POLITECNICO
MILANO 1863

# Design Document (DD)

Davide Rossetto 894029, Alessandro Tatti 883861

Delivery date: 2017 Nov 26

v0.4

# Contents

# Section 1

# Introduction

## 1.1 Purpose

The structure of the following document is divided into two sections:

1. Design Document (main section)
2. Integration Test Plan Document (secondary section)

The Design Document (DD) is intended to provide a more detailed functional description of the Travlendar + system to - be by providing technical details and describing the main architectural components, their interfaces, and their interactions.

The relationships among the different modules are highlighted using UML standards and other useful diagrams that show the structure of the system.

The document has to guide the software development team to implement the project architecture, providing a stable reference and a unique view of all parts of the software, defining their operation.

The second part of the document is intended to provide guidelines to adequately carry out the planning of the integration test phase.

The document includes determining which necessary tools, drivers and data structures that will be useful during the test process.

## 1.2 Scope

The system aims to support a personal event management service, providing features for choosing the best path, means of transport, and insertion of the break. The system must also make sure there are no overlap among events and among events and pauses.

The system is structured with a four-layer architecture. The purpose of the document is to describe this architecture in detail.

The system described is suitable for different types of customers: different actors that interact with the system - to be by generating a client-server dualism, so the flow of requests and responses.

The architecture must be designed with the intention of being maintainable and extensible, to make future possible changes.

This document aims to guide the implementation phase so that cohesion and decoupling are increased as much as possible. In order to do this, individual components must not include too many independent functions and reduce interdependence.

This document will follow specific architectural styles and design templates used for future implementation, as well as common design paradigms that combine useful features of this concepts.

## 1.3 Definitions, Acronyms, Abbreviations

**ACID:** Atomicity, Consistency, Isolation and Durability. This is the set of properties of database transactions.

**DD:** Design Document.

**DBMS:** DataBase Management System.

**E-R diagram:** entity relationship diagram

**ITPD:** Integration Test Plan Document.

**JPA:** Java Persistence API.

**MVC:** Model-View-Controller.

**RASD:** Requirements Analysis and Specification Document.

**UML:** Unified Modeling Language

**UX:** User Experience.

## 1.4 Revision history

**v0.1** Construct basic document's structure, add *Overview*.

**v0.2** Add *Purpose*, *Scope*, *Definitions, Acronyms, Abbreviations*, *Reference documents* and *Document structure*.

**v0.3** Add *High level components*, *Appendix* and *Bibliography*.

**v0.4** Add *Component view*.

## 1.5 Reference documents

This document follows the guidelines provided by ISO/IEC/IEEE 1016:2009 [3] related to system design and software design descriptions for complex software systems.

The indications given in this document are based on those given in the previous delivery for the project, the RASD document [1].

This document is strictly based on the RASD assignment [2] and the test plan example [4] presented during the lessons for the project of Software Engineering II, course held by Elisabetta Di Nitto and Matteo Giovanni Rossi at the Politecnico di Milano, A.Y. 2017/2018.

## 1.6 Document structure

This document consists of six sections:

**Section 1:** Introduction. This section provides a general introduction and overview of the Design Document and the covered topics that were not previously taken into account by the RASD.

**Section 2:** Architectural Design. This section shows the main system components together with subcomponents and their relationship. This section is divided into different parts whose focus is mainly on design choices, interactions, architectural styles and patterns.

**Section 3:** Algorithm Design. This section focuses on the definition of the most relevant algorithms to be implemented by the system to be.

**Section 4:** User Interface Design. It provides an overview on how the user interface will look like.

**Section 5:** Requirements Traceability. This section explains how the requirements you have defined in the RASD map to the design elements that you have defined in this document.

**Section 6:** Implementation, Integration and test plan. This section identifies the order in which the developer plan to implement the subcomponents of the system and the order in which he plans to integrate such subcomponents and test the integration.

At the end of the document are an Appendix and a Bibliography, providing additional information about the sections listed above.

# Section 2

# Architectural Design

## 2.1 Overview

This section gives a detailed overview of physical and logical infrastructures of the system-to-be and descrives the main components and their interactions.

A top-down approach is adopted for the description of the architectural design of the system:

**High level components:** A description of the high-level components and their interactions.

**Component view:** A detailed vision of the components descrived in the previous section.

**Deployment view:** A set of indications on how to deploy the components on physical tiers.

**Runtime view:** A detailed vision of the dynamic behaviour of the software.

**Component interfaces:** A description of the different interfaces.

**Architectural styles and patterns:** A set of Architectural styles, design patterns and paradigms used in the design phase.

**Other design decisions:** A list of all relevant decisions taken during the design process and not mentioned before.

## 2.2 High level components

The high-level components of the system are the following:

**Database:** The *Data Layer* of the system; All the data structures and entities concerning data storage. This level does not contain any application logic.

**Application Server:** The layer that contains all the *application logic* and *key algorithms* of the system.

**Web Server:** The Layer that provides all the *web pages* to the *web-based application.* This level does not contain any application logic.

**Mobile Application:** The *Presentation Layer* of the *mobile application*; This level does not contain any application logic.

**Web Browser:** The *Presentation Layer* of the *web-based application*; It just renders the pages obtained from the *Web server* and executes its scripts.

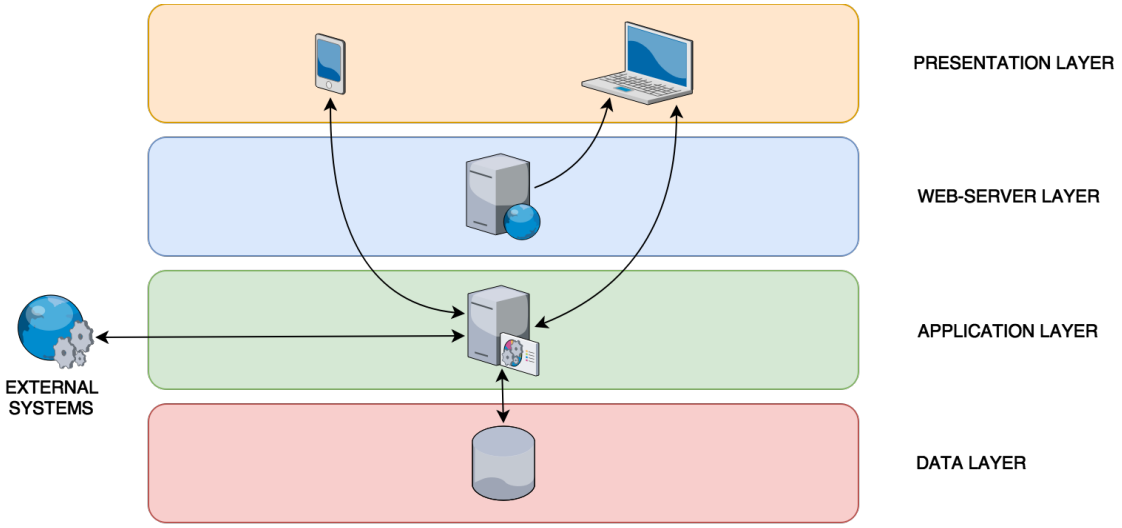


Figure 1: Layer structure of the system.

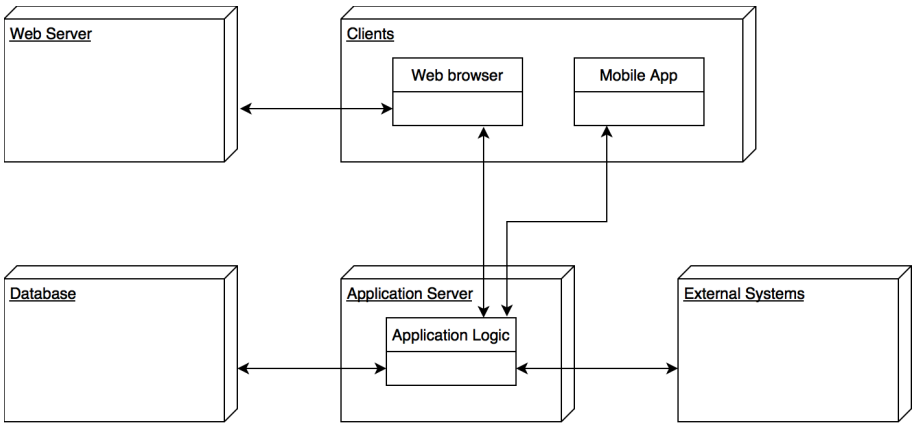


Figure 2: High-level components of the system.

## 2.3 Component view

This section shows in a more detailed way the components introduced in the overview, their role and their interactions.

### 2.3.1 Database

The database layer must include a DBMS component in order to manage the insertion, modification, deletion, and registration of data transactions within the storage memory.

The DBMS must guarantee the correct functioning of simultaneous transactions and ACID properties; the DBMS must be relational because the requirements of the application in terms of data storage do not require a more complex structure than the one provided by the relational data structure.

The data layer must be accessible only through Application Server via a dedicated interface. The Application Server must provide a persistence unit to handle the dynamic behavior of all persistent application data.

The database must be optimized during the implementation phase to ensure security by granting access to data according to the applicant's privilege level. Sensible data, for example passwords and personal information, must be encrypted correctly before being stored. Users must be granted access only on the provision of correct and valid credentials.

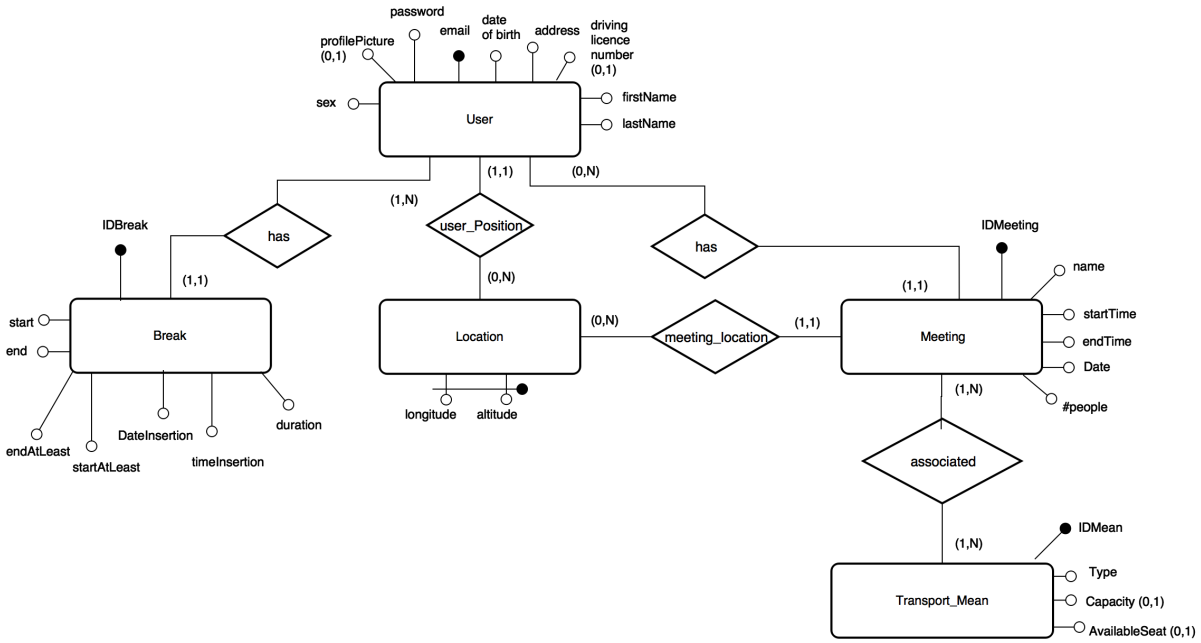


Figure 3: ER Diagram.

### 2.3.2 Application Server

### 2.3.3 Web Server

The Web Server Layer is responsible to provide the web pages to the users that intends to access the application's services via web.

This layer does not contain any application logic; it only provides static web pages that, through propers Javascript scripts, are able to access the REST APIs of the Application Server and make it fruible to end users.

The scripts must be able tu consume all the API's endpoints for the functionalities intended to be fruible via web. The comunication with the APIs must be through textual data files over HTTPS (e.g. XML or JSON), as previously specified in *Application Server* section.

### 2.3.4 Mobile Application Client

The Mobile Application Client must be designed to allow easy communication and implementation independent with the Application Server. The mobile application user interface should be designed following the guidelines provided by Android and iOS producers. The application must provide a software module that manages the GPS connection so that it can track location data by providing it to application servers.

### 2.3.5 Web Application Client

A *Web Application Client* is any modern browser (eg. IE, Firefox, Chrome, Safari) able tu run Javascript to allow the pages to consume the REST APIs of the *Application Server*.

# Section 3

# Algorithm Design

# Section 4

# User Interface Design



# Section 5

# Requirements Traceability



# Section 6

# Implementation, Integration and Test Plan

# Section A

# Appendix

## A.1 Software and tools used

The software and tools used during the drawing of this document are:

**LaTex:** used to build this document.

**Google Drive / Google Documents:** used to always update and share the documents.

**draw.io:** used to draw diagrams (https://www.draw.io).

**Marvel:** used to build the mockups (https://marvelapp.com).

**GitHub:** used as version control and to keep it shared between group members and teachers (https://www.github.com).

## A.2 Hours of work

Most of the work on this document was made in the presence of both members of the group. The approximate number of hours worked by each member of the group is as follow (including hours spent in group work):

Davide Rossetto: about ... hours
Alessandro Tatti: about ... hours

# Section B

# Bibliography


[**1**] AA 2017/2018 Software Engineering 2 - Requirements Analysis and Specification Document - Davide Rossetto, Alessandro Tatti

[**2**] AA 2017/2018 Software Engineering 2 - Project goal, schedule and rules

[**3**] IEEE Standard 1016:2009 System design - Software design descriptions

[**4**]