Software Engineeering 2
Design Document

Tobias U. Rasmussen – 894983

November 2017

POLITECNICO
MILANO 1863

Contents

1 Introduction **1**
- 1.1 Purpose . . . 1
- 1.2 Scope and Problem Description . . . 1
- 1.3 Definitions and Abbreviations . . . 2

2 Architectural Design **3**
- 2.1 Component view . . . 3
- 2.2 Runtime View . . . 6
  - 2.2.1 Create New/edit an Appointment . . . 7
- 2.3 Component Interfaces . . . 9
  - 2.3.1 First release . . . 9
  - 2.3.2 Potential full system . . . 12
- 2.4 High Level Components and Interactions . . . 13
  - 2.4.1 Description . . . 13
- 2.5 Deployment View . . . 14
- 2.6 Selected Architectural Styles and Patterns . . . 17

3 User Interface Design **18**

4 Requirements Traceability **21**
- 4.1 Travlendar$^+$ User(Logged off) / New User . . . 21
- 4.2 Travlendar$^+$ User(logged in) . . . 21

5 Work Log **22**

6 Tools Used **23**

7 References **23**

1 Introduction

1.1 Purpose

Our team will project a suggested realization of the **Travlendar**$^+$. More specifically the **Travlendar**$^+$ will be an combined web and phone-application. The application are going to work as a combined calendar and travel-companion. If a user makes an appointment in the calendar, they will also be able to specify the exact position of said appointment. The system will then calculate the optimal route to the desired meeting point to ensure that the user get to the appointment in due time.
This document is document is meant to:

- Give a description on software design and architecture.
- Describe more in detail the algorithmical parts of the software.
- Give a general insight into how the user will experience the product.

This document is meant to be a tool for the developers to use in a production phase. With the intent that the system becomes consistent and by design.

1.2 Scope and Problem Description

In today's society many appointments are at various locations across the city or region where we live. This projects goal is to create a combined calendar and travel-companion. The application is meant to work as a normal calendar with some additional features. The application will ask for a position for each appointment, and given this position it will calculate the travel-route from your position to the next one. In addition it will calculate the route between your current/next appointment and the upcoming appointments. If a appointment is unreachable the application will notify the user. The user can specify certain travel-constraints. For example: only public transport or maybe only car. etc.

1.3 Definitions and Abbreviations

Definitions:

Software or System - The collection/part of programs which this document describe and together constitute the application itself.

Appointment - A scheduled meeting with one or more persons.

User - Any person that uses the application.

Actor - Any user or program interacting with the application.

Unreachable - The user cant reach the appointment in time.

User-Constraints - Constraints in terms of transport given by the user.

Abbreviations:

API - Application Programming Interface, a set of subroutines, protocols, and tools for building application software.

JSON - Json encoding, a tool to serialize data-structures.

HTTP - HTTP-protocol, a network protocol based on TCP.

ETA - Estimated Time to Arrival.

PW - Password,

App - Software application for any mobile device or web-browser.

GPS - Global Positioning System, a world wide service to provide accurate position data.

WIFI - Wireless local area network, here implicit used as internet connection.

POI - Point of interest. i.e a restaurant, famous church, city center etc.

This is just a small list over immediate improvements however the possibilities are wider than we as a team can determine in this here document.

2 Architectural Design

Here we will give a general description of the system. More specific a general introduction to Travlendar$^+$'s design and it's components. We will go through the components in order. Starting with the most general and going more into detail as we proceed.

2.1 Component view

In this section we will describe the low level components and their purpose. This is a introduction to each component individually. The basic idea is: To have a software in a the form of an application, which communicate directly with external API's. Making the need for an dedicated server for the application unnecessary. The reason is to make it cheap and simple. However in the future, the structure of the system may be renewed and a server added, taking over some of the tasks which the Presentation layer does in this scope.

Google Maps API Is a service provided by google in the form of several API's. It contains a lot of features which is essential when making a location based application. Some features: Interactive and static maps, estimated travel times, directions (tram, train, car etc.), live traffic-updates and location detection just to mention a few. All of these are useful to our system.

Google Calendar API Its an API which makes you able to access the google calendar, display events and link activities to certain appointments in your calendar. This requires log-in through google, but can also be used by non registered users. In our case this is benefitial since we will use google calendar as our database and calendar representation. It also has the posibility to invite other users to events, and appointments. This is also useful in the future if such a feature was to be implemented in Travlendar$^+$.

Travlendar$^+$ Settings Manager Manages all settings set by the user. This is more important in later versions when users may set permanent traveling constraints, and not have to set every travel constraint for every appointment(current solution).

Travlendar$^+$ Travel Manager This is the part of the algorithm that takes care of route making and setting routes. It must first communicate with Maps-API, before the appointment manager can set the appoinment time, and store it in the calendar database.

Travlendar$^+$ Appointment Manager This part only takes care of appoinments and communication with the calendar-API, this part must work alongside with the Travel manager to set right time for notifying the user before an agreement.

Travlendar$^+$ Presentation/GUI This is a pure presentation layer, just to show the GUI of the system and let the user interact with it. This is the component that lets the user interract with the accessable features.

Travlendar$^+$ communication manager This is the main communicative component, this communicates with all external databases/servers. This is purely a HTTP based network module with some extra core functionalities. The idea is also to have some kind of security implemented in this module, making this a barrier into the system. This component will remain in the application for the future version.

We will propose 2 systems, one for a future full release and another (test-system) meant to check if there are a market for our product. The last mentioned is the one we will see in detail in this document. However since both systems are buildt on the same principle, we will present this as well(in both deployment view and component view). The thought is not to use much time and resources for developing something that may or may not be a sustainable software.

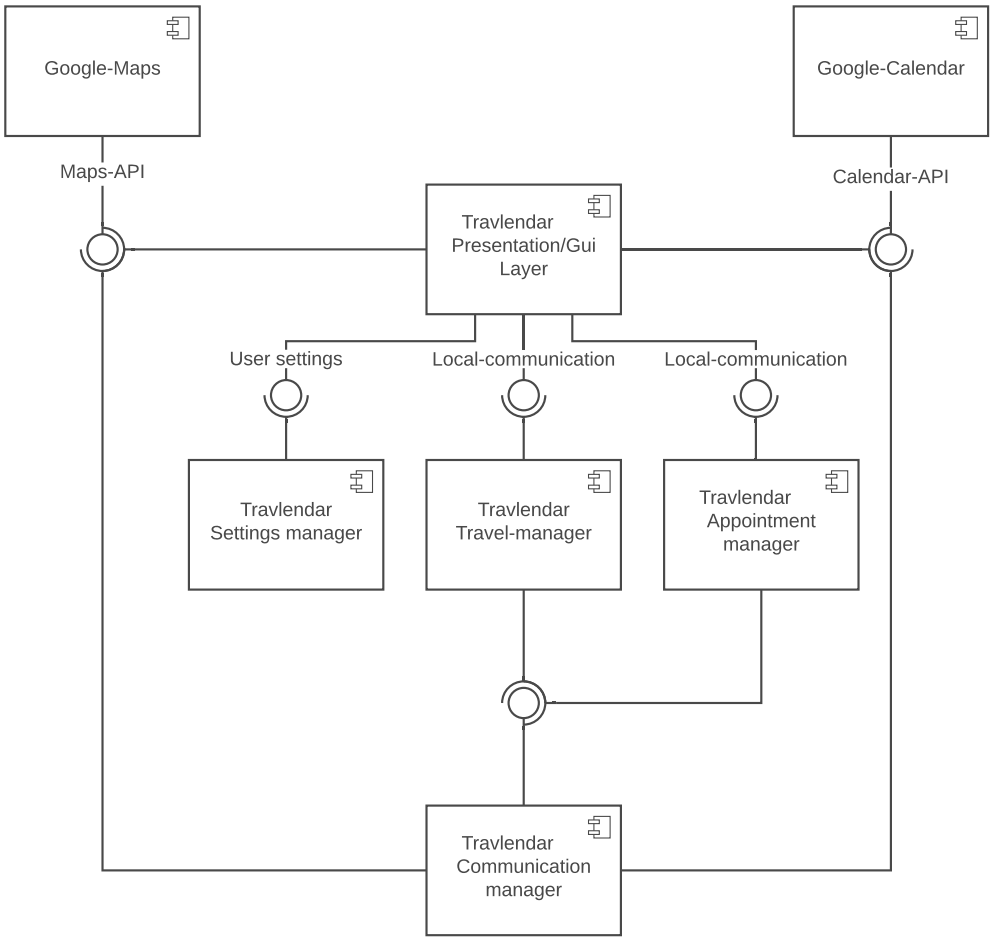

Figure 1: System Component Diagram

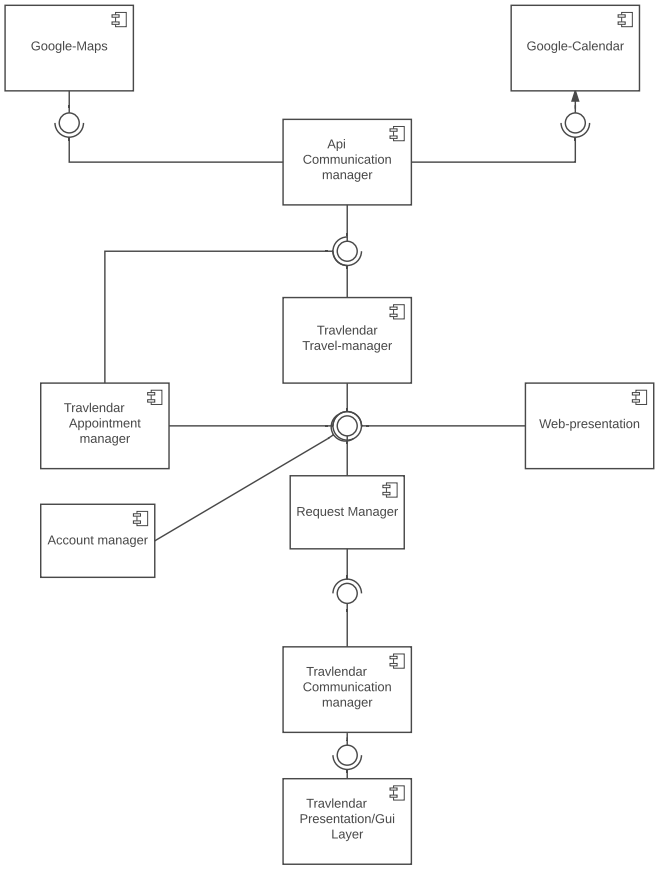

Figure 2: System Component Diagram Full system view

2.2 Runtime View

We will only see the runtime for the test-system:

2.2.1 Create New/edit an Appointment

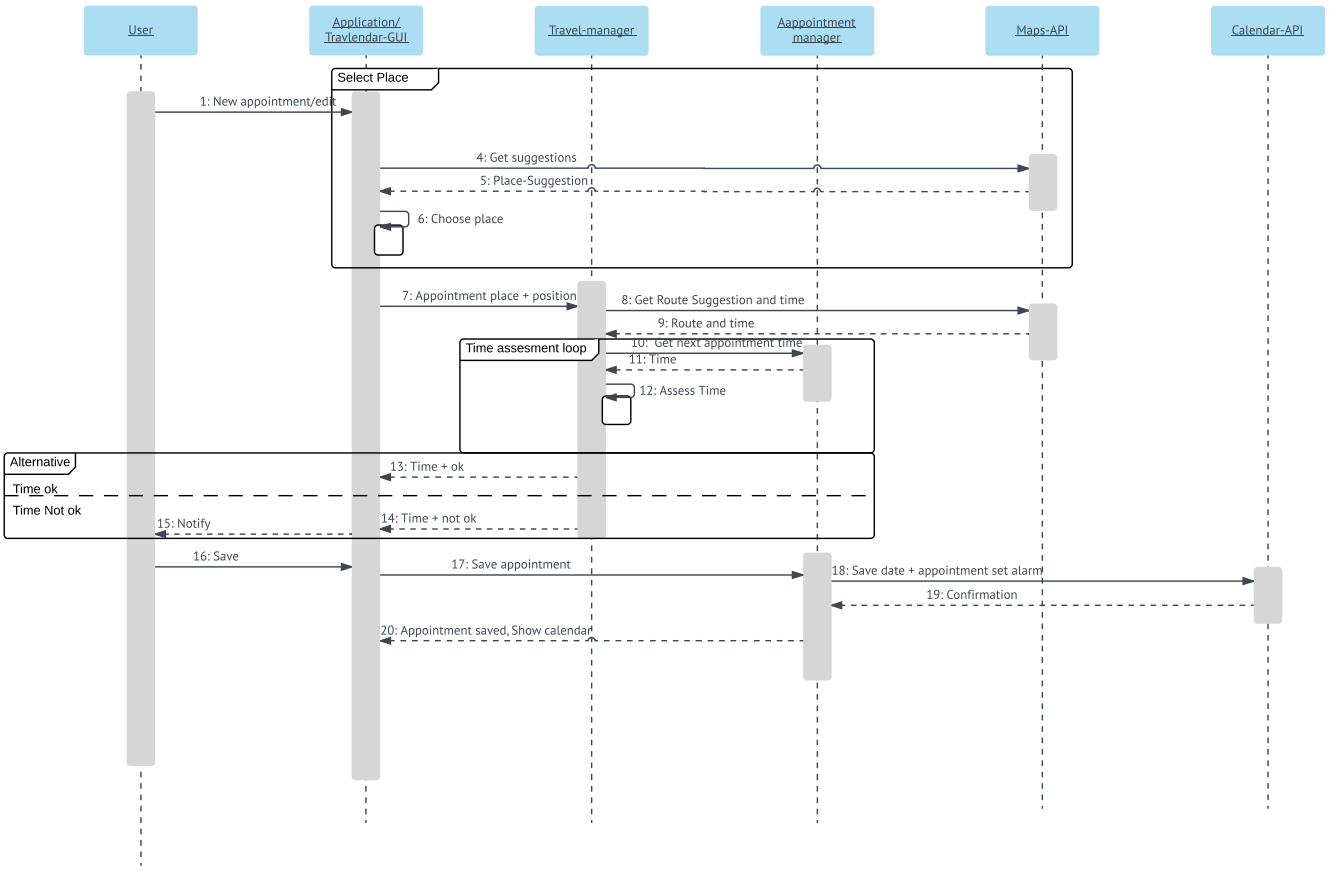

Figure 3: New appointment run-time

This sequence diagram (figure 3), shows how the system works when a user is making an appointment. In short it works this way:

- The GUI-layer communicate with maps to choose place and gets suggestions
- The travel manager uses the chosen place and current position to make the travel. And also fetch Map graphics from google.
- It then assert that the time is feasible by asking the appointment manager.
- Lastly the appointment is saved and alarm set by the appointment manager. It stores the appointment by using the calendar API.

Note that the communication manager is nowhere to bee seen on this diagram, that's because the component basically only is the systems linc to the web. It only logs faulty messages and modes, and raise error if either things take to long, or error occurs. This is to make an solid component(embedded style) between our application and the internet. By embedded-style we mean: simple, easy, does one thing(but does it great) and robust.

Calendar and notifications The calendar is integrated through googles api. The specific way to do this is to use google calendars repository called: calendar provider. This makes an easy and smooth implementation of the calendar in our systen. However, we still do have to notify the user in certain situations. This is easily solved by using google's own notification system which comes with the calendar provider. Using the repository we can engage several types of notifications: pop-up(local), email and sms. This can easily be turned on and off, this is the sort of options that will be available through the Settings-manager.
But this won't not be enough, because the Travclendar GUI, must also be able to raise local notifications, this is solved by using something called alarm class in android. This is a class which enables us to raise local notifications for an application even though it is not running. This only needs a time and date. This should then be embedded in the GUI component.

2.3 Component Interfaces

2.3.1 First release

In this section we will uncover the basic interfaces for the following components:

Google Maps API This component provides the following functionality:

- Graphical presentation of map and route
- ETA for travels
- Evaluation of traffic(live)
- Suggestions for places/adresses, POI's

It is mainly used by the travel manager to calculate travel time. But it is also used by the GUI to provide a graphical presentation of the route.

Google Calendar API This component provides the following functionality:

- Ability to create and store events
- graphical presentation of a calendar
- Notification before an appointment
-

This is used by the appointment manager to create, edit and store appointments. But also by the GUI layer to provide a graphical presentation of the calendar and to raise notifications before appointments.

Travlendar$^+$ Settings Manager This component provides the following functionality:

- Saving personal settings (travel-constraints)
- Change notification settings(sms,email and/or local).

Travlendar$^+$ Travel Manager Provides following functionality:

- Get suggested fastest route
- Get ETA
- Evaluate appointment-locations with each other.
- Notify if one or more appointments are unreachable.

To realize this component we may have these interfaces:

- **Calculate Route** - Returns the calculated route between 2 coordinates, given travel constraints
- **Calculate ETA** - Returns the calculated ETA between 2 coordinates, given travel constraints.
- **Get graphics** - Can be used to get any sort of graphical presentation of the map, route etc.

Travlendar$^+$ Appointment Manager This module manages everything that has anything to do with appointments:

- Save appointments
- Provide notification/alarm
- Edit appointments
- Delete appointments
- Check appointments

These interfaces is used by the Presentation layer and the travel-manager(to check appointment-feasibility).
The component may have these interfaces:

- **New appointment** - makes a new appoinment and saves it.
- **edit appoinment** - Edit an already existing appointment.
- **Delete appoinment** - Deletes an apointment.
- **Get Graphics** - used to get the graphical representation of the calendar.
- **Check appointments** - checks all apointments for the next 24h if they are reachable. Uses the Get Eta interface for travel-manager.

Travlendar$^+$ Presentation/GUI The part of the system which ties all together. This component should:

- Provide graphical presentation of the whole system
- Take input from user
- Prompt info to user(from calendar and travel).

This interfaces is used by travel-manager and the appointment-manager.

Travlendar$^+$ communication manager The sole purpose of this component is communication with the external services needed for the application.

This means that the interfaces is very general and simple. Both the Appointment manager and the travel manager uses this component to communicate with the Google servers.

- Communication interface with Maps-servers
- Communication interface with Calendar-servers
- Communication Log

Used by travel-manager, appointment-manager and presentation GUI.

2.3.2 Potential full system

Travlendar$^+$ Request Manager Takes care of any request posted to the server. Using http with JSON packages. Both the web-browser(web application) and the mobile-application communicate through this module.

Travlendar$^+$ API-communication manager Takes care of the communication between the travlendar-server and google's servers. This to use the API provided by google. This is used by the appointment-manager and the travel-manager

Travlendar$^+$ Account Manager Takes care of Travlendar$^+$-accounts, interacts with a database which contains all account information. This component gives the web-browser or mobile application access to use the service. Will have these following interfaces:

- Log-in
- Register
- Delete User(used by developer/admin).
- Password Recover

Travlendar$^+$ Web-presentation Is the graphical web-page/web-application which can be accessed through any browser.

2.4 High Level Components and Interactions

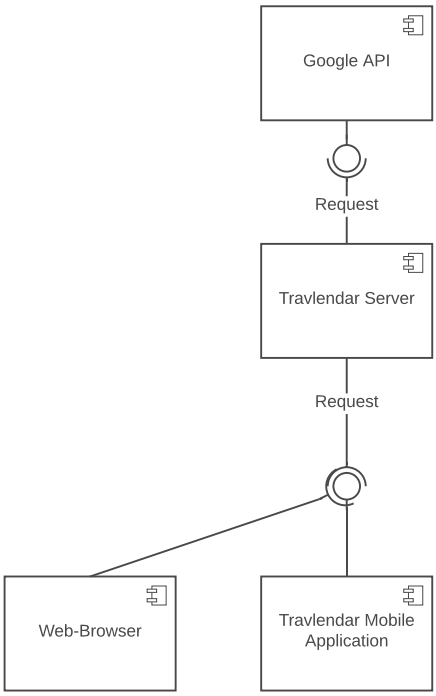

Figure 4: High level component diagram

2.4.1 Description

Google API This is the API provided by Google, this includes both the maps and calendar API.

Travlendar Server The Travlendar server is the server to which both the application and the web-browser send their requests to. Everything from log-in to downloading the maps. This is the core part of the system, where all the work is done, calculations fetching account info, changing appointments, everything goes through the server.

Travlendar Mobile application The application for mobile phones which will be available for everyone at either the play-store or the app-store.

Web-Browser This is basically just the browser from which anyone can use the service. Any web-browser should work.

2.5 Deployment View

Here we will provide a full overview of all hardware and software working on said hardware.

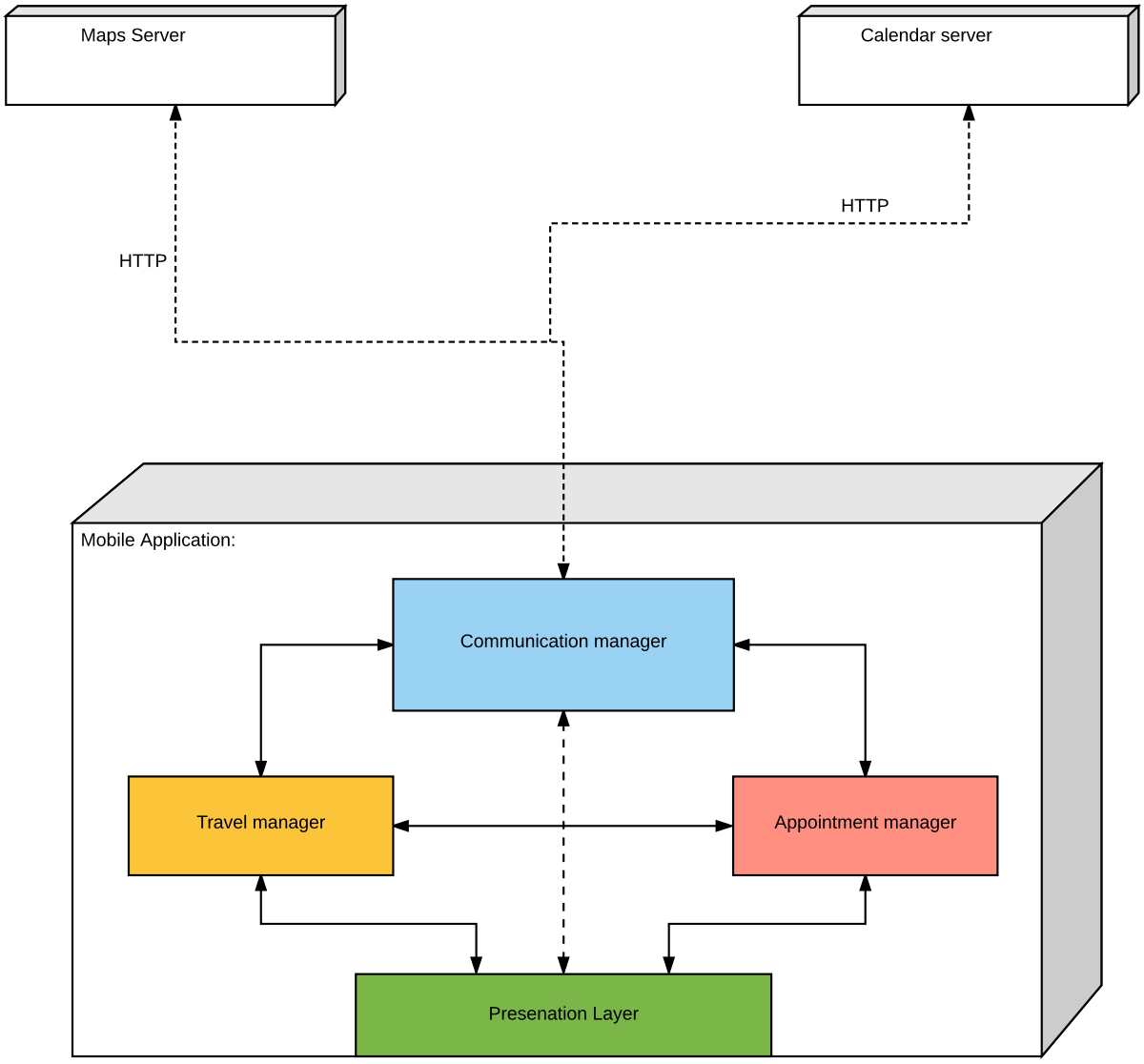

Figure 5: Deployment-view smaller test-system

We start with the test-system. As seen in figure 5, we can see that it consist of only one working part, which is the mobile-application. This is a basic master-slave system, where the application is the master and api is the slave. The slave only serves the master when asked to, this is simple and compact. The main function of this system should be to demonstrate Travlendar$^+$. This to uncover

whether there is a market or not.

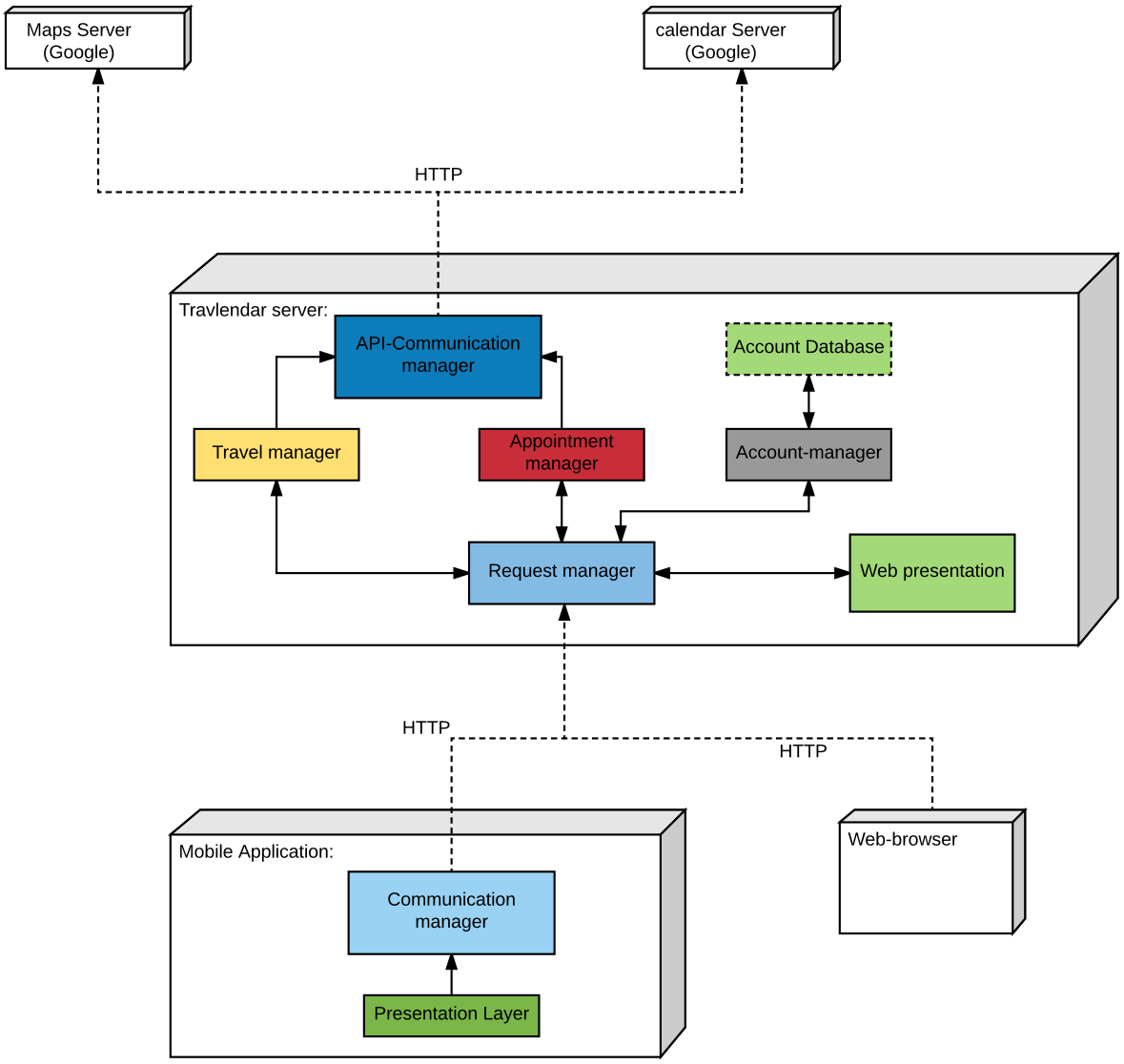

Figure 6: Deployment-view full system

This is shows the hardware of the system, and which software that are running

on each component. As we can see the main working component in our system is the server, while the other components are merely clients. As in any server/client relationship. This system supports access through a web-application as well as mobile applications. In addition it contains an account-system to make the system safer to use and user-information remain private.

2.6 Selected Architectural Styles and Patterns

Programming Language This is merely up to the developer and should be decided based on what the strengths and weaknesses of the language.
However we recommend using Python. The reason being the large toolbox and the diversity in python. Setting up classes, HTTP servers and threads in python is easy. It is also becoming more and more widespread. The drawback of said language is run-time(kind of slow compared to c) and the fact that synchronization of shared variables can be difficult/complex(this may be easier with go-channels or java-protection). But considering the architecture of the system Python should be a fine choice.

3 User Interface Design

This section contains GUI-mockups. These illustrations are not intended as a final design, only as means of illustrating the functionality and use of the software.

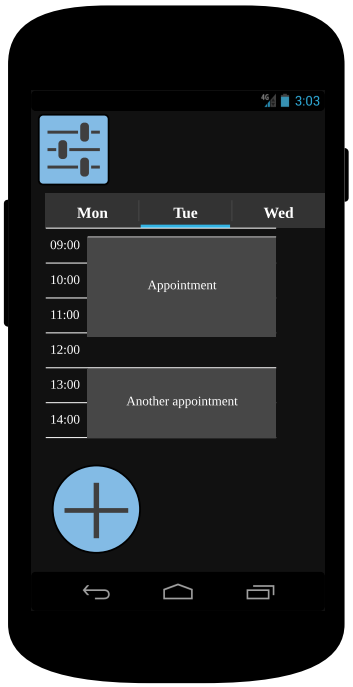

Figure 7: The application home-page

Figure 8: Make a new appointment

Figure 9: Choose type of transport

Figure 10: Appoinment information

Figure 11: Edit an appointment

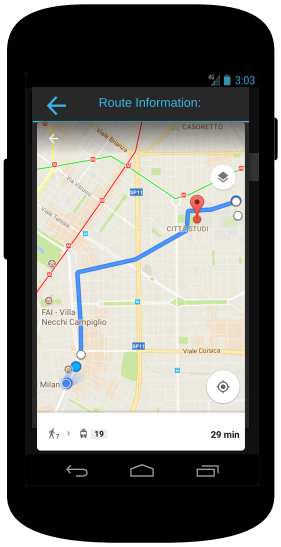

Figure 12: Travel route map

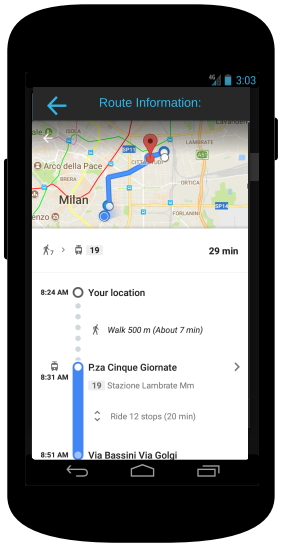

Figure 13: Travel route information

4 Requirements Traceability

The design given above will meet the requirements given in RASD in the following way(see Functional Requirements RASD):

4.1 Travlendar$^+$ User(Logged off) / New User

Requirement:	Solution:
Register as a new user.	Account manager - Register
User be able to Log-in	Account manager - Login
Recover lost PW	Account manager - Recover

4.2 Travlendar$^+$ User(logged in)

Requirement:	Solution:
Make an appointment	Appointment manager - Save appointment
Cancel an appointment	appointment manager - delete appointment
Update Personal Settings	Settings-manager

5 Work Log

Date:	Name:	Hours:	Total:
8.11.2017	Tobias Rasmussen	3	3
9.11.2017	Tobias Rasmussen	2	5
10.11.2017	Tobias Rasmussen	2	7
14.11.2017	Tobias Rasmussen	4	11
15.11.2017	Tobias Rasmussen	4	15
16.11.2017	Tobias Rasmussen	3	18
20.11.2017	Tobias Rasmussen	2	20
21.11.2017	Tobias Rasmussen	2	22
22.11.2017	Tobias Rasmussen	3	25
23.11.2017	Tobias Rasmussen	2	27
24.11.2017	Tobias Rasmussen	3	30
26.11.2017	Tobias Rasmussen	3	33

Table 1: Table showing worklog

6 Tools Used

- Sharelatex - `www.sharelatex.com`
  - To create the document itself
- Lucidchart - `www.lucidchart.com`
  - To create diagrams and GUI

7 References

- Google Maps API -
  `developers.google.com/maps/`
- Google Calendar API -
  `developers.google.com/google-apps/calendar/`