Software Engineering 2

TrackMe

Implementation and Testing Document

POLITECNICO
MILANO 1863

Romeo Paolo
Scotti Andrea
Staccone Francesco

January 11, 2019

Contents

1 Introduction

1.1 Purpose and Scope

The following document represents the Implementation and Testing Document for the TrackMe project. The purpose of this document is to provide a comprehensive overview of the implementation and testing activity for the development of the software application. The TrackMe application aims at be a Health Data Management and Run-Friendly Mobile App, useful for a quite wide range of users with different features and necessities.

1.2 Definitions, Acronyms, Abbreviations

1.2.1 Definitions

- **Framework**: is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code.
- **Third party**: a company, association or, more in general, a public or private entity that uses TrackMe to acquire data of users or to organize running events;
- **User**:a person who uses TrackMe;
- **Individual**:equivalent to User;
- **JWT**: (or JSON Web Token) is a JSON-based open standard (RFC 7519) for creating access tokens.

1.2.2 Acronyms

DBMS	Data Base Management System
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
API	Application Program Interface
REST	REpresentational State Transfer
MVC	Model View Controller
JSON	JavaScript Object Notation
VAT	Value Added Tax

1.3 Reference Documents

- Design document

- RASD document
- Project assignment

1.4 Overview

The rest of the document is organized in this way:

- **Implemented requirements:** explains which functional requirements outlined in the RASD are accomplished, and how they are performed.
- **Adopted frameworks:** provides reasons about the implementation decisions taken in order to develop the application.
- **Source code structure:** explains and motivates how the source code is structured both in the front end and in the back end.
- **Testing:** provides the main testing cases applied to the the application

2 Implemented requirements

In this section we describe the implemented functionalities with reference to the requirements outlined in the RASD document.
The requirements with their reference number green are the ones fully available in the current implementation. Requirements in red are not fully implemented but the base for a further complete development is present.

2.1 Third Party - Registration

- **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, name and password.

Database

The database stores the Third party information and credentials inside the table denominated "THIRD_PARTY". It also saves the VAT number and password in another table called 'USERS' with an attribute that specifies the role of the user (in this case the role is Third party) to determine which requests can be accepted and performed in future.

Back-end

The Third party registration API handles the registration process for the Third party and the communication with the database. It performs a check on the presence of an already registered Third party with the same VAT number.

Front-end

The homepage of the client application for a Third party contains a registration section showing a form to fill with registration information. The client application performs simple checks on the lenght and the type of data, sends the request to the server and, if the response is 'success', shows the login form for the third party.

2.2 Individual - Registration

- **R15)** Two different users cannot have the same username.
- **R13)** The system must allow the individual to register to the application by selecting a password and providing his/her data, fiscal code included;

Database

The database stores the Individual information and credentials inside the table denominated 'INDIVIDUAL'. It also saves the VAT number and password in another table called 'USERS' with an attribute that specifies the role of the user (in this case the role is Third party) to determine which requests can be accepted and performed in future.

Back-end

The Individual registration API handles the registration process for the individual and the communication with the database. It performs a check on the presence of an already registered Individual with the same fiscal code.

Front-end

The homepage of the client application for an individual contains a registration section showing a form to fill with registration information. The client application performs simple checks on the lenght and the type of data, sends the request to the server and, if the response is 'success', shows the login form for the individual.

2.3 Login

- **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.
- **R14)** The system must allow the individual to log in to the application by providing the combination of a fiscal code and a password that matches an account.

Back-end

The Login API manages the login of both Third parties and Individuals by validating the provided credentials and generating a unique access token for the user. The access token is generated by taking into account the username and password of the user and the current time. Every further request to the server must contain this token to be accepted.

Front-end

The front end is the same for Third party and Individual. The application in both cases shows the login form if the user is not logged in. The client application performs a simple check on the lenght of the data and sends the request to the server. If the response is 'success', the client receives the access token and shows the home page for the Third party or the Individual, otherwise shows an error.

2.4 Third Party - Settings Management

- **R33)** The system must allow the Third party to change its password.

Database

The database updates the data of the Individual by performing an update of the existing tuple.

Back-end

The Modify Third party password API first check if the old password value is really the password of the individual and then communicates to the database the new one.

Front-end

The client shows a form in which the user can insert the old and the new password. It performs a check on the minimum lenght of the passwords and forwards the data to the back-end, then it shows the response message.

2.5 Individual - Settings Management

- **R31)** The system must allow the user to change his/her personal info.
- **R32)** The system must allow the individual to change his/her password.
- **R34)** The system must allow the user to connect a smartwatch or a similar device to its smartphone.

Database

The database updates the data of the Individual by performing an update of the existing tuple. At this point of the implementation only password and location's coordinates can be updated.

Back-end

The Edit Individual Information API

Front-end

The client shows a form in which the user can insert the old and the new password. It performs a check on the minimum lenght of the passwords and forwards the data to the back-end, then it shows the response message.

2.6 Individual - AutomatedSOS service

- **R25)** The AutomatedSOS service must be enabled.
- **R18)** When the health status values go below the threshold, the system must send an SOS within 5 seconds;
- **R26)** The system must allow the user to enable/disable the AutomatedSOS service at any time.

Front-end

The client gives the user the chance to enable and disable the service by simply pressing a button. If the service is enabled, the application performs a check, every second, on the last N (N equal to 3 in the prototype) received data in order to detect the necessity of help. If all the N data are outside the safety bounds a message is shown and the service is disabled until the Individual reactivates it. The application must have a connected device sending data to provide the service.

2.7 Individual - Data acquisition

- **R27)** The system must be able to store data retrieved from registered users.
- **R3)** The system must be able to retrieve data from the smartwatches and similar devices;

Database

The database stores the data received by the Individuals in the 'INDIVIDUAL_DATA' table.

Back-end

The Send data API allows the client to send data to the server and transfer them to the database if they are sent by a registered Individual.

Front-end

The client should retreive data from external devices. It is supposed that data are received every second when a device is connected. The client accumulates data in a list and every time the list reaches size N (N equal to 12 in the prototype) the group of data is sent to the server. In this first implementation data from external devices are randomly generated when a fictitious device is connected.

2.8 Data Management

- **R11)** The system is optimized to send the data received from the mobile application to the third parties as soon as possible.
- **R9)** The third party is not allowed to access the users data until he/she accepts the request.
- **R2)** The system must be able to provide to the third party the location and the health status of individuals;
- **R1)** The users must have given the consensus to the treatment of their information to the third party;

Database

The database keeps all the data in the table 'INDIVIDUAL_DATA'

Back-end

Front-end

2.9 AutomatedSOS service

- **R25)** The AutomatedSOS service must be enabled.
- **R18)** When the health status values go below the threshold, the system must send an SOS within 5 seconds;
- **R26)** The system must allow the user to enable/disable the AutomatedSOS service at any time.

Front-end

2.10 Individual Requests

- **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.
- **R7)** The system must be able to forward the requests from the third party to the user;
- **R8)** The system must save the preference of the user;

Database

The database stores the requests from Third parties to Individuals in the table 'INDIVIDUAL_REQUEST'.

Back-end

Front-end

The Third party client shows a list of the Individuals registered to TrackMe and allows the Third party to make a

2.11 Anonymous Requests

- **R5)** The system must be able to provide to the third party the health status of individuals in an anonymous way;
- **R4)** The groups must be composed at least by 1000 individuals;
- **R6)** The system must be able to aggregate the data of the individuals, as requested by the third party;

Database

Back-end

Front-end

3 Adopted Frameworks

3.1 Spring Boot

Spring is a framework which provides comprehensive infrastructure support for developing Java applications and Spring Boot is basically an extension of Spring which eliminated the boilerplate configurations required for setting up a Spring application. In other words, while the Spring framework focuses on providing flexibility to you, Spring Boot aims to shorten the code length and provide you with the easiest way to develop a web application. With annotation configuration and default codes, Spring Boot shortens the time involved in developing an application. It helps create a stand-alone application with less or almost zero-configuration.

Why Spring Boot?

- To ease the Java-based applications Development, Unit Test and Integration Test Process.
- To reduce Development, Unit Test and Integration Test time by providing some defaults.
- To increase Productivity.

Why Spring?

- Spring is a light weight framework and It minimally invasive development with POJO.
- Spring achieves the loose coupling through dependency injection and interface based programming.
- Spring supports declarative programming through aspects and common conventions.

Inversion of Control

Inversion of Control is a principle in software engineering by which the control of objects or portions of a program is transferred to a container or framework. It's most often used in the context of object-oriented programming.

By contrast with traditional programming, in which our custom code makes calls to a library, IoC enables a framework to take control of the flow of a program and make calls to our custom code. To enable this, frameworks use abstractions with additional behavior built in. If we want to add our own behavior, we need to extend the classes of the framework or plugin our own classes.

The advantages of this architecture are:

- decoupling the execution of a task from its implementation
- making it easier to switch between different implementations
- greater modularity of a program
- greater ease in testing a program by isolating a component or mocking its dependencies and allowing components to communicate through contracts

Inversion of Control can be achieved through various mechanisms such as: Strategy design pattern, Service Locator pattern, Factory pattern, and Dependency Injection (DI).

3.2 AngularJS

Designed by Google, AngularJS is an open-source framework that addresses the challenges of web development processes, it is a framework for internet applications, it is one of the most powerful front-end frameworks. Angular framework allows the use of HTML as the template language and allows the extension of HTML's syntax to express application's components in a brief and clear manner. It can automatically synchronize with models and views. Angular could literally be described as a hybrid HTML editable version that creates Apps.

Why AngularJS?

- Promotes Code Reusability: with Angular, developers can reuse the codes that were previously used in other different application, thus promoting code reusability and making Angular a unique and outstanding framework.
- Faster application development: Angular makes everything from development to testing and maintenance quite fast and quick. The MVC (model view controller) architecture assures this.
- Gives Developers full Control: developers can test, construct, inject and do almost anything with this framework, this is due to the fact that directives offer a free hand to experiment with HTML and attributes.
- Data Binding: is easy with Angular, data binding happens with ease.
- Time-Saving: all that Angular requires you to do is split your web app into multiple MVC(model view controller) components. Once that is done, Angular automatically takes over and performs the rest of the functions. It saves you from the trouble of writing another code.

3.3 Ionic

Ionic is a complete open-source SDK for hybrid mobile app originally built on top of AngularJS and Apache Cordova. Ionic provides tools and services for developing hybrid mobile apps using Web technologies like CSS, HTML5, and Sass. Apps can be built with these Web technologies and then distributed through native app stores to be installed on devices by leveraging Cordova. Ionic Framework is designed to work and display beautifully out-of-the-box across all platforms. It's a library of UI Components, which are reusable elements that serve as the building blocks for an application. Ionic Components are built with web standards using HTML, CSS, and JavaScript. Though the components are pre-built, they're designed from the ground up to be highly customizable so apps can make each component their own, allowing each app to have its own look and feel. More specifically, Ionic components can be easily themed to globally change appearance across an entire app.

Why Ionic?

- Ionic is completely free and open source.
- Ionic is built on Angular.
- Ionic is a really "Native Like" framework.
- Ionic has a beautiful default UI that is easy to customise.
- It has a lot of tools and services.
- Ionic integrates easily with native functionalities.

3.4 Drawbacks

Since the drawing up of the Design Document, we have been asking ourself if a hybrid client (with no local storage) for the third party could be a good choice or not. Now we can say that this choice requires few effort from the client point of view but it really reduces the performances of the server because it doesn't allow the server to exploit the caching capability of the client. For simplicity, since we are developing a prototype, we have decided to continue with this choice but we have structured the third party front end code, in such a way to easily move this web browser app in a native desktop app, for Windows, MacOs, Linux. Of course this operation will require a change in the Design Document and in this document. By choosing AngularJs as a front-end framework it will be much easier to do this transformation. Also the back-end will require just few changes, in particular the

Storage Controller needs to retrieve data with timestamp greater than the timestamp of the last cached data on the client, and the ThirdParty Controller needs to be able to retrieve the timestamp of the last cached data on the client with REST API, call the proper method on the Storage Controller and then send the new data to the client.

Electron

Electron is a framework for creating desktop applications with all the emerging technologies including JavaScript, HTML and CSS. Basically, Electron framework takes care of the hard parts so that you can focus on the core of the application and revolutionize its design. Designed as an open-source framework, Electron combines the best web technologies and is a cross-platform – meaning that it is easily compatible with Mac, Windows and Linux.

Why Electron?

- HTML, CSS, JS: Of course this is the most important point. It is amazing that you can now build Desktop Apps using these languages as it is very easy to learn and use them.
- Electron Apps Are Similar To Web Apps: Part of what makes Electron Apps a good alternative to a native desktop app is the fact that Electron apps behave like Web Apps. What sets them apart is that Web Apps can only download files to the computer's file system but Electron Apps can access the file system and can also read and write data.
- Chromium: Electron uses Chromium engine for rendering UI. This means that you can get several benefits from this like Developer Tools, Storage Access, etc.

4 Source Code Structure

The source code follows the structure of a Spring project. ALl the code is located in the **src** folder and the distinction between the code of the application and the code of the tests performed on it is clear. The **main** folder contains the code for the back-end and the front-end of the system while the tests are in the **test** folder. All the back-end code is written in Java while the front-end code is written using Javascript, HTML and CSS.
The Spring configuration file is in **resources**. It manages The following of this section describes more in detail the structure of back-end and front-end code.

4.1 Back End

The back-end code is all inside the **java** folder and it is divided into the following packages:

- **controller**;
- **model**;
- **storageController**.

Controller package

This package contains the following controllers:

- **authenticationController**;
- **individualController**;
- **thirdPartyController**.

AuthenticationController

The authenticationController package is composed by the entities that manage the requests of login and sign up for both Individuals and Third parties. For this purpose, a pre-exisisting implementation of these functionalities using JWT has been imported and adapted to the needs of this application.

IndividualController
The individualController package contains the spring controller IndividualController and the spring service IndividualService. The controller defines all the APIs to manage the functionalities provided to the Individuals and uses the service to communicates with the database reflecting the effects of the APIs' calls.

ThirdPartyController
The thirdPartyController package contains the spring controller ThirdPartyController and the spring services ThirdPartyService and AnonymousRequestBuilder. The controller defines the APIs to manage the functionalities provided to the Third parties and uses the two services to communicates with the database reflecting the effects of the APIs' calls.

Model package

This package is composed by the classes that realize the Object-relational mapping

StorageController package

This package contains the interfaces that offer methods to perform queries on the database. All these interfaces extends the CrudRepository interface. The CrudRepository is an interface that extends Spring data Repository interface. CrudRepository provides generic CRUD operation on a repository for a specific type. It has generic methods for CRUD operation.

4.2 Front End

The code for the front-end is located in **resources** in the **static** folder. Here, a distinction between the code for Individuals and Third parties is made. The **individual** folder contains the CSS and Javascript files and HTML pages that are responsible for the Individuals mobile application. CSS and html are optimized for the mobile version of the application, visualizing the same pages in large screens could present layout problems. The **thirdParty** folder is the corresponding one for the Third parties web application.

5 Testing

6 REST API

6.1 Authentication Controller

Third Party SignUp

endpoint	*/auth/thirdparty/signUp
method	POST
url params	
data params	VAT: [alphanumeric] name: [alphanumeric] password : [alphanumeric]
success response	code: 200
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 500 INTERNAL SERVER ERROR Content : {error: "Could not commit JPA transaction"} code: 409 CONFLICT Content : {error: "This user already exists"}
Notes	Allows a third party to register to the system.
Request Example	POST /api/v1/user HTTP/1.1 Host: addr:8080 User-Agent: * Content-Type: application/json Accept: application/json {"vat": "thirdParty", "password": "thirdParty", "name": "thirdParty" }

Individual SignUp

endpoint	*/auth/individual/signUp
method	POST
url params	
data params	username: [alphanumeric] password : [alphanumeric] name: [text] surname: [text] latitude: [numeric] longitude: [numeric] birthDate: [date]
success response	code: 200
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 500 INTERNAL SERVER ERROR Content : {error: "Could not commit JPA transaction"} code: 409 CONFLICT Content : {error: "This user already exists"}
Notes	Allows an individual to register to the system.

Request Example	POST /api/v1/user HTTP/1.1 Host: addr:8080 User-Agent: * Content-Type: application/json Accept: application/json {"fiscalCode": "individualindivi", "password": "individual", "name": "individual" "surname": "individual", "birthDate": "2019-01-21T23:00:00.000Z", "latitude": "5.0", "longitude": "4.5" }

Login

endpoint	*/auth
method	POST
url params	
data params	username: [alphanumeric] password : [alphanumeric]
success response	code: 200 Content : {token: [alphanumeric]}
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} Code: 401 UNAUTHORIZED Content : {error: "Bad Credentials!"}
Notes	Allows an individual or a third party to obtain an authentication Token

NOTE:

- all successive requests need to contain the token, retrieved during the login request, in order to be allowed;
- if a logged in user try to use REST API trying to be another user a excepction is launched (code: 401 UNAUTHORIZED Content : {error: "Trying to be another user!"}.

6.2 Third Party Controller

Make and individual request

endpoint	*thirdparty/individualRequest
method	POST
url params	
data params	vat: [alphanumeric] fiscalcode: [alphanumeric] subscribedToNewData: [boolean]
success response	code: 200
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 409 CONFLICT Content : {error: "This user already exists"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Third Party Not Found"} code: 404 NOT FOUND Content : {error: "Individual Not Found"} Code: 401 CONFLICT Content : {error: ""This request has been already done""}
Notes	Allows the third party to do an individual request of data.

Make and anonymous request

endpoint	*thirdparty/anonymousRequest
method	POST
url params	
data params	vat: [alphanumeric] startAge: [numeric] endAge: [numeric] lat1: [float] lon1: [float] lat2: [float] lon2: [float] subscribedToNewData: [boolean]
success response	code: 200
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 500 INTERNAL SERVER ERROR Content : {error: "Could not commit JPA transaction"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Third Party Not Found"}
Notes	Allows the third party to do a group request of data.

Modify Third Party Password

endpoint	*/thirdParty/username/changePassword
method	PUT
url params	username: [alphanumeric]

data params	newPassword: [alphanumeric] oldPassword:[alphanumeric]
success response	Code: 200
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Third Party Not Found"} code: 422 UNPROCESSABLE ENTITY Content : {error: "Bad Credentials"} code: 422 UNPROCESSABLE ENTITY Content : {error: "Data are not well formed"}
Notes	Allows a third party to change its password.

Get individual requests

endpoint	*/thirdParty/{thirdParty}/individualRequests
method	GET
data params	
url params	vat: [alphanumeric]
success response	code: 200 Content : {individualRequests: List¡IndividualRequest¿}
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Third Party Not Found"}
Notes	Allows the third parties to request for all individual requests it has done.

Get individual request notifications

endpoint	*/thirdParty/{thirdParty}/notifications/individualRequests
method	GET
data params	
url params	vat: [alphanumeric]
success response	code: 200 Content : {notifications: List¡IndividualRequest¿}

error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Third Party Not Found"}
Notes	Allows the third parties to request for notifications of individual requests.

Get individual request notifications counter

endpoint	*/thirdParty/{thirdParty}/notifications/countIndividualRequests
method	GET
data params	
url params	vat: [alphanumeric]
success response	code: 200 Content : {counter: [integer]}
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Third Party Not Found"}
Notes	Allows the third parties to request for the number of new notifications of individual requests.

Get past individual data

endpoint	*/thirdParty/{thirdParty}/{individual}/data
method	GET

data params	
url params	vat: [alphanumeric] fiscalCode: [alphanumeric]
success response	code: 200 Content : {individual data: List<IndividualData>}
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Third Party Not Found"} code: 404 NOT FOUND Content : {error: "Individual Not Found"} code: 404 NOT FOUND Content : {error: "The thirdParty has not the right to receive data from the individual because you never asked for it"} code: 400 BAD REQUEST Content : {error: "You can't acces this data"}
Notes	Allows the third parties to request for past data of a specific individual.

Get new individual data

endpoint	*/thirdParty/{thirdParty}/notifications/{individual}
method	GET
data params	
url params	vat: [alphanumeric] fiscalCode: [alphanumeric]

success response	code: 200 Content : {notifications: List<IndividualData>}
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Third Party Not Found"} code: 404 NOT FOUND Content : {error: "Individual Not Found"} code: 404 NOT FOUND Content : {error: "The thirdParty has not the right to receive data from the individual because you never asked for it"} code: 400 BAD REQUEST Content : {error: "You can't acces this data"}
Notes	Allows the third parties to request for new data of a specific individual.

Get anonymous requests

endpoint	*/thirdParty/{thirdParty}/anonymousRequests
method	GET
data params	
url params	vat: [alphanumeric]
success response	code: 200 Content : {anonymousRequests: List<AnonymousRequest>}

error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Third Party Not Found"}
Notes	Allows the third parties to request for all anonymous requests it has done.

Get past anonymous request data

endpoint	*/thirdParty/{thirdParty}/anonymousAnswer/{anonymousAnswer}
method	GET
data params	
url params	vat: [alphanumeric] anonymousRequestId: [alphanumeric]
success response	code: 200 Content : {anonymous answers: List<AnonymousAnswer>}
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Third Party Not Found"} code: 404 NOT FOUND Content : {error: "Anonymous Request Not Found"} code: 400 BAD REQUEST Content : {error: "Not your request"}

Notes	Allows the third parties to request for past data of an anonymous request.

Get new anonymous request data

endpoint	*/thirdParty/{thirdParty}/anonymousAnswer/notifications/{anonymousAnsw
method	GET
data params	
url params	vat: [alphanumeric] anonymousRequestId: [alphanumeric]
success response	code: 200 Content : {anonymous answers: List<AnonymousAnswer>}
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Third Party Not Found"} code: 404 NOT FOUND Content : {error: "Anonymous Request Not Found"} code: 400 BAD REQUEST Content : {error: "Not your request"}
Notes	Allows the third parties to request for new data of an anonymous request.

6.3 Individual Controller

Answer to Request

endpoint	*individual/individualRequest/answer
method	POST
url params	
data params	vat: [alphanumeric] fiscalCode: [alphanumeric] accepted: [boolean]
success response	code: 200
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Individual Request Not Found"} code: 404 NOT FOUND Content : {error: "Individual Found"}
Notes	Allows the individual to accept or refuse an individual request.

Modify Individual Password

endpoint	*/individual/username/changePassword
method	PUT
url params	username: [alphanumeric]
data params	newPassword: [alphanumeric] oldPassword:[alphanumeric]
success response	Code: 200

error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Individual Not Found"} code: 422 UNPROCESSABLE ENTITY Content : {error: "Bad Credentials"} code: 422 UNPROCESSABLE ENTITY Content : {error: "Data are not well formed"}
Notes	Allows an individual to change its password.

Modify Individual Residence

endpoint	*/individual/username/changeLocation
method	PUT
url params	username: [alphanumeric]
data params	newLatitude: [float] newLongitude:[float]
success response	Code: 200
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 422 UNPROCESSABLE ENTITY Content : {error: "Provided values are not valid"}
Notes	Allows an individual to change its password.

Get individual pending requests

endpoint	*/individual/{individual}/individualRequests
method	GET
data params	
url params	fiscalCode: [alphanumeric]
success response	code: 200 Content : {individualRequests: List¡IndividualRequest¿}
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Individual Not Found"}
Notes	Allows the individual to request for all individual requests pending for him.

Get individual accepted requests

endpoint	*/individual/{individual}/acceptedRequests
method	GET
data params	
url params	fiscalCode: [alphanumeric]
success response	code: 200 Content : {individualRequests: List<IndividualRequest>}

error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Individual Not Found"}
Notes	Allows the individual to request for all individual requests that he has already accepted.

Get individual request notifications

endpoint	*/individual/{individual}/notifications
method	GET
data params	
url params	fiscalCode: [alphanumeric]
success response	code: 200 Content : {notifications: List<IndividualRequest>}
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Individual Not Found"}
Notes	Allows the individual to request for all new individual requests for him (that he hasn't seen yet).

Get individual request notifications counter

endpoint	*/individual/{individual}/countNotifications
method	GET

data params	
url params	fiscalCode: [alphanumeric]
success response	code: 200 Content : {counter: [integer]}
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 404 NOT FOUND Content : {error: "Third Party Not Found"}
Notes	Allows an individual to request for the number of new notifications of individual requests.

Send Data

endpoint	*individual/{id}/data
method	POST
url params	
data params	accessToken: [alphanumeric] data: [] (implementation detail)
success response	code: 200 Content : {message: "Data received correctly."}
error response	code: 400 BAD REQUEST Content : {error: "JSON parse error"} code: 401 UNAUTHORIZED Content : {error: "Bad credentials!"} code: 422 UNPROCESSABLE ENTITY Content : {error: "There is no data."}

Notes	Allows the individual to send data.

7 Installation Instructions

8 Source Code Structure

8.1 Romeo Paolo

8.2 Scotti Andrea

8.3 Staccone Francesco