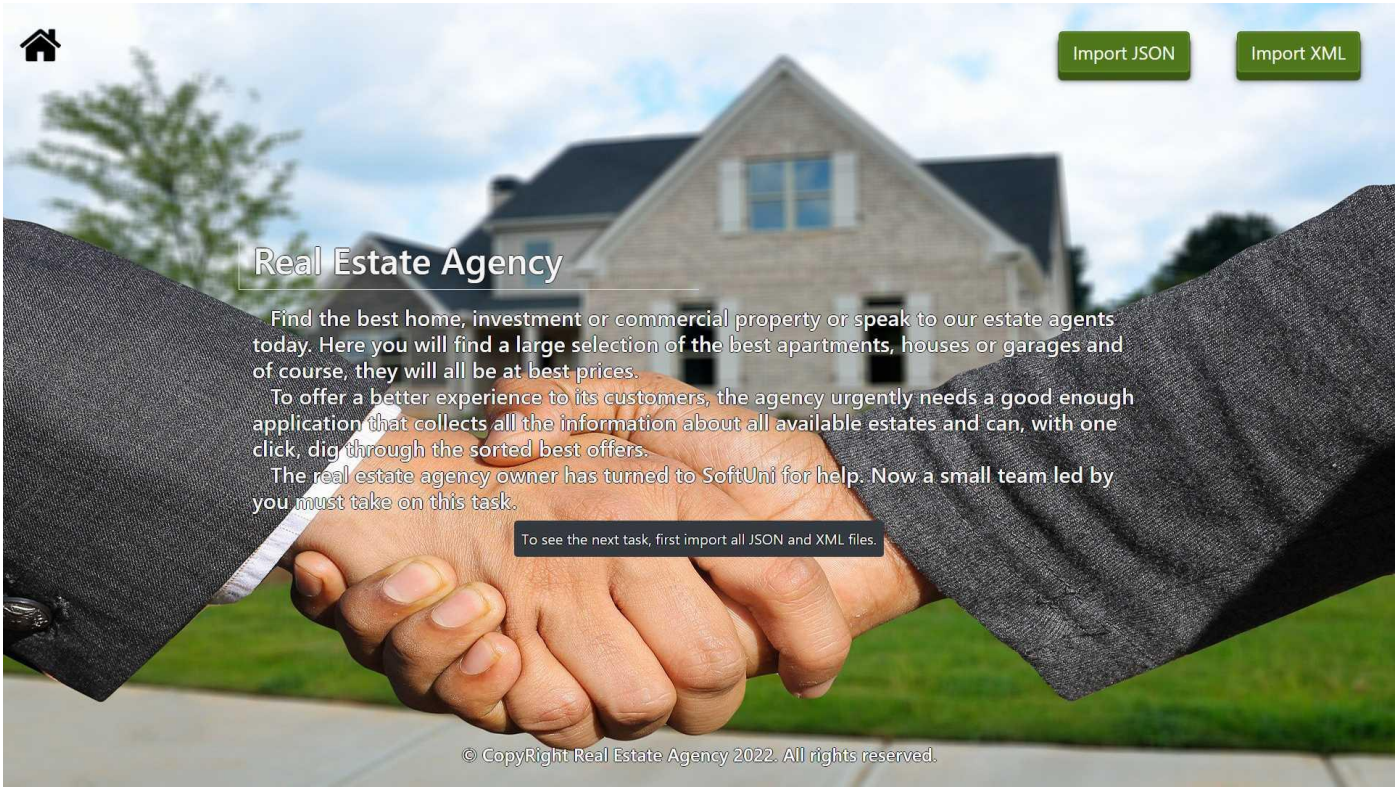
Find the best home, investment or commercial property or speak to our estate agents today. Here you will find a large selection of the best apartments, houses or garages and of course, they will all be at best prices. To offer a better experience to its customers, the agency urgently needs a good enough application that collects all the information about all available estates and can, with one click, dig through the sorted best offers. The real estate agency owner has turned to SoftUni for help. Now a small team led by you must take on this task.

1. Functionality Overview

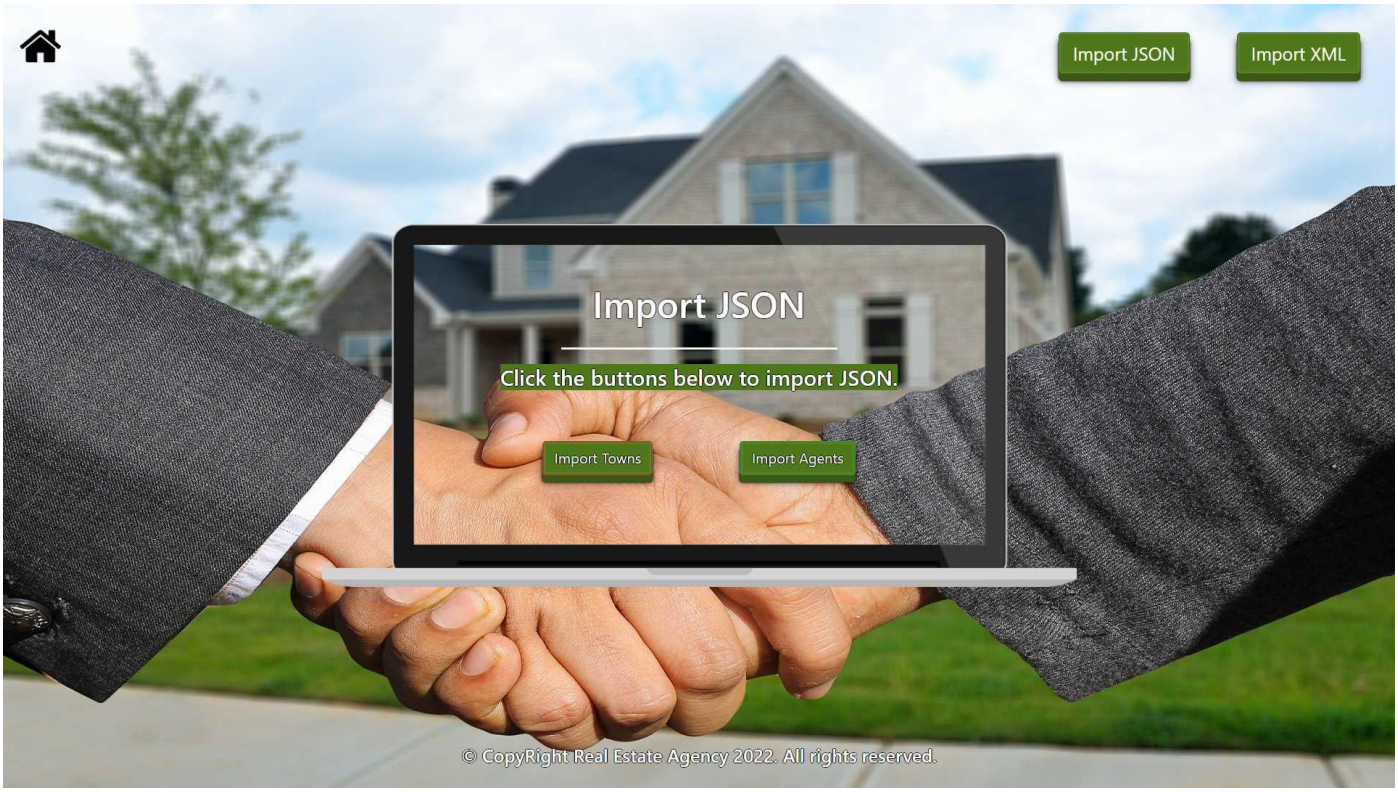
The application should be able to easily **import** hard-formatted data and **support functionalities** for also **exporting** the imported data. The application is called – **Real Estate Agency**.

Look at the pictures below to see what must happen:

- The home page before importing anything:

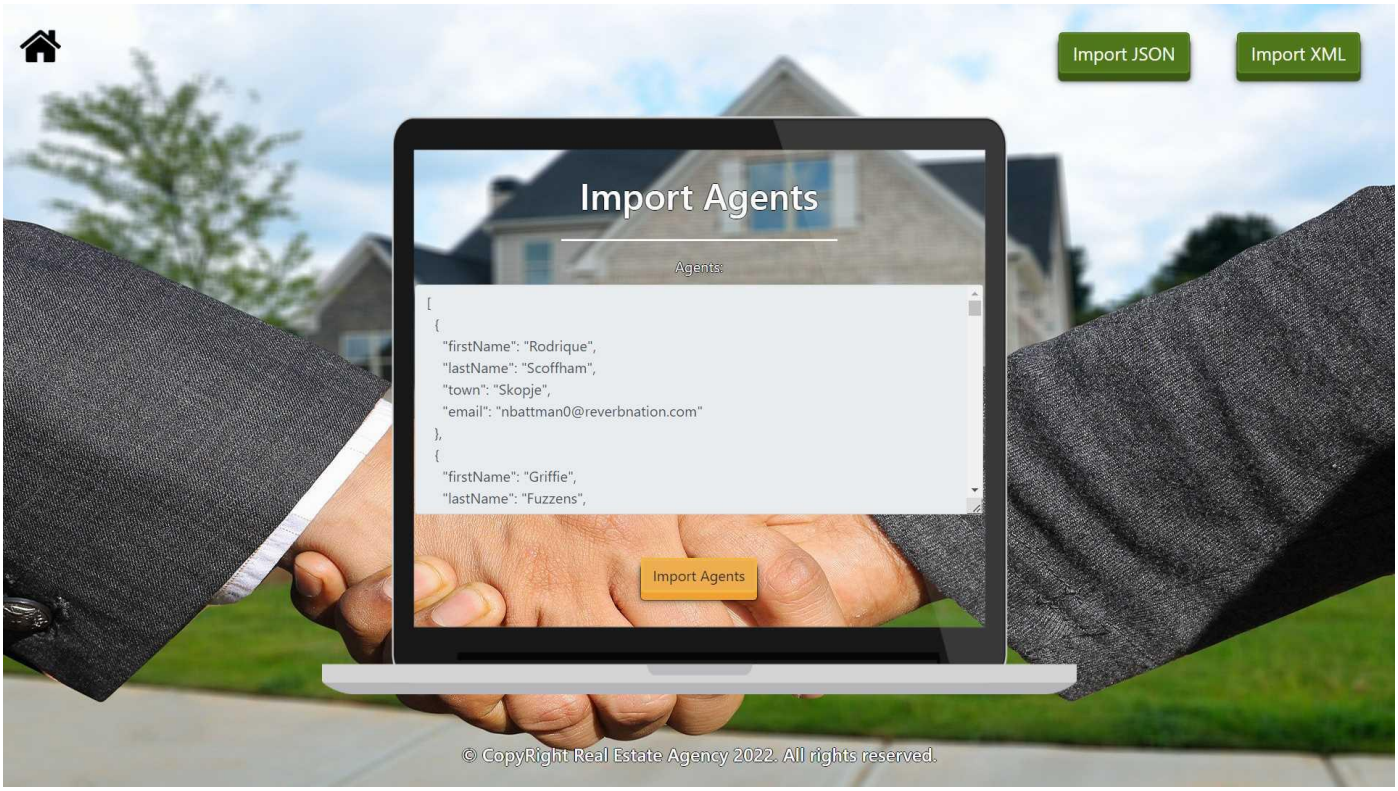


- The import JSON page before importing anything:

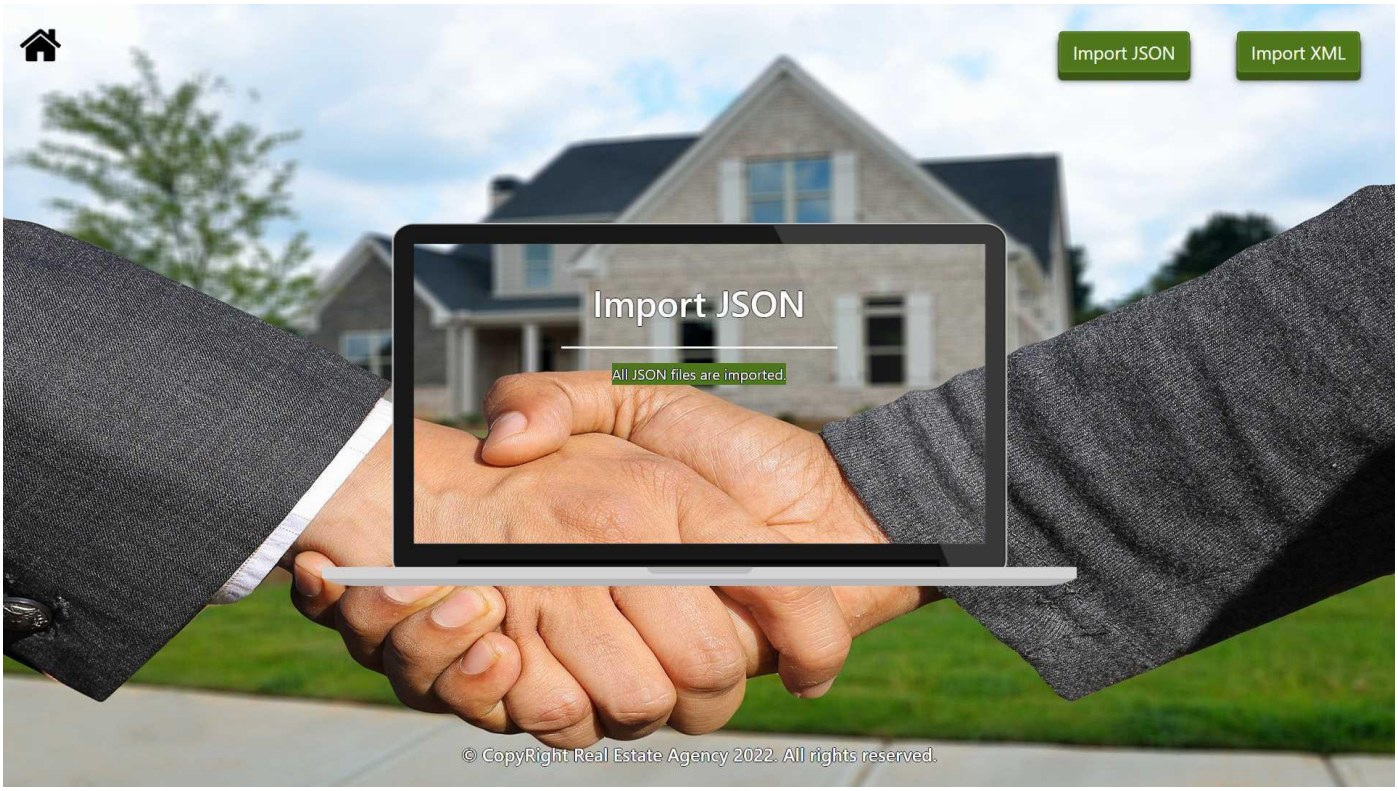


- Import the towns first:

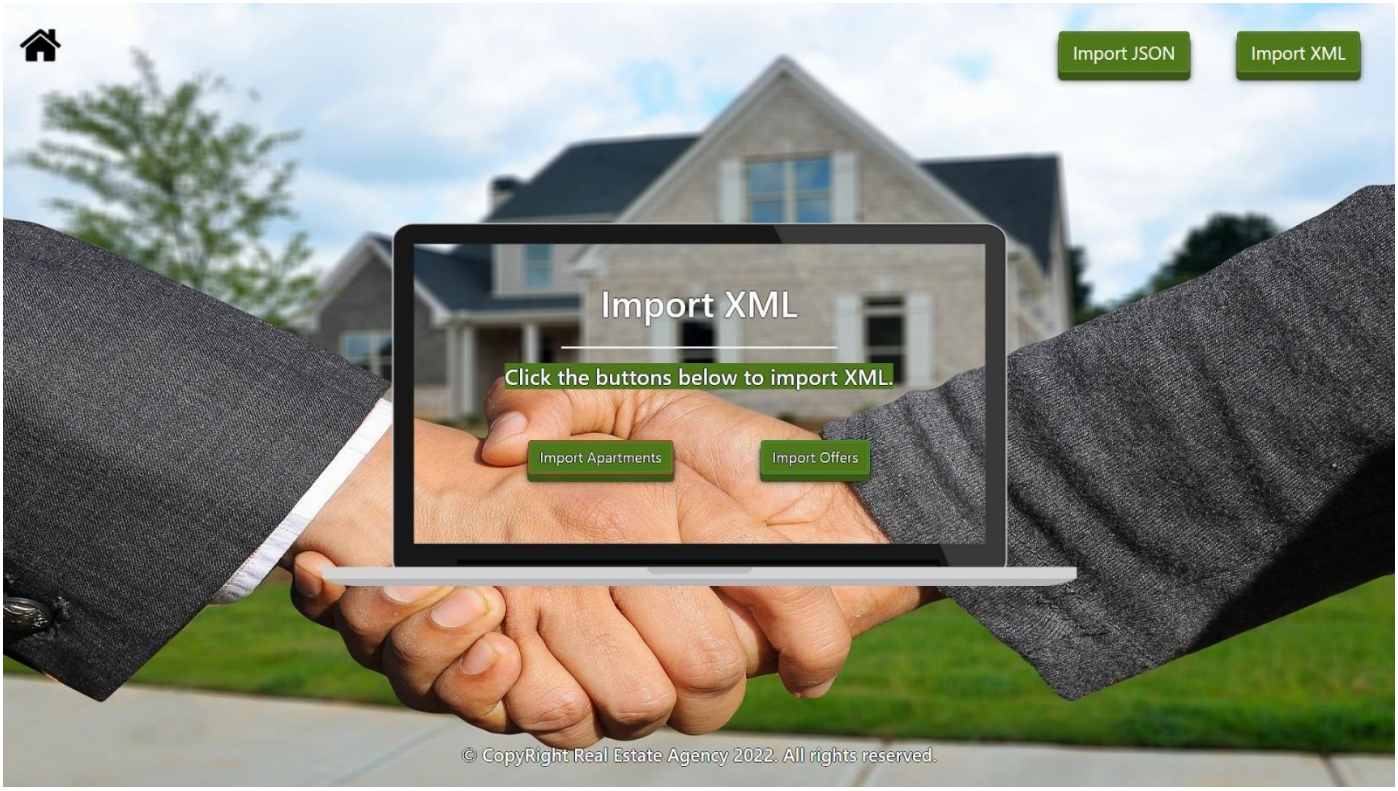
- Import the agents second:



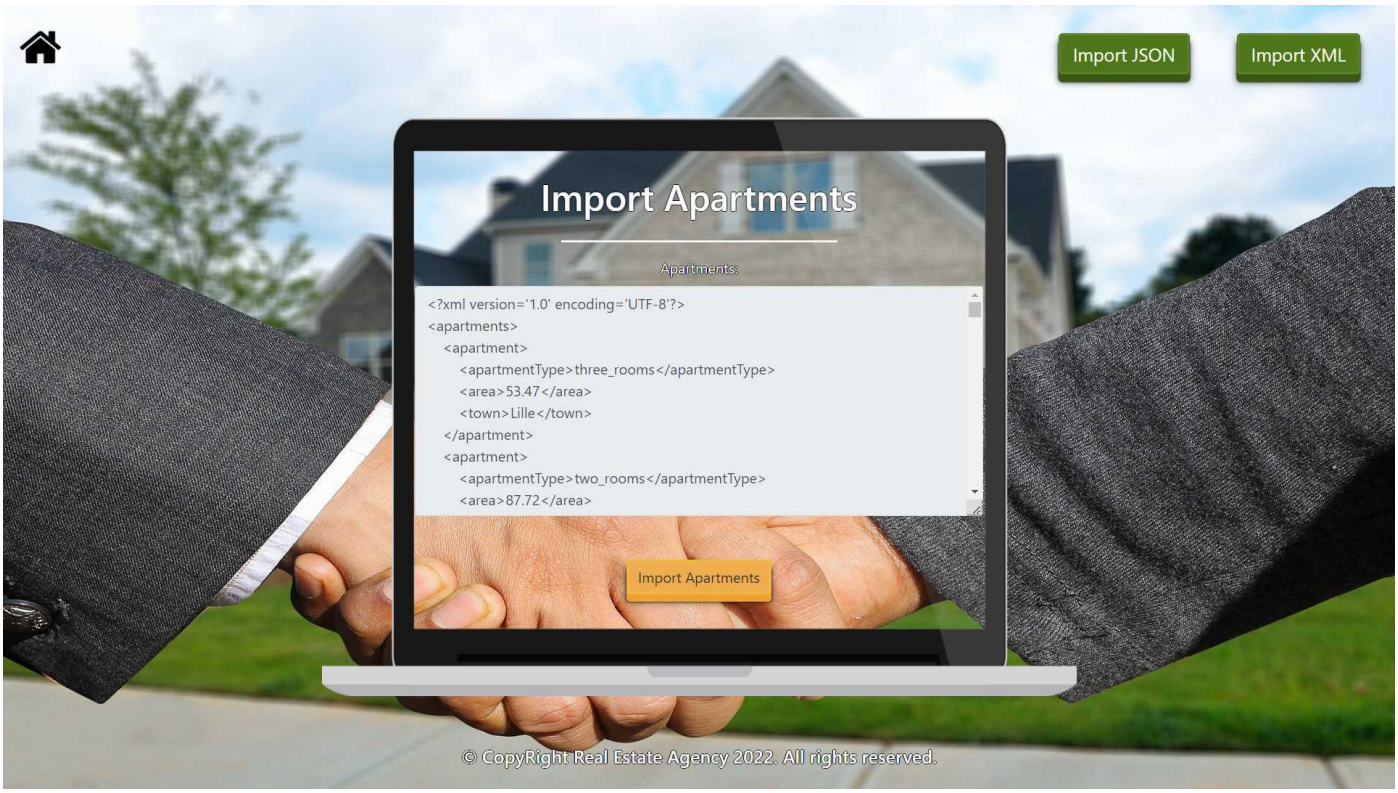
- The import JSON page after importing both files:



- The import XML page before importing the given data:

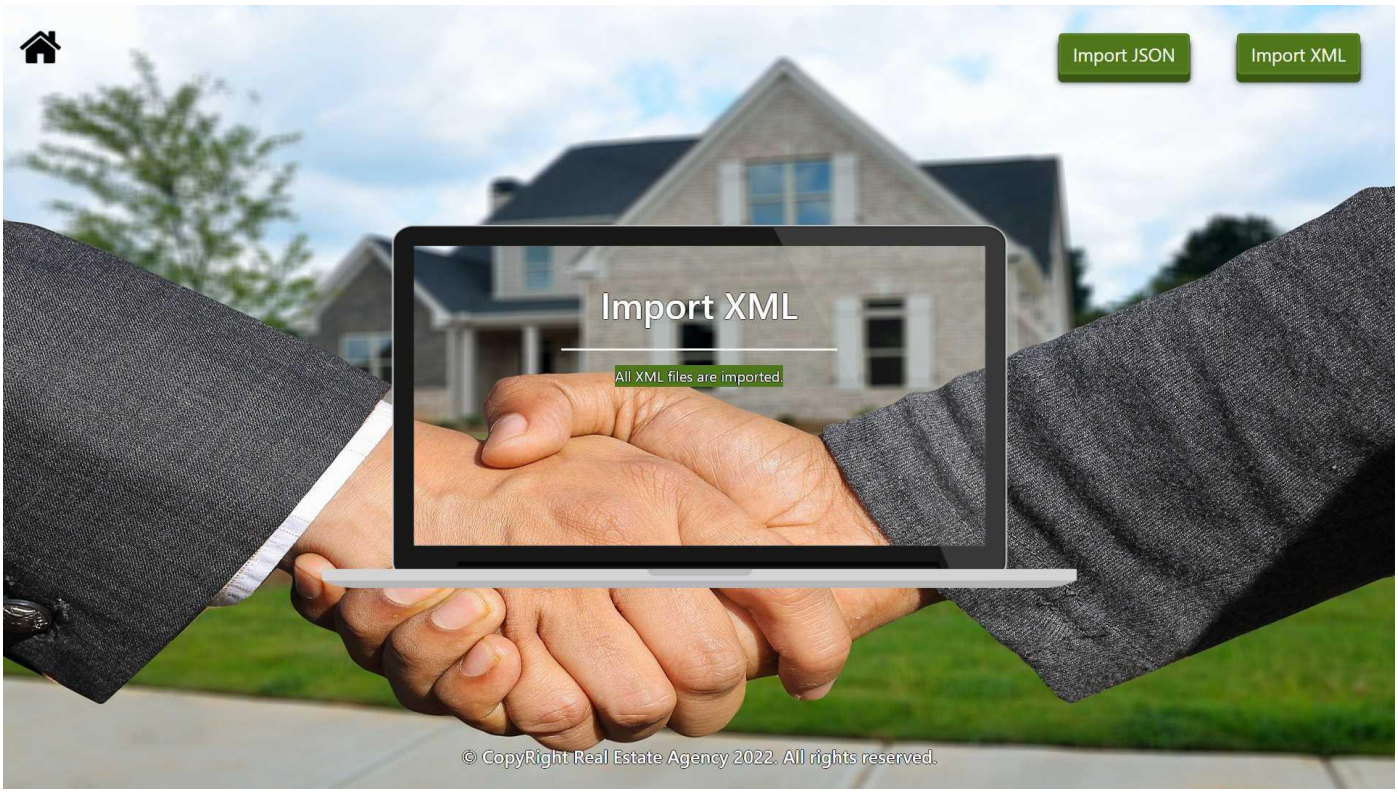


- Import the apartments data:

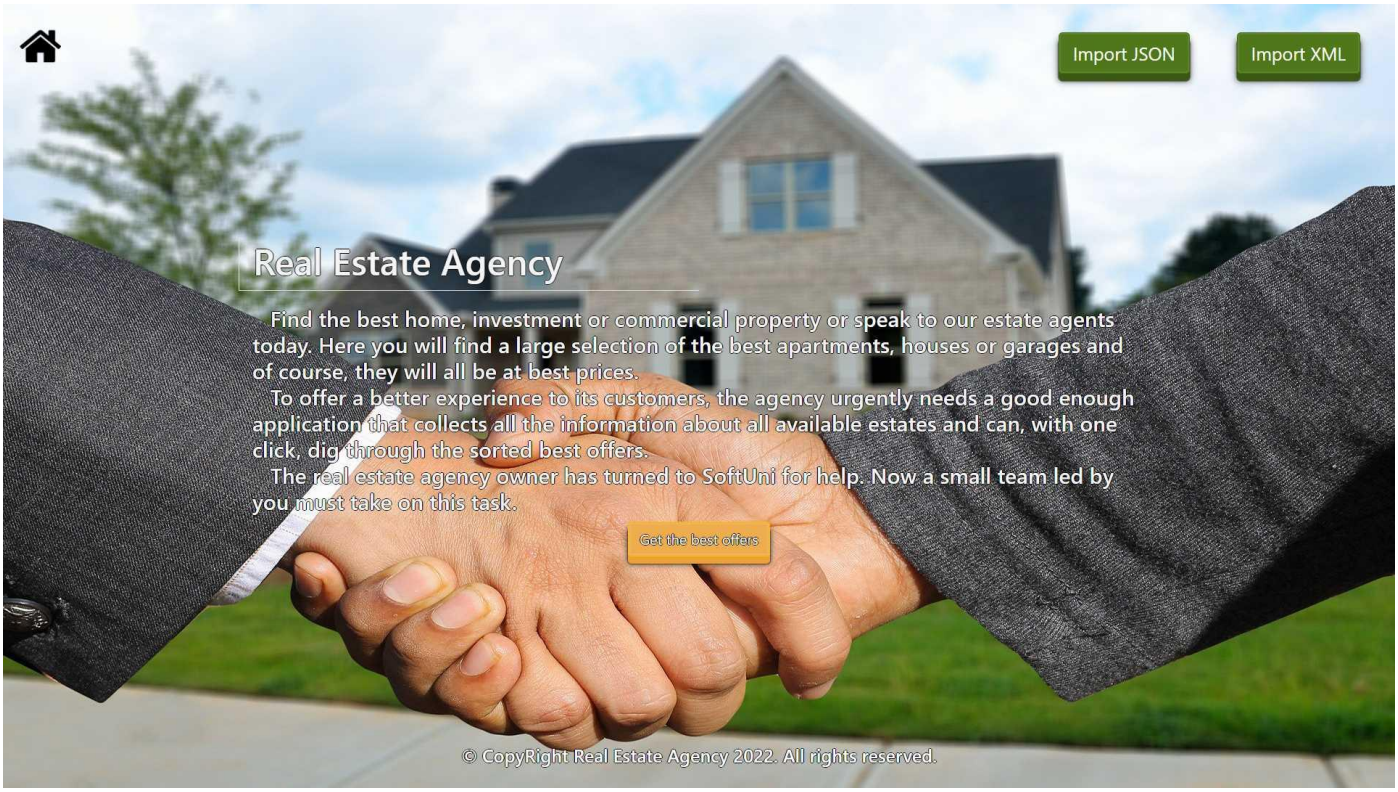


- Import the offers data:

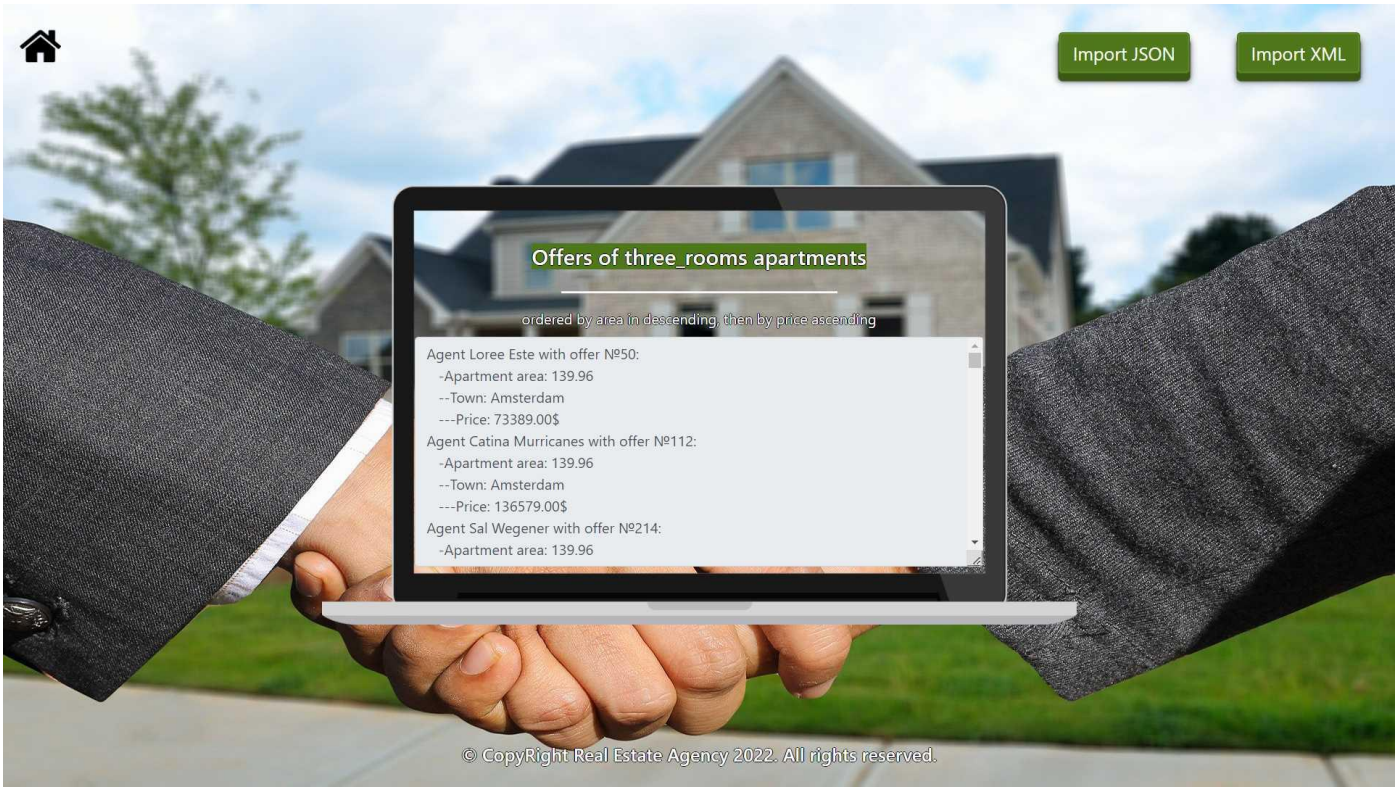
- The import XML page after importing the data:



- The home page after the data is imported:



- Export offers for three_rooms apartments:



2. Project Skeleton Overview

You will be given a **skeleton**, containing a **certain architecture (MVC)** with **several classes**, some of which are completely empty. The **Skeleton** will include the **files** with which you will **seed** the **database**.

3. Model Definition

There are 4 main models that the **Real Estate Agency database** application should contain in its functionality.

Design them in the **most appropriate** way, considering the following **data constraints**:

Town

- **id** – accepts **integer** values, a **primary identification field, an auto incremented field**.
- **town name** – accepts **char sequences** as values where their character length value is **higher than or equal to 2.** The values are **unique in the database**.
- **population** – accepts **number** values (must be positive), 0 as a value is **exclusive**.

Agent

- **id** – accepts **integer** values, a **primary identification field, an auto incremented field**.
- **first name** – accepts **char sequences** as values where their character length value **higher than or equal to 2.** The values are **unique in the database**.
- **last name –** accepts **char sequences** as values where their character length value **higher than or equal to 2.**
- **email** – an **email –** (must contains '@' and '.' – dot). The email of a seller is **unique**.
- **Constraint:** The agents table has a relation with the towns table.

Apartment

- **id** – accepts **integer** values, a **primary identification field, an auto incremented field**.
- **apartment type –** the enumeration, one of the following – **two_rooms, three_rooms, four_rooms**
- **area** – accepts **number** values that are **more** than or **equal** to **40.00.**
- **Constraint:** The apartment table has a relation with the towns table.

Offer

- **id** – accepts **integer** values, a **primary identification field, an auto incremented field**.
- **price** – accepts a **positive number**.
- **published on** – a date in the "**dd/MM/yyyy**" format.
- **Constraint:** The offers table has a relation with the apartments table.
- **Constraint:** The offers table has a relation with the agents table.

Relationships

Your partners gave you a little hint about the more complex relationships in the database, so that you can implement it correctly.

One **Agent** may have only one **Town**, but one **Town** may have many **Agents**.

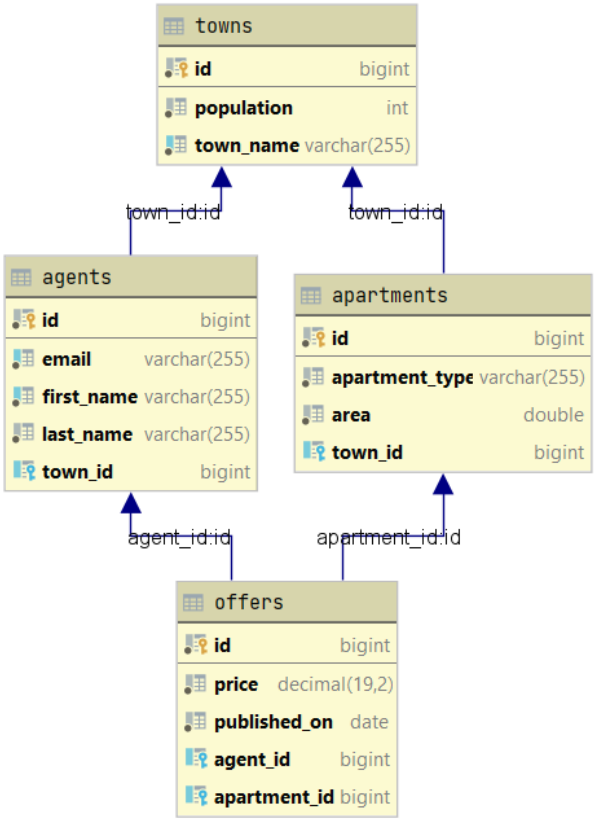
One **Apartment** may have only one **Town**, but one **Town** may have many **Apartments**.

One **Offer** may have only one **Apartment**, but one **Apartment** can be in many **Offers**.

One **Offer** may have only one **Agent**, but one **Agent** can have many **Offers**.

Constraint:

- Name the entities and their class members **exactly** in the **format stated** above.
- All fields are **NOT NULL** unless explicitly stated to be nullable.



4. Data Import

Use the provided files to populate the database with data. Import all the information from those files into the database.

You are not allowed to modify the provided files.

ANY INCORRECT data should be **ignored** and a message:

"**Invalid {town / agent / apartment / offer}**" should be printed.

When the import is finished:

"**Successfully imported {town / agent / apartment / offer} {name - population / first name last name / type – area / price}**"

JSON Import

Your new colleagues have prepared some JSON data for you to import.

Towns (towns.json)

```
[
  {
    "townName": "Matingain",
    "population": 2162142
  },
  {
    "townName": "Frankfurt",
    "population": -140256
  },
  {
    "townName": "Prague",
    "population": 2651904
  },
  {
    "townName": "Tubuhue",
    "population": 1886927
  },
  {
    "townName": "General Vedia",
    "population": 1592612
  },
  {
    "townName": "Xunzhong",
    "population": 3091280
  },
...
```

```
Successfully imported town Matingain - 2162142
Invalid town
Successfully imported town Prague - 2651904
Successfully imported town Tubuhue - 1886927
Successfully imported town General Vedia - 1592612
Successfully imported town Xunzhong - 3091280
...
```

Constraint:

- **If the agent's first name already exists in the DB return "Invalid agent".**

Agents (agents.json)

```
[
  {
    "firstName": "Rodrique",
    "lastName": "Scoffham",
    "town": "Skopje",
    "email": "nbattman0@reverbnation.com"
  },
  {
    "firstName": "Griffie",
    "lastName": "Fuzzens",
    "town": "Amsterdam",
    "email": "cfitchell1@edublogs.org"
  },
  {
    "firstName": "Griffie",
    "lastName": "Bergman",
    "town": "Oslo",
    "email": "cfitcl1@ublogs.com"
  },
  {
    "firstName": "Kimberlee",
    "lastName": "Goshawk",
    "town": "Oslo",
    "email": "alosty2@slate.com"
  },
  {
    "firstName": "s",
    "lastName": "Petet",
    "town": "Amsterdam",
    "email": "fal1@edugs.org"
  },
  {
    "firstName": "Rickys",
    "lastName": "D",
    "town": "Minsk",
    "email": "shell1@blogs.org"
  },
  {
    "firstName": "Erastus",
    "lastName": "Clute",
    "town": "Oslo",
    "email": "blefeaver3@cpanel.net"
  },
  ...
```

```
Successfully imported agent - Rodrique Scoffham
Successfully imported agent - Griffie Fuzzens
Invalid agent
Successfully imported agent - Kimberlee Goshawk
Invalid agent
Invalid agent
Successfully imported agent - Erastus Clute
…
```

XML Import

Your new colleagues have prepared some XML data for you to import.

Constraint:

- **If the apartment with the same town name and area already exists in the DB return "Invalid apartment".**

- **The provided town names will always be valid.**

Apartments (apartments.xml)

Apartments (apartments.xml)

```
<?xml version='1.0' encoding='UTF-8'?>
<apartments>
    <apartment>
        <apartmentType>three_rooms</apartmentType>
        <area>53.47</area>
        <town>Lille</town>
    </apartment>
    <apartment>
        <apartmentType>two_rooms</apartmentType>
        <area>87.72</area>
        <town>Nicosia</town>
    </apartment>
    <apartment>
        <apartmentType>three_rooms</apartmentType>
        <area>63.52</area>
        <town>Sofia</town>
    </apartment>
    <apartment>
        <apartmentType>two_rooms</apartmentType>
        <area>53.47</area>
        <town>Lille</town>
    </apartment>
    <apartment>
        <apartmentType>two_rooms</apartmentType>
        <area>40.45</area>
        <town>Nicosia</town>
    </apartment>

...
```

```
Successfully imported apartment three_rooms - 53.47
Successfully imported apartment two_rooms - 87.72
Successfully imported apartment three_rooms - 63.52
Invalid apartment
Successfully imported apartment two_rooms - 40.45
...
```

Offer (offers.xml)

Constraint:

- **If agent with the given name doesn't already exist in the DB return "Invalid offer".**
- **The provided apartment ids will always be valid.**
- **Format the price to the second digit after the decimal point.**

Offers (offer.xml)

```
<?xml version='1.0' encoding='UTF-8'?>
<offers>
    <offer>
        <price>206934.00</price>
        <agent>
            <name>Dotti</name>
        </agent>
        <apartment>
            <id>56</id>
        </apartment>
        <publishedOn>28/12/2005</publishedOn>
    </offer>
    <offer>
        <price>178562.00</price>
        <agent>
            <name>Pablo</name>
        </agent>
        <apartment>
            <id>21</id>
        </apartment>
        <publishedOn>18/11/2011</publishedOn>
    </offer>

    <offer>
        <price>857076.00</price>
        <agent>
            <name>Glennie</name>
        </agent>
        <apartment>
            <id>49</id>
        </apartment>
        <publishedOn>26/07/2005</publishedOn>
    </offer>
    <offer>
        <price>648855.00</price>
        <agent>
            <name>Chadd</name>
        </agent>
        <apartment>
            <id>58</id>
        </apartment>
        <publishedOn>25/08/2014</publishedOn>
    </offer>
...
```

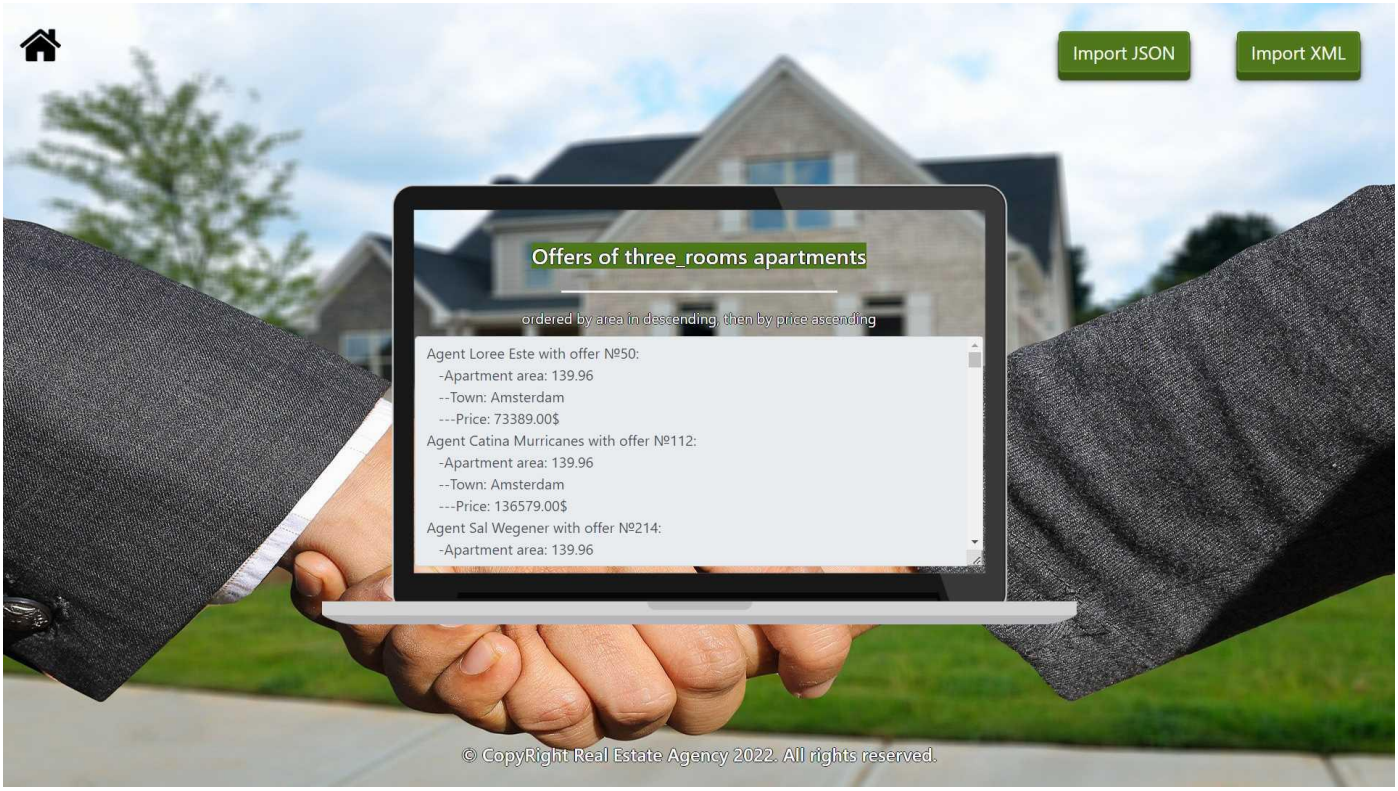
```
Successfully imported offer 206934.00
Invalid offer
Successfully imported offer 857076.00
Successfully imported offer 648855.00
```

5. Data Export

Get ready to export the data you have imported in the previous task. Here you will have some complex database querying. Export the data in the formats specified below.

Export The Best Offers from the Data Base

- Extract from the database, the **agent full name, offer's id**, **apartment area (to second digit after decimal point), town name of the apartment** and the **price (to second digit after decimal point)** of the **offer**.
- **Filter only three_rooms apartments and order them by the area in descending order, then by the price in ascending order.**
- Return the information in this format:
- ```
  "Agent {firstName} {lastName} with offer №{offerId}:
        -Apartment area: {area}
        --Town: {townName}
        ---Price: {price}$
  . . . "
  ```


  ```