**Web CWK2 Report:**

There are 3 main relevant files:

Views.py, models.py, urls.py

**Models.py:**

This file contains the models necessary to create a functional database with all the relations included

Text

Description automatically generated

Text

Description automatically generated

The Airport model:

This includes the airport codes, name of the airport, city and the country it is in. The relation is later defined in the flighInfo model.

The FlightInfo models:

The flight number is used to identify the flight. The departure and destination airport have a foreign key relation to the airport model meaning one flight object can be related to multiple airports. The other fields are all self-explanatory.

The Booking model:

All the fields for the booking model can be left empty in order to create an empty booking when the seat reservation is initially being associated with a seat. In the design report these fields were all considered but due to complications and time constraints the fields for passenger\_type, email, transaction\_id , seat\_class and total\_price will not be filled.

The need for passenger type, seat class are not of use as these have are indicated by the PassengerSeatInfo model

The PassengerSeatInfo model:

In order to create seats associated to a flight a foreign key relating each PassengerSeatInfo object to a flight is needed, hence the use of the flight’s fields. The booking id is also a foreign key as many PassengerSeatInfo objects can be associated to a single booking, this field is initially set to null so that the seats are not associated to any booking.

**Urls.py:**

This file contains the paths to each view of the web API.

Text

Description automatically generated

The first, is the path to the admin site.

The second, is the path to request the flights available, the input parameters are the departure, destination, date, seat class, number of adult tickets, number of child tickets, these parameters are then passed to the flights\_availbale function which we will later discuss. The return response is the flight number, seats available, departure airport, departure date, arrival date, duration of the flight, and the total price.

The third is the path to temporarily reserve seats until the payment is complete, it takes as input parameters the flight number and the seats separated by a comma.

The fourth path is to confirm the booking and set the seats to permanently reserved.

The final is a path to cancel the booking which takes the booking id as input and deletes the booking with that id. It also sets the seats associated with the booking to unreserved.

**View.py:**

The Flights\_available function takes in the departure, destination, date, seat\_class, number of adult\_ticket, number of child\_ticket.

A query is then created to filter the flights with the corresponding values and if none exist then a 404-error code is returned.

A for loop, iterating over each of the flight object values of the flights query, stores each of the corresponding attributes into variables so that the flight duration and cost can be calculated, and the variables can be sent back in the correct format. Within each flight object another for loop is executed which iterates over the passenger seats that correspond to the current flight object, it then appends the seats available to a seat data array by filtering the seats with reserved set to false. Finally, the total price is then calculated from the number of tickets and whether they are economy or business class.

All the data is then appended to the array as a dictionary in the correct format and is then returned as a json response.

The seat selection function takes in the flight number and the seat codes. The seat codes are input as a string with commas separating them, they are then split and assigned to the seat\_list variable. A query for a flight object with the corresponding flight number is executed and returns the values of the flight id. Another query for the corresponding passenger seat info with filters for the flight id and the seat codes is executed.

An empty booking is created so that the seats can be associated with them.

The seats in the query are iterated over and check if the seats are reserved, if not then the reserveation value is changed and the booking id attribute of the passenger seat object is associated with the booking just created, this is then saved to the database.

A timer then begins set for 20 minutes. To ensure functionality with multiple clients a list of timers holds every timer that is started, each timer has the current booking id set as its index so that it can be identified for the confirm\_booking function.

If the timer runs out, then the release\_seat\_cancel\_booking function is executed which takes in the passenger seat objects and the booking id. The booking with that id is then deleted, and the seats are iterated over and set to unreserved so that they can later be seen as available for other users.

The booking id is then returned as an Http response for later use.

The confirm booking function makes use of the booking id sent by the previous function. The function retrieves the relevant timer and cancels the timer which ensures that the seats stay reserved.

The cancel booking function also uses the booking id. A query is made using the booking id and the associated passenger seat info objects are returned which is then used to get the departure time of the flight linked with those seats. A check to ensure that the cancellation is not carried out within 48 hours of the flight departure; if it is before 2 days then the seats are released, and the booking is deleted. A 200 response is returned if the cancelation is successful, otherwise a 404 error is returned.