Spring Data Exam – 29 March 2020

Real Deal

Most people, at least once in their lives, have faced the horror of looking for a good second hand car. We've all heard phrases like "it was driven by a grandmother in Germany", "it stood only in the garage, that's why I'm selling it", "everything in the car is perfect, it left the official service yesterday, but I lost the documents". That's why a small group of SoftUni students have created a web app - RealDeal.

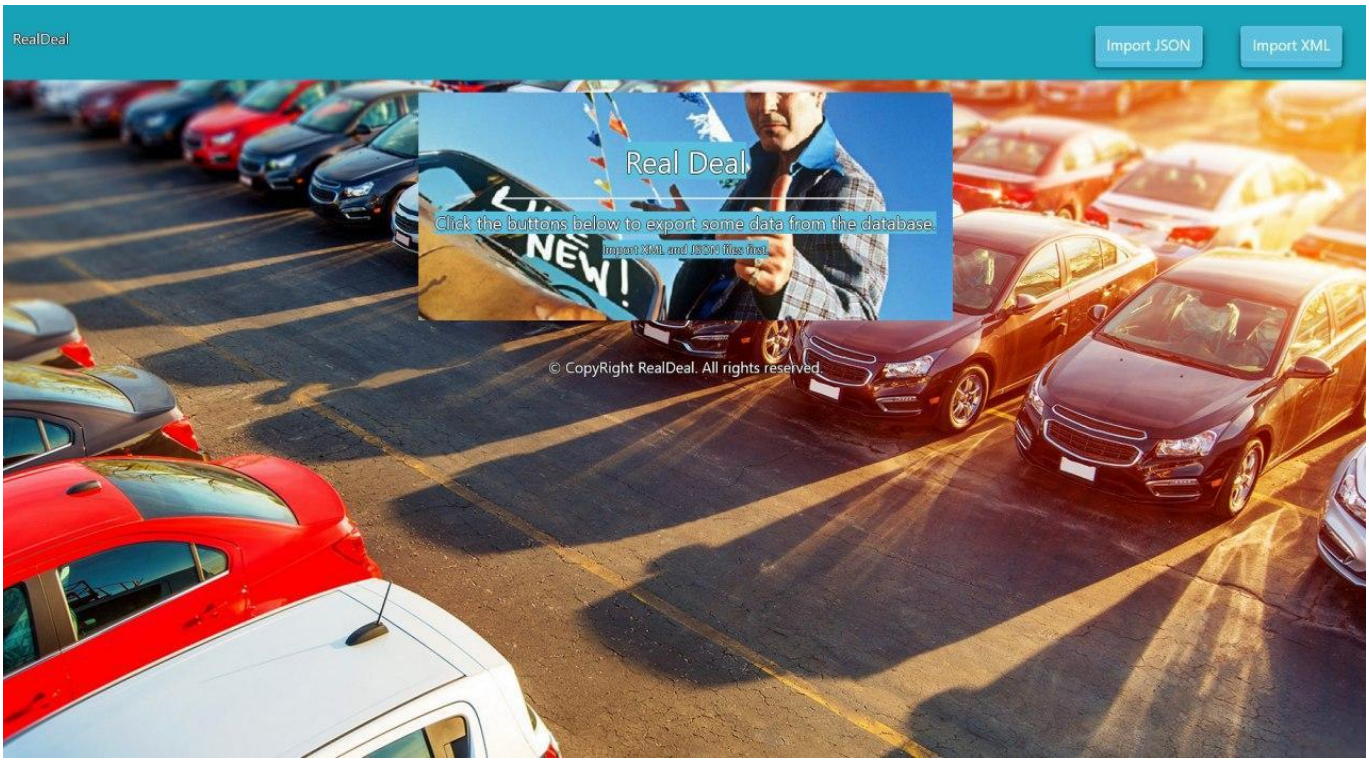
Thanks to artificial intelligence, the app filters out all the fraudulent and incorrect ads and leaves only those who really deserve attention. They need a little help with their great project and ask you to help them to handle data received from different sources and formats (json & xml).

1. Functionality Overview

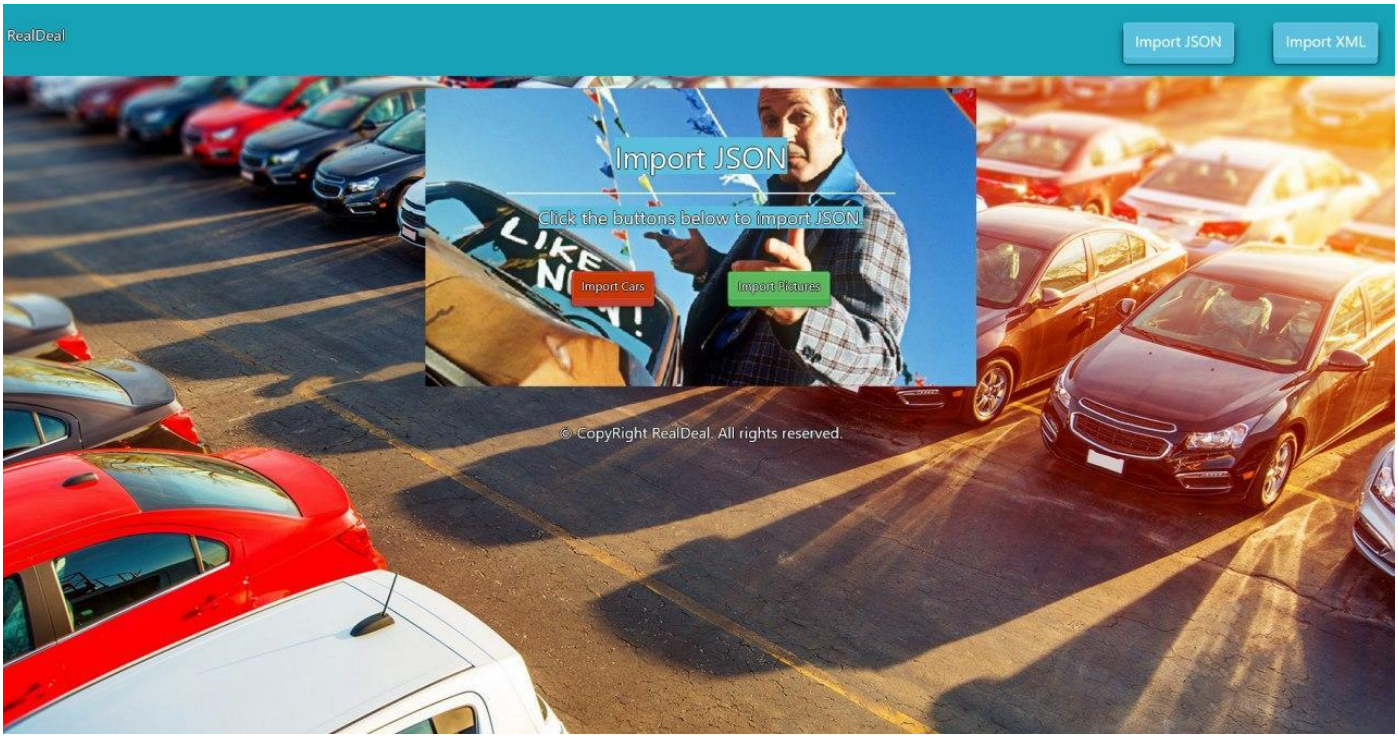
The application should be able to easily **import** hard-formatted data and **support functionality** for also **exporting** the imported data. The application is called – **Real Deal**.

Look at the pictures below to see what must happen:

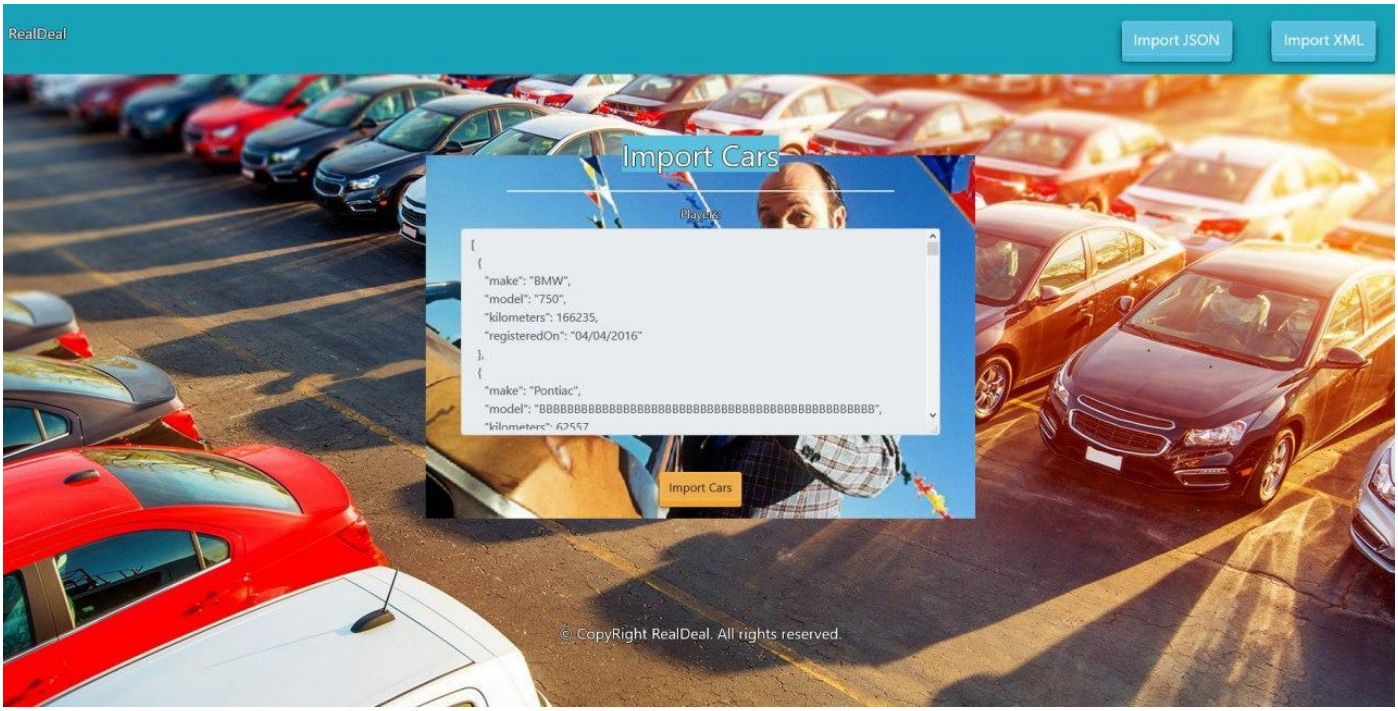
- Home page before importing anything:



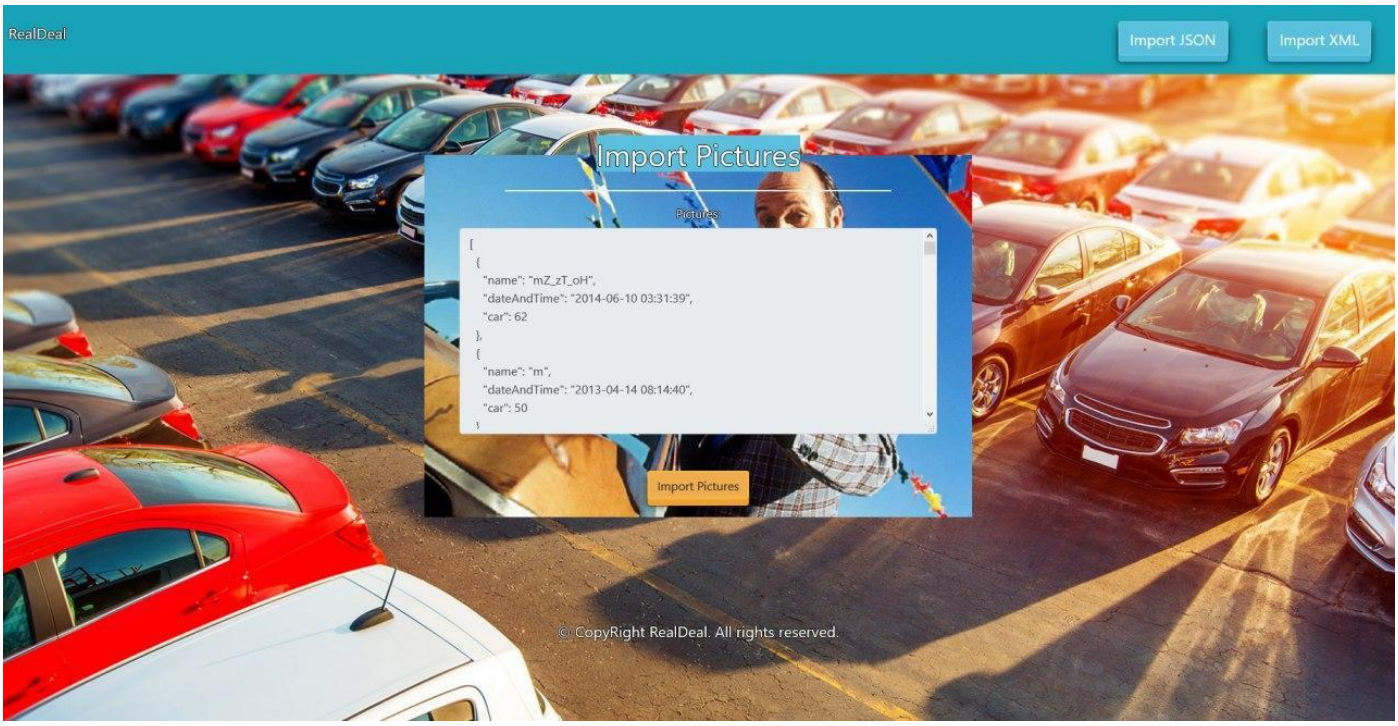
- Import JSON page before importing anything:



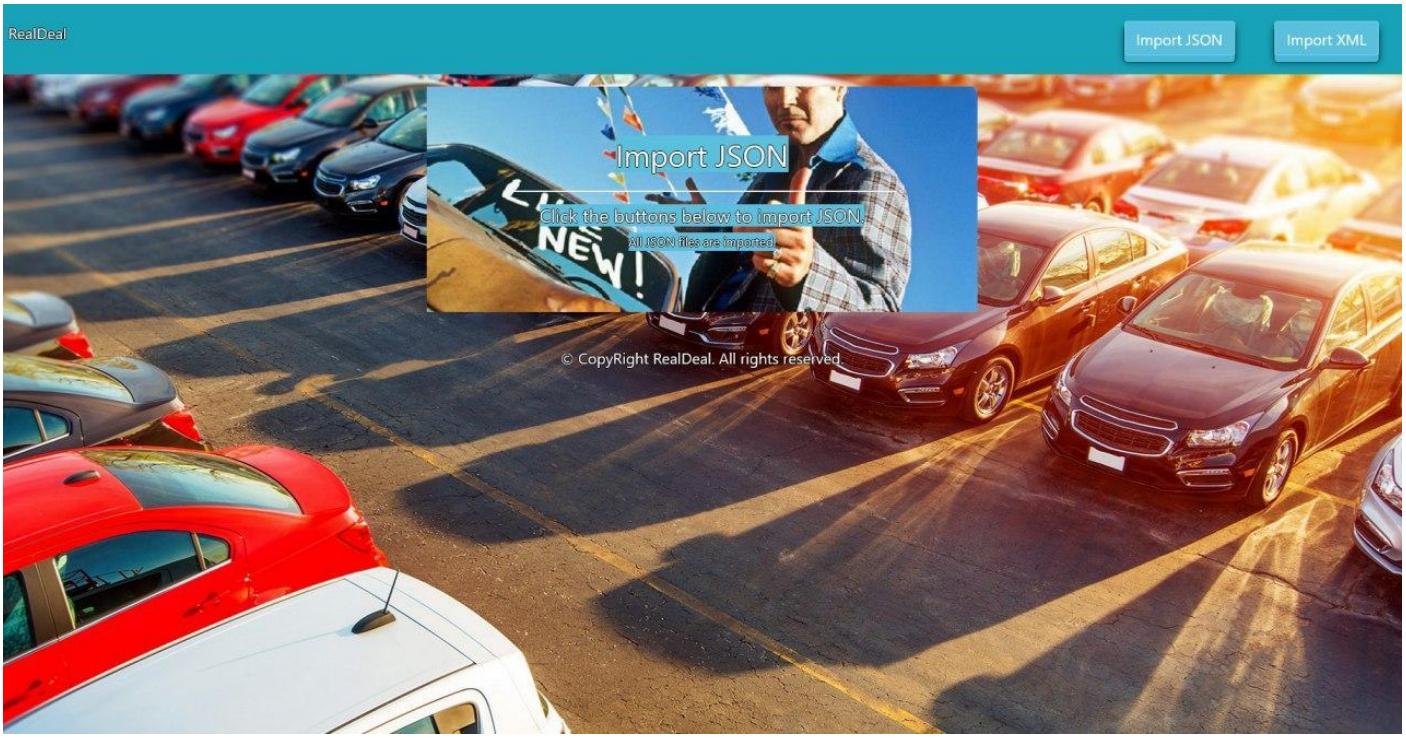
- Import Cars first:



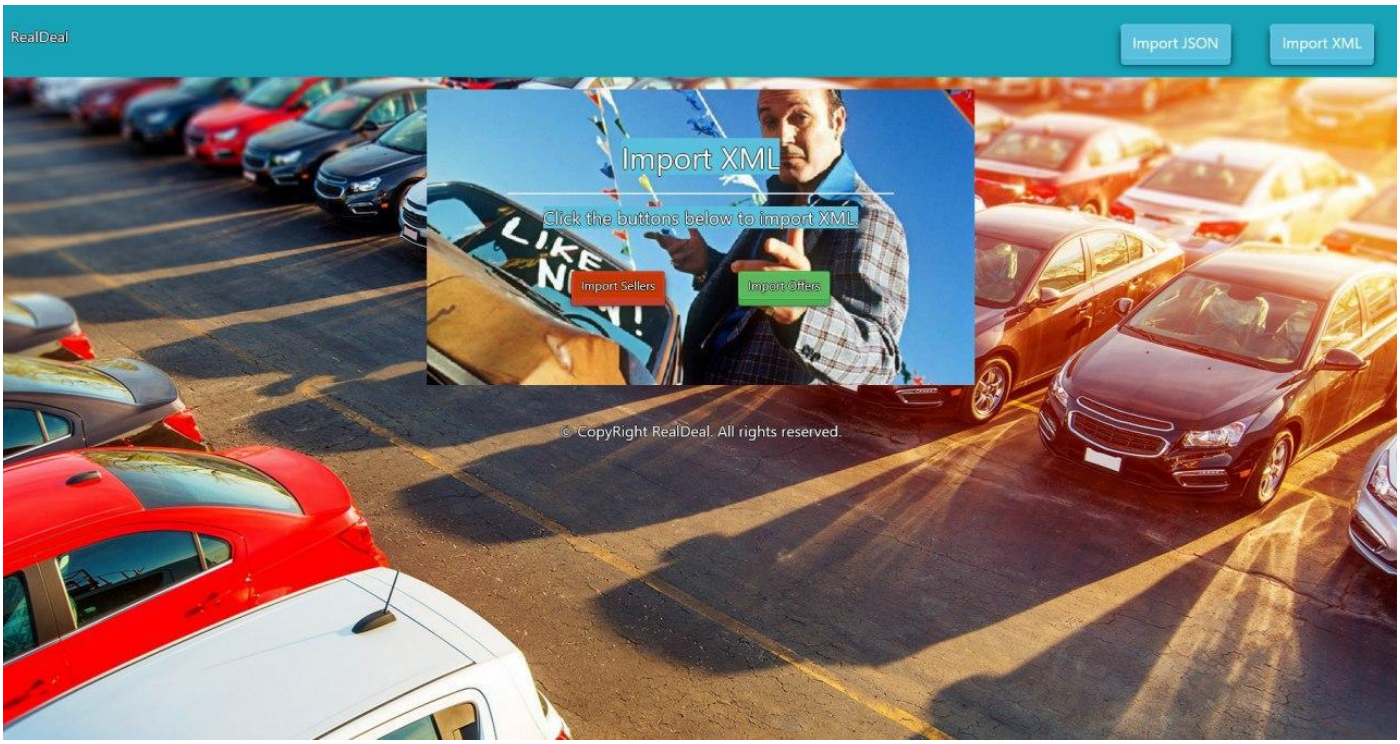
- Import Pictures second:



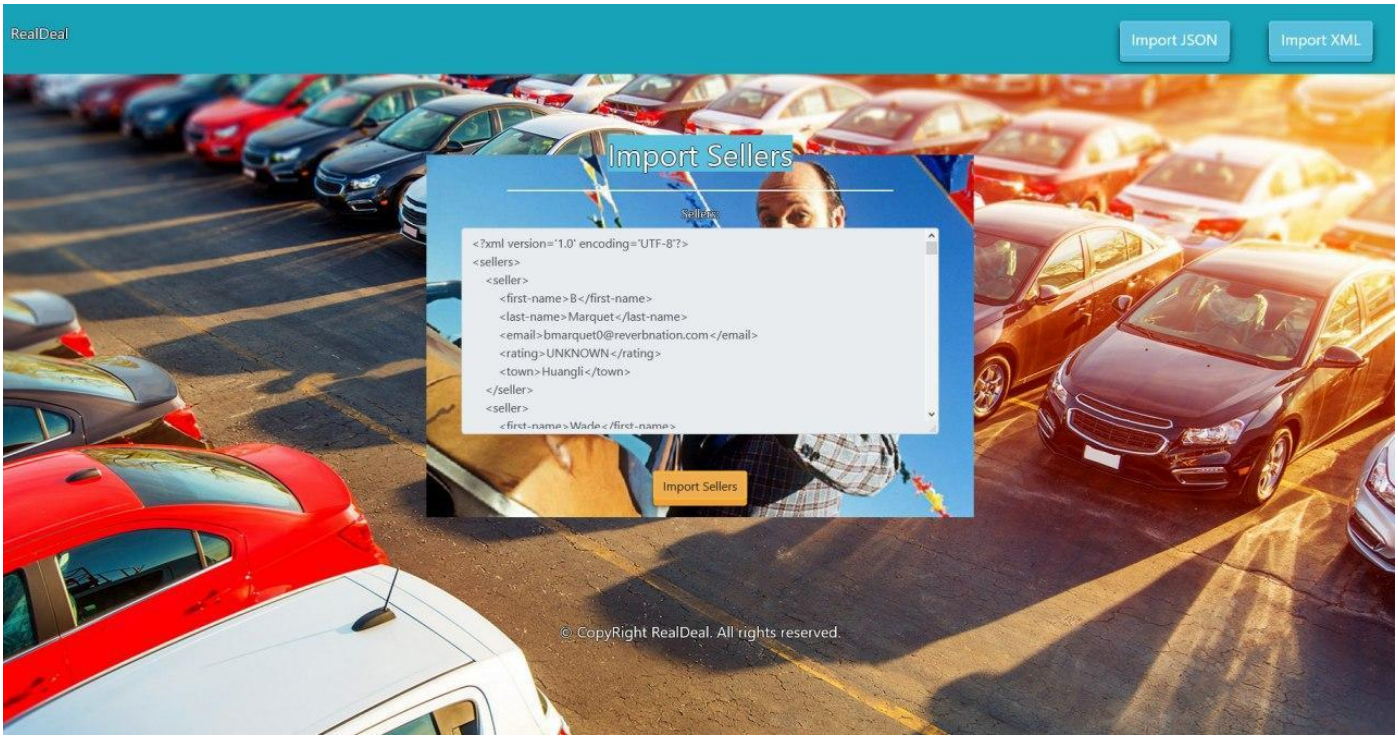
- Import JSON page after importing both files:



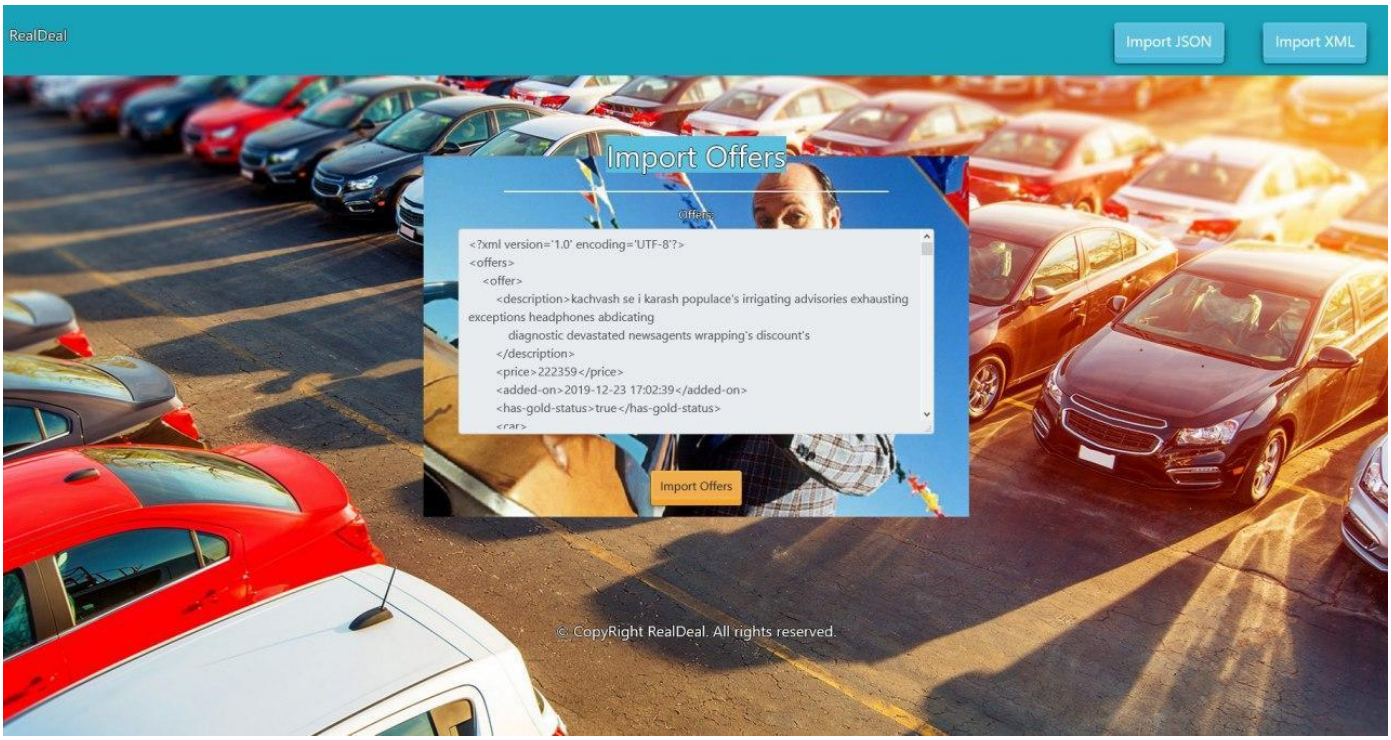
- Import XML page before importing the given data:



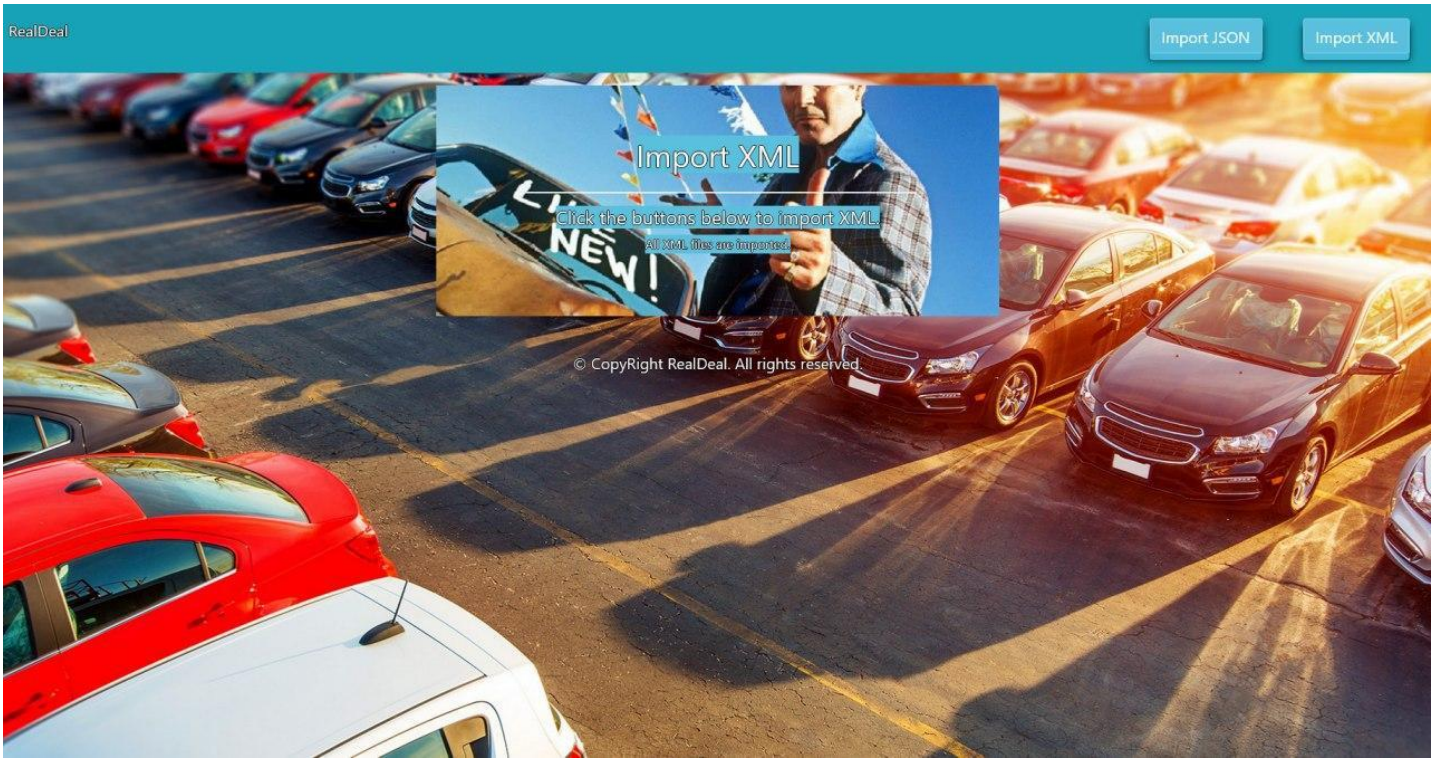
- Import Sellers data:



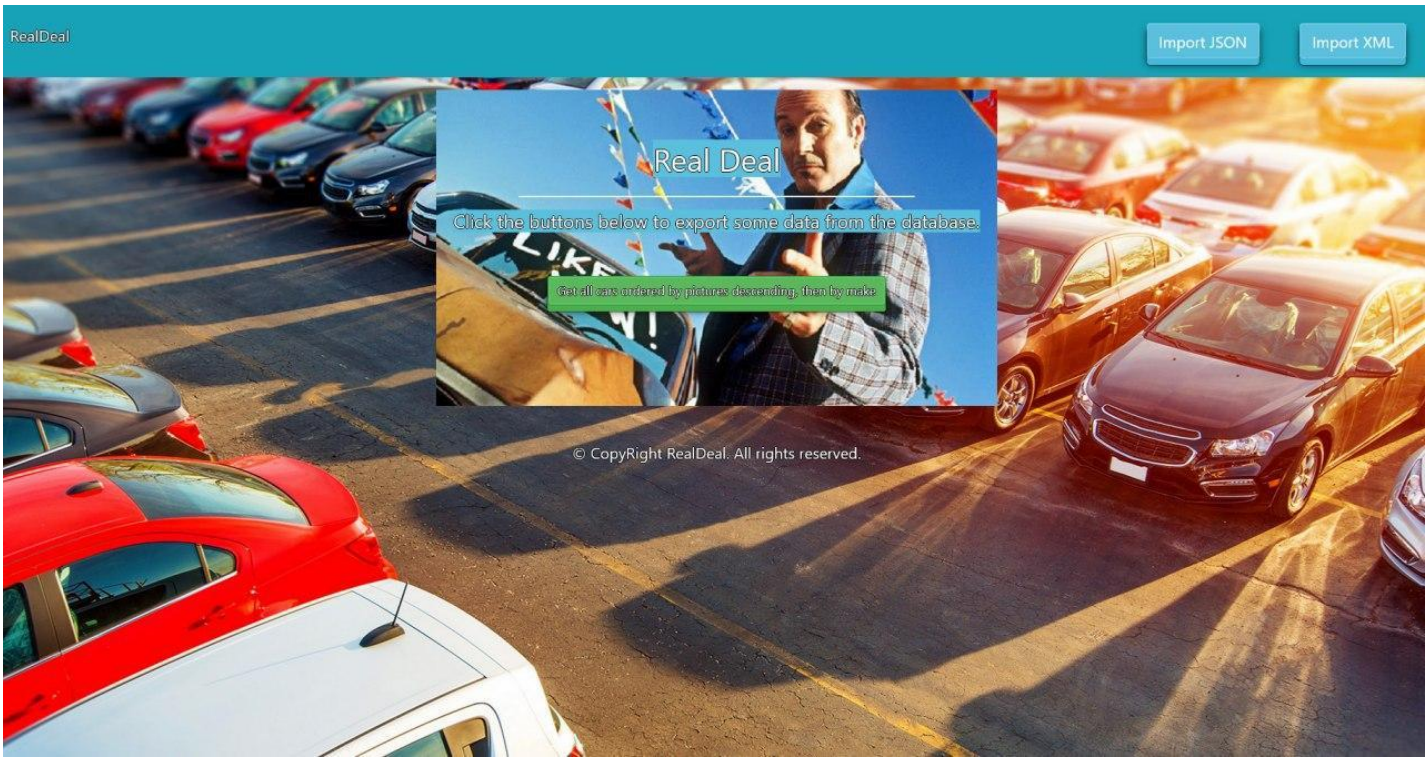
- Import Offers data:



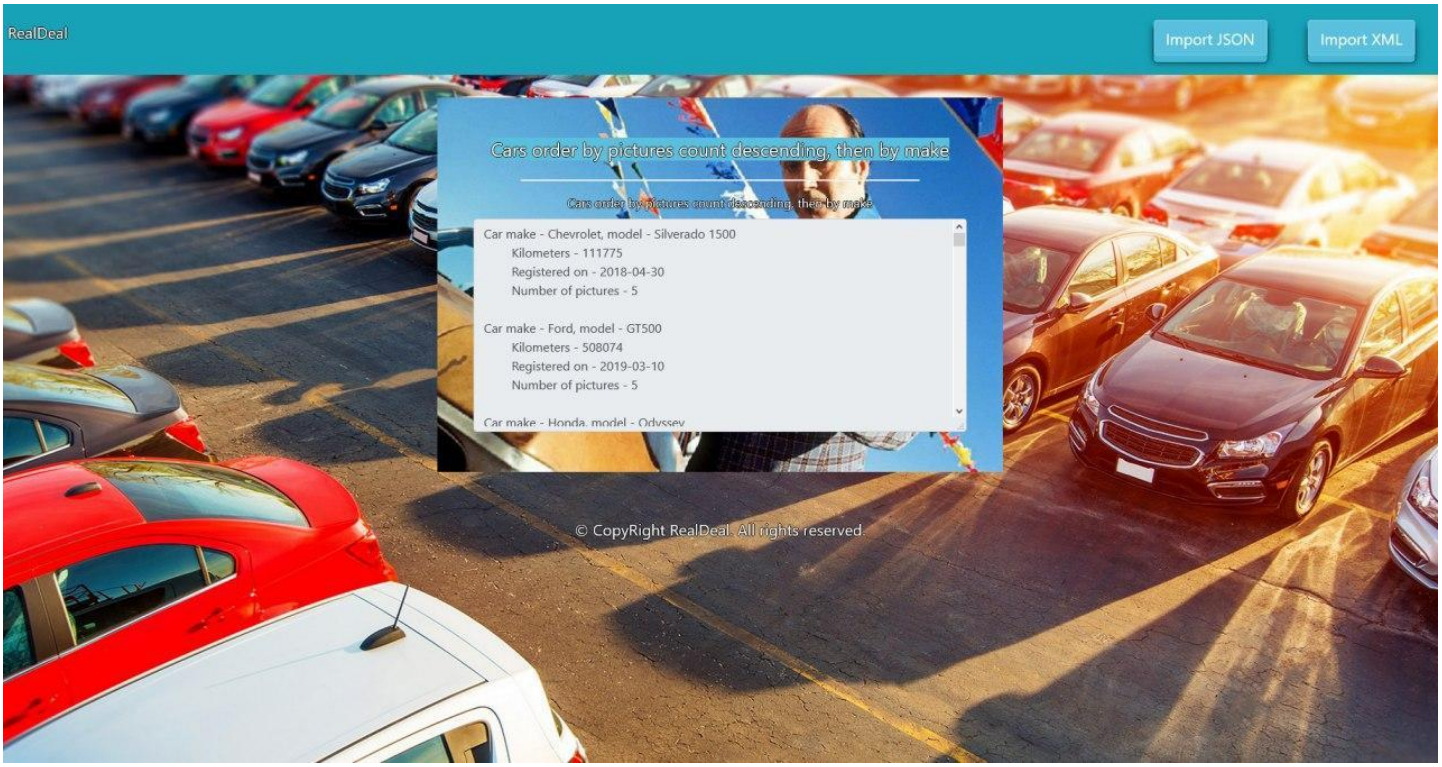
- Import JSON page after importing the data:



- Home page after the data is imported:



- Export cars by pictures count descending, then by make:



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 **Real Deal database** application should contain in its functionality.

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

Car

- **id** – **integer** number, **primary identification field**.
- **make** – a **char sequence** (between 2 to 20 exclusive).
- **model** – a **char sequence** (between 2 to 20 exclusive).
- **kilometers** – a **number** (must be positive).
- **registeredOn** – a **date.**
  The combination of **make**, **model** and **kilometers** makes a car **unique**.

Seller

- **id** – **integer** number, **primary identification field**.
- **firstName** – a **char sequence** (between 2 to 20 exclusive).
- **lastName** – a **char sequence** (between 2 to 20 exclusive).
- **email** – an **email –** (must contains '@' and '.' – dot). The email of a seller is **unique**.
- **rating** – **enumerated** value **must be** one of these **GOOD**, **BAD** or **UNKNOWN**. Cannot be null.
- **town** – a **char sequence –** the name of a town. Cannot be null.

Picture

- **id** – **integer** number, **primary identification field**.
- **name** – a **char sequence** (between 2 to 20 exclusive). The name of a picture is **unique**.
- **dateAndTime** – The date and time of a picture.

Offer

- **id** – **integer** number, **primary identification field**.
- **price** – a **number** (must be positive number).
- **description** – a very long text with **minimum length** 5.
- **hasGoldStatus** – can be **true** or **false**.
- **addedOn** – date and time of adding the offer.
  The combination of **description** and **addedOn** makes an offer **unique**.

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.

Relationships

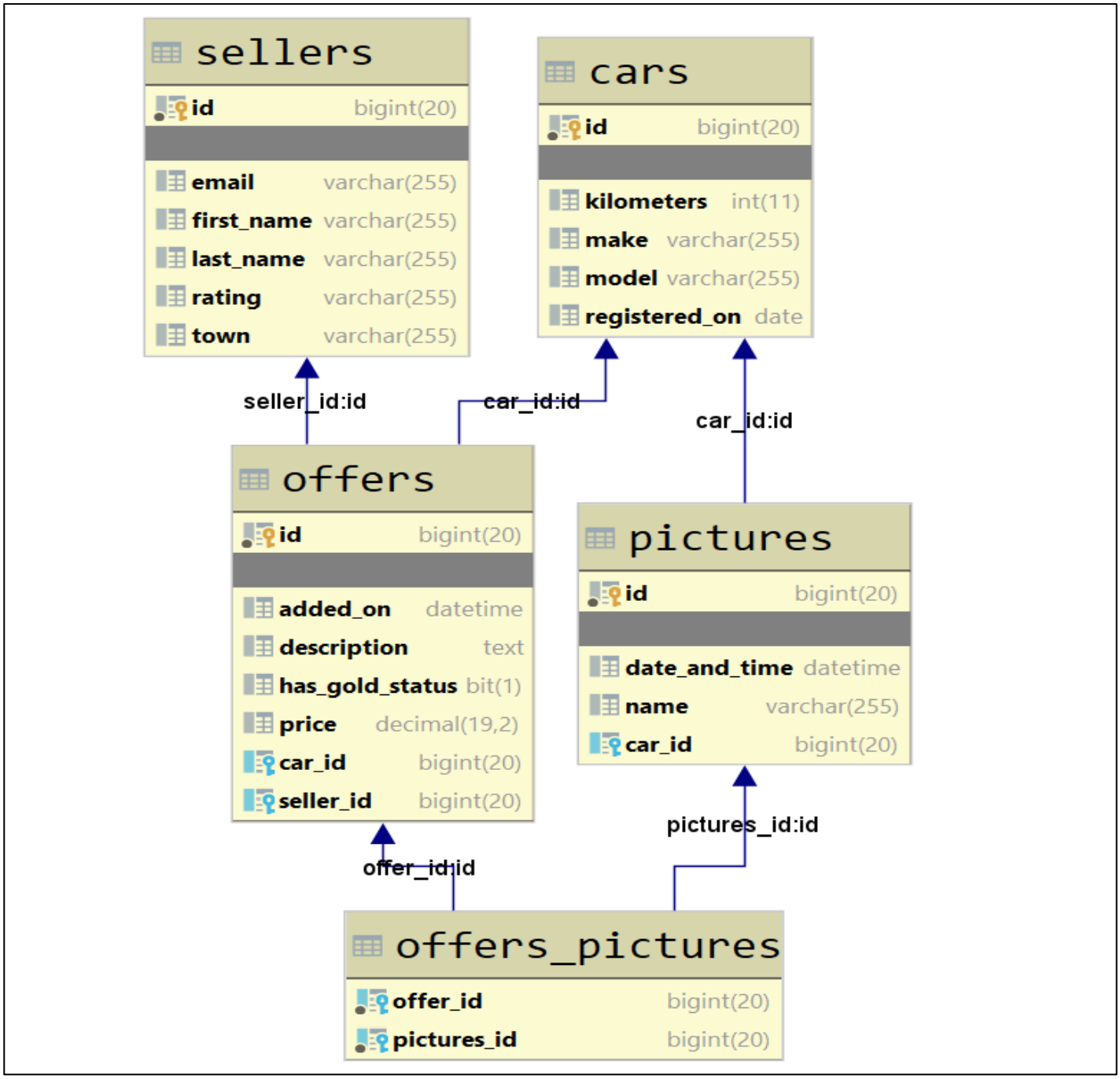
Your partners gave you a little hint about the more complex relationships in the database, so that you can implement it correctly.

One **Picture** may have only one **Car**, and one **Car** may have many **Pictures**.

One **Offer** may have only one **Car**, and one **Car** may have many **Offer**.

One **Offer** may have only one **Seller**, and one **Seller** may have many **Offers**.

One **Offer** may have many **Pictures**, and one **Picture** may have many **Offers.**



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 {car / seller / picture / offer}**" should be printed.

When the import is finished

"**Successfully imported {car/seller/picture/offer} - {make - model / lastName - email / name / addedOn - status}**"

JSON Import

Cars (cars.json)

cars.json
[{ "make": "BMW", "model": "750", "kilometers": 166235, "registeredOn": "04/04/2016" }, { "make": "Pontiac", "model": "BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB", "kilometers": 62557, "registeredOn": "21/10/2018" }, { "make": "G", "model": "Yukon", "kilometers": 211779, "registeredOn": "05/06/2019" }, { "make": "Mercury", "model": "Cougar", "kilometers": 647748, "registeredOn": "30/08/2017" }, { "make": "GMC", "model": "Sierra 2500", "kilometers": -1000, "registeredOn": "06/06/2018" }, ...
Successfully imported car - BMW - 750 Invalid car Invalid car Successfully imported car - Mercury - Cougar Invalid car ...

Pictures (pictures.json)

pictures.json


```
[
 {
  "name": "mZ_zT_oH",
  "dateAndTime": "2014-06-10 03:31:39",
  "car": 62
 },
 {
  "name": "m",
  "dateAndTime": "2013-04-14 08:14:40",
  "car": 50
 },
 {
  "name": "jI_xK_z1",
  "dateAndTime": "2011-09-21 22:57:24",
  "car": 32
 },
 {
  "name": "vA_rX_kN_tooooooooooo_looooooooooooonnnnnnnngggg_name",
  "dateAndTime": "2013-08-02 10:35:35",
  "car": 20
 },
 {
  "name": "oM_wX_5W",
  "dateAndTime": "2016-07-19 01:12:57",
  "car": 11
 },
  . . .
```


```
Successfully import picture - mZ_zT_oH
Invalid picture
Successfully import picture - jI_xK_z1
Invalid picture
Successfully import picture - oM_wX_5W
…
```

XML Import

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

Sellers (sellers.xml)

sellers.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<sellers>
    <seller>
        <first-name>B</first-name>
        <last-name>Marquet</last-name>
        <email>bmarquet0@reverbnation.com</email>
        <rating>UNKNOWN</rating>
        <town>Huangli</town>
    </seller>
    <seller>
        <first-name>Wade</first-name>
        <last-name>F</last-name>
        <email>wforseith1@umich.edu</email>
        <rating>GOOD</rating>
        <town>Juhut</town>
    </seller>
    <seller>
        <first-name>Kasper</first-name>
        <last-name>Juanes</last-name>
        <email>kjuanes2@google.com.br</email>
        <rating>BAD</rating>
        <town>Prakhon Chai</town>
    </seller>
    <seller>
        <first-name>Shaina</first-name>
        <last-name>Waterworth</last-name>
        <email>swaterworth3@nasa.gov</email>
        <rating>VERY_GOOD</rating>
        <town>Hailin</town>
    </seller>
    <seller>
        <first-name>Gordon</first-name>
        <last-name>Collumbell</last-name>
        <email>gcollumbell4economist.com</email>
        <rating>GOOD</rating>
        <town>Rancaerang</town>
    </seller>
  ...
<pictures/>
```

```
Invalid seller
Invalid seller
Successfully import seller Juanes - kjuanes2@google.com.br
Invalid seller
Invalid seller
```

Offers (offers.xml)

Note: When importing the offers, set all the pictures, which corresponds to this car with the given id.

offers.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<offers>
    <offer>
        <description>kachvash se i karash populace's irrigating advisories
exhausting exceptions headphones abdicating
            diagnostic devastated newsagents wrapping's discount's
        </description>
        <price>222359</price>
        <added-on>2019-12-23 17:02:39</added-on>
        <has-gold-status>true</has-gold-status>
        <car>
            <id>70</id>
        </car>
        <seller>
            <id>84</id>
        </seller>
    </offer>
    <offer>
        <description>info
        </description>
        <price>199222</price>
        <added-on>2019-03-25 17:36:18</added-on>
        <has-gold-status>false</has-gold-status>
        <car>
            <id>24</id>
        </car>
        <seller>
            <id>67</id>
        </seller>
    </offer>
    <offer>
        <description>kachvash se i karash unionizing relaxation carnations
ceremonial overthrown recoveries interlocks
            culminates aptitude's pretenders networking ordinaries
        </description>
        <price>186841</price>
        <added-on>2019-08-23 10:10:19</added-on>
        <has-gold-status>false</has-gold-status>
        <car>
            <id>30</id>
        </car>
        <seller>
            <id>11</id>
        </seller>
    </offer>
```

```
    <offer>
        <description>kachvash se i karash suitcase's unbeatable tantalizes
calamity's researches unwrapping copperhead
            tolerating stallion's transposed neglecting dishearten
        </description>
        <price>-226038</price>
        <added-on>2020-02-16 11:51:54</added-on>
        <has-gold-status>false</has-gold-status>
        <car>
            <id>63</id>
        </car>
        <seller>
            <id>76</id>
        </seller>
    </offer>
...
```

```
Successfully import offer 2019-12-23T17:02:39 - true
Invalid offer
Successfully import offer 2019-08-23T10:10:19 - false
Invalid offer
Successfully import offer 2019-09-24T16:50:15 - false
```

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 cars order by pictures count in descending order, then by make

- Extract from the database, the make, model, kilometers, date of registration and count of pictures of all cars. Order them first by pictures count in descending order then by make alphabetically.
- Return the information in this format:

  "Car make - {make}, model - {model}

  Kilometers - {kilometers}

  Registered on - {registered on}

  Number of pictures - {number of pictures}

  . . ."