Politecnico di Milano

Software Engineering 2 Course

Travlendar +

di

Gianluigi Oliva, Marco Mussi e Lukasz Moskwa

November 13, 2017

Contents

Abstract

1 Introduction

1.1 Purpose

The purpose of this document is to provide technical details about the information contained in RASD of Travelandar+ Application and lead the developers that viewing this document can develop the application in the correct way.

The output of this document is an architectural description that shows all the critical feature of the problem taken into account. In particular in this document will be treated:

- The high architectural level;
- The design patterns;
- The main component and the interface that the application provides;
- The runtime behavior;
- The data structure used for the developing;

The relation among the different models is represented by using UML diagram and other useful kind of diagram that show the structure of the system (Entity – Relation diagram).

1.2 Scope

The scope of this application (Travlendar +) is provide a tool for the target users to schedule in an effective way their time optimizing the travels.

Travlendar+ allows to create a calendar which fits the meetings and other kind of commitments. The key feature of this application is to determine if the meeting location is reachable in the scheduled time and then provide a fast way to get to the destionation, otherwise it notify the user that it is not possibile by any mean to fullfill the request. Furthermore the application also arranges the schedule in a flexible way, taking into account weather condition and user's preferences as well, for some kind of events (like the lunch) and offers the possibility to slightly change its time.

With Travlendar+ is possible as well to buy public transport's tickets on the fly for the journey. If the user requires often the same path, he is suggested by the application to buy the best offered subscription.

The architecture must be designed with the intent of being maintainable and extensible. For this purpose, we will implement some known design patterns which will facilitate the development of the system.

1.3 Definitions, Acronyms, Abbreviations

Definitions

- **Platform**: system/application as a whole.
- **User**: An end user who is currently registered to the Travlendar+ application and has credentials to access.
- **Guest**: Person not registered yet and with limited access to features.
- **Event**: A scheduled meeting or other kind of appointment a user has to attend.
- **Journey**: The path chosen by the application as the one with all the fullfilled requirements.
- **Bad Weather**: A weather that prevents the user from choosing some path options. The listed bad weathers are snow, rain, storm and others.
- **Framework**: Reusable set of libraries or classes for a software system.
- **Cross-Platform**: software able to run on different platforms with same code.
- **Port**: in the internet protocol suite, it is an endpoint of communication in operating system
- **Web-Socket**:is a computer communications protocol, providing full-duplex communication channels over a single TCP connection
- **REST**: is a way of providing interoperability between computer systems on the Internet.
- **RESTful**: a system using REST
- **Client Server Architecture**: is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.
- **Client Tier**: is the tier/layer involving the client.
- **Fat Client**: is a client computer in client–server architecture or networks that typically provides rich functionality independent of the central server.

- **Layer**:In a logical multilayered architecture for an information system with an object-oriented design, the main layers are Presentation layer, Application layer and Data access layer.

Acronyms

- **RASD**: Requirements Analysis and Specification Document
- **DB**: Database
- **DBMS**: Database Management System
- **OS**: Operating System
- **HTML**: HyperText Markup Language
- **CSS**: Cascading Style Sheets
- **JS**: JavaScript
- **JSON**: JavaScript Object Notation
- **API**: Application Programming Interface
- **IDE**: Integrated Development Environment
- **RAM**: Random Access Memory
- **HTTP**: HyperText Transfer Protocol
- **HTTPS**: HyperText Transfer Protocol Secure
- **TCP**: Transmission Control Protocol
- **ACID**: Atomicity Consistency Isolation and Durability
- **DD**: Design Document
- **MVC**: Model View Component
- **UX**: User Experience
- **URL**: Uniform Resource Locator
- **OLTP**: On Line Transaction Processing

Abbreviations

- **Gn**: n-th goal
- **Rn**: n-th functional requirement
- **Dn**: n-th domain
- **Mn**: n-th mockup
- **WebApp**: WebApplication

1.4 Revision History

Version, date and summary

Version	**Date**	**Summary**
1.0.0	November 13, 2017	First release of this document

1.5 Reference Documents

We used the following documents:

1. The orginal Travlendar application:
   http://score-contest.org/2018/projects/travlendar.php
2. The revised document of the assignment:
   https://goo.gl/9m1ojy
3. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.
4. IEEE Std 1016tm-2009 Standard for Information Tecnology-System Design-Software Design Descriptions.

1.6 Document Structure

- **Introduction**: This section provides a general introduction and overview of the Design Document
- **Architectural Design**: Overall description of the main system component and relationship between them. This part is divided in various subsection in which are show the main component view, deployment view, runtime view, component interface and design patterns.

- **Algorithm Design**: Overall description of the high level details about the most critical part of the algorithms that must be implemented for the system.
- **User Interface Design**: Overview of the way the user can interact with the system and the response of the application to the user input.
- **Requirements Traceability**: Overview of how the design part descripted in this document satisfy the functional requirement and the constraints explain the RASD.
- **Implementation, Integration and Test Plan**: Identification of the order we will use to realize the subcomponents of the system and how we will integrate and test them together.
- **Effort Spent**: Time and resource effort during the development of the application.
- **References**: References and software used during the process of creation of the system.

2 Architectural Design

2.1 Overview: Highlevel components and their interaction

High level component

2.2 Component View

2.3 Deployment View

2.4 Runtime View

2.5 Other design decisions

3 Algorithm Design

4 User Interface Design

5 Requirements Traceability

6 Implementation,Integration and Test Plan

7 Effort Spent

8 References