**Flight Information API Using Postman and Strapi**

**Introduction**

In this module's assignment, an API dedicated to providing comprehensive flight information will be created. The chosen tool for this task is Strapi, a versatile and user-friendly headless CMS that enables the effortless design and publishing of APIs. Through Strapi's capabilities, a robust Flight Information API will be constructed, catering to the requirements of both travelers and aviation enthusiasts.

Throughout the assignment, essential steps will be followed to build the API, including defining collection types, establishing relationships, and configuring data fields to capture pertinent flight details. Strapi's intuitive interface will be utilized to develop the fundamental components, transforming the API into a reliable source of information for airlines, airports, and flight data.

Using Postman's precision and Strapi's efficiency, the objective is to create a comprehensive Flight Information API that enables seamless access to valuable travel insights. By leveraging technology, this assignment aims to facilitate smoother journeys for travelers worldwide, empowering them with essential flight data for their travel planning needs. Let's now embark on this exciting journey of API development together.

**Steps**

The first step is to create a collection type named "Airline". For the field type, "Text” is selected, and for the field name, "Name" is written.

A screenshot of a computer

Description automatically generated

Then another collection is created named "Airport" and the following text fields are added: "AirportCode", "AirportName", "Country", "State", and "City".

A screenshot of a computer

Description automatically generated

Then, a collection named "Flight" is created, and the following fields are added:

1. "FlightNumber" as a text field.
2. "Airline" as a Relation to the Collection Type "Airline". Flight has one airline.
3. "Seats" as a number field.
4. "OriginAirport" as a Relation to the Collection Type "Airport". Flight origin has one airport.
5. "DestinationAirport" as a Relation to the Collection Type "Airport". Flight destination has one airport.

A screenshot of a computer

Description automatically generated

Three rows of real flight data have been entered for each table in the Flight collection, retrieved from <https://flightaware.com/>. By publishing each collection, the data is now accessible through the API, allowing users to access valuable flight information.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated Additionally, the necessary steps have been taken to modify the permissions for the application plugin, granting it "Select all" permissions for each collection. These permissions, including "count," "create," "delete," "find," "findone," and "update," empower the API to efficiently perform various operations, catering to the needs of users interacting with the Flight Information API.

A screenshot of a computer

Description automatically generated

Then each collection is published.

A screenshot of a computer

Description automatically generated

Now, the collections can be viewed by accessing the corresponding URL for each of them through a web browser. This functionality allows easy observation of the data and information stored within the "Airline," "Airport," and "Flight" collections directly in the browser. The respective URLs will provide an easy-to-read representation of the flight data available in the Flight Information API.A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Using Postman, a GET request is sent to retrieve all the data from the Flights application. This GET request enables access to and retrieval of the entire dataset available in the Flights collection of the Flight Information API.A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A GET request can be sent to retrieve the flight with ID = 1. To do this, simply a GET request is sent to the API endpoint that corresponds to flights and the ID is included as a parameter in the URL, like this: GET <http://localhost:1337/flights/1>. But this can also be achieved by the following.

A screenshot of a computer

Description automatically generated

Next, a POST request is made to create a new flight. To do this, it is started by adding a header with the content-type set to "application/json."

A screenshot of a computer

Description automatically generated

Afterward, the body of the request is written in JSON format, providing the necessary flight details. For the "Airline," "DestinationAirport," and "OriginAirport" fields, only the ID of each parameter is entered. These IDs serve as references to the corresponding records in the "Airline," "Airport," and "Flight" collections, linking the flight data to the specific airline and airport information in the Flight Information API.A screenshot of a computer

Description automatically generated

The newly created flight can be viewed in the browser by navigating to the flights collection. Upon pointing the browser to the flights collection URL, the details of the flight that was just created can be seen, confirming its successful addition to the Flight Information API.A screen shot of a computer

Description automatically generated

An airline named “American Airlines” is created using Strapi.

A screenshot of a computer

Description automatically generated

Now, the newly created flight with ID 5 is being updated. The original Ryanair flight is being changed to American Airlines, and the number of seats is being updated to 60.A screenshot of a computer

Description automatically generated

Changes can be viewed in the browser by navigating to the flights collection.

A screen shot of a computer

Description automatically generated

It can also be seen in Strapi.

A screenshot of a computer

Description automatically generated

And in Postman.

A screenshot of a computer program

Description automatically generated

This wrong flight is deleted using DELETE request.

A computer screen shot of a black screen

Description automatically generated

Changes can be viewed in the browser by navigating to the flights collection.

A screenshot of a computer

Description automatically generated

It can also be seen in Strapi.

A screenshot of a computer

Description automatically generated

The Python code to interact with the API can be retrieved from Postman. This can be done for both retrieving all data and accessing the airlines application.

Airlines application:A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

All data (populated):

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

**Conclusion**

In conclusion, the Flight Information API project successfully utilized Strapi to create and publish an API for managing flight-related data. The project involved setting up three collection types: "Airline," "Airport," and "Flight," each with their respective fields and relations. Real flight data was entered into the collections, and appropriate permissions were configured to ensure smooth access and operations within the application.

Through Postman, various HTTP requests were performed to interact with the API, including creating, updating, retrieving, and deleting flights. The API was also accessed through web browsers, allowing for easy visualization of the flight data.

Furthermore, the Python code generated from Postman facilitated seamless interaction with the API, enabling data retrieval from both the "all data" and "airlines" applications.

Overall, the Flight Information API successfully demonstrates its capability to serve as a reliable and efficient source for accessing and managing flight-related information. The project showcases the importance and versatility of APIs in modern applications, providing users with valuable insights into flight details and fostering a seamless user experience.

With this project completed, there are opportunities to expand and enhance the API's functionality in the future, catering to the needs of aviation enthusiasts, travelers, and various stakeholders within the aviation industry.

In conclusion, this project exemplifies the power of APIs and their significance in the realm of flight information management, presenting a valuable tool for aviation-related applications and services.