15/01/2023
SUPSI
Miro Rava
Oleg Lastocichin

Rocket Space Stuff

Technical Report

Project by Miro Rava and Oleg Lastocichin for SUPSI's DS&AI 3rd semester.

Teachers of the module in interest:

- *Guidi Roberto*
- *Rubegni Elisa*

Contents

Rocket-Space-Stuff

15/01/2023
SUPSI
Miro Rava
Oleg Lastocichin

Introduction

Our project centers around the topic of space-related subjects. The goal of this project is to develop a web service that offers an API centered around this topic. To achieve this, we will be utilizing the Flask web framework and incorporating our library to enhance the functionality of the web service. Ultimately, we aim to create an app that seamlessly interacts with the web service, making it more user-friendly and allowing for easy data transmission and retrieval.

Background

We are students of SUPSI studying in the DS&AI bachelor's degree this is a project we made for the purpose of creating a webservice. We used the information that our teachers gave us such that we could make a project that meets their expectations or needs, we have somewhat knowledge of programming by previous studies but not on this narrow topic, we did instead work on private small projects before this one that were implemented using an app and online requests, but this project has been by far the most time consuming and important of these all, our biggest group work in SUPSI done till now.

Requirements

Libraries/languages used for the creation of our own library:

- Python 3.10.*
- flask==2.2
- pandas==1.5.2
- jupyter==1.0.0
- requests==2.28.2
- pytest==7.2.1
- tk==0.1.0

Python was a clear choice for the programming language since it’s the only one that we studied until now.
We used flask to manage the webservice and be able to create it, together with pandas we made it working thanks to Pandas functions that permit to be able to manage our datasets without having to create a database.

Jupyter was using to initially test our dataset and cleaning it and was used to manually test our functions.

Requests were used for making our library use the webservice and making them able to communicate with each other.

Pytest was used to test our webservice and library.

Tk is the library we used to make the main demo app.

15/01/2023
SUPSI
Miro Rava
Oleg Lastocichin

Data Gathering and Managing

The data was gathered from the public API https://thespacedevs.com/llapi , we stored it locally and built our project around it.

The datasets we are going to use are:

- Launches (Event or mission that was done)
- Launchers
- Astronauts

These were gathered using their API, the reasoning around this choice was the usefulness that this data could give and our common interest on such topic.

We have employed the use of the popular Python library, Pandas, in order to efficiently manage and manipulate the data used by the web service app. Pandas is a powerful and widely-used library for data manipulation and analysis that provides a variety of data structures and data analysis tools for handling large and complex datasets.

By utilizing Pandas, we are able to effectively manage and manipulate the data stored in the app's database. This includes tasks such as filtering, sorting, and aggregating data, as well as performing advanced data analysis.

We cleaned the data and used the columns we're going to be interested in having without overcomplicating the libraries.

Web service

The web service app is a Flask-based application that allows users to interact with various space-related data through a series of RESTful API endpoints. The app leverages various custom-built functions to provide users with the ability to search, retrieve, add, update, and delete data about astronauts, launches, and launchers.

The app's API endpoints are divided into three main categories: astronauts, launches, and launchers. Each category includes a variety of endpoints that allow users to perform different actions, such as searching for data, retrieving specific data, and adding, updating, and deleting data.

- "/{name of the dataset}/search"
  with "GET" method this permits the user to search in one of the three datasets ex: /astronauts/search?name=a
- "/{name of the dataset}/<column_name>/<value>"
  with "GET" method permits to search for an exact value ex /astronauts/id/83
- "/{name of the dataset}"
  with "GET" method permits to get all the dataset, the output will be in json format ex /launches

- “/{name of the dataset}”
  with “POST” method permits to insert in the dataset, the input must be a JSON
- “/{name of the dataset}/<column_name>/’<value>’”
  with “DELETE” method permits to delete an exact value from the dataset ex /astronauts/id/740
- “/{name of the dataset}/<column_name>/’<value>’”
  with “PUT” method permits to update an exact value in the dataset the input must be a JSON

These endpoint allow users to access the core functionality of the web service.

Library

Rocket Space Stuff is a powerful and versatile Python library that provides an easy-to-use interface for interacting with various space-related API endpoints. With this library, developers and users can easily retrieve and manipulate data related to astronauts, launches, and launchers.

The library is designed to be highly intuitive and easy to use, making it a great tool for researchers and scientists working in the field of space exploration. The library allows users to find, add, update, and delete data about astronauts, launches, and launchers, as well as retrieve detailed information about each.

One of the key features of the Rocket Space Stuff library is its ability to search and filter data using a variety of parameters. For example, users can search for astronauts by name, birth date, or agency, and filter launch data by date or location. This powerful functionality makes it easy to perform complex data analysis and gain valuable insights into space-related data. In addition to its powerful data retrieval capabilities, the Rocket Space Stuff library also includes a number of tools for working with and manipulating data. For example, the library includes methods for converting data to JSON format, as well as methods for retrieving basic information about astronauts, launches, and launchers.

Overall, the Rocket Space Stuff library is a powerful and versatile tool for working with space-related data. Whether you're a researcher, scientist, or developer, this library is an invaluable resource for gaining insights into the exciting and rapidly-evolving field of space exploration.

Specific Features:

The library offers a wide range of features for working with space-related data:

- Astronaut class: The astronaut class provides a representation of an astronaut and includes parameters such as name, age, and experience. It also includes basic functions such as getting the astronaut's name and age etc.

- Launch class: The launch class provides a representation of a launch event and includes parameters such as launch date, rocket. It also includes various characteristics such as getting the launch date and rocket information.

- Launcher class: The launcher class provides a representation of a launcher, including parameters such as name, country, and launch history. It also includes basic functions such as getting the launcher's name and country.

- SpaceApi class: The SpaceApi class is the main class of the library, it provides an interface to interact with the web service, it includes functions to retrieve data from the web service and use it to instantiate the above-mentioned classes and perform operations with the data.

The library provides a simple and flexible way to work with space-related data and interact with the web service. It allows developers to retrieve data from the web service, create classes to work with the data, and perform operations with the data in an easy and efficient way.

Extended documentation, usage, and installation methods are available at this link:

https://pypi.org/project/rocket-space-stuff/

It is highly suggested to check out the extended documentation in order to learn how the library works.

We decided to publish the library in order to learn also how to package python files into wheels and how to publish them using Twine

The library can be installed by executing this command in the terminal:

```
python -m pip install rocket-space-stuff
```

App

Our Python GUI app, built using the Tkinter library, provides a user-friendly interface for interacting with the data provided by our space library, Rocket Space Stuff. The app allows users to easily navigate and visualize data related to astronauts, launches, and launchers, providing a convenient and accessible way to access and analyze space-related data

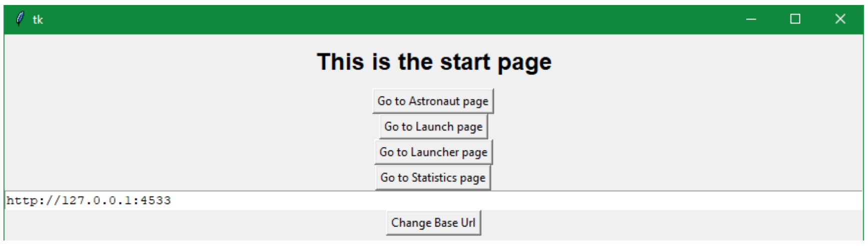
The Tkinter library allows us to create a graphical user interface (GUI) that is intuitive and easy to use. The app includes a variety of features such as search, find, post, delete and update data functionality. Users can perform data analysis and gain insights into space-related data by using the GUI, making it an easy-to-use tool for non-experts, and spreading the use of our database and library.

By integrating the functionality of the Rocket Space Stuff library into the GUI, we managed to simplify all the calls to the web service, making them easy for everyone.

The app fully exploits all the functionality provided by the Rocket-Space-Stuff library:

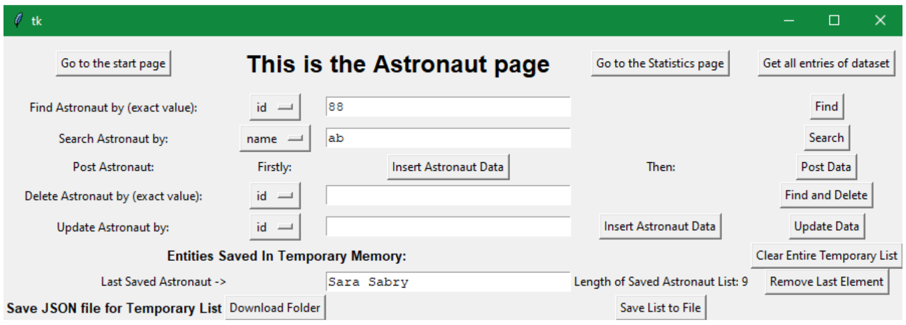
Main Page:

On the main page you can set the base URL for the API endpoints of the web service. There are also four buttons that can take you to the other pages.



Astronaut, Launch, Launcher Pages:

In these three pages all the CRUD (Create, Read, Update, Delete) methods to interact with the database through the web service are provided.



Find Astronaut by (exact value): id 88
Search Astronaut by: name ab
Last Saved Astronaut -> Sara Sabry
Length of Saved Astronaut List: 9

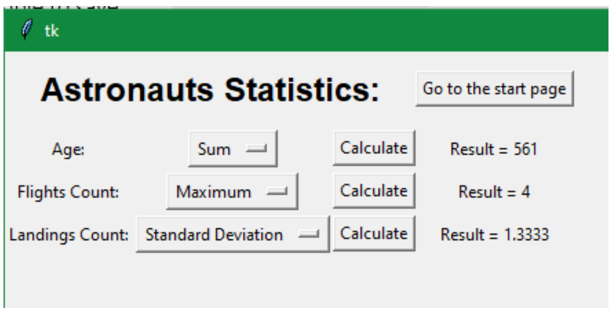
It is also possible to save the current element or more than one into a temporary list, on which you can perform statistics operations in the respective pages.

The Temporary List can be cleared, and the last element of the list is always displayed. Once the user puts some elements into the list, it is possible also to save the current list as a JSON file in a folder selected by the user.

Statistic's Pages

Inside these pages the user can perform operations on certain fields of the Entries in the Temporary List.

These are only simple Calculations, but are used as a proof of concept for what possibilities this approach to the database can offer.



Age: Sum Result = 561
Flights Count: Maximum Result = 4
Landings Count: Standard Deviation Result = 1.3333

Testing

The library and web service have been tested using the Pytest framework. Pytest is a popular testing framework for Python that makes it easy to write tests and run them. It provides a variety of useful features such as test discovery, parameterized tests, and detailed test reports.

The tests were written to cover a wide range of functionality in the library and web service, including the retrieval of data from the web service, the instantiation of classes, and the execution of basic functions. The tests were run multiple times to ensure that the library and web service are functioning correctly and without any bugs.

Manual testing was also performed on the application to ensure that it is easy to use and that all of the features are working as expected.

However, as with any software, it is possible that some bugs may still be present. If you encounter any issues or bugs while using the library or web service, please report them to the developers so that they can be fixed. Be sure to include as much information as possible, such as the steps to reproduce the problem, the expected behavior, and the actual behavior.

15/01/2023
SUPSI
Miro Rava
Oleg Lastocichin

Conclusions

In conclusion, the library and web service developed in this project provide a robust tool for working with space-related data. The choice of programming language, Python, and the use of popular libraries such as Flask, Pandas, Jupyter, Requests, Pytest, and Tk allowed for the creation of a functional and efficient tool. The data was gathered from a public API and was cleaned and used to build the project. The web service, implemented with Flask, allows for basic CRUD operations on the data, while the library provides classes for working with the data and interacting with the web service. The library and web service were tested using Pytest and manual testing, and the app was manually tested to ensure ease of use.

Any bugs encountered should be reported to the developers for fixing.

References

- https://stackoverflow.com/
- https://realpython.com/
- Slides on the subject provided by Prof. Guidi Roberto