Architecture and Specifications

# Introduction

This document will give the specifications of the architecture proposed to describe it (for each component which appears in the architecture). The architecture can be found on Google Drive.

# API

The API will be the link between the front-side (the user and his application) and the server-side. This API will call the different functions in the different libraries according to the actions of the user on the application or will send information. Then, the arrow between the API and the web applications means that the web applications will use the API to get some information or will use it to trigger an event on the server side (like the booking of a room). That is why this a double arrow (the information can go in both directions).

# Server

The server will be the way to make the application available on internet or in local. Without the server, the application cannot work.

# Libraries

In the server-side, there will be a folder “libraries” with the different libraries which are required to make the application works. Each library will contain some functions for specific goals. There are two arrows which come to this folder because the different libraries will be able to ask the G.C.API[[1]](#footnote-1) or the database to have information. Furthermore, even if the libraries can ask for information or can send information to the database and the G.C.API, the arrow is simple because this is only the library which can call method to make the transfer of information (the G.C.API will not call one of the functions of a library to have or to send information, same thing for the database).

## Booking

The first library will be used to book a room. Then, this library will be able to ask the G.C.API to have the different information which are relevant to book and will make the bookings (store the booking in the database and send the information to the G.C.API). To do this, this library will contain at least:

* A function to verify if the room and each attendee are available for the meeting (by using the G.C.API to obtain the information).
* A function to store the information in the database and to create the event on the Google Calendar through the G.C.API.

## IoT Access

The second will be for the IoT Access (card). Then, this library will be in charge of the connection of a user who wants to use his card to connect on the application. It will also use to associate a new user to a card. To do it, this library will contain at least:

* A function for the connection (scan the card and connect the user according to the card).
* A function to associate a user to a card (scan the card, store the information in database)

## Occupancy, Status and Calendar

The last library will be used to display the calendars (and adapt/change the time scale), the availability/occupancy of a room and will manage be useful to manage the status of a person. Then, this library will use the G.C.API to ask information and will use the database to store information, especially the status of a person. To do all these things, this library will contain at least:

* A function to display the calendar of the user on the home page (will ask the G.C.API to have the information).
* A function to adapt the time scale according to the sparsity of a calendar and to change it according to the selection of the user.
* A function to adapt the status of a user according to his calendar (for example: if a user has a meeting in 30 minutes, the status can change).
* A function to change the status of a user according to his change (if a user change manually his status) and update it in database.
* A function to display the calendar of a room (will ask the G.C.API) and its occupancy (will ask the database)

The possible status: free (green), in meeting (red), yellow (soon in meeting). The status can be adapted automatically, especially with the G.C.API.

# Google Calendar API

The G.C.API appears in the architecture as it will be useful to get some information about the calendar of the user or of a room (which is useful to display a calendar or to update the status), or to create events on the Google Calendar when a user will make a booking (and if the booking is successful).

# Database and Make the connection to database

To make this project, it will be necessary to have a database to store some information. Then, the database will at least contain the three following tables:

* A table to store the association between a card number and a user.
* A table to store information about the user (especially for the status)
* A table to store the reservation which are made, to make data analysis on it in a next version of the application.

There is also “Make the connection to database” which will be here to manage the connection to the database. Then, there is an arrow which comes from the database to “Make the connection to database” as the request to the database will pass through this file, but the database will not call a function in this file.

# Web Application

While the precedent components are server-side components, the web application means the front-side and all files which are required to make it works. As the product will be a web application, it is necessary to have HTML files (to have the different pages), CSS files (to make it more esthetical for the user) and JavaScript files to make it dynamic and interact with the server-side (here the API, according to the architecture).

# Google Authentication

This block is just here for the connection with the account. If the user choose to connect with his account, he will enters his logs to connect on the application, and Google will allow the connection or not.

1. G.C.API: Google Calendar API [↑](#footnote-ref-1)