i-TRAVEL REPORT

Requirements Phase
The basic goal of the i-Travel app is to efficiently plan, organise and give suggestions regarding a user's trip, using the user's basic input data.
This report goes further in details in explaining the basic execution of third-party services and APIs, the architectural design and some other black boxes necessary for a fine understanding of the functionality of the application. We will briefly look at the layers

Data Services Layers: This layer comprise of the services that handles data persistence and retrieval. It may comprise of some relational database and relates directly with the adapter layers. The data layers used in this project include:

- *Wiki API*: this gives the wikipedia description of the destination entered by the user.
- *Amadeus API*: the Amadeus API handles references to hotel offers, airlines, check- in links and some flight offers which are also part of the data service layers.
- *Google Maps*: This service retrieves data from google map API and give the exact location direction on the map.
- *G.T.S:* This service is the API responsible for google text search

Adapter Services Layers: Various adapter service layer in this project are:
- *Places*: this service fetches data from the G.T.S service.
- *Place details*: this service fetches data from the wiki API and google maps API. It gives the description and location of any destination entered by the user.
- *Flight*: this service fetches data from Amadeus flight-offers API. It gives the list of available flights and their details at a particular time of entry.
- *Airline link*: this service relates directly to the IATA business logic service, converting the airline links to human readable names:
- *Airline name*: this service fetches data from Amadeus airlines data service layer and relates directly with the sorting business logic layer. It basically sorts the name all available airlines/flights in an alphabetical order.
- *Hotel*: it fetches data from the Amadeus hotel-offers data service layer. It presents the user with a list of hotels around the predefined destination.
- *Food*: it fetches data from the G.T.S

Business Logic Services Layers: this layer deals with manipulations, decision making, calculations and the likes. It fetches data from the data service layer, process these requests and sends results back to the user. Some of the business logic layer used in this project include:

- *IATA*: this relates directly with flights in adapter service layer which fetches data from the Amadeus flight offers. It basically converts the data collected from Amadeus flight-offers data service layer into human readable language.
- *Sorting*: this basically sorts the data collected from Amadeus airlines & check-in links in alphabetical order.

Process Centric Services Layer: The major process centric services layers present in this project include:
- *Flight Process*: this presents the user with a well defined output of all available flights, departure & arrival time, cost, airline names, and the likes. Here, the user is able to search flights, having provided useful information like: check-in & check-out dates, place of departure and destination.
- *Food Process*: here, the user is expected to choose from a list of options (restaurants, bars or/and fast foods) as well as the destination to serve as input data

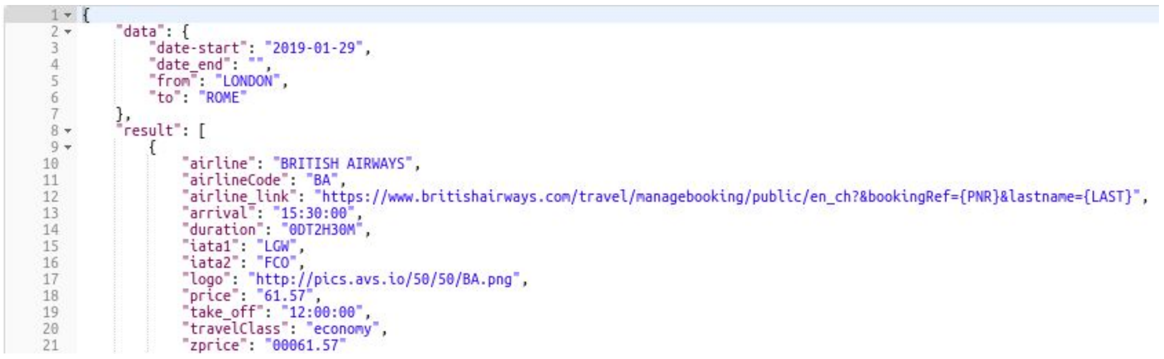
for processing. The user is then presented a list of suggested restaurants, bars and/or fast food spots, located within the vicinity of the user's destination.

- *Hotels*: this process presents the user a list of suggested hotels within the flight destination the user enters. It gives the details of the hotels (wiki API), the geo-location (google maps API), the ratings of the hotels. This process allows an authenticated user save suggestions the he/she hopes to revisit, in a bucket list. The user must have entered inputs such as: expected number of rooms, the number of adult & children, departure and arrival dates.
- *Point of Interest:* this process basically gives the user recommendations after entering a preferred destination.
- *User Authentication/Authorization*: this process saves basic user information like username, password and e-mail address. After user authentication, a user is allowed to save suggestions offered by the hotel process, unlike a guest user.

Design Phase

JSON Data Structure

An example of a JSON response for the Amadeus API:



```
{
    "data": {
        "date-start": "2019-01-29",
        "date_end": "",
        "from": "LONDON",
        "to": "ROME"
    },
    "result": [
        {
            "airline": "BRITISH AIRWAYS",
            "airlineCode": "BA",
            "airline_link": "https://www.britishairways.com/travel/managebooking/public/en_ch?&bookingRef={PNR}&lastname={LAST}",
            "arrival": "15:30:00",
            "duration": "0DT2H30M",
            "iata1": "LGW",
            "iata2": "FCO",
            "logo": "http://pics.avs.io/50/50/BA.png",
            "price": "61.57",
            "take_off": "12:00:00",
            "travelClass": "economy",
            "zprice": "00061.57"
```

The fundamental outlook of the architecture is given below:

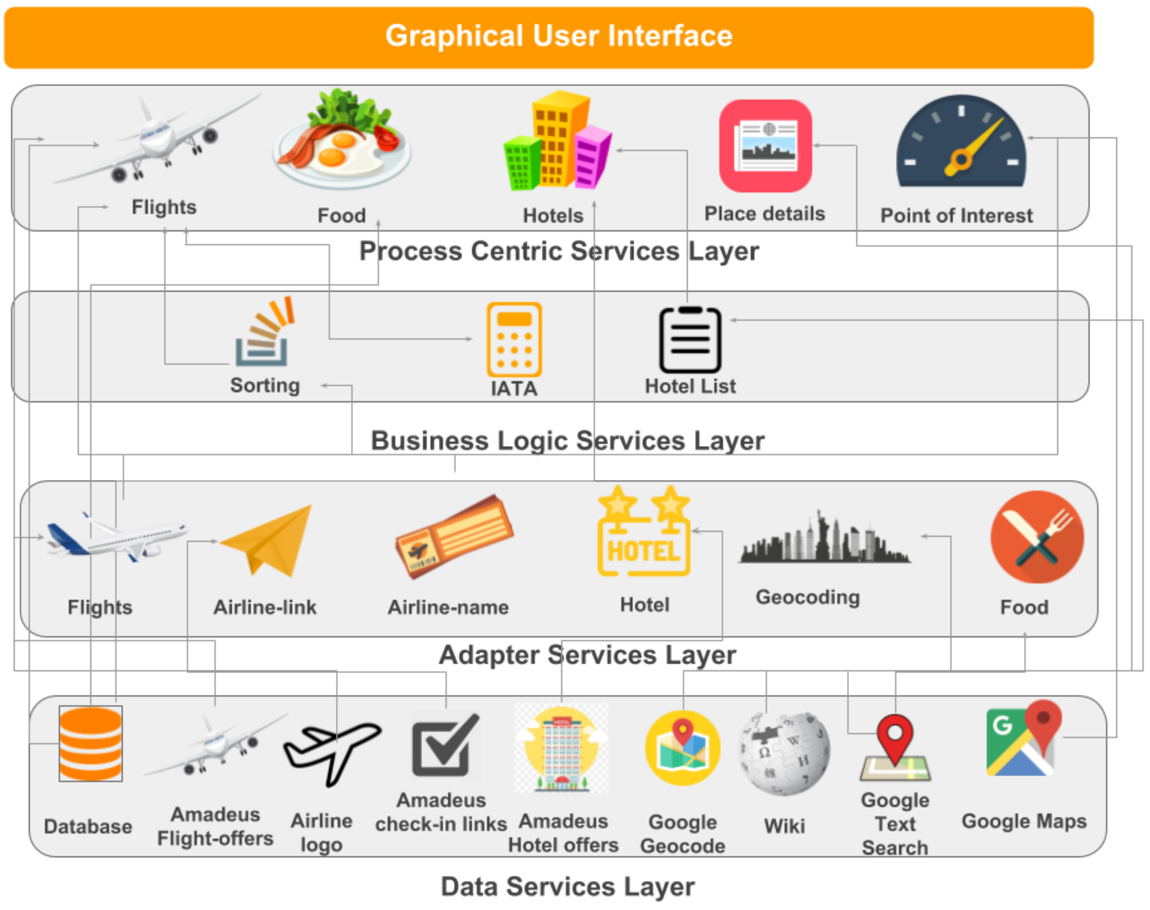


Image 2. Architecture of iTravel Application

Entity Relationship Diagram:

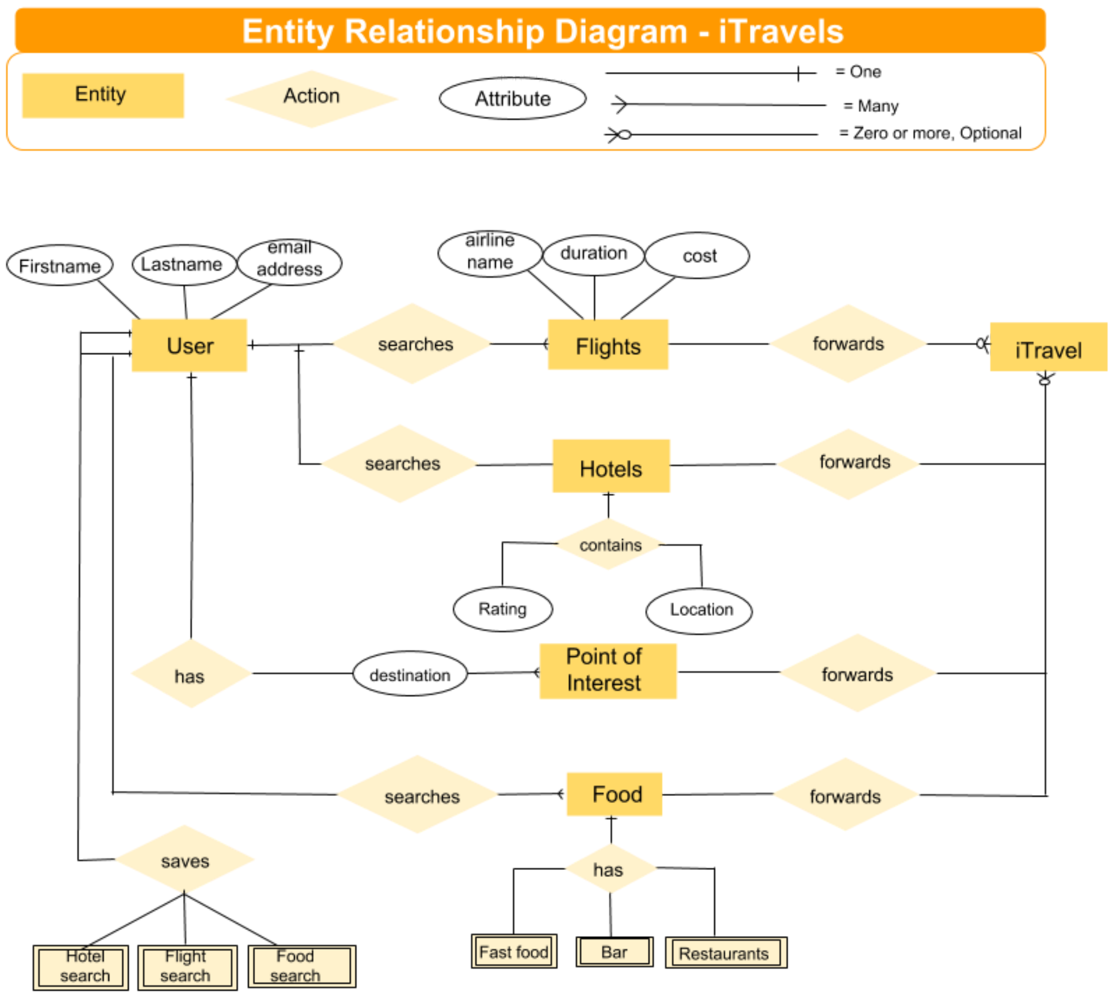


Image 1. E-R Diagram of iTravel Application

Implementation Phase

Web services: The web services used here are: *Wiki API, Amadeus API, Google Maps and G.T.S.* Their implementation is well explained in the data service layer section of this report.
Documentation: Here is the link to the published documentation of the five basic process used in this project:
https://documenter.getpostman.com/view/4627562/RztoLo8j

Database: The database is built using sqlalchemy, receiving users' basic login details. The database consist of six tables

- two tables are for storing the user's details (USERS, USER_Google)
- one table is for the registration form (REGISTRATION)
- one table is for saving login information (LOGIN_FORM)
- one table is for storing places to eat (FOODS)
- the final table is for storing the places to visit (POIS)

User Interface: the user interface for this project is a web page. it was built with basic HTML 5, CSS and Javascript.