Christian Hummel

Hammertrips – An API for a traveling agency

Installation:

Clone the repository and run “pip install -r requirements.txt” in the terminal to get all the necessary dependencies

To run the program execute “python start.py” in the terminal or open the “start.py” file and <right click><Run ‘start’>

After “logging in” as a supervisor, a jwt access token will appear and the contents of the string has to be copied inside the text field after clicking on a lock symbol or the “Authorize” button. The jwt access token has to be inserted in the following format:

”Bearer <jwt access token>”

General Information about the application:

This application is meant to simulate the possible processes of a Travel agency with a predefined hierarchy in place: There is one owner of the agency (not accounted for this application) and he can employ many supervisors. Each supervisor has got his own team of employees, who are called travelAgents. One travelAgent can be assigned to as many customers as his supervisor wants to, but a customer can be assigned to one travelAgent only.

There can be many Customers and they can have:

- A preference for a specific country:
- A preference for a specific country and require an expert later on
- No preferences for a specific country

After a supervisor is registered to the agency, he needs to create an account to unlock all the necessary features for adding a travelAgent, getting information about customers, other supervisors and travelAgents etc.

Countries and Activities are combined in one Namespace and are meant to be available for travelAgents and supervisors. This way, a travelAgent as well as a supervisor can add register a country and activities and check their stats to be able to optimize offers to a customer. It would make sense to make certain functionality like adding countries and activities only available to a supervisor, but I think this way would be more realistic since the supervisors most likely do not have too much time to be scouting for new countries and activities as well.

To possible shorten this Readme, I will start with the design choices/personal Notes and present each route afterwards, some of them are straightforward, so you can either choose to have a quick look of the following overview of all the routes implemented in the application or skip the last section and have a look at them in swagger/pythonIDE

Design Choices:

Supervisor

The supervisor is the "heart" or the most powerful user of this application.

A supervisor is also able to have a look at other supervisors of this agency to see if there are more than only himself. It is also possible to take a closer look at one of them, to see the number of team members for the requested supervisor. If a supervisor logs in, it is also possible to extract the attached supervisor_id(employee_id) of this person, since it had to be included in the registration for the user account.

A supervisor, once registered, can add travelAgents to the agency and assign any number of customers and countries to them. When a travelAgent is added, the country matched to the nationality of this agent is inserted as the first element of the countries list.

If the "expert country" – nationality of a travelAgent is not already registered for this agency, then an instance of this country will be created automatically. You could argue that it makes no sense to employ a travelAgent who is an expert for a country not registered in the agency, but there could be other arguments to still accept this person as a team member. For example, the travelAgent in question could have been working for another agency for a long time and in order to get an already experienced new employee you will have to make some compromises or the scouting for new activities for that country will start shortly afterwards.

Then other countries can be registered for a travelAgent to be used for offers or to be assigned to customers with special preferences. A supervisor can get general information about a travelAgent, or have a detailed view of statistics, including number of customers, number of successful trips booked and total revenue earned for the company.

A supervisor can display all customers registered in the agency, to see if there are customers who are in need of a travelAgent, or to see which customer is assigned to which travelAgent.

A supervisor is able to raise the salary of a travelAgent, but only if the agent sent him a request and the travelAgent has earned more than 5000 revenue at least. Originally I was planning to include a minimum amount of trips as well to be eligible for a raise, but I previously learnt that revenue is one of the driving factors in a commercial business. I was thinking about implementing a choice, between a flat amount that gets added one time and a percentage that will increase the monthly earnings. I decided against it, because the salary is intended to display the monthly earnings and it is supposedly more realistic to increase it by a certain percentage.

A supervisor is able to lower the total price of a previously offered trip to a customer, but only if the status of the offer got changed to budget and the supervisor got a request from the travelAgent to do so. Then if a 0 is entered in the percentage cell, the recommendation from the travelAgent is accepted and the total price will be lowered. A travelAgent can only request for a maximum of 40 percent, it is possible for the supervisor to overrule the provided percentage of the request, so the final discount can be lower than requested, or even higher but to a maximum of 50 percent. The status of the offer will be set from “budget” to “changed” and will be immediately presented again to the customer. This way, it would be possible to lower the total price for an offer multiple times until you get close to zero, but in reality a travelAgent and supervisor would know how much is needed to meet the budget of the customer, so I did not implement additional constraints for that.

Deletion of a travelAgent can get complicated, since a travelAgent could have customers assigned to and they have to be reassigned to other travelAgents, so it is designed to only being able to remove a travelAgent if all of his customers, including the offers that are still available for changes – this excludes offers with status accepted and declined - , can be transferred to other travelAgents of the corresponding team of his or her supervisor. Removing the last travelAgent is possible, since the supervisor is able to just employ new travelAgent at any time.

TravelAgent

The basic idea is to let the travelAgent concentrate on the main parts of his work that he got employed for, to create revenue for the company. So administrative tasks will be mainly done by the corresponding supervisor and the agents can focus on creating and managing offers for their customers.

A travelAgent can just update his or her name and address, since all the other attributes are either predefined, require higher authority to change, or follow a certain format that cannot be changed.

For creating offers, if a 0 gets inserted as a value for the offer_id, it will be counted as a new offer and if it passes all the validations, it will be presented to the customer via the Offer table. If a customer does not decline an offer, but wants a different one, then the status of this offer will get changed to “resend” and only if this is the status of the offer, it can be resend, otherwise it will throw an error.

So the pattern for sending offers is something like this:

Create or change offer – send to customer – wait for reaction – depending on the reaction of the customer resend offer or request discount

If an offer gets declined it will be declined forever and a totally new offer needs to be created

I was planning to create more advanced functionality for the requests of a travelAgent, namely request_raise and request_discount. I was thinking about using multithreading at first, or looking more closely at other libraries like for example flask-socketio, but in the short version I am lacking in ability to make it work at the moment. I am not believing in something like my approach with the message table is better than nothing, I just wanted to do it for “completion” and will have to accept any penalty for this.

Flask Mail would have been my first choice for this, since I was already part of a business where also the internal communication was mainly handled via emails.

Offer

An offer can be checked by either a travelAgent or a customer, it can have one of these “status codes”:

“pending” – that means an offer was created and sent to a customer

“resend” – a customer does want an improved version of this offer

“changed” – an improved version of an offer is sent again to a customer

“budged” – the total price of an offer exceeds the budget of a customer

“declined” – a customer cancels the trip

“accept” – a customer accepts the offer and it counts as visited for the statistic table of the corresponding TravelAgent

Offers can be transferred to another travelAgent, only if a travelAgent gets removed from the agency. Offers with the status “accepted” or “declined” will not be transferred to another agent and they can still exist in the database, even if the corresponding travelAgent got released. This way, a supervisor can still have a look at certain offers to see what country might be well accepted for customers who have got no preference or to see what was bad when looking at declined offers.

Customer

A customer is able to register him or herself to the agency, request the assistance of an expert of this country and get an overview of all the offers available for this person and to handle them.

For the /customer/<customer_id>/offer/<offer_id> route possible options are:

“accept” – accept an offer

“change” – request a new version of this offer

“decline” – cancel this offer

To allow travelAgents to resend an offer for a different country than the preference of the customer is kind of a shortcut, but I did only think about the possibility of updating a customer when I finished this application, in an improved version, it would be possible for a customer to update their data including preferences, so that it is possible to receive new offers in accordance with their new preferences, but only if they do not have Offers of the status “changed” or “pending”.

Country/ Activity

Meant to be accessible for Supervisors and TravelAgents, as a first step a new country will be registered to the agency and after that activities can be added to the catalogue of this country. Some activities are unique to a country and some activities will be available for multiple countries with the same identifier, but they can have different prices for each country. There would be certain possibilities for categories, seasons and other extensions for activities, but I tried to keep it very simple to be able to concentrate on the more important functionalities of this application.

Calculations for the country stats is done via a custom json object, because it would be easier this way than having to implement statistics tables for Country and Activity individually and to increment their counters for methods like adding a country or adding an activity for a country.

General comments:

I am aware that I am using error code 400 for the abort function every time in the namespaces and there are many more options, like code 404 if something is not found, but I chose to do it this way, because then the error messages will be clearer this way. (For error code 404 for example you get a suggestion for an alternative url, but this does not match the context of the error most of the time)

I did recognize way too late that I am using the legacy style of queries for sqlalchemy, and continued to do so as a matter of consistency.

Structure of the routes for each namespace follows on the next page

Structure of the routes for the namespaces

Supervisor

Routes of Supervisor Namespace:

Post:

/supervisor/ – *not restricted, no account needed*

Adds a supervisor to the agency

/supervisor/<supervisor_id>/register – *not restricted, no account needed*

Creates an account for a supervisor in order to gain access to more functionality

/supervisor/login – *not restricted, no account needed*

Creates the access token needed for utilization of the manager functions

/supervisor/employee

Adds a new employee to the agency

/supervisor/agent/<employee_id>/assign

Assigns a customer to a travelAgent

/supervisor/agent/<employee_id>/country

Assigns a country to a travelAgent

/supervisor/offer/<offer_id>/discount

Lowers the total price of an offer

/supervisor/agent/<employee_id>/raise

Raises the salary of an offer

Get:

/supervisor/managers

Displays all managers in the agency

/supervisor/<supervisor_id>/info

Displays information about one manager

/supervisor/team

Shows all travelAgents connected to this supervisor

/supervisor/agent/<employee_id>

Displays information about one travelAgent

/supervisor/agent/<employee_id>/stats

Displays Information about the achievements (trip successes) of a travelAgent

/supervisor/customers

Displays all registered customers and their status

/supervisor/customer/<customer_id>

Displays information about one customer

/supervisor/inbox

Opens the message board of this supervisor

Delete:

/supervisor/agent/<employee_id>/remove

Removes a travelAgent from the agency

TravelAgent

Routes of TravelAgent Namespace:

Post:

/travelAgent/<employee_id>/update

Updates personal information about a travelAgent

/travelAgent/<employee_id>/offer

Sends an Offer to a customer

/travelAgent/<employee_id>/offer/<offer_id>/discount

Sends a request for a discount in total price for a specific offer to his or her supervisor

Get:

/travelAgent/<employee_id>/offer

Displays all offers managed by this travelAgent

Customer:

Routes of Customer Namespace:

Post:

/customer/

Adds a customer to the agency

/customer/<customer_id>/expert

Requests an expert for the preferred country

/customer/<customer_id>/offer/<offer_id>

Handle an offer presented by a travelAgent, possible options are “accept”, “decline” and “change”

Get:

/customer/<customer_id>/offers

Displays all valid offers sent to this customer

Country/Activity

Routes of Country Namespace:

Post:

/country

Adds a country to the agency

/country/<country_id>/activity

Adds an activity for a certain country

/country/<country_id>/activity/<activity_id>/update

Updates name and or price of an activity for a certain country

Get:

/country/

Displays about all countries

/country/<country_id>

Displays information about a single country

/country/<country_id>/stats

Displays information about successful trips for this country

/country/<country_id>/activity/<activity_id>

Gets information about a specific activity

Delete:

/country/<country_id>/activity/<activity_id>/delete

Deletes an activity from the catalogue of a specific country