# Spring Data Exam Preparation – 18 March 2024

# Find a Star

[Link to Judge](https://judge.softuni.org/Contests/4234/Java-DB-Spring-Data-Regular-Exam-29-July-2023)

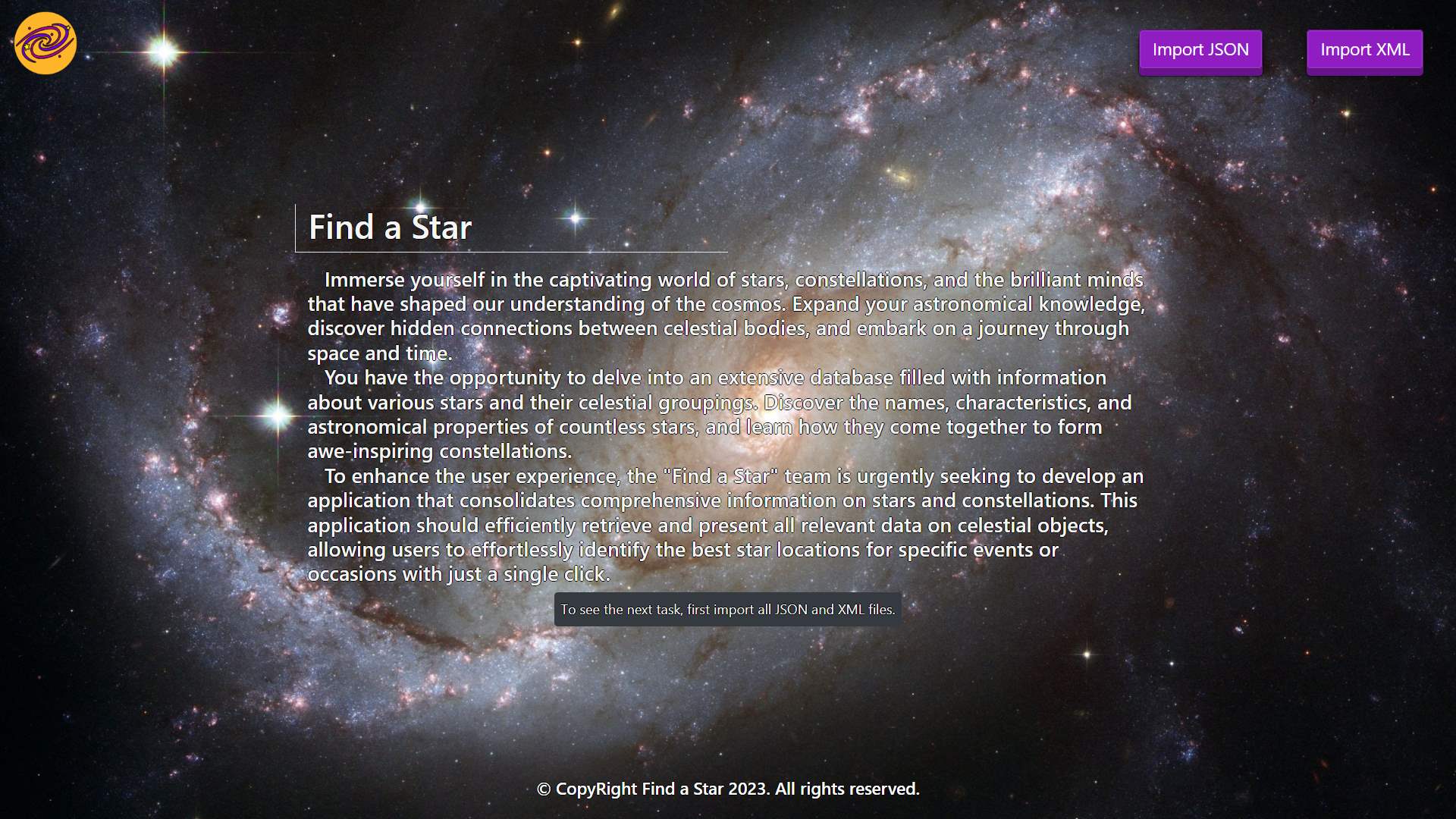
*Our Immerse yourself in the captivating world of stars, constellations, and the brilliant minds that have shaped our understanding of the cosmos. Expand your astronomical knowledge, discover hidden connections between celestial bodies, and embark on a journey through space and time. You have the opportunity to delve into an extensive database filled with information about various stars and their celestial groupings. Discover the names, characteristics, and astronomical properties of countless stars, and learn how they come together to form awe-inspiring constellations. To enhance the user experience, the "Find a Star" team is urgently seeking to develop an application that consolidates comprehensive information on stars and constellations. This application should efficiently retrieve and present all relevant data on celestial objects, allowing users to effortlessly identify the best star locations for specific events or occasions with just a single click.*

## Functionality Overview

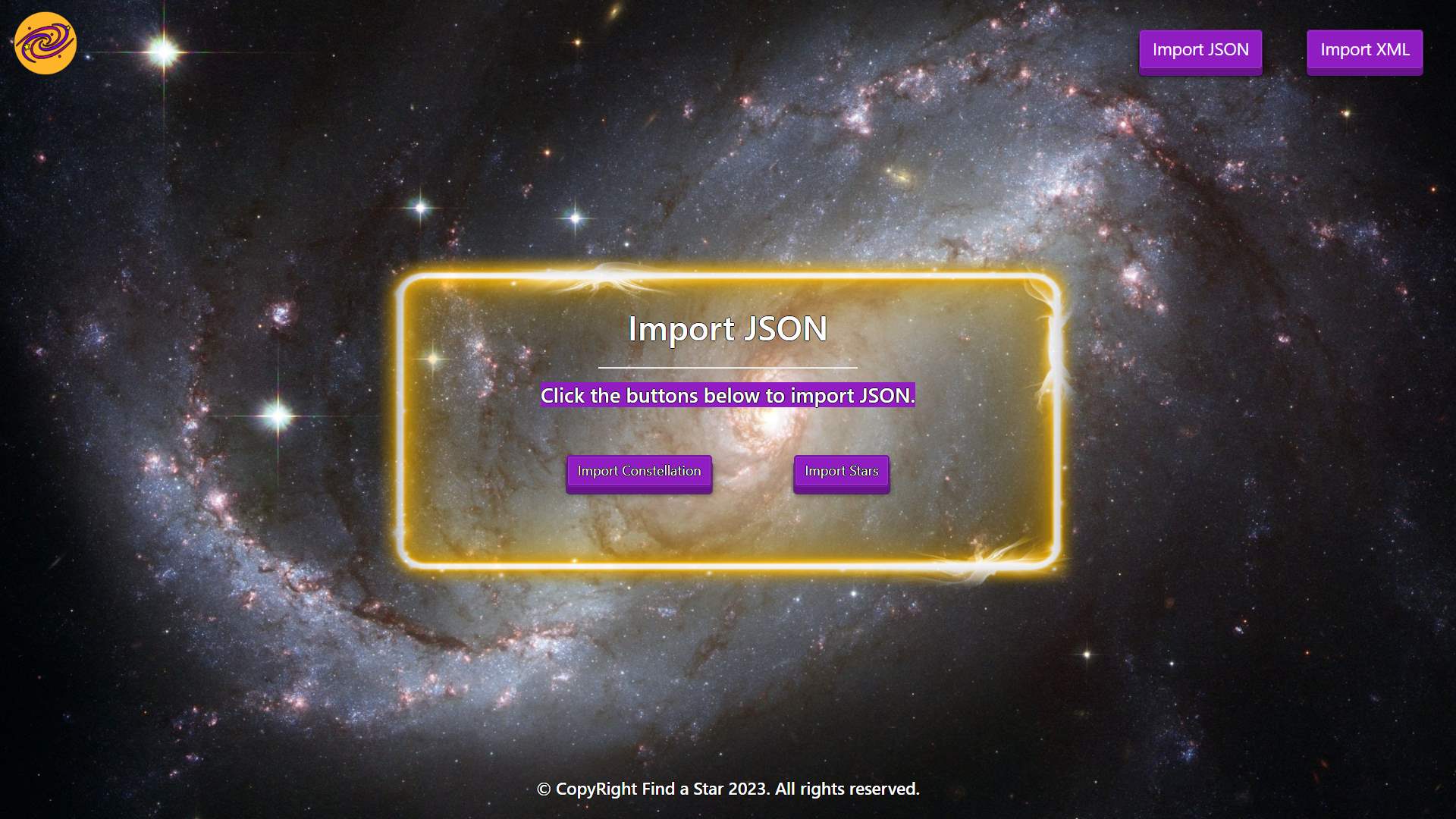
The application should be able to easily **import** hard-formatted data and **support functionalities** for also **exporting** the imported data. The application is called – **Find a Star**.

Look at the pictures below to see what must happen:

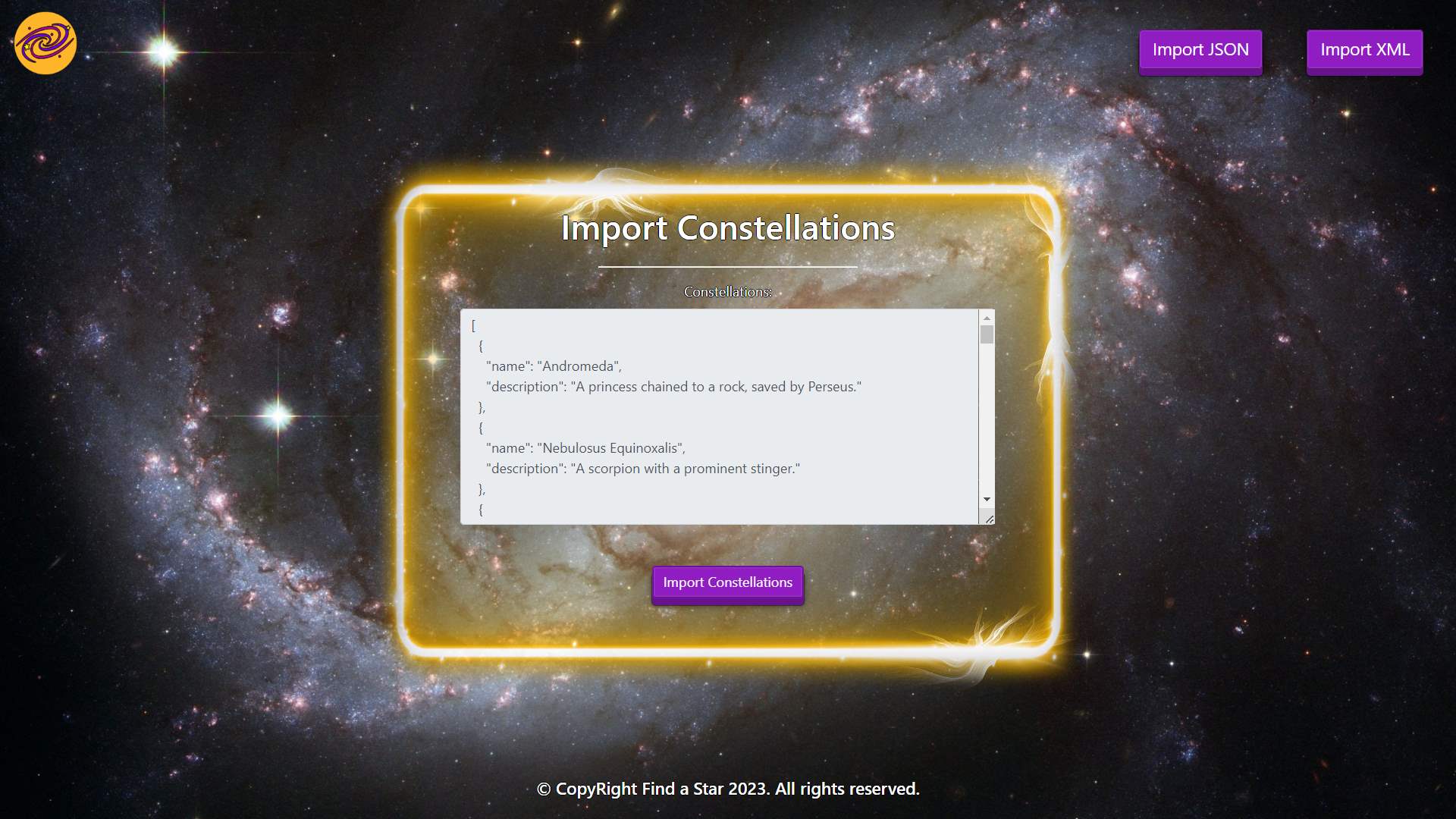
* The home page before importing anything:



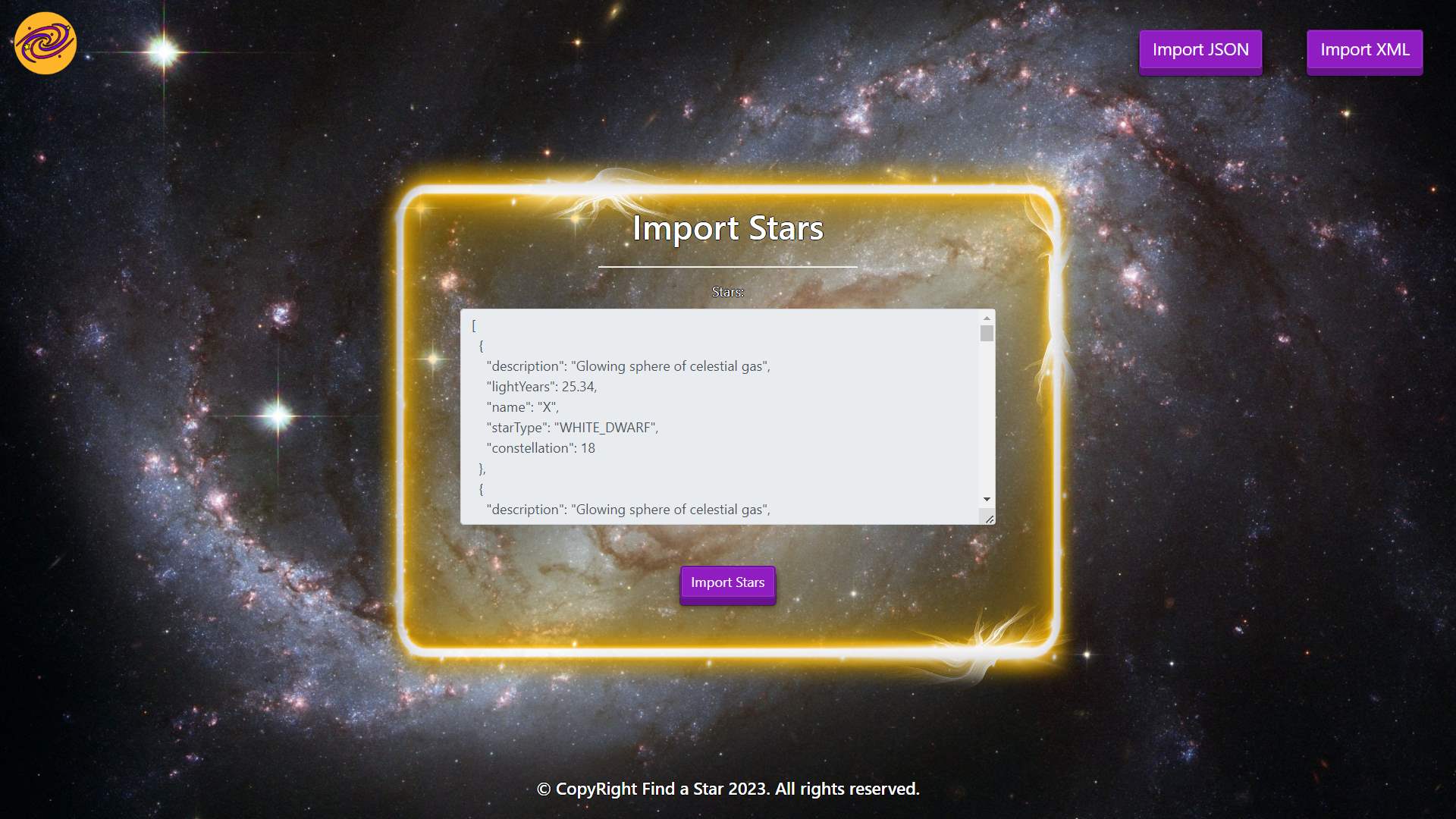
* The import JSON page before importing anything:



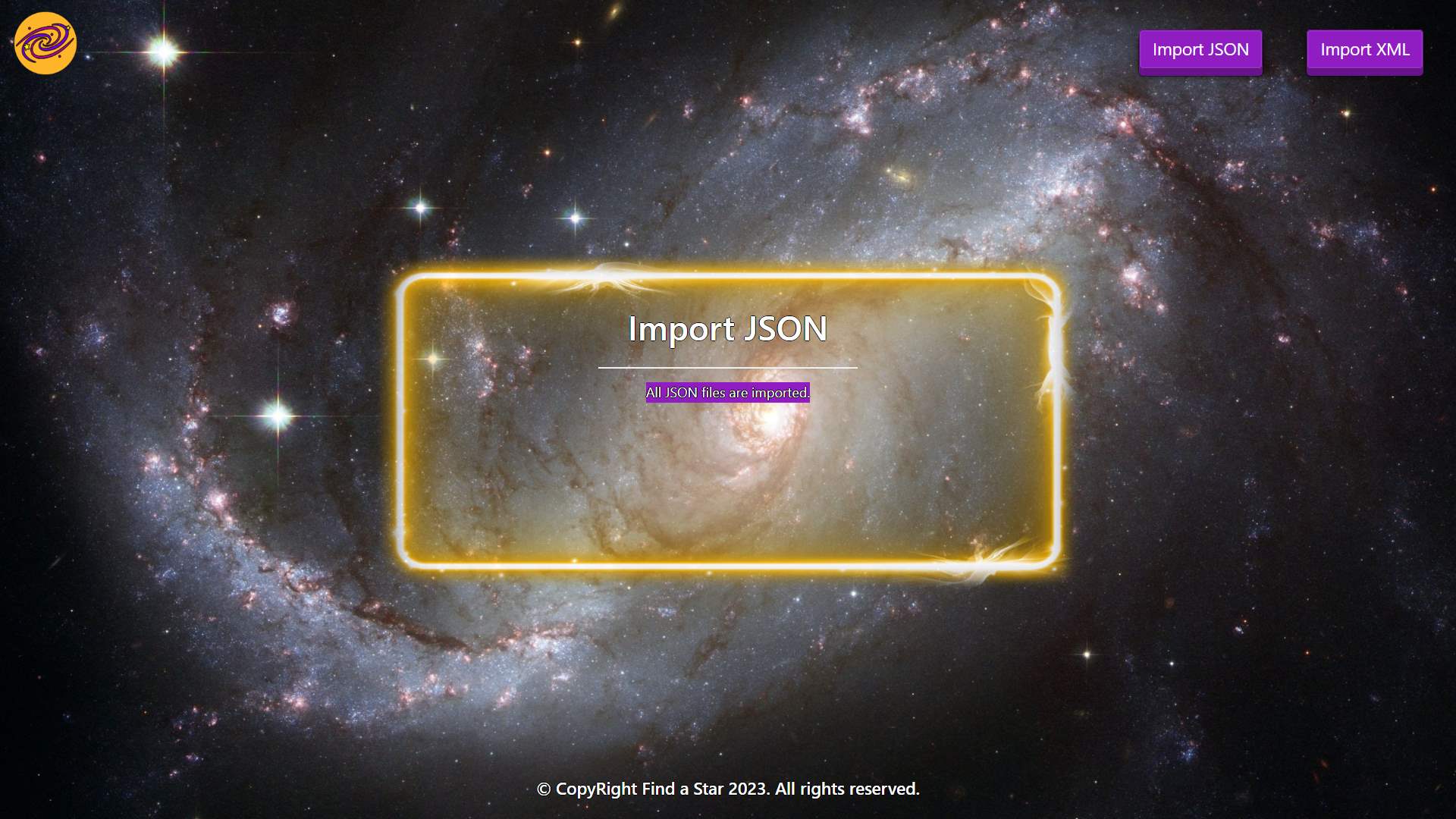
* Import the constellations first:



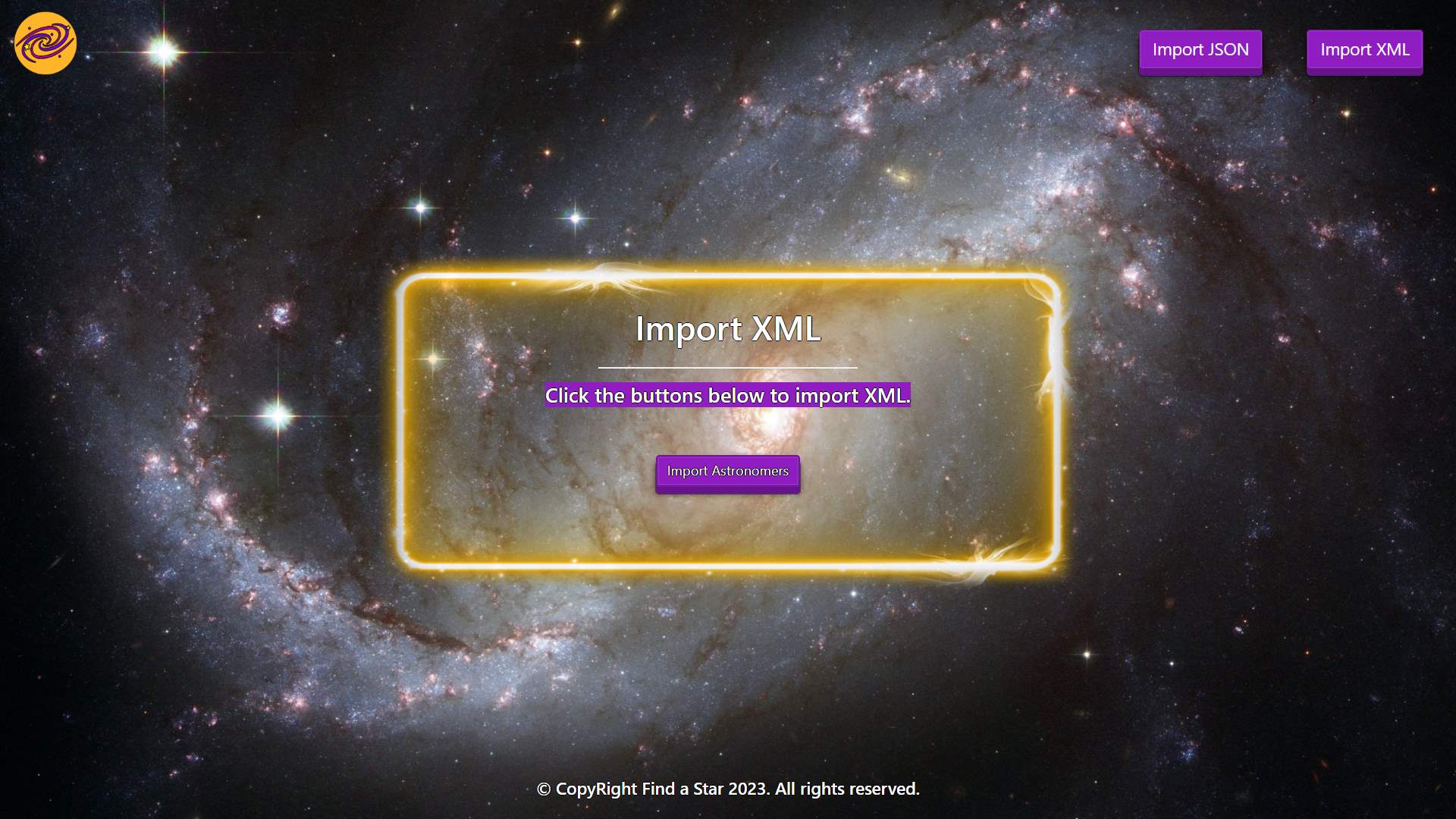
* Import the stars second:



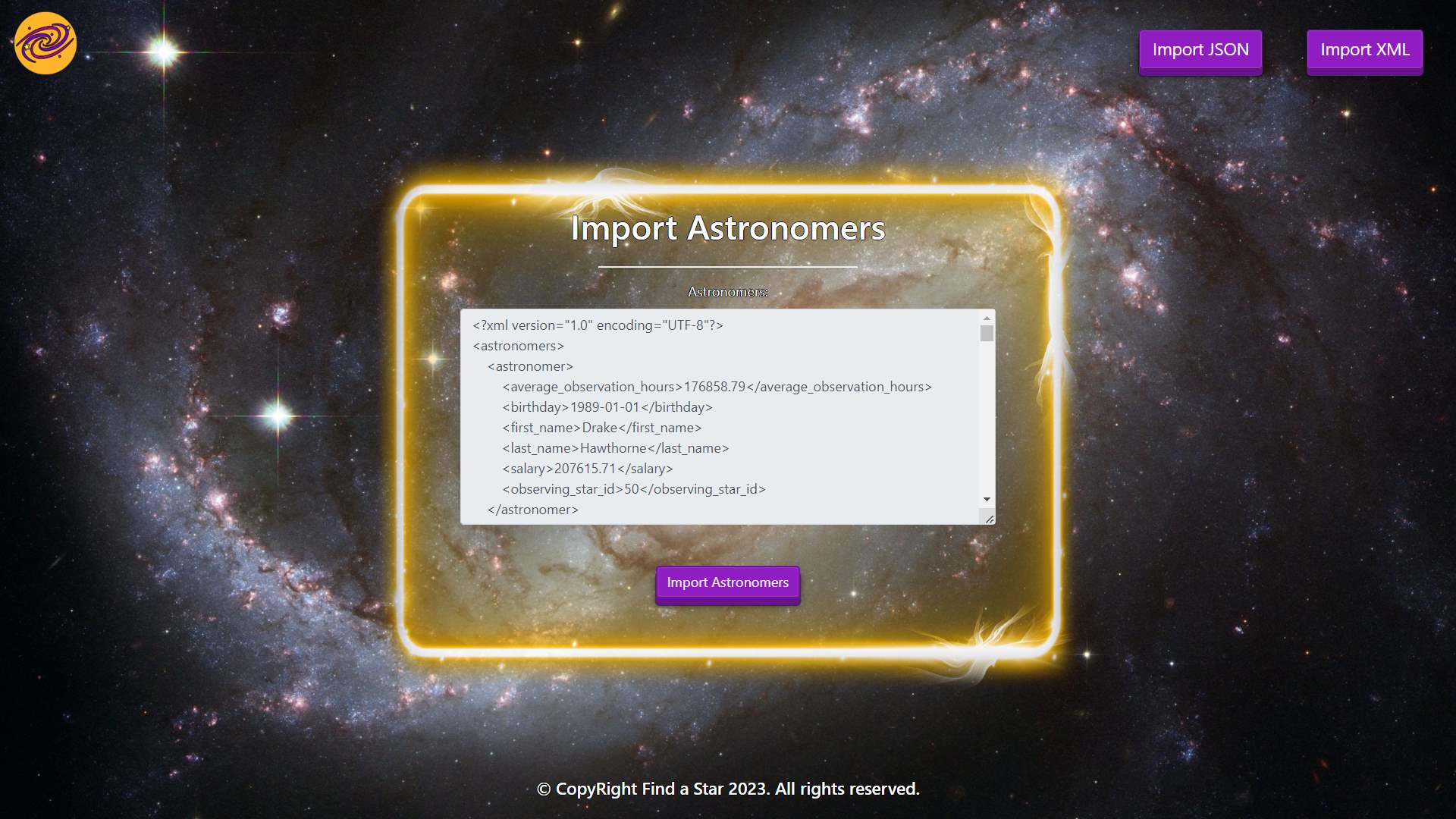
* The import JSON page after importing both files:



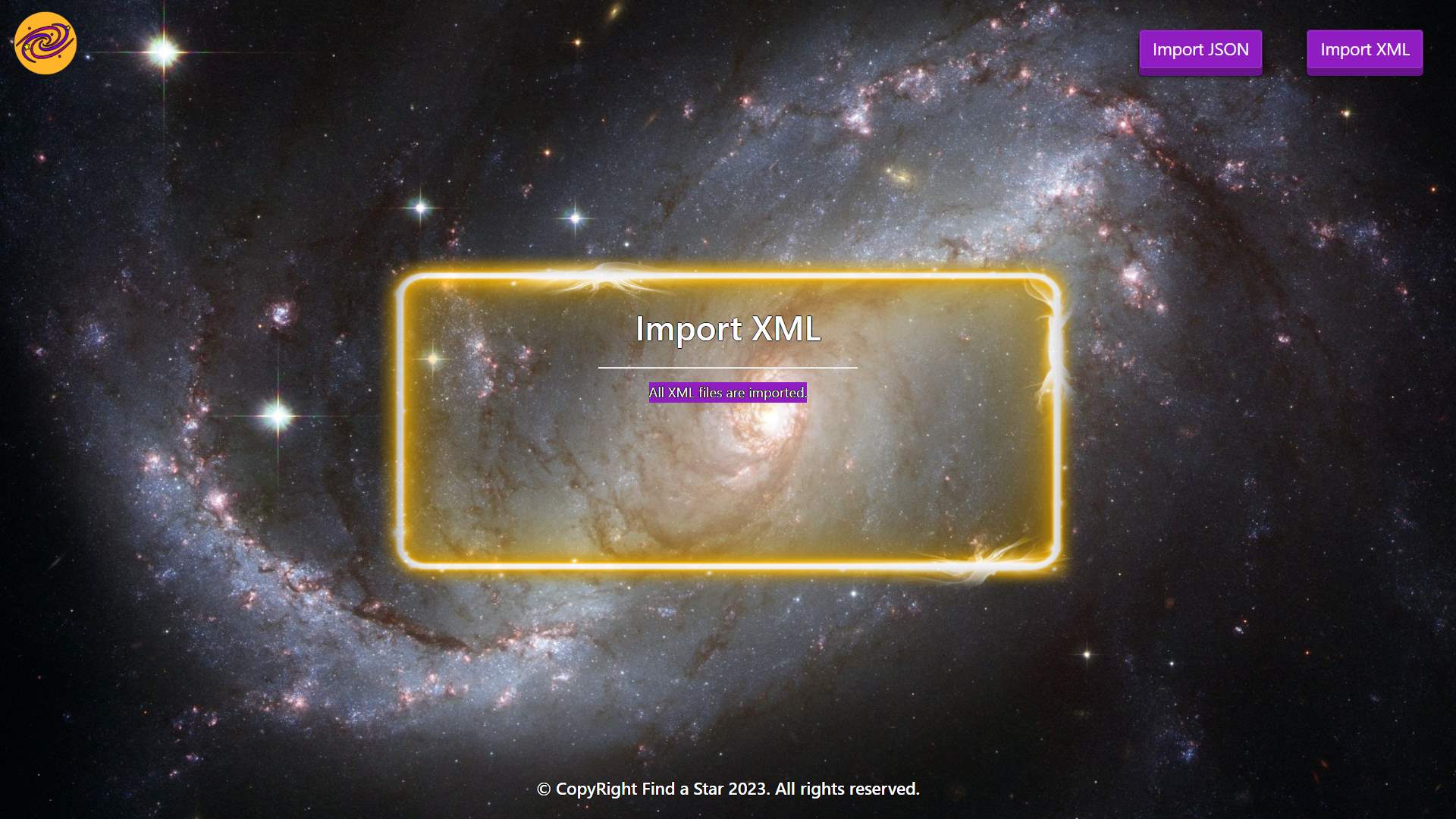
* The import XML page before importing the given data:



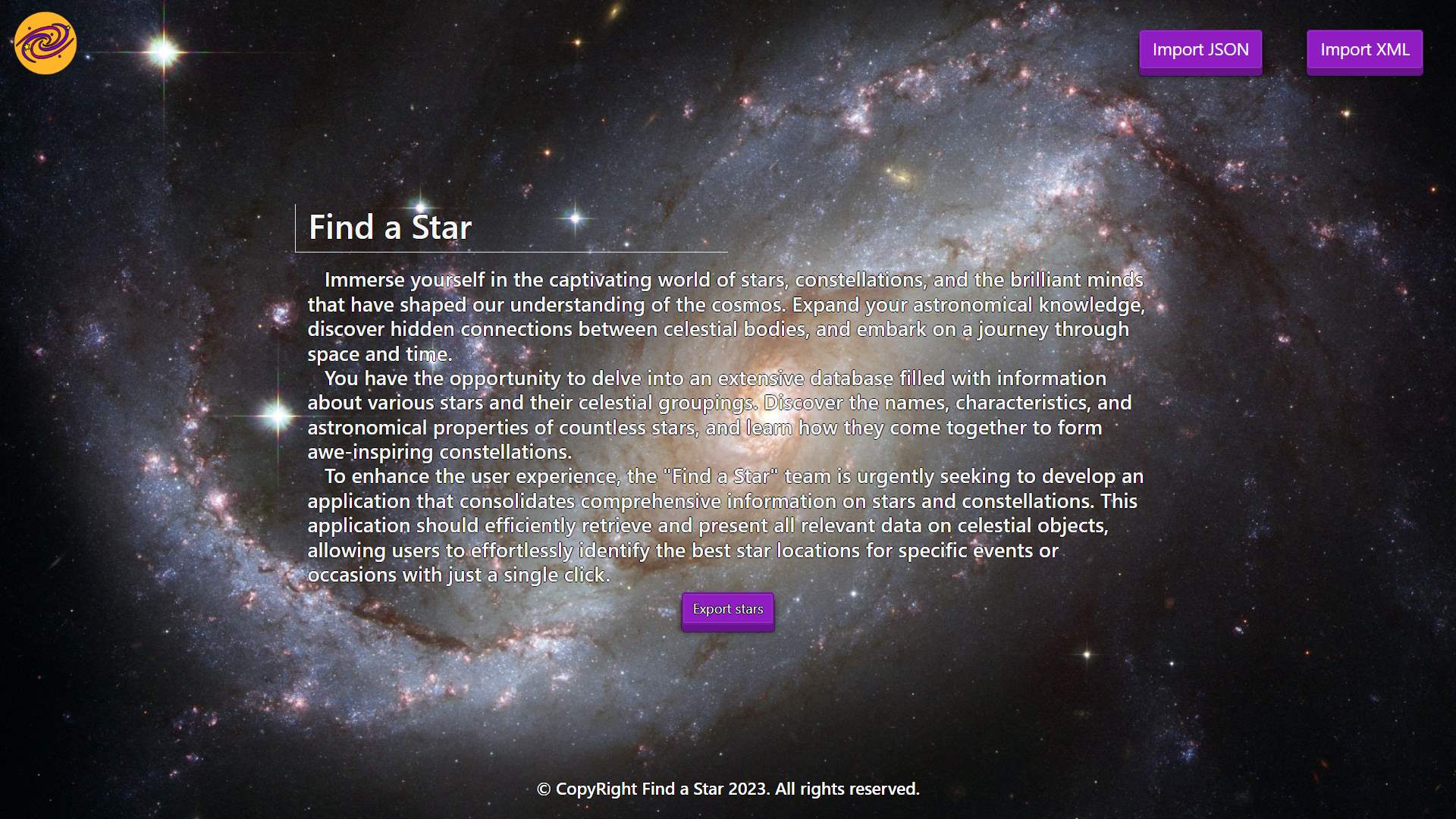
* Import the astronomers data:



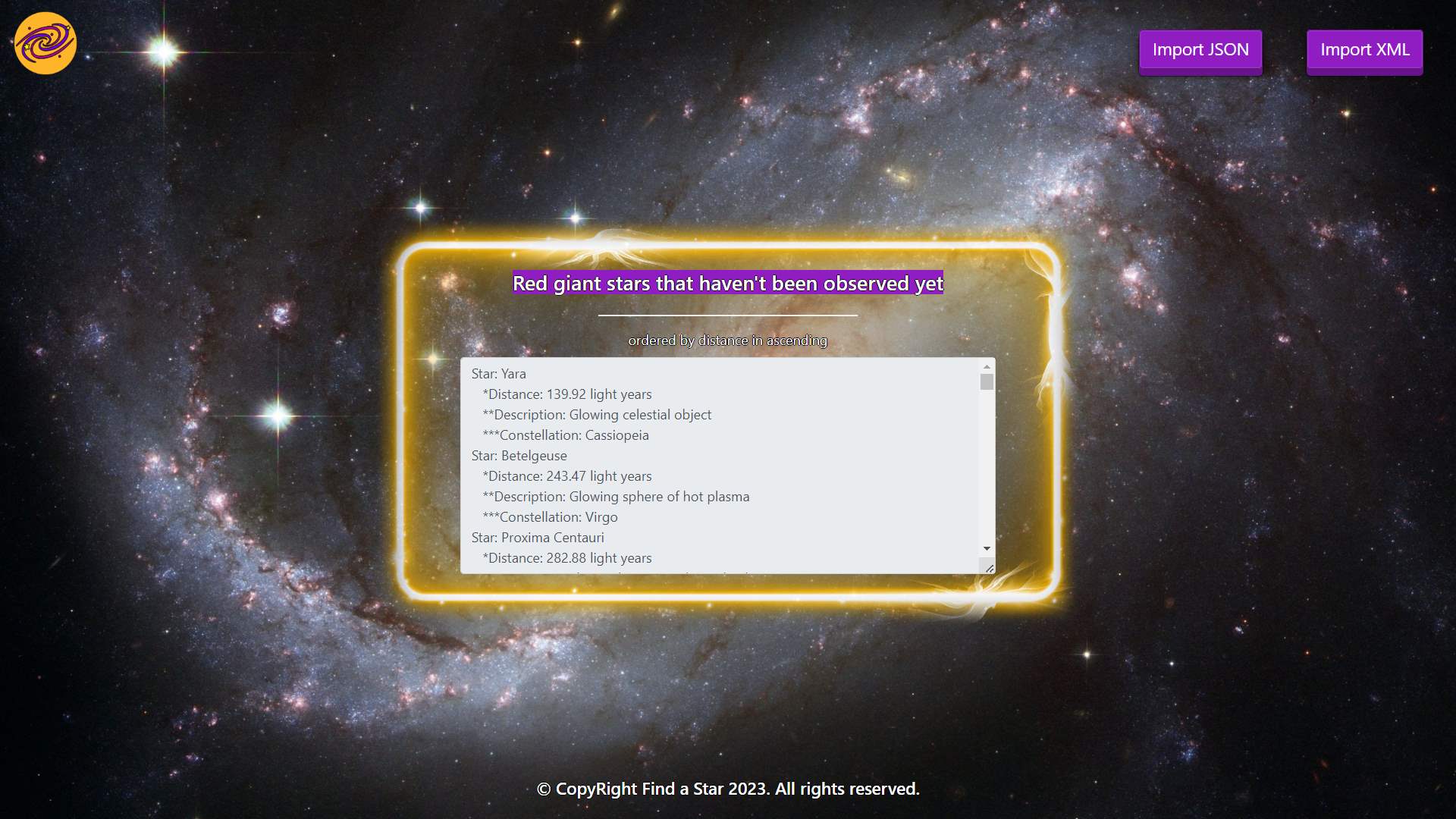
* The import XML page after importing the data:



* The home page after the data is imported:



* Export the non-observed stars:



## 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**.

**Don't change the skeleton.**

When submitting your project in Judge please make a **.zip** file only with **src** folder and **pom.xml** file.  
There are four problems in **Judge**:

**1. Database** will test the structure of your database.

**2. Service** will test the two methods **areImported()** and **readClassFromFile()** in every service class.  
**3. Import JSON** will test the logic of importing **JSON** files.

**4. Import XML** will test the logic of importing **XML** files.

**5. Export** will test the logic of the final **export** button which is in the **ExportController**.

Please note that the automated system Judge may take up to **5 minutes** to show results for each problem. We kindly ask you to be patient.

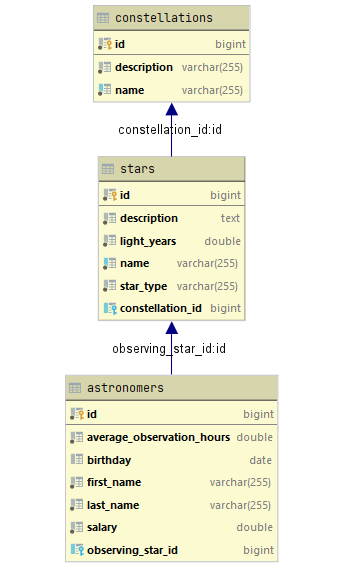
Be aware that Jude might **not** recognize the **var** keyword, leading to compilation errors.

To help you ensure the correctness of your solutions and **gain immediate feedback**, we have included a set of **commented-out tests** within the provided skeleton project. These tests are designed to verify the functionality of the tasks you are required to implement.

## Model Definition

There are 3 main models that the **Find a Star database** application should contain in its functionality.

We have provided you with an Entity-Relationship (ER) diagram that represents the database schema. The ER diagram displays the tables and their relationships, along with the field types for each table.



Ensure that the **Java** code **validates** data against the specified constraints before persisting it into the database. Handle any validation errors gracefully and provide meaningful feedback to the users.

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

### Constellation

* id – accepts **integer** values, a **primary identification field, an auto incremented field**.
* name – accepts **char sequence** (between **3** to **20** inclusive). The values are **unique in the database**.
* **description** - accepts **char** **sequence** about the naming of the constellation with a character length value higher than or equal to **5**.
* **stars** – a collection of all stars that are part the constellation
* Constraint: The constellations table has a relation with the stars table.

### Star

* id – accepts **integer** values, a **primary identification field, an auto incremented field**.
* name - accepts **char** **sequence** (between **2** to **30** inclusive). The values are **unique in the database**.
* light years - The distance from Earth in light years. Accepts only positive number.
* **description** - a long and detailed description about the star with a character length value higher than or equal to **6**.
* star type - categorization of the stars. String enumeration, one of the following – **RED\_GIANT, WHITE\_DWARF, NEUTRON\_STAR**
* **observers** – a collection with all the astronomers that are studying the star.
* Constraint: The stars table has a relation with the constellations table.
* Constraint: The stars table has a relation with the astronomers table.

### Astronomer

* id - accepts **integer** values, a **primary identification field, an auto incremented field**.
* first name - accepts **char** **sequence** (between **2** to **30** inclusive).
* last name - accepts **char** **sequence** (between **2** to **30** inclusive).
* salary - accepts number values that are more than or equal to **15000.00**.
* averageObservationHours - accepts number values that are more than **500.00**.
* birthday - a date in the "**yyyy-MM-dd**" format. Can be nullable.
* observing star - the current star that the astronomer is studying.
* Constraint: The astronomers table has a relation with stars 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 **Astronomer** may observe only one **Star**, but one **Star** may be observed by many **Astronomers**.

One **Constellation** can have many **Stars**, but one **Star** is part from only one **Constellation**.

#### 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.

## 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 {constellation / star / astronomer} should be printed."**

**When the import is finished:**

**"Successfully imported {constellation / star / astronomer} {name – description / name – light years/ full name – average observation hours}"**

**Judge** will only accept **file paths** in a specific format. When dealing with file paths for files, please adhere to the following format: "src/main/resources/files/xml/format-example.xml"

### JSON Import

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

#### Constellations (constellations.json)

##### Constraint

**If a constellation with the same name already exists in the DB return "Invalid constellation".**

|  |
| --- |
| **Constellation (constellations.json)** |
| [  {  "name": "Andromeda",  "description": "A princess chained to a rock, saved by Perseus."  },  {  "name": "Nebulosus Equinoxalis",  "description": "A scorpion with a prominent stinger."  },  {  "name": "Centaurus",  "description": "A half-human, half-horse creature wielding a spear."  },  {  "name": "Cassiopeia",  "description": "Queen punished for her vanity by being forced to sit on a throne."  },  {  "name": "Cassiopeia",  "description": "Description punished for her vanity by being forced to sit onX a throne."  },  {  "name": "As",  "description": "A maiden holding wheat or a sheaf of wheat."  },  {  "name": "Cepheus",  "description": "King and husband of Cassiopeia."  },  {  "name": "Wrongeus",  "description": "Bad"  },  {  "name": "Aquila",  "description": "The eagle carrying Zeus's thunderbolts."  },  ... |
| Successfully imported constellation Andromeda - A princess chained to a rock, saved by Perseus.  Invalid constellation  Successfully imported constellation Centaurus - A half-human, half-horse creature wielding a spear.  Successfully imported constellation Cassiopeia - Queen punished for her vanity by being forced to sit on a throne.  Invalid constellation  Invalid constellation  Successfully imported constellation Cepheus - King and husband of Cassiopeia.  Invalid constellation  … |

#### Stars (stars.json)

##### Constraint

* **If a star with the same name already exists in the DB return "Invalid star".**
* **The provided constellation ids will always be valid.**

|  |
| --- |
| **Star (stars.json)** |
| [  {  "description": "Glowing sphere of celestial gas",  "lightYears": 25.34,  "name": "X",  "starType": "WHITE\_DWARF",  "constellation": 18  },  {  "description": "Glowing sphere of celestial gas",  "lightYears": 25.34,  "name": "Sirius",  "starType": "WHITE\_DWARF",  "constellation": 18  },  {  "description": "Giant star",  "lightYears": 3197.45,  "name": "Galacticus Caelum Octavus Magnificensis",  "starType": "NEUTRON\_STAR",  "constellation": 12  },  {  "description": "Luminous beacon in the vast expanse",  "lightYears": 1378.32,  "name": "Spica",  "starType": "RED\_GIANT",  "constellation": 9  },  {  "description": "Luminous beacon in the vast expanse",  "lightYears": 1378.32,  "name": "Spica",  "starType": "RED\_GIANT",  "constellation": 9  },  {  "description": "Distant cosmic entity",  "lightYears": -63.24,  "name": "Its wrong",  "starType": "NEUTRON\_STAR",  "constellation": 14  },    ... |
| Invalid star  Successfully imported star Sirius - 25.34 light years  Invalid star  Successfully imported star Spica - 1378.32 light years  Invalid star  Invalid star  … |

### XML Import

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

#### Astronomers (astronomers.xml)

##### Constraint

* **If an astronaut with the same full name (first name and last name) already exists in the DB return "Invalid astronomer".**
* **If an astronaut is observing star that doesn't exist in the DB return "Invalid astronomer ".**

|  |
| --- |
| **Astronomers (astronomers.xml)** |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <astronomers>  <astronomer>  <average\_observation\_hours>176858.79</average\_observation\_hours>  <birthday>1989-01-01</birthday>  <first\_name>Drake</first\_name>  <last\_name>Hawthorne</last\_name>  <salary>207615.71</salary>  <observing\_star\_id>50</observing\_star\_id>  </astronomer>  <astronomer>  <average\_observation\_hours>300.79</average\_observation\_hours>  <birthday>1989-01-01</birthday>  <first\_name>Drake</first\_name>  <last\_name>Hawthorne</last\_name>  <salary>207615.71</salary>  <observing\_star\_id>50</observing\_star\_id>  </astronomer>  <astronomer>  <average\_observation\_hours>176858.79</average\_observation\_hours>  <birthday>1989-01-01</birthday>  <first\_name>Drake</first\_name>  <last\_name>Hawthorne</last\_name>  <salary>207615.71</salary>  <observing\_star\_id>50</observing\_star\_id>  </astronomer>  <astronomer>  <average\_observation\_hours>55537.43</average\_observation\_hours>  <birthday>1966-11-03</birthday>  <first\_name>Elena</first\_name>  <last\_name>Sullivan</last\_name>  <salary>98319.24</salary>  <observing\_star\_id>91</observing\_star\_id>  </astronomer>  <astronomer>  <average\_observation\_hours>105443.87</average\_observation\_hours>  <birthday>1954-09-02</birthday>  <first\_name>Cassandra</first\_name>  <last\_name>Bellamy</last\_name>  <salary>1499.52</salary>  <observing\_star\_id>28</observing\_star\_id>  </astronomer>  ... |
| Successfully imported astronomer Drake Hawthorne - 176858.79  Invalid astronomer  Invalid astronomer  Successfully imported astronomer Elena Sullivan - 55537.43  Invalid astronomer  … |

Please be aware that due to variations in local settings on different computers, the representation of decimal numbers (Double) may differ. In some regions, the decimal separator is a comma (,), while in others, it is a dot (.).

Judge local settings represent all decimal numbers using a dot (.) as the decimal separator. *Locale.US*

## 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 Non-Observed Stars from the Database

* Extract from the database, the **star name, distance in light years (to second digit after decimal point), description and the constellation name**.
* **Filter only stars who are Red Giants and have never been observed and order them by the light years in ascending order.**
* Return the information in this format:

**"Star: {starName}**

**" \*Distance: {lightYears} light years**

**" \*\*Description: {description}**

**" \*\*\*Constellation: {constellationName}**

**. . ."**

