Spring Data Retake Exam – 3 April 2020

Airline Company

A new airline company is need of a fresh developer to work on their new project and you're the right person for the job. You're tasked to work on their project called "Airline". The application should accept data from familiar formats (json & xml) and return the data that is listed. It needs to hold the information of all the tickets, that are being bought, all the passengers, their destination, etc.

1. Functionality Overview

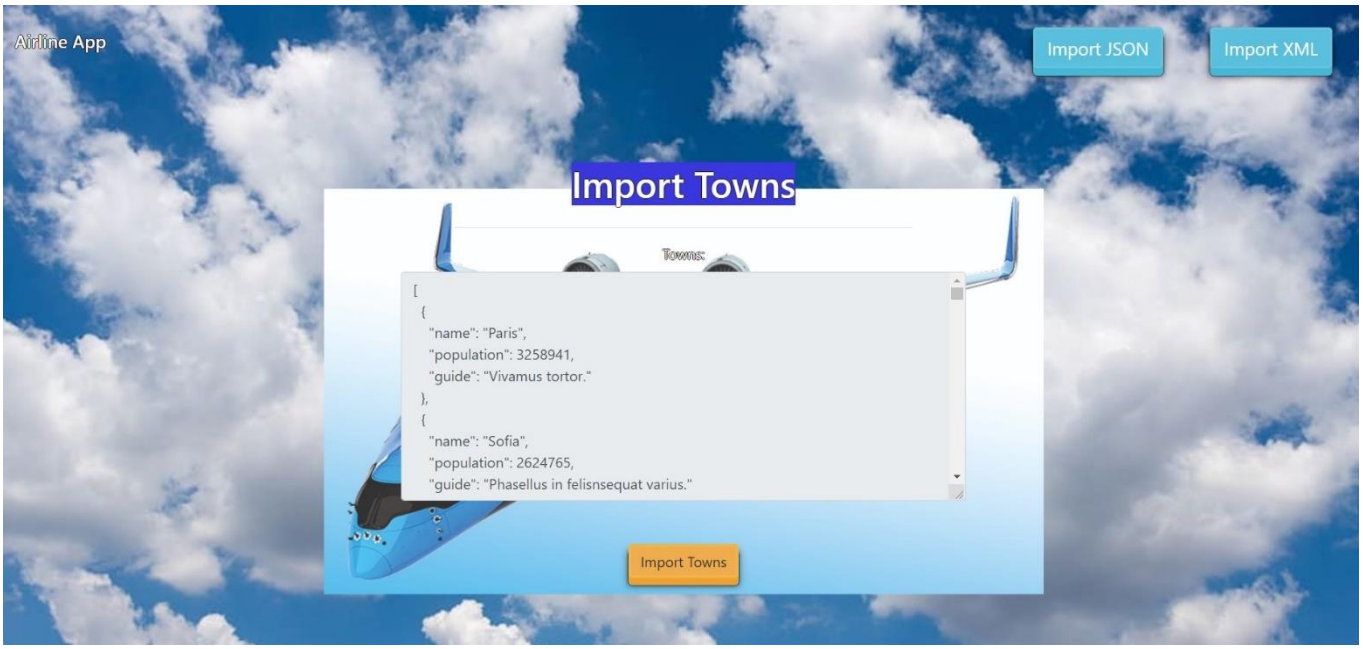
The application should be able to easily **import** hard-formatted data and **support functionality** for also **exporting** the imported data. The application is called – **Airline App**.

Look at the pictures below to see what must happen:

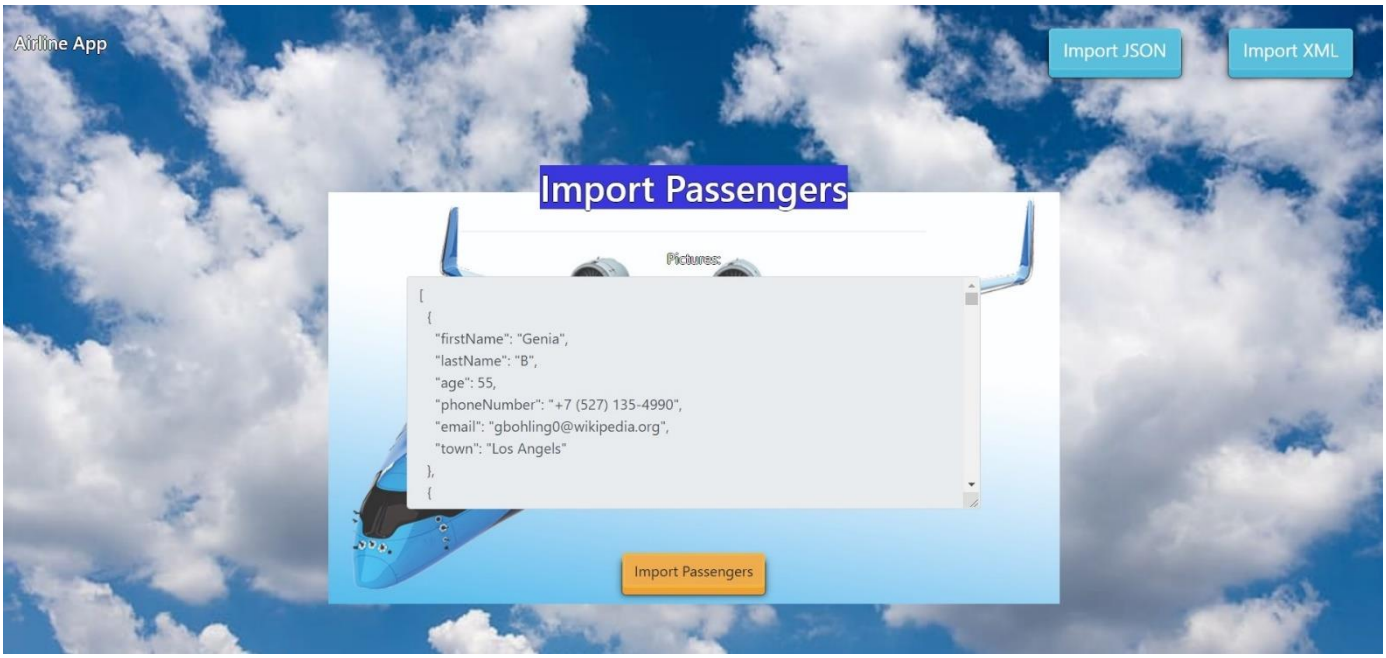
- Home page before importing anything:

- Import JSON page before importing anything:

- Import Towns first:

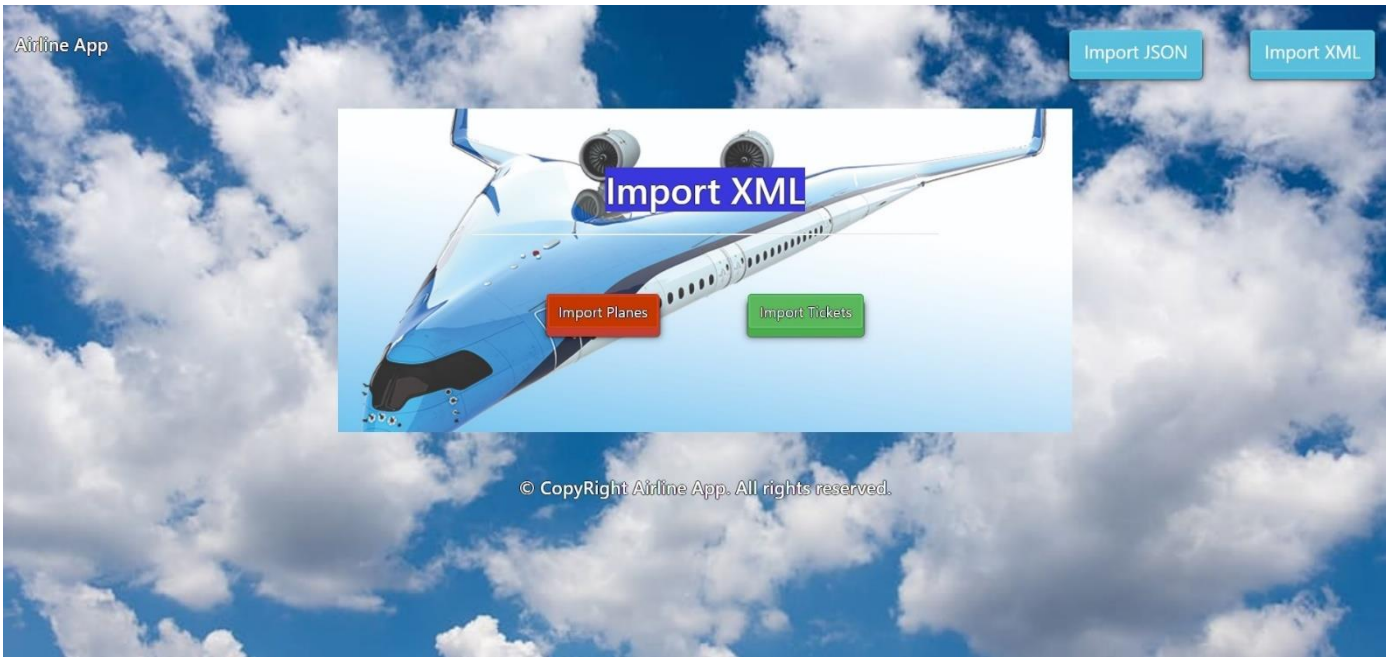


- Import Passengers second:

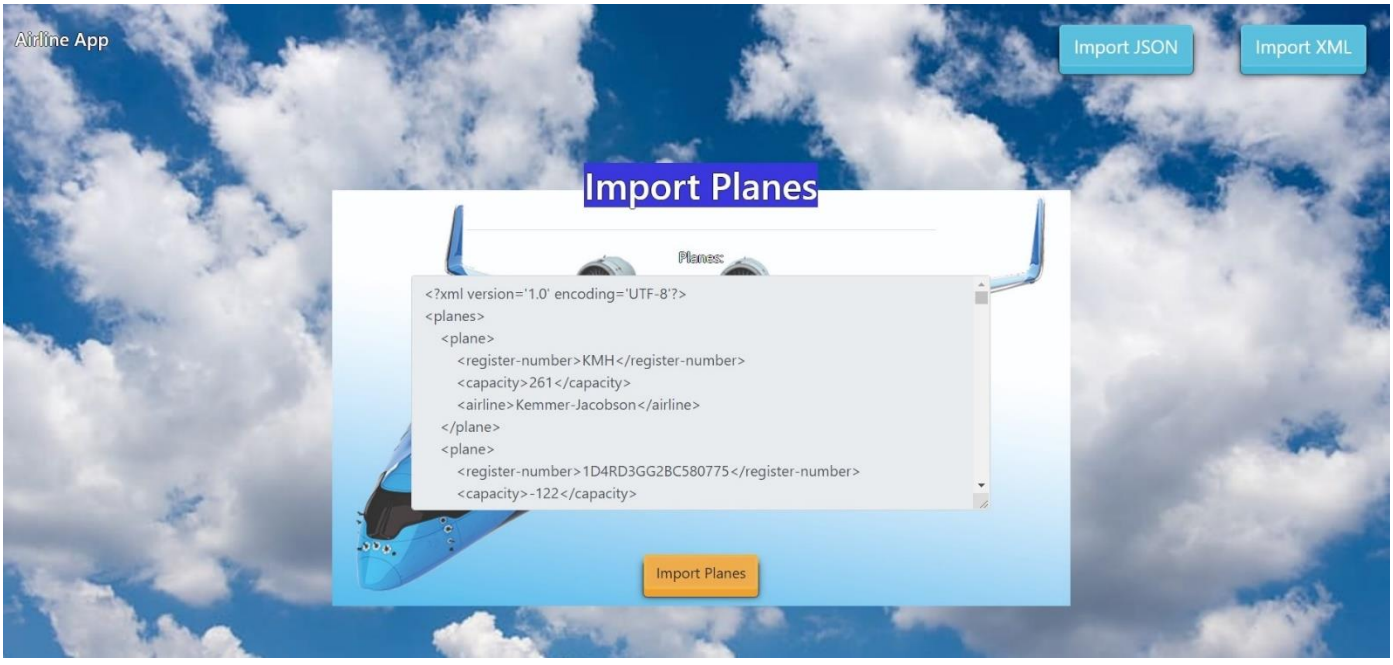


- Import JSON page after importing both files:

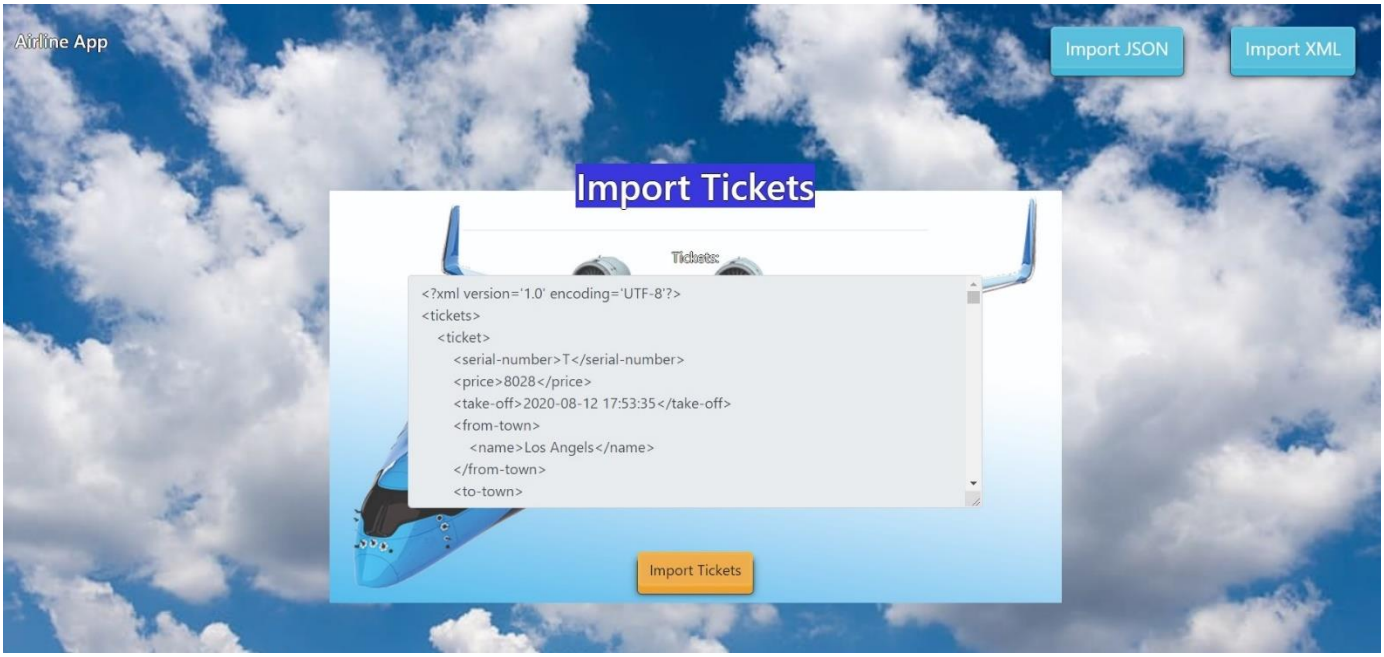
- Import XML page before importing the given data:



- Import Planes data:



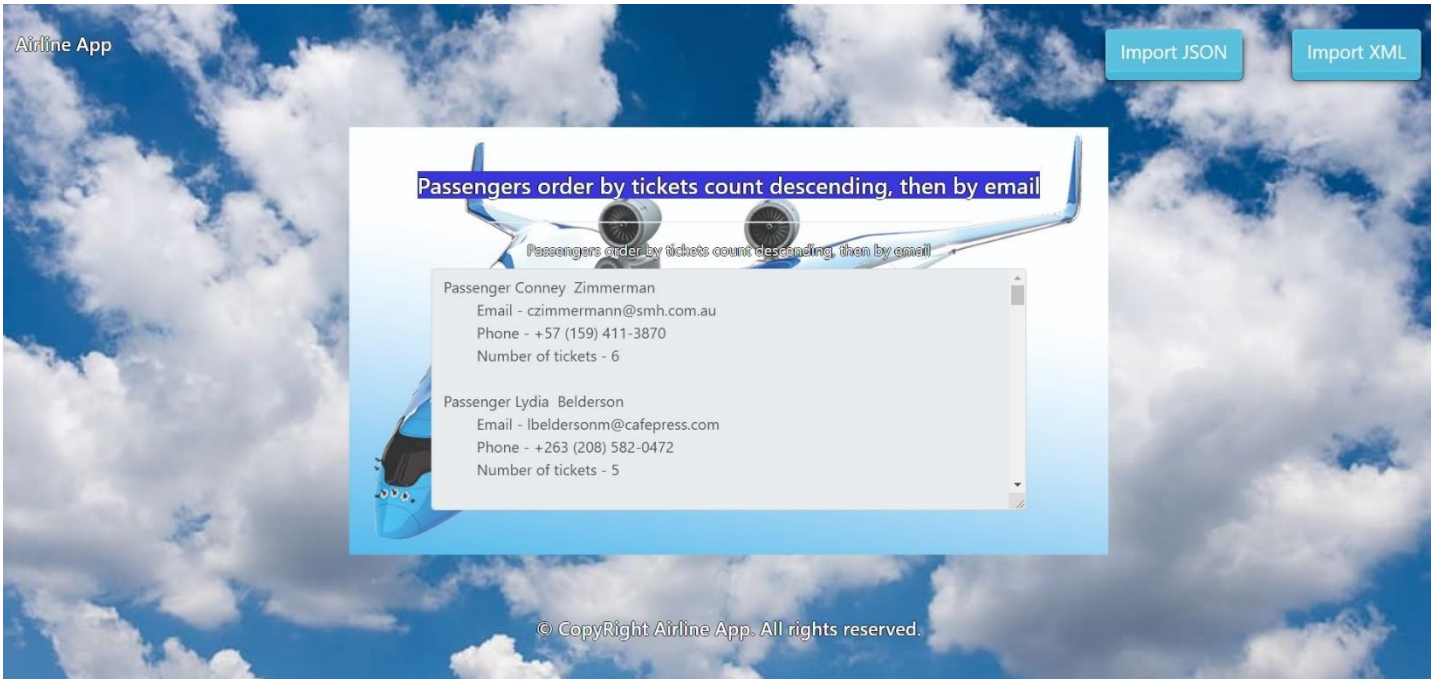
- Import Tickets data:



- Import XML page after importing the data:

- Home page after the data is imported:

- Export passengers by tickets count descending, then by email:



2. Project Skeleton Overview

You will be given a **Skeleton**, containing a **certain architecture(MVC)** with **several classes**, some of which – 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 **Airline database** application should contain in its functionality.

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

Town

- **id** – **integer** number, **primary identification field**.
- **name** – a **char sequence** with **minimum** length **2**. The **name** is **unique**.
- **population** – a **number** (must be positive).
- **guide – Long** and **detailed description** of all known places

Passenger

- **id** – **integer** number, **primary identification field**.
- **firstName** – a **char sequence** with **minimum** length **2**.
- **lastName** – a **char sequence** with **minimum** length **2**.
- **age** – a **number** (must be positive).
- **phoneNumber** – a **char sequence –** phone number.
- **email** – an **email –** (must contains '@' and '.' – dot). The **email** of a **person** is **unique**.
    - **Note**: Passenger has a relation with Town

Plane

- **id** – **integer** number, **primary identification field**.
- **registerNumber** – a **char sequence** (**minimum** length **5**). The register number is **unique**.
- **capacity –** number of passenger (**must** be positive).
- **airline** – name of the airline company with **min length** of **2.**

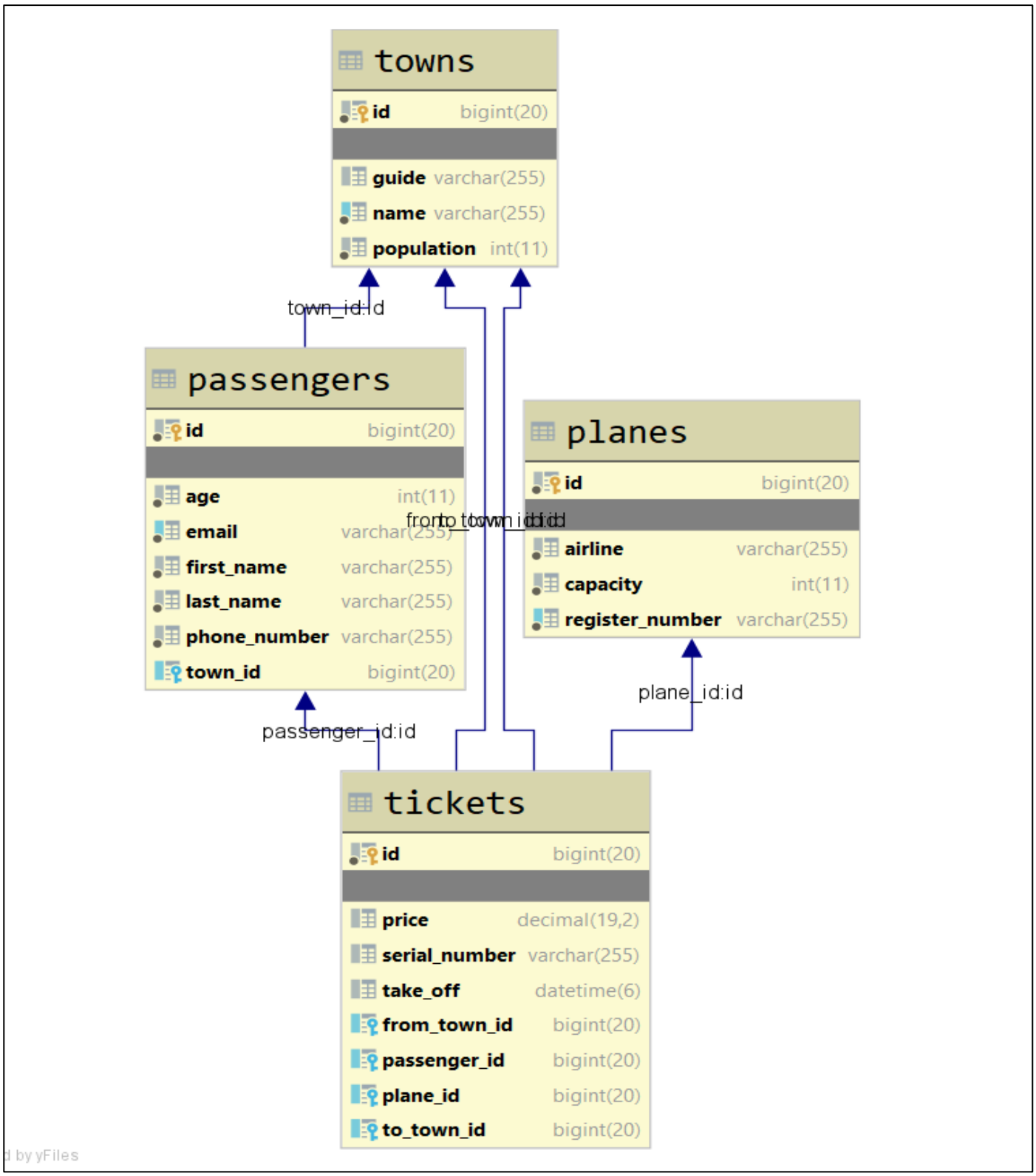
Ticket

- **id** – **integer** number, **primary identification field**.
- **serialNumber** – a combination from letters and numbers with **minimum length** of **2** .

    The serial numbers are **unique**.

- **price** – a price of the ticket. **Must** be **positive**.
- **takeoff** – a **date** and **time** of plane taking off.
    - **Note:** Tickets have two **foreign keys** to Town, because of **fromTown**(Town) and **toTown**(Town)
    - **Note2**: Tickets have relations with Towns, Passengers and Planes.

NOTE: Name the entities and their class members, **exactly** in the **format stated** above. Do not name them in snake case with the dashes, of course.



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 / Passenger / Plane / Ticket}**" should be printed.

When the import is finished:

"**Successfully imported {Town / Passenger / Plane / Ticket} {name - population / lastName - email / registerNumber / fromTown - toTown}**"

JSON Import

Towns (towns.json)

towns.json

```
[
  {
    "name": "Paris",
    "population": 3258941,
    "guide": "Vivamus tortor."
  },
  {
    "name": "Sofia",
    "population": 2624765,
    "guide": "Phasellus in felisnsequat varius."
  },
  {
    "name": "London",
    "population": -4636897,
    "guide": "In sagittis dui ve odio. In hac habitasse platea dictumst."
  },
  {
    "name": "Los Angels",
    "population": 4845321,
    "guide": "Sed ante. Vivamus tortor. Duis mattis egestas metus."
  },
  {
    "name": "M",
    "population": 3644365,
    "guide": "Proin leo odio, porttitor id, consequa Ut at dolor quis odio consequat
varius."
  },


...
```

```
Successfully imported Town Paris - 3258941
Successfully imported Town Sofia - 2624765
Invalid Town
Successfully imported Town Los Angels - 4845321
Invalid Town
…
```

Passengers (passengers.json)

passengers.json


```
[
  {
    "firstName": "Genia",
    "lastName": "B",
    "age": 55,
    "phoneNumber": "+7 (527) 135-4990",
    "email": "gbohling0@wikipedia.org",
    "town": "Los Angels"
  },
  {
    "firstName": "Adams",
    "lastName": "Writer",
    "age": -49,
    "phoneNumber": "+62 (628) 637-1305",
    "email": "awriter1@163.com",
    "town": "Sofia"
  },
  {
    "firstName": "Georgianne",
    "lastName": "McKirdy",
    "age": 71,
    "phoneNumber": "+381 (756) 508-0669",
    "email": "gmckirdy2@opensource.org",
    "town": "Barcelona"
  },
  {
    "firstName": "Shana",
    "lastName": "Leaburn",
    "age": 40,
    "phoneNumber": "+44 (414) 788-3495",
    "email": "sleaburn3ycombinator.com",
    "town": "New York"
  },
  {
    "firstName": "Sim",
    "lastName": "Tordiffe",
    "age": 75,
    "phoneNumber": "+62 (604) 992-8295",
    "email": "stordiffe4@usa.gov",
    "town": "Rome"
  },
  ...
```


Invalid Passenger
Invalid Passenger
Successfully imported Passenger McKirdy - gmckirdy2@opensource.org
Invalid Passenger
Successfully imported Passenger Tordiffe - stordiffe4@usa.gov
…

XML Import

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

Planes (planes.xml)

planes.xml

```xml
<?xml version='1.0' encoding='UTF-8'?>
<planes>
    <plane>
        <registerNumber>KMH</registerNumber>
        <capacity>261</capacity>
        <airline>Kemmer-Jacobson</airline>
    </plane>
    <plane>
        <registerNumber>1D4RD3GG2BC580775</registerNumber>
        <capacity>-122</capacity>
        <airline>Steuber and Sons</airline>
    </plane>
    <plane>
        <registerNumber>19XFB4F27DE919933</registerNumber>
        <capacity>342</capacity>
        <airline>Schaden and Sons</airline>
    </plane>
    <plane>
        <registerNumber>WAUEH94F06N603718</registerNumber>
        <capacity>411</capacity>
        <airline>Cremin LLC</airline>
    </plane>
    <plane>
        <registerNumber>3D7TT2CT7AG224875</registerNumber>
        <capacity>485</capacity>
        <airline>O</airline>
    </plane>
 ...
```

```
Invalid Plane
Invalid Plane
Successfully imported Plane 19XFB4F27DE919933
Successfully imported Plane WAUEH94F06N603718
Invalid Plane
...
```

Tickets (tickets.xml)

tickets.xml

```xml
<?xml version='1.0' encoding='UTF-8'?>
<tickets>
    <ticket>
        <serial-number>T</serial-number>
        <price>8028</price>
        <take-off>2020-08-12 17:53:35</take-off>
        <from-town>
            <name>Los Angels</name>
        </from-town>
        <to-town>
            <name>Sofia</name>
        </to-town>
        <passenger>
            <email>gfraschettil@theglobeandmail.com</email>
        </passenger>
        <plane>
            <register-number>JN1CV6AP3BM793273</register-number>
        </plane>
    </ticket>
    <ticket>
        <serial-number>PT28 3182 7144 4000 7605 6669 2</serial-number>
        <price>-8028</price>
        <take-off>2020-08-12 17:53:35</take-off>
        <from-town>
            <name>Los Angels</name>
        </from-town>
        <to-town>
            <name>Sofia</name>
        </to-town>
        <passenger>
            <email>gfraschettil@theglobeandmail.com</email>
        </passenger>
        <plane>
            <register-number>JN1CV6AP3BM793273</register-number>
        </plane>
    </ticket>
    <ticket>
        <serial-number>LT98 2760 1932 6442 0298</serial-number>
        <price>5211</price>
        <take-off>2020-05-03 06:35:04</take-off>
        <from-town>
            <name>Sofia</name>
        </from-town>
        <to-town>
            <name>Los Angels</name>
        </to-town>
        <passenger>
            <email>czimmermann@smh.com.au</email>
        </passenger>
        <plane>
```

```
            <register-number>3D7TT2CT0BG174323</register-number>
        </plane>
    </ticket>
...
```

```
Invalid Ticket
Invalid Ticket
Successfully imported Ticket Sofia - Los Angels
Successfully imported Ticket New York - Rome
Successfully imported Ticket Milano – Sofia
. . .
```

5. Data Export

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

Export passengers from data base.

Order them by tickets count in descending order, then by email

- Extract from the database, the first name, last name, email, phone and count of tickets. Order them first by tickets count in descending order then by email alphabetically.
- Return the information in this format:

```
"Passenger {firstName}  {lastName}
     Email - {email}
     Phone - {phoneNumber}
     Number of tickets - {number of tickets}
. . . "
```