**Architectural Design**

**High level components and their interaction**

*Database* :

the data layer is responsible for the data storage and retrieval.

It does not implement any application logic. This layer must guarantee ACID properties.

*Application* server :

this layer contains all the application logic of the system.

All the policies, the algorithms and the computation are performed

here. This layer offers a service-oriented interfaces.

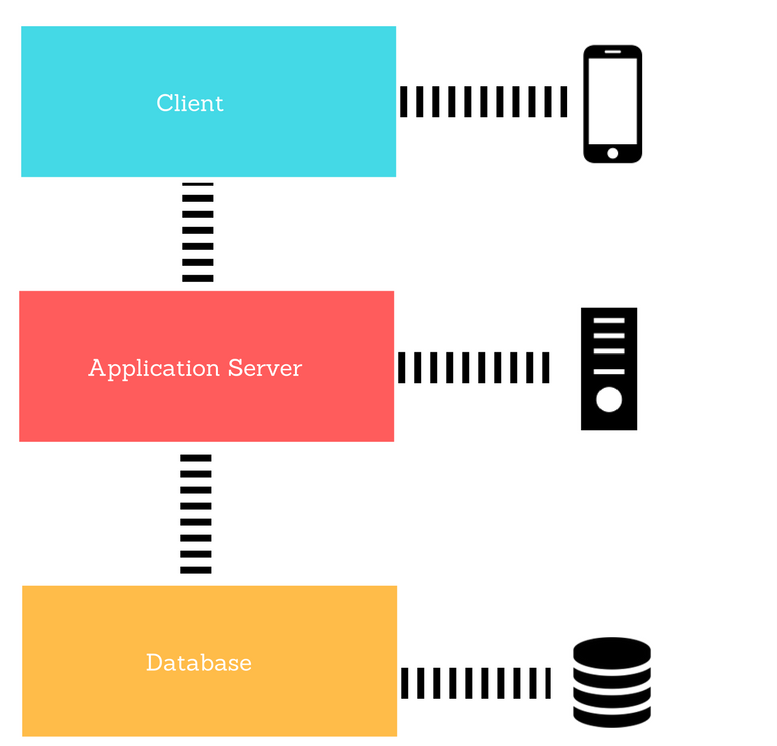
*Client: :*

This layer consists in an Android Mobile application. It’s both presentation layer and logic/client layer, it communicates directly with the application server and it represents the user’s interface.

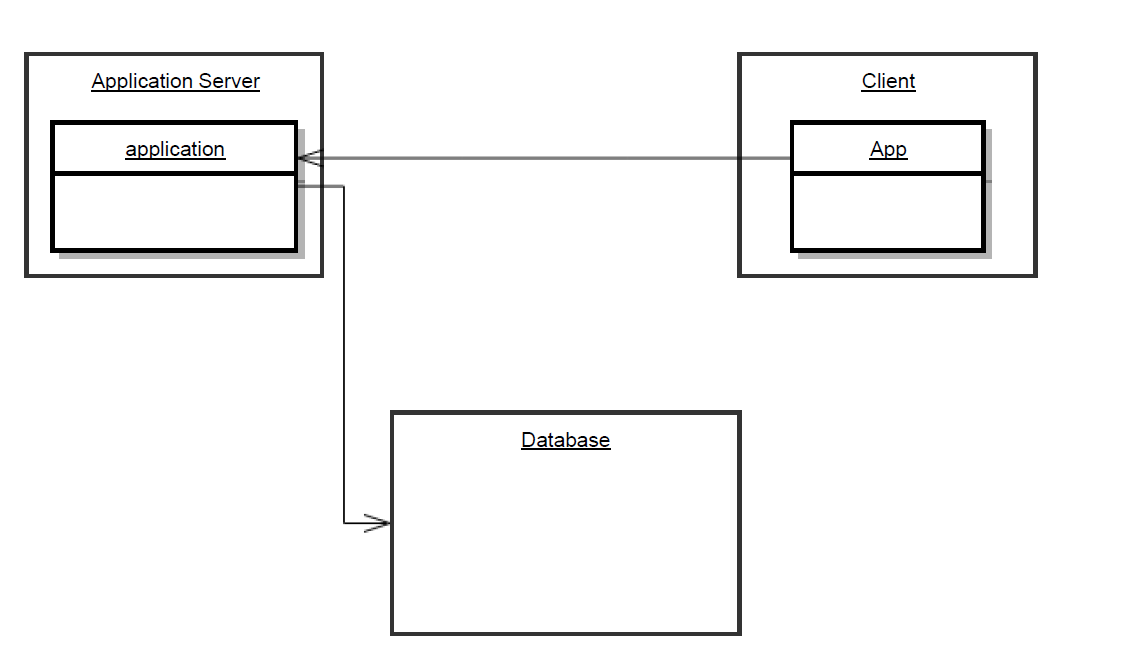
These high-level components are structured into three layers, shown in figure 2.1 .

This choice give us the possibility to compute all the business logic in the Application Server layer, to make more light and efficient to provide a comfortable user experience.

Furthermore this design allows to extend the system, inserting a Web Server layer to consult Travlendar in every device that has a generic browser. In this first release we focus on the Android Mobile client in order to implement the system in the expected times.



*Figure 2.1: Layers of the system.*



*Figure 2.2: High level components of the system.*

**Component View**

The following diagrams describe the main component of the system and the interfaces through which they interact.

The client side is identified with the mobile application. Is a very thin client that allows the user to interact with the application server which contains the main part of the business logic of the system. (AppLocalDb component).

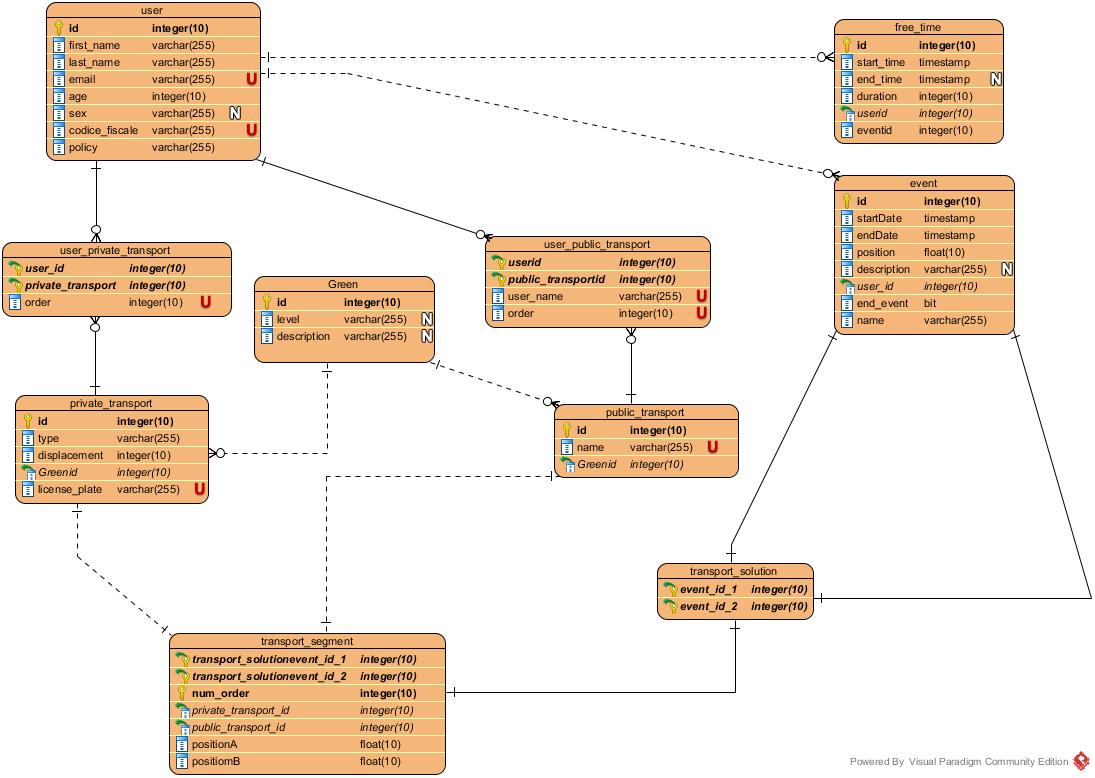
**Server Side Database**

The database tier runs in an external database service that allow us to store data more safely than in an internal Db. We use InnoDB as the database engine: the DBMS has to support transactions and ensure ACID properties.

Access to the data must be granted only to authorized users possessing the

right credentials. Every software component that needs to access the DBMS.

It’ important to remark that the user entity in the following schema is useful for the business tier(for example, we can store here the custom user policies). The user credentials are handled by a external backend service, for high security reasons.

****

*Figure 2.4: The Entity-Relationship diagram of the database schema.*

**Client Side Database**

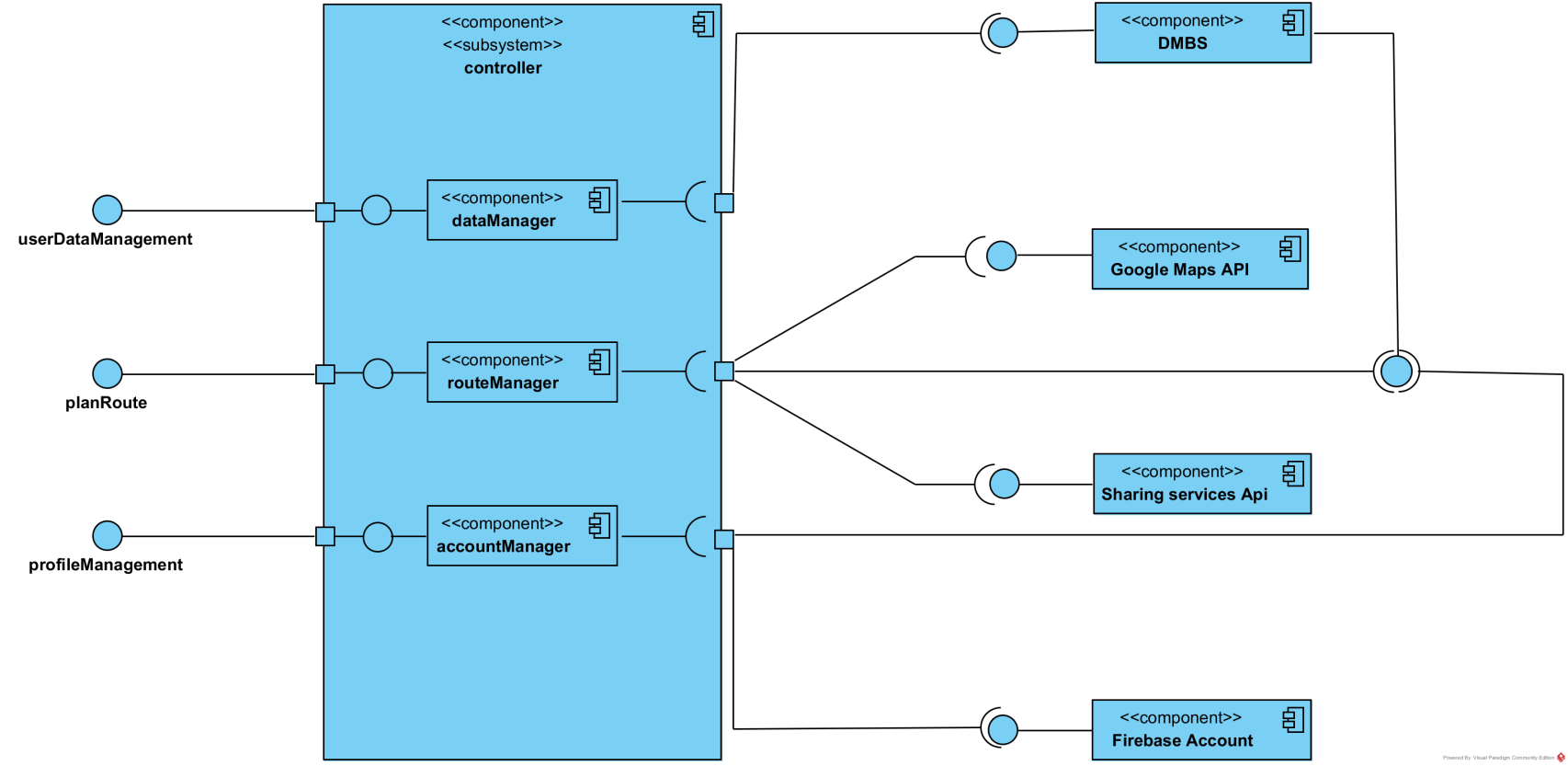
The Client Db schema is composed by a synchronized light copy of the Server side Db.

We chose SQlite dbms because is a best practice implement that on an Android application. The interaction with that is handled by an Android ORM (Object Relational Mapping), in particular we use greeDAO ORM service, like in the picture below.



**Controller projection**

Controller subsystem is composed by three main components: *DataManager, accountManager* and *routemanager.* The first one concerns all the business logic about the management of the data that will be putted/getted into/from the database. The routeManager provides the logic about find the best solutions to link the events of the use with se Google Maps Api and Sharing services Api support . The accountManager has the goal to identify the user by Firebase Api.

******

***Deployment view***

The system architecture is divided in three tiers and it is based.

* The client tier is composed by the mobile application that communicate with the business tier
* The business tier is based on Spring framework because it represent the best practice for these types of system. The details of this framework are discussed in the next page.
* The Database tier is mainly composed by the External Database Server. The communication with the business tier is performed via JDBC connector.

****

***Why Spring framework ?***

We chose to use Spring because is a powerful framework based on Java Enterprise Edition, that simply the devolpment of the server side providing specific functionalities. It’s a framework based on MVC paradigm (Architectural Pattern used in this system as specified in the dedicated section).

Spring comes with some of the existing technologies like ORM framework, logging framework, J2EE and JDK Timers and more, hence we don’t need to integrate explicitly those technologies.

Spring can eliminate the creation of the singleton classes.

Spring framework is both complete and modular, because spring framework has a layered architecture.

Spring framework includes support for managing business objects and exposing their services to the presentation tier components, so that the web and desktop applications can access the same objects.

Spring framework has taken the best practice that have been proven over the years in several applications and formalized as design patterns.

Spring application can be used for the development of different kind of applications, like standalone applications, standalone GUI applications, Web applications and applets as well.

In the following figures shows clearly how the Spring modules are integrated with our designed system.

******