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.

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.

1.3. Definitions, Acronyms, Abbreviations

1.4. Reference Documents

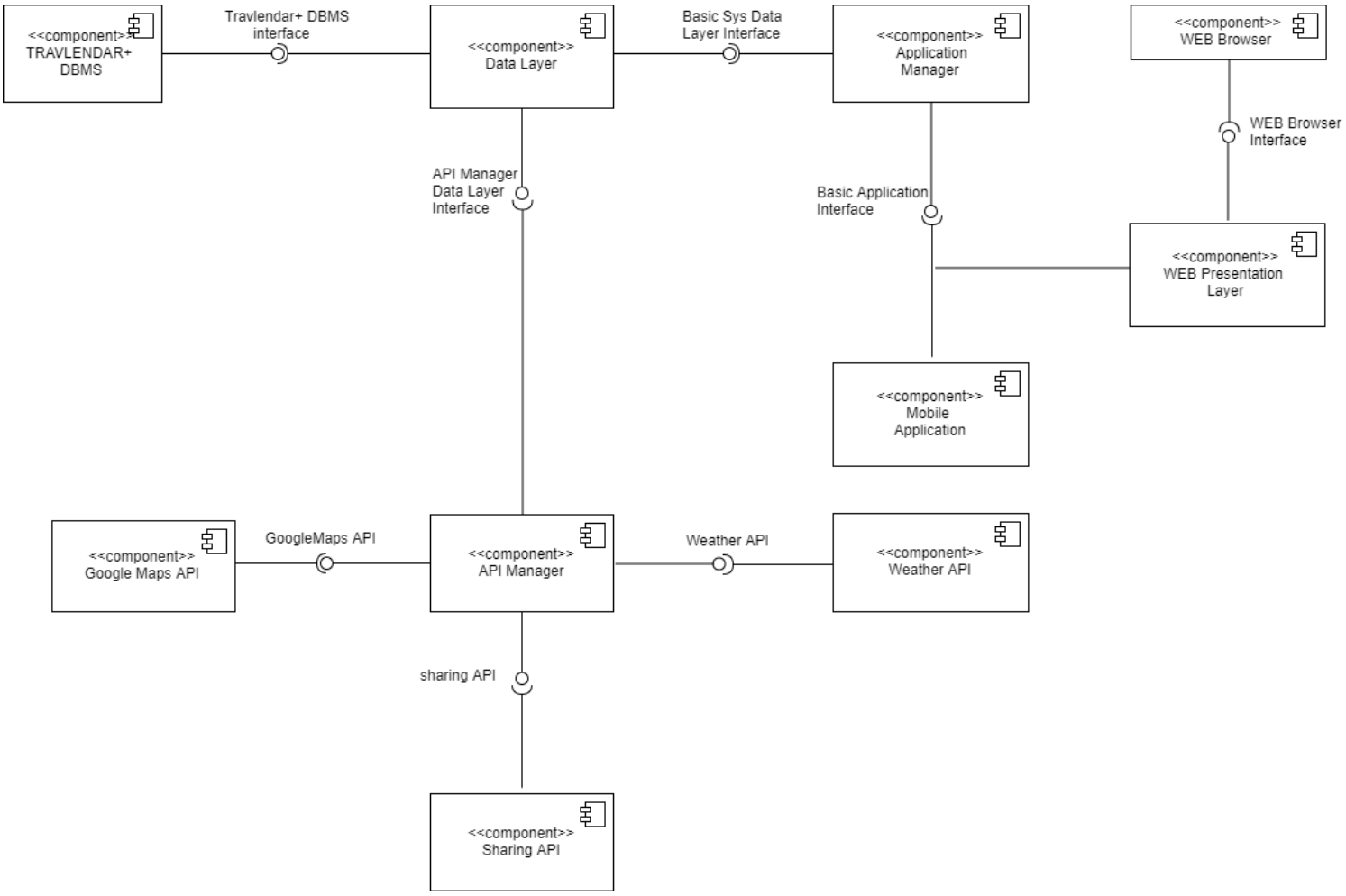
1.5. Document Structure

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 an 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. Devi aggiungere a fine specifica come scegli il data layer!!

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

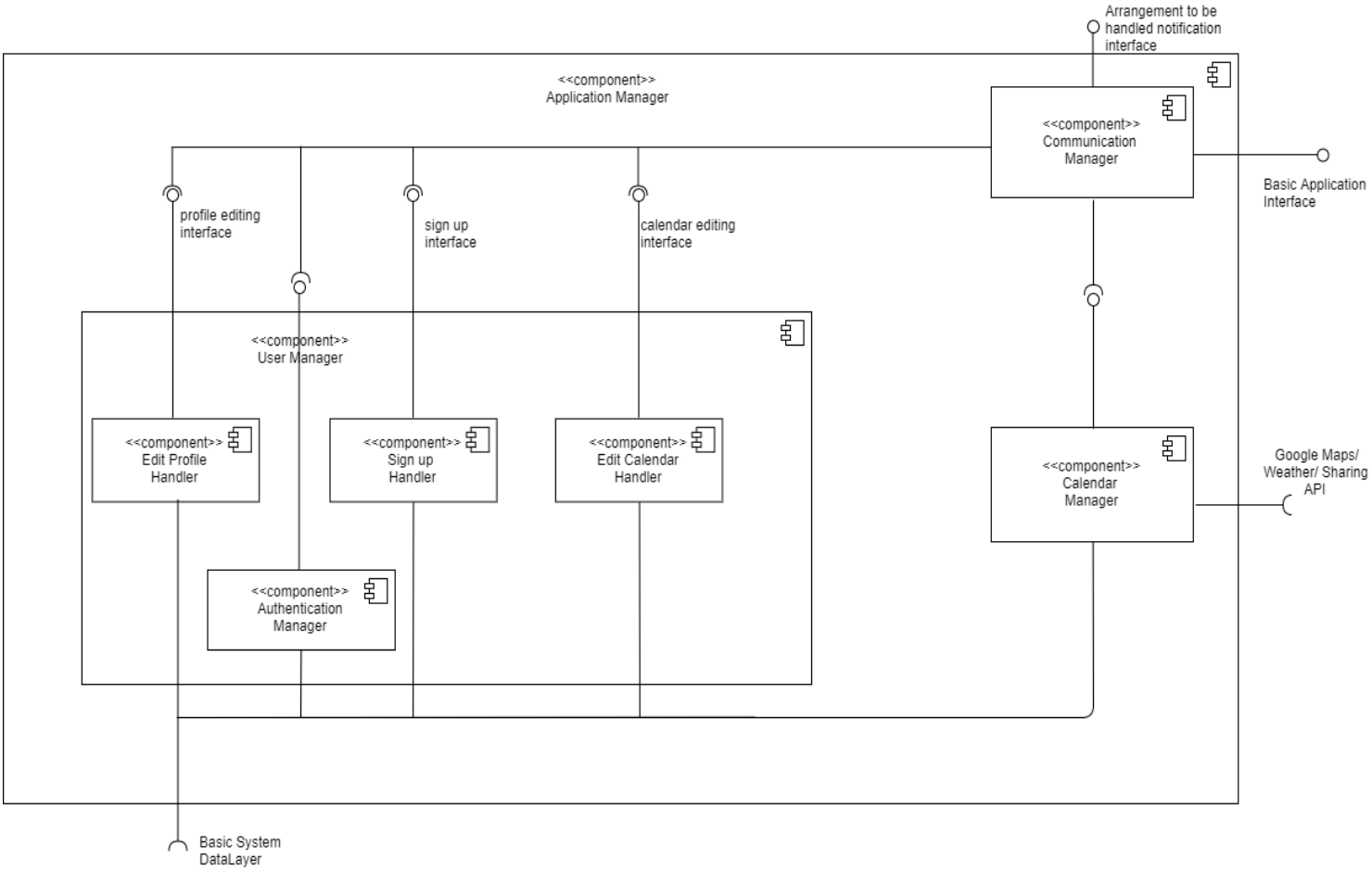
This component of the system is used to handle all the calls coming from web/mobile application. Of course to answer this calls the API Manager has

to use other components which are Data Layer and Google/Sharing/Weather API.

In this sections 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

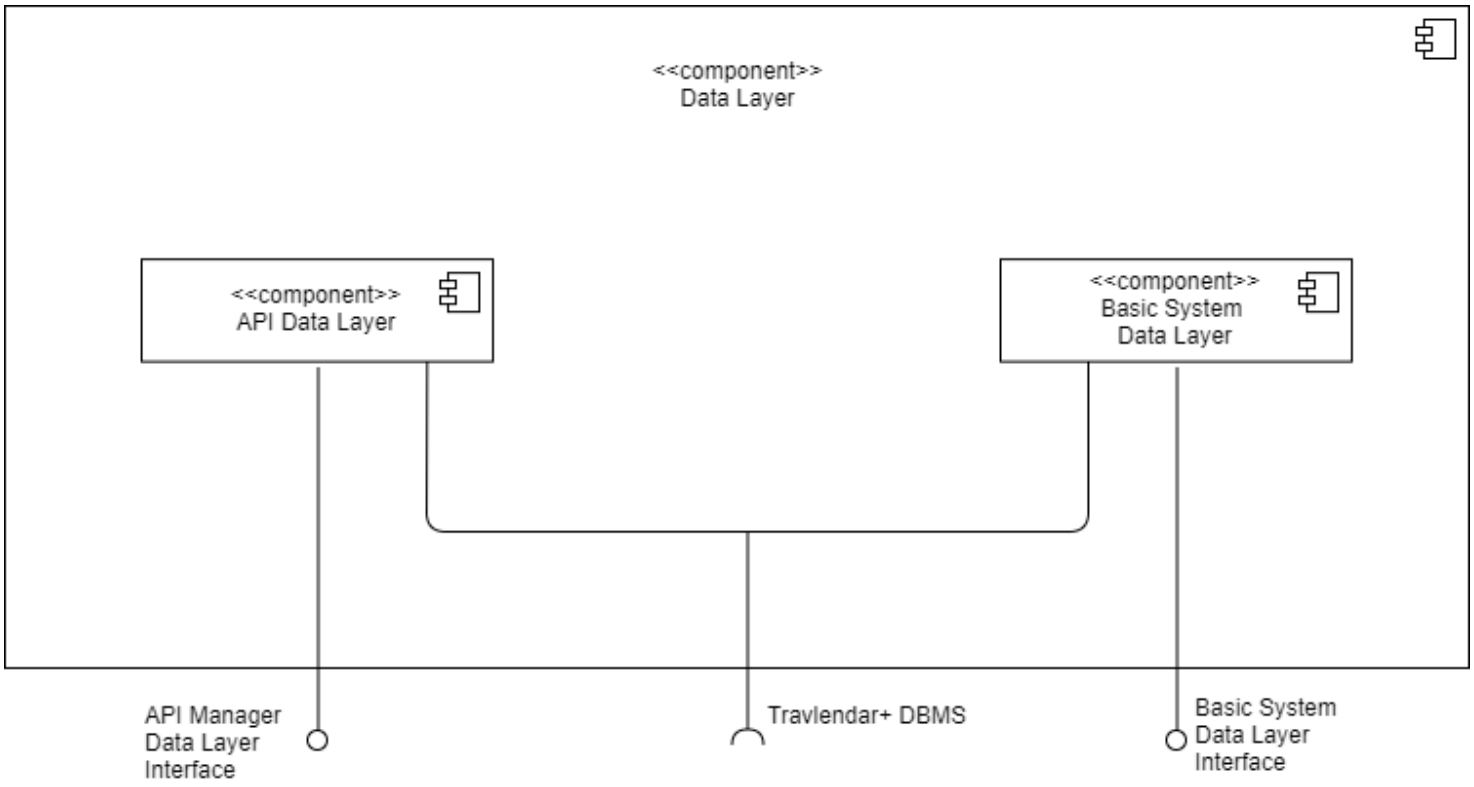
To this component is entrusted the registration of new Travlendar+' users.

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

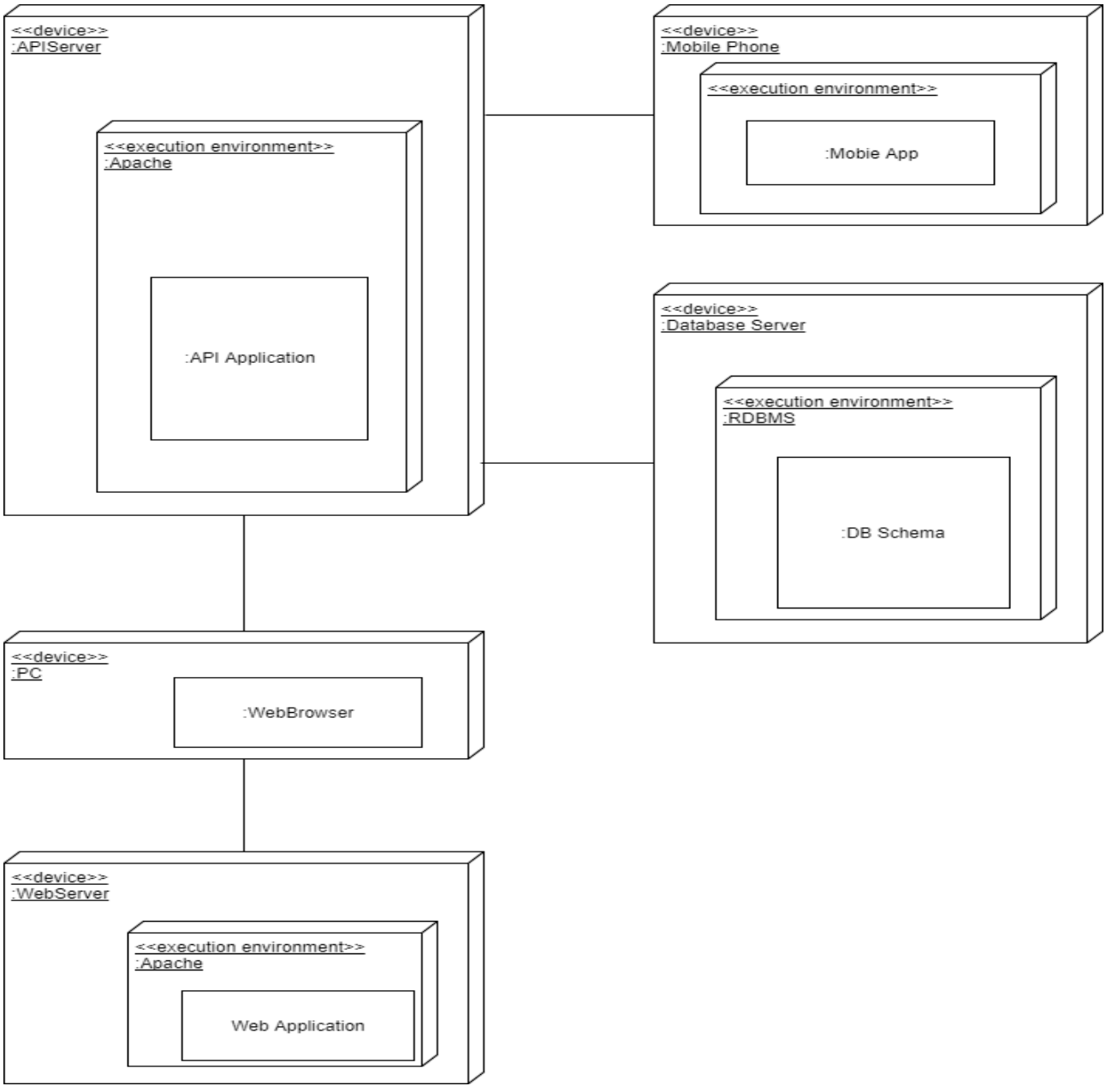
This component has to manage all the request coming from the API manager which cannot perform operations of behalf of a calendar. It also has to translate the Application Manager request in query for DBMS.

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

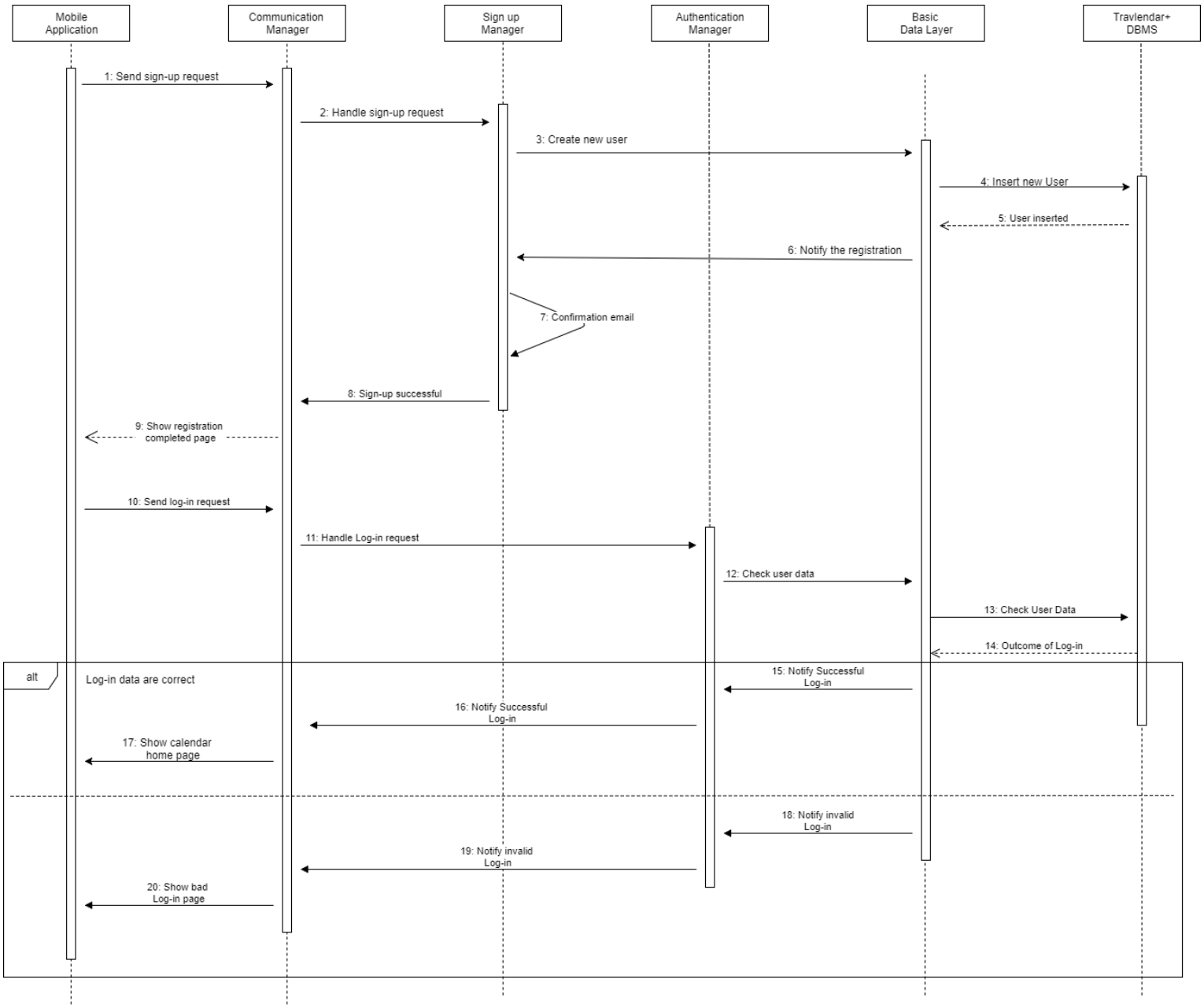
Here I will describe the deployment of the component previously described.



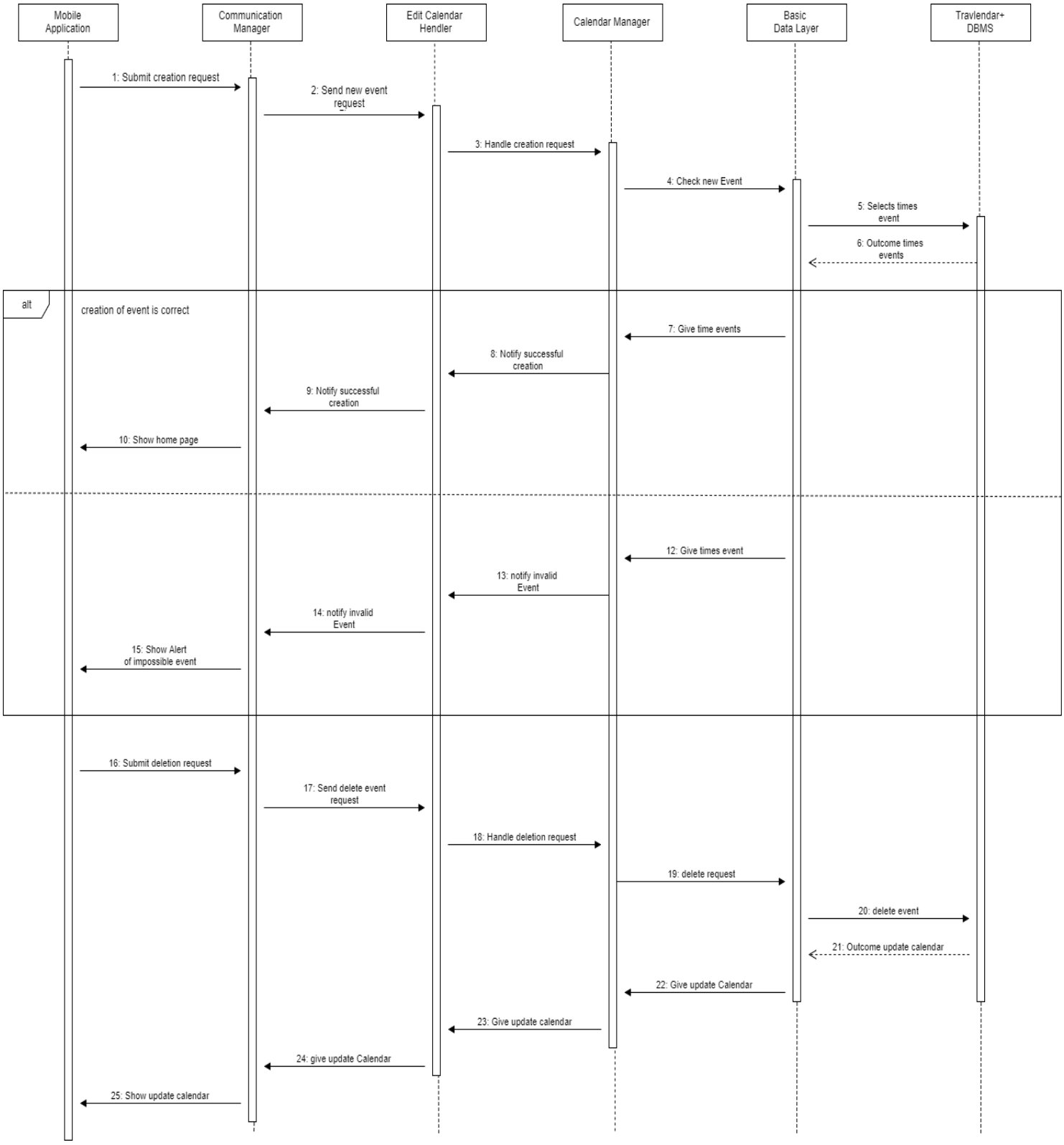
2.4. Runtime view

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

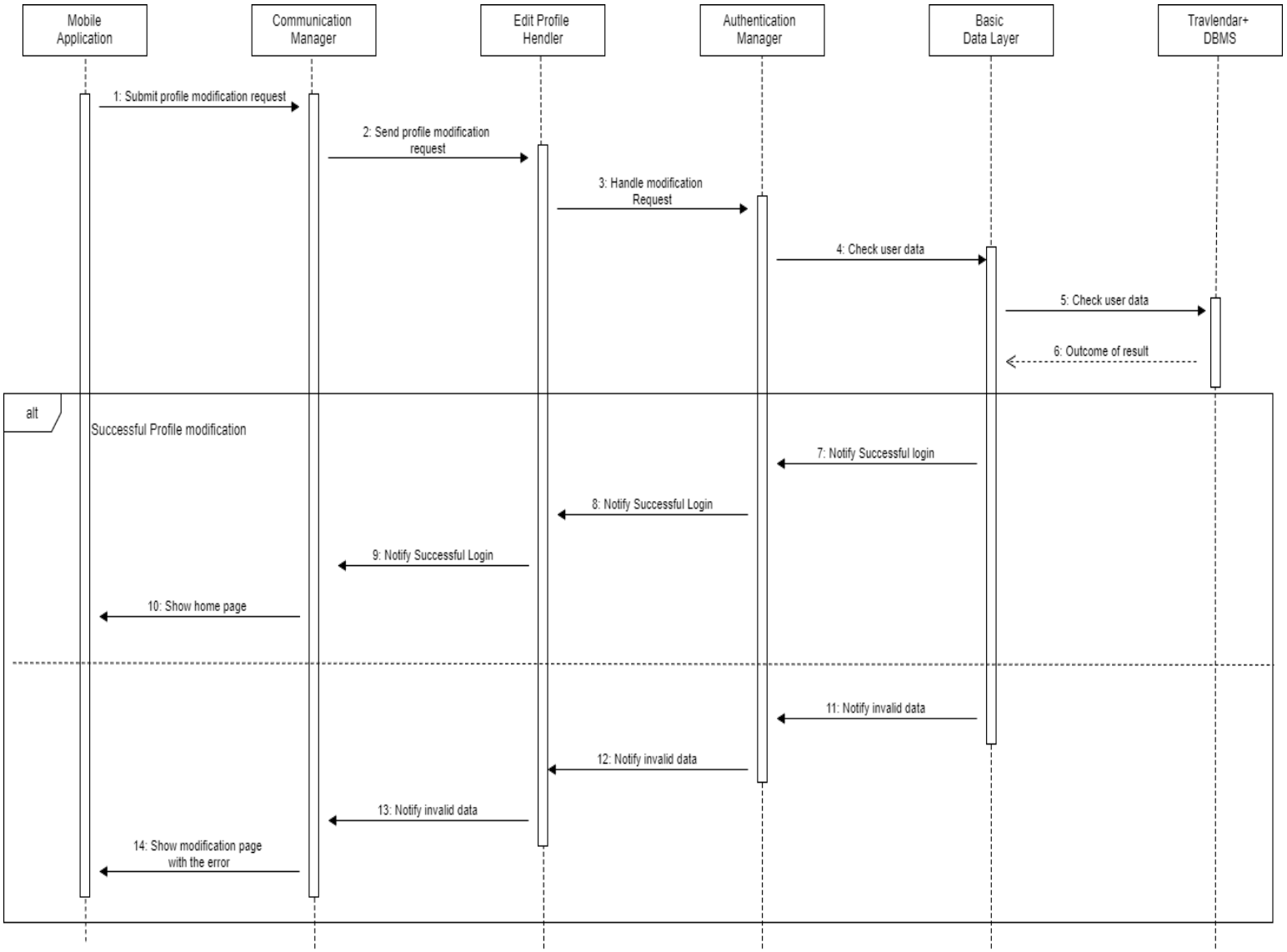
[Rw 1] This runtime view shows components' interaction in a scenario in which someone , through the mobile application, wants to register himself to the service as an user. Subsequently, the diagram shows the scenario of an User Log-in. I include also the messages exchanged in case of failure.



[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 constrains. 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.

MODIFICARE application manager MANCA IL LEGAME CON IL CALENDAR MANAGER HANDLE CALENDAR MANAGER INTERFACE E MANACA AUTHENTICATION INTERFACE levare arrangment

- 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 lay3er 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 thin clients, mobile application and web browser are the view, model logic is the model and part of the controller and the database is part of the controller. 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 king of systems (javaEE).