Rest API Design

Oliver Holder, Nik Dijkema, Tai-Ting Chen (Group 18)

February 2019

1 Introduction

1.1 Framework choice

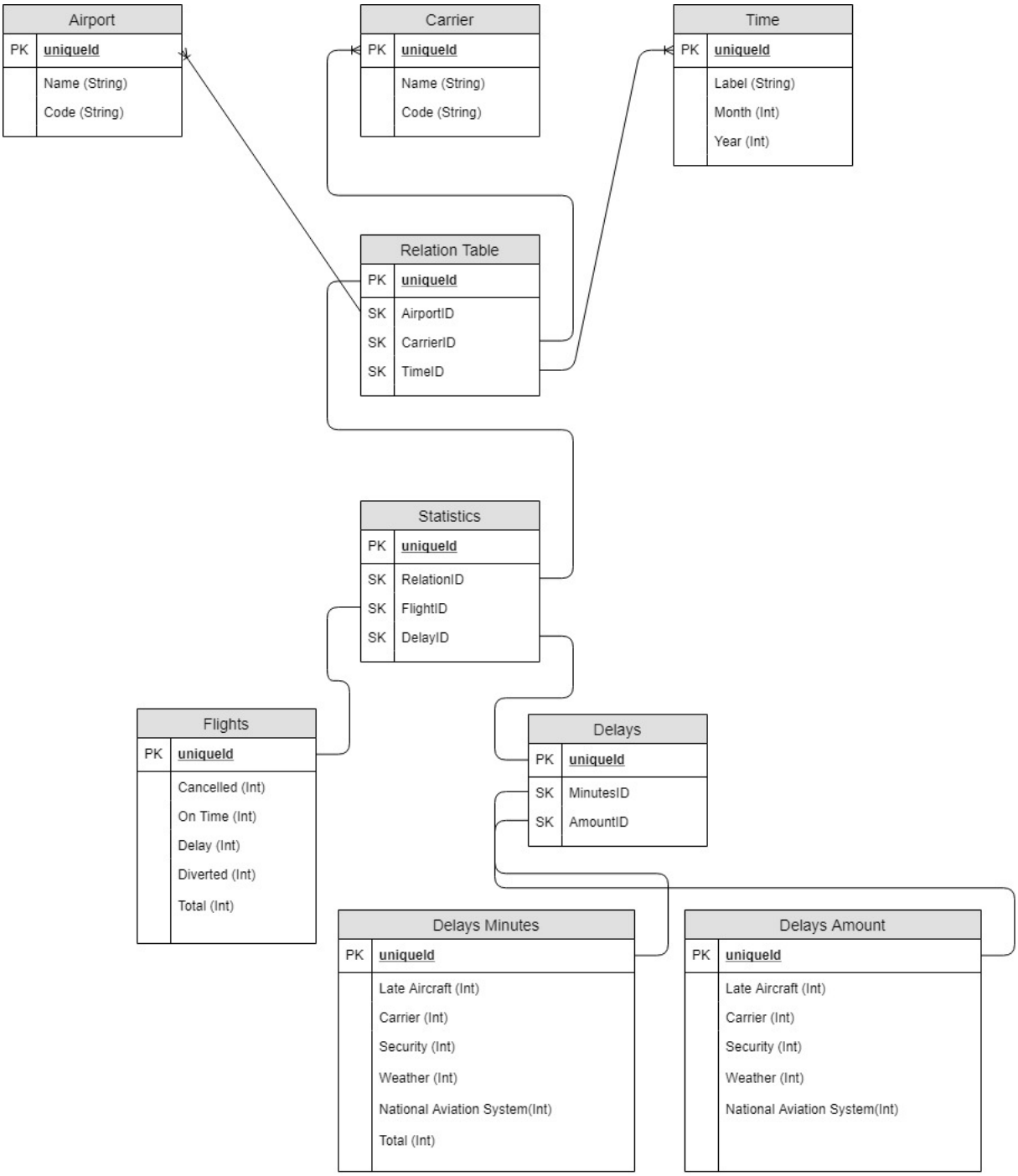
Python will as the main language of the API. The framework flask implements restful design.

1.2 Data storage options

In this project, the initial data we receive is in JSON and CSV format. Either of these formats can be directly parsed and loaded into memory as lists in python. This way of processing data works well for read only data, however, there will be PUSH requests in the API, meaning the data will be subject to change. This causes problems for data loaded in memory, saving the data as text for every PUSH request is very inefficient. Therefore SQL will be used as the main storage method. Because the API will be made in python, the SQL handling will be performed by SQLAlchamy.

2 Data model

2.1 Database structure/relations



2.2 URI/resource hierarchy

The resource and their relations must be defined before an API can be designed. The database design will be different from the URI Hierarchies used (For example, time is a query parameter whereas carriers is a hierarchical parameter. Visible Objects (with URI):

–Airport
–Carrier
–Statistics
–Delays
–Amount
–Minutes
–Flights

2.3 URIs

Entity	**URI**
airports	website/airports/< *code* >
carriers	website/carriers/< *code* >
statistics	website/carriers/< *code* >/statistics
delays(minutes)	website/carriers/< *code* >/statistics/delays/minutes
delays(amount)	website/carriers/< *code* >/statistics/delays/amount
flights	website/carriers/< *code* >/statistics/flights

3 End Points

3.1 Airports

Endpoint	/airports [1]
Type	GET
Return	List of airport URIs
Query options	content-type

Endpoint	/airports/< *airportcode* > [3]
Type	GET
return	Airport name + List of carrier URIs related to that airport
Query options	content-type

3.2 Carriers

Endpoint	/carriers [2]
Type	GET
Return	List of carrier URIs
Query options	airport, content-type

Endpoint	/carriers/< *carriercode* >
Type	GET
Return	Statistics URI + Carrier Name (string).
Query options	airport, content-type

3.3 Statistics

Endpoint	/carriers/< *carriercode* >/statistics [4]
TYPE	GET
Return	Statistics URI for given carrier returned as delay and flight resources.
Query options	month, airport, content-type

Endpoint	/carriers/< *carriercode* >/statistics [4]
TYPE	POST
Return	Success for fail indication.
Query options	airport, month, ,content-type
Form data	flights data, delay-minutes data, delay-amount data.

Endpoint	/carriers/< *carriercode* >/statistics [4]
TYPE	PUT
Return	Modified dictionary of flights, delay-minutes and delay-amount.
Query options	airport, month, content-type
Form data	flights data, delay-minutes data, delay-amount data.

Endpoint	/carriers/< *carriercode* >/statistics [4]
TYPE	DELETE
Return	Success or fail indication.
Query options	airport, month.

Endpoint	/carriers/< *carriercode* >/statistics/flights [5]
TYPE	GET
Return	Flights information list + carrier URI
Query options	month, airport, content-type

Endpoint	/carriers/< *carriercode* >/statistics/delays/minutes [6]
TYPE	GET
Return	(minute) delays information list + carrier URI
Query options	delay-type, month, airport, content-type

Endpoint	/carriers/< *carriercode* >/statistics/delays/minutes/averages [7]
Type	GET
Return	(minute) average delays information list + carrier URI
Query options	delay-type, airport1, airport2, content-type

4 GET methods summary

- /airports
- /airports/< *airportcode* >?content-type=< *type* >
- /carriers?*airport* =< *airportcode* >?content-type=< *type* >
- /carriers/< *carriercode* >?content-type=< *type* >
- /carriers/< *carriercode* >/statistics
  ?*month* =< *month* > &content-type=< *type* >

- /carriers/< *carriercode* >/statistics/flights
  ?*month* =< *month* > &content-type=< *type* >
- /carriers/< *carriercode* >/statistics/delays/minutes
  ?*delay* − *type* =< *type* > &*month* =< *month* > &content-type=< *type* >
- /carriers/< *carriercode* >/statistics/delays/minutes/averages
  ?*delay* − *type* =< *delaytype* > &*airport*1 =< *airportcode* > &*airport*2 =< *airportcode* > &content-type=< *type* >

5 PUSH methods summary

- /carriers.< *carriercode* >/statisticss?*airportcode* =< *airportcode* > &*month* =< *month* >

6 PUT methods summary

- /carriers.< *carriercode* >/statistics?s?*airportcode* =< *airportcode* > &*month* =< *month* >, *content* − *type* =< *type* >

7 DELETE methods summary

- /carriers.< *carriercode* >/statistics?*airportcode* =< *airportcode* > &*month* =< *month* >

8 Query values

Query variable	**Type/values**
month	Integer from 1-12 indicating january-december
delaytype	{carrier, weather, security, national-aviation-system(noa), total}
airportcode	String of the airport code.
content-type	application/json or text/csv

9 Error codes

The error codes used are inspired by amazons rest API design.

Code	**Description**
200	Successful request.
201	Created.
400	Bad request (parameter invalid). Will indicate which parameter is invalid and the expected type/format.
404	Page not found.
405	Method invalid.
500	Internal server error (Vague and to be avoided as much as possible).

10 Testing

Unit testing will be implemented for the testing of the API endpoints (each endpoint will have a unit test). For testing, a mock database will be created and populated.

11 References

(1) implementation of method 1 from M1 requirements.
(2) implementation of method 2 from M1 requirements.
(3) implementation of method 3 from M1 requirements.
(4) implementation of method 4 from M1 requirements.
(5) implementation of method 5 from M1 requirements.
(6) implementation of method 6 from M1 requirements.
(7) implementation of method 7 from M1 requirements.