Web Engineering

API ARCHITECTURE REPORT

March 22, 2019

Udrescu Bianca
Student nr: s3136728
Jakob Vokac
Student nr: s3179222

University of Groningen
-group 2-

Contents

1 Restful Web API

In this Architecture document, we are presenting the implementation of a REST API that will allow the manipulation and visualization of the ”Airlines” database.
The database provided contains statistics information about all reported delays per carrier per airport per month in the USA from 2003 to 2016.
The API endpoints support JSON and CSV representations of the resources but have JSON set as the default.

1.1 REST

The key elements in following the RESTFUL principles and designing our API are :

Keeping it simple
Making sure we use nouns instead of verbs and using plurals and singulars accordingly.
Using proper HTTP codes
”200 OK”:This is most commonly used HTTP code to show that the operation performed is successful.
”201 CREATED”:This can be used when you use POST method to create a new resource.
”202 ACCEPTED”:This can be used to acknowledge the request sent to the server.
”204 NO CONTENT”This status response code indicates that the request has succeeded, but that the client doesn’t need to go away from its current

page
″400 BAD REQUEST″:This can be used when client side input validation fails.
″401 UNAUTHORIZED″: This can be used if the user or the system is not authorised to perform certain operation.
″404 NOT FOUND″: This can be used if you are looking for certain resource and it is not available in the system.
″5XX SERVER ERROR″:This should never be thrown explicitly but might occur if the system fails,if server received an invalid response from the upstream server or there is temporary unavailability of the action, program or file.

1.2 Python

The language we have chosen for our project is python as it provides multiple perks.
It is a high level programming language that facilitates easy and maintable coding practices and offers a large and diverse standard library.
Since having a wide range of web frameworks to choose from was an important starting step for us , we found python to be ideal in this domain as it offers multiple frameworks such as Django, Flask, Pyramid , Bottle and Cherrypy.

2 Models

In the Models chapter we will focus on discussing about our chosen framework : Django and the actual implementation of our REST API.

2.1 Django

Django is a high-level Python Web framework that encourages clean,scalable and secure architectural design without the need to use Structured Query Language.

Django officially supports the following databases: :

- PostgreSQL
- MySQL
- SQLite
- Oracle

Since Django installs and configures SQLite automatically, with no input from the user and our project is relatively small, we will be using SQLite as our database.
In Django, a model is the object that is mapped to the database.There is no need for user input. When a model is created, Django executes SQL to create a corresponding table in the database.
Django prefixes the table name with the name of the Django application. The model also links related information in the database.
Django's models provide an Object-relational Mapping (ORM) to the underlying database.

ORM is a powerful programming technique that makes working with data and relational databases much easier.

2.2 Our Models

A brief description of our main models:
Airport:
Represents airports in the US that are identified by the primary key ″code″ and the field ″name″ which is the name of the airport.
Carrier:
Represents carriers to US airports that are identified by the primary key ″code″ and the field ″name″ which is the name of the carrier.
Time:
Represents a time period that is identified by the primary key ″label″ and has the fields ″year″ and ″month″.
StatisticsGroup:
Represents statistics about flights of a carrier to/from a US airport for a given month/all months available.
A unique statistics group is defined by a combination of airport carrier and time.
Each of these models represents a foreign key for our statistics group.It also contains a one-to-one relationship with the ″Statistics″ model
Statistics:
Represents statistics about flights of a carrier to/from a US airport for a given month/all months available.
It contains the fields :″flights″,″num_del″and ″minutes_del″ and one-to-one relationships with the models ″NumDelays″,″MinutesDelayed″and ″Flights″.
NumDelays:
Represents the statistics about delays.
It contains the fields:
″late_aircraft″,″weather″,″carrier″,″security″ and ″nat_avi_sys″.
MinutesDelayed:
Represents the statistics about the minutes delayed.

It contains the fields:
"late_aircraft","weather","carrier","security","total" and "nat_avi_sys".
Flights:
Represents the number of on-time, delayed, and cancelled flights of a carrier.
It contains the fields:
"cancelled","on_time","total","delayed" and "diverted".

2.3 Migrations

We have used Migrations in order to apply the changes we make to our models (adding a field, deleting a model, etc.) into the database schema.
By using the command "$ python manage.py makemigrations" we have created the new migrations based on the changes we have made to our models whereas "$ python manage.py migrate" has successfully applied those changes.
After running these commands the migration files will be stored in a "migrations" directory inside of our app.

2.4 Views

In our views folder we have stored the functions that are responsible for taking Web requests such as GET POST PUT and returning Web responses such as showing all the statistics regarding US airports or an error accompanied by an error message etc.
We specify the the type of the response with the "Äccept" header in the HTTP request.This could be 'ápplication/json','text/csv' or none.
Later on, we map those function to the URL path expressions we have defined in our path directory
The defined url patterns allow for Django to run through them one by one and find the first one that matches the request.When this happens, the corresponding functions in the view directory get called .

If the matching procedure fails , or if an exception is raised during any point in this process, Django invokes an appropriate error-handling view.

Our Url patterns will be showed in the following code snippet:

```
urlpatterns = [
    url(r'^airports/$', views.ListAirports.as_view()),
    url(r'^carriers/$', views.ListCarriers.as_view()),
    url(r'^airports/(?P<airport_id>[\w]+)/carriers/', views
    .ListCarriersOfAirport.as_view()),
    url(r'^statistics/$', views.AllStatistics.as_view()),
    url(r'^statistics/flights/$', views.FlightsStatistics.as_view()),
    url(r'^statistics/delays/$', views.DelayStatistics.as_view()),
    url(r'^statistics/description/$', views.FancyStatistics.as_view()),
    url(r'^docs/', include_docs_urls(title='My API title'))
]
```

2.5 Further Notes

Lastly, it is important to note this report will suffer further modifications as our project is a work in progress that will be documented allong the way. For informations regarding our endpoints please reffer to the API Design document.