POLITECNICO
MILANO 1863

Implementation & Test Document

Travlendar+

Federico Betti - Tommaso Bianchi

Deliverable:	ITD
Title:	Implementation & Test Document
Authors:	Federico Betti - 899914
	Tommaso Bianchi - 894183
Version:	1.0
Date:	January 7, 2018
Download page:	Github
Copyright:	Copyright © 2018, Federico Betti - Tommaso Bianchi
	All rights reserved

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

This document is an Implementation & Test Document (ITD). Its main purpose is to describe the structure of the code of the Travlendar+ first implementation, together with the main development choices such as the programming language adopted and the frameworks used. It also describes the testing approach and the results obtained while doing it.
This document is primarily addressed to the team that will perform the acceptance testing, and to anyone will need to extend the Travlendar+ project with new functionalities.

1.2 Scope

Travlendar+ is a calendar-based service that helps users in managing the scheduling of their meetings, whether for work or personal reasons. The system will be able to automatically compute and account for travel time between meetings to make sure that the user is never late and to support the user in its travels by identifying the best mobility option according to its preferences.
Users can create meetings, and when meetings are created at locations that are unreachable in the allotted time, a warning is created. Travlendar+ should support a multitude of travel means, including walking, biking, public transportation and driving. A user should be able to provide reasonable constraints on different travel means. Additionally a user will also be able to specify flexible breaks, so that the system would then be sure to reserve enough time for them each day.
Some complementary services, such as planners or maps, already exists on the market; however they do not offer both the possibility to schedule events and to have meaningful information on how to travel between them.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Since the domain of our system remains the same, the terminology that we will use throughout this document is still the one we defined in the RASD document.

1.3.2 Acronyms

- **Implementation & Test Document (ITD)**: the present document.
- **Object Relational Mapping (ORM)**: a programming technique for converting data between incompatible type systems using object-oriented programming languages.
- **Ruby on Rails (RoR)**: a server-side web application framework written in Ruby under the MIT License.

1.4 Reference Documents

The following list contains the main documents we have taken as reference.

- Assignment: "Assignments.pdf" (can be found in our Github repository here).
- Implementation and Testing Assignment: "Implementation and testing assignment.pdf" (can be found in our Github repository here).

- RASD: "RASDv1.2.pdf" (can be found in our Github repository here).
- DD: "DDv1.0.pdf" (can be found in our Github repository here).

1.5 Document Structure

The DD is organized into 6 main sections:

1. **Introduction**: this section contains an overview on the purpose and the scope of this document, toghether with a glossary with the definitions of the main terms we will refer to in the followings.
2. **Implemented Requirements**: this section contains a table that lists which of the the requirements of the project, already presented in detail in the RASD document, have been implemented or not.
3. **Development Framework**: this section contains a deep analysis of the framework and language that we have used to develop the project and the reasons of these choices.
4. **Structure of the Code**: this section describes the general structure of the final implementation project; it lists all the folders with a brief summary of their content.
5. **Testing**: this section describes which types of tests have been made on the project.
6. **Installation Instructions**: this section contains a detailed guide to install our project on the main operative systems; please note that due to the short time the installation procedure has been tested and is guaranteed to work only on Ubuntu 16.04 Xenial, and thus we suggest to use it or another similar Linux distribution.
7. **Appendix**: this section contains some information about the work behind the drafting of this document, such as the amount of time spent by each of the authors, the tools used to produce all the material and a revision history that keeps track of all the modifications.

2 Implemented Requirements

We managed to implement most of the requirements of the Travlendar+ project. We have decided to develop them following the implementation plan we presented in the DD document and, at the end, we succeeded to reach a fully usable version of the system, passing through lot of prototypes.
All the requirements that form the foundation of the system, concerning meetings, travels, users and breaks have been fully developed in order to be able to provide the stakeholders a sufficiently complete first version of Travlendar+.
The requirements that have not been implemented are mostly nice-to-have features that do not impact the core of the system but mainly enhance the user experience and provide a more complete set of tools for managing meetings. They have not been implemented in order to allocate more time for ensuring completeness, robustness and stability of the main features.
Below we present a table that enumerates all the requirements listed in the RASD document, pointing if they have been implemented or not.

Requirements	**Implementation**
R1 Each user must provide an email that is not already present in the system	Implemented
R2 Each user must provide a nickname that is not already present in the system	Implemented
R3 Users have to be registered into the system before logging in	Implemented
R4 Users have to provide an existing email or nickname and the associated password in order to log in; in alternative they can use a supported external login system	Partially Implemented(not the External Login)
R5 A user must be logged into the system to perform any action except registering and logging in	Implemented
R6 Users cannot have meetings while their status is set to auto-decline	Not Implemented
R7 A user cannot have different default locations sharing the start time	Implemented
R8 Time travel between subsequent default locations should be less than the difference between their start time	Not Implemented
R9 Each meeting that is not an Instant Meeting has at least two participants	Not Implemented
R10 Each meeting has at least one administrator	Implemented
R11 Each meeting has a title, a date and a location	Implemented

R12 Each participant in a meeting can access shared files and the chat	Not Implemented
R13 Users participate in a meeting if and only if they accept the invitation	Implemented
R14 Users do not participate in a meeting if they decline the invitation	Implemented
R15 Users can write in the chat of a meeting if and only if they have received and accepted an invitation to it	Not Implemented
R16 For each meeting there is a warning iff the meeting is inconsistent	Implemented
R17 The system suggests you a time, according to your settings, to have a break such that no meeting overlaps with it; if no time slot is valid, a warning is generated	Implemented
R18 At least one travel mean is available in the preference list	Implemented
R19 The travel mean suggested by the system is always the first in the weighted preference list that satisfied all the constraints; if no travel mean satisfied all the constraints than the system suggests the fastest one	Implemented

Table 1: Implemented Requirements

3 Development Framework

3.1 Ruby on Rails

Ruby on Rails, or Rails, is a server-side web application framework written in Ruby under the MIT License. Rails is a model-view-controller (MVC) framework, providing default structures for a database, a web service, and web pages. It encourages and facilitates the use of web standards such as JSON or XML for data transfer, and HTML, CSS and JavaScript for display and user interfacing. In addition to MVC, Rails emphasizes the use of other well-known software engineering patterns and paradigms, including Convention over Configuration (CoC), Don't Repeat Yourself (DRY) and the active record pattern.
We have chosen to implement the Travlendar+ application using the Rails framework and the Ruby language because of their very fast development cycles that permitted us to have a first working prototype in days, and from there to rapidly build up all the functionalities that we decided to include. Another reason for choosing Rails is the big and very active community it has, that helped us to easily find answers for most of the problems that we had to face during the development. A secondary motivation is the tight integration with Heroku, a cloud Platform-as-a-service where we have setup an automatic deploy from our GitHub repository and that is currently hosting a working version of the application. It can be found at http://travlendar.it

3.2 Advantages

- **Ruby on Rails is an opinionated framework**: this means it guides you into their way of doing things and promotes the best standards and practices of web development. The central pillar of the Rails philosophy is the DRY (Don't Repeat Yourself) principle that ensures a clear separation of concerns and maintainability of your application. The framework embraces the principle of 'convention over configuration', according to which Rails defaults to a set of conventions that specify the best way of doing many things.

- **Speed of Development**: Rails' well-developed system of modules, generator scripts, and an efficient package management system allow scaffolding a complex application in just a few commands. We can achieve rapid application development thanks to the expressive and concise nature of the Ruby language, and also to dozens of open-source libraries for just about any purpose, which the Ruby community calls 'gems'. As an added bonus, Rails ships with a default ORM system (ActiveRecord), which helps developers quickly put application and data logic together and deploy a fully functional prototype.

- **Rails is an open-source web framework supported by an active community**: Rails developers are interested in the constant improvement of the code base and incorporation of new functionalities. As a result, with Rails, there is no need to reinvent the wheel in your projects. RoR's ecosystem contains many 'gems'. Almost any functionality you might need for your web project has already been created. A vibrant community that runs Rails also ensures that the framework is regularly updated, issues are fixed, and security is kept up-to-date with the best industry standards. Also, a big active community means a lot of online material for learning and fast and competent answers to all the problems that may arise during the development.

3.3 Disadvantages

- **Runtime Speed and Performance**: Rails has a fairly 'slow' runtime speed compared to other frameworks, which makes it harder to scale RoR applications. This is due to the fact that Ruby is an interpreted language, and thus by definition slower compared to the compiled ones such as

Java, and to the high number of modules managing many different layers of abstraction in a typical RoR application.

- **A lot of hard dependencies and modules included out of the box**: while this is also somewhat an advantage because it gives you access right away to a lot of useful library code, it also means that if you do not want to keep the default settings for all of them you need to change a lot of configuration code. This also means that debugging is often harder, as you have to track down where the error came from through a lot of different active modules.
- **High cost of wrong decisions in development**: wrong architecture decisions during the initial stages of your project might cost you more in Rails than in any other framework. Since prototyping with Rails is incredibly fast, an engineering team inexperienced in Rails might make unobvious mistakes that will erode your application's performance in the future. These structural deficiencies will be hard to fix because Rails is an open framework, where all components are tightly coupled and depend on each other.
- **Lacks tools for very large projects**: as it has been developed with ease of use and fast development in mind, Rails is not very well suited for building very large and complex applications.

4 Structure of the Code

Ruby on Rails is a web framework that implements the Model-View-Controller design pattern, and as a consequence the code is divided in different folders following this distinction. Moreover, we have separate locations for test code and for the logs.
Here the main folders in which you can find our code:

- **app/controllers**: contains a file for each controller, each handling web requests for different modules of the application.
- **app/models**: contains a file for each model, representing the data stored in the database and the associations between them; they also contains methods to easily access all the information they hold.
- **app/view**: contains a file for each view of the application, that is more or less for each web page. Views are written in HTML enhanced with ERB, an implementation of the eRuby (Embedded Ruby) templating system.
- **app/helpers**: contains some helper functions to mainly assist the work of controllers. Here you can find the core logic that handles the scheduling of meetings, travels and breaks.
- **app/assets**: contains all the images, stylesheets and javascript files loaded by the web pages.
- **app/jobs**: contains files specifying a handful of asynchronous jobs. They are used to perform long calculations, such as the scheduling, outside the web request thread.
- **config**: contains configuration code, that remained mostly untouched from the default. Here you can find also the routing file, defining all the URLs of the application and the controller methods that are called to handle each request.
- **db**: contains the structure of the database defined via migrations, a tool to specify incremental, reversible changes to relational database schemas. Here you can find also the seeder, a file we used to fill in the database with fake data for development and testing purposes.
- **log**: contains a file for the log of each environment (development, test and production). When running the default Rails server this will be populated with a record of all routes visited and all database queries executed.
- **test/model**: contains the code of the model tests.
- **test/system**: contains the code of the system tests.
- **doc**: contains the HTML documentation for all the models, controllers and helpers.

5 Testing

Testing is a crucial part of the development process of a system and Ruby on Rails knows it perfectly. It dedicates a whole independent folder of the system to facilitate their implementation.
In our particular case, since we made the choice of providing small prototypes, tests became precious: after a test, that certifies that a specific functionality of the project works, has been created, every subsequent modification will have to deal with it to ensure no bug have been introduced.
We decided to build two types of test: Model Tests and System Tests.

5.1 Model Tests

This type of test runs on the models of the project, that contain the description of all the data on which the system is based. For this reason having consistent data is crucial; they must be formatted properly and must respect some constraints.
The model, on a Ruby on Rails system, is directly related with the tables on the database so it must provide proper validations that has to be tested correctly. It is important to have the largest possible number of model objects tested since the beginning of the implementation process: in this way we are sure that all the components, that will create and update data, will not insert wrong and inconsistent information.
All model tests can be found in the proper folder of the project.

5.2 System Tests

System testing of software is testing conducted on a complete, integrated system to evaluate its compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic. Ruby on Rails provide out-of-the-box a great system test framework called Capybara, that enabled us to test how the system should appear and should interact with users ignoring all or most of the logic running on the server.
We have tested in this way the features that we consider to be the key ones for the interaction of the user with our system.

Login and Signup

The purpose of this test is very easy: if a new user registers in our website, then the next time he comes back he should be able to login using the same credentials. Also, users should be able to log out and be redirected back to the homepage.

Section Navigation

Our website provides users a top-page navigation bar, or header, and a bottom-page navigation bar, or footer, to give them easy access to the main sections of the Travlendar+ application. The purpose of this test is just to check that each of those sections is reachable by clicking on the proper button.

Meeting Creation

This system test contains two test cases: the former's purpose is to check that the meeting creation page works as expected and that a user is able to visualize the new meeting on his calendar afterwards; the latter's one is to check that if a user creates two overlapping meetings, a warning is correctly generated and listed in the appropriate slot in the notification page.

Default Location Creation

The purpose of this test is to ensure that users are able to create a new default location and visualize it in the appropriate slot in the settings page afterwards.

Break Creation

This system test is divided in three test cases: the first one's purpose is to check that a user can create a break and visualize it in the appropriate slot in the settings page; the second one's purpose is to check that if a user creates both a break and a meeting partially overlapping with it, the break is correctly recomputed and its start time moved to comply with the meeting's start and end time; the third one's purpose is to check that if a user creates a meeting that completely covers the entire available time slots for a break, than the break is flagged as undoable and listed in the appropriate slot in the notification page.

6 Installation Instructions

Travlendar+ can be test at travlendar.it

Install Ruby

You can find full installation instruction for the Ruby language here. Below there is a short summary for the most used operating systems.

- **Linux rvm (tested and suggested)**

  RVM (Ruby Version Manager) is a command-line tool which allows you to easily install, manage, and work with multiple ruby environments from interpreters to sets of gems. To install it open a terminal and copy-paste the following code:

```
$> sudo apt-get install libgdbm-dev libncurses5-dev automake
    libtool bison libffi-dev curl
$> gpg --keyserver hkp://keys.gnupg.net --recv-keys 409
   B6B1796C275462A1703113804BB82D39DC0E3
$> curl -sSL https://get.rvm.io | bash -s stable
$> source ~/.rvm/scripts/rvm
$> rvm install 2.5.0
$> rvm use 2.5.0 --default
$> gem install bundler
```

- **Linux (Debian or Ubuntu)**

  Open a console and type in

```
$> sudo apt-get install ruby-full
```

- **Windows**

  Download the installer from here, then run it.

- **MacOSX (Using Homebrew)**

  Open a console and type

```
$> brew install ruby
```

- **MacOSX (Installer)**

  Download the installer from here, then run it.

Install Rails

If you have installed Ruby through the installer you should already have also Rails, otherwise just open a console and type

```
$> gem install rails
```

(NOTE: this may require to manually solve some dependencies if you havent't used rvm).
It is also required to have nodejs in the environment for the Rails Assets Pipeline, and you can install it typing the following in a console (linux):

```
$> curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
$> sudo apt-get install -y nodejs
```

Install PostgreSQL

Download the installer for you operating system from here, selecting the 9.6.6 version.
After the installation is complete, open the postgreSQL console by typing in a console

```
$> sudo -u postgres psql
```

on Linux/MacOSX (if command psql is not found navigate to the installation folder of postgreSQL and you can find it under the bin/ folder) or

```
$> c:\textbackslash~path\textbackslash~to\textbackslash~psql.exe
   -U postgres;
```

on Windows; then create a user typing

```
$> create user travlendar with createdb password '0000';
```

Setup and Run the code

Clone the source code from here if you haven't already, then navigate to the Travlendar folder. Open a console and type

```
$> bundle install
```

to download all the dependencies, then

```
$> rake db:setup
```



```
$> rails server
```

to run the rails server on localhost.
In case you have problems installing the pg gem, try running

```
$> bundle config build.pg --with-pg-config=[
    path_to_postgres_installation]/bin/pg_config;
```

You should now be able to access the application on http://localhost:3000/.

7 Appendix

7.1 Effort Spent

Date	Tommaso Bianchi	Federico Betti
28/11/17	3	3
29/11/17	1	1
30/11/17	4	4
01/12/17	2	2
02/12/17	2	0
03/12/17	4	0
04/12/17	1,5	0
05/12/17	6	6
06/12/17	5,5	0
07/12/17	0	6
08/12/17	6	3,5
10/12/17	4	2
11/12/17	2,5	2,5
12/12/17	5	2,5
13/12/17	0	3,5
14/12/17	3	0
15/12/17	5,5	1,5
16/12/17	3,5	6
17/12/17	5,5	5
19/12/17	5,5	5,5
20/12/17	0	2
21/12/17	4	0
22/12/17	3	1
27/12/17	2,5	4
28/12/17	0	2
29/12/17	0	2
02/01/18	2	4
03/01/18	4	2
04/01/18	4	8,5
05/01/18	4	0
06/01/18	0	2
07/01/18	6	6

Table 2: Effort Spent

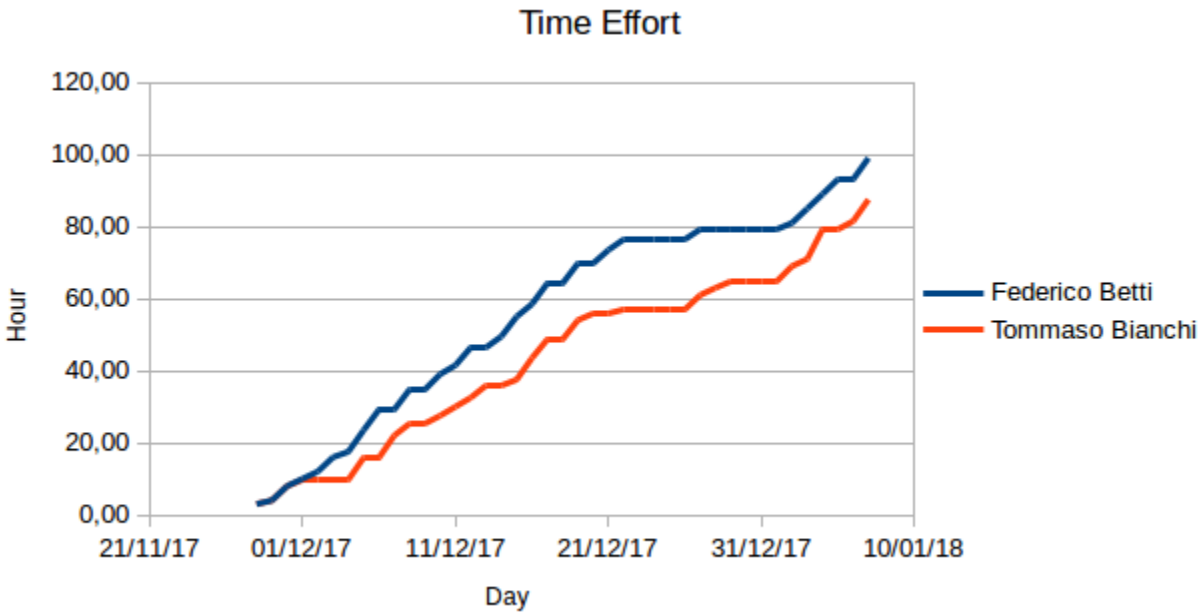

Figure 1: Effort Spent

7.2 Tools Used

- **TeXstudio**: a cross-platform open source LaTeX editor
- **Git**: a version control system for tracking changes in computer files and coordinating work on those files among multiple people

7.3 Revision History

This is the first version of the document, released on 07/01/2018. All subsequent modification will be tracked in the github repository and briefly listed here.

