TrackMe project Julián Cuéllar Mangut
Javier Fernández Rodríguez

Design Document

Deliverable:	DD
Title:	Design Document
Authors:	Julián Cuéllar Mangut, Javier Fernández Rodríguez
Version:	1.0
Date:	10-December-2018
Download page:	https://github.com/javferrod/CuellarFernandez
Copyright:	Copyright © 2018, Julián Cuéllar, Javier Fernández

Revision history:		
	V 1.0:	First version of the document.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

The aim of this document is to determine in a more detailed way which software requirements are going to be used for the development of the project, it is also expected that this document will serve as a model to follow for the development of the application.

The objectives that the project is expected to meet and the different services provided by TrackMe can be read in the **RASD** project document.

1.2 Scope

The expected scope of the system were presented in the **RASD** document.

The list of goals will be re-submitted for discussion throughout the document.

ID	Goal
GL1	The system should provide accounting and authorisation for users and clients.
GL2	The system should store the recollected data.
GL3	The system should recollect the data using the sensors available in the users' devices, asking the user directly the information when no sensor is available for recollecting the information (for example, weight).
GL4	The system should recollect the data from the users at time intervals.
GL5	The system should store and display the data in a time series format, allowing the client to consult the changes in the parameters along the time.
GL6	The system should allow the clients to easily query the already recollected data of the users.
GL7	The system should allow the clients to query the data of an specific user.
GL8	The system should allow the clients to subscribe to a query, providing new data as arrives.
GL9	The system should protect the privacy of the users. A data batch displayed to a client should not enable the differentiation between individuals.
GL10	The system should allow users to monitor some of their parameters, alerting the emergency system when any of these parameter gets out of a threshold.

Table 1: Goals

1.3 Definitions, acronyms, abbreviations

1.3.1 Definitions

- **Android Studio:** Official development environment for Android developed by Google.
- **Container:** Standard unit of software that packages up code and all its dependencies.
- **Docker:** Software developed by Docker Inc. Provides operating-system-level vitalisation also knows as containerisation.
- **InfluxDB:** Open-source time series database.
- **Koa.js:** Minimalist web framework from the creators of Express.

- **Kubernetes:** Open-source system for automating deployment, scaling, and management of containerized applications.
- **Mockups:** Models of device interfaces.
- **MongoDB:** Open-source document-oriented database.
- **Orchestration:** Automated arrangement and coordination of computer systems and services.

1.3.2 Acronyms

- **API:** Application Programming Interface.
- **HTTP:** Hypertext Transfer Protocol.
- **HTTPS:** Hypertext Transfer Protocol Secure.
- **IP:** Internet Protocol.
- **XML:** Extensible Markup Language.

1.3.3 Abbreviations

1.4 Revision history

The revision history can be find on page ► Ref a pagina 2 ◄.

1.5 Reference documents

References used during the development of this document can be found at the bottom of the document on the page 27.

1.6 Document structure

The structure of this document is given in the table of contents (Page 4) but in this section we will take a closer look at everything contained in the document.

1. **INTRODUCTION**

   In the first section we will deal with the introduction. As in the RASD, the document that is being presented will be presented in an incoming form with references, an introduction to the objectives that the project is expected to achieve, definitions, abbreviations, and so on.

   You can start reading this section on page 7.

2. **ARCHITECTURAL DESIGN**

   The second section of the document presents the design of the architecture that will be followed throughout the development, this part is very important because it is the one that presents in a detailed way all the functioning that is behind and the different connections that are made.

   We present different schemes and designs from an external point to how the different parts of the systems interact with each other and what they use.

   You can start reading this section on page 10.

3. **USER INTERFACE DESIGN**

   The third section presents the designs of the user interface, at this point we will not go into depth since the designs and explanations have been given in the RASD.

   You can start reading this section on page 21.

4. **REQUIREMENTS TRACEABILITY**

   The fourth section presents the objectives and requirements of the service which were presented in the RASD and how these are resolved through the architectures presented in this document.

   You can start this reading this section on page 22.

5. **IMPLEMENTATION, INTEGRATION AND TEST PLAN**

   The last section deals with how has been made the implementation of the different parts that make up the service and what elements (frameworks, services, etc.) have been used during its implementation.

   It is also treated as all these elements have been integrated into the system and testing plan has been used to verify the proper functioning of all parts and service.

   You can start reading this section on page 24.

2 Architectural Design

2.1 Overview

An overview of the system is shown in figure 1. The dashed lines represents connections between elements of the TrackMe system while the solid ones corresponds to outside ones.

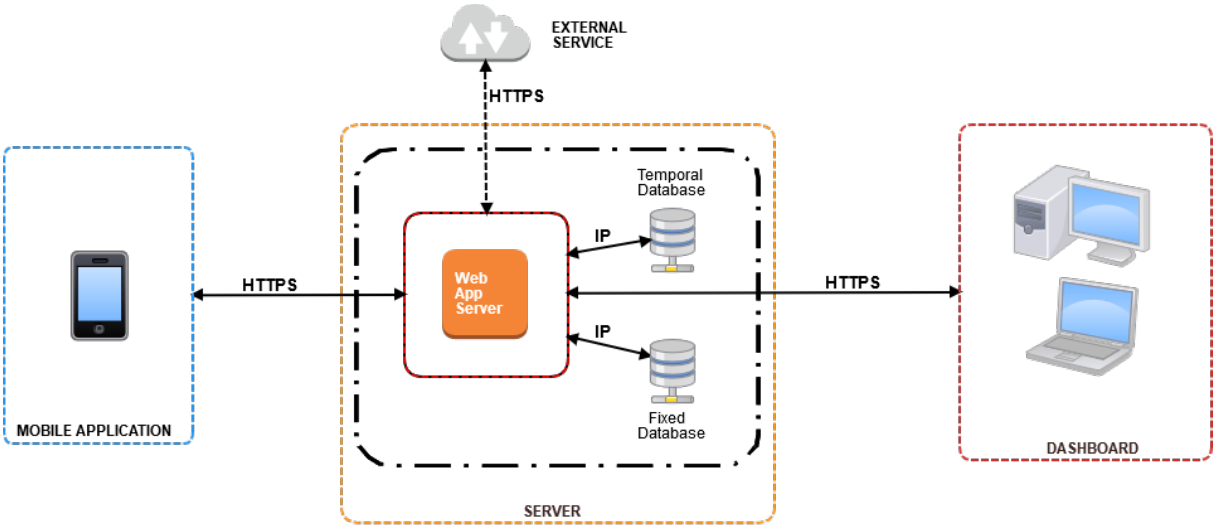


Figure 1: Overview of the system

The system have three main blocks: the mobile application, the server and the dashboard. HTTPS is used to interconnect them. There are also external services, the emergency system and the payment system. The connection with them is realised using also HTTPS.

- **Mobile:** Executed in user's smartphones. Recollects the data at fixed intervals and summit it to the server.
  - **Smartwatch:** Executed in the user's smartwatch. Recollects 8 when needed and send it over Bluetooth to the mobile.
- **Server:** The most complex component. Contains two databases and the API itself. Handle the data submitted by the users and stores it. Replies to the clients queries and is in charge of authorising users and clients.
- **Dashboard:** Entirely executed in the client's web browser. It is centred in providing a visual interface to the clients.

Globally, the architecture follows the classical client-server style, the paradigms used are discussed more profoundly in section 2.6.

Worth mentioning that the API offered to the clients is a subset of the API used by the dashboard, therefore no special effort is needed in that direction.

2.2 Component view

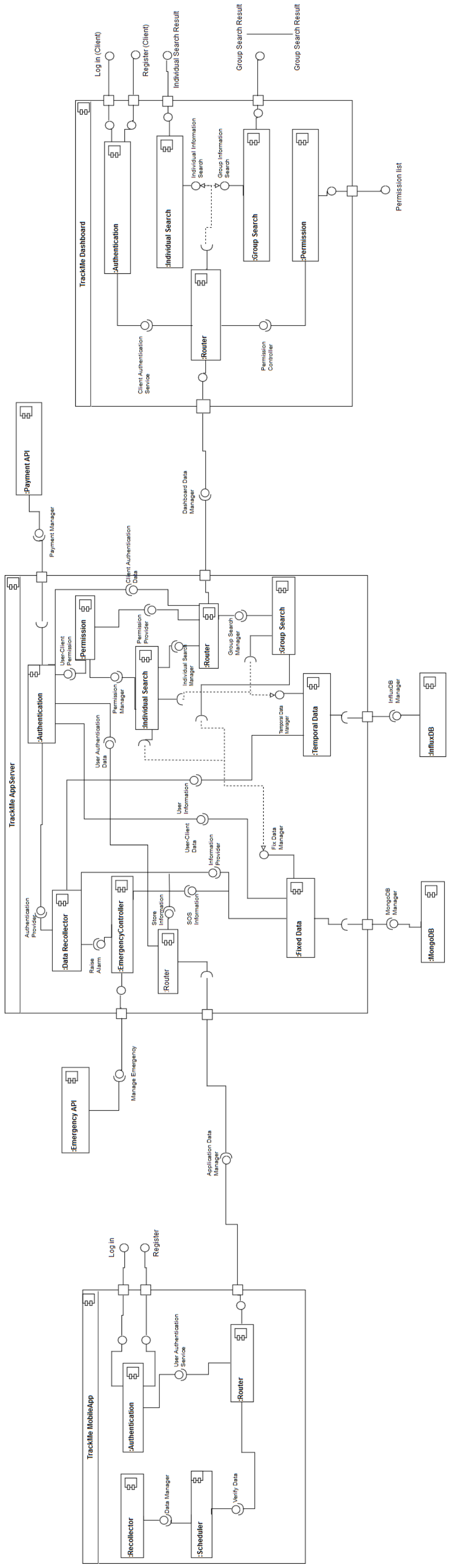


Figure 2: Deployment of the mobile side components

2.3 Deployment view

Figure 3 shows the deployment of the server side components. The use of Docker introduces an isolation layer between the Operating system and the source code, which enables portability across all operating systems that supports Docker.

Furthermore, the inclusion of Docker facilitates a future deployment in which the containers are distributed. Kubernetes can handle the orchestration while Docker will tackle the containerisation.

The connection between containers is specified in a Docker Compose file, which will provide IP connectivity between the different containers as shown in the figure 3.

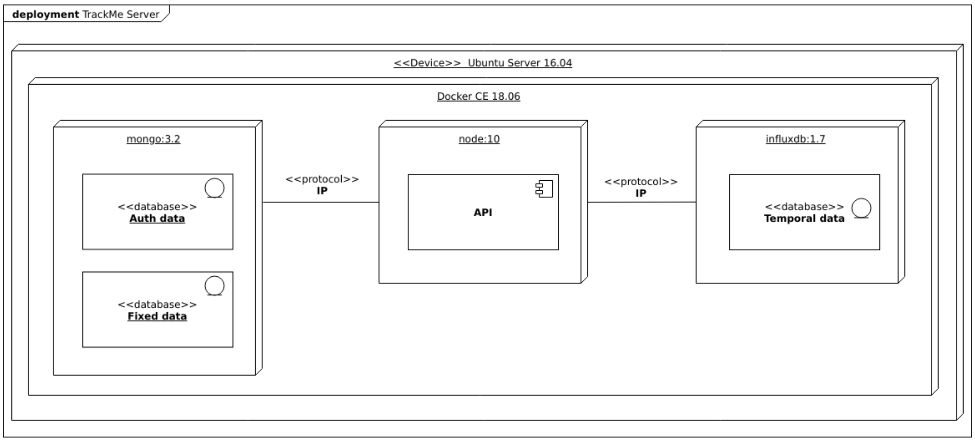


Figure 3: Deployment of the server side components

Figures 4 and 5 shows the deployment of mobile and dashboard respectively. The diagrams are rather simpler, the deployment of mobile and dashboard do not require any special requirements.

In the case of the mobile, the most relevant information is contained in the version of Android required.

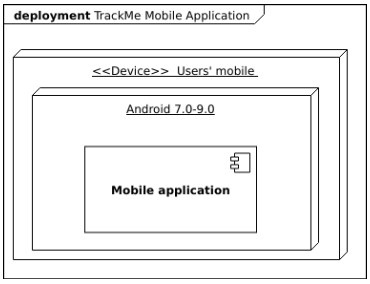


Figure 4: Deployment of the mobile side components

The dashboard will be written in React, and therefore the JavaScript code will be downloaded by the client to be executed in the web browser JavaScript interpreter.

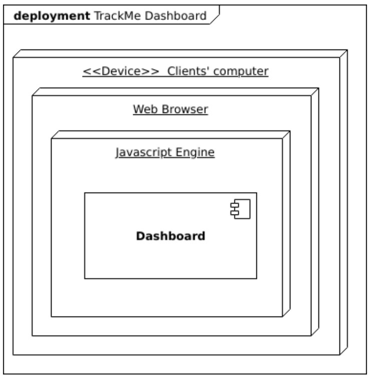


Figure 5: Deployment of the dashboard components

2.4 Runtime view

Along this section, almost all operations performed by users or clients are explained in terms of calls between the different actors of the system. The following diagrams are high level ones and therefore do not include any component, although the interactions that takes places between components are easily extrapolated thanks to the names of the components.

Only chains of correct operations are shown in the diagrams, error handling is out of the scope of this section.

The calls between the NodeJS server and the databases do not represent a real query, functions are used instead of real queries to represent the intention of the call rather than the semantics of each database.

Figure 6 represents the download of the dashboard in the cients' web browser. An NginX frontend is used to deliver the static files needed to render the dashboard as NginX excels in this tasks [1].

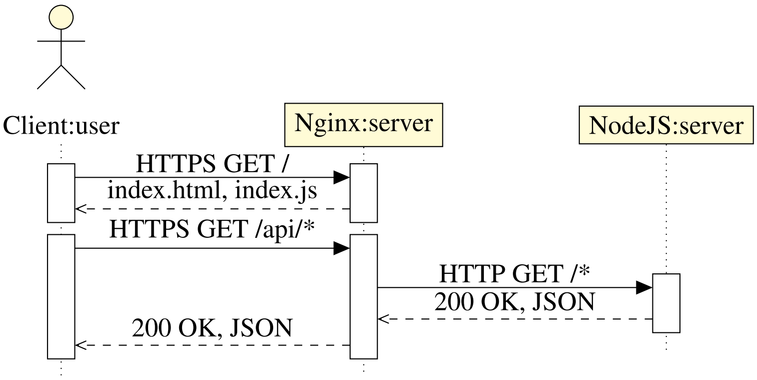


Figure 6: Download of the dashboard

Figure 7 shows the login process by a client or a user. The final target of the client or user is to obtain a token that is used to authenticate him in future calls.

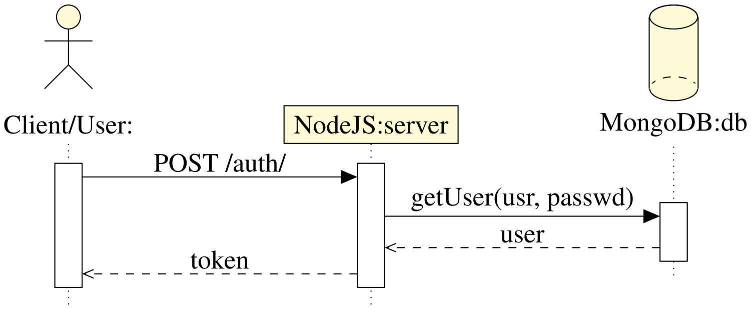


Figure 7: Login by a Client or User

In figure 8 there are two calls to a database. The first one takes place between the fixed database and the server whilst the second one is performed against the temporal database. The former represents the authentication process and is performed by Authentication component. This call is included in almost every process.

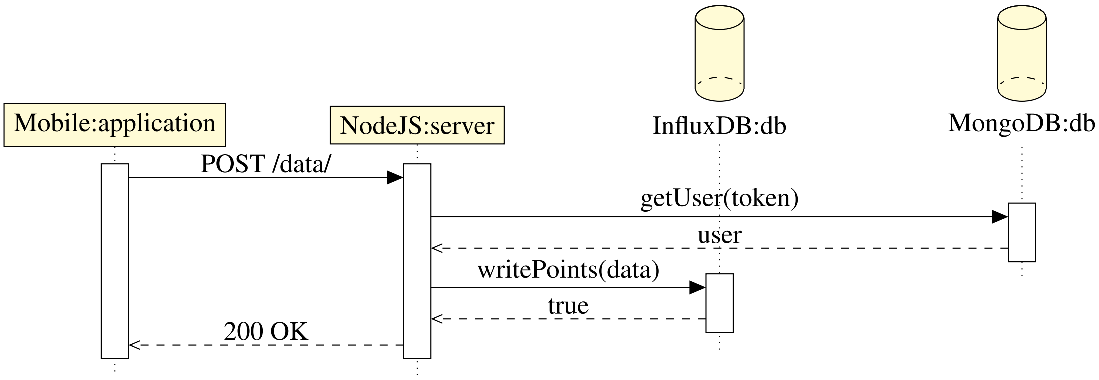


Figure 8: Data sent from the mobile application

In figure 9, the most relevant is the double search performed in the temporal and fixed databases. Worth mentioning that if no results are obtained in the first call, the second one is not conducted.

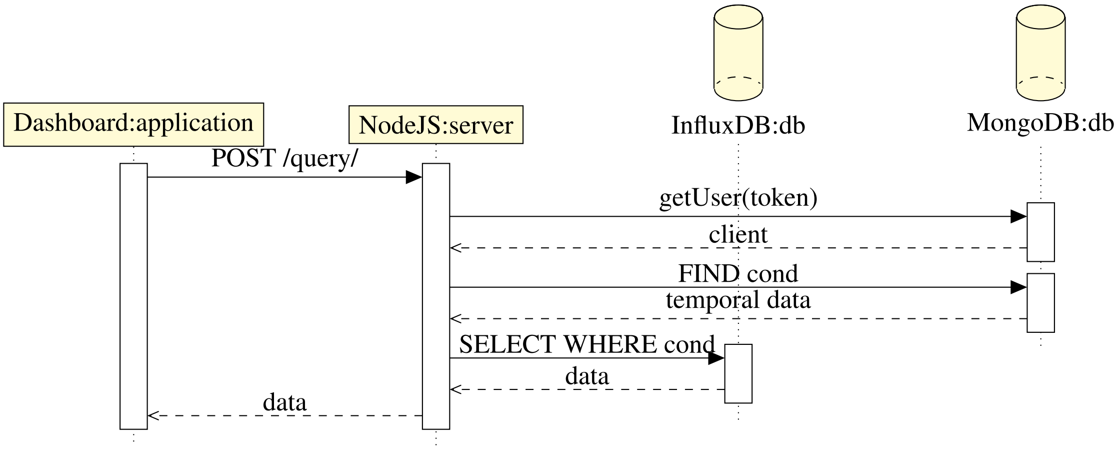


Figure 9: Query performed by a client

Figure 10 shows a individual search. The management of permissions is reduced to check if the user is included in the list of permissions of the client.

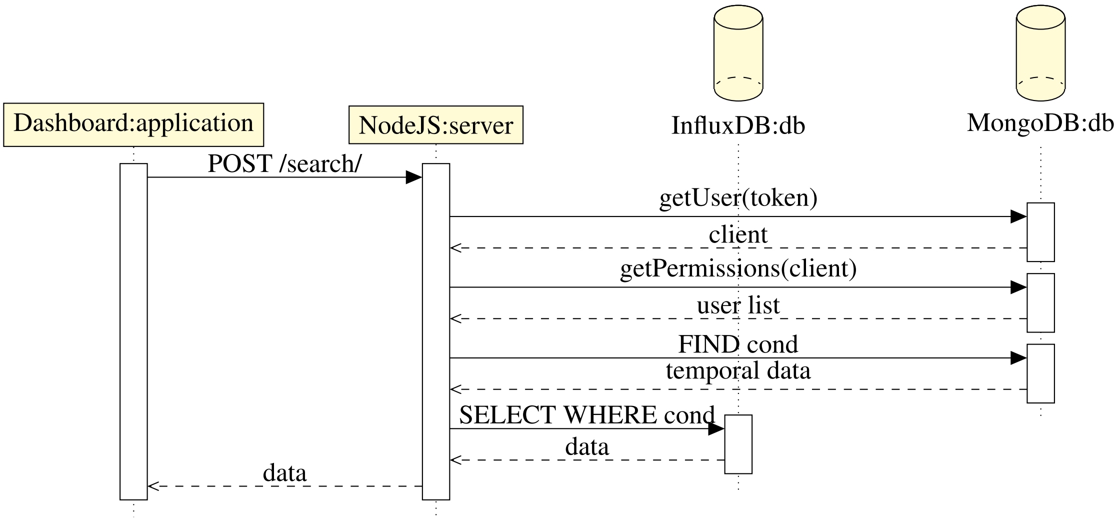


Figure 10: Individual search performed by a client

Figures 11 and 12 represents the process of asking and granting permission.

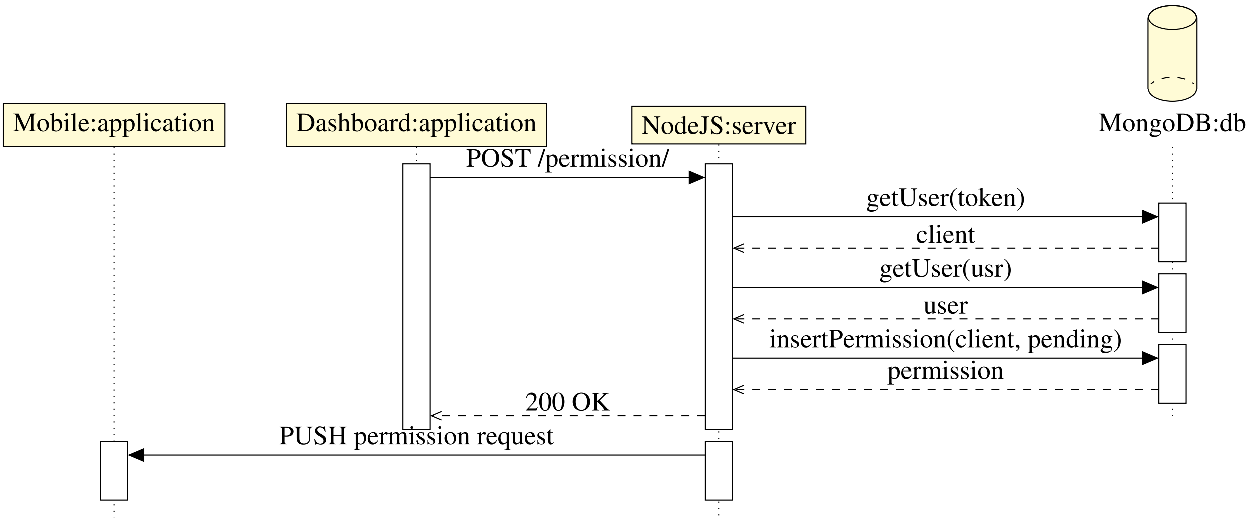


Figure 11: Permission request performed by a client

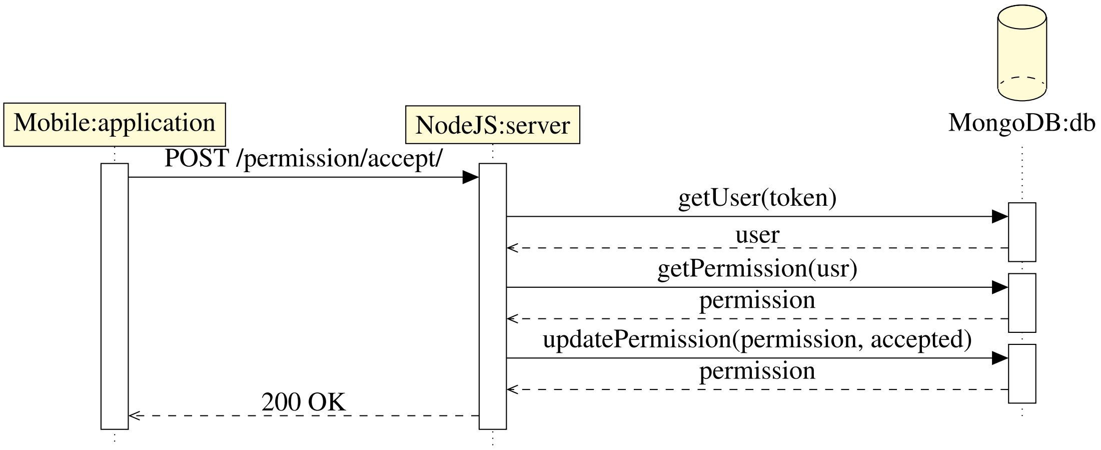


Figure 12: Permission request accepted by an user

2.5 Component interfaces

2.5.1 Server

Name	Receives	type	Description
login	username/clientname, password	External	Returns a token which can be used to identify the user/client in the future.
registerUser	username, password, fixed parameters	External	Creates a user with the given username, password and fixed parameters
registerClient	clientname, password, credit card information	External	Creates a client with the given clientname and password. Interacts with Stripe to start the charge to the given credit card
getUser	token	Internal	Returns all the user/client associated to the given token

Table 2: Interfaces of Authentication component

Name	Receives	Type	Description
sendData	user's token, parameters from user	External	Receives the data from the user, sanitise and stores it.

Table 3: Interfaces of Data Recollector component

Name	Receives	type	Description
query	client token, query	External	Returns all the users that fulfils the query if the query can be anonymised.

Table 4: Interfaces of GroupSearch component

Name	Receives	type	Description
search	client token, codice fiscale	External	Returns the user associated to the given codice fiscale if the clients have the needed permission.

Table 5: Interfaces of IndividualSearch component

Name	Receives	type	Description
request	client token, codice fiscale	External	Raise a request of permission to the user associated to the given codice fiscale.
accept	user token, permission ID	External	Sets the given permission as accepted.

Table 6: Interfaces of Permission component

Name	Receives	type	Description
raise	user location, user name, parameter	Internal	Raise a emergency call to the external emergency system.

Table 7: Interfaces of EmergencyController component

Name	Receives	type	Description
saveTemporalParameter	userID, parameter	Internal	Stores the parameter and its value in InfluxDB database
temporalSearch	userID, query	Internal	Returns all the temporal parameters that fulfils the query from the user indicated by userID
temporalSearch	query	Internal	Returns all the temporal parameters with their corresponding userID that fulfils the query

Table 8: Interfaces of TemporalData component

Name	Receives	type	Description
createUser	username, password, fixed parameters	Internal	Creates a user with the given username, password and fixed parameters
fixedSearch	userID, query	Internal	Returns all the users that fulfils the query
getUser	username/clientname	Internal	Returns the user/client that match the given username
getPermissions	clientname	Internal	Returns the list of users for which the given client have permissions.
savePermission	username, clientname, status	Internal	Creates a permission if not exists and sets its status to the given one.

Table 9: Interfaces of FixedData component

2.5.2 Dashboard interfaces

The dashboard only present interfaces to the user, which are already defined in the RASD.

2.5.3 Mobile application

Name	Receives	type	Description
getLocation	latitude, longitude, hour	Internal	Obtains the location where the user is at that moment.
getHearthRate	hearth rate	Internal	Gets the user's heart rate at that time.

Table 10: Interfaces of Recollector component

Name	Receives	type	Description
getUserData	latitude, longitude, hour, hearth rate	Internal	Receives the information obtained from the user

Table 11: Interfaces of Scheduler component

2.6 Selected architectural styles and patterns

The general architecture will be discussed first. Later on, the styles and patterns used in each part of the system will be introduced.

2.6.1 General architecture

As shown in figure 1, the system follows the client-server paradigm. There are two *clients*[1] in the platform, the Android application used by users and the Dashboard used by the clients. The server is composed by an application which responds to the requests of the *clients* and two servers which store the data.

Client-server architecture is a widespread style, used in almost all the mobile applications which requires interaction between systems. Moreover, the usage of applications based entirely in the browser, and therefore client-server, is common.

2.6.2 Android application

The android application is presented in Android Studio, a tool with a lot of potential that allows to develop it in the best possible way. For this we use two programming languages that are JAVA and XML.

JAVA and XML combine very well providing each other the needs that the other presents and giving a final set with a lot of potential.

2.6.3 Server application

The server can be divided in two parts, the API and the databases. The former is written in Koa.js whilst the databases are MongoDB for the fixed data and InfluxDB for the temporal one.

The API centres around endpoints and the corresponding handlers. Therefore, the logic will be arranged in five main blocks: Authentication, Data Recollector, Group Search, IndividualSearch and Permissions. Each of these blocks contains the endpoints, handlers and helpers needed to offer the service.

MongoDB and InfluxDB are NoSQL databases, hence there is not need of any configuration of the databases prior to insert data into them.

Functional programming is embraced when possible. Although JavaScript in its latest versions includes several ideas from this paradigm, libraries like ramda are used when needed.

[1] The italic is used to differentiate the *client* in client-server paradigm from the clients of TrackMe platform.

2.6.4 Dashboard application

The dashboard is written in React, and therefore the component-based and declarative styles are embraced. To maintain and distribute the state across the React application, Redux paradigm is used.

Redux plays well with React providing a single source of truth for the entire application which unleash the declarativeness of React.

As in the server application, the functional programming style is adopted. This paradigm avoids to mutate data, which becomes useful when dealing with React as eases the detection of changes in the state.

2.6.5 Protocols

The protocol to be used for the connections between the blocks is the HTTPS protocol.

This protocol is based on the HTTP protocol and allows us to use an encrypted channel to transmit data securely providing the necessary security in the transfer of information between the different blocks that make up the service.

The internal connections between the web application and the databases will be made using the IP protocol.

This protocol is used because it is a protocol of a more internal level than the HTTPS and that allows the passage of data packets between units.

2.7 Other design decisions

2.7.1 Chosen databases

TrackMe service is entirely dedicated to gather data and store it. Databases are essentials in almost every application, even more in the project at hand.

As mention earlier, the data is structured into parameters, which can be classified in two main groups: fixed and temporal. These two groups have different needs in terms of treatment and storage[2]. Therefore, different databases are needed.

Table 12 summarises the reasons why MongoDB and InfluxDB have been chosen as databases.

Parameter type	Requirements of the data	Database	Advantages
Temporal	Retention policy Timestamps	InfluxDB	Fulfils requirements Driver for NodeJS Distributable
Fixed	-	MongoDB	Flexible Driver for NodeJS Easly distributable

Table 12: Data types, requirements and databases

[2] For example, a retention policy is needed to store only 3 months of temporal data (PR**??**)

The bottom line is that two databases are needed, a conventional database and a time-series based one. There are plenty of options that would fulfil the requirements, the main reason to choose MongoDB and InfluxDB is the expertise that the developer team have with them.

2.7.2 Databases design

This section states the chosen structure for the data stored in MongoDB and InfluxDB.

InfluxDB imposes a rigid structure for the data[3]. The structure per each parameter is shown in figure 13, the tag field is used to link the temporal data with the fixed one.

```
{
    "measurement": parameter,
    "tags": { "user": username },
    "fields": { "value": value },
}
```

Figure 13: Structure of one parameter in InfluxDB

MongoDB is more flexible, and allows any kind of structure as long as it is JSON. The selected structure is stated in figure 14.

```
{
    permissions: [{
        "client": userID
        "user": userID
    }],
    users: [{
            "id": uniqueID
            "type": user/client,
            "auth": {
                    "username": username,
                    "password": digest(password)
                    },
           ... fixed parameters
    }]
}
```

Figure 14: Structure of one parameter in InfluxDB

[3] called measures in InfluxDB

3 User Interface Design

All user interfaces (mobile application, web application and smartwatch) were presented in the RASD. There is no need to provide more.

4 Requirements Traceability

In the following points we will treat that objectives (those presented in the RASD that have also been presented in the Scope of this document) that have been fulfilled through the implementations treated in this document.

► En el documento del DD pone explicitamente mapear requisitos, si se quiere se puede añadir mas mapeos pero requisitos seguro que si ◄

Use case ID	Run time diagram
UC2 UC4	Figure 7
UC1 UC3	Figure 7
UC5	Figure 10
UC6	Figure 9

Table 13: Correspondence between use cases and run time diagrams

Requirement ID	Component name
FR1	

Table 14: Correspondence between requirements and components

Interface ID	Component	Table
SI6	Authentication	Table 2
SI7	GroupSearch	Table 4
SI8	IndividualSearch	Table 5

Table 15: Correspondence between interfaces and component interfaces

► Si mapeamos los requisitos, pienso que los goals sobran ya que cada requisito va a asociado a un goal ◄

- **GL1:** The system should provide accounting and authorisation for users and clients.

  1.

- **GL2:** The system should store the recollected data.

  1.

- **GL3:** The system should recollect the data using the sensors available in the users' devices, asking the user directly the information when no sensor is available for recollecting the information (for example, weight).

  1.

- **GL4:** The system should recollect the data from the users at time intervals.

  1.

- **GL5:** The system should store and display the data in a time series format, allowing the client to consult the changes in the parameters along the time.

  1.

- **GL6:** The system should allow the clients to easily query the already recollected data of the users.

  1.

- **GL7:** The system should allow the clients to query the data of an specific user.

  1.

- **GL8:** The system should allow the clients to subscribe to a query, providing new data as arrives.

  1.

- **GL9:** The system should protect the privacy of the users. A data batch displayed to a client should not enable the differentiation between individuals.

  1.

- **GL10:** The system should allow users to monitor some of their parameters, alerting the emergency system when any of these parameter gets out of a threshold.

  1.

5 Implementation, Integration and Test plan

The implementation is carried out at all times in parallel in order to meet the deadlines. The following points explain the formats followed in the development of each part:

- **Mobile application:** The mobile application is developed with Android Studio following the languages of use of this, in this case the application is being developed with XML (for the visual part) and Java (for the functional part).

  The development of this starts with the different layouts that will make up the application, once all the layouts have been developed following the mockups presented in the RASD, the development of the functional systems begins, many of these systems will be spun to XML (so it is done first) as is the case of buttons or areas where to show text.

- **Server:**

  The development of the server could be distinguished as in 3 phases, all these parts are necessary for the service to work. Following the order of development:

  - The first phase would be Docker, Docker allows us to perform a virtualization without worrying about the system that will run the service, thanks to this allows us to focus more on the code and not if it worked in the system.

  - The second phase would belong to Node.js, Docker will contain for its virtualization a Node.js which provides an optimization in the server and allows us to use Javascript on the server side.

  - Finally we will use the Koa.js framework, one of the most modern frameworks of Node.js.

  All this development is done on a private server managed by JJSoftware.

  Inside the server will be the databases, the databases used are InfluxDB and MongoDB, these databases will be virtualized and also carried by Docker. Docker will make connections between the databases and the service using the IP protocol.

  The databases are developed externally using the programs provided by MongoDB and InfluxDB, in this development is made the internal configuration and adjust the data that will process them.

- **Dashboard:**

  The Front-End of the web application will be done using React in conjunction with Redux, all this will be done under the JavaScript programming language.

  The different layouts that make up the web application will be developed first, they will follow the Mockups presented in the RASD, all these layouts will be created using the main React languages (HTML, CSS, JavaScript (together with JSX)). Following the implementation of the layouts, the Redux library is added to control the states of the application, which is done in its main JavaScript language.

5.1 Time spent on implementation

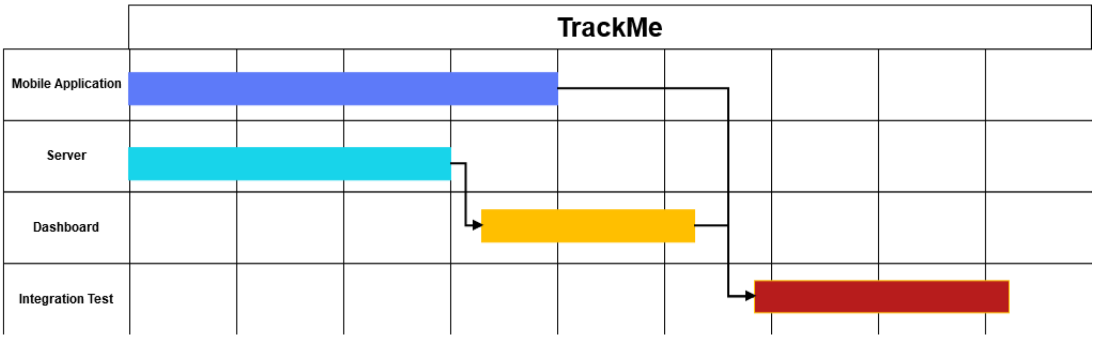


Figure 15: Gantt chart

Figure 15 shows the spaces of time needed to develop each block of which the service is composed.

These blocks, as can be seen in some cases, depend on the completion of another, given that for their operation or implementation it is necessary that another part of the service is in operation.

The times for the development of each block have been estimated on the basis of the number of people working in it as well as for the previous knowledge that was had of that architecture and the complexity that they present.

5.2 Test plan:

The testing plan that will be used for the verification of the systems and the service is based on TDD programming, by means of this mode of testing we will carry out a wide verification of the systems allowing to give this way a safe, effective and optimal service.

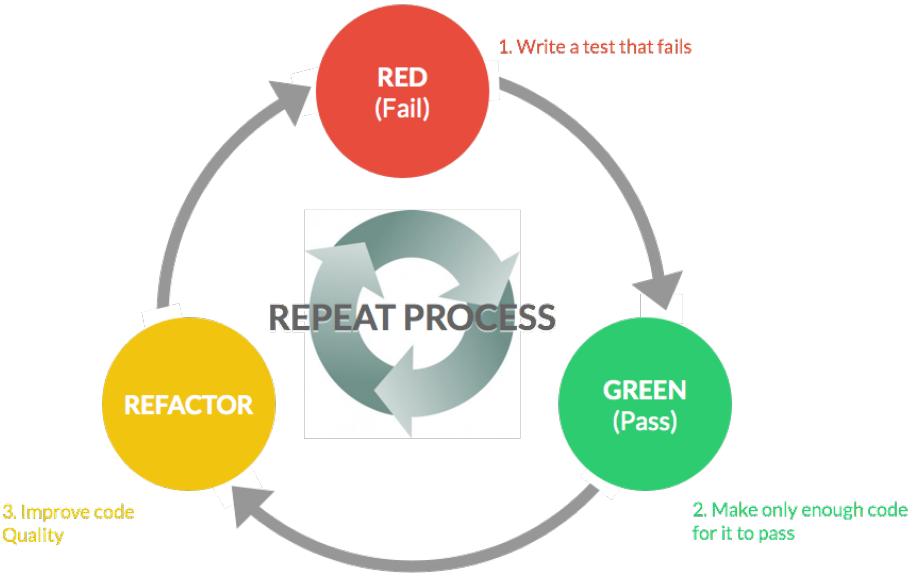


Figure 16: Test-Driven Development

6 Effort Spent

Person	Task performed	Time spent
Javier Fernández	Definitions	1h

Table 16: Effort spent in Section 1 (page 7)

Person	Task performed	Time spent
Javier Fernández	Overview	1h
Javier Fernández	Component view	2h
Javier Fernández	Deployment view	3h
Javier Fernández	Runtime view	3h
Javier Fernández	Server interfaces	2h
Javier Fernández	Selected architectural style and patterns	2h
Javier Fernández	Other design decisions	1:30h

Table 17: Effort spent in Section 2 (page 10)

Person	Task performed	Time spent

Table 18: Effort spent in Section 3 (page 21)

Person	Task performed	Time spent
Javier Fernández	Mapping of use cases and interfaces	30 minutes

Table 19: Effort spent in Section 4 (page 22)

Person	Task performed	Time spent

Table 20: Effort spent in Section 5 (page 24)

References

[1] Nicolas Bonvin, “Serving static files: a comparison between Apache, Nginx, Varnish and G-WAN,” *Spoot!*, March, 2011. URL: `https://nbonvin.wordpress.com/2011/03/24/serving-small-static-files-which-server-to-use/`.

[2] Edward R. Laskowski, “What’s a normal resting heart rate?,” *Mayo Clinic*, Aug., 2018. URL: `https://www.mayoclinic.org/healthy-lifestyle/fitness/expert-answers/heart-rate/faq-20057979`.

Appendix A Use cases

ID	UC1
Name	Sing up of Clients.
Actor	Client.
Entry conditions	The client have to be on the web application.
Events flow	1. The client have to click on the button of sing up of the web application (figure 17). 2. Fill in the necessary information requested in the form that will appear as well as a form of payment. 3. After the confirmation the system will save the data and the client will be registered.
Exit conditions	The client will be registered and able to work with TrackMe.
Exceptions	1. The form of payment is not accepted or is incorrect. 2. The client is already registered. 3. The client has not filled in one of the necessary information fields or a field is not filled in correctly.

Table 21: Sing up of a Client use case

ID	UC2
Name	Log in of Clients.
Actor	Client.
Entry conditions	1. The client have to be registered on TrackMe. 2. The client have to be on the web application.
Events flow	1. Press the log in button in the web application (figure 17). 2. Complete the username and password sections of the log in window (figure 18). 3. After clicking on the log in button, if it is correct the username and password will access to user account and the system will redirect to the main window (figure 19).
Exit conditions	The client will access his account.
Exceptions	Username and password do not match or do not exist.

Table 22: Log in of a Client use case

ID	UC3
Name	Sing up of Users.
Actor	User.
Entry conditions	The user must have the application installed and be in it.
Events flow	1. The user have to click on the button of sing up of the application (figure 29). 2. Fill in the necessary information requested in the form that will appear. 3. After the confirmation the system will save the data and the user will be registered.
Exit conditions	The user will be registered and able to use TrackMe services.
Exceptions	1. The username already exists in the system. 2. The email already exists in the system. 3. The user has not filled in one of the necessary information fields or a field is not filled in correctly.

Table 23: Sing up of a User use case

ID	UC4
Name	Log in of Users.
Actor	User.
Entry conditions	1. The user have to be registered on TrackMe. 2. The user needs to have the application installed.
Events flow	1. Press the log in button in the application (figure 29). 2. Complete the username and password sections of the log in window. 3. After clicking on the log in button, if it is correct the username and password the user will access be redirected to the main window (figure 30).
Exit conditions	The user will access his account.
Exceptions	Username and password do not match or do not exist.

Table 24: Log in of a User use case

ID	UC5
Name	Search of an individual.
Actor	Client.
Entry conditions	The client have to be registered on TrackMe and log in on the web application.
Events flow	1. Click on the individual data search button. (Figure 19) 2. Enter the Codice Fiscale in the search area that appears on the new page (in the area where it is requested). (Figure 20) 3. The system will show the data of the user if the client have the necessary permissions, another search can be performed also. (Figures 21, 22, 23)
Exit conditions	The client will be able to see the information of the requested user.
Exceptions	1. The Codice Fiscale does not exist. 2. The Codice Fiscale is misspelled. 3. The client does not have the permissions to view the user's data.

Table 25: Case of use of individual data search

ID	UC6
Name	Querying group data.
Actor	Client.
Entry conditions	The client have to be registered on TrackMe and log in on the web application.
Events flow	1. Click on the group data search button of the web application (figure 19). 2. Client will be redirected to a search page, figure 24, where a query can be formulated. 3. After pressing the search button the system will show the information of the users (anonymously) who meet the criteria given (figures 25, 26 and 27).
Exit conditions	The client will be able to see the data of the group that fulfills the given requirements.
Exceptions	1. Insert a search criteria that the set of users that satisfy them is less than 1000. 2. Any of the search criteria given is misspelled or does not exist.

Table 26: Case of use of group data search

Appendix B Requirements

ID	Goal	Description
FR1	GL1	When an user opens the application and no login had been performed, the system shall show the welcome page (figure 29).
FR2	GL1	When the welcome page is shown, the system shall show two buttons (figure 29). When clicked, one of them shall redirect to the login page and the other to the registration page.
FR3	GL1	When the registration page is completed, the system shall show the terms and conditions page and only users that accept the terms and conditions will successfully registered.
FR4	GL3	When the user logs in for the first time in the application, the application shall check what sensors are available an issue an Android Permission Request for each of them.
FR5	GL3	If the user declines an Android Permission Request, the application shall issue again an Android Permission Request for the same sensor.
FR6	GL3 GL4	The system shall poll the available sensors in the background at fixed time intervals and store the measures in the server.
FR7	GL4	The fixed intervals at which each sensor shall be polled are stated in 35.
FR8	GL3 GL4	The system shall prompt the user to introduce the manual parameters at fixed time intervals and store the measures in the server.
FR9	GL4	The fixed intervals at which the manual parameters shall be asked to the user are stated in 35.
FR10	GL7	When a client has sent a request for access, the system shall display a notification in the user's device showing the name of the client which requires the permission and a button to accept.

Table 27: Functional requirements of user application

ID	Goal	Description
FR11	GL6	A query consists of a set of parameters with associated logical constraints. The result of the query must comply all the logical constraint in the query.
FR12	GL6	The numerical parameters' logical constraints can be equal (=), greater (>), greater or equal (=>), smaller (<) and smaller or equal (=<). The PM7 parameter do not follow this requirement.
FR13	GL6	The PM7 parameter's logical constraint is expressed as a set of points in which the searched values are geographically inside.
FR14	GL6	The system should provide specific inputs adapted to the type of data to introduce the logical constraints of the query. Table 36 states the parameters and its inputs.
FR15	GL6 GL9	When the client introduces a query from the dashboard page (figure 24) and the number of entries that fullfil the query are equal or more than 1000, the system shall show the data in page (Figures 25, 26 and 27).
FR16	GL6 GL9	When the client introduces a query from the dashboard page (figure 24) and the number of entries that fullfil the query are less than 1000, the system shall warn the client about the impossibility to show the results in page.
FR17	GL8	When the system is showing a data batch in the dashboard that fullfils a query (figures individual search: 21, 22 and 23; figures group search: 25, 26 and 27) and new data that also fullfils the query arrives, the system shall update the view of the data without intervention of the client.
FR18	GL7	When the client introduces a codice fiscale from the dashboard page (figure 20), the user exists and the client have already obtained the permission of the user, the system shall return the data associated to the individual.
FR19	GL7	When the client introduces a codice fiscale from the dashboard page (figure 20), the user exists and the client do not have the permission of the user, the system shall prompt the client to ask permission to the user.
FR20	GL7	When the client introduces a codice fiscale from the dashboard page (figure 20), and the user do not exists, the system shall prompt the client to ask permission to the user.
FR21	GL7	When the client is requesting permission to a concrete user in page and clicks on *Yes*, the system shall emit a to the appropriate user application requesting their permission.
FR22	GL7	When a user approves the request of access made by a client, the system shall store that permission.
FR23	GL7	The system shall show the client a list of all users that had give their permission of access in page in descending alphabetical order.

Table 28: Functional requirements of the client dashboard

In table 29 the phrase *When a message with a correct format reach* is used often. The correct format reefers to the one stated in section **??** for each corresponding SI interface.

ID	Goal	Description
FR24	GL1	When a message with a correct format reach the interface SI6 with an existing pair of username and password, the system shall replay with a token that will identify the client in the next api calls. The token have a validity of 3 days.
FR25	GL6, GL9	When a message with a correct format reach the interface SI7 with a well form query and the result of the query have 1000 entries or more, the system shall replay with a data batch that complies the logical constraints expressed in the query.
FR26	GL6, GL9	When a message with a correct format reach the interface SI7 with a well form query and the result of the query have less than 1000 entries, the system shall replay with a 403 error.
FR27	GL7	When a message with a correct format reach the interface SI8 with a valid codice fiscale, the user exists and the client have already obtained the permission of the user, the system shall return the data associated to the individual.

Table 29: Functional requirements of the client API

ID	Goal	Description
FR28	GL10	When a parameter sent by an user's application arrives at the server and is below a defined threshold and the user is sign up in AutomatedSOS, the system shall rise an alarm to the Emergency System using interface SI1 within 5 seconds.

Table 30: Functional requirements of the AutomatedSOS service

Appendix C Dependencies

ID	Method	URL	Parameters	Return	Description
SI1	POST	/ambulance	location, phone number, phone number of a family member, triggered parameter	estimated time	Call when an emergency service is needed, the important data of the customer is send to the Emergency system

Table 31: Software interfaces of Emergency API

ID	Method	URL	Parameters	Return	Description
SI2	POST	/v1/tokens	Card Nº, Card expiration, CVC	Card ID	Register a card to be used in SI3. One time use.
SI3	POST	/v1/customer	Card ID, email	Customer ID	Register a client in the payment system, returns the ID that identifies the client in the payment system.
SI4	POST	/v1/subscriptions	Customer ID, Plan ID	Subscription ID	Starts charging the client the amont indicated by the Plan ID
SI5	DELETE	/v1/subscriptions	Subscription ID	Subscription ID	Register a client in the payment system, returns the ID that identifies the client in the payment system.

Table 32: Software interfaces of Payment API

Appendix D User interfaces

- **Web Application Interface**

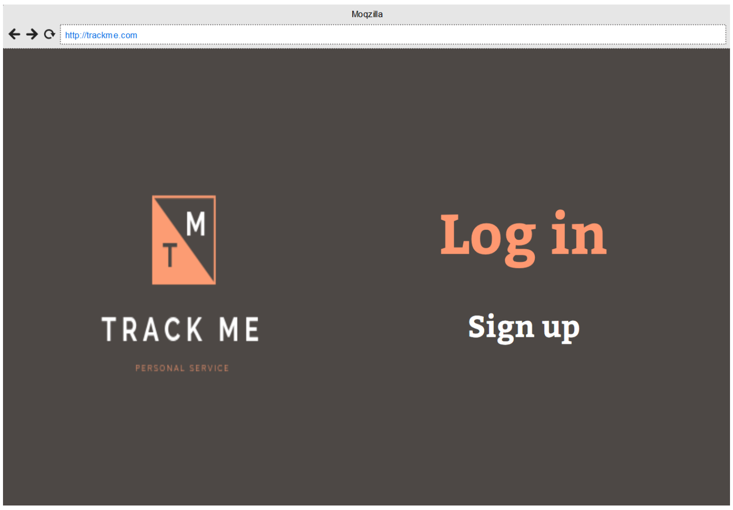


Figure 17: Log in and sing up on the Web Application

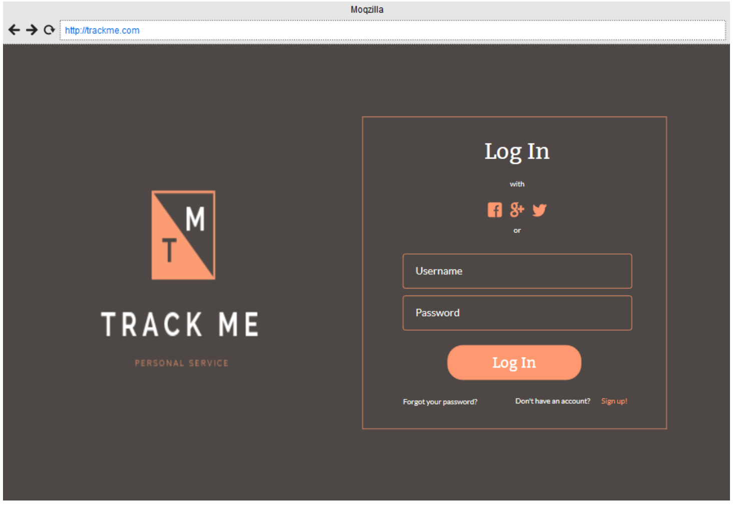


Figure 18: Log in screen of the Web Application

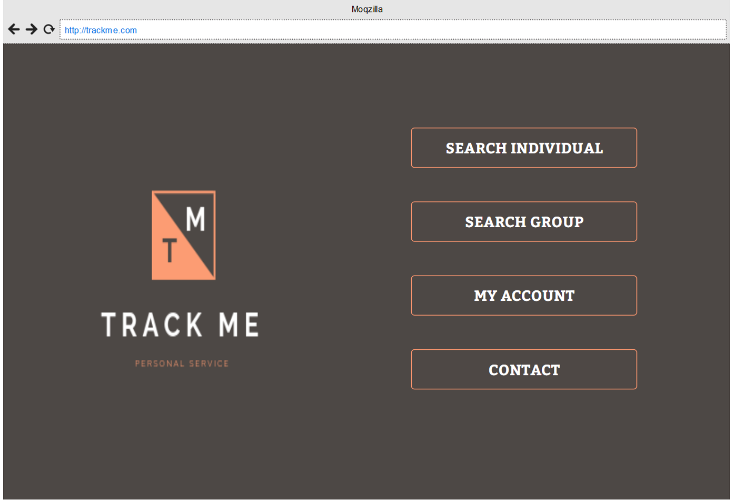


Figure 19: Principal page of the Web Application

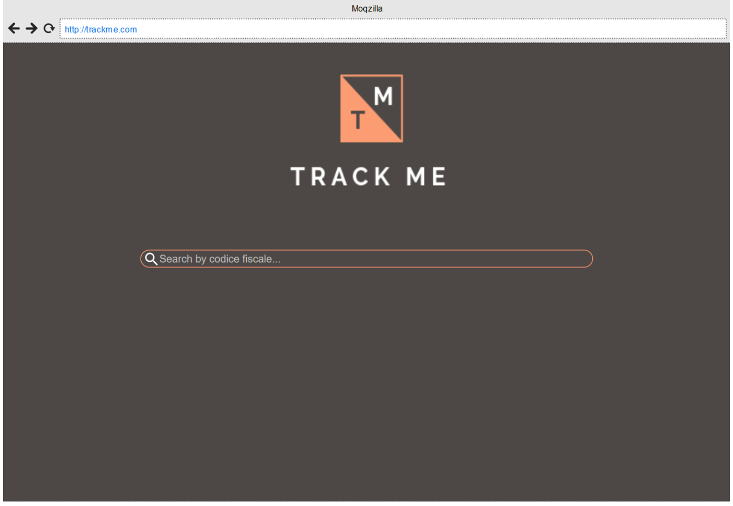


Figure 20: Search for individual data in the Web Application

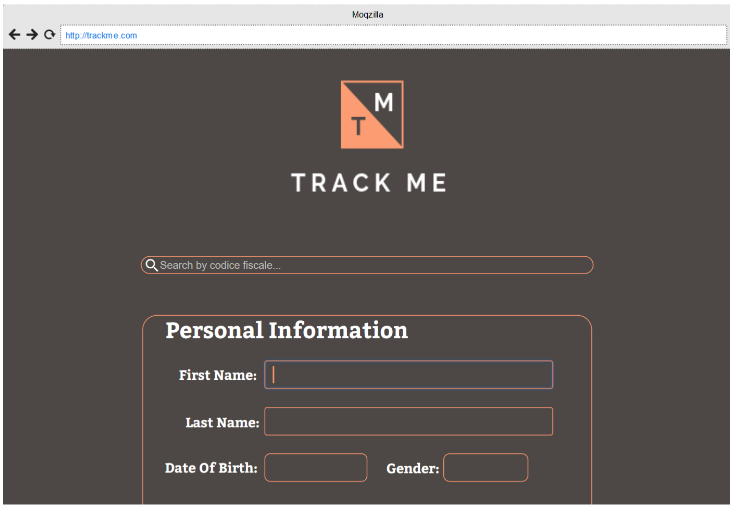
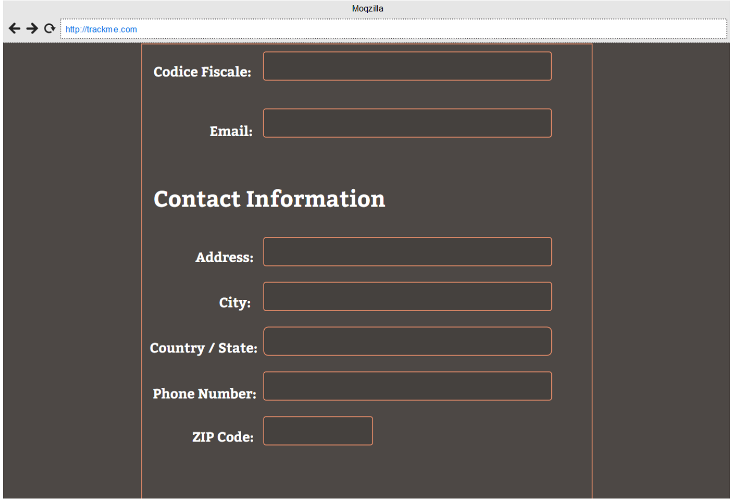


Figure 21: Results of the individual search in the Web Application (First part)



Moqzilla

http://trackme.com

Codice Fiscale:

Email:

Contact Information

Address:

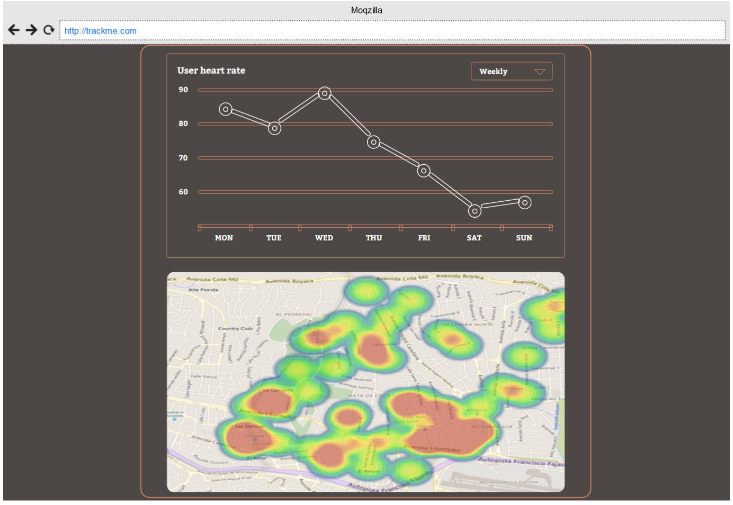
City:

Country / State:

Phone Number:

ZIP Code:

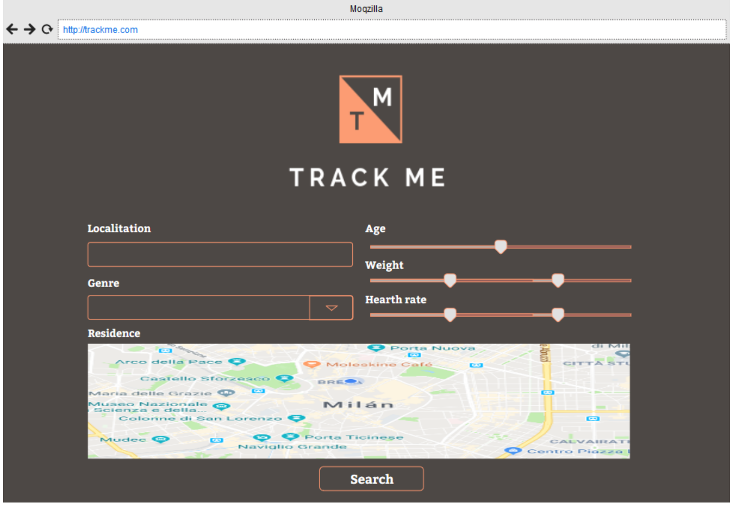
Figure 22: Results of the individual search in the Web Application (Second part)



Moqzilla

http://trackme.com

Figure 23: Results of the individual search in the Web Application (Third part)



Moqzilla

http://trackme.com

Figure 24: Search for group data in the Web Application

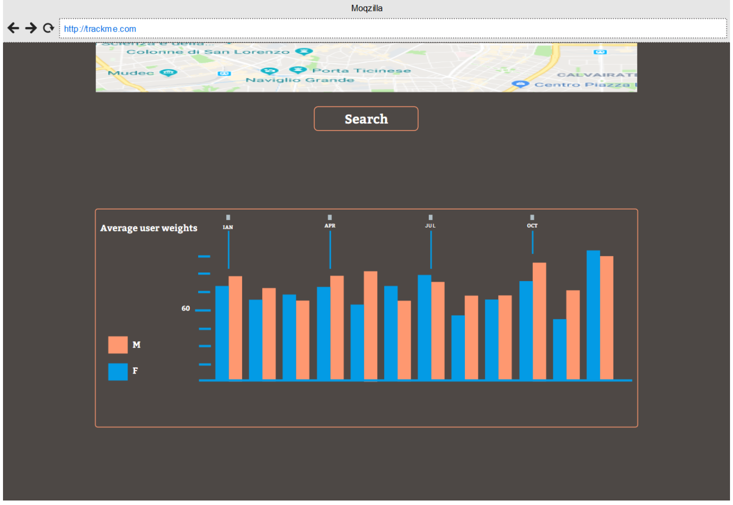


Figure 25: Result of the group search in the Web Application (First part)

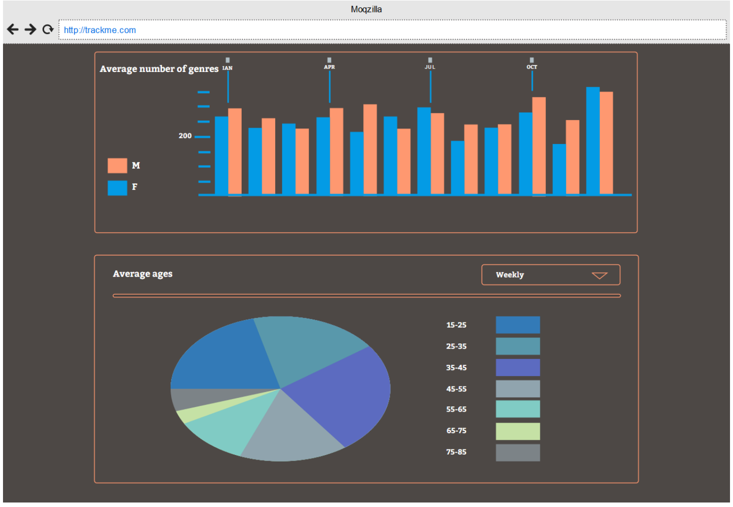


Figure 26: Result of the group search in the Web Application (Second part)

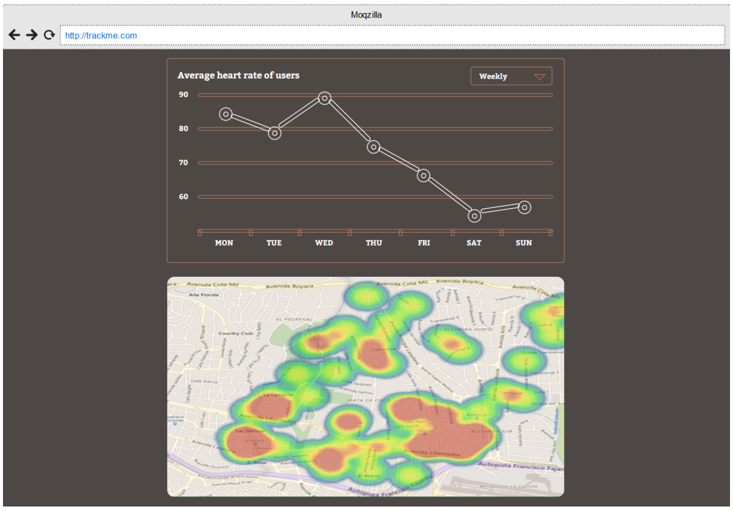
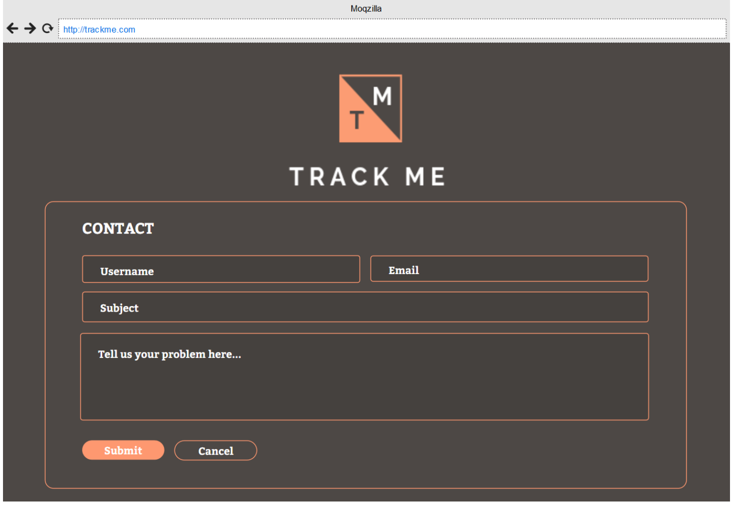


Figure 27: Result of the group search in the Web Application (Third part)



CONTACT

Username

Email

Subject

Tell us your problem here...

Figure 28: Contact zone of the web application

- **Application Interface**

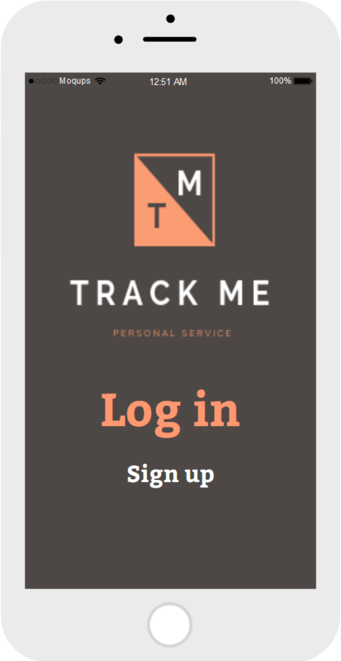


Figure 29: Log in and sing up window of the application

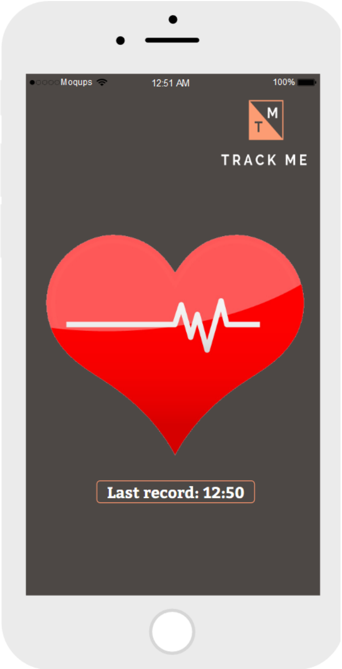


Figure 30: Main window of the application

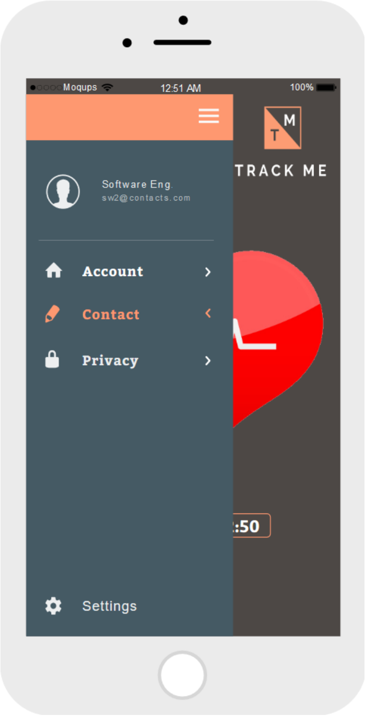


Figure 31: Main application window with slider tab

- **Smartwatch Interface**

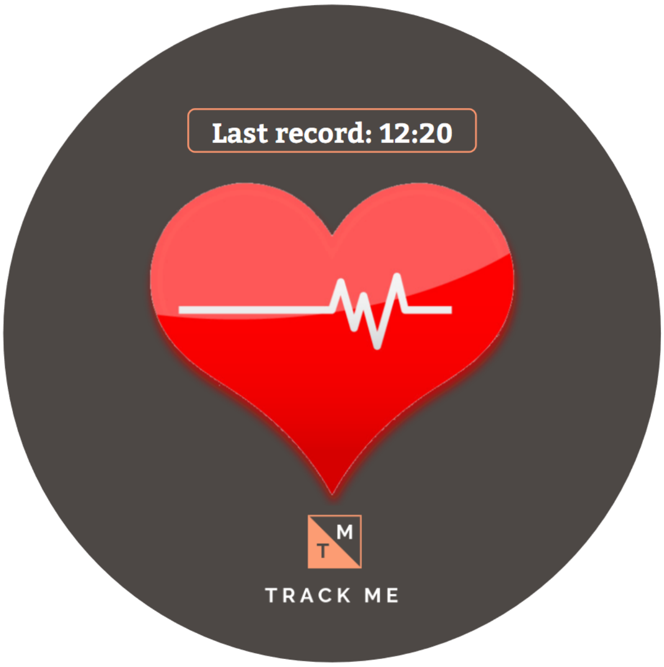


Figure 32: Smartwatch window

Appendix E Software interfaces

ID	Method	URL	Parameters	Return	Description
SI6	POST	/login	Username, password	Client token	Returns a token that will be use to authenticate the user in the next API calls.
SI7	POST	/query	Client token, query	Data batch	Return a data batch containing all the entries that corresponds to the query.
SI8	POST	/search-user	Client token, User ID	Data of the user	Returns the available data of the searched user if the client has permission

Table 33: Software interfaces of TrackMe API for the client

Appendix F Parameters

ID	Parameter	Units	Type	Query	Individual search
PM1	Codice fiscale	String	Fixed, manual	×	✓
PM2	Name	String	Fixed, manual	×	✓
PM3	Surname	String	Fixed, manual	×	✓
PM4	Birth date	dd/mm/yyyy	Fixed, manual	year	✓
PM5	Genre	M/F	Fixed, manual	✓	✓
PM6	Residence	Latitude, longitude	Fixed, manual	Searchable, not shown	✓
PM7	Location	Latitude, longitude	Temporal, automatic	✓	✓
PM8	Hearth rate	bpm	Temporal, automatic	✓	✓
PM9	Weight	Kilograms	Temporal, manual	✓	✓

Table 34: List of parameters and its type

Appendix G Inputs and intervals associated to parameters

Parameter	Interval	Motivation
PM7	5 minutes	Necessary interval for a correct control of the state of health.
PM8	5 minutes	Since AutomatedSOS is build on top of the data recollected by TrackMe, 5 minutes allows the system monitor the health state of the user.
PM9	7 days	Since this parameters is entered manually by the user, 7 days is a period long enough to not disturb users and to collect enough data to be useful.

Table 35: Intervals at which recollection is performed

Parameter	Input	Description of input
PM4	Slider (8 to 100)	An slider with a minimum of 8 and a maximum of 100 years. The client will be able to select two numbers using two handlers. The input will formulate a query in which all the dates between the 1° of January of the actual year minus the second number and the 1° of January of the actual year minus the first number are included.
PM5	Dropdown	A dropdown with two options. The first option is M and the second F. The input will formulate a query in which if the first option is selected, the query will return data from male users. If the second option is selected, the query will return data from female users.
PM6	Map	An interactive map centred in the city of Milan. The map should allow the drawing of an area. The input will formulate a query in which all the points inside the aforementioned area are include.
PM7	Map	An interactive map centred in the city of Milan. The map should allow the drawing of an area. The input will formulate a query in which all the points inside the aforementioned area are include.
PM8	Slider (40 to 120)	An slider with a minimum of 40 and a maximum of 120 bpm, these values are based on [2]. The client will be able to select two numbers using two handlers. The input will formulate a query in which all the numbers between the first number and the second number are included.
PM9	Slider (40 to 300)	An slider with a minimum of 40 and a maximum of 300 kg. The client will be able to select two numbers using two handlers. The input will formulate a query in which all the numbers between the first number and the second number are included.

Table 36: Inputs to be displayed to the clients