Travlendar+

DESIGN

DOCUMENT

Authors:

Edoardo D'Amico
Gabbolini Giovanni
Parroni Federico

1st November 2017

Indice

Chapter 1

Introduction

1.1 Purpose

1.2 Scope

Travlendar+ is an application that aims to ease the management of the daily appointments of a registered user by providing a desired schedule for those. The application runs within the context of a restricted geographic area and thus should consider only the travel means available in this scope. The application is able to interact with different data sources, in particular Travel Means APIs **FARE DEFINIZIONE** and User device GPS, and to arrange the appointments by exploiting these information. Once the schedule is computed, the application gives to the user the possibility to buy tickets for the involved travel means, then he/she can run it and follow directions provided by the application.

1.3 Definitions, Acronyms, Synonims

1.3.1 Synonims

1.3.2 Definitions

Definition 1.3.1. A device is a PC, a Tablet or a Smartphone in which run the last version of his O.S.;

Definition 1.3.2. A Schedule is a set of time-ordered and not overlapping appointments where their starting times are fixed and they're linked to each other by a path covered with a specific transportation mean;

1.3.3 Acronyms

- API: Application Programming Interface;

1.4 Revision history

1.5 Document structure

Chapter 2

Architectural Design

2.1 Overview
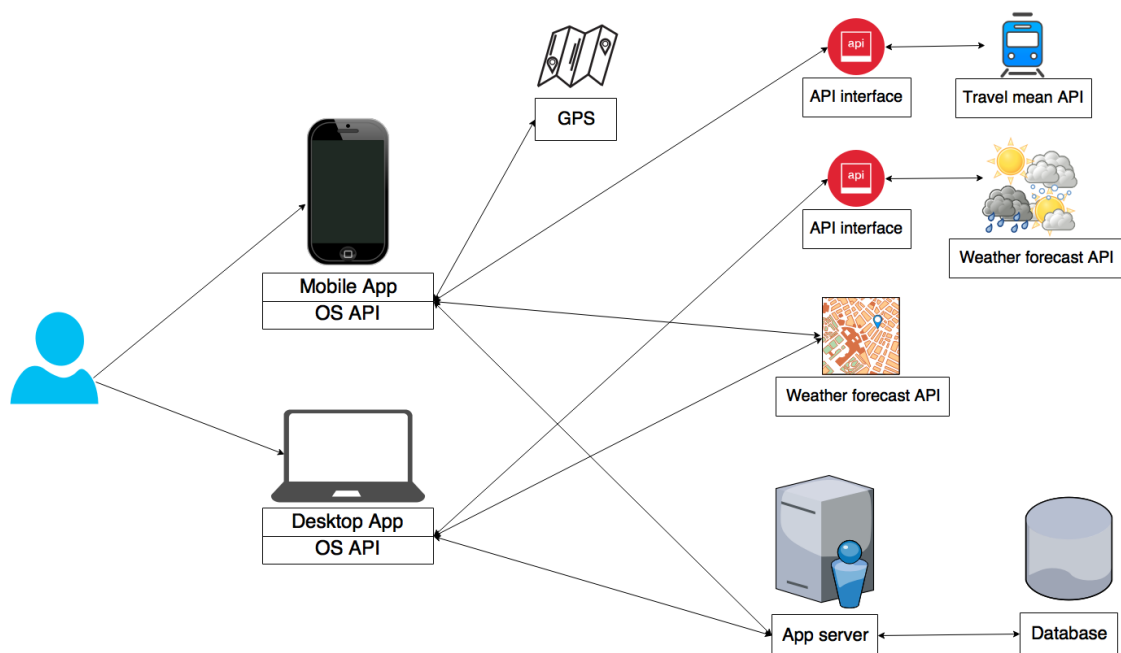
Travlendar+ has a multi-tier architecture.

Figura 2.1: High level components

User can interact directly with both version of the application: desktop and mobile. The schedule computation is performed locally on the device in which the application runs by querying the external APIs:

- Mapping Service API

- Travel Means APIs

- Wheather Forecasts APIs

For the last two ones, the communication happens through different interfaces (wrappers), in order to improve the capability of the system to be expanded. In this way we provide a common pattern for the data acquired from the APIs, so that new ones can be easily included without the need of drastic changes on the central core.

2.2 Component View

2.3 Deployment View

2.4 Runtime View

2.5 Component Interfaces

2.6 Selected architectural styles and patterns

2.7 Other Design Decisions