

Software Engineering 2

Travlendar DD

DESIGN DOCUMENT

Version 1.0 - 27/10/2017

Plamen Pasliev – 898793

Victor Álvaro Gonzalez - 897700

Table of contents

[1. Introduction 3](#_Toc498342861)

[1.1 Objectives 3](#_Toc498342862)

[1.2 Scope 3](#_Toc498342863)

[1.3 Definitions, acronyms and abbreviations 3](#_Toc498342864)

[1.3.1 Definitions 3](#_Toc498342865)

[1.3.2 Acronyms 4](#_Toc498342866)

[1.3.3 Abbreviations 4](#_Toc498342867)

[1.4 Reference documents 4](#_Toc498342868)

[1.5 Document Structure 4](#_Toc498342869)

[2. ARCHITECTURAL DESIGN 5](#_Toc498342870)

[2.1 OVERVIEW 5](#_Toc498342871)

[2.2 High level components and their interaction 6](#_Toc498342872)

[2.3 Component view 7](#_Toc498342873)

[2.4 DEPLOYMENT VIEW 9](#_Toc498342874)

[3.Algorithm Design 10](#_Toc498342875)

[4.User Interface Design 11](#_Toc498342876)

[5.Requirements Traceability 12](#_Toc498342877)

[6.Implementation, Integration and Test plan 13](#_Toc498342878)

[7. Effort spent 14](#_Toc498342879)

[8. References 15](#_Toc498342880)

# 1. Introduction

## 1.1 Objectives

This document represents the Design Document (DD). The main goals of this document are to give more details about the development of the application and its design. This way it will be specify the architectural design, algorithm design and user interface design. In addition to this, the order in which the application will be implemented and the tested will be identified. This document is addressed to the developers who have to implement the requirements.

## 1.2 Scope

In this project we are going to develop and implement an application called Travlendar. It is a calendar-based application which allows you to create a calendar according to the events you have such as appointments related to work or personal reasons. On top of calculating the time that the user has between meetings so that he does not arrive late, the application will suggest the best mobility option between events and will alert him when it is impossible to reach a specific event on time. In addition to these functionalities, the system will allow the user to buy public transportation tickets and locating the nearest point to hire another type of service (bike of a bike sharing system, car of a car-sharing service, etc.). Users could define their transportation preferences, they can activate or deactivate any kind of transportation (including walking). The application will also take into account the weather in the location of the user. If it is raining at the time the user has to move to another event, the system will take this into account and will change the way of transport if it is necessary. The application will also allow the user to define breaks to eat or to develop other types of activities. In this way the system will organize the meetings of the user according to their breaks and the time they need to do these activities. Finally, users should also be able, if they wish to, to select combinations of transportation means that minimize carbon footprint.

## 1.3 Definitions, acronyms and abbreviations

### 1.3.1 Definitions

* *Meeting:* personal or work-related appointment. In some parts of the text, meetings are referred as appointments.
* *User*: a user of the Travlendar system
* *Calendar:* a timetable containing meetings sorted by date
* *Alert:* a notification or a pop-up screen.
* *Customer:* user that downloads the application.

### 1.3.2 Acronyms

* *RASD*: Requirement Analysis and Specification Document.
* *DD*: Design Document
* *API*: Application Programming Interface

### 1.3.3 Abbreviations

* [Gn]: n-goal.
* [Dn]: n-domain assumption.
* [Rn]: n-functional requirement.
* [Pn]: n-performance requirement.

## 1.4 Reference documents

## 1.5 Document Structure

1. In the first part of the document specify what the design document is about. In the same way is explained, without going into much detail, how the application works. Finally, is given some information about definitions and abbreviations to better understand the rest of the document.

# 2. ARCHITECTURAL DESIGN

## 2.1 OVERVIEW

In our system we only have one element, the user application that will serve as an intermediary between the user and Google servers to perform certain queries. Thus, the user will interact with the application, the application will communicate the request to the servers. The servers will return information that the system will process and display to the user through the graphic interface of the application:

1. Mobile application: represents the user who has downloaded the Travlendar application. The application allows the user to create their own calendar in such a way that they can create, edit and delete events. Within the events, the user will have to specify when creating a new one if this is a break or a meeting. Therefore, the only actors of the application will be the user and the software of the application. The user makes queries to the application through a graphical interface and the software responds with notifications, creating, deleting or editing new events, calculating how to reach the destination taking into account parameters acquired by the user's location (bad weather, strikes, distances, etc.) and user preferences. The application does not have a server or a database because the information is stored directly in the Google account of the user. For this reason, the user must have a google account which must be logged in, in such a way that the Google Calendar API can be accessed through the Google servers. This way we can avoid the purchase and maintenance of a server using this way a safe and cheap data storage such as Google. Saving the information in the user's account so that if the user changes the device they can continue using their calendar and do not have to create the events again.

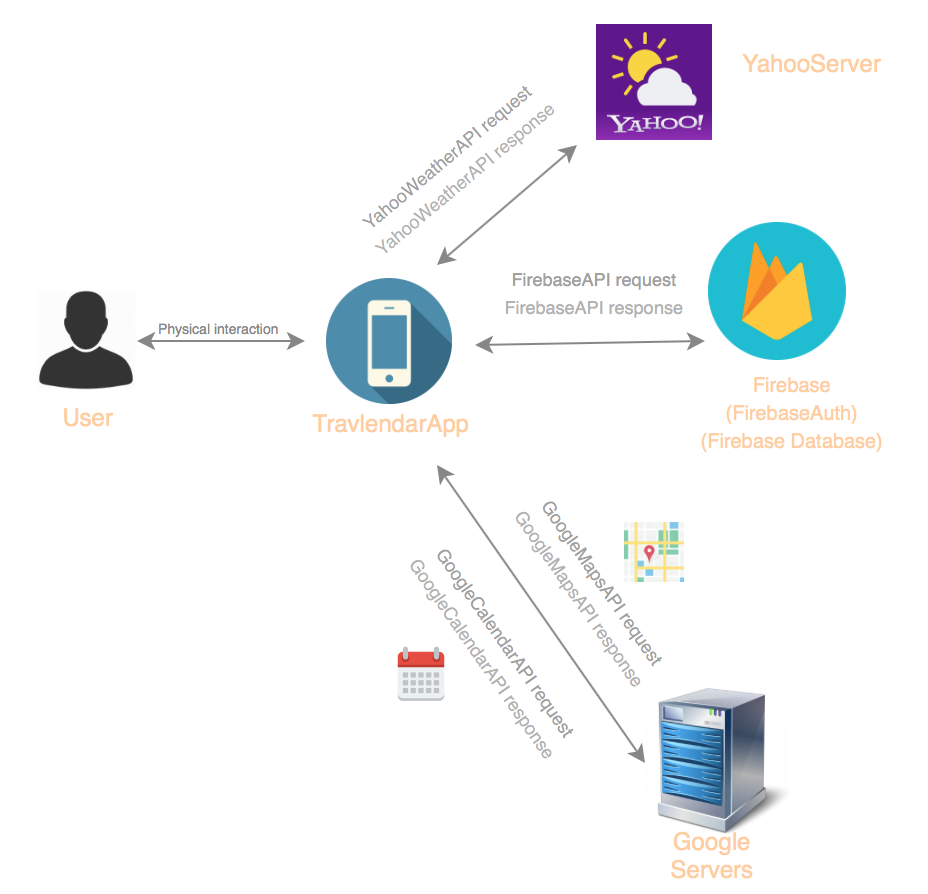
In addition, in the diagram there are also represented the external actors involved in the process of providing the service:

- Google Maps: representation of the Google Maps Web Services. In particular:

* GeoCoding API for the conversion of a string of text into a GPS position.
* GeoLocation API for the GPS localization of devices.

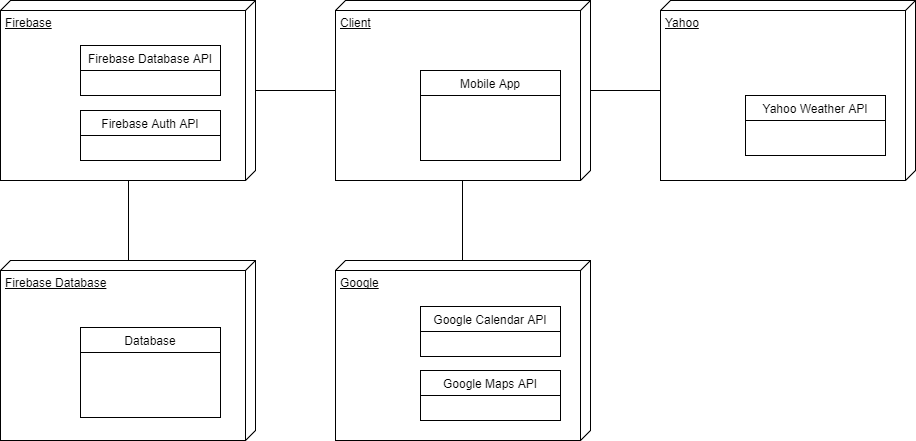
- Google Calendar API: manipulates events and other calendar data.

To store the information in the google account, the user must register with a Google account and the user's authentication with google will be done through the Firebase platform. In order to use this platform, the application must have installed Firebase SDK that will provide us with the authentication service through Google (FirebaseAuth).

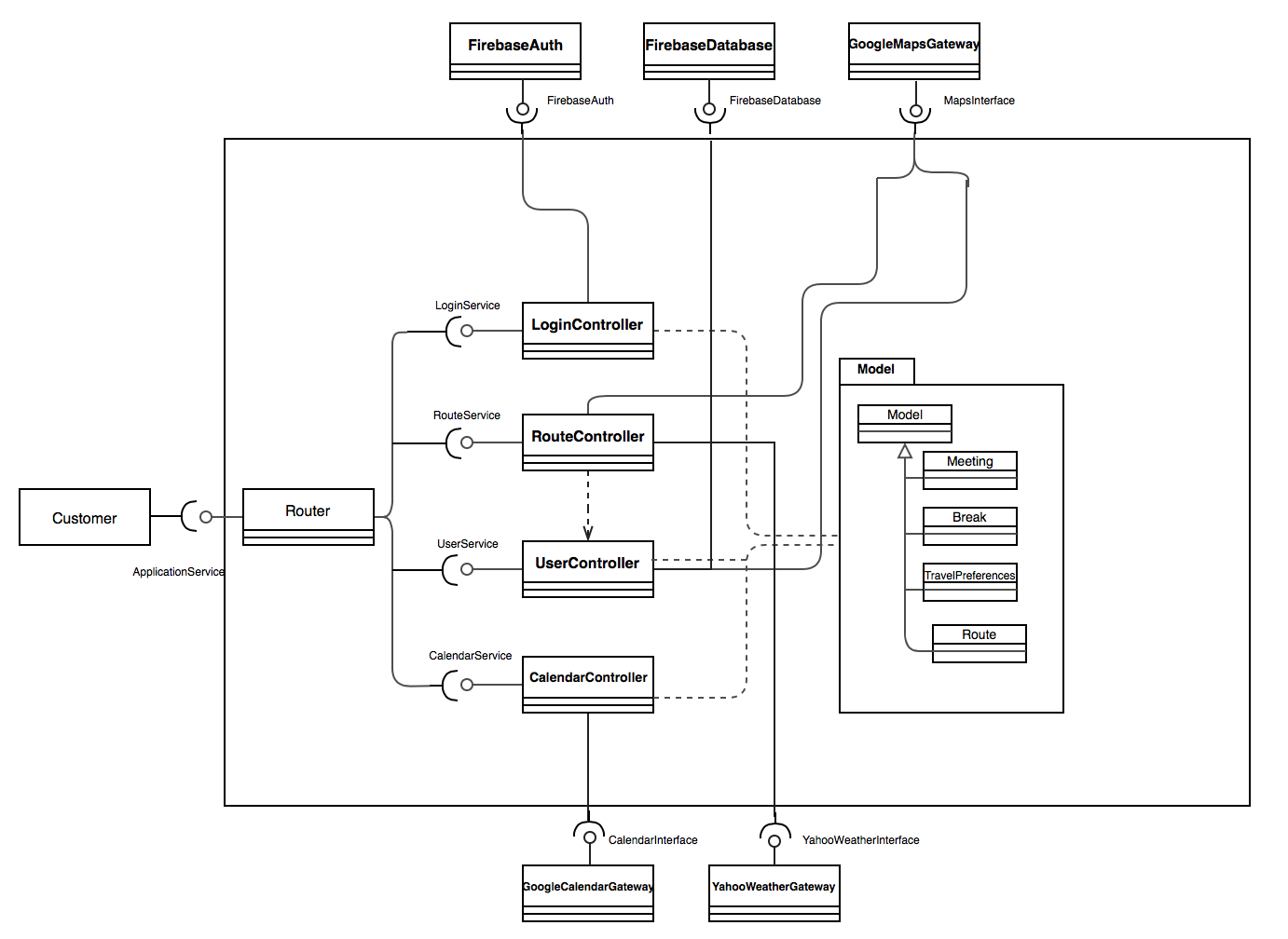


## 2.2 High level components and their interaction

The overview scheme presented in the last point naturally leads to the high-level components view. The application is just composed by three tiers, the customer mobile application, the server we use to access and compute some information (Google server) and the database to storage some information about each user as the travel preferences. All the information related to the calendar is storage in the Google account of the user.



## 2.3 Component view



- ApplicationService: interface through which the user and the system communicate. For this it is necessary that the user has activated both the geolocation and the internet on his smartphone.

- Router: routes the requests to the appropriate system component.

- Customer: the customer´s smartphone application.

- LoginController: requires the resources of FirebaseAuth to authenticate the user correctly with his Google account. Also provides the user a Googlelogin token to access the calendar resources.

- FirebaseAuth: provides the resources for user authentication.

-RouteController: requires information about the possible routes to reach the destination to the Google API and provides the router and then the user with a route to reach their destination according to the travel preferences of the user. Also the RouteController get the location of the user through the UserController.

- UserController: requires the information about the location of the user and get the information about the user travel preferences so that the system takes it into account when offering the user the routes.

- GoogleMapsGateway: Google Maps API providing the services which enable the system to locate devices and convert a string of text provided by the customer into a location.

- MapsInterface : Internet interface providing a standard HTTPS requests handler.

- Model: representation of the world and the information with which the system operates.

- CalendarController: requires the services of the Google Calendar API and provides the calendar of the user to the router. To access to the Google Calendar the user must have previously logged in.

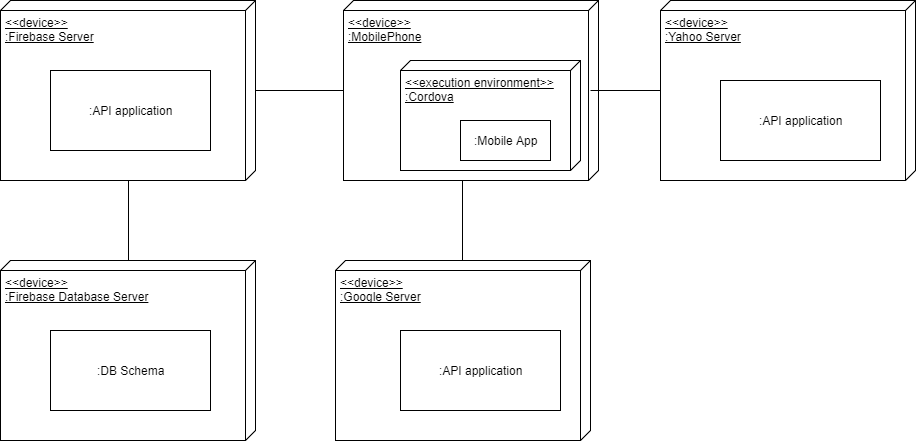
- GoogleCalendarGateway: Google Calendar API providing the services which enable the system to access and modify the private calendar of the user.

- FirebaseDatabase: provides the resources to store information about the user.

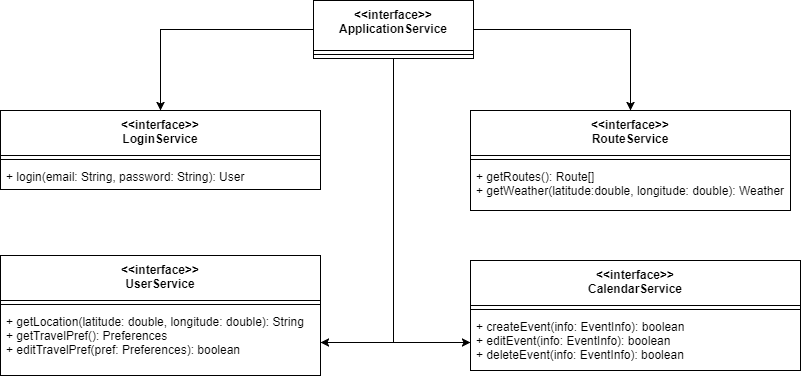
- YahooWeatherGateway: Yahoo Weather API providing services which enable the system to get access to the weather at the location of the user to inform him about what routes the system does not recommend.

The notifications that the application wants to give to the user will be carried out by the Google Calendar API.

## 2.4 DEPLOYMENT VIEW



## 2.6 COmponent Interfaces



## 2.7 Selected architectural styles and patterns

### Architecture

The Travlendar+ system is solely a mobile application. We use preexisting services provided from Google and Yahoo in order to provide functionality. The communication between the mobile app component and the external components is done by APIs.

### Design Patterns

The architecture of the mobile app itself follows the MVC (Model-View-Controller) design pattern. We are using Apache Cordova for our application which inherently uses this design pattern (HTML and CSS files handle the view; JS files handle the logic of the app, sending commands and manipulating the model).

The adapter design patter is used as well since the APIs allow the interface of the existing external services (Google maps, firebase authentication, etc.) to be used as other interfaces which we can access through our mobile app.

# 3.Algorithm Design

# 4.User Interface Design

This part of the document presents the application interface. The application is divided mainly in two views by a tap bar controller. In the main one the user activates or deactivates the means of transport they want. The second view presents the list with the events we have on the selected day. The user can change the day by browsing the calendar at the top of the view. Also in this view there is a button in the top right to add a meeting.

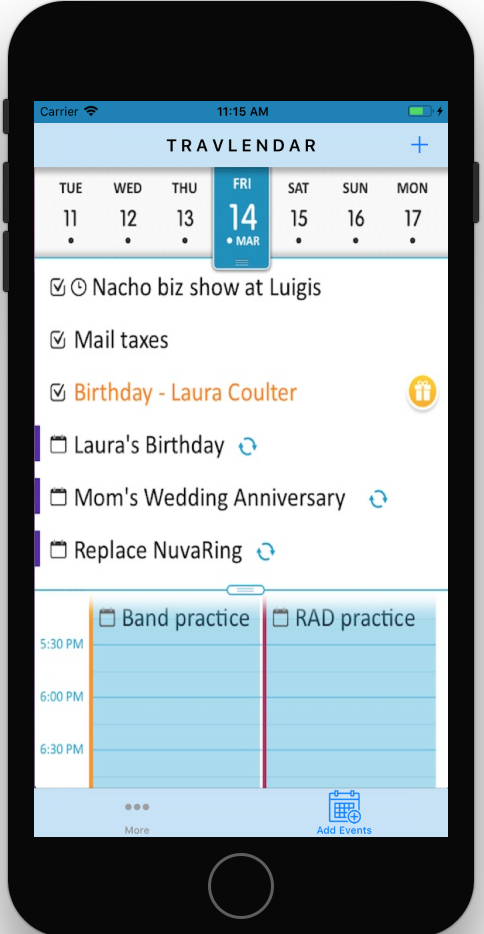


Figure 2: A list of events at a specific date

Figure 1: Main Travlendar screen

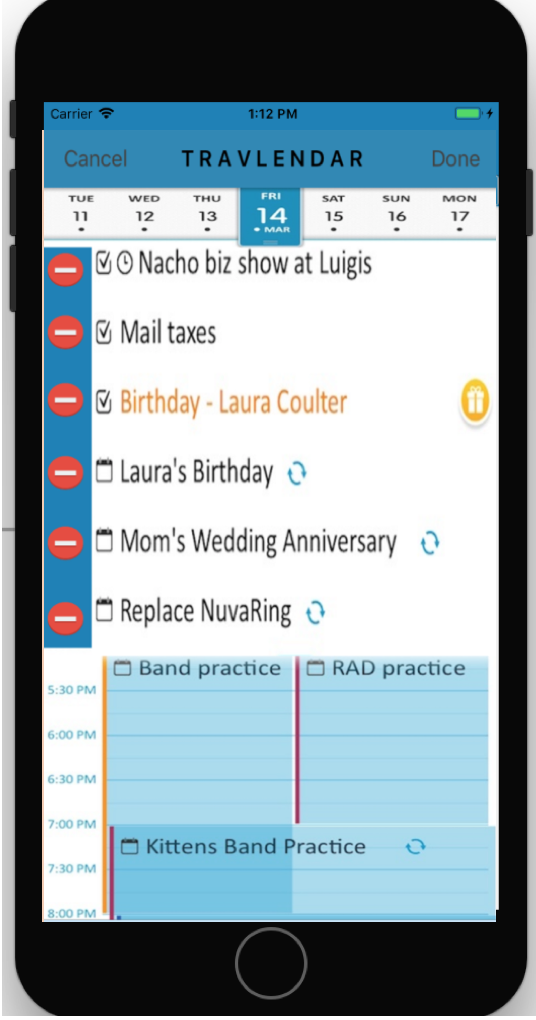


Figure 4: Deleting events

Figure 3: Adding an event

# 5.Requirements Traceability

* **[G1] Allow a User to manage appointments:**

[R1] The system must allow the user to place an event in his agenda.

* This requirement is implemented in the CalendarController component

[R2] The system must allow the user to add a title to his meeting.

* This requirement is implemented in the CalendarController component

[R3] The system must allow the user to add a start and an end location to his meeting.

* This requirement is implemented in the CalendarController component

[R4] The system must allow the user to add the start and the end time of his meeting.

* This requirement is implemented in the CalendarController component

[R5] The system must detect the current location of the mobile app user.

* This requirement is implemented in the UserController component

[R6] The system must calculate the estimated travel time to reach a meeting.

* This requirement is implemented in the RouteController component

[R7] The system must notify the user if there is an event (including travel time) overlap.

* This requirement is implemented in the CalendarController component

[R9] The system must be able to obtain different routes between the events using different means of transport in such a way that the user has several options to choose from.

* This requirement is implemented in the RouteController component
* **[G2] Allow a User to specify their own travel preferences:**

[R8] The system must allow the user to activate and deactivate preferred means of transport.

* This requirement is implemented in the UserController component

[R9] The system must be able to obtain different routes between the events using different means of transport in such a way that the user has several options to choose from.

* This requirement is implemented in the RouteController component
* **[G3] Allow a User to introduce the breaks that he requires during the day and the temporal range in which he wants to do the rest:**

[R10] The system must allow the user to place a break in his agenda.

* This requirement is implemented in the CalendarController component

[R11] The system must allow the user to add the time interval when the break should take place.

* This requirement is implemented in the CalendarController component

[R12] The system must allow the user to add the duration of his break.

* This requirement is implemented in the CalendarController component

[R7] The system must notify the user if there is an event (including travel time) overlap.

* This requirement is implemented in the CalendarController component
* **[G4] Users should receive alerts**

[R7] The system must notify the user if there is an event (including travel time) overlap.

* This requirement is implemented in the CalendarController component

[R13] The system should be able to check the weather at given location.

* This requirement is implemented in the RouteController component

[R14] The system should be able to check the availability of transport at given location.

* This requirement is implemented in the RouteController component

[R15] Users must receive an alert if the weather conditions are not pleasant when using means of transport such as walking or cycling.

* This requirement is implemented in the RouteController component

[R16] Users must receive an alert if a certain way of transport to their appointment is not available.

* This requirement is implemented in the RouteController component

[R17] Users must receive an alert 15 minutes before they have to leave for their next meeting.

* This requirement is implemented in the CalendarController component
* **[G5] The user should receive directions from his current location to the location of the event:**

[R5] The system must detect the current location of the mobile app user.

* This requirement is implemented in the UserController component

[R6] The system must calculate the estimated travel time to reach a meeting.

* This requirement is implemented in the RouteController component

[R18] The system must assist the user in his travels and display his current and target location while he is travelling towards an event.

* This requirement is implemented in the RouteController component
* **[G6] The system must consider days in which the public transport or the transport chosen by the user is not available or delayed:**

[R14] The system should be able to check the availability of transport at given location.

* This requirement is implemented in the RouteController component
* **[G7] The system must consider the weather when an event takes place**

[R13] The system should be able to check the weather at given location.

* This requirement is implemented in the RouteController component
* **[G8] The system must allow users to log in.**

[R19] The system must allow a user to log in using their Google account.

* This requirement is implemented in the LoginController component

# 6.Implementation, Integration and Test plan

Our aim is to follow the principles defined by Agile software development model. That means that during implementation we can form cross-functional teams and work on different components simultaneously.

The development team will also follow the TDD (Test-Drive-Development) practice which tells us that requirements are first turned into very specific test cases and then software is improved to pass those test cases.

After we create the Router, the plan is to start implementing the LoginController and UserController components first. After that, implementing the RouteController can be possible since it uses UserController for some of its operations. Finally, when we have everything working we can start handling events by implementing the CalendarController.

As mentioned above, testing will be done side by side with implementation and new test cases are continuously going to be added while the code evolves.

# 7. Effort spent

|  |  |  |
| --- | --- | --- |
| Name | Date | Hours spend |
| Plamen | 12/10/17 | 2 |
| Plamen | 15/10/17 | 5.5 |
| Plamen | 16/10/17 | 4 |
| Plamen | 17/10/17 | 1.5 |
| Plamen | 19/10/17 | 4 |
| Plamen | 21/10/17 | 5 |
| Plamen | 22/10/17 | 2 |
| Plamen | 24/10/17 | 6 |
| Plamen | 27/10/17 | 1 |
| Victor | 12/10/17 | 2 |
| Victor | 14/10/17 | 7 |
| Victor | 15/10/17 | 2 |
| Victor | 16/10/17 | 4 |
| Victor | 19/10/17 | 7 |
| Victor | 21/10/17 | 3 |
| Victor | 22/10/17 | 3.5 |
| Victor | 24/10/17 | 3 |

Total: Plamen: 31 hours

Victor:31.5 hours

# 8. References