TrackMe project Julián Cuéllar Mangut
Javier Fernández Rodríguez

Implementation and Test deliverable

Deliverable:	DD
Title:	Design Document
Authors:	Julián Cuéllar Mangut, Javier Fernández Rodríguez
Version:	1.0
Date:	13-January-2019
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 Introduction

This document is intended to provide a deeper understanding of how the TrackMe project has been implemented and how the different parts of which it is composed have been dealt with.

1.2 Scope

The objectives of the project were exposed in the RASD as well as in the DD, can be found in their respective sections "Scope".

The objectives that are expected to be achieved with this document are those presented below:

- Learn more about how TrackMe works.
- Learn more about how TrackMe implementation has been developed.
- Know what steps to follow to use TrackMe.

2 Functions Implemented

Based on the functional requirements that were presented in the SADR and that can be seen at the end of this document (Appendix C) in this version of the system are implemented:

Requirement ID
FR1
FR2
FR4
FR5
FR6
FR7
FR8
FR9
FR10
FR11
FR13
FR14
FR15
FR16
FR17
FR18
FR19
FR21
FR22
FR23
FR24
FR25
FR26
FR27

Table 1: Functional requirements implemented

Almost all the functions have been implemented leaving a fully functional system, the functions that have not been decided to implement (or have been decided to leave in last place) have been because they were not necessary for the proper functioning of the system.

3 Frameworks Adopted

3.1 API

The framework used is Koa.js. Koa.js is based on Express, which is well established as framework for JavaScript based APIs. The advantages of the former over the latter is the minimalist approach and the support to ES2017 functionality.

The database chosen is Timescale, a relative new database which reconciles SQL with timeseries data. Since its newness, there are not specific libraries to deal with the connection between database and backend. To tackle this issue, knex.js is used.

Knex.js provides a thin client with plays well with Postgres, in which timescale is based. Although it do not have fancy features, Knex.js provides a fine-grained control over the SQL statements executed and, in the worst case, a fallback to raw SQL[1].

3.1.1 Build and runtime

Javascript is well known for causing numerous headaches in terms of creating the necessary environment for its execution. Babel is used to transpile the code into something that NodeJS can understand and yarn/npm is the option chosen to manage the dependencies.

The runtime is handled by Docker and docker-compose. The former takes care of the virtualisation of the containers whilst the latest carry on the orchestration between them.

There are two containers, one for the database and another for NodeJS. The database container can be obtained automatically by docker[2] but the NodeJS one needs to be build locally, since the API code is packed inside.

3.2 Frontend

React is used to bring the user interface to life. Today, considered one of the kings as far as frontend is concerned. Redux is employed to handle the state of the application.

There are two main advantage in the approach suggested by Redux: centralisation and immutability. The former helps in controlling the state that would otherwise be scattered over various components. The latter plays well with React, that needs immutability to compute the differences and render the components correspondingly.

3.2.1 Build and runtime

As in the API, Babel is in charge of the transpilation. The code generated by Babel is supported by the versions of the browser indicated in the RASD.

The building process is handled by parcel, which provides HRM*Hot Module Replacement*, multicore and cached compilation.

[1]Which is used to build the hypertables of Timescale

[2]The image is timescale/timescaledb, can be found in dockerhub

Worth mentioning that in a production application the code will be minified in a single file. This file will be included in a Docker container in front of the API one, serving the frontend. NginX may be a good option for this task, as excels serving static files.

3.3 Mobile Application

For the operation of the application applies Java and XML for Android, thanks to the union of these two languages can get an excellent job thanks to the great potential they present.

4 Structure of the Source Code

4.1 API

The API code can be found in Implementation/API/src. The code are divided in the following structure:

- **database**: Since the main mission of the API is to store and query data, this folder comprises the great majority of the code . Below the main files are stated.
  - **auth**: Insert and deletions related with the authentication. Also contains the code responsible of the generation of the tokens.
  - **init**: Connection to the database and creation of the tables if needed.
  - **insert**: Insert related logic.
  - **names**: Names of the tables, shared across the database functions.
  - **populate**: Functions to generate fake data and insert it in the database.
  - **search**: Functions to query the database.
  - **update**: Functions to update the entries of the database.
- **routers**: The functions contained in this folder handles the HTTP requests. It is primarily responsible for adapting incoming and outgoing data.

Logically speaking, each *resource* of the application are handled by a different router. These routers handles the HTTP requests, checks the presence of the needed parameters and then calls the needed function of the database package.

Worth mentioning that the authentication is carried out by a middleware which can be found in the auth router.

4.2 Dashboard

The Dashboard code can be found in Implementation/frontend/src. The structure of the code follows the Redux guidelines.

- **actions**: Functions that dispatch changes to the global store. All the request to the backend are carried out here.
- **components**: Code that handles the frontend, all the React code can be found here.
  - **auth**: Login related components.
  - **data-display**: Components used to display the data. There are maps, graphs and cards.
  - **permissions**: Permissions related components.
  - **query**: Query related components, filters included. To display the data leverages on data-display package.
  - **search**: Search related components. To display the data leverages on data-display package.
- **reducers**: Handles the actions and performs the needed changes in the global state. Each reducer handles its part of the state.

The functioning of the frontend is rather standard and it is guided by Redux methodology. There are three main components: QueryPage, SearchPage and PermissionsPage.

These components have a portion of the global state assigned as read-only and a set of actions that can fire. Both, state and actions are propagated as needed to smaller components.

When one action is fired, the reducers takes the action and performs the needed changes on the state, which are assigned again to the main components. Then, changes in the frontend are rendered thanks to React.

4.3 Mobile Application

The Mobile Application code can be found in Implementation/android/TrackMe-User (there is an app for development that allows you to see what data is being processed and collected) . The structure of the code follow a simple structure:

- **java classes**: 4 main classes are presented:
  - **InitialScreen** : Controls the entry screen and performs the log in and registration functions.
  - **Recollector** : It performs the main functions of TrackMe, is the class that controls all the parameters of the service and works with the main interface of the application.
  - **Scheduler** : Intermediate class between the Collector and the Router. It is the one that carries out the movement of data between the classes and stores them.
  - **Router** : It is the class that connects the application with the TrackMeAPI, it is the only entry/out point of the application.
- **layouts**: They provide the user interface, the java classes are connected to them and perform the internal functions.

The operation of the application starts with the start screen that is moved by the InitialScreen class, from here you control the Login and Sing up of the application.

Once the Login is done correctly, the Recollector class is emitted, which is assigned to the main screen of the application. This Recollector class controls the timers for data collection, applying the different operations once they are finished. Once received the data are sent to the Scheduler which collects the data stores them and makes them reach the Router who with POST operations connects to the API and sends the different data.

5 Testing

Information about the type of testing you want to apply in TrackMe can be found in detail in the DD document.

6 Installation Instructions

In order to use TrackMeas a user you must download the APK from the mobile application, this APK can be found at `https://i.diawi.com/677GyX` .

The only thing to do after downloading the APK is to install it on the mobile phone and open the application. (The application works when you have in the foreground, background or with the device asleep, it does not work if you close).

References

Appendix A Goals

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 2: Goals

Appendix B 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 1). 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 3: 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 1). 2. Complete the username and password sections of the log in window (figure 2). 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 3).
Exit conditions	The client will access his account.
Exceptions	Username and password do not match or do not exist.

Table 4: 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 13). 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 5: 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 13). 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 14).
Exit conditions	The user will access his account.
Exceptions	Username and password do not match or do not exist.

Table 6: 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 3) 2. Enter the Codice Fiscale in the search area that appears on the new page (in the area where it is requested). (Figure 4) 3. The system will show the data of the user if the client have the necessary permissions, another search can be performed also. (Figures 5, 6, 7)
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 7: 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 3). 2. Client will be redirected to a search page, figure 8, 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 9, 10 and 11).
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 8: Case of use of group data search

Appendix C 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 13).
FR2	GL1	When the welcome page is shown, the system shall show two buttons (figure 13). 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 17.
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 17.
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 9: 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 18 states the parameters and its inputs.
FR15	GL6 GL9	When the client introduces a query from the dashboard page (figure 8) and the number of entries that fullfil the query are equal or more than 1000, the system shall show the data in page (Figures 9, 10 and 11).
FR16	GL6 GL9	When the client introduces a query from the dashboard page (figure 8) 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: 5, 6 and 7; figures group search: 9, 10 and 11) 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 4), 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 4), 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 4), 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 10: Functional requirements of the client dashboard

In table 11 the phrase *When a message with a correct format reach* is used often.

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 11: 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 12: Functional requirements of the AutomatedSOS service

Appendix D 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 13: 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 14: Software interfaces of Payment API

Appendix E User interfaces

- **Web Application Interface**

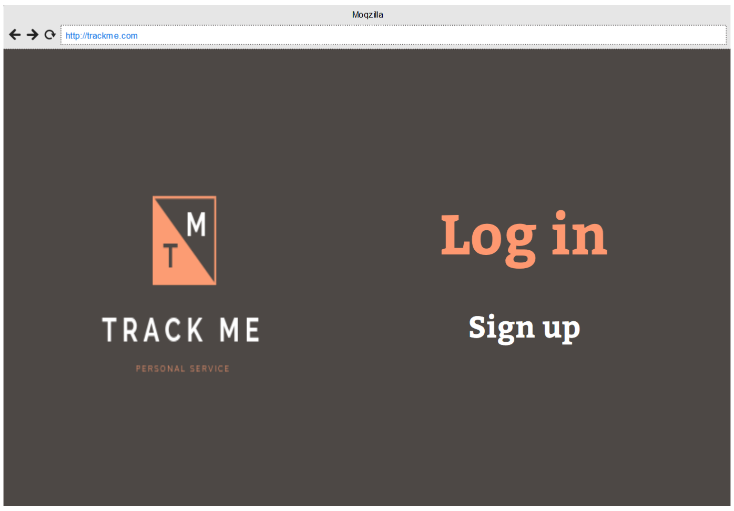


Figure 1: Log in and sing up on the Web Application

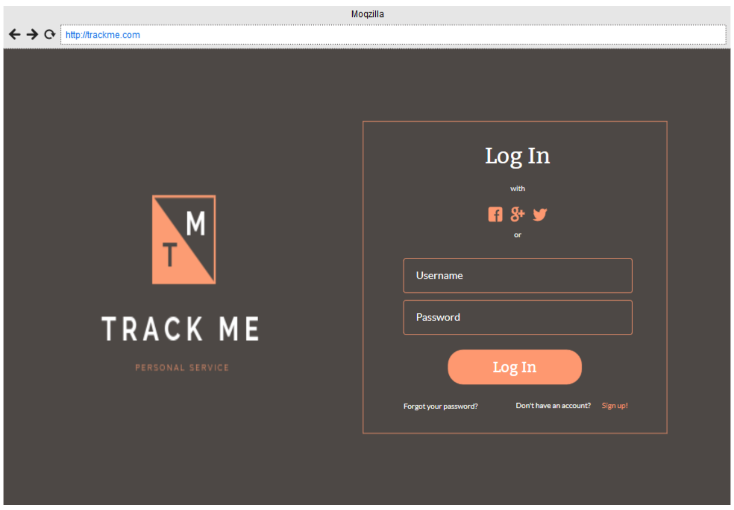


Figure 2: Log in screen of the Web Application

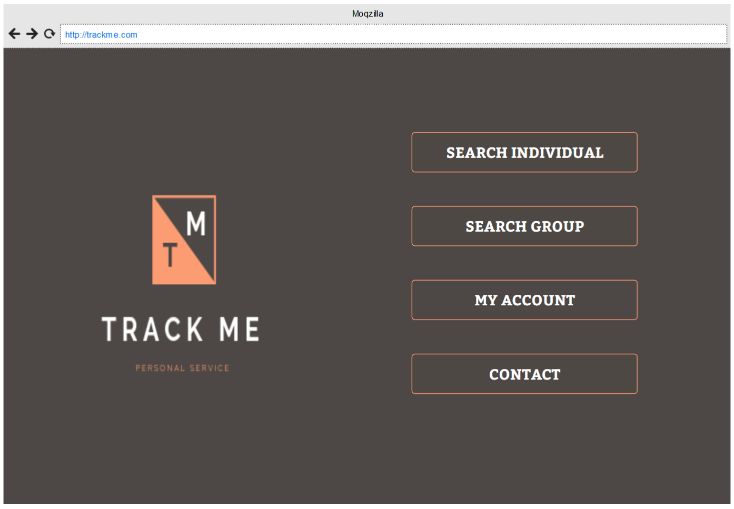


Figure 3: Principal page of the Web Application

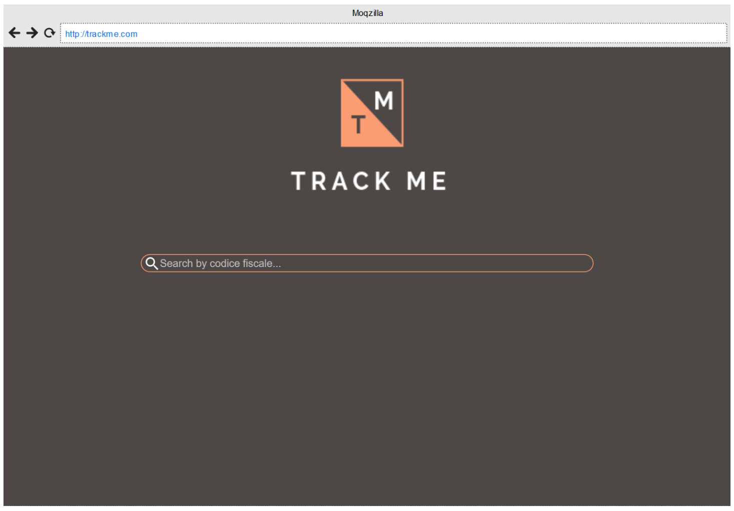


Figure 4: Search for individual data in the Web Application

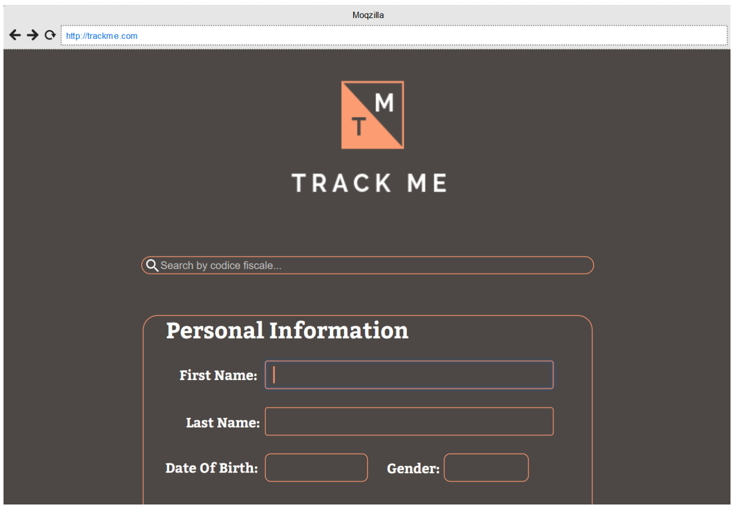
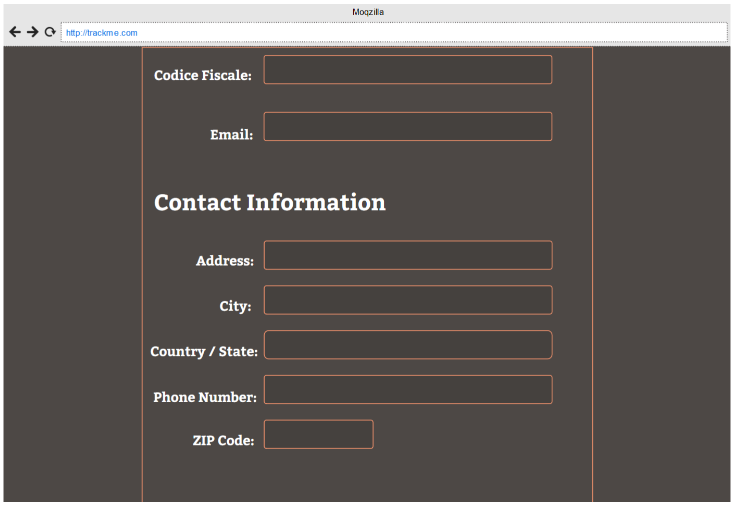


Figure 5: Results of the individual search in the Web Application (First part)


Moqzilla

http://trackme.com

Codice Fiscale:

Email:

Contact Information

Address:

City:

Country / State:

Phone Number:

ZIP Code:

Figure 6: Results of the individual search in the Web Application (Second part)

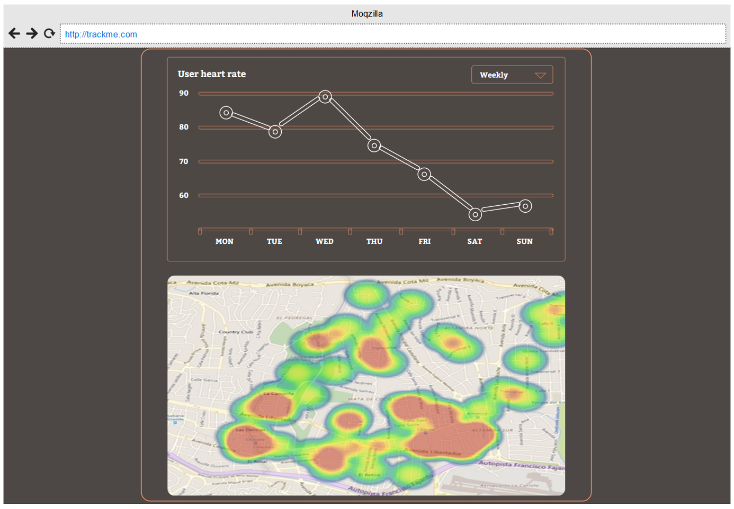
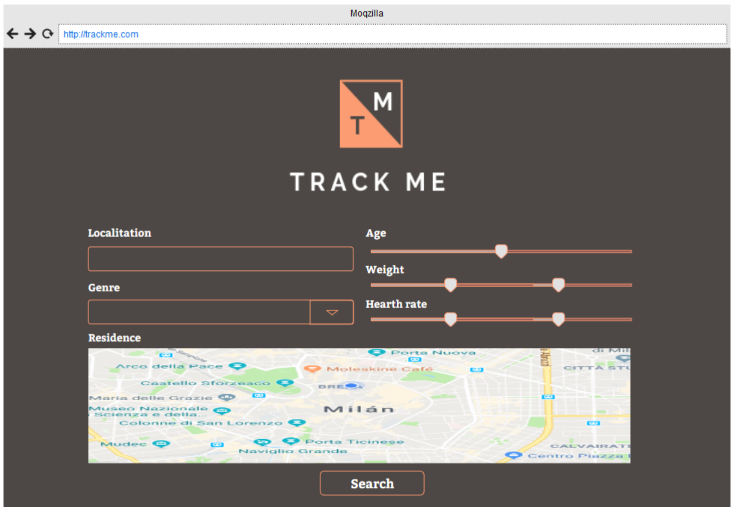


Figure 7: Results of the individual search in the Web Application (Third part)


Moqzilla

http://trackme.com

TRACK ME

Localitation

Age

Weight

Genre

Hearth rate

Residence

Search

Figure 8: Search for group data in the Web Application

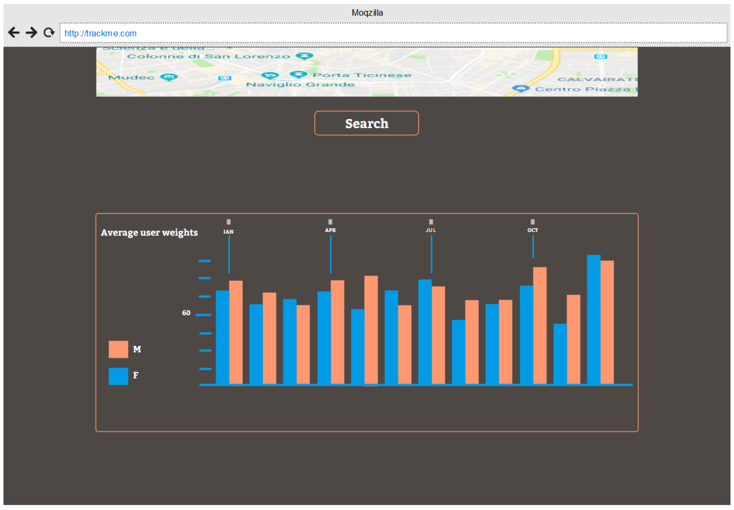


Figure 9: Result of the group search in the Web Application (First part)

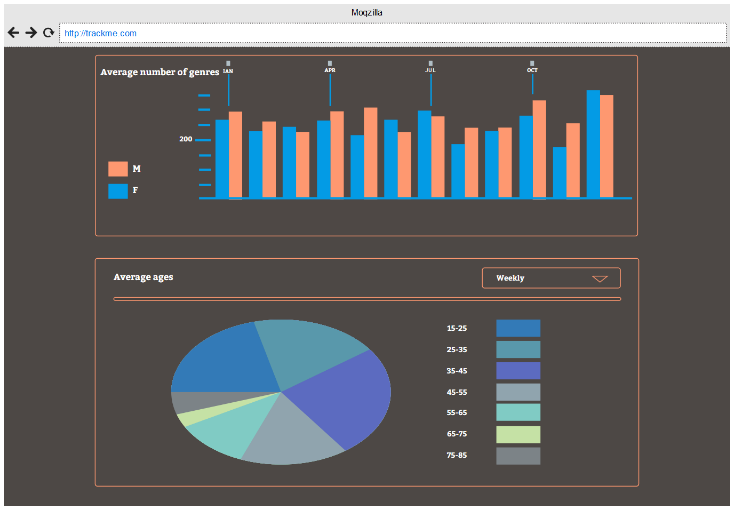


Figure 10: Result of the group search in the Web Application (Second part)

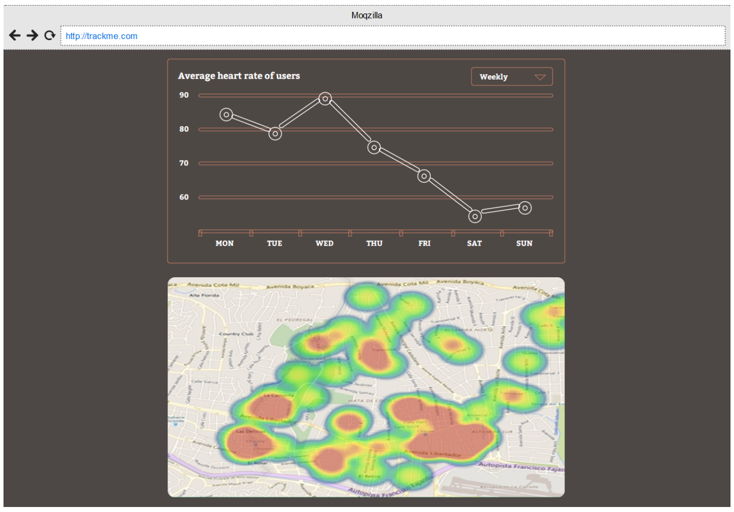


Figure 11: Result of the group search in the Web Application (Third part)

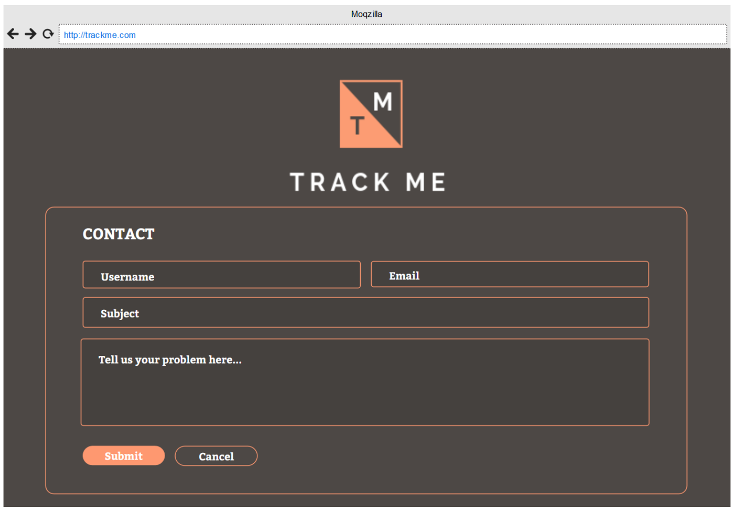


Figure 12: Contact zone of the web application

- **Application Interface**

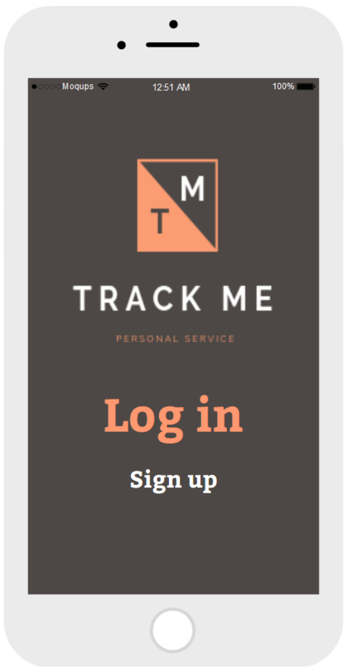


Figure 13: Log in and sing up window of the application

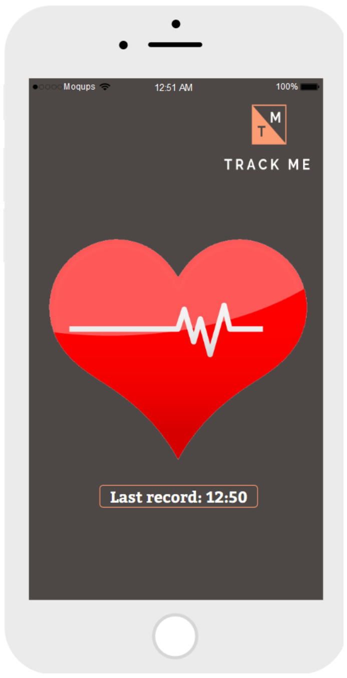


Figure 14: Main window of the application

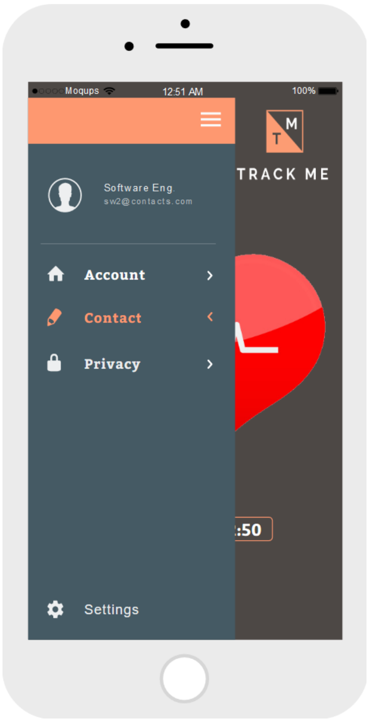


Figure 15: Main application window with slider tab

- **Smartwatch Interface**

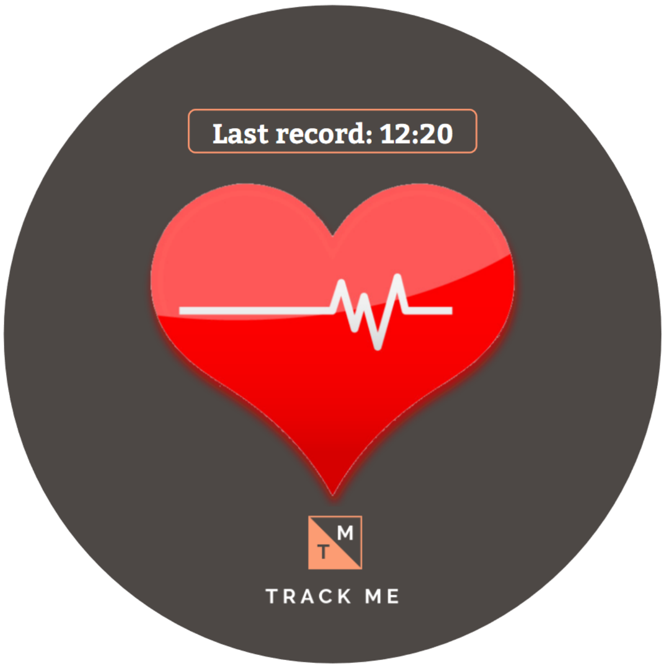


Figure 16: Smartwatch window

Appendix F 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 15: Software interfaces of TrackMe API for the client

Appendix G 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 16: List of parameters and its type

Appendix H 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 17: 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 [**?**]. 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 18: Inputs to be displayed to the clients