



Politecnico di Milano

AA 2017-2018

Computer Science and Engineering

Software Engineering 2 Project

# TRAVLENDAR+

## DD

Design Document

Spoto Marco 829633

1. Introduction
  - 1.1. Purpose
  - 1.2. Scope
  - 1.3. Definition, Acronyms, Abbreviations
  - 1.4. Reference Documents
  - 1.5. Document Structure
2. Architectural Design
  - 2.1. Overview
  - 2.2. Component view
  - 2.3. Deployment View
  - 2.4. Runtime view
  - 2.5. Component interfaces
  - 2.6. Selected architectural styles and patterns
    - 2.6.1. Overall Architecture
    - 2.6.2. Patterns
  - 2.7. Other design decision
3. Algorithm design
  - 3.1. Add an User
  - 3.2. Add an event
4. User interface design
5. Requirements traceability
6. Implementation, integration and test plan
7. Effort spent

# 1 Introduction

## 1.1. Purpose

This document is written for the developers of the software and the financial stakeholders and its objective is to guide the development phase in such way to respect all the requirements deriving from the RASD document. This document will give all information on software design to allow the developers to implement the system.

## 1.2. Scope

This document specifies all the software components included in the system. It explains how components must be deployed and how they interact at run-time. It also gives an idea of the design decisions taken during the design phase itself. This is done using different diagrams in order to better explain the software structure.

## 1.3. Definitions, Acronyms, Abbreviations

- Definitions

Time slot: Time slots are the slot in which you can create an event, basically they are standard, but every User can personalize them changing starting time and duration.

Preferences: all the possibility of means of transport.

Model logic: is the core of the software.

- Acronyms

API: application program interface.

- Abbreviations

[RW<sub>n</sub>] The n-th run time view.

[UI<sub>n</sub>] The n-th user interface

## 1.4. Reference Documents

- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, which can be retrieved on the beep page of the course

## 1.5. Document Structure

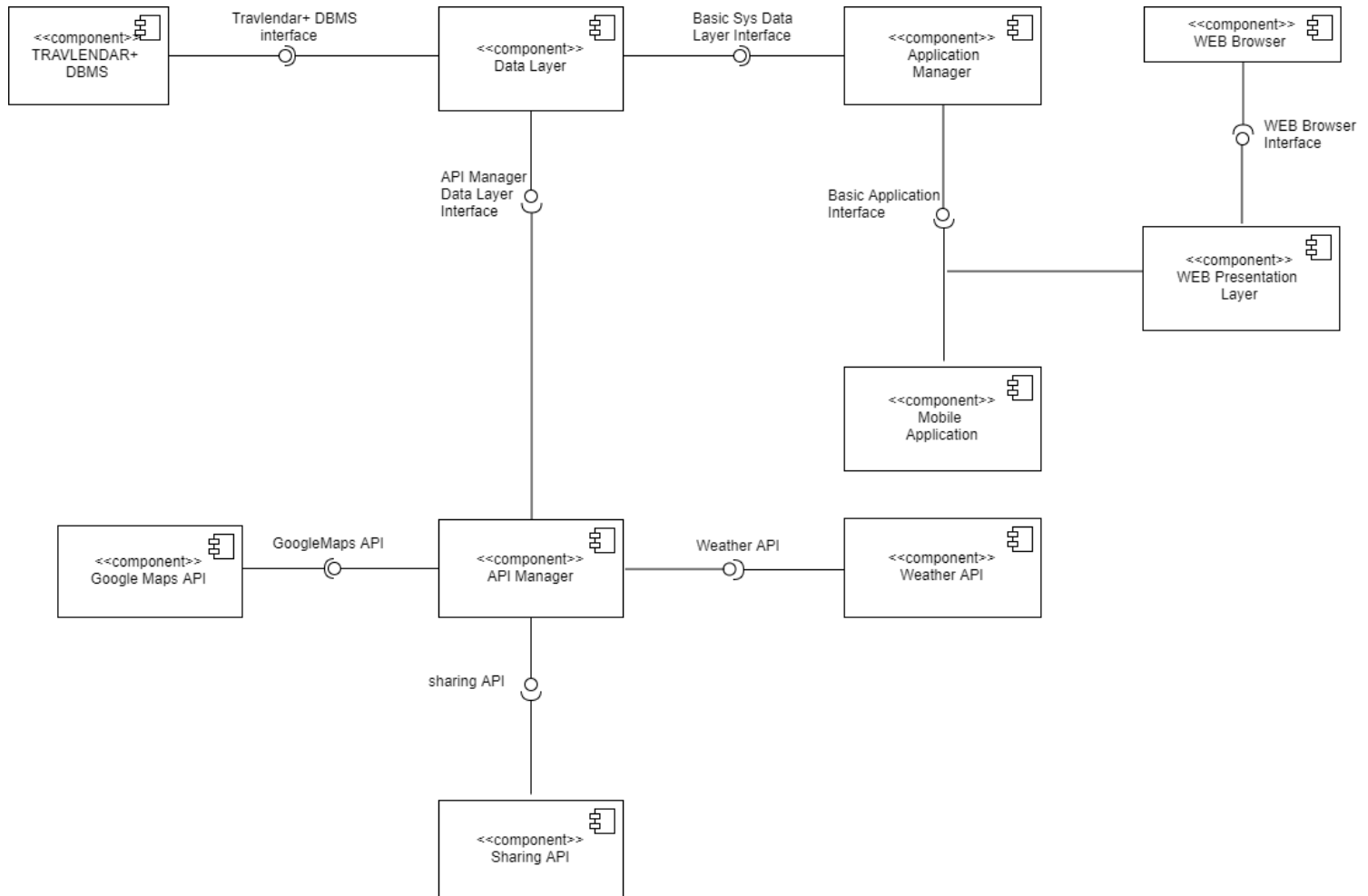
- Introduction: A description of the content provide purpose and scope of the document.
- Architectural Design: This section will provide the design of the application. Here there are all the UML diagrams to specify how the software will be implemented. And all the architectural choices.
- Algorithms Design: This section will provide the way in which algorithms of this application will work.
- User Interface Design: This section will provide a more specific view over design of User interface. Here we can find some modification compared to the one of RASD document, this change are done after some study to solve problems found in the RASD version.
- Requirements Traceability: this section will give the bond between requirements in RASD and the Design decision.
- Implementation, Integration and TestPlan: This is the section in which the plan and the approach to the implementation is described. Than there is an explanation of the integration and test plan.
- Effort Spent: working time spent.

# 2. Architectural Design

## 2.1. Overview

This section is addressed to the architectural design of Travlendar+. In the following subsections are show the most relevant part of the system-to-build. Here we also have a deepening of the software components of the system and the interfaces about them.

## 2.2. Component view



### Travlendar+ DBMS

This is the data base management system which will manage all the important information of the system. Some of these are the users profiles and their calendars. Obviously the DBMS will provide an interface which allow the Data Layer to perform queries over the data.

### Google Maps API

This component represents the part of software that include all the services provided by Google and Waze to build an application aware of the Users position and able to calculate all the alternatives routes. The application software takes the data provided from these and provide the one with the longest time estimate (it is better to arrive 2 minutes before than 2 minutes

late) and offers an interface with the route, arriving time, duration. If you use public transports it also provide the timetables of them or if you use taxi/uber it provide the starting point and the starting time of the ride. This component will be used by the API manager and the Application Manager.

## Weather API

This component represents the part of the software that include all the service provided by MeteoAM to build an application that can advise you when weather conditions are not favourable. This component will be used by the API manager and the Application Manager.

## Sharing API

This component represents the part of the software that include all the service about sharing (car, motorbike or bike). It offers an interface that provide to the users all the available sharing alternatives. This component will be used by the API manager and the Application Manager.

## WEB Browser

Is the component representing the software through which a user can access the web application.

## Data Layer

This piece of software will deal with the DBMSs, giving them the possibility to recover or to store relevant information to the system to build. Data Layer is useful to use an unique interface to access in different kind of data which are stored in different databases.

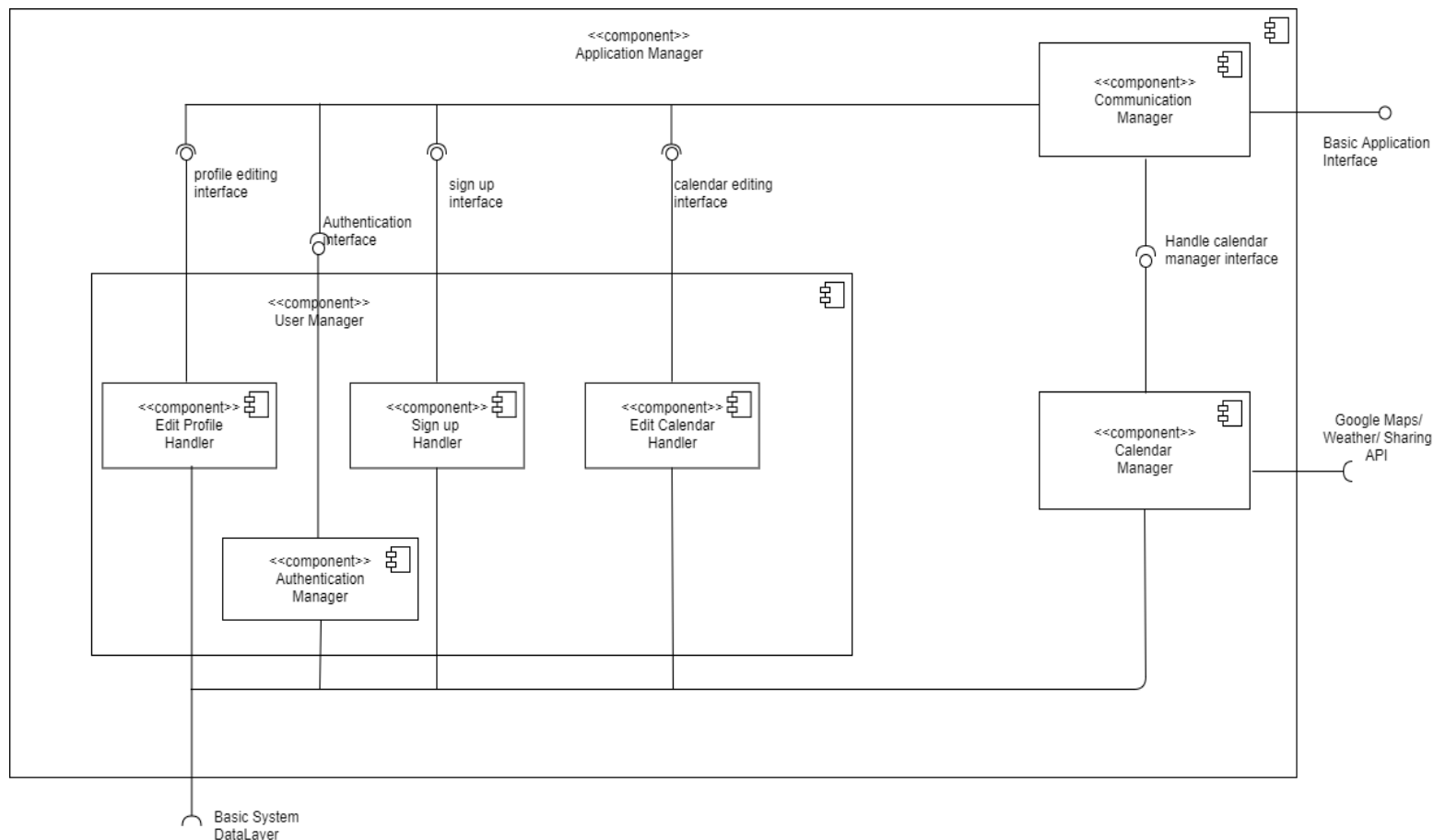
## API Manager

This component of the system is used to handle all the API calls coming from web/mobile application. Of course to answer this calls the API Manager has to use other components which are Data Layer and Google/Sharing/Weather API.

Now I will deepen some part of the applications.

## Application Manager

The basic application manager is divided in three sub-components which are: Communication Manager, Calendar Manager and the User Manager. This subdivision because the Application Manager has to provide a way to notify the user, has to handle user' credentials and has to handle user' calendars.



## Communication Manager

This part of the Application Manager receives all operations requests from mobile application and from web presentation layer. It also handles all the

notifications which must be sent to the users due to planning their appointments.

## Calendar Manager

This sub component handles all the arrangements belonging to calendar' planning. So it has to store the events through the Basic System Data Layer and calculate if it is a possible event. It has to calculate all possible route crossing Google maps and Waze information giving time, costs and means of transport for it.

## User Manager

The User Manager is in turn decomposed edit profile handler, sign up handler and edit calendar handler. This decomposition is due to the fact it has to offer three different services. Obviously to make possible this three operation these will use the basic system data layer.

### Edit Profile Handler

This component is made to edit user' profiles.

### Sign up Handler

To this component is entrusted the registration of new Travlendar+' users.

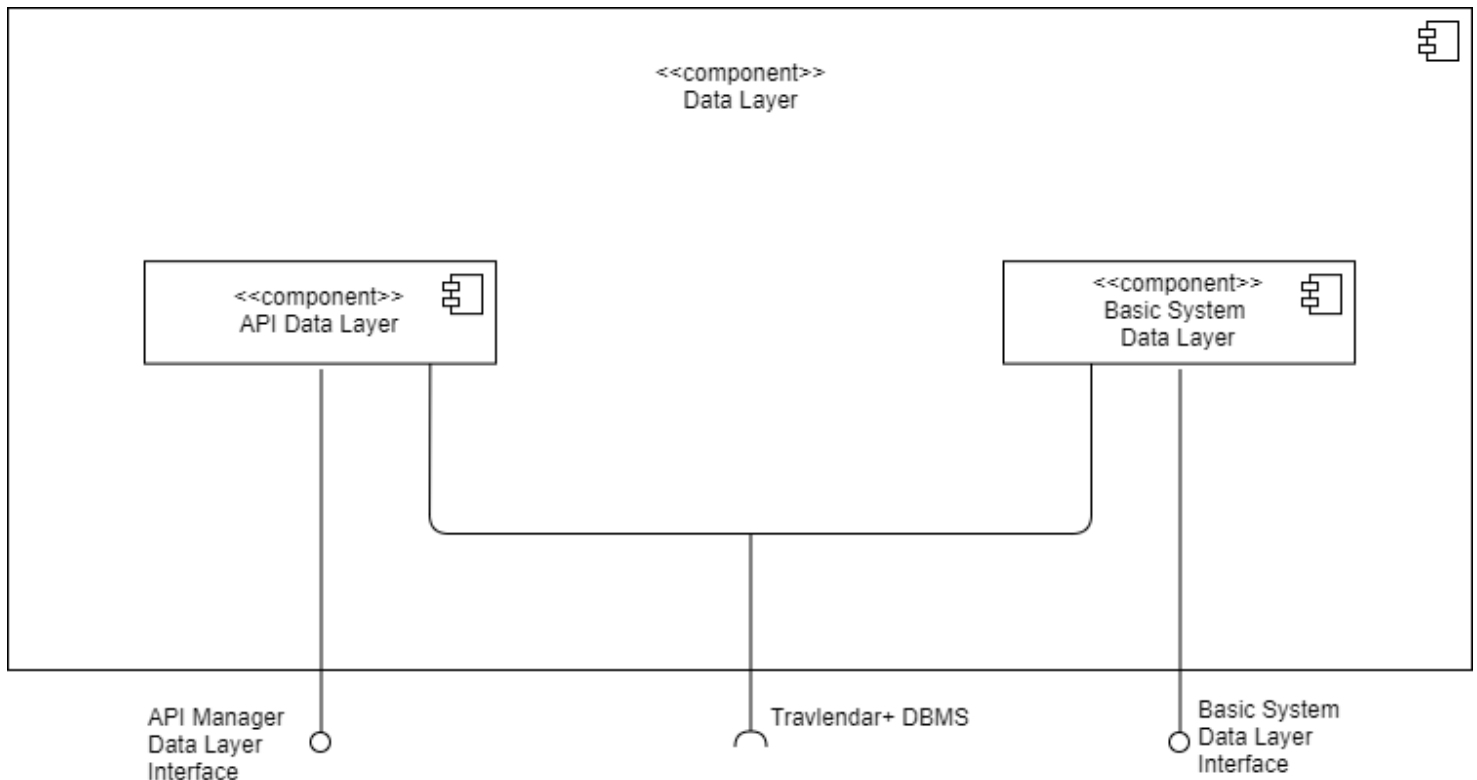
### Edit Calendar Handler

This component will handle all the creation/deletion and modification operations make by the users in their calendar.



## Data Layer

The data layer has two sub-components according in order to separate requests' management coming from API Manager and from the Application Manager.



### API Data Layer

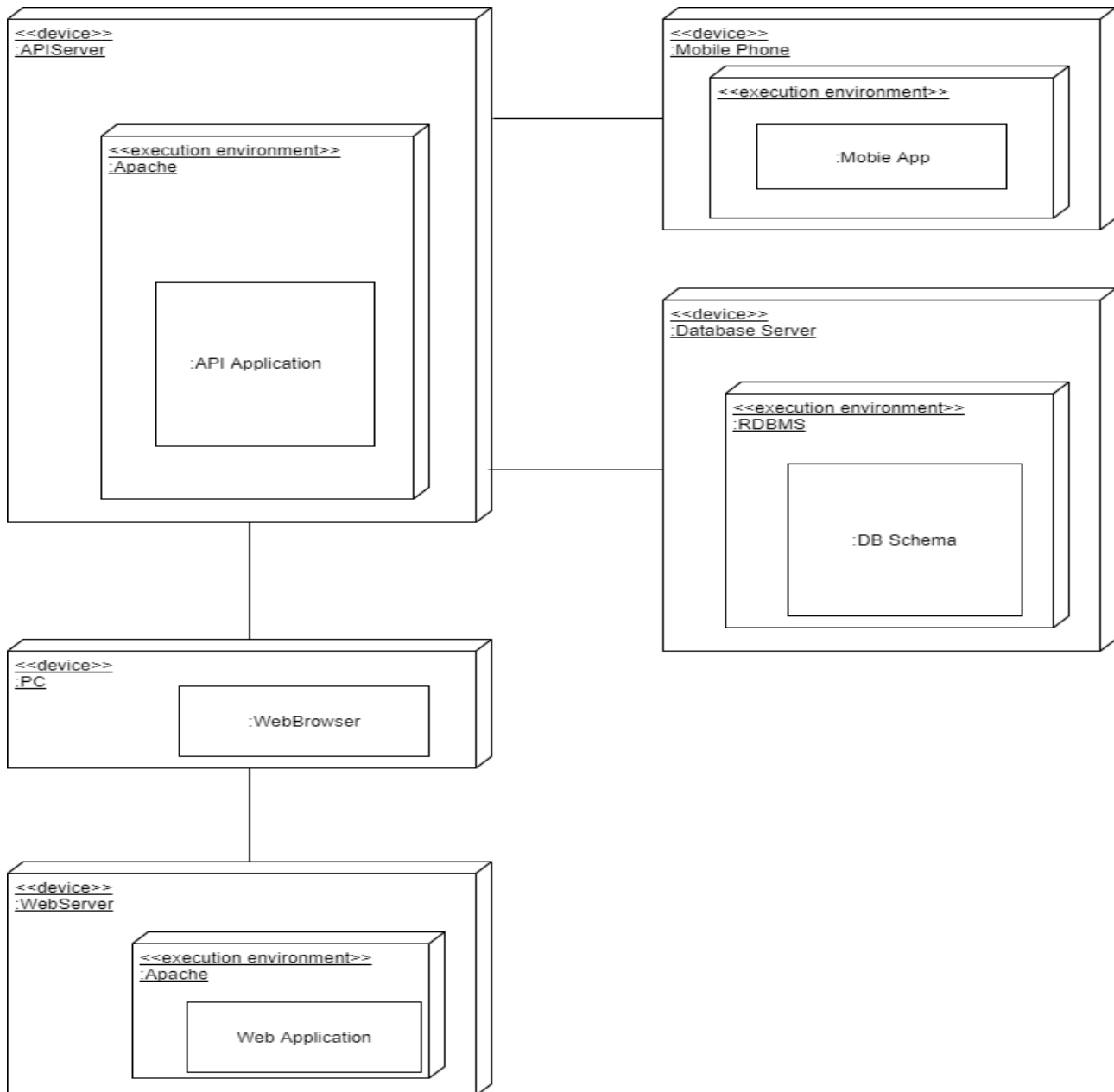
This component has to manage all the request coming from the API manager which cannot perform operations of behalf of a calendar. It also has to translate the Application Manager request in query for DBMS.

### Basic System Data Layer

To this component is entrusted the management of the request coming from the Application Manager. This component has access to Travlendar+ DBMS because the Application Manager has to handle requests related to the users. It has also a security function. It makes impossible a direct access to the DBMS but before you must make a further step from the server.

## 2.3. Deployment View

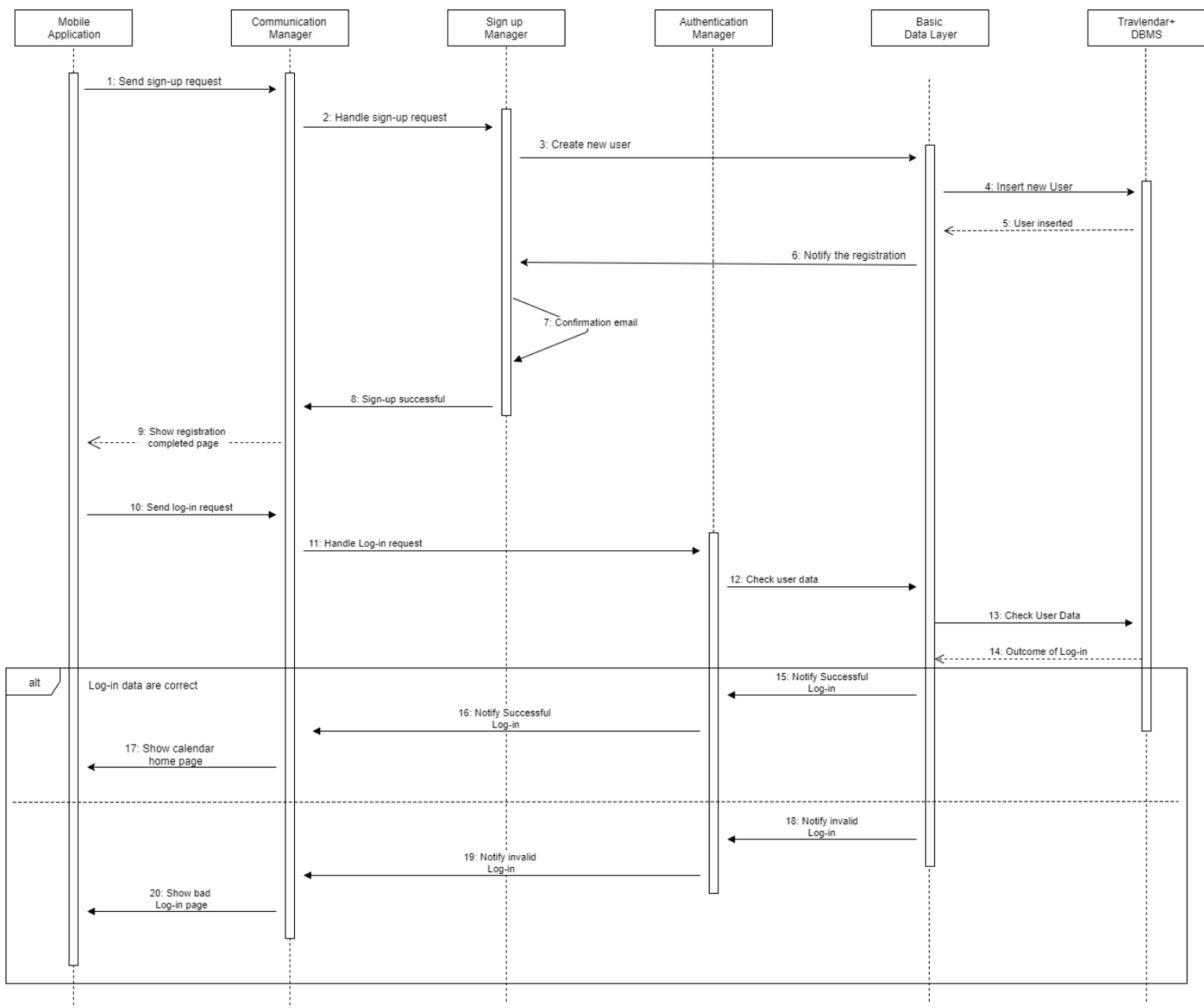
Here I will describe the deployment of the component previously described.



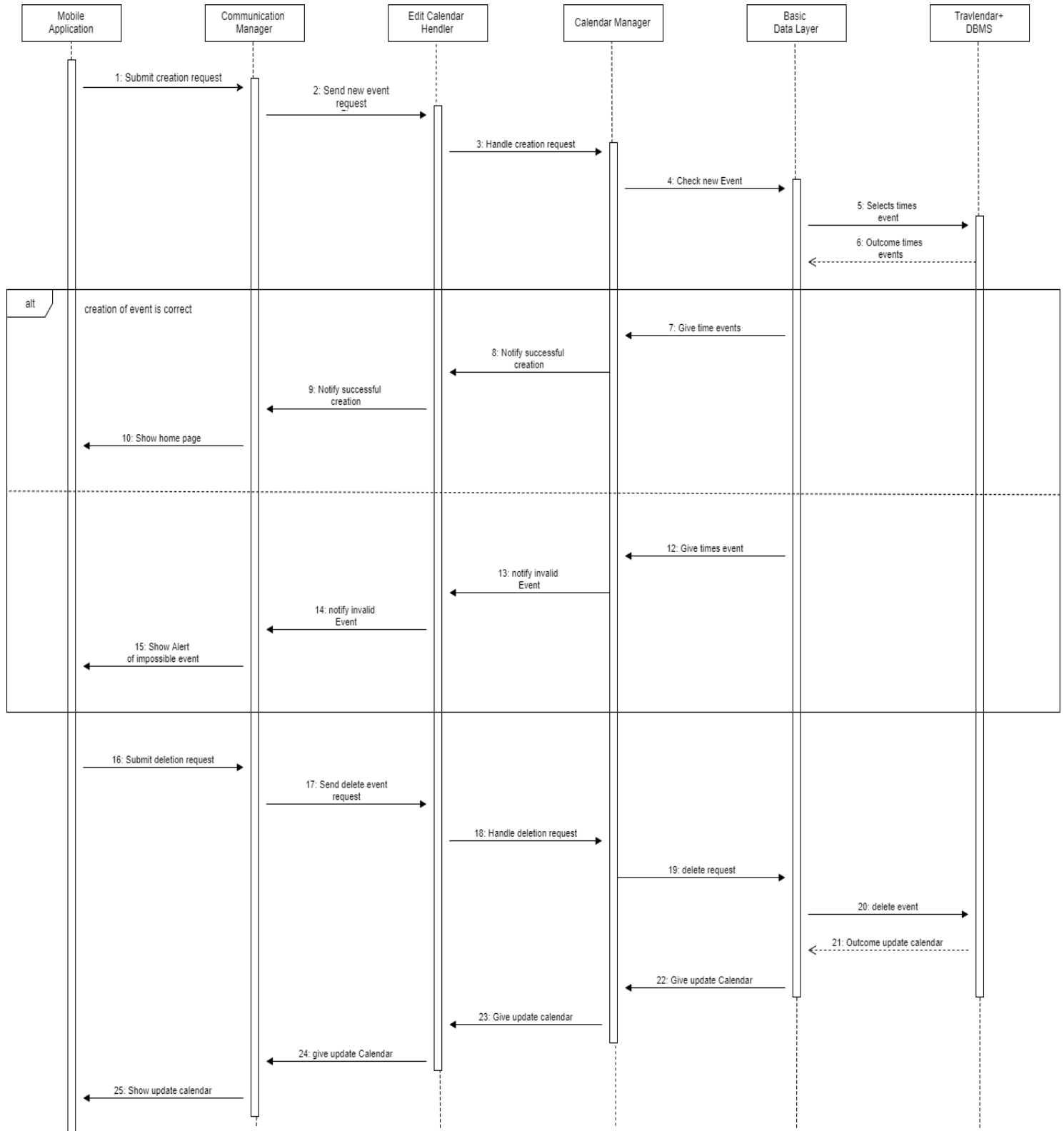
## 2.4. Runtime view

This section is made to explain how the components interact each other to accomplish tasks related to the use cases of the RASD document.

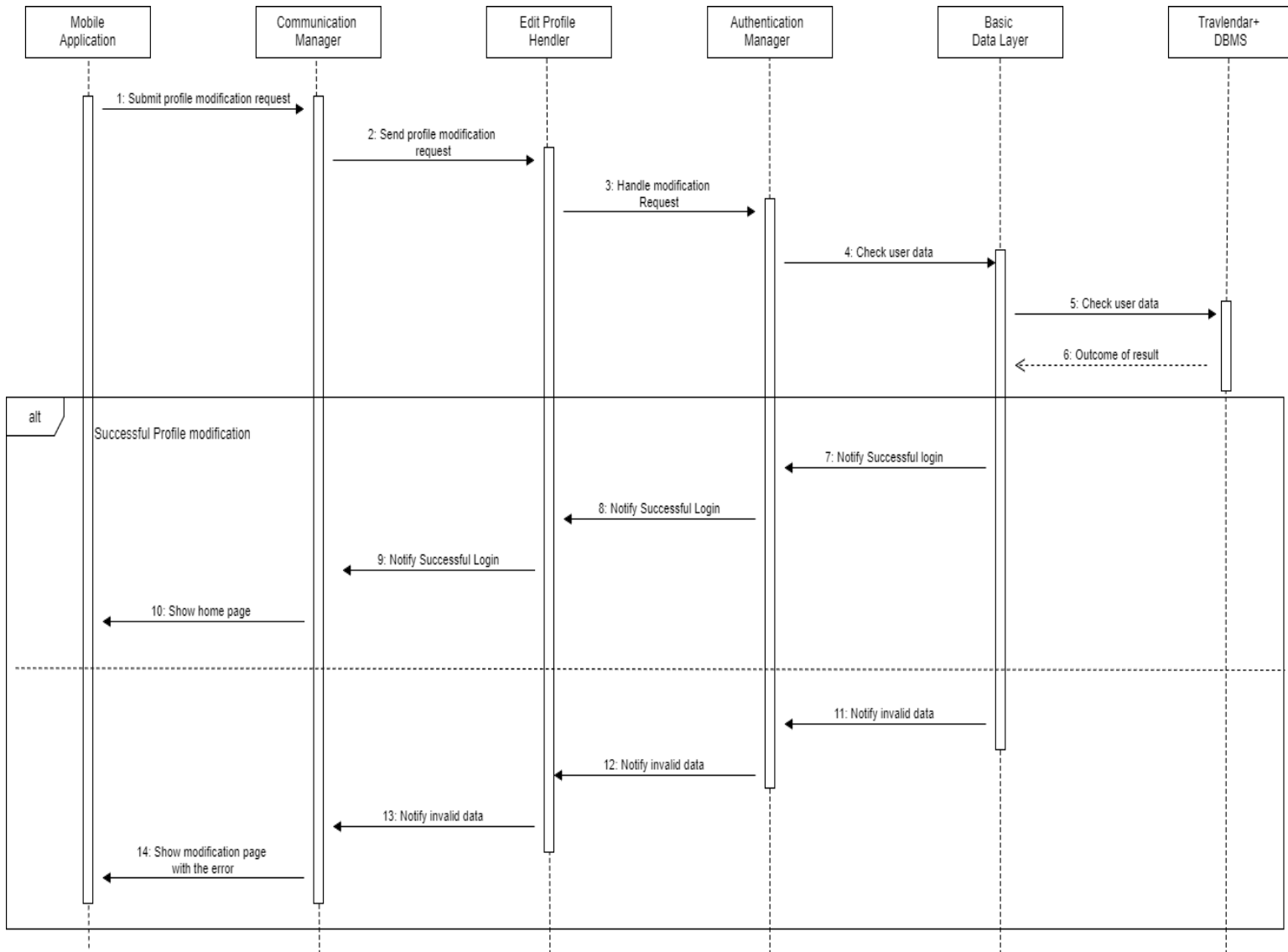
**[Rw 1]** This runtime view shows components' interaction in a scenario in which someone , through the mobile application, wants to register himself to the service as an user. Subsequently, the diagram shows the scenario of an User Log-in. I include also the messages exchanged in case of failure.



**[Rw 2]** This runtime view shows the interaction between components in a scenario in which an User wants to create an event. All the operations exchange of messages are shown, even in case of failure. It also contains a scenario in which an user wants to delete an event.



**[Rw 3]** This runtime view shows the interaction between components in a scenario in which an user wants to change his profile credentials. All the operations exchange of messages are shown, even in case of failure.



## 2.5. Component Interfaces

- **Travlendar+ DBMS:** This data base management system will be chosen taking account the solution which best fit the constraints. The only constraint is that the chosen solution must offer an interface accessible through JDBC, which allows to access the database, performing queries and storing new data, from a java application.
- **Data Layer:** This component offers two interfaces: the basic system data layer interface and the API manager data layer interface.
  - **Basic system data layer interface:** It offers all the functionalities to access and store data over the DBMS the system works with. The basic functionalities offered are: modify, edit user profile; delete event; authentication, to check the credential of an user; store new event, to save a new event over the data bases. It also has a security functionalities.
  - **API manager data layer:** It offers functionalities to access data over the Travlendar+ DBMS. It has to translate Application Manager request in query to the DBMS.
- **API Manager:** It offers all the functionalities needed by an API based system to access the basic service that are: request, to allow an API based system to yield a request.
- **Application Manager:** It is divided into three sub-components so here are described the interfaces offered by these sub-components.
  - **Basic Application interface:** It has to offer all the functionalities which allow the web presentation layer and the mobile application to render the view. This functionalities are: authentication, to grant access to the service; profile editing, to edit profile information; sign up, to register new user to the service; delete event, to delete an event previously asked.
  - **Handle calendar manager interface:** It has to offer all the functionalities to allow the communication manager to properly request operation over calendar. This functionalities are: handle calendar, which ask to handle a new event.

- Sign up interface: It has to offer the functionality to register a new User over the system.
- Authentication interface: It has to offer the functionality to authenticate a user which wants access to the service.
- Profile editing interface: It has to offer the functionality to edit the profile of an user.
- Calendar editing interface: It has to offer the functionality to edit the calendar of an user.
- Google Maps/Weather/Sharing API :
  - Google Maps API interface: This interface is used to compute the estimated arrival time of a path, the duration of the alternatives.
  - Weather API interface: This interface is used to send an alert in case of bad weather.
  - Sharing API interface: This interface is used to see all affordable means near the User.
- Web presentation layer:
  - Web browser interface: This interface allow the User to surf on web pages representing the web application and obviously collects the information related to the action performed by the user over the web pages.
  -

## 2.6. Selected architectural styles and patterns

### 2.6.1. Overall Architecture

This application will be divided in three tiers:

1. Database (Data Access Layer).
2. Model Logic (Business Logic Layer).
3. Client (an easy interface for the model).

### 2.6.2. Patterns

Some part of this application is designed using a client-server application architecture like: the log in functionalities in which an user send his log-in information to the application manager and it answers with the data to be

rendered by the web presentation layer or the mobile application. In the same way is implemented the profile editing, the sign-up and the event deletion.

Other parts have an event-based architecture like: the alert that must be send in case of bad weather or the impossibility to create an event because the impossibility to arrive on time.

One of the most used pattern is Model-View Controller in which the view is composed by the thin clients, mobile application and web browser, the model by the model logic and the controller by part of the model logic and database. According to this MVC the model is continuously in communication with view and controller due to update the new information and to administrate the application.

## 2.7. Other design Decision

I want to grant an high level of availability, according to this it must be implemented an active-active redundancy pattern over the critical part of the service. This pattern is used for API Manager and the Data layer.

Travlendar+ DBMS must be an active database it must be able to reserve a timeslot for an event and to delete an event if it is required. It must be able to guarantee the constraints of the authentication.

The language implementation of this system is Java the smoothly implement this kind of systems (javaEE).

## 3. Algorithm Design

In this part I will provide a definition for most important algorithm implemented in the application; I want to explain not the code but I want to describe it in terms of input to execute the algorithm and the output that it must return, also giving the steps that the model logic does during the execution.

### 1. **ADD AN USER**

To use Travlendar+ functionalities people must register them in the its database.

Input ->

User has to fill the box provide by the application (p.18 UI2) in the correct way.

Travlendar+ - DD 1.0



Execution ->

Model logic check:

1. Password with more than eight characters.
2. Username is not already token.
3. E-mail is not already in use

If these three conditions are respected it gives Visitor's data to the DBMS and the visitor is added as an User.

Output ->

The visitor become an User and the system show the UI 4 (p.19) or something wrong occur and the system generate an alert to report the problem

## **2. ADD AN EVENT**

This algorithm concerns the addition of anew event in a calendar by the user. This algorithm is very relevant because it contains one of the principal goals of Travlendar+ application. It has also to check that an unreachable event cannot be created.

Input ->

The User has to create a new event clicking on "i" and giving all information about it:

1. Day
2. Title
3. Starting time
4. Duration
5. Preferences

Execution ->

Model logic check:

1. Starting time is different from other starting time in the same date
2. Staring time is not in a busy slot
3. Starting time is not in the past
4. Event is reachable on time. For this check is used an application like G. Maps. The model give starting and arrival points, view how much time is used to reach the goal and calculate if the User can reach it five minute before the required time.

Output ->

User's calendar is updated with the new event or the system generate an alert to communicate some problems.

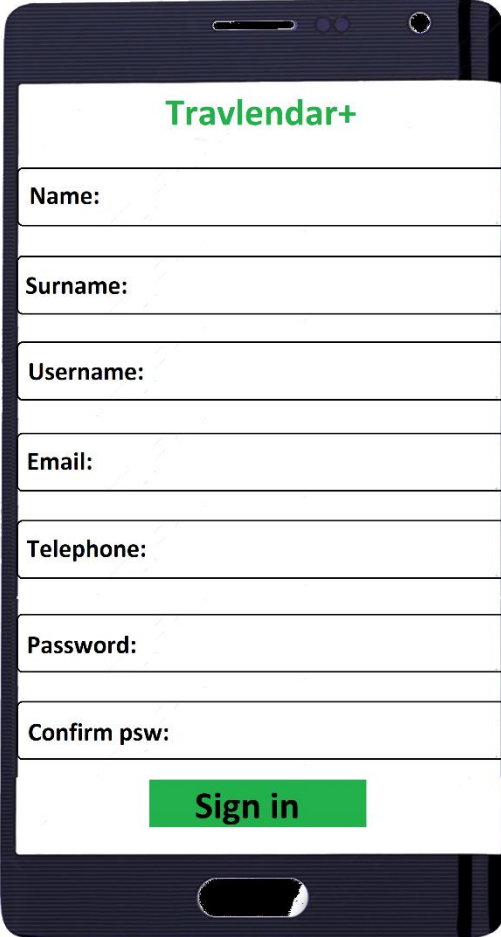
## 4. User Interface Design

In this section it is provided an overview of the user interfaces of Travlendar+ system. I'll present only the mobile version due to the fact that the web application interfaces are the same, at least in the essential core.

**[UI1]** Application Home Page. This application represent the start interface and it is shown when a not logged-in user opens the applications.

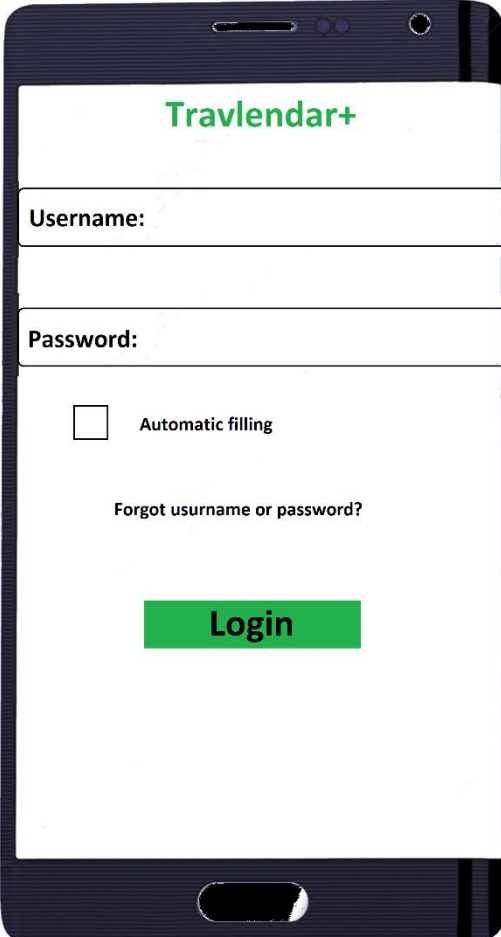


**[UI2]** User Sign-up.



The image shows a mobile application interface for user sign-up. At the top, the app name "Travlendar+" is displayed in green. Below it, there are seven input fields for "Name:", "Surname:", "Username:", "Email:", "Telephone:", "Password:", and "Confirm psw:". Each field has a light gray border and a small icon on the right. At the bottom, there is a green button labeled "Sign in".

**[UI3]** User Log-in.



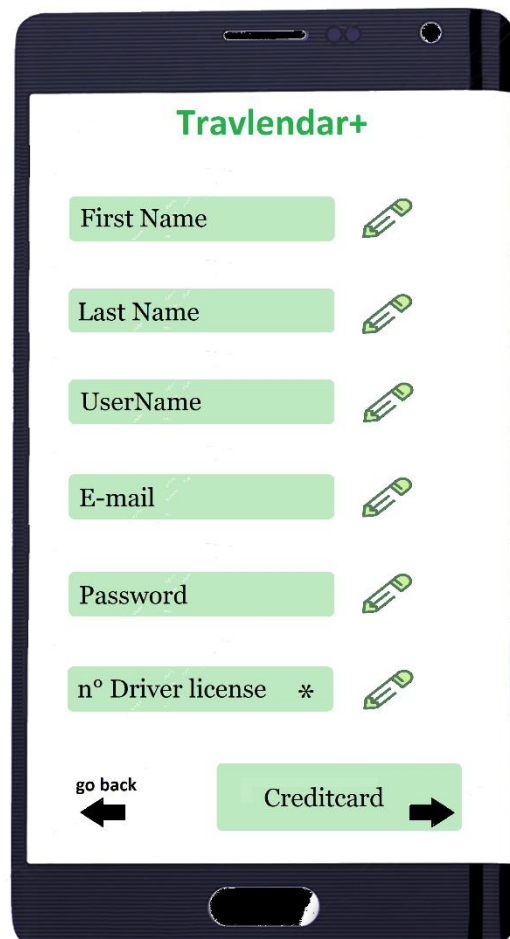
The image shows a mobile application interface for user login. At the top, the app name "Travlendar+" is displayed in green. Below it, there are two input fields for "Username:" and "Password:". Below the password field, there is a checkbox labeled "Automatic filling". Below the checkbox, there is a link that says "Forgot usurname or password?". At the bottom, there is a green button labeled "Login".

#### [UI4] User Home page.

In this page there are all the timeslot, alert and the day that the user is looking is highlight in blue. The user can change day by clicking on the number over the first timeslot.

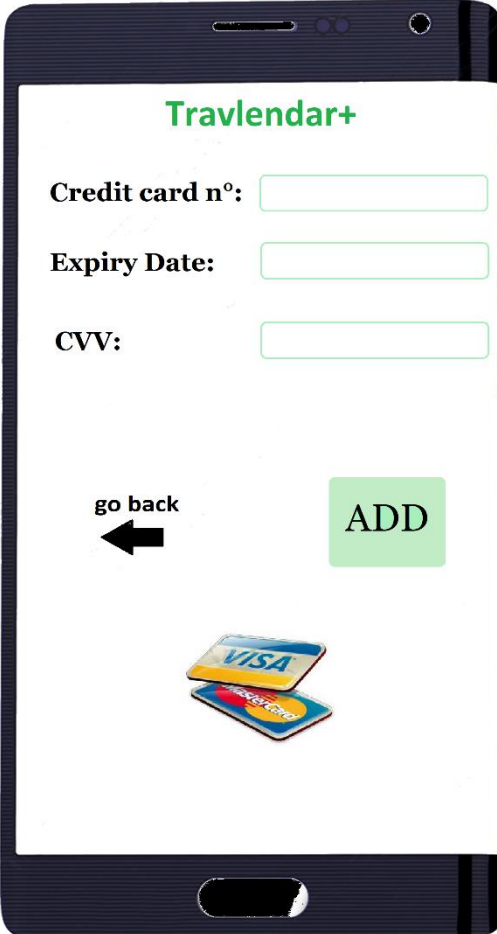


#### [UI5] Edit Profile.



## [UI6] Add Credit card.

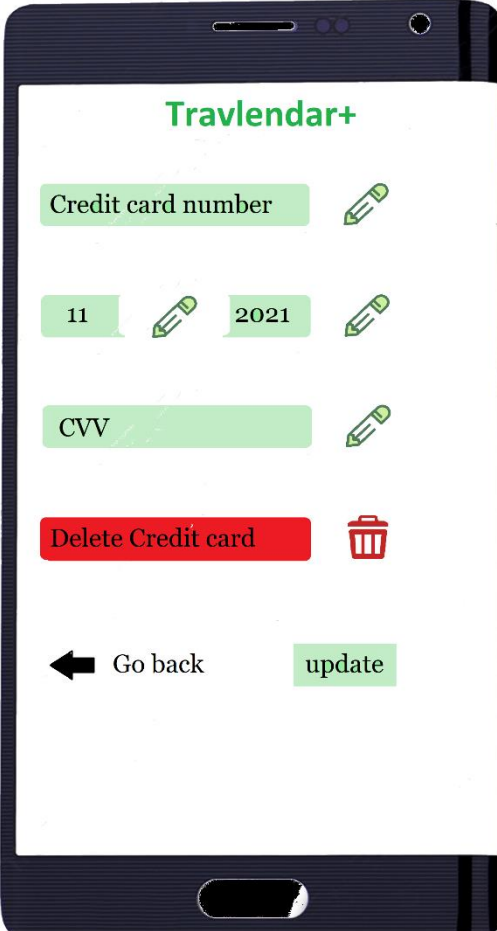
(If the user click-on Credit card and he does not have a credit card added yet the application open this page).



The screenshot shows the 'Travlendar+' app interface for adding a credit card. It features three input fields for 'Credit card n°', 'Expiry Date', and 'CVV'. Below these fields are two buttons: 'go back' with a left-pointing arrow and a green 'ADD' button. At the bottom center, there is an illustration of two overlapping credit cards, one Visa and one Mastercard.

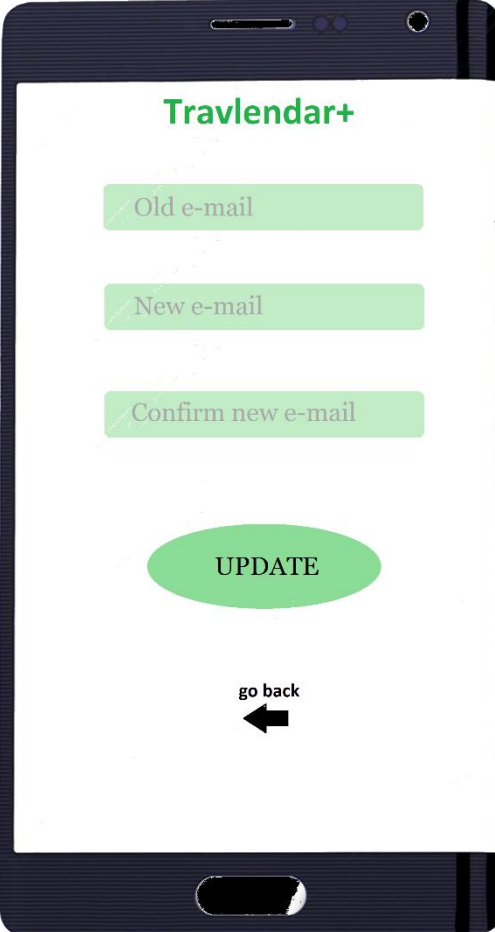
## [UI7] Delete/Modify Credit card

(If the user click-on Credit card and he has a credit card added yet the application open this page).



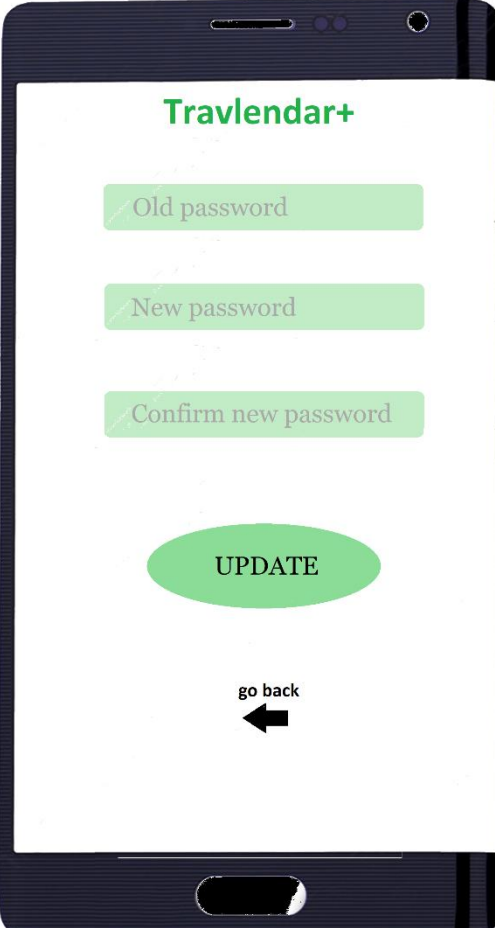
The screenshot shows the 'Travlendar+' app interface for deleting or modifying a credit card. It displays the card details in green boxes: 'Credit card number', '11' (representing the first two digits of the number), '2021' (representing the year), and 'CVV'. Each detail has a green pencil icon to its right for editing. Below these details is a red button labeled 'Delete Credit card' with a red trash can icon to its right. At the bottom, there is a 'Go back' button with a left-pointing arrow and a green 'update' button.

**[UI8]** Change user e-mail.



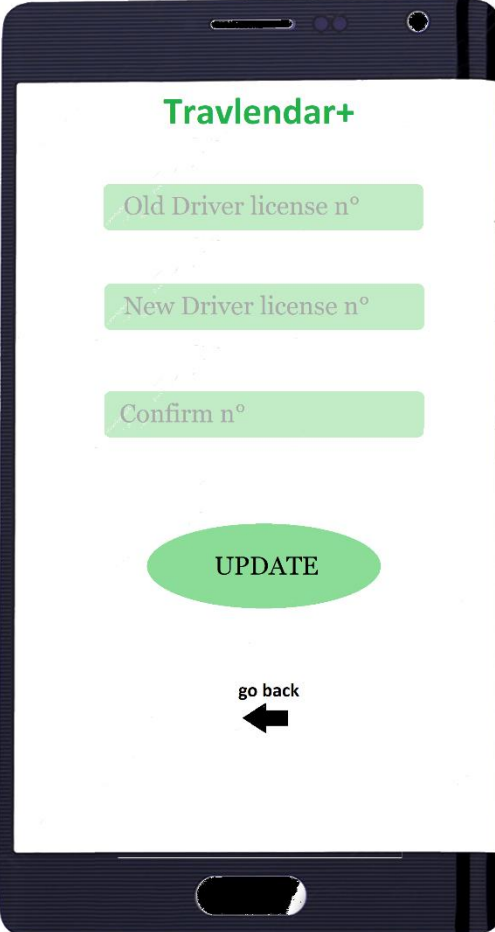
The image shows a mobile application interface for changing an email address. The app is titled "Travlendar+" in green text at the top. Below the title, there are three light green rectangular input fields with rounded corners, containing the placeholder text "Old e-mail", "New e-mail", and "Confirm new e-mail" respectively. Below these fields is a large, light green oval button with the word "UPDATE" in black capital letters. At the bottom of the screen, there is a "go back" link in black text next to a black left-pointing arrow. The entire interface is displayed within a dark blue smartphone frame.

**[UI9]** Change user password.



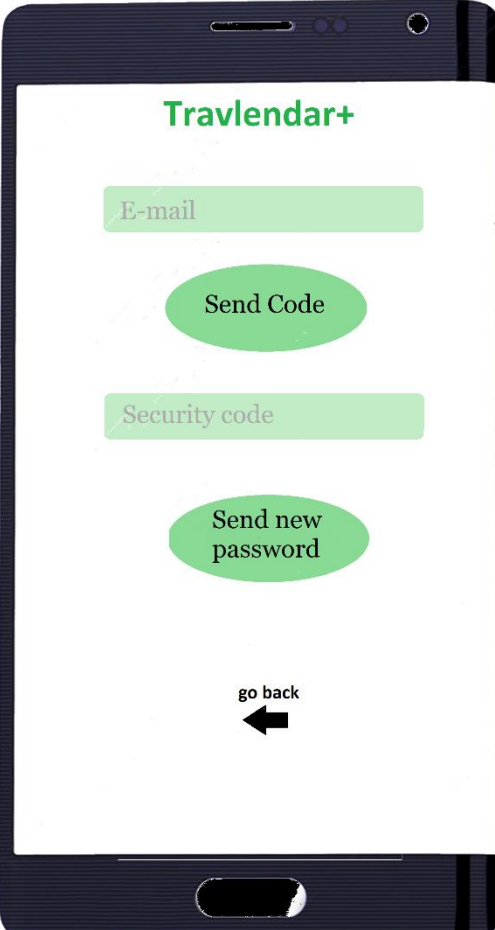
The image shows a mobile application interface for changing a password. The app is titled "Travlendar+" in green text at the top. Below the title, there are three light green rectangular input fields with rounded corners, containing the placeholder text "Old password", "New password", and "Confirm new password" respectively. Below these fields is a large, light green oval button with the word "UPDATE" in black capital letters. At the bottom of the screen, there is a "go back" link in black text next to a black left-pointing arrow. The entire interface is displayed within a dark blue smartphone frame.

**[UI10]** Change driver license number.



The image shows a smartphone screen with the Travlendar+ app interface. At the top, the app name "Travlendar+" is displayed in green. Below it, there are three light green rectangular input fields labeled "Old Driver license n°", "New Driver license n°", and "Confirm n°". In the center, there is a large green oval button labeled "UPDATE". At the bottom, there is a "go back" link with a left-pointing arrow.

**[UI11]** Password recovery

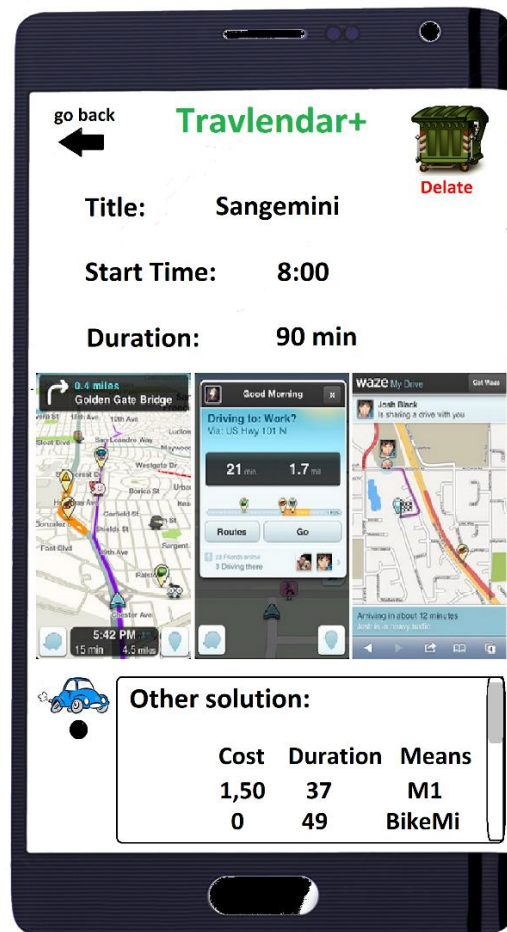


The image shows a smartphone screen with the Travlendar+ app interface for password recovery. At the top, the app name "Travlendar+" is displayed in green. Below it, there is a light green rectangular input field labeled "E-mail". In the center, there is a green oval button labeled "Send Code". Below that, there is another light green rectangular input field labeled "Security code". Further down, there is a green oval button labeled "Send new password". At the bottom, there is a "go back" link with a left-pointing arrow.

## [UI12] Appointment information

In this section we can find the title of our appointment the duration and the starting time. Then the solution chosen according user's preferences.

In the bottom left preferences chosen and in the bottom right other solution with all the information.



## 5. Requirements Traceability

In this section I will explain how the requirements defined in the RASD document are created in this document. In order to understand the mapping of requirements are implemented. It is suggested to read the RASD document to better understand this chapter.

-[G1] Allow an User to log in the system.

- The Application Manager through the Authentication Manager, it will provide the Log-in functionality, it also has to provide a password recovery functionality.

-[G2] Allow an User to view the calendar & [G3] Allow an User to create/delete an appointment.

- Application Manager provide the home page.



-[G3] Allow an User to create/delete an appointment.

- Application manager through edit calendar and Calendar manager, they provide the page through which the user can create, modify, delete an event.

-[G4] Allow an User to change his credentials and to add a credit card used for pay automatically.

- Application manager with edit credential, provide the page used to change User' credential
- Application manager with edit credential, give the possibility to add a credit card through which the user can do automatic payment.

-[G5] Allow an User to choose any type of preferences.

- Application manager through edit calendar and Calendar manager, when the User is in the page to create, modify, delete an event or in which he can view an event he can chose or change the preferences for the path.
- Application manager with calendar manager have to provide all possible alternatives of transport, giving costs and duration.

-[G6] Allow a System Manager to do operations on the system for updating and maintenance.

- Application manager with calendar manager, update weather traffic conditions, must be able to change trip solution according user request and send an alert fifteen minutes before every event.

-[G7] Allow a Security System to avoid any problem.

- Application manager through calendar manager, send an alert in case of overlap events or unreachable event in the moment of the creation of it.
- Data layer with basic system data layer, guarantee privacy .

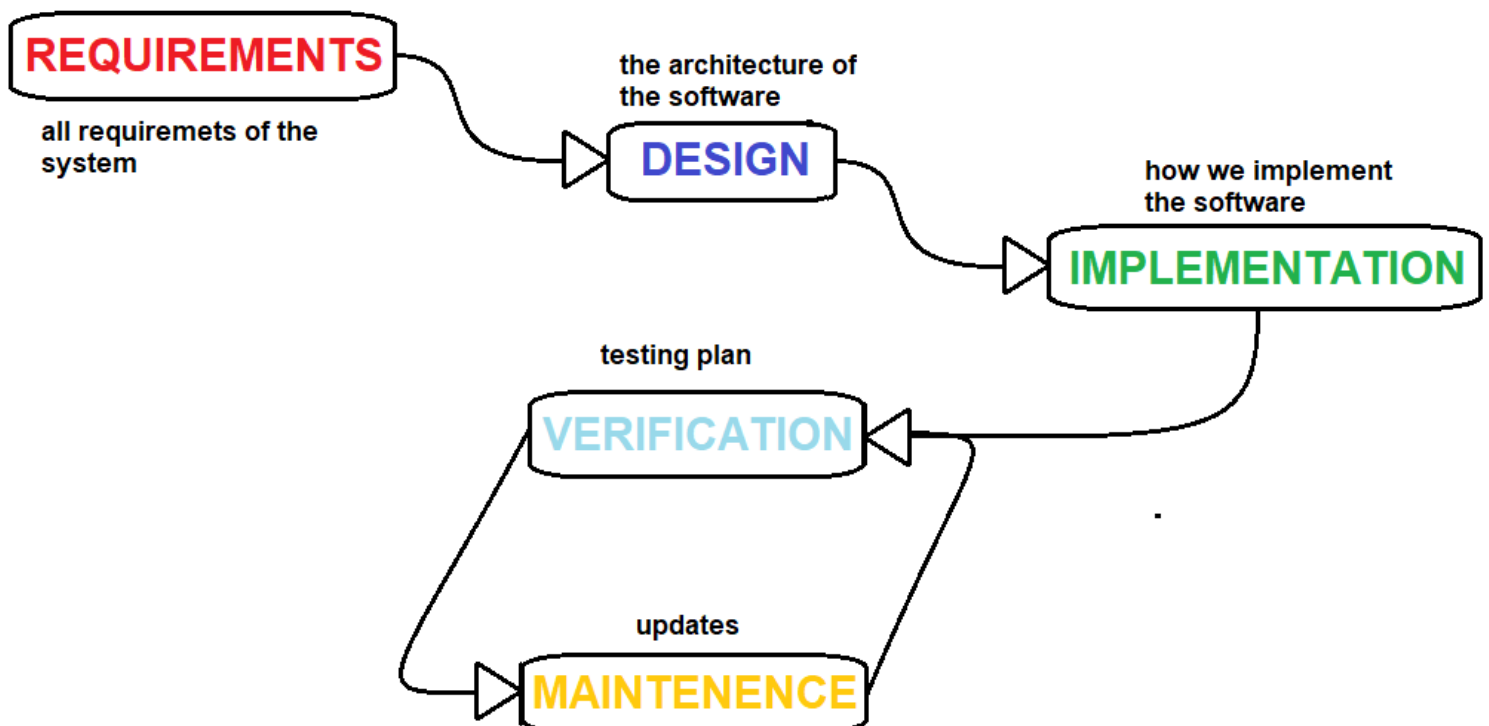
## 6. Implementation, integration and test plan

In this section I will identify the plan to implement the subcomponent of my system and the order in which I plan to integrate all subcomponents and the way in which I want to test.

To implement Travlendar+ I will use an waterfall model approach. First of all the project must be evaluated and there is a study of the context and a feasibility study, in which are analysed all requirements.

Then there will be a study of how the system will done all the requirements established before. In this section of the model I will also define all the subcomponents and how they will be linked to each other.

Once this is done there will be the code part in which all subcomponents are created in java code.



When I will finished the development part I will start the testing part, in which all part will be tested both alone, with a precompile testing code, and in group with the other to test if all the connections are done in the correct way. The testing phases will be done mixing black and white box testing. In the first case I will charge a programmer to select valid and invalid input and determines the correct output so that he can test all without know nothing about the internal structure; in the second case I will test the internal structure, I use this part of testing also to make sure that all bottleneck are implemented in order to minimize delay.

At the end of all this process the system is ready to the delivery to the users. When it will be finished and delivered there will be all maintenance operations.

Application components implementation, integration and test plan are splitted up as reported below:

1. Skeleton of the FrontEnd component implemented and tested
2. App Manager with its subcomponents are the most important and complicated part to be implemented.
3. Once created all the subcomponents there is the testing phase in which all subcomponents are tested alone.
4. API manager will now be implemented and tested.
5. DataLayer will now be implemented and tested with a virtual database.
6. Now all Application manager's subcomponents must be integrated each other, with DataLayer and with API manager . The testing phases takes place.
7. Application manager is integrated to FrontEnd.
8. DBMS attached to DataLayer to test App Manager with real Data.
9. System is ready to final testing.

## 7. EFFORT SPENT:

The effort spent by the team is of 70 hours.