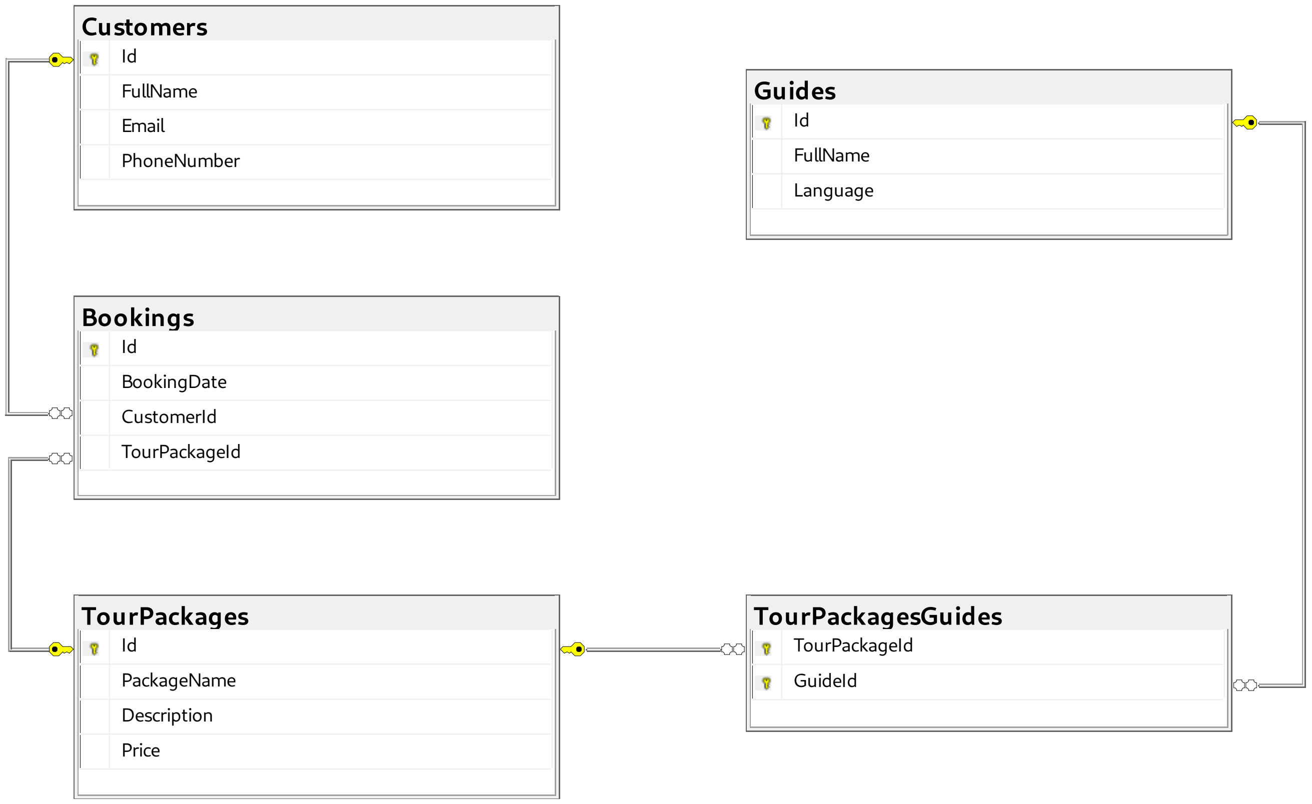
Entity Framework Core
Exam Prep I

Submit your solutions in the **SoftUni Judge** system (delete all **bin/obj** and **packages** folders) here.

Before submitting your solutions in the **SoftUni Judge** system, delete all **bin/obj** and **packages** folders. If the **zip** file is still too large, you can delete the **ImportResults**, **ExportsResults** and **Datasets** folders too.

Your task is to create a **database application**, using **Entity Framework Core,** using the **Code First** approach. Design the **domain models** and **methods** for manipulating the data, as described below.

Travel Agency



1. Project Skeleton Overview

You are given a **project skeleton**, which includes the following folders:

- **Data** – contains the **TravelAgencyContext** class, **Models** folder, which contains the **entity classes** and the **Configuration** class with the **connection string**
- **DataProcessor** – contains the **Serializer** and **Deserializer** classes, which are used for **importing** and **exporting** data
- **Datasets** – contains the **.json** and **.xml** files for the import part

- **ImportResults** – contains the **import** results you make in the **Deserializer** class
- **ExportResults** – contains the **export** results you make in the **Serializer** class

2. Model Definition (60 pts)

The application needs to store the following data:

Customer

- **Id** – integer, **Primary Key**
- **FullName** – **text** with length **[4, 60] (required)**
- **Email** – **text** with length **[6, 50] (required)**
- **PhoneNumber** – text with **length 13. (required)**
  - All phone numbers **must have the following structure**: a **plus sign** followed by **12 digits**, **without spaces or special characters**:
    - Example -> **+359888555444**
    - HINT -> use **DataAnnotation [RegularExpression]**
- **Bookings -** a collection of type **Booking**

Booking

- **Id** – integer, **Primary Key**
- **BookingDate – DateTime (required)**
- **CustomerId** – **integer**, **foreign key (required)**
- **Customer** – **Customer**
- **TourPackageId – integer**, **foreign key (required)**
- **TourPackage – TourPackage**

Guide

- **Id** – integer, **Primary Key**
- **FullName** – **text** with length **[4, 60]** (**required**)
- **Language** – **Language enum (English = 0, German, French, Spanish, Russian) (required)**
- **TourPackagesGuides -** collection of type **TourPackageGuide**

TourPackage

- **Id**
- **PackageName** – **text** with length **[2, 40] (required)**
- **Description** – **text** with **max length 200 (not required)**
- **Price** – a **positive decimal value** (**required**)
- **Bookings -** a collection of type **Booking**
- **TourPackagesGuides -** collection of type **TourPackageGuide**

TourPackageGuide

- **TourPackageId – integer, Primary Key, foreign key (required)**
- **TourPackage – TourPackage**
- **GuideId – integer, Primary Key, foreign key (required)**
- **Guide – Guide**

3. Data Import (20pts)

For the functionality of the application, you need to create several methods that manipulate the database. The **project skeleton** already provides you with these methods, inside the **Deserializer class**. Usage of **Data Transfer Objects** or **AutoMapper** is **optional**.

To ensure the application's functionality, it is essential to **populate the database with initial data**. Inside the **DbContext class**, you will find a **commented-out section** specifically designed for seeding data.
Before applying migrations and updating the database, please **uncomment this section**.

Use the provided **JSON** and **XML** files to populate the database with data. **Import all the valid information** from the files into the database.

You are **not allowed** to modify the provided **JSON** and **XML** files.

If a record does not meet the requirements from the first section, print an error message:

Error message
Invalid data format!

If some data appears to be duplicated, do not import the entity, print a duplication data message:

Error message
Error! Data duplicated.

Error message and Duplication message will be provided as constants in the skeleton.

XML Import

Import Customers

Using the file "**customers.xml"**, **import the data from the file** into the database.

Each imported **customer should be validated** and **added to the database if it meets the specified criteria**. The method should **return a string containing information about each import attempt**, formatted as described.

Constraints

- **Validation of Customer Entities** - Each customer entity must be validated against the following criteria:
    - **FullName** – Must meet the constraints for the property, described above
    - **Email** – Must meet the constraints for the property, described above
    - **PhoneNumber** - Must meet the constraints for the property, described above
- **Duplication Check** - Before adding a customer to the database**, ensure there are no existing records with the same**:
    - **FullName** OR **Email** OR **PhoneNumber**
- If **any validation error occurs** for a customer entity **or any of the fields match an existing record**, the **customer entity should not be imported**, and the appropriate **error message** or **duplication message should be appended** to the method's output

- **Success Messages**
    - For **each successfully imported customer**, append a **success message** to the output, formatted as **Successfully imported customer - {FullName}**
- **Data Persistence**
    - After processing all customers from the XML file, **add the valid customer entities** to the proper collection
    - **Save the changes** to the database

Success message
Successfully imported customer - {**customerFullName**}

Example

customers.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<Customers>
    <Customer phoneNumber="+357683444233">
        <FullName>Robert Simons</FullName>
        <Email>robert.simons@mail.dm</Email>
    </Customer>
    <Customer phoneNumber="+357183414234">
        <FullName>Alice Johnson</FullName>
        <Email>alice.johnson@mail.du</Email>
    </Customer>
    <Customer phoneNumber="+357683444035">
        <FullName>John Doe</FullName>
        <Email>john.doe@mail.dm</Email>
    </Customer>
    <Customer phoneNumber="+357600444236">
        <FullName>Emma Brown</FullName>
        <Email>emma.brown@mail.dm</Email>
    </Customer>
...
<Customers>
```

Output

```
Successfully imported customer - Donald Sanders
Invalid data format!
Successfully imported customer - Alice Johnson
Successfully imported customer - John Doe
Invalid data format!
Error! Data duplicated.
...
```

Upon **correct import logic**, you should have imported **21 customers**

JSON Import

Import Bookings

Using the file **"bookings.json"**, import the data from that file into the database. Print information about each imported object in the format described below.

Constraints

- If **any validation error occurs** for the **booking** entity (**invalid date**), **do not** import any part of the entity and **append an error message** to the **method output**.
    - The **DateTime data** in the document will be in the following format: "yyyy-MM-dd"

    - Make sure you use **CultureInfo.InvariantCulture**
- The **Customers** and **TourPackages** associated with every single Booking will be string values, which **could be matched to already existing records in the database**.

Success message
Successfully imported booking – TourPackage: {**tourPackageName**}, Date: {**date.ToString(**"yyyy-MM-dd"**)**}

Example

bookings.json

```
[
  {
   "BookingDate": "2024-09-21",
   "CustomerName": "Donald Sanders",
   "TourPackageName": "Horse Riding Tour"
  },
  {
   "BookingDate": "2024-09-22",
   "CustomerName": "Donald Sanders",
   "TourPackageName": "Sightseeing Tour"
  },
  {
   "BookingDate": "2024-10-01",
   "CustomerName": "William Garcia",
   "TourPackageName": "Historical Sites"
  },
  {
   "BookingDate": "2024-11-01",
   "CustomerName": "William Garcia",
   "TourPackageName": "Horse Riding Tour"
  },
...
]
```

Output

```
Successfully imported booking. TourPackage: Horse Riding Tour, Date: 2024-09-21
Successfully imported booking. TourPackage: Sightseeing Tour, Date: 2024-09-22
Successfully imported booking. TourPackage: Historical Sites, Date: 2024-10-01
Successfully imported booking. TourPackage: Horse Riding Tour, Date: 2024-11-01
Successfully imported booking. TourPackage: Sightseeing Tour, Date: 2024-09-20
Successfully imported booking. TourPackage: Historical Sites, Date: 2024-12-06
Successfully imported booking. TourPackage: Horse Riding Tour, Date: 2024-09-15
Successfully imported booking. TourPackage: Historical Sites, Date: 2024-09-23
Successfully imported booking. TourPackage: Sunset Cruise, Date: 2024-09-27
Successfully imported booking. TourPackage: Horse Riding Tour, Date: 2024-09-28
Successfully imported booking. TourPackage: Wildlife Safari, Date: 2024-09-29
Successfully imported booking. TourPackage: Sunset Cruise, Date: 2024-09-30
Successfully imported booking. TourPackage: Sightseeing Tour, Date: 2024-10-05
Invalid data format!
...
```

Upon **correct import logic**, you should have imported **25 bookings**

4. Data Export (20 pts)

Use the provided methods in the Serializer class**.** Usage of **Data Transfer Objects and AutoMapper** is **optional**.

XML Export

Export All Guides Speaking Spanish Language With All Their Packages

Export **all guides** who speak the **Spanish language** along with **all their associated tour packages**. The exported data should be in **XML format**. Order the **guides by the number of tour packages in descending order**. If two guides have the same number of packages, **order them alphabetically by their full name.**

For each guide**, include all their tour packages**. Order the **tour packages by price in descending order**. If two tour packages have the same price, **order them alphabetically by their name**.

Data Fields:

- Guide: Export the full name of the guide and their tour packages
- Tour Package: Export the tour package name, description, and price

Expected XML Output:

- The root element should be <Guides>
- Each guide should be represented by a <Guide> element
- All TourPackages should be presented as an array of TourPackage
- Each tour package should be represented by a <TourPackage> element within its associated guide

Example

ExportGuidesWithSpanishLanguageWithAllTheirTourPackages(context)

```
<?xml version="1.0" encoding="utf-16"?>
<Guides>
    <Guide>
        <FullName>Alex Johnson</FullName>
        <TourPackages>
            <TourPackage>
                <Name>Horse Riding Tour</Name>
                <Description>Experience the thrill of a guided horse riding tour through picturesque landscapes. Suitable for all skill levels. Enjoy nature and create unforgettable memories. Duration: 3 hours.</Description>
                <Price>199.99</Price>
            </TourPackage>
            <TourPackage>
                <Name>Historical Sites</Name>
                <Description>Explore ancient ruins, museums, and landmarks on a guided tour. Learn about the rich history and culture of the area. Ideal for history buffs. Duration: 4 hours.</Description>
                <Price>159.99</Price>
            </TourPackage>
            <TourPackage>
                <Name>City Tour</Name>
                <Description>Discover the charm of the city with a guided tour. Visit famous landmarks, bustling markets, and hidden gems. Perfect for all ages. Duration: 3 hours.</Description>
                <Price>129.99</Price>
            </TourPackage>
        </TourPackages>
    </Guide>
    <Guide>
        <FullName>Chris Martin</FullName>
        <TourPackages>
```

```
...
</TourPackages>
...
</Guide>
...
<Guides>
```

JSON Export

All Customers That Have Booked Horse Riding Tour Package

Export all customers who have booked the "**Horse Riding Tour**" package. The exported data should be in JSON format and adhere to the following specifications:

- **Selection Criteria**:
  - Select **all customers** who have **at least one booking for the "Horse Riding Tour"** package
  - For each customer, export their **full name** and **phone number**
  - For each booking, export the **tour package name** and the **booking date**
- **Data Fields**:
  - Customer – **FullName**, **PhoneNumber**
  - Booking – **TourPackageName**, **Date**(formatted as "yyyy-MM-dd")
- **Ordering:**
  - Order **customers by the number of bookings (descending)**
  - If two customers have the same number of bookings, **order them alphabetically by their full name**
  - Order the **bookings by date (ascending)**

Example

ExportCustomersThatHaveBookedHorseRidingTourPackage(context)

```
[
  {
   "FullName": "Donald Sanders",
   "PhoneNumber": "+357683444233",
   "Bookings": [
     {
      "TourPackageName": "Horse Riding Tour",
      "Date": "2024-09-21"
     }
   ]
  },
  {
   "FullName": "Henry White",
   "PhoneNumber": "+357611144251",
   "Bookings": [
     {
      "TourPackageName": "Horse Riding Tour",
      "Date": "2024-09-28"
     }
   ]
  },
  {
   "FullName": "Michael Smith",
   "PhoneNumber": "+357683411237",
   "Bookings": [
     {
      "TourPackageName": "Horse Riding Tour",
      "Date": "2024-09-15"
```

```
    }
  ]
 },
...
]
```