# Travelando - Service and Design Engineering Project

Valentina Odorizzi [203314] and Michela Lorandi [207522]

September 9, 2020

## 1 Introduction

Travelando is a Telegram bot which is able to understand the human language using Dialogflow and it is used in order to organise your personal trip. It includes several internal services in order to search information related to hotels, shops, trails and emergency services and to save the parameters of a specific search or the information about a particular result. All external information is taken sending queries to the Knowledge Graph service.

## 2 Travelando structure

Travelando is written in Python using Django and is composed by an internal Postgres SQL database. Moreover, it is composed by nine internal services as you can see in the Figure 1.

- **Choreographer** receives HTTP requests from Dialogflow, which interprets the sentences written in Telegram by the user. It analyses the HTTP requests and forwards them to the correct Process Centric layer. If the HTTP request is related to retrieve information from Knowledge Graph, then the request is sent to Process Centric Search. Otherwise, if the HTTP request is related to perform some operations to the internal database, then the HTTP request is sent to Process Centric DB.
- **Process Centric DB** forwards the HTTP requests related to the internal database to the Business Logic DB choosing the correct path.
- **Process Centric Search** forwards the HTTP requests to the Business Logic Search in order to properly manage and elaborate the user request.
- **Business Logic DB** manages the requests related to save, retrieve and delete operations. More in detail, it analyses the input parameters in order to understand if the user wants to perform operations on search or result. Then, it contacts the DB Adapter in order to save a result or the DB Data in order to save a search, retrieve and delete information about one or more searches or results. For example, the user can type on telegram the following sentence *"Can you please give me the first result?"* in order to retrieve the first result in the internal DB, then the Business Logic DB understands the intention of the user contacts the DB Data in order to retrieve results information and parse them based on the user decision.

- **Business Logic Search** manages the requests related to retrieve information for a specific search. First of all, it sends a GET HTTP request to Utility service in order to retrieve the query number that is used in order to get information about a particular search. Then, it sends a GET HTTP request to Knowledge Graph DB in order to retrieve search information and elaborate them in JSON format in order to send the data correctly.

- **Utility** has the goal to understand which query execute based on the parameters received. It returns the number of the query that the Knowledge Graph Data will execute.

- **DB Adapter** contacts the Knowledge Graph Data in order to retrieve external data, such as the results of a specific search and it gets only particular fields in order to fit to the internal database. Therefore, the DB Adapter reduces the data, then send it to the Business Logic DB in order to save in the internal database.

- **DB Data** performs the operations in order to save, retrieve and delete search or result in the internal database. It has an internal database and performs the operations on it.

- **Knowledge Graph Data** retrieves the information from the external Knowledge Graph using the parameter values and the query number obtained by Business Logic layers.

Almost all the services are deployed on Heroku, while the Knowledge Graph is deployed on AWS using rdf4j.

More information about the services and APIs are available in the GitHub repositories. You can also find in all repositories some example of accepted sentences by the Telegram bot and Dialogflow.

## Useful links

- *GitHub:* https://github.com/SDEProject

- *Telegram bot:* http://t.me/TravelandoBot

- *Complete list of accepted type of sentences for search:* https://github.com/SDEProject/ProcessCentricSearch

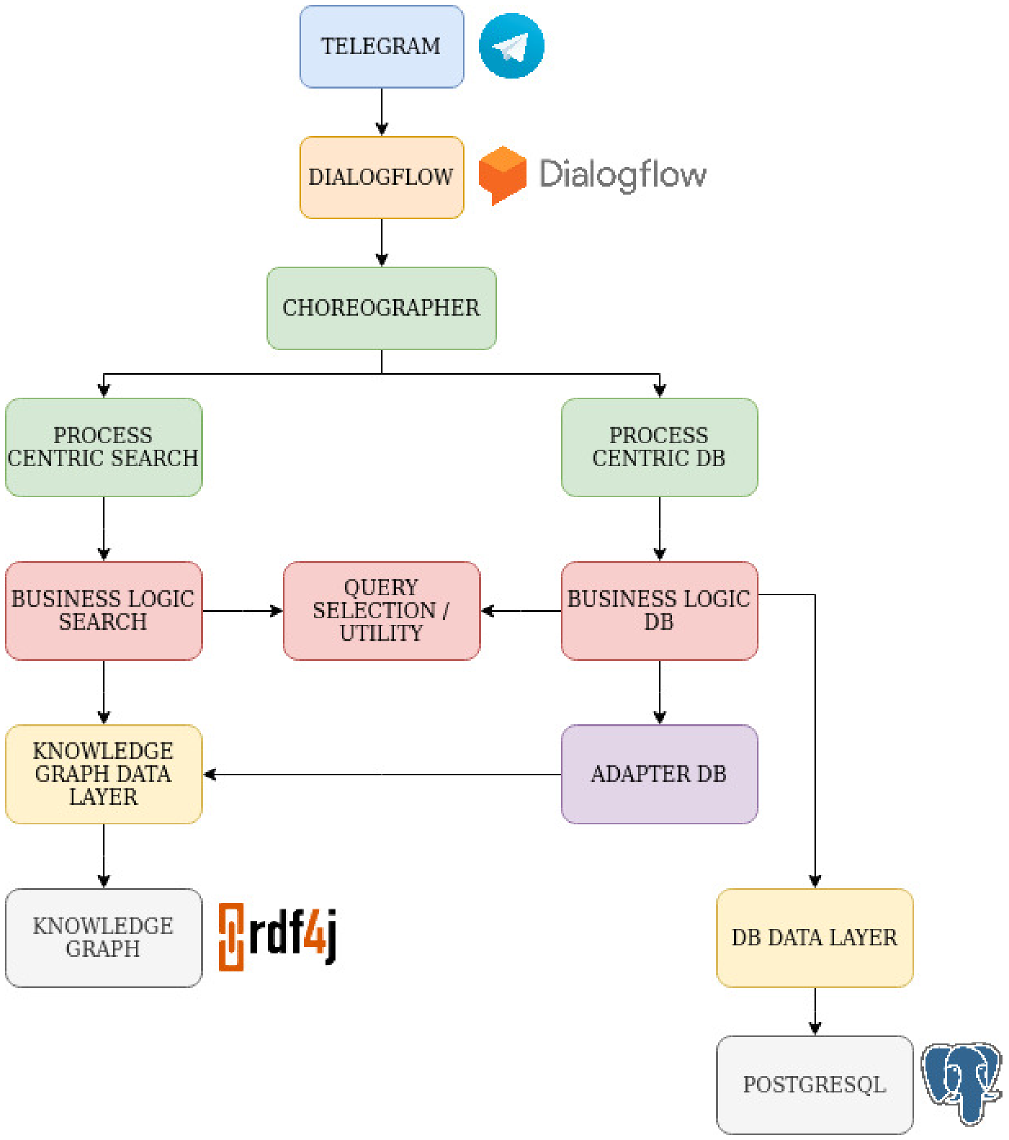

Figure 1: Travelando service structure