POLITECNICO
MILANO 1863

Software Engineering 2
Design Document

Work team:
Paolo Romeo, Andrea Scotti, Francesco Staccone

AY 2018-2019

Table of contents

1. Introduction

1.1. *Purpose*

This document is thought to be an overview of the TrackMe application, in which is explained how to satisfy the several project requirements stated in the RASD. This document is principally intended for the developers and the testers, with the purpose of providing a functional description of the main architectural components, their interfaces and their interactions, along with the design patterns.

1.2. *Scope*

—Presentation—
TrackMe is structured in a multi-tier architecture. More specifically, the Business logic layer has the task of taking charge of the incoming requests/data, computing checks, and interacting with external third-party services through the use of interfaces. This layer is connected with the Data layer, in which are stored all the Users data (credentials, health data). The Presentation layer is build through the fat Client paradigm in which the client needs to perform computation to temporarily store health data in the local memory in case of missing internet connection between the client and the server.

1.3. *Definitions, acronyms, abbreviations*

1.3.1. Definitions

- User: a registered individual who can use the TrackMe services.
- Third party: a registered entity that can use the TrackMe services.
- Client: A client is a piece of computer hardware or software that accesses a service made available by a server;
- Firewall: A network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules;
- Server: A computer program or a device that provides functionality for other programs or devices, called "clients".

1.3.2. Acronyms

- API: Application Program Interface;
- DBMS: Database Management System;
- DD: Design Document
- GUI: Graphical User Interface;
- HTTP: Hypet Text Transfer Protocol;
- MVC: Model View Controller pattern;
- OS: Operating System;
- RASD: Requirements Analysis and Specifications Document;
- REST: REpresentational State Transfer;

1.3.3. Abbreviations

- Gn: n-goal in the RASD;
- Rn: n-functional requirement in the RASD;

1.4. *Revision history*

- 10/12/2018 Version 1.0

1.5. *Reference documents*

- RASD document;
- Mandatory project assignment;

1.6. *Document structure*

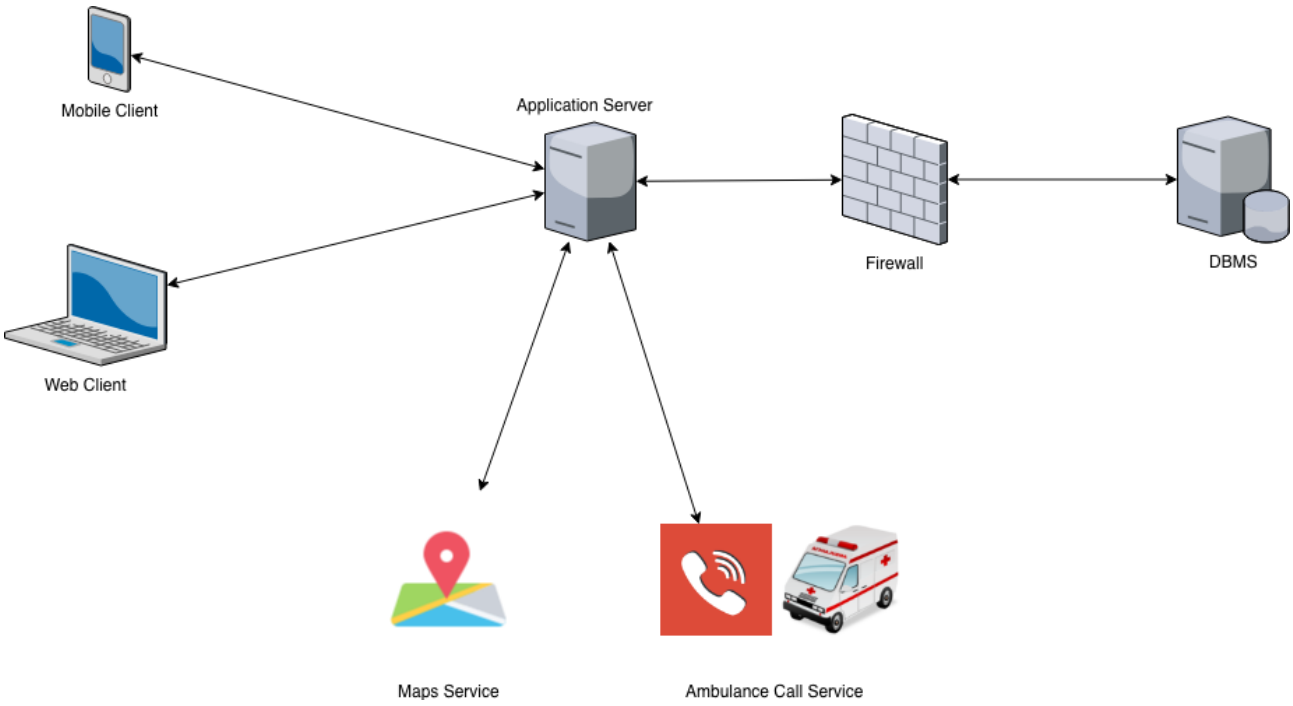
The following document is organised in this way:

- Architectural Design: this section shows the main components of the system and the connections among them. It will also focus on design choices, styles and patterns.
- User Interface Design: this section includes an improvement of the user interface given in the RASD document. It will be described through the use of UX modeling.
- Requirements Traceability: this section shows how the requirements in the RASD are mapped to the design components presented in the DD;
- Implementation, Integration and Test plan: this section shows the order in which the implementation and the integration of the subcomponents will occur and how the integration will be tested.

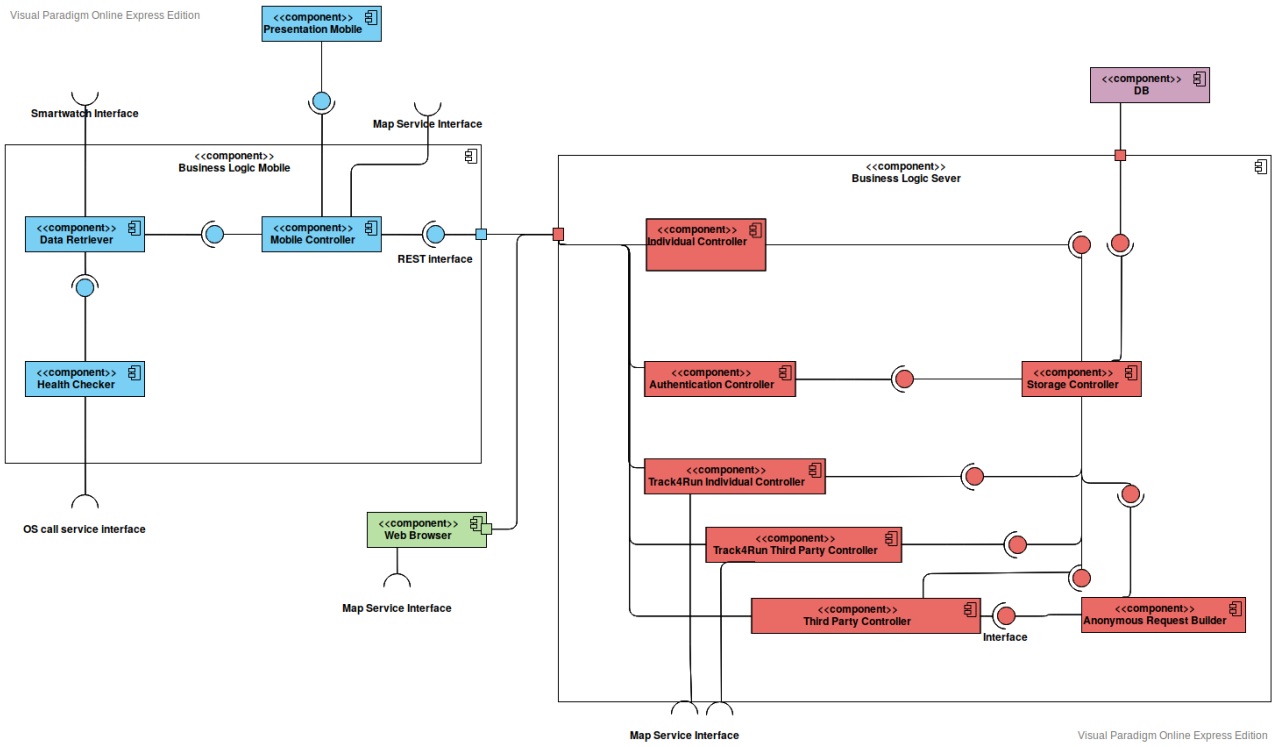
2. Architectural design

2.1. *Overview: high-level components and their interaction*

The figure below represents an high level overview of the system. Further details on the system components and their interaction will be further explained in the next sections.



2.2. *Component view*



The UML component diagram shows the internal structure of the system highlighting the individual

modules and the connections among them. Individual components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. Below is the description of the components:

- **User mobile client**
  This is the component that represents the client machine that accesses to the API of the Business Logic container for the user. ?????????????????????????? Da completare in base a come lo vogliamo implementare (dati lato utente o no?)

- **Third party mobile client**
  This component represents the client machine that accesses to the API of the Business Logic container for the third party. ??????????????????????????????????

- **User controller**
  The main role of this component is to manage the transfer of data from the client to the server using the interface provided by the Data component. It also provides methods to accept/refuse a request of agreement and to change login credentials. It can communicate with the Notification controller component to display notifications.

- **Third party controller**
  This component is built to manage the transfer of data from the system to the third party using the interface provided by the Data component. It also provides methods to send requests of agreement and requests of data to the system. It can communicate with the Notification controller component to display notifications.

- **Authorization controller**
  This component provides authentication and registration processes for both the users and the third parties. It takes the needed data from the Data component and communicates the result of the operations trough the Notification controller.

- **User run controller**
  This component provides the methods to permit a user to join, leave, check and watch a run. It communicates with the external service that provides maps trough the Map services interface. Every time a user wants to join a run, it calls the Feasability checker that checks if all the constraints are satisfied. Finally, it can access data of the runs trough the Data component and it can call the Notification controller to send notifications.

- **Third party run controller**
  This component provides the methods to allow a third party to organize a run. It communicates with the external service that provides maps trough the Map services interface to allow the third party to select the track for the run.

- **Notification controller**
  This component manages the notifications forwarded by the other components. It only shows notifications inside the application.

- **Health checker**
  This component is the core of the service AutomatedSOS. It receives data from the Data component, elaborates them and, if needed, communicates to the Call service interface to send an SOS trough the related external service.

- **Feasability checker**
  This component is built to check if a user can actually join a run that is shown trough the Mobile client. ——— Serve davvero??????????????????????????????????

- **Data**
  The Data component provides the set of Classes corresponding to the tables contained in the

Database.

- **Storage interface** This component provides the methods for querying the Database.

- **Database** This component represent the DBMS. It provides the interfaces to retrieve and store data. In the database there are data about users, third parties and the set of agreements among them. It also contains data about the runs.

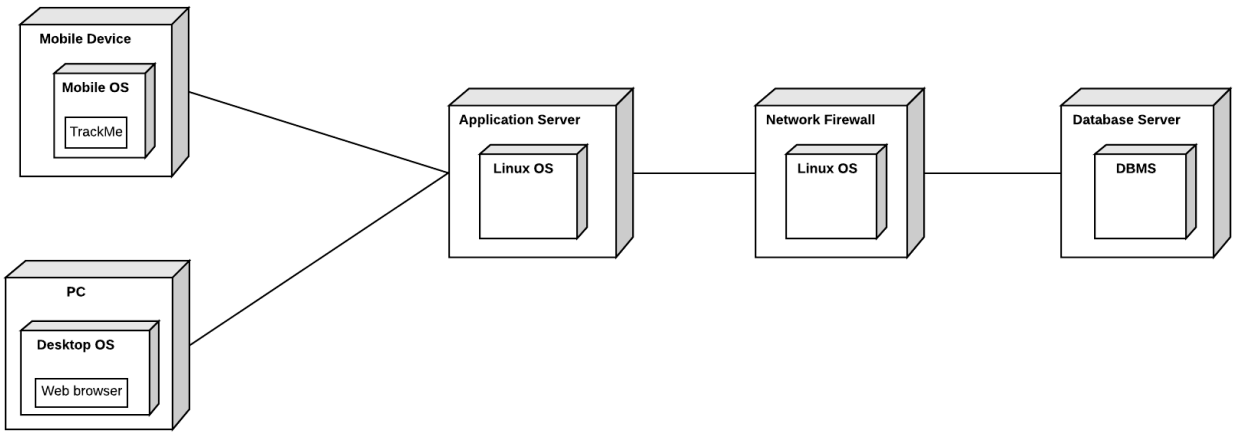
- **External services interfaces**

  - **Map services interface**
    It communicates with the map service to allow the third party to select the track and to show the position of runners in real time on the map.

  - **Call service interface**
    It communicates with the external service to dispatch automatic calls (or emails?) in case of emergency.

2.3. *Deployment view*

The figure below shows the deployment diagram of the whole system. Its main goal is to describe the distribution of components capturing the topology of the system's hardware.



As previously stated in the section 1, the system is structured in a multi-tier architecture. The specific role of each node is clarified here:

Clients
The first tier is composed by the clients machines (mobile for individuals and desktop for third parties). The individuals will be able to access TrackMe functionalities through the dedicated native application while the third parties through any web browser.

Application Server
This is the middleware level of the architecture: all the business logic of the system is contained in this server.

Network Firewall
The access to the Database is mediated by a network firewall in order to avoid unauthorised access to the data and the credentials of the user.

Database Server

This is the last layer of the architecture: all the data are stored in a Database Server accessed through a relational DBMS.

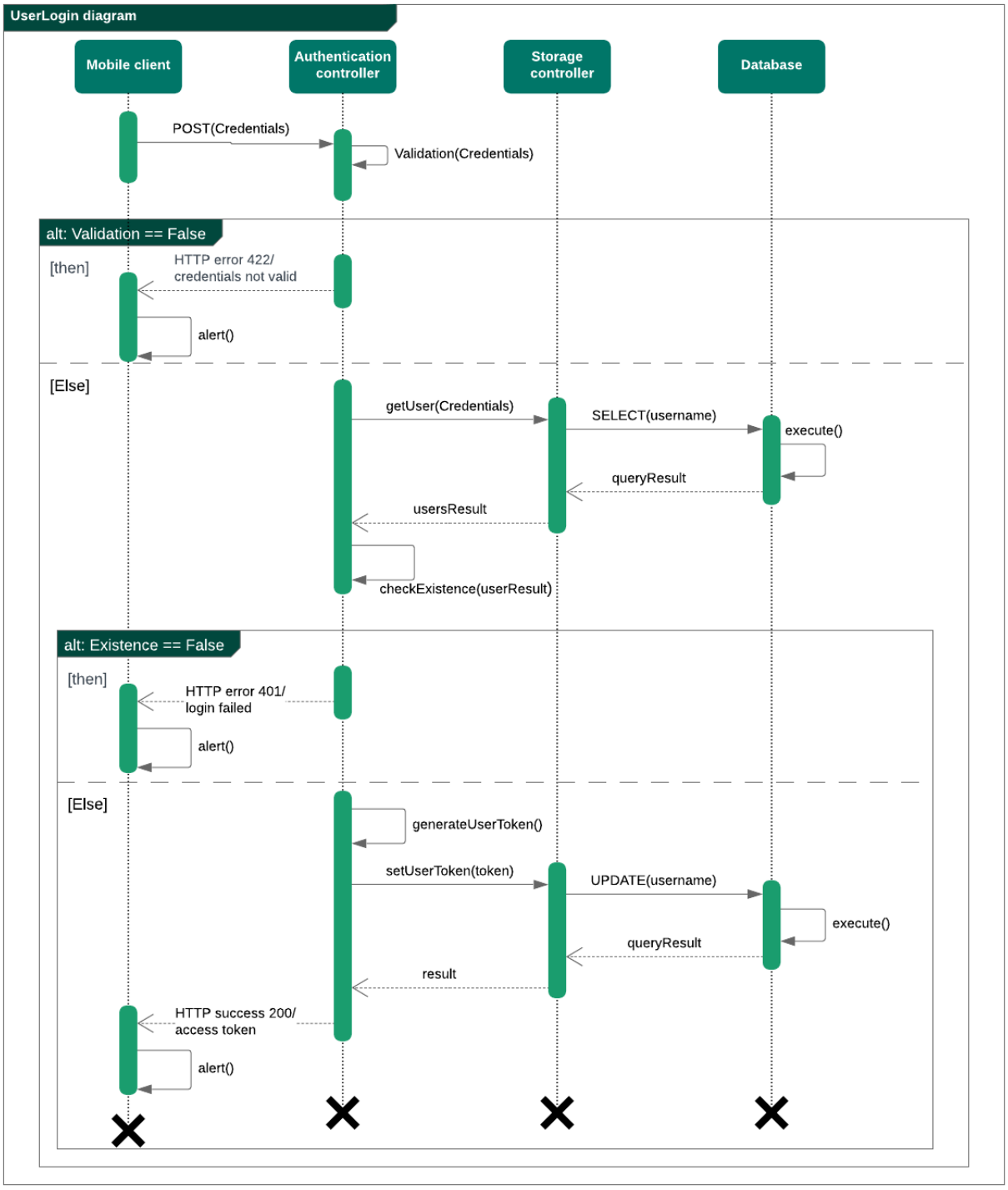
2.4. *Runtime view*

2.4.1. **Sign up Runtime View**

Third parties and individuals have to sign up to the application before being able to access its functionalities. The registration process is the same for both third parties and individuals, the only thing that differentiates it is the information provided by them. The third party must fill up a form with its VAT number and a password. To reach the same goal an individual must fill up a form with username, password and personal data. The information is serialized and then sent to the Application Server through an HTTP POST request. The Authorization controller handles the request and, through the Storage interface, creates a new entry in the User table or in the ThirdParty table on the Database.

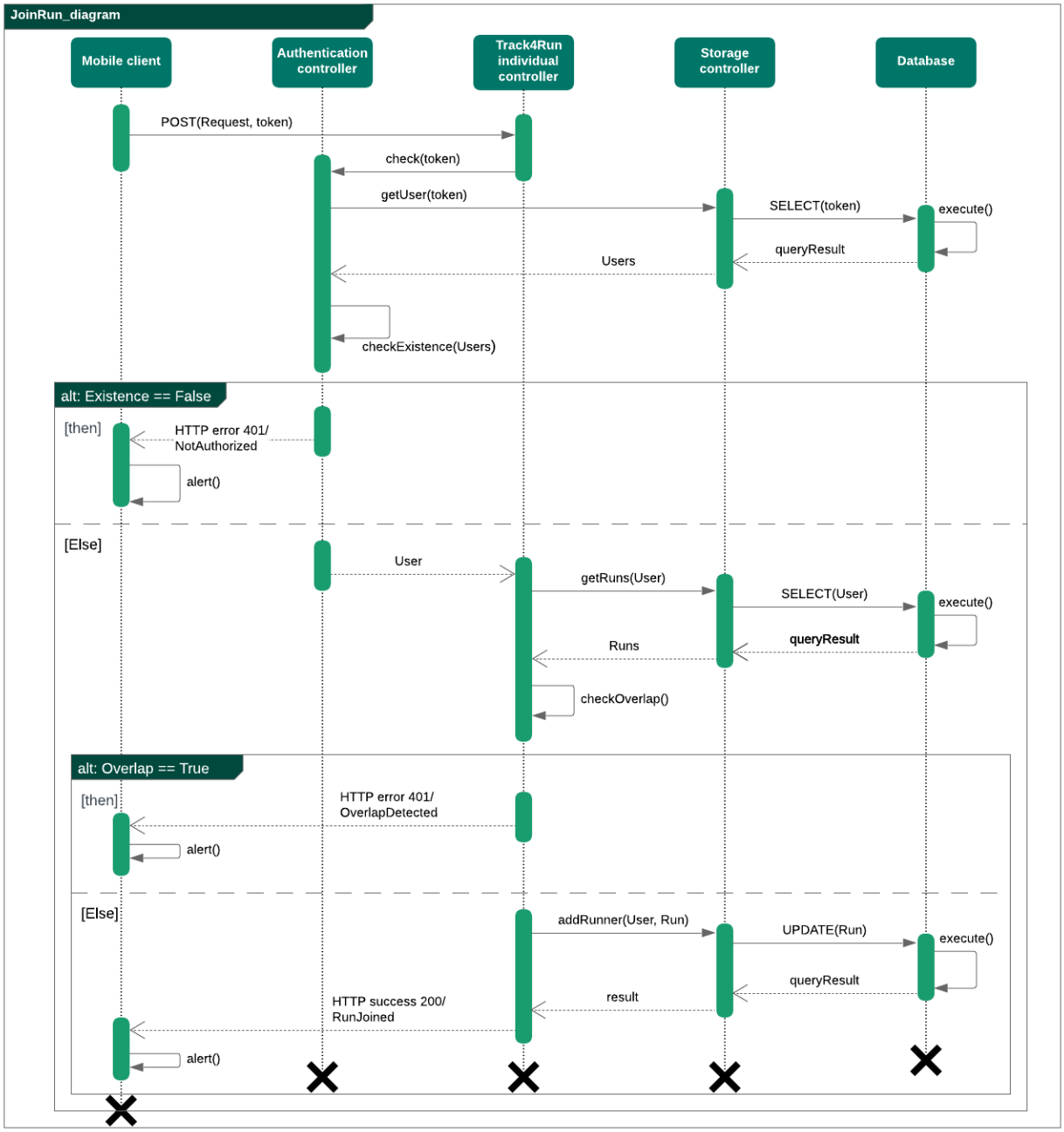
2.4.2. User Login Runtime View

Third parties and individuals have to be logged to the application before being able to access its functionalities. Both third parties and users submit the login information through and HTTP POST request. The request is handled by the Authentication Controller that validates the request and checks if the Database has an entry for the requested account. If the entity is present and the credentials are correct, the Authentication Controller generates an Access Token, sets it as the current access Token for the specified entity and returns it to the entity itself. The following sequence diagram shows how the process works in detail for the login of an individual user.



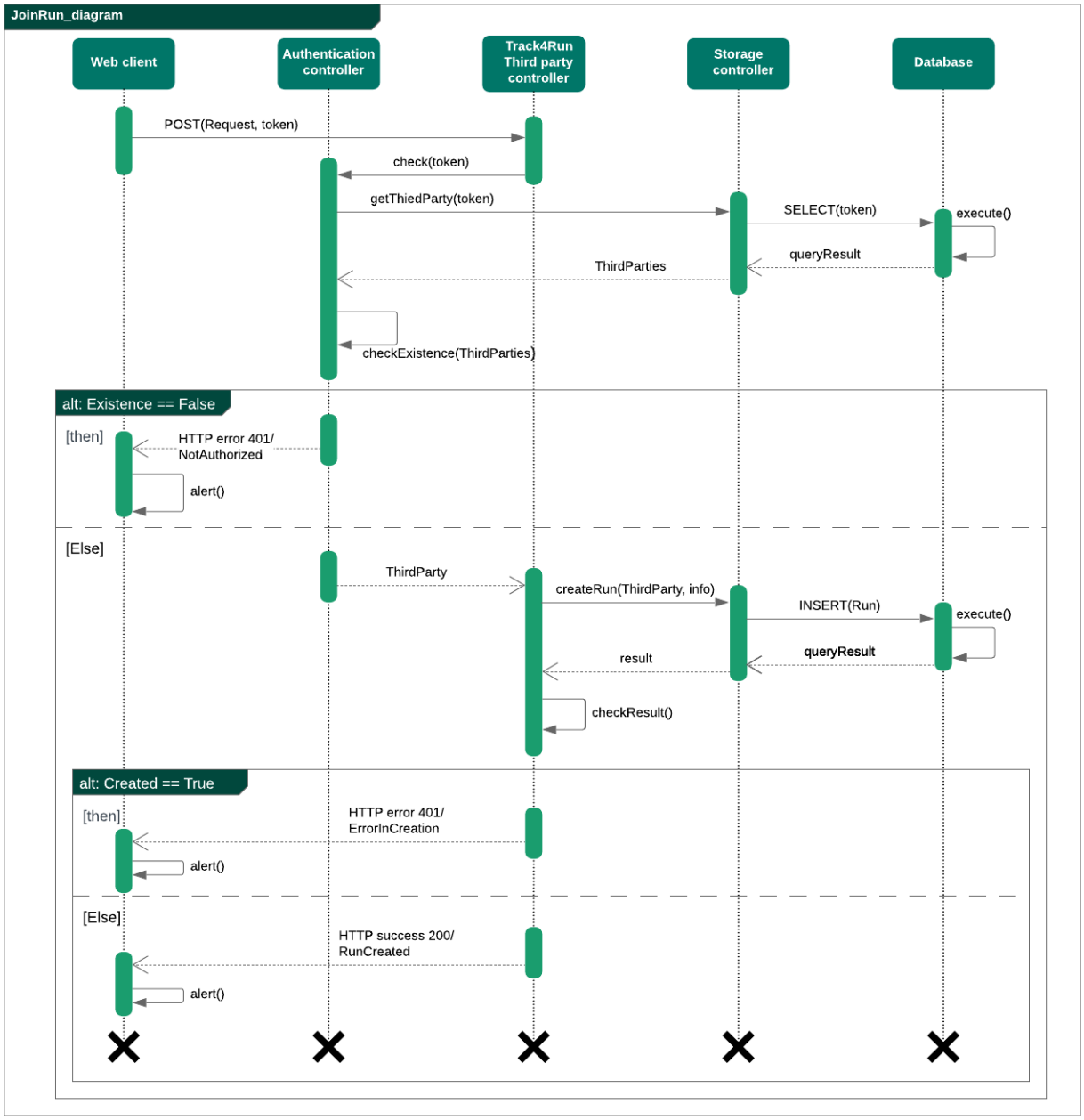
2.4.3. Join a run Runtime View

The User opens the Track4Run section of the application and, after a GET request to the Application Server he/she receives the list of all the available runs and he/she can choose to join one of them (if there is at least one displayed run). The User can send a POST request containg the Access Token and the selected run to the Track4Run Individual Controller. The Track4Run Individual Controller passes the Access Token to the Authentication Controller that checks if the Mobile Client provided a valid Token. If the Authentication succeeds the control goes back to the Track4Run Individual Controller that checks if the user has already joined an overlapping run or if the maximum number of participants was reached. If the control is satisfied the Track4Run Individual Controller asks the Storage Controller to update the tuple of the selected run in the Database. To avoid repetitions, the following sequence diagram shows the process assuming that the User already received the list of available runs.



2.4.4. Organise a run Runtime View

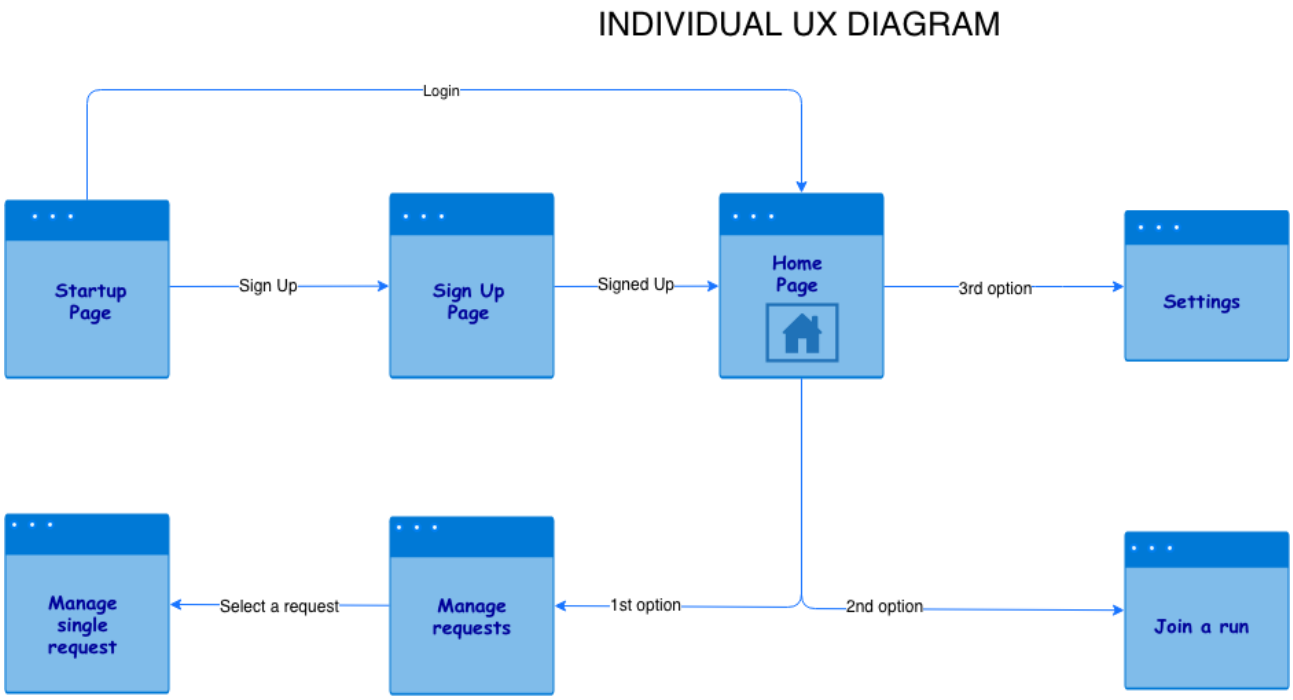
The Third party opens the Track4Run section of the application and, after filling the form containing the information about the run it can send a POST request containg the Access Token and the filled form to the Track4Run Third Party Controller. The Track4Run Third Party Controller passes the Access Token to the Authentication Controller that checks if the Web Client provided a valid Token. If the Authentication succeeds the control goes back to the Track4Run Third Party Controller that asks the Storage Controller to insert a new tuple for the run in the Database.

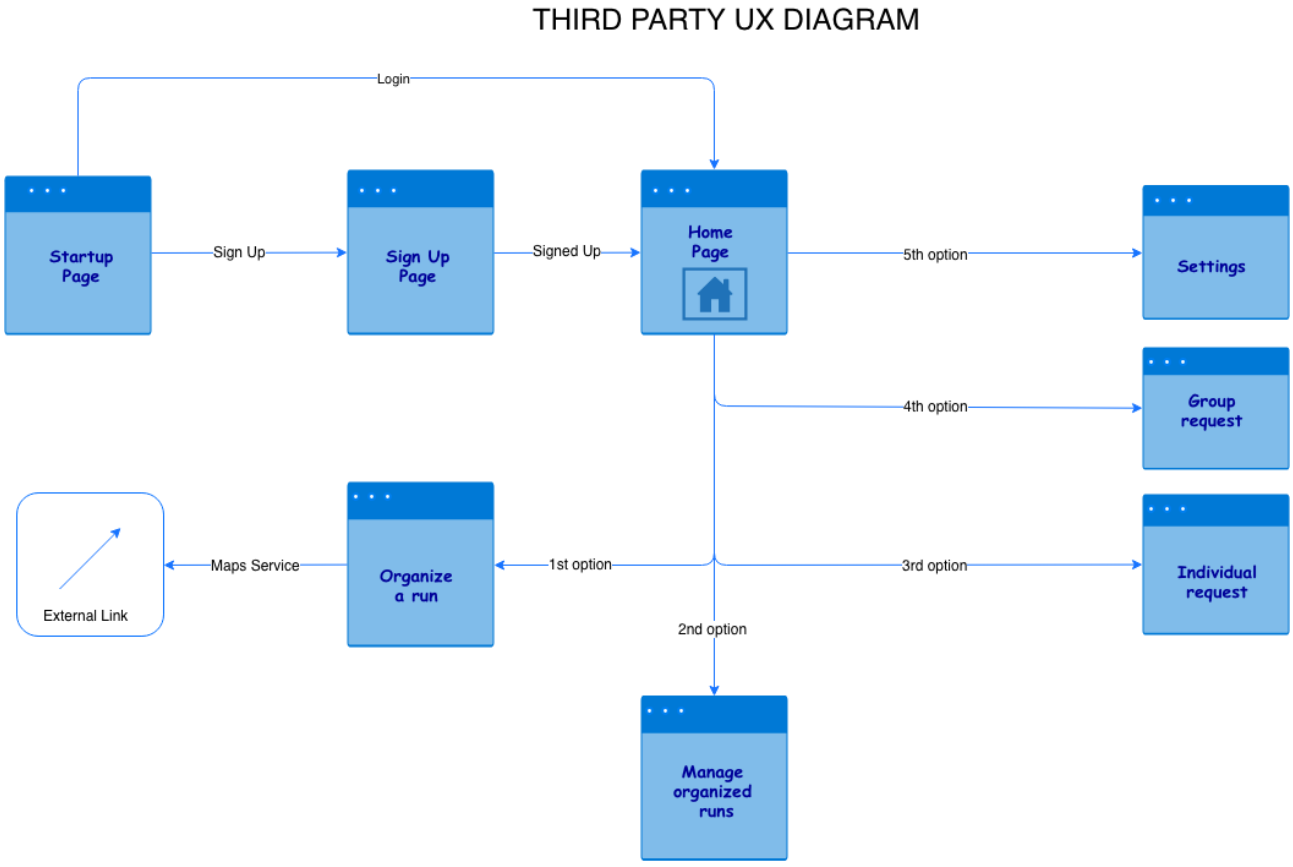


2.4.5. **Individual request Runtime View**

3. User interface design

3.1. UX Diagram





For the user interface design, we refer to section 3.1.1 of the RASD document. This section is aimed to enrich what was shown in the RASD through two User Experience diagrams. Their main goal is to explain as clear as possible the relationships between the various sections of the application (addressed to

the Individuals) and the various pages of the website (addressed to the Third parties).
The two diagrams logically share the operations that allow the client to log into the application:

- Startup Page;
- Login;
- Sign Up.

Anyway, their presentation will be different due to the Mobile configuration for the Individuals and the Web based configuration for the Third Parties. The two diagrams start diverging from the principal window of the application/website, that is the Home Page. In the Individual UX Diagram, it is shown that through the Home Page the user can choose different options:

- Join a run, that allows the user to visualize the available races and select the one he wants to join;
- Manage requests, that allows the user to accept/refuse the pending requests coming from the Third parties or visualize/modify the already accepted requests;
- Open settings, that allows the user to modify his/her password, personal info, manage the connection between the smartphone and the smartwatch (or a similar device) or enable/disable the Automated-SOS service.

In the Third Party UX Diagram, the Home Page is essentially different from the other one and it allows the third party to choose the following options:

- Organize a run, that allows the third party to select che date, time and place of the event. Moreover, it is possible to select the race track thanks to the use of the Maps external service;
- Manage runs, whose goal is to allow a third party to visualize and check/modify the status of the runs organized by itself;
- Send an Individual request, through which the third party can directly ask a user for his health data, or subscribe to them;
- Send a Group request, through which the third party can ask the TrackMe system for aggregated data regarding more than 1000 users grouped by age and residence;
- Open settings, that allows the Third party to modify its password.

4. **Requirements traceability**

- **Third Party Request Handler**
  - R1) The users must have given the consensus to the treatment of their information to the third party;
  - R2) The system must be able to provide to the third party the location and the health status of individuals;
  - R4) The groups must be composed at least by 1000 individuals;
  - R5) The system must be able to provide to the third party the health status of individuals in an anonymous way;
  - R6) The system must be able to aggregate the data of the individuals, as requested by the third party;
  - R9) The third party is not allowed to access the user's data until he/she accepts the request.
  - R11) The system is optimized to send the data received from the mobile application to the third parties as soon as possible.
  - R33) The system must allow the Third party to change its password.
- **Mobile Data Retriever**
  - R3) The system must be able to retrieve data from the smartwatches and similar devices;
- **Data and Storage Interfaces**
  - R27) The system must be able to store data retrieved from registered users.
  - R8) The system must save the preference of the user;
- **Notification Controller**
  - R7) The system must be able to forward the requests from the third party to the user;
- **User Controller**
  - R28) The user must have an active subscription to stop it;
  - R29) The system must be able to allow the user to unsubscribe to the third party and to stop the transmission of his/her data.
  - R26) The system must allow the user to enable/disable the AutomatedSOS service at any time.
  - R31) The system must allow the user to change his/her personal info.
  - R32) The system must allow the user to change his/her password.
- **Authentification Controller**
  - R12) The users must provide their personal data to the application during the registration process, SSN (or fiscal code) included;
  - R13) The system must allow the user to register to the application by selecting a username and a password;

  - R14) The system must allow the user to log in to the application by providing the combination of a username and a password that matches an account;

  - R15) Two different users cannot have the same username.

  - R16) The system must allow the third party to register to the application, by specifying its VAT registration number and a password;

  - R17) The system must allow the third party to log in to the application by providing the combination of a VAT registration number and a password that match an account;

- **Track4Run third party controller**
  - R19) The system must be able to retrieve the position of all the runners;

  - R24) The system must allow the third party to organize a race by de
ning its track andits time.

- **Health checker**
  - R18) When the health status values go below the threshold, the system must send an SOS within 5 seconds;

  - R25) The AutomatedSOS service must be enabled.

- **Track4Run user controller**
  - R20) The system must be able to provide the position of all the runners in the track in real time.

  - R21) The system must allow the user to check the list of available races at any time.

  - R22) The system must allow the user to join an available race only before its starting time.

  - R23) The user cannot join two different overlapping races.

  - R30) The system must avoid the registration of users after having reached of the maximum number of participants.

5. **Implementation, integration and test plan**

6. **Effort spent**

- **Paolo Romeo**

Description of the task	**Hours spent**
Architectural design	7
User interface design	2
Requirements traceability	2
Implementation, intergration and test plan	6
Correction of the RASD	3
Other related activities	5

- **Andrea Scotti**

Description of the task	**Hours spent**
Architectural design	8
User interface design	2
Requirements traceability	2
Implementation, intergration and test plan	6
Correction of the RASD	2
Other related activities	5

- **Francesco Staccone**

Description of the task	**Hours spent**
Architectural design	7
User interface design	4
Requirements traceability	2
Implementation, intergration and test plan	5
Correction of the RASD	3
Other related activities	4

7. **References**