POLITECNICO
MILANO 1863

TrackMe project - Argiro' Anna Sofia,
Battaglia Gabriele, Bernardo Casasole

Design Document

Deliverable:	DD
Title:	Design Document
Authors:	Argiro' Anna Sofia, Battaglia Gabriele, Bernardo Casasole
Version:	0.4
Date:	December 8, 2018
Download page:	https://github.com/BernardoCasasole/ArgiroBattagliaCasasole.git

Contents

1. Introduction

1.1 Purpose

This document means to provide developers and stakeholders with detailed information regarding the architecture of Data4Help software. A particular effort has been devoted to testing plan and integration testing plan, since the Integration Test Plan Document is not required.

1.2 Scope

Data4Help means to provide services to authenticated users only. Those services are addressed to both:

- Individual Users
- Third parties Users

To dispatch specific functionalities to the user they are reserved, Data4Help System avails itself of:

- a Mobile App, reserved to individual users
- a Web Page, reserved to third party users

The mobile app, using the GPS location provided by the smartphone, allows the individual user to:

- check his own health parameters (measured by a smartwatch)
- enable and disable additional services (AutomatedSOS and Track4Run)
- give or deny authorization to every third party to access health data about himself

Data4Helps System handles both data of the past and real time ones. The web page allows the Third-party user to:

- make requests for statistical data of the past or real time
- make requests for individual data of the past or real time (the requests are forwarded to the individual user)
- organize and watch run competitions

This factorization allows the system to be accurate in providing every user with all and only resources he has the right to access: authentication and authorization processes rely on the access control.

The necessity to use a mobile app could prevent third parties from choosing Data4Help over other services of data collection: Data4Help Web Page can be easily accessed from a browser hosted on a computer or a mobile.

1.3 Definitions

- *User*: a person, third-party or user, that has registered;
- *Individual User*: every registered person from whom the system collects data;
- *Third-Party User*:every entity registered with the purpose to request data for external use;
- *non-human Third-Party User*: a software Third-Party User that access to the offered D4H services thorugh the exposed APIs
- *Live Data*: the data on a IU produced in real time.
- *Stored Data*: the data on a IU collected so far.
- *Data Request*: a request for data made from a TPU.
- *Stored Data Request*: a data request for stored data.
- *Subscription Request*: a request for subscribing to newly generated data.

1.4 Acronyms

- API: Application Programming Interface
- TPU: Third-party User
- D4H: Data4Help
- ASOS: AutomatedSOS
- T4R: Track4Run
- UX: User experience
- REST: REpresentational State Transfer

1.5 Abbreviations

- Gn: n-goal
- Dn: n-Domain assumption
- Rn: n-Requirement

1.6 Revision history

- **v0.1 - 27/11/18** Document created
- **v0.2 - 30/11/18** Component view
- **v0.3 - 2/12/18** Model diagrams, User inteface and High level overview
- **v0.4 - 8/12/18** Architectural patterns, interfaces, deployement, high level architecture review

1.7 Document Structure

Introduction
This section means to present briefly the software and the world its going to live in. The terminology that is going to be used through the document is specified.

Architectural Design
This section illustrates:

- high level components and their interaction
- main components the system is divided into and their interaction
- diagrams reporting entities relationship
- different representation of the model that highlights the interaction with server, its formalization in the deployment view
- sequence diagram showing the runtime behavior
- component interfaces
- patterns.

User Interface Design
In this section each user interface (presented in RASD) is reported and accurately explained. The interaction among interfaces clarifies the path to reach every interface starting from any other.

Requirements Traceability
This section means to map all the requirements with the corresponding design components.

Implementation, Integration and Test plan
This section reports how the implementation of components has been designated, including: the implementation order, the integration order, components testing, components integration testing. Details about implementation and testing are provided.

Effort Spent
This section reports, with a tabular representation, the effort spent by each member of the group.

References
This section is a list of documents and web sites consulted in order to realize the Design Document.

2. Architectural Design

2.1 Overview

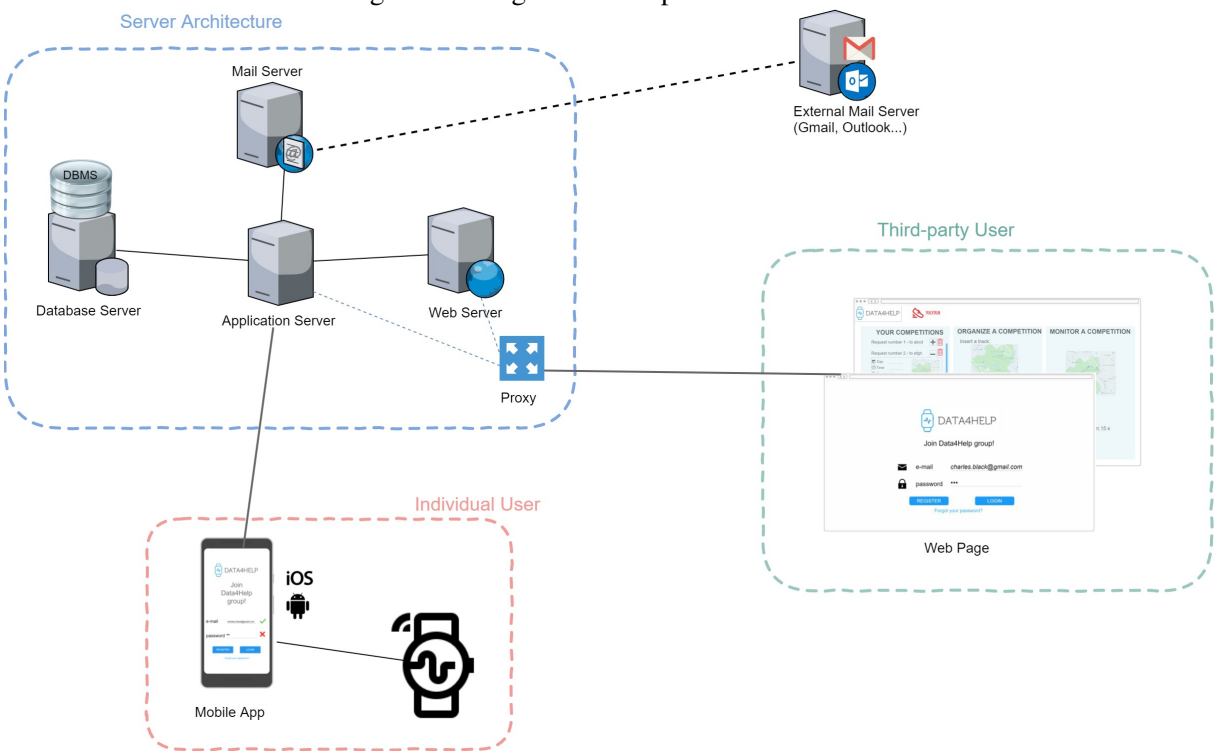
The architecture style used is a client/server structure with multiple tiers while an event-backbone will handle the dispatch of live data through the system. The presentation layer will be hosted on both client (IUs and TPUs clients) while the application server will host the logic layer and the database server the data layer. The IU client is going to be a thick client, hosting a brach of the application logic to handle better and faster the system functionalities.
A server structure can either be hosted locally on a dedicated physical server or on a cloud server furnished by a cloud-server-hosting provider. Cloud server solutions, over local servers, are the ideal fit for system having variable demands and workload (such as Data4Help) and meet the following needs:

- to avoid hardware faults: Data4Help availability percentage needs to be equal or superior to 99.995. Since this percentage do not take into account hardware faults and a local server cannot guarantee an hundred percent availability, Data4Help availability cannot be affected by the one of a local server
- to enhance security
- to only pay for the exact amount of server space used
- minimization of data losses and recovery time: the availabitly of a system is affected by recovery (restart and repair) time. Data4Help system, to guarantee an availability of 99.995 percent, needs to avail itself with a recovery-time minimize solution

Many cloud-server-hosting providers exist, solutions furnished are quite similar and valid. Google Cloud Platform might be chosen over other cloud-server-hosting providers because of the possibility to use both SQL and NoSQL databases. Its up to developers to evaluate the provider to use, according to the budget assigned and a future possible differentiation of services that nowadays seems to offer similar solution at the same price. Data4Help System can avail itself of a CDBMS which is a Database Manager as a cloud service: this solution is designated to be scalable and flexibles and to run on a cloud server architecture. Google Could Platform offers Cloud SQL which is a database services fully integrated with MySQL and PostgreSQL(beta).

2.1.1 High level components and basic interactions

Figure 2.1: High level components' elements

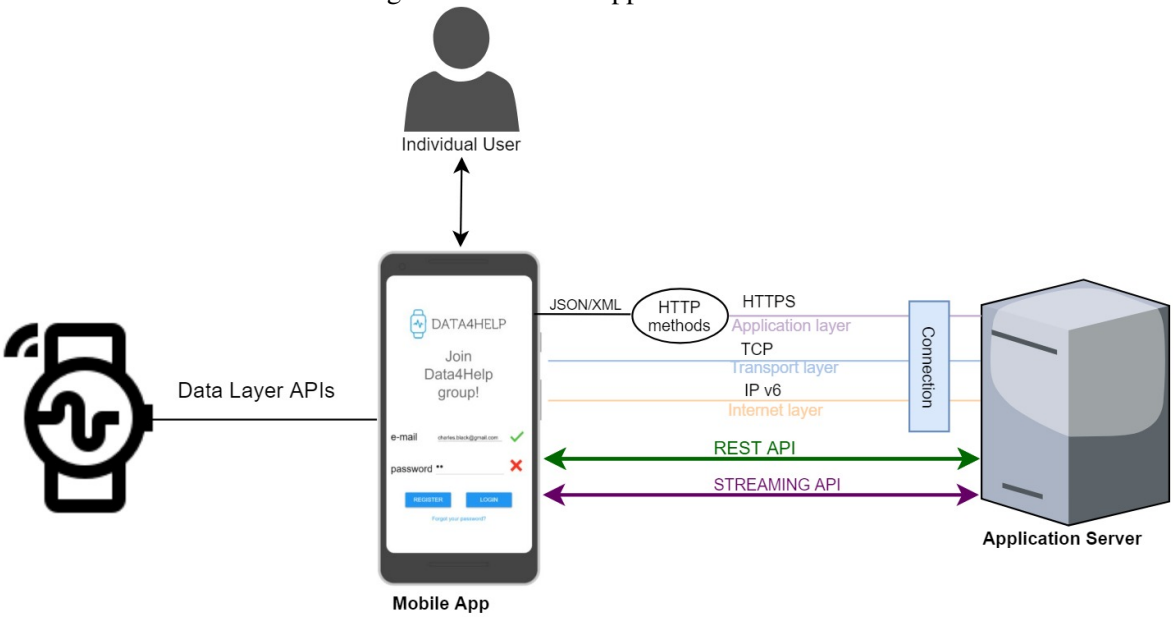


The overall structure, at high level, is made of three main components and their interaction.

The component on the bottom shows Data4Help mobile application and a smartwatch which are the tools that allows the Individual User to interface with Data4Help System. The mobile application communicates with the Application Server which is part of the component that represents the Server Architecture. This last is composed by a Database Server that includes the DBMS, a Mail Server which means to exchange SMTP messages with other Mail Servers (external to the system), an Application Server communicating with any other element in the Server Architecture, a Web Server and a Proxy (meant to dispatch requests to Application and Web Servers). The proxy links the Server Architecture with the green component charged with the interaction with the third party user that takes place through Data4Help Web Page.

2.1.2 Interaction between Server Architecture and Individual User

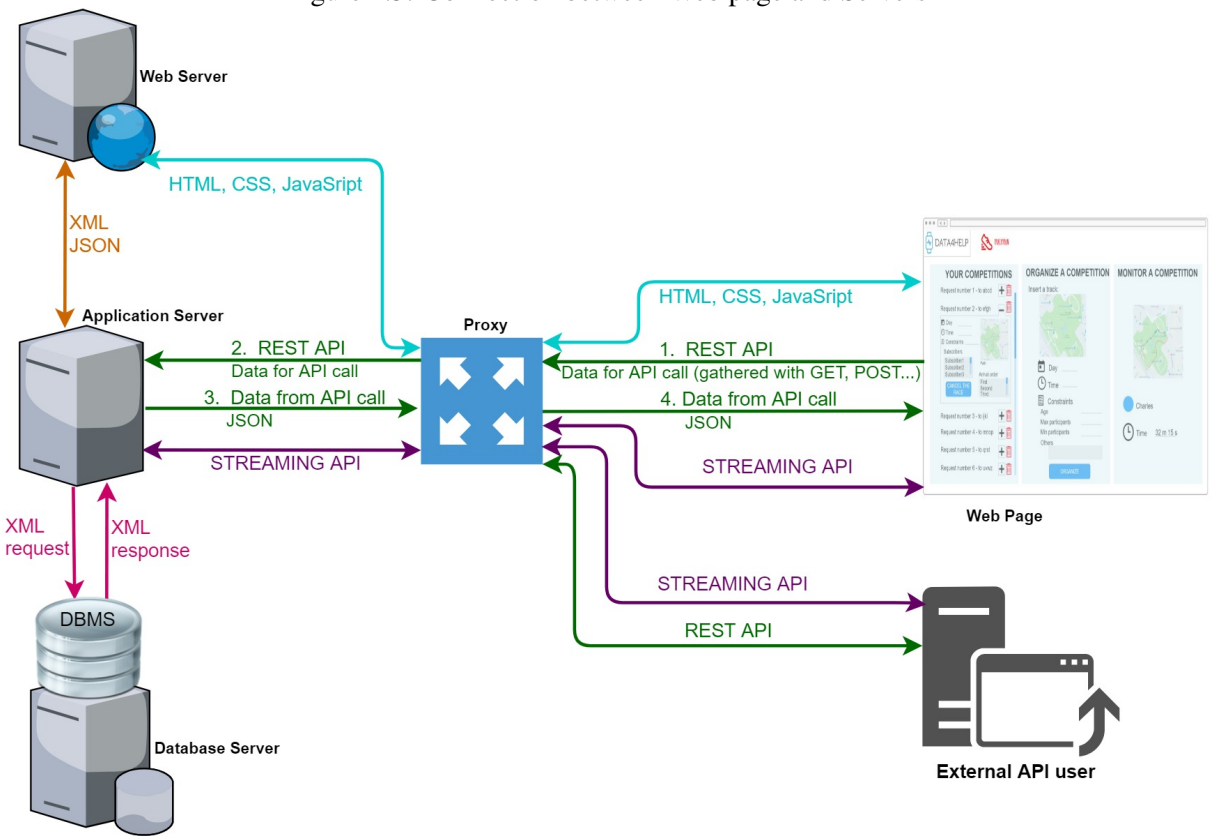
Figure 2.2: Mobile App connections



Data4Help Mobile App receives data from the smartwatch and exchanges informations (using differen modalities) with the Individual User and the Application Server. At 3 different levels are specified protocols that are supposed to generate the connection with the Application Server; a STREAMING API communication need to be established to exchange real time data, all other information to be exchanged use REST API communication.

2.1.3 Interaction between Server Architecture and Third-Party User

Figure 2.3: Connection between Web page and Servers



The browser hosting the Web Page needs to communicate both with the Web Server and the Application Server. The Web Server can easily handle and exchange HTML, CSS and JavaScript files with the client; the Application Server manages methods like GET, POST receiving a REST API call (used to exchange every information but real-time data) and forwarding data in JSON format. Real time data are exchanged using STREAMING APIs: both Data4Help web page and an external-API-user request and receive real time data. Data to forward are provided by the Database Server which includes the DBMS: a request in XML is sent by the Application Server, the DBMS processes the request and extracts data from the database that are sent back to the Application Server in a XML file. To establish a communication channel between the Application Server and the Web Server is not a necessity, however it provides an alternative to REST API: developers are up to decide to implement them both or to keep the REST and STREAMING APIs alone.

2.2 Component view

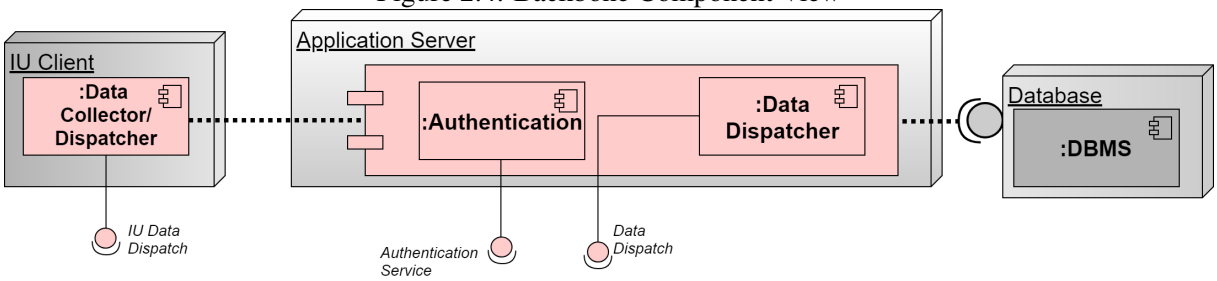
The system is divided in four subsystem:

- **Backbone**
- **Data4Help**
- **AutomatedSOS**
- **Track4Run**

The Backbone is the core of the system: all other subsystems interact with it and don't interact with each other. The backbone provides interfaces for authentication and to receive live data published form the Backbone with a event-based paradigm.
The last three are divided, on the Application server, in a router that provide an interface gathering all the subsystem functionalities, and a module, containing all other components of the subsystem, which uses the exposed method of the DBMS to be able to work indipendently.
On the IU and TPU clients the view component represent the presentation layer of the system, which Users can access directly.
The relation between the components and the model il further defined in figure 2.10.

2.2.1 Backbone

Figure 2.4: Backbone Component View



This is the backbone of the system: collects the data on the device, keep it syncronized though the system, stores it onto the database and provide the functionalities to receive live data; Furthermore provide functionality concernig authentication.

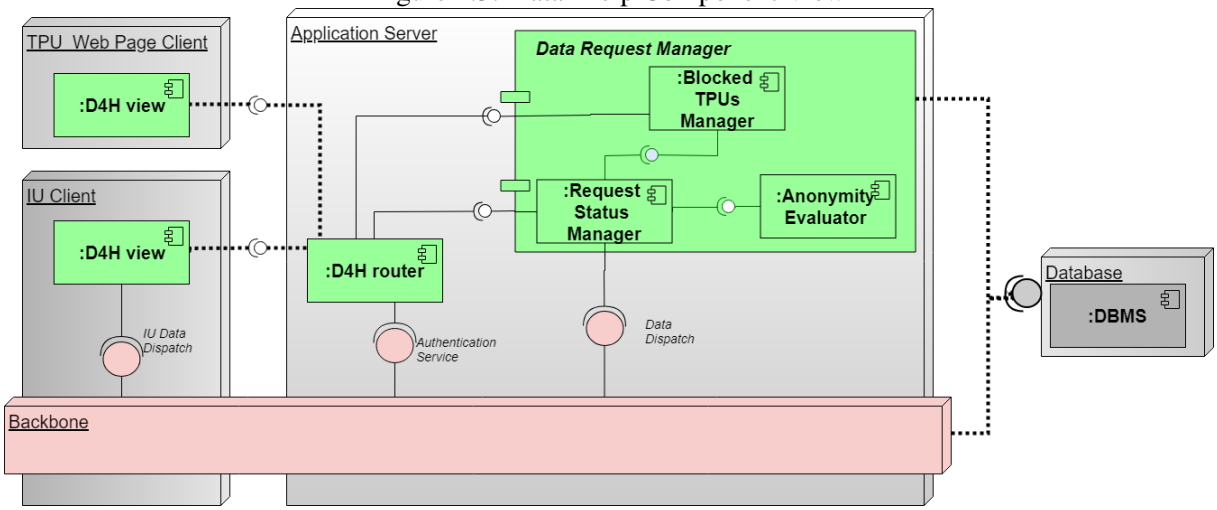
Data collector/dispatcher Allow subscribtion from other components on the IU client and publishes/dispatches the collected live data of the Individual User logged in from the device.

Autenthication Offers services related to User authentication and the functionalities to handle their info.

Data Dispatcher Allow subscribtion from other components on the application server and publishes/dispatches the collected live data of all Users and it stores it onto the database.

2.2.2 Data4Help

Figure 2.5: Data4Help Component View

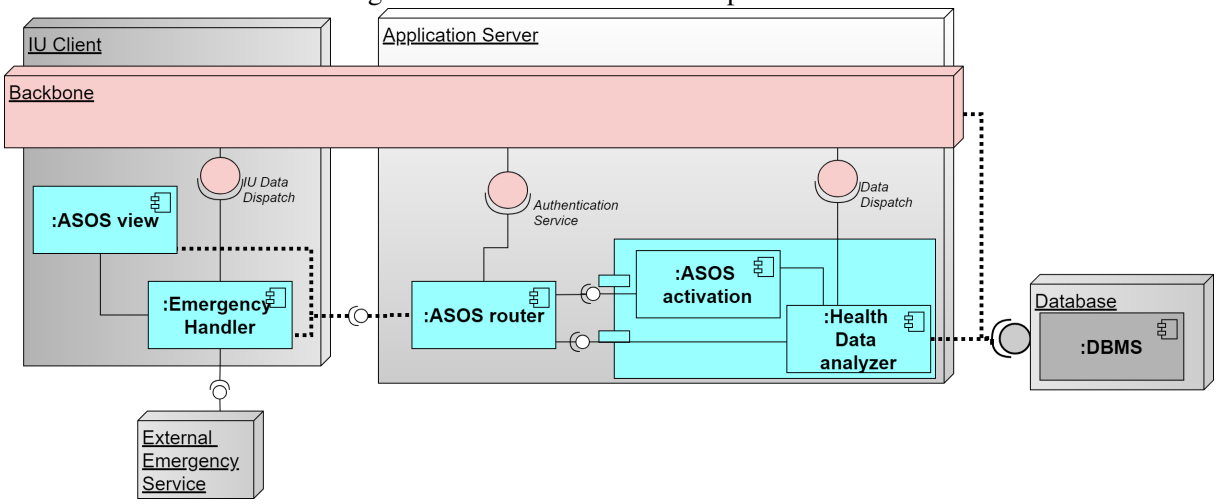


D4H router Validate the requests received from the client and dispatch them to the corresponding module or component.

Data Request Manager Provides functionality to create, approve, deny requests, block users and provide the relative data; Anonymity Evaluator is responsible to check anonymity constraints.

2.2.3 AutomatedSOS

Figure 2.6: AutomatedSOS Component View



ASOS router Validate the requests received from the client and dispatch them to the corresponding module or component.

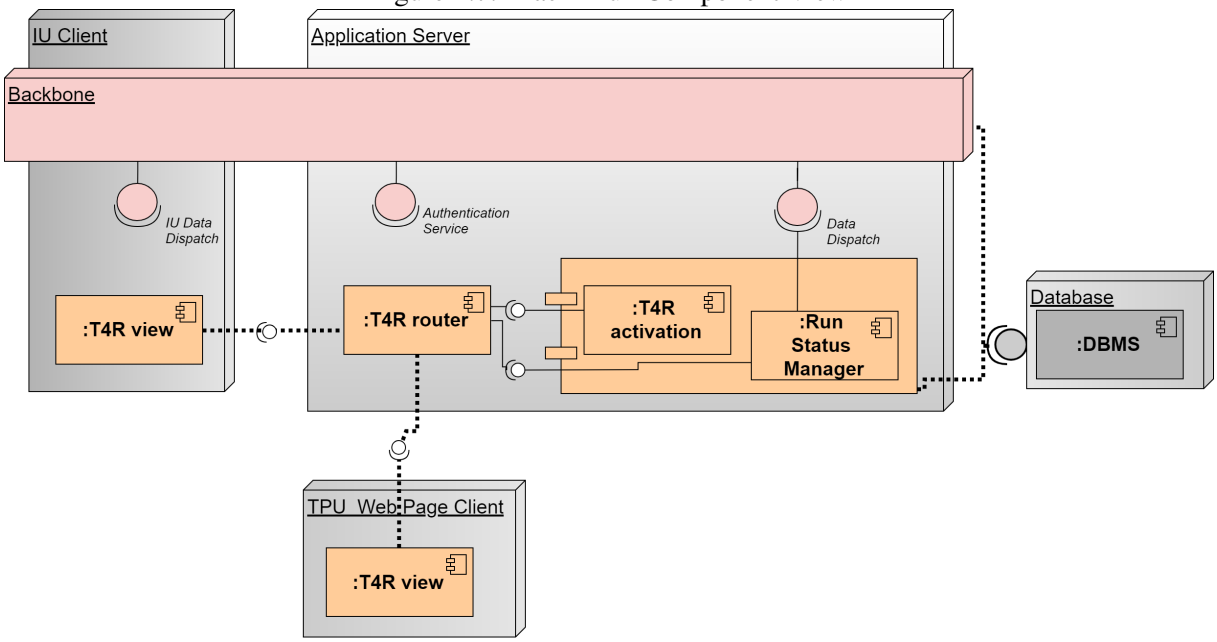
ASOS Activation Offers the functionality for the activation and deactivation of the ASOS service.

Health Data analyzer Offers functionality to extrapolate the critical health parameters for every Individual User;

Emergency Handler Responsible to handle critical health conditions based on the data published by the *Data collector/dispatcher*

2.2.4 Track4Run

Figure 2.7: Track4Run Component View



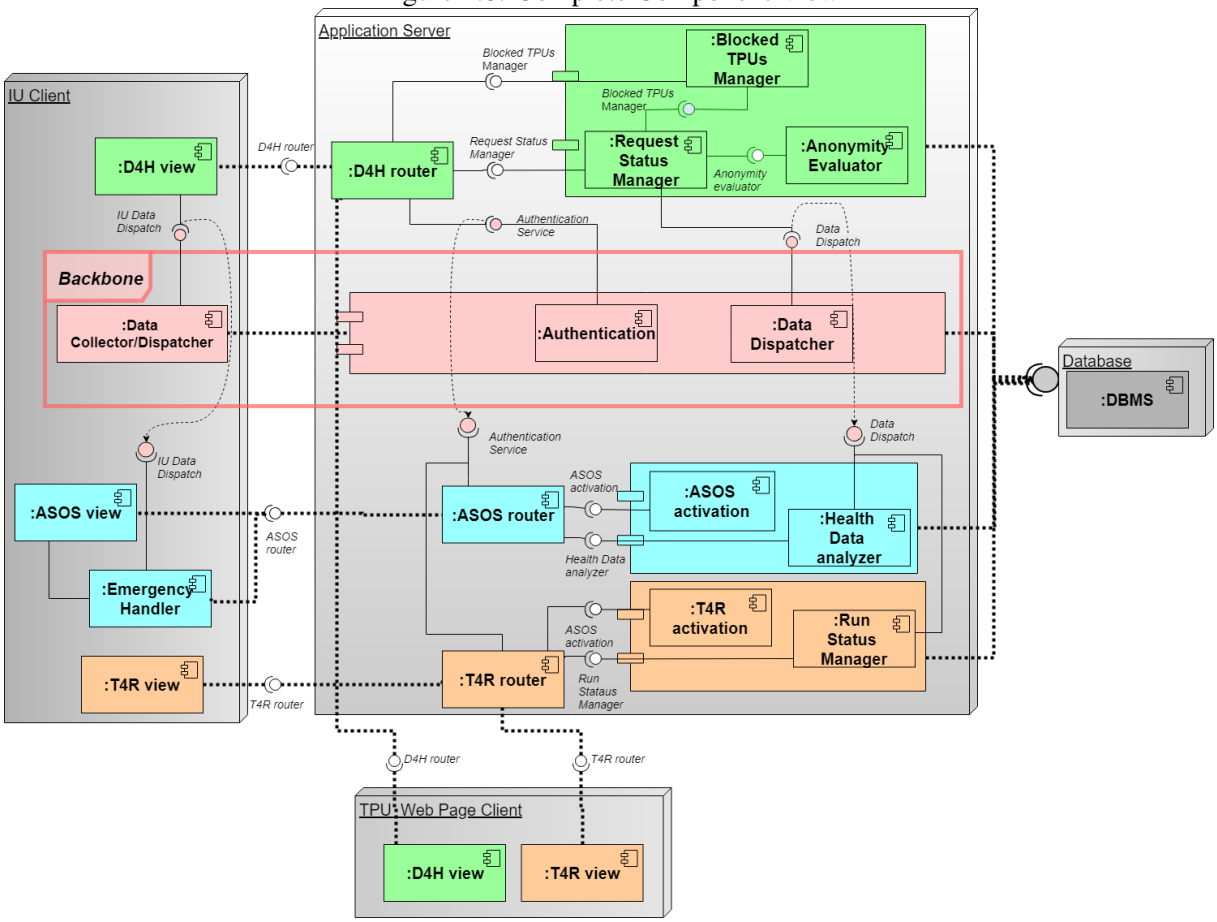
T4R router Validate the requests received from the client and dispatch them to the corresponding module or component.

T4R Activation Offers the functionality for the activation and deactivation of the T4R service.

Run Manager Provides functionality to create, cancel and enrol in runs.

2.2.5 Full system

Figure 2.8: Complete Component View

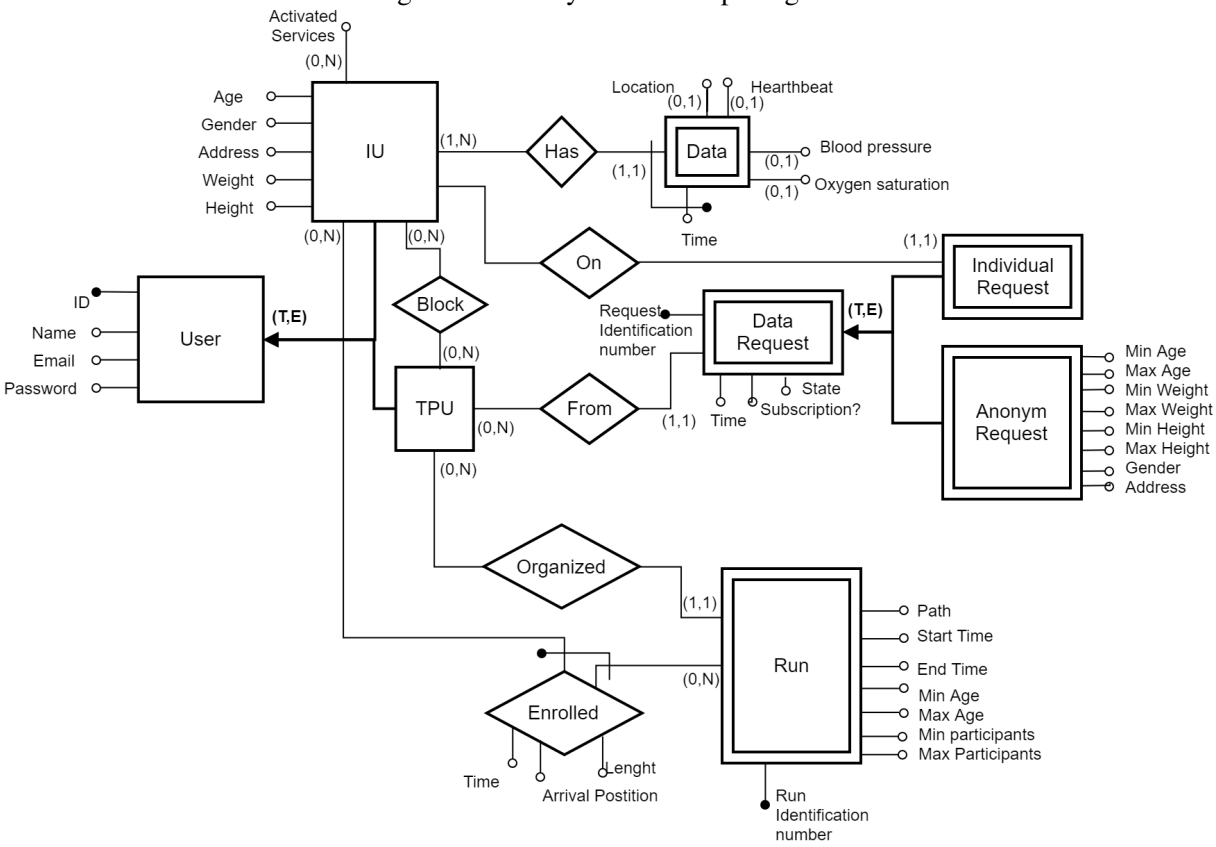


Data Managing From a more high level point of view, the backbone provides services to retrive the Individual Users live data.
This makes the red components and modules of the architecture the backbone, collecting and dispatching data, while the other subsystems can handle their unique authorization condition: D4H authorizing data dispatching based on approved requests, ASOS on the activation of the service and T4R on the enrollement in competitions.
This way all subsystem can work independently from each other.

2.2.6 Entity Relationship Diagram

The following section provides a conceptual representation of the model.

Figure 2.9: Entity Relationship Diagram



Tables

- ***User***(ID, Name, Email, Password)
- ***TPU***(ID, Name, Email, Password)
- ***IU***(ID, Name, Email, Password, Age, Gender, Address, Weight, Height)
- ***Data***(IU, Time, Location, Heartbeat, Blood pressure, Oxygen saturation)

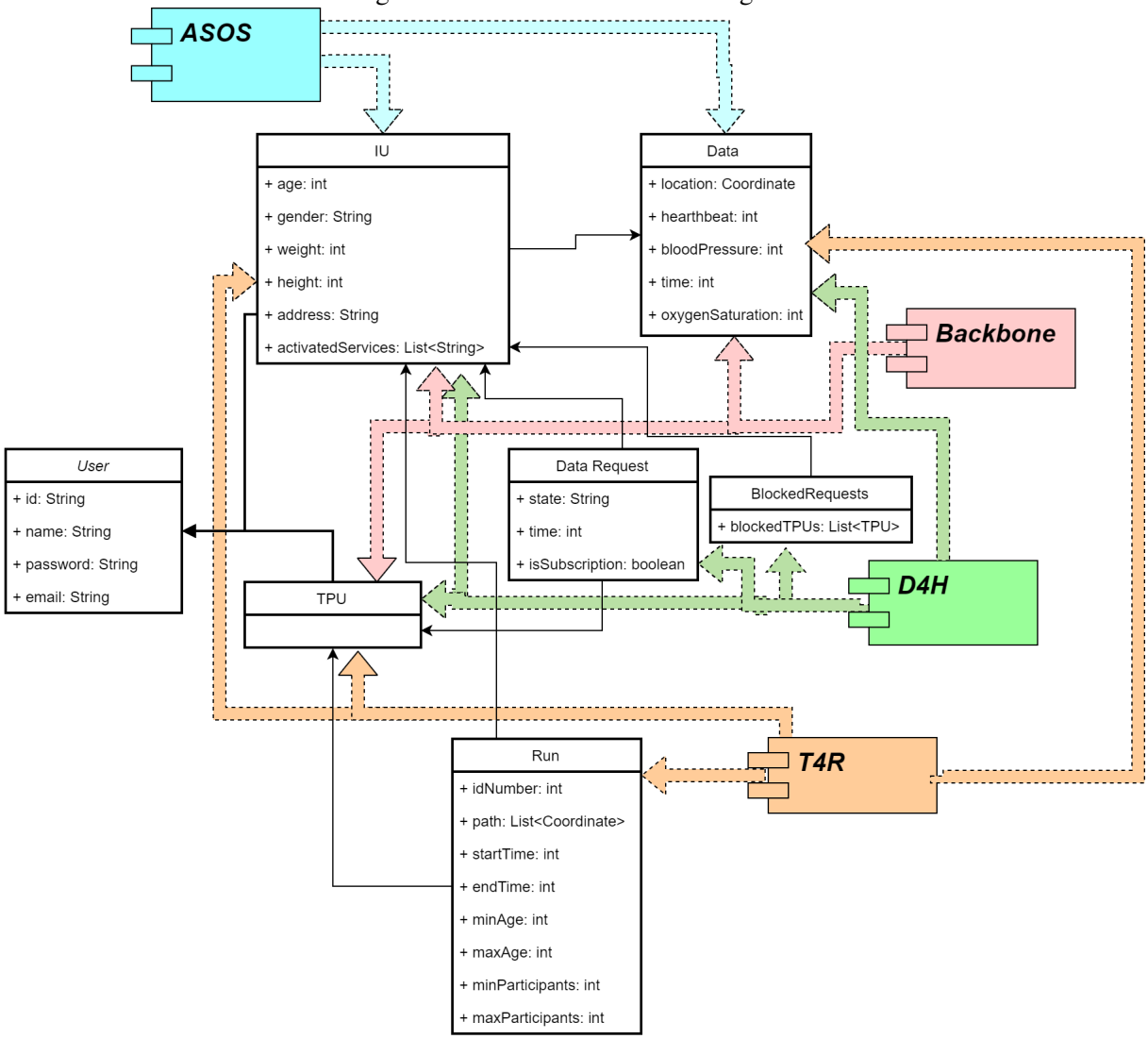
- ***Individual Request***(Request Identification Number, IU, TPU, Time, State, Subscription?)
- ***Anonym Request***(Request Identification Number, TPU, Time, State, Subscription?, Min Age, Max Age, Min Weight, Max Weight, Min Height, Max Height, Gender, Address)

- ***Run***(Run Identification number, TPU, IU, Path, Start Time, End Time, Min Age, Max Age, Min participants, Max Participants)
- ***Run Result***(Run Identification number, IU, Lenght, Time, Arrival Position)

2.2.7 Model Interaction Diagram

The following diagram show a different representation of the model to better highlight its interaction with the application server. For each subsystem module that was connected to the DBMS in 2.2.5 is shown its relationship with the module.

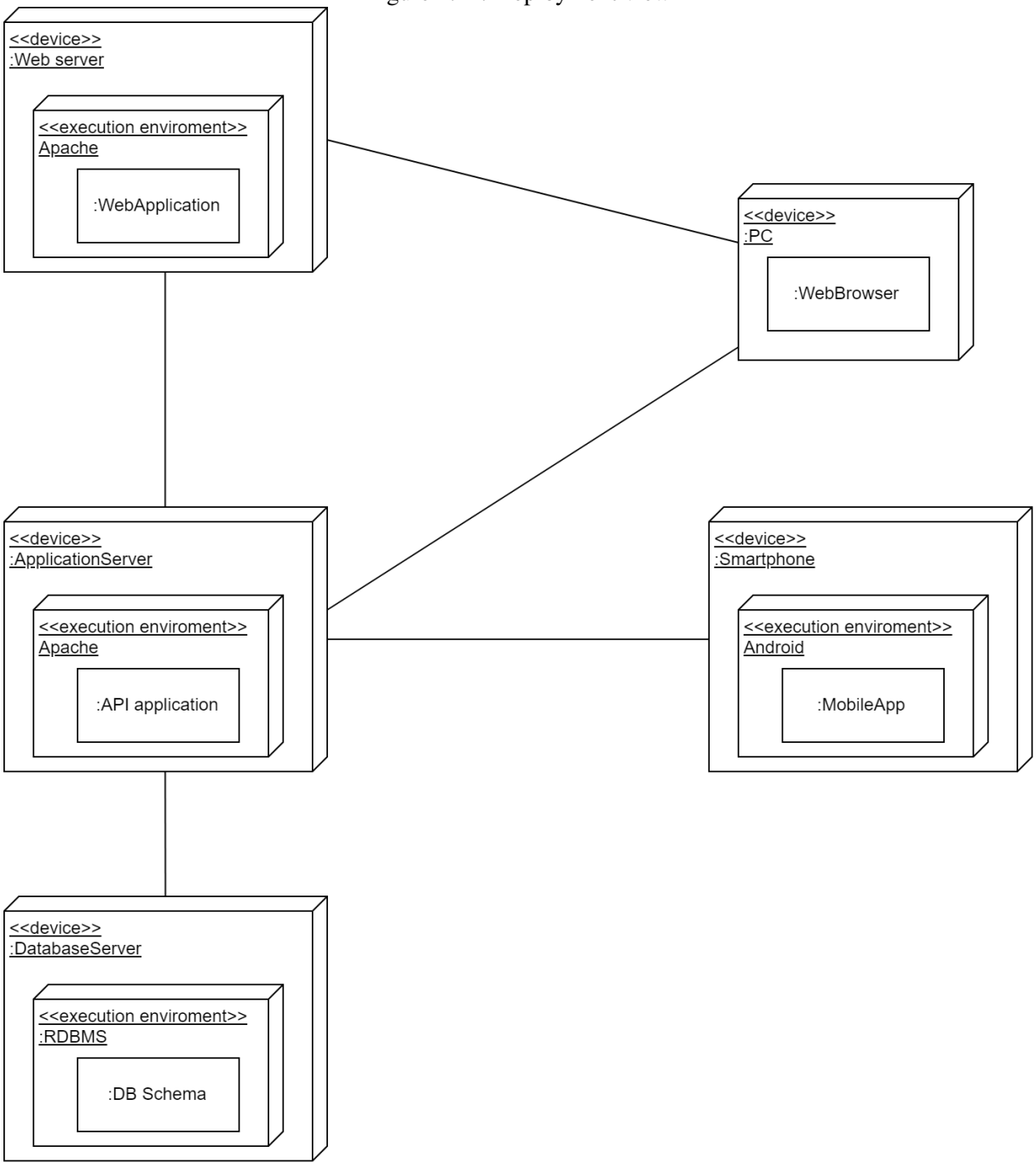
Figure 2.10: Model Interaction Diagram



2.3 Deployment view

As stated in the previous sections the system is composed by the two clients, one hosted on a web browser and the other on moblie application. They both rely on the application server while the former also interacts with the web server which host the web application. The application server provide the logic of the system and interacts with the database server which hosts the data layer of the system.

Figure 2.11: Deployment view



2.4 Runtime view

Figure 2.12: IU Registration

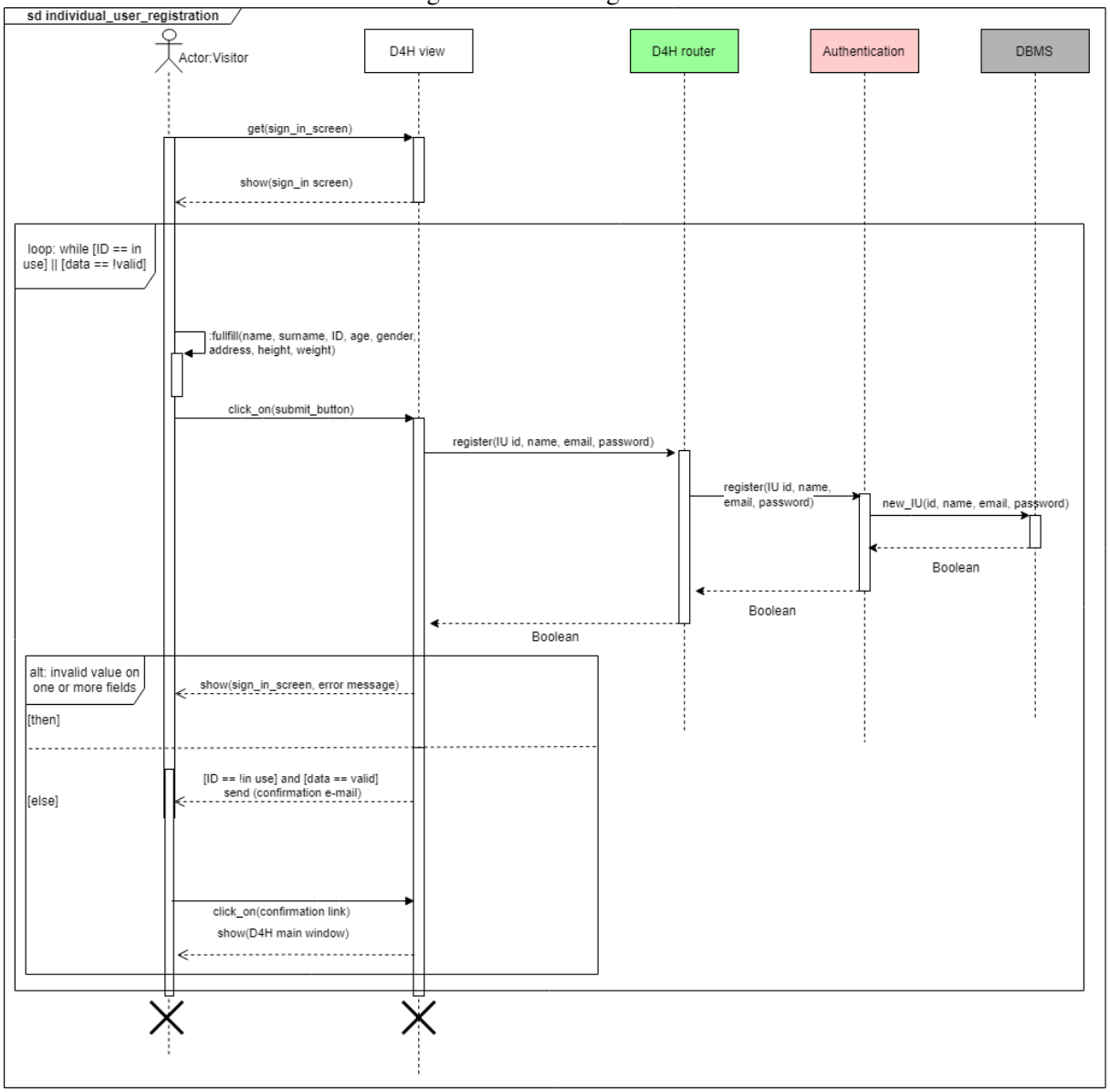


Figure 2.13: Data Requests

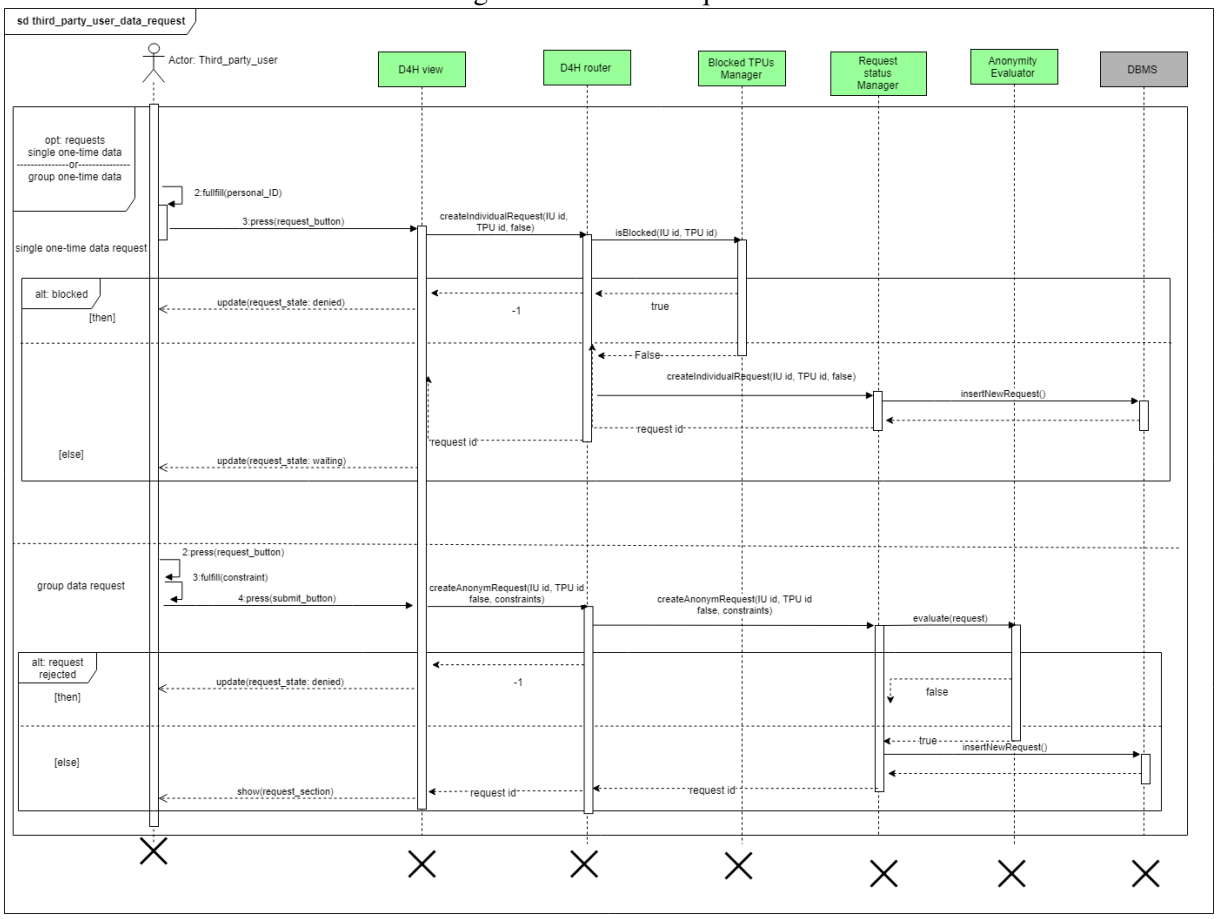
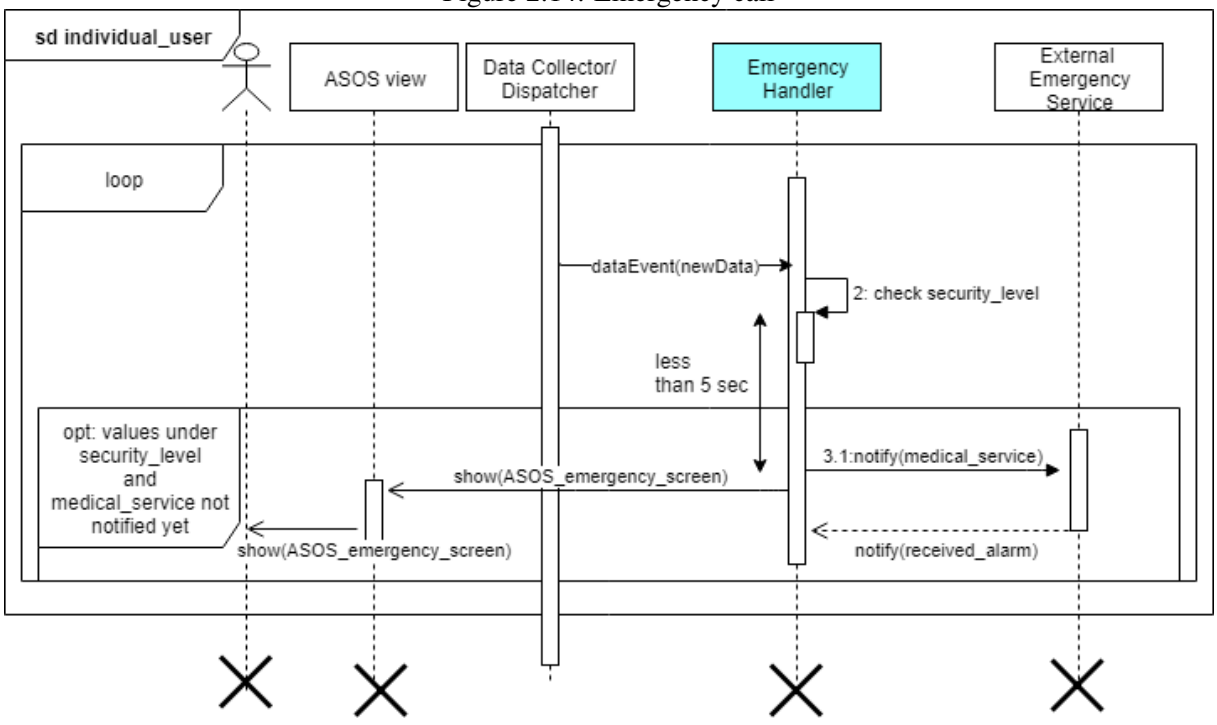


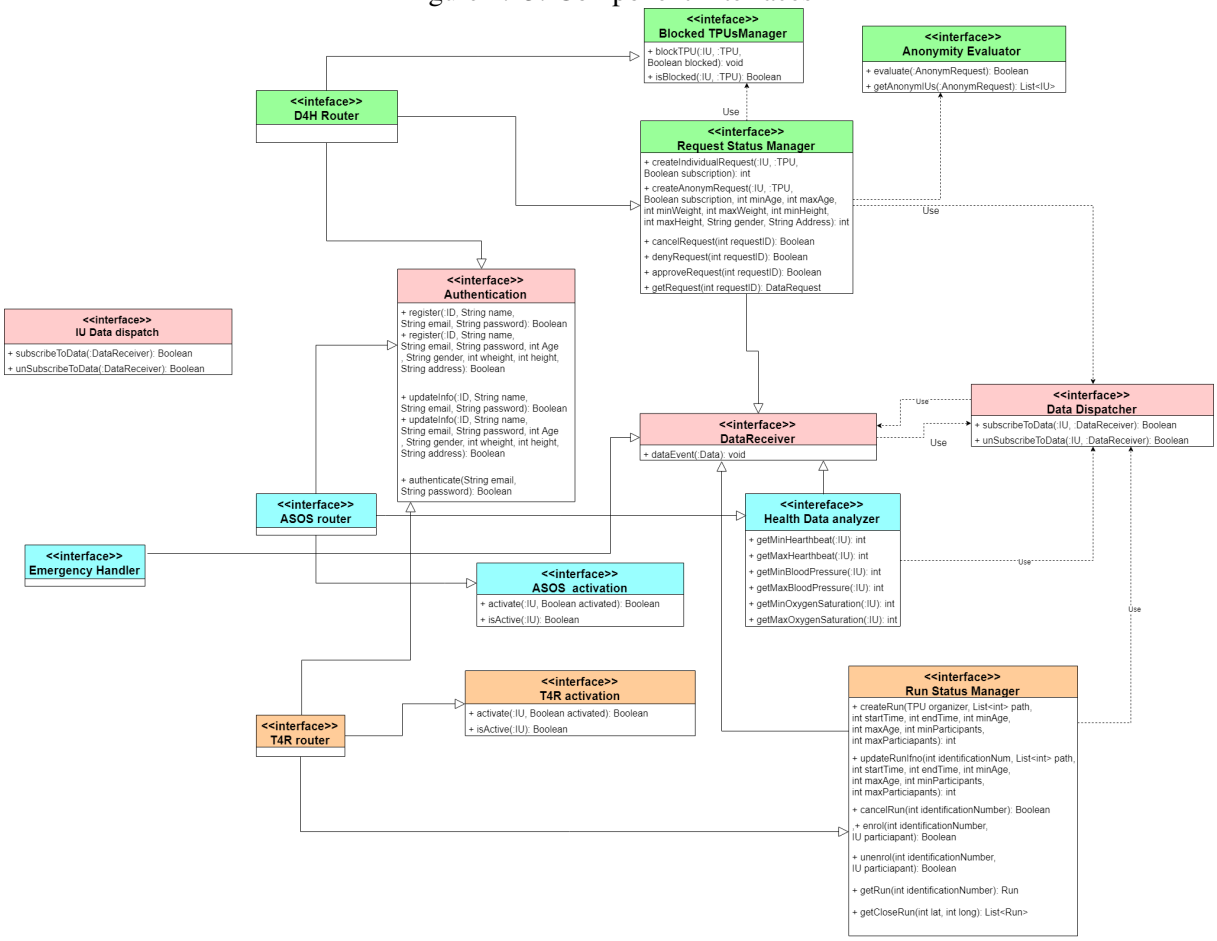
Figure 2.14: Emergency call



2.5 Component interfaces

The next diagram shows the most important methods of the components interfaces which, for clarity, are named in figure 2.2.5 tracing the components names.
The routers gather oll the method required to provide the client with the corresponding subsystem services and expose the relative APIs for the clients (for the D4H router also non-human TPUs) to use.
An generic interface *Data Receiver* is extended by all the interfaces that use the *Data Dispatcher* service, to receive the updates.

Figure 2.15: Component Interfaces



2.6 Selected architectural styles and patterns

Client/server multi-tier The architecture style chosen is a client/server structure with multiple tiers. The presentation layer is diveded between the two clients (IUs and TPUs clients) which are thick clients since they host a branch of the application logic to handle better and faster the system functionalities; namely, to provide the fastest possible emergency response time, the client directly handles critical conditions contacting the emergency service and the backbone handles the dispatching of the IU live data to other components on the client.
The application server hosts the logic layer, exposing API the clients to access the subsystem functionalities and, for the D4H router, to non-human TPUs which might access directly through the APIs; The application server is divided in four subsystems, each handling a piece of logic: a backbone, handling the core logic, storing data and user authorization, and providing interfaces to other subsystem to use its functionality, while the other subsystem independently handle the functionalities of the three services

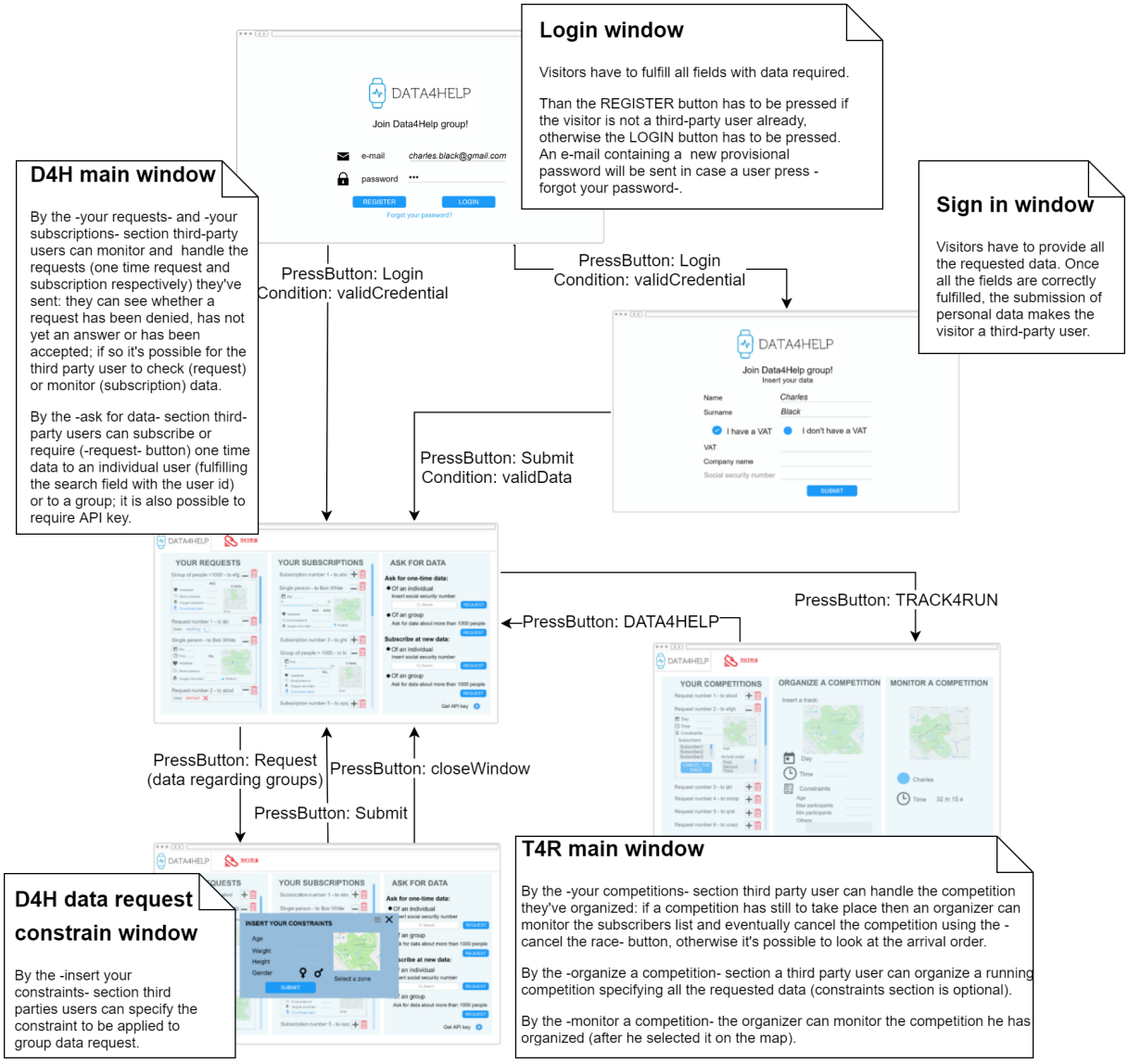
offered: D4H, ASOS and T4R.
The database server host the data layer and all the subsystems on the application server independtly interact with it.
This will make for a modular software, enabling a fairly independent implementation and testing of each subsystem; Morover it, alongside the tiered structure, will improve scalability and maintainability.

Event based paradigm The backbone, namely the Data Dispatcher components, is an event-based subsystem that handles the dispatch of live data through the system. Live data collected by the Data Collector/Dispatcher serves as the event, broadcasted to all registered components. While introducing potential scalability problems, it simplify the addition of the other subsystem.

3. User Interface Design

The user interfaces mock-ups are represented in sections 3.1.1, 3.1.2 of RASD. The following UX schemes represents a complete description of the user experience. The screen -T4R unsubscribe screen- has been added and a better description of each mock-up has been provided.

Figure 3.1: Third-party user - UX graphical representation



Third-party user The scheme above represents the main desktop screens and the way -condition and action needed- how the third-party user can move trough them.

Figure 3.2: Individual user - UX graphical representation (left part)

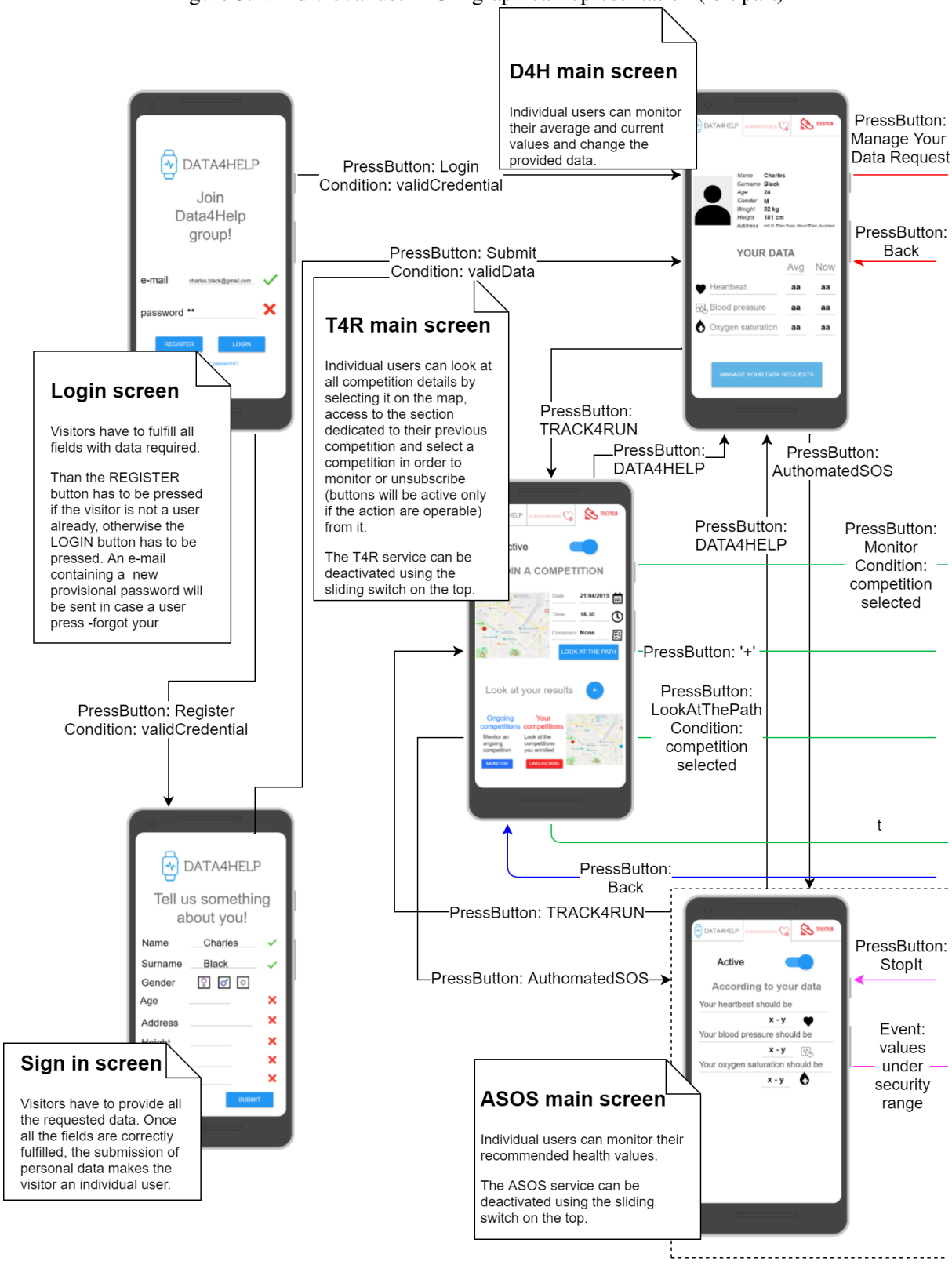
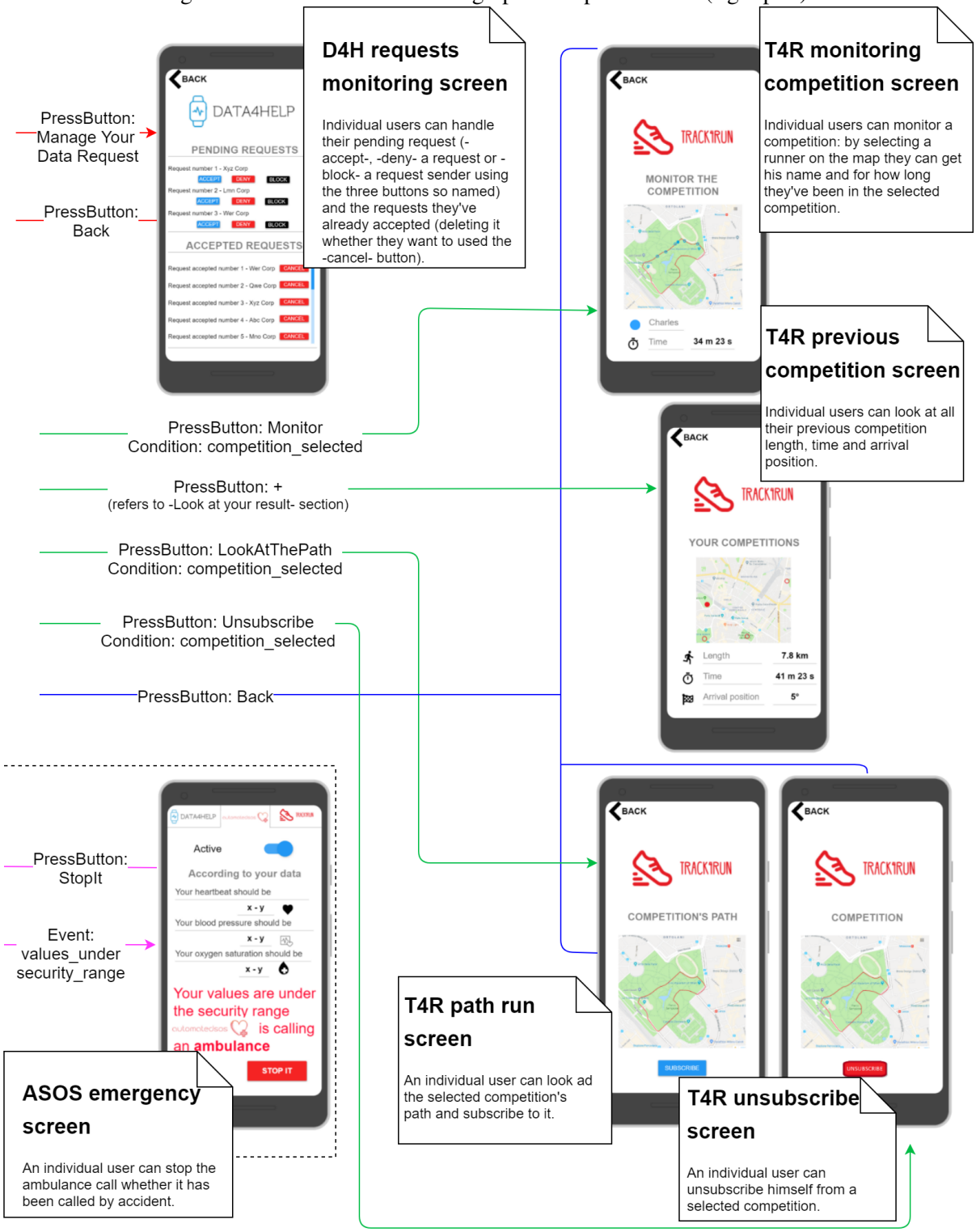


Figure 3.3: Individual user - UX graphical representation (right part)



Individual user The two schemes above represents the main mobile screens and the way -condition and action needed- how an individual user can move trough them. The scheme has been divided in two parts in order to provide a better readability.

4. Requirements Traceability

In the following pages requirements are mapped with their respective components, goals -requirements refers themselves to- are also reported. For every requirement are specified major and secondary (auxiliary) components according to their relevance for the specific requirement. When multiple requirements (associated to the same goal) need to be mapped to the same set of components, those requirements are reported in sequence and followed by the set ot components.

G1 Allow users to properly use the services they wish to employ

R1 The Users must be able to log in

Major:

- Authentication

Secondary:

- Data Dispatcher
- DBMS
- IU Data Dispatcher
- D4H Components on Application Server

R2 An IU must be able to opt in and out of any additional service besides D4H

Major:

- T4R Activation
- ASOS Activation

Secondary:

- Data Dispatcher
- Authentication
- DBMS
- IU Data Dispatcher

G2 Allow visitor to register as individual or third-party user

R4 A visitor must be able to register to D4H

R4.1 A visitor must insert name, age, gender, height, weight, ID, address, email and password

R4.2 A third party must insert name, ID, email and password

Major:

- Authentication

Secondary:

- Data Dispatcher
- DBMS
- IU Data Dispatcher
- D4H Application Server

G3 Allow individual users to monitor their location and health parameters

R5 The system must provide to logged IU their current location and health parameters

Major:

- IU Data Dispatcher

Secondary:

- Data Dispatcher
- D4H view
- Authentication

R6 The system must collect and keep track of the location and health data

Major:

- Data Dispatcher
- IU Data Dispatcher

Secondary:

- Authentication
- DBMS

G4 Allow third-party users to request the data on specific users

R7 A logged in TPU must be able to request the stored data or subscribe to newly produced data, after providing an ID of the user

R7.1 A TPU must be warned whether the data they requested is available

Major:

- Data Request Manager

Secondary:

- Data Dispatcher
- Authentication
- DBMS

R8 A TPU must be able to obtain the data it has access to

Major:

- Data Request Manager

Secondary:

- Data Dispatcher
- IU Data Dispatcher
- Authentication
- DBMS

G5 Allow individual users to approve or deny the specific request for their data

R9 The IU must be notified when a TPU request their data

Major:

- Data Request Manager

Secondary:

- Data Dispatcher
- IU Data Dispatcher
- Authentication
- DBMS

R10 An IU must be able to accept the request on their data

R11.1 An IU must be able to deny access to their data

R11.2 An IU must be able to deny block a TPU to automatically deny their future requests for data

R12 User must be able to cancel the TPU subscriptions to their data that they previously accepted

Major:

- Data Request Manager

Secondary:

- Data Dispatcher
- IU Data Dispatcher
- Authentication
- DBMS

G6 Allow third-party users to request data on anonymized groups of individual users

R13.1 A logged in TPU must be able to request the stored data or subscribe to newly produced data, after providing the specifics of the group it is interested in

R13.2 A TPU must be warned whether it is possible to properly anonymize the data requested

Major:

- Data Request Manager

Secondary:

- Authentication
- Data Dispatcher

G7 Call an ambulance if the system detects a critical health condition

R14 When a critical condition is registered, the system should contact the ES and provide it location and health status of the IU

Major:

- Health Data analyzer
- Emergency Handler

Secondary:

- ASOS view
- ASOS router

R15 The IU must be able to cancel the call

Major:

- ASOS view
- ASOS router

Secondary:

- Emergency Handler

G8 Allow third-party users to organize running competitions

R16.1 A TPU must be able to create a new run, specifying time, duration, path, restriction for enrolment

R16.2 A run organizer must be able to cancel a run

Major:

- Run Status Manager

Secondary:

- T4R router
- Autenthication

R17 Run organizers, participants and spectators must be able to see the status of the ongoing run

Major:

- T4R view

Secondary:

- Run Status Manager
- T4R router
- Autenthication

G9 Allow Individual users to enrol in existing running competitions as participants

R18.1 An IU must be able to enrol in a future run

R18.2A participant must be able to cancel their inscription to the run

Major:

- Run Status Manager

Secondary:

- T4R router
- Autenthication

G10 Allow Individual users to subscribe in existing running competition as spectators to monitor underway competitions

R19 An IU must be able to spectate a ongoing run

Major:

- T4R view

Secondary:

- Run Status Manager
- T4R router
- Autenthication

5. Implementation, Integration and Test plan

6. Effort Spent

6.1 ARGIRO' ANNA SOFIA

DATE	DESCRIPTION OF THE TASK	HOURS SPENT
27/11/18	group work	3
2/12/18	high level overview	4
2/12/18	group work	4
8/12/18	Architecture revision, Introduction	4
8/12/18	group work	4

6.2 BATTAGLIA GABRIELE

DATE	DESCRIPTION OF THE TASK	HOURS SPENT
27/11/18	group work	3
30/11/18	component view	4
2/12/18	model diagrams	4
2/12/18	group work	4
6/12/18	Components interfaces	8
8/12/18	Deployement, sequence diagram	4
8/12/18	group work	4

6.3 CASASOLE BERNARDO

DATE	DESCRIPTION OF THE TASK	HOURS SPENT
27/11/18	group work	3
2/12/18	User interface design	4
2/12/18	group work	4
5/12/18	Implementation and testing	5
8/12/18	Implementation and testing	4
8/12/18	group work	4

7. References

7.1 Reference Documents

7.2 Software

- TeXWorks v0.6.2
- Draw.io v9.4.1
- proto.io v6.3.2.3