Usafe

Julie Bulanda

C16726719

Submitted in partial fulfilment of the requirements for the degree of

B.Sc. in Business Computing

Dublin Institute of Technology

Year 4

Supervised by – Jenny Munnelly

May 2020

Acknowledgments

I would like to take this opportunity to say a sincere thank you to everyone who helped me accomplishing my final year project.

First, I would like to sincerely thank my project supervisor Jenny Munnelly for her advice and guidance that helped me successfully completing my project. Her ideas helped me bring this project beyond expectations.

Secondly, I would like to thank my second reader Don Ryan for advising me as well during the checkpoints. Giving me additional relevant ideas that brought value to my project.

I would also like to thank my friend Laura Pacaud for helping me designing the logo for my application.

I would also like to acknowledge my friends and family for their encouragement.

Finally, I would like to say a big thank you to my boyfriend for the incredible support he gave me over the last four years of this course.

Abstract

Safety and security are 2 major concerns for individuals. Yet, today, there are little products that address this issue. And even fewer of them that manages to tackle this problematic in a simple way, that would contribute in democratizing those products. This project aims to find an easy and reliable solution to find a safe way from where you are to where you want to go.

The main idea of this project, is to create a simple but efficient application that provides a score established from user's preferences and aggregated geospatial data. This, in order to provide users a comprehensive way to evaluate which route is best for him/her to follow.

The project also contains other features that helps ensuring users are safe while travelling to their destination.

The project relies on the use of a mobile device. Even though it only starts with Android devices, it has been built in order to be able to add new types of device easily.

This report establishes the reasons why such project is relevant in today's world. But also, details its technical design, implementation and researches that lead to its final completion.

Table of Contents

1 Requirements Specification 6
1.1 Project Introduction 6
1.2 Purpose of the project 6
1.3 Requirements 7
1.3.1 Functional Requirements 7
1.3.2 Non-Functional Requirements 8
2 Analysis 9
2.1 User Analysis 9
2.2 Data Analysis (ERDs, Data Dictionary) 10
2.2.1 Geospatial data 10
2.2.2 Types of Geospatial Data 10
2.2.3 Spatial Reference System 12
3 Design 14
3.1 Database Design 14
3.2 Elasticsearch Entities 14
3.3 Business Rules 15
3.4 System Architecture 15
3.4.1 Architecture Details 15
3.4.2 Benefits of the architecture 17
3.4.3 Drawbacks of the architecture 17
3.5 UI/UX 18
3.6 Technologies 18
4 Implementation 24
4.1 Journey lifecycle 24
4.1.1 Start a journey 24
4.1.2 Performing the Journey 25
4.1.3 Stopping a journey 26
4.2 Calculating the safety score 26
4.3 Sending a notification 27
5 Test Plan 28
6 Installation Manual 31
6.1 Install the API 31
6.1.1 Pre-requisites 31

6.1.2 Create a jar of the API 31
6.1.3 Start the API Stack 31
6.2 Install the Android Client 31
6.2.1 Pre-requisites 31
6.2.2 Running the Android client 32
7 User Manual 32
7.1.1 Create an account 32
7.1.2 Login 33
7.1.3 Define user preferences 33
7.1.4 Add a contact 34
7.1.5 Start a journey 34
7.1.6 Select your route 35
7.1.7 Journey features 36
7.1.8 Followers features 37
8 Conclusion 38
9 References 39
10 Lexical 40
10.1 ETL 40

1 Requirements Specification

1.1 Project Introduction

Safety is among the basic needs defined in Maslow's Hierarchy of needs. As a matter of fact, it is the second level of the pyramid, right after the physiological needs. In today's world, safety is also an index that people use to decide whether they should consider a city to move, but also, whether an investment is worth in a city.

It is therefore unfortunate to find surveys showing that 99 percent of women experienced street harassment. Furthermore, a survey from Stop Harassment made in January 2018 have found that 81 percent of women and 43 percent of men experienced some form sexual harassment in their lifetime.

Not only those numbers can shock, they contribute to an overall feeling of insecurity in public spaces, and therefore, lead to a decrease in the quality of life of individuals.

1.2 Purpose of the project

The purpose of the Urban Safe (USafe) project, to a larger extent, is to increase the quality of life of individuals by increasing their feeling of safety in public spaces.

This is done, by modelling the parameters that contribute to the feeling of safety with people. And incorporating those parameters in the elaboration of a score that helps users navigating in the city they are in, mostly at night.

With the help of Open Data as well as Big Data technologies, USafe is able to integrate and process a large volume of information about the location of public lights in a city, as well as the location of Garda stations. Users can then determine the importance of each parameter, which will then be used when elaborating the Safety Score of the paths that leads to his/her destination. And choose the most appropriate path according what matters the most to them.

Once the path chosen, USafe helps users feeling safer by allowing them to choose trusty people among its contact to follow his/her journey. Allowing those following contacts to interact with him/her whenever a problem may happen.

Usafe's smart features automatically detect problems that may occur during the user's commute. For example, the user deviating the planned route, the user running.

Furthermore, if the user feels in danger, USafe allows him/her to associate a vocal keyword to an emergency contact. When saying the keyword on the phone, the emergency contact will automatically be contacted without further interaction from the user.

Finally, in case the user notifies something suspicious during his commuting, USafe allows him to easily capture video snapshots in order to keep potentially important elements in his phone.

1.3 Requirements

1.3.1 Functional Requirements

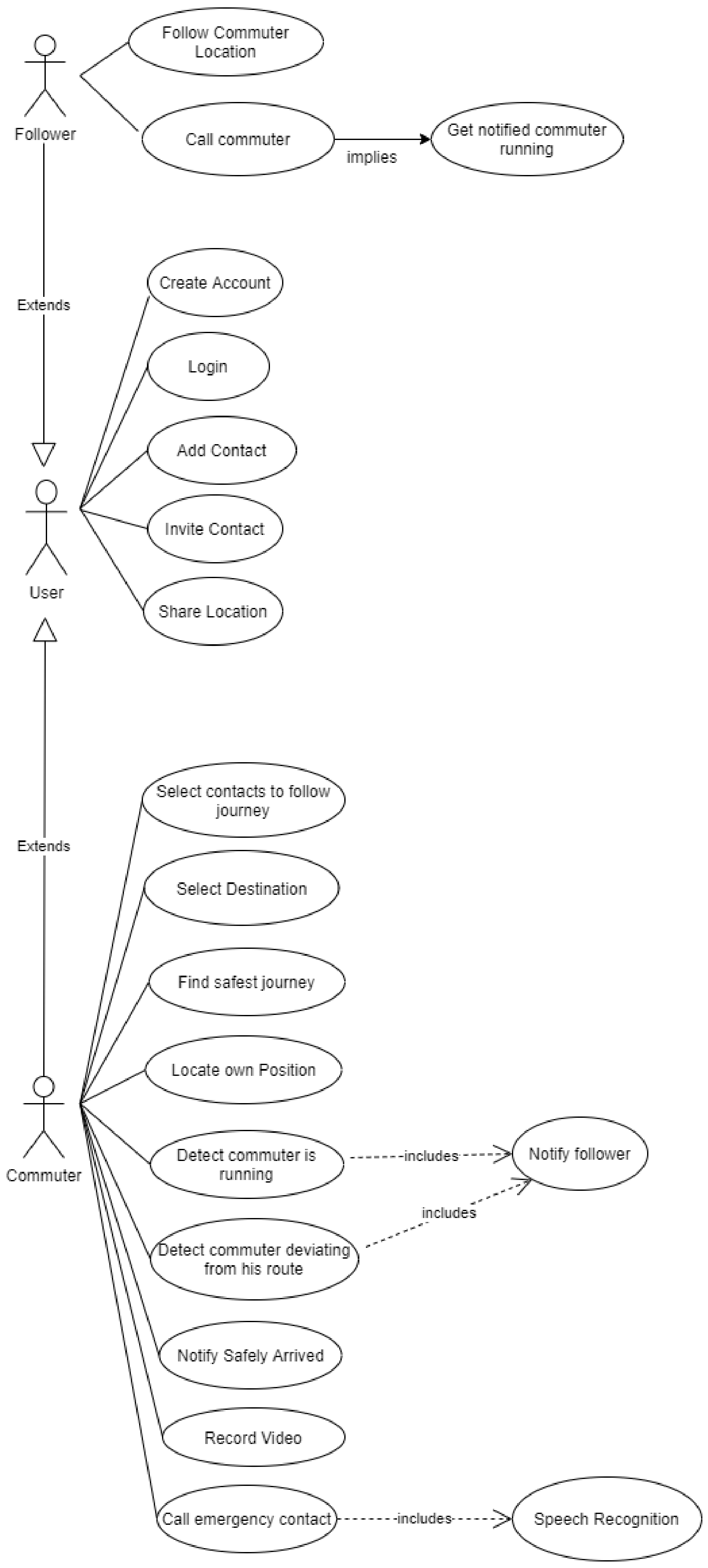
Function	Requirement
Registration	The user should be able to input his personal details in a form to create an account.
Login	The user should be able to reuse the password and email he used when registering in order to login to the application.
Contact Management	The user should be able to view the list of contacts from his phone.
	The user should be able to add a contact that is using the app.
Preferences Management	The user should be able to act (determine the importance of each criteria according to the others) on the parameters that determine the safety score calculated for the routes.
	The user should be able to select an emergency contact from his phone contacts.
Journey Management	The commuter should be able to start a journey by entering a destination and add some followers.
	The commuter should visualise the safety score for each route.
	The commuter should be able to select the route that seems the most appropriate.
	The commuter should be able to record snapshot videos from his phone when pressing the SOS button.
	Phone should call the emergency contact when the application detects/hear that the commuter is in danger.
Notification Management	The followers should receive a notification when the commuter is starting his journey.
	The followers should be able to follow the commuter’s journey from the received notification.
	The followers should be able to follow the commuter’s location in real time from the received notification.
	The followers should be notified when the commuter is running
	The commuter should receive a popup notification when running is detected.
	The followers should be notified when the commuter is deviating from his current route.
	The followers should be able to call the commuter from the notifications (running & deviation)
	The followers should be notified when the commuter is safely arrived.

1.3.2 Non-Functional Requirements

Function	Requirement
Responsiveness	If the application is on hold (ie. a call), then the app should be able to return to the same state/page than it was before.
Usability	User should be able to use the application without any guideline.
Performance	The user should not be blocked during data loading times. Meaning, even though the data is being loaded , he should be able to navigate on the map.
Reliability	The app should be able to work for an undefined period of time without experiencing any crash. Meaning, the user should be able to perform the same action multiple times without any problem.

2 Analysis

2.1 User Analysis



2.2 Data Analysis (ERDs, Data Dictionary)

2.2.1 Geospatial data

Geospatial Data and Geographic Information System (GIS) are at the heart of Usafe. Indeed, whether it is to calculate the safety score or display the User's path, the application uses location data and geometries. This section tells a bit more about how geospatial data can be used to solve problems and how does it work.

First, let us start with a bit of history. Back in 1854, a cholera outbreak hit the city of London in England. At the time, nobody knew how the outbreak started and how it spreads, air was suspected to be the main vector of transmission. But, British physician John Snow had the idea to start mapping the outbreak. He did this by mapping the location of infected people on a map of London. Keeping track of the roads, properties and water lines.

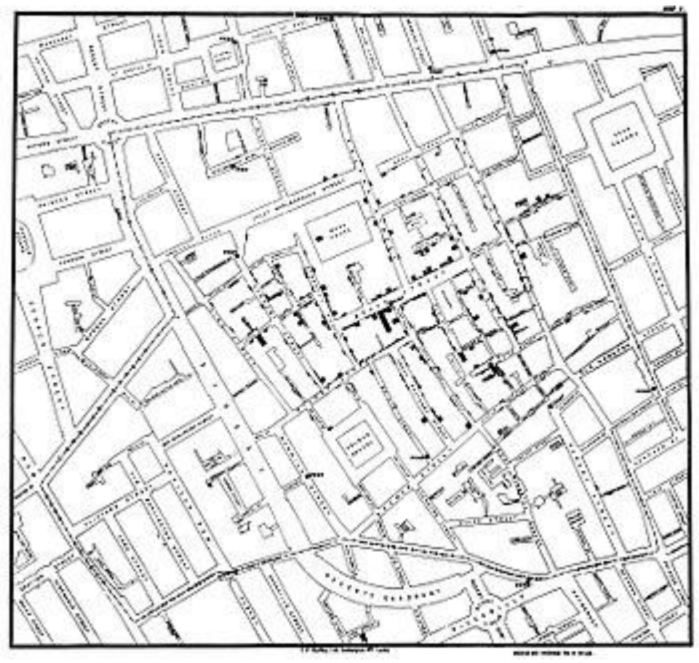


Figure 1: John Snow's map (source ....)

While doing this, John Snow noticed that cholera cases were only along water line. This major discovery allowed to understand that spread was made via water. But also, locate the source of the epidemic which was from a public well dug close to an old cesspit.

This, was only the start of using Geospatial Data and GIS to help solving problem on a broad scale. Today, geospatial analysis as well as geospatial data is anchored in our everyday lives. We will now talk about the different components and types of geospatial data.

2.2.2 Types of Geospatial Data

There are different types of geospatial data. Both have benefits and inconvenients that we will address in this section.

Raster

Raster store data in a grid cells format (rows and columns). They are usually used to represent data that varies continuously, such as elevation of mountain surfaces or temperature variations.

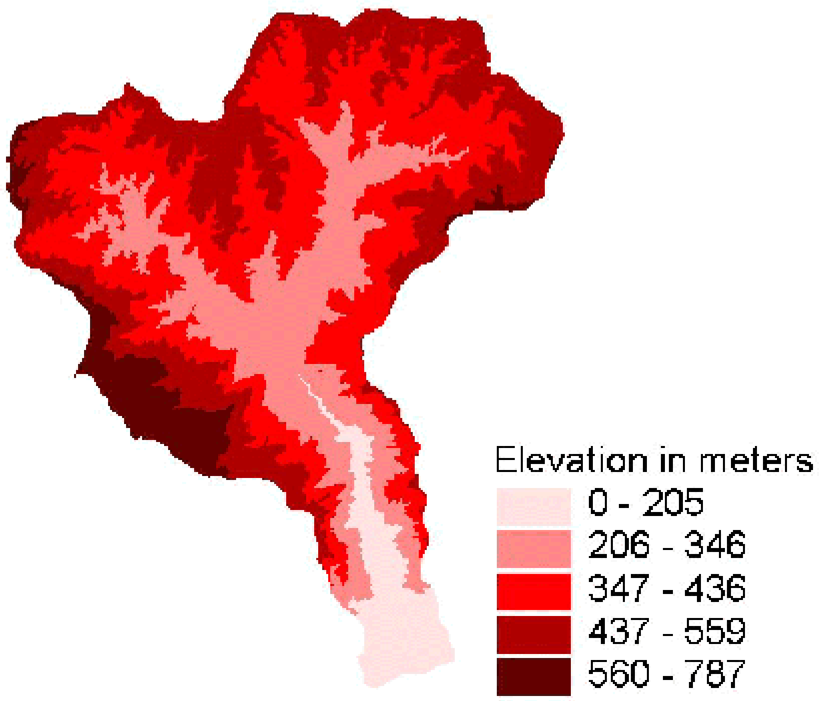


Figure 2 Example of Raster data

The benefits of Raster data is that it offers simple data structures that are cheap to display using a computer. On the other hand, the disadvantage of this type of data is that due to their grid nature, Raster data often looks pixelated, also less precise than Vector data.

Vector

Unlike Raster data, Vector data gives higher geographic accuracy as they are composed by paths and vertices. Meaning, the data does not depends on a grid size specifically.

They are composed by Points, Polygons, Lines and other geometries.

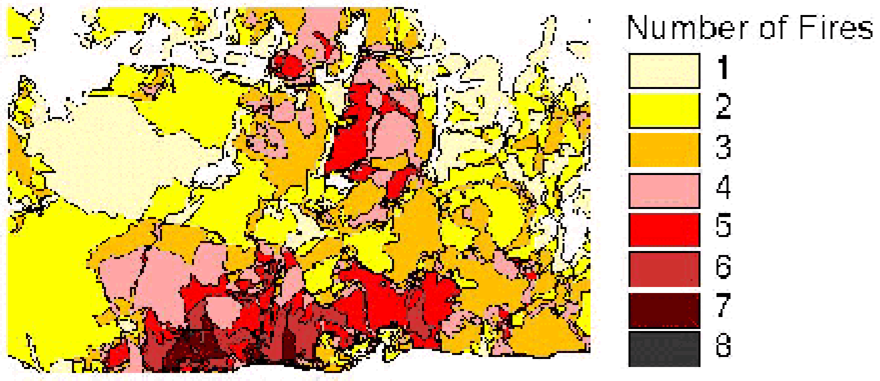


Figure 3 Example of Vector data

The main benefit of using Vector data is to get an accurate representation of it when displayed. On the other hand, Vector data is expensive to store when it represents a high volume of data. It is also more processing intensive than raster data and requires more complex manipulation algorithms.

The data used in Usafe is only Vector data. Indeed, the application was requiring small sets of data in order to display paths, check their interactions to get the score and display locations of points of interest such as Garda Stations.

However, Raster data could be used in the future in order to include more data in the score calculation. For example, the criminality depending on a specific area. Or, display layers about criminality or the level of lightening on specific areas.

2.2.3 Spatial Reference System

Geospatial Data only make sense to be displayed depending on a Spatial Reference System (SRS) also called Coordinate Reference System (CRS). Those, are coordinate-based local, regional or global system used to locate geographical entities. SRS define specific map projections, as well as transformation necessary to convert geospatial data between 2 different Spatial Reference Systems.

In order to identify them, SRS have a Spatial Reference System Identifier (SRID) that is a unique value identifying the SRS. For example, the World Geodetic System (WGS84) SRID is EPSG:4326 for coordinates. In comparison, the ING (Irish National Grid System) is EPSG:29902.

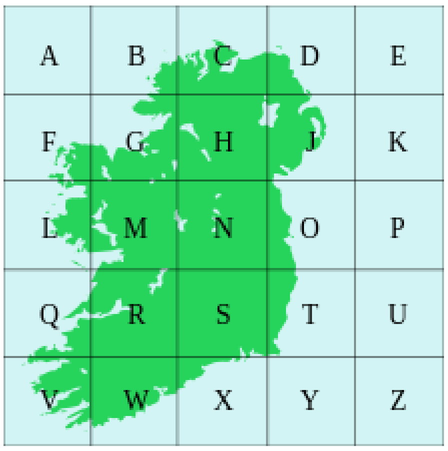


Figure 4 Irish National Grid SRS

In my case, while developing Usafe, the datasets (lights and garda stations location) I could find were using the ING format with easting / northing to represent the latitude and longitude. I had to convert them prior persisting them, in order to be more efficient when comparing them to data coming from Google Maps.

Garda Stations Dataset

Crimes at Garda Station contains nine reported crime categories for each of the 563 Garda Stations in the Republic of Ireland from 2003 to 2015.

From this dataset, I used the following values:

Attribute	Description	Example
Station	The name of the garda station.	Kevin Street
x	A numeric value that represents the easting coordinate of the garda station. This value correspond to the latitude of the garda station.	315268
y	A numeric value that represents the northing coordinate of the garda station. This value correspond to the longitude of the garda station.	233451

Lights Dataset

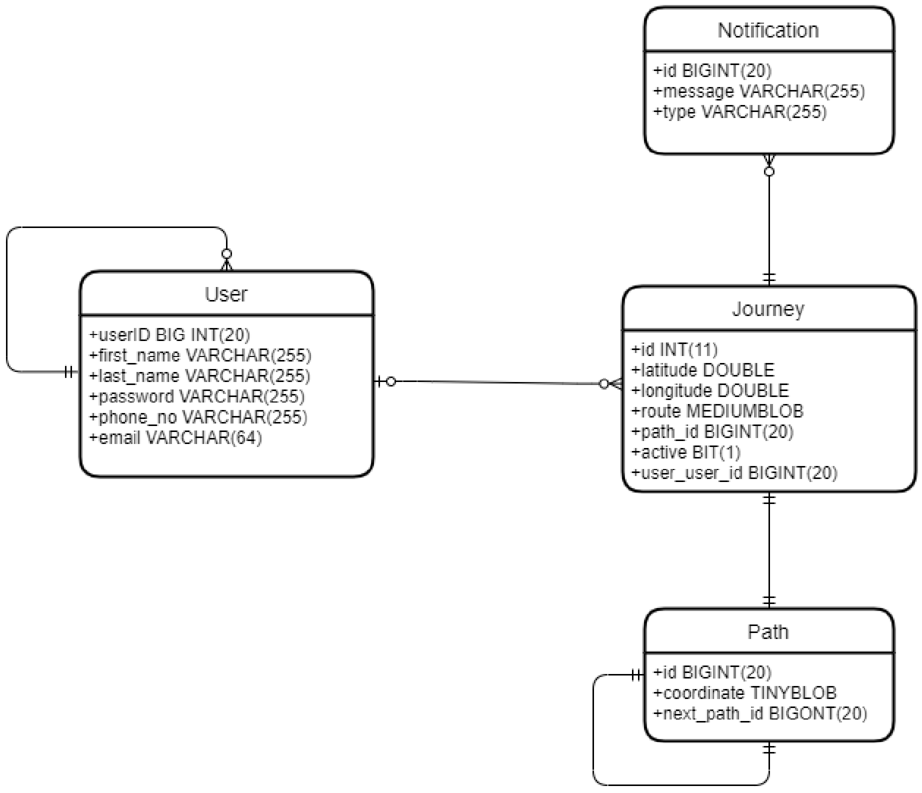
This dataset is extracted from Dublin City's council's street light management system. It consists of all street lighting pole locations in Dublin City.

From this dataset, I used the following values:

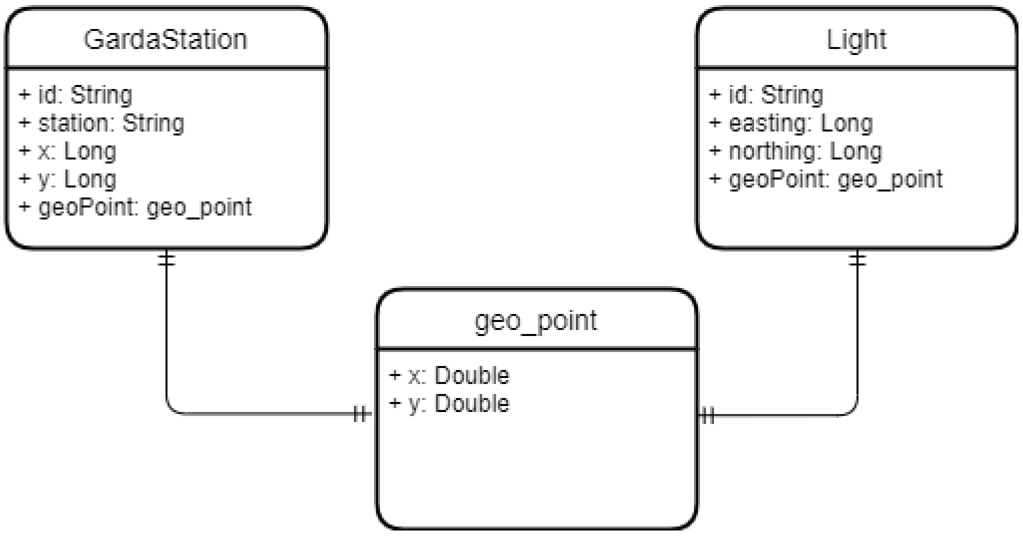
Attribute	Description	Example
EASTING	A numeric value that represents the latitude coordinate of the light pole in project EPSG:29902 (Irish National Grid)	315268
NORTHING	A numeric value that represents the longitude coordinate of the light pole in project EPSG:29902 (Irish National Grid)	233451

3 Design

3.1 Database Design



3.2 Elasticsearch Entities



The above diagram illustrates the different entities stored in Elasticsearch. In Elasticsearch, the schema of an entity is called a mapping. This mapping is a Json representation of the entity. The below Json snippet is the mapping of the garda stations:

```
  "gardastations" : {
    "mappings" : {
      "gardastation" : {
        "properties" : {
          "easting" : {
            "type" : "long"
          },
          "geoPoint" : {
            "type" : "geo_point"
          },
          "northing" : {
            "type" : "long"
          },
          "station" : {
            "type" : "text",
            "fields" : {
              "keyword" : {
                "type" : "keyword",
                "ignore_above" : 256
              }
            }
          }
        }
      }
    }
  }
}
```

The geoPoint field of type geo_point allows Elasticsearch to perform Geospatial queries on the entity.

3.3 Business Rules

- User must be logged into the system.
- Each user can only have one account associated with his/her email.
- Users must accept following permissions: location, contacts, camera, microphone, storage, telephone and physical activity.
- Location must be actively shared.
- Followers must have logged at least once to be able to receive notifications

3.4 System Architecture

3.4.1 Architecture Details

The overall architecture of the application can be split in 3 distinct parts:

- The server / internal API side
- The client’s side
- The external APIs side

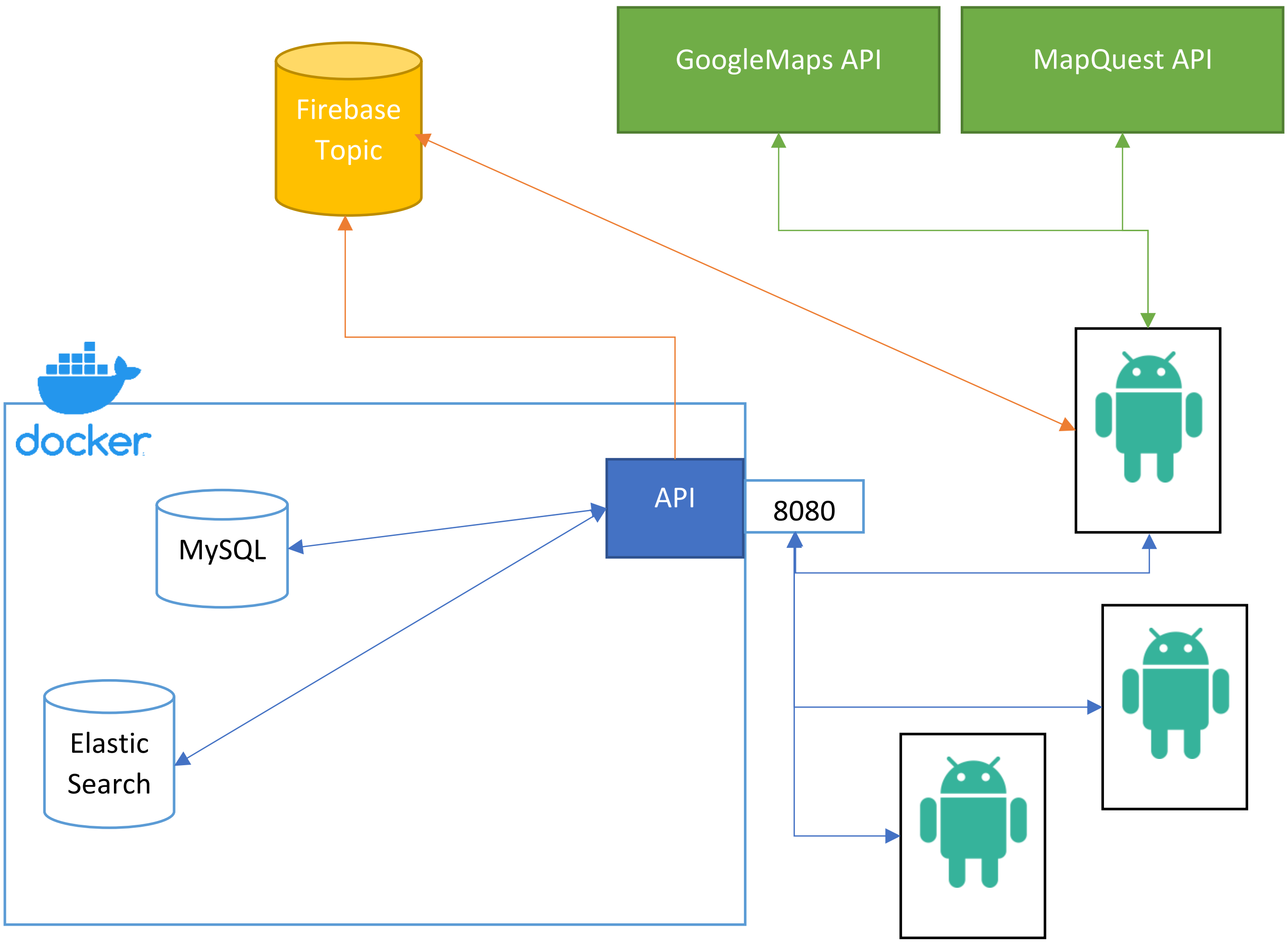
Internal API side

The internal API is the heart of USafe. It holds all the business logic (score calculation, data updates and data ingestion). This side is composed of an ElasticSearch Database that stores all the geospatial information about city lights and garda stations. In addition to the ElasticSearch database, a MySQL Database that stores the user’s information as well as the information about the ongoing journey of a user.

The decision to go with 2 distinct databases has been made on the base that ElasticSearch is a No-SQL database. It is hence very efficient for data processing and querying. But not to store real-time information that often changes.

The business logic is handled in a Java microservice that is called the API. The API deals with client's requests made through a REST API. It is responsible to deal with the data ingestion and ETL transformation. It also handles requests from clients through a REST API and is responsible for calculating the Safety Score, storing real-time information such as active user's journey, new contacts requests, new accounts being created. The benefit of having it as a microservice is that it can easily be replicated and run in parallel in order to handle an increase of client requests.

All the components run as Docker containers in a Docker environment. This has the benefit that the entire stack can easily run from one machine to another that is also running Docker. It also allows to expose ports (ElasticSearch, MySQL and the API itself) to the relevant components instead of everywhere. Hence, increasing security.



Client side

For now, USafe only supports Android clients. The client deals with displaying information to the user. This includes displaying the user's location on an interactive map, the address searches, storing the user's preferences and fetching data from the internal API.

Any REST call or display action are using callbacks in order to make the action non-blocking for the user. Making it a smoother experience when using the application.

Third party APIs

The use of third-party APIs allows USafe to enhance user experience of the application. There are 3 main third party APIs. The main one being GoogleMaps allows to show an interactive world map that contains relevant geospatial information. The second API, MapQuest is used to find the different paths from a start location to an end location. Finally, the Firebase Cloud Messaging API is used in order to deal with notification between user and its followers.

3.4.2 Benefits of the architecture

The main benefit of this architecture is that it removes coupling between the business logic and the display logic of the application. This way, it will be easier in the future to add new clients (iOS and Web for example) in order to attract new users into using USafe.

Another major benefit of this architecture is that all components can be scaled independently from each other. As a result, any bottleneck in the application can be easily detected and fixed. For example, if the java API was not able to handle any more requests, it would be easy to add an extra one without impacting the other components (Android clients, MySQL and ElasticSearch).

Finally, the use of Docker makes it easier bringing to life the entire stack of components. That, on any new machine that is running Docker.

3.4.3 Drawbacks of the architecture

Great benefits always come at a price. This is what I have learnt while developing USafe in such way. The main drawback is the system complexity. A lot of features would have been faster to develop in Android only. Also, the debugging of the features would have been easier as it would not have required to constantly jump between the Android application itself and the java API.

Another drawback was the data persistence in Docker. While developing USafe, I have found that although Docker offers major benefits. It makes the data persistence more difficult to deal with as any data in a container would disappear when the container goes down.

3.5 UI/UX

I do think that a simple design is really important as it will help the user to navigate through the application more easily. As a result, each view of the application only contains a minimum information to make it easier for the user to locate the desired action to perform.

However, I have took care not to remove all information blindly as it would prevent users from fully be able to use the application. Furthermore, a lot of features have been thought in an automated way (running detection for example), in order to minimise user's manual interaction.

I tried to minimise user input as much as possible as typing on a mobile device is not the most comfortable thing. To make the use of the application more comfortable I used the place autocomplete to enable users to enter a destination with more facility that they would have with typing the entire address. I also customised the keyboard for each type of input, for example display the numeric keyboard for a phone number.

3.6 Technologies

Java

Java is the main technology I have used in the project. This for several reasons:

1. Android is mainly based in Java
2. Java has a lot of mature frameworks such as Spring Boot
3. Java is well known language that is broadly used. That ensured me I could find enough support and information online

Android

Usafe is a mobile application. As I wanted to use features from the phone itself such as detecting movement that are easier to access than from a web application. Also, it allows greater responsiveness than a web application.

Spring Boot

I did not want my android application to hold all the business logic. Only the logic that was making sense, such as displaying the google map, searching addresses. As a result, I separated the UI logic from the pure business logic in another application that acts a REST endpoint to retrieve information such as the number of lights, all the garda stations that are on a path or check if a user exists.

This way, it will be easier to extend the application with an iOS or a Web application. All the common features will be accessible through REST endpoints and only the display logic will be necessary to implement with new apps.

Modules

Spring boot has a lot of different modules. Here is a list of modules that I have used in Usafe.

Spring MVC

Spring MVC allows to easily create and expose REST endpoints.

Spring DATA

Spring data is a Spring module that provides a unified way to implement the DAO layer. As I was using different Databases (MySQL, ElasticSearch), such technology was particularly interesting for me.

Geotools

Usafe extensively uses Geospatial Data. As Geospatial data is quite a complex topic, I decided to use a library dedicated to deal with those. This was particularly useful in order to translate different References Systems (from Irish National Grid easting, northing to World Geodetic System latitude, longitude for example) that were present in the different datasets I was using.

Docker

Building an architecture can become a complex task when it comes to: Choosing the technologies to be used, maintaining them or even dealing with security. Indeed, open ports to a component of the system can cause a lot of damages when accessed by the wrong persons. Another question can also be, what happens if my computer / server would stop working? That would mean every component would need to be reinstalled fully.

Docker helped me solved those issues. The technology can be seen as a declarative way to create architectures. It is mainly composed of a Dockerfile that declaratively contains all the steps to take to build the component. It then possible by simple commands to start the component. Not only Docker offers a flexible way to create components, it also allows to deal with networking between them. As a result, it allows to keep all components safe together and only exposes what the users need to see.

MySQL

In order to store persistent data, I have chosen to use MySQL. The database stores all the information about the users and their active journeys. This was sufficient for me as it was not requiring complex manipulation of the data. While offering all the features required, such as storing geospatial data.

ElasticSearch

For more complex data that will eventually grow a lot more than the users, and requires more complex treatment. I have decided to use ElasticSearch.

ElasticSearch is a powerful distributed search engine based on the Lucene library that can deal with a lot more data than a regular database. It has very strong capabilities in dealing with geospatial data, which made it a first choice for me to store information about data concerning the lightnings and garda stations that, today are in Dublin. But in the future, will be around the globe.

Firebase Cloud Messaging

Firebase is known to be a database for Android applications. Although it was tempting to use it in the first place, I have decided to keep a MySQL database for that purpose.

However, the library also has a Cloud Messaging module that revealed to be useful to implement the notification features.

Indeed, Firebase Cloud Messaging allow users to subscribe to topics. It is then possible for any user to send notifications and be alerted automatically when they receive one.

GoogleMaps API

At the heart of Usafe is the GoogleMaps API. The library allowed me to display a world map to the users. Furthermore, it provides a developer friendly API to interact with it. This allows to easily display paths and interactive information to users.

Also, the API provides a Search and a Geocoding API that allow users to search for addresses and translate those into actual latitude and longitude data. It was then possible for me to use those for my searches, display and other needs.

MapQuest by Verizon

One drawback of the GoogleMaps API is that certain components uses billing. Which quickly raised concerns to me over cost per requests made by users through the application. This, was the case of their Direction API that was required for me to determine the walking path between 2 places.

As a result, I have decided to look for another API that would allow me to use such feature and found the MapQuest Direction API developed by Verizon. As opposed to Google's Direction API, MapQuest had a monthly limit of 15000 requests. Which would be more than enough for me considering today's situation.

Google Activity Recognition API

The Activity Recognition API is built on top of the sensors available in a device. It is using machine learning models that process bursts of sensor data from the device's sensors. The API returns a list of activities that the user may be performing (walking, running, cycling…) with a confidence property for each activity (integer from 0 to 100). In my project I used it to detect if a commuter is running.

Retrofit

Retrofit is a REST Client for Java and Android. It makes it easier to retrieve and upload JSON via a REST based webservice. Retrofit lets you configure which converter to use for data serialization. In my case, I have used the recommended one: GSON from Google in order to transform POJOs into JSON data.

Git

I used Git in my project for version control. It is a very useful tool to keep a version history of my project, see the changes I have done and revert to the latest version when necessary.

Maven

Maven is a powerful project management tool that is used for projects build, dependency and documentation.

OpenData Datasets

In addition to Geospatial Data, Usafe also relies on Open Data in order to access large amounts of data. Similar to Open Source, Open Data is data that can be freely used, re-used and redistributed.

In Usafe's context I have used data from the Irish government Open data portal. The portal contains more than 10 000 datasets across different categories. The ones that have been used in my project are the public lightening locations and the crimes at garda stations in order to get the stations location.

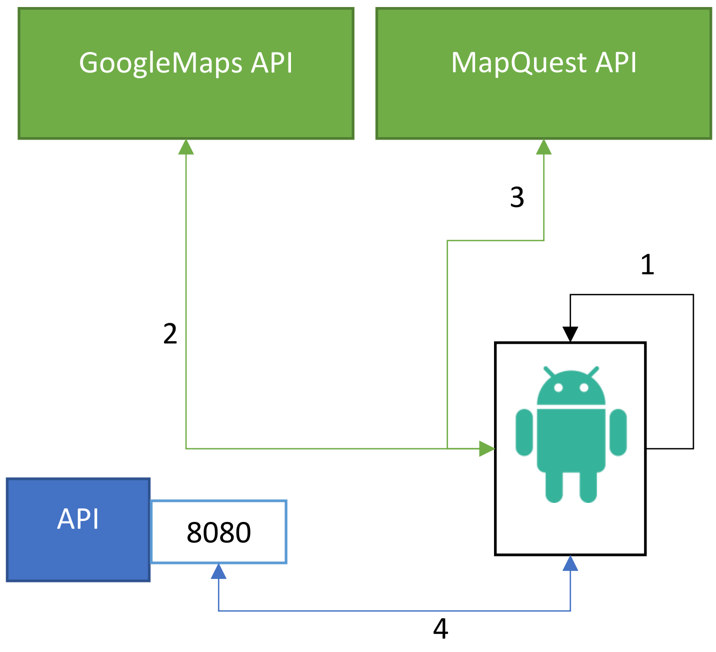
4 Implementation

4.1 Journey lifecycle

4.1.1 Start a journey

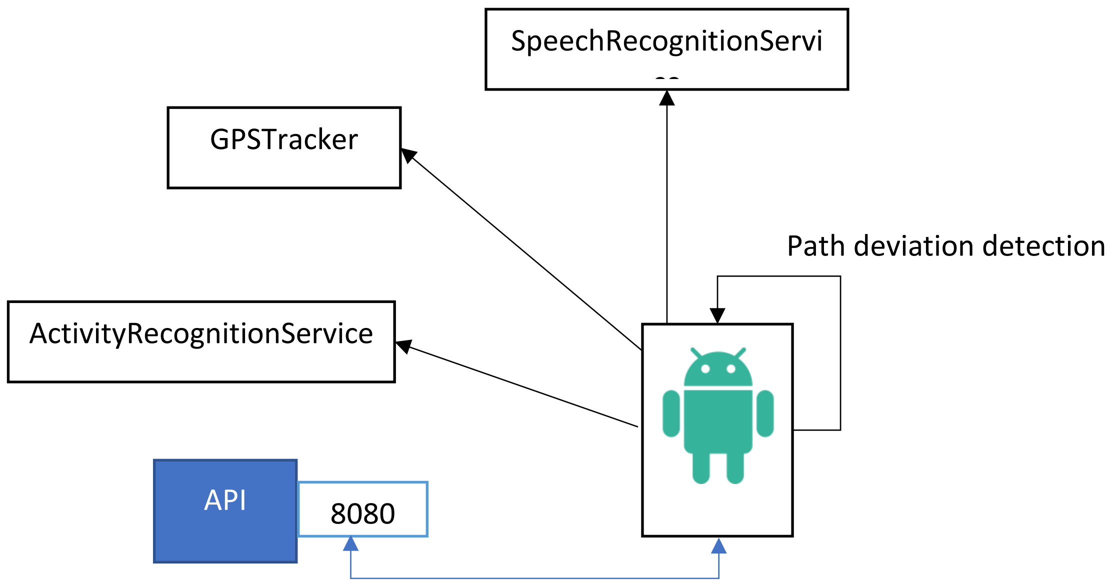
This section describes how the workflow to start a Journey has been implemented.

1. The commuter presses the Start Journey button on the main screen
   a. **Optional**: He picks some followers from the proposed contact list.
   b. He starts typing his destination address.
2. GoogleMaps API Autocomplete then search and output a list of relevant results.
   a. The user selects his destination address from the output results
3. Once the address selected, the destination address and the current location of the user are sent to the MapQuestAPI to find all the possible routes for the commuter.
4. All the routes are sent to the internal API which finds out how many public lights and how many garda stations are on each route
   a. The score gets calculated and sent back to the Android client.
   b. All the routes and their score are displayed on the map.
   c. The user selects the route that he prefers by clicking on the desired route.
   d. The route is saved as a Journey object that includes the current user location, a Path object which represents all the route's segments in a LinkedList Data Structure format. The Journey is set as active. Meaning, it is possible from the system to know if the user is currently commuting.



4.1.2 Performing the Journey

The commuter is now following the route he chose while starting a new journey. The below workflow illustrates the implementation workflow of an on-going journey.



When a commuter is on an active Journey, the USafe Android client performs several operations in background. Those operations allow the application to keep the commuter as safe as possible with the least action from the commuter as possible.

GPS Tracker

The GPS Tracker is an asynchronous task running every 5 seconds. The objective of this task is to get the current user's position. Once the latest location fetched, it is saved in the matching journey in the MySQL database in order to keep track of the latest user's location so followers can see where the commuter in near real time is.

Path deviation detection

When getting the most recent location of the commuter, the Android client performs a check of the latest location against the route that has been chosen. If the commuter is detected more than 10 meters away from the planned route, the client sends a notification to the followers. The process of sending notifications will be described at a later stage of this report.

Activity Recognition Service

The Activity Recognition Service is part of the smart features from USafe. It uses Google's Activity Recognition API that uses machine learning in order to detect the type of activity a user is performing as well as when a change in activity type has happened. This allow the client to be able to detect when the commuter went from walking to running. When the change is detected, the Android client prompts a popup window to the commuter waiting for a feedback. If no feedback from the commuter has been received, the Android client will send a notification to the journey followers.

Speech Recognition Service

Another smart feature in USafe is the Speech Recognition Service. This feature uses a speech recognition library that performs speech to text translation. Using the library, USafe let users define a text keyword that is then used by the speech recognition service to detect the user is danger. When the commuter says the keyword during an active journey, USafe's Android client will call the emergency contact defined in user's preferences or 999 (emergency telephone number) by default.

4.1.3 Stopping a journey

Once the commuter has finished his journey (arrived at destination for example), he can choose to stop its active Journey and come back to the main screen.

In order to do so, from the journey screen, he presses the stop button in the middle. This action sends a signal to the internal API that sets the journey as inactive. As a result, all journeys made by a user are stored in database.

All followers are notified the commuter has ended his journey.

4.2 Calculating the safety score

USafe's main goal is to provide and comprehensive and easy way to differentiate 2 routes leading to their destination. Allowing them to choose the safest for them according their own preferences. In order to achieve this, USafe calculates a score, currently based on 2 criterias: the number of public lights every 15 meters, the number of Garda station every 300 meters.

The distance between each public light have been decided based on an article from City Metric ("What makes people feel safe at night? On the science of street lights", 2018) that states the normal height of a light pole is between 4.5 to 6m, with a spacing of roughly 2.5 to 3 times this distance. Giving a minimum of 11.25 meters between 2 poles and a maximum of 18 meters between 2 poles. Leading to an average of 14,625 meters between 2 poles, which finally gives 15 meters when rounded.

Data about average distances of 2 garda stations opened 24 hours have been taken from The Central Statistics Office ("Population breakdown by distance to services", n.d.) and gave an average value of 1.9km in Dublin city. As I have decided to focus mainly on Dublin city first, it made sense to me to start by using this value as a reference.

The full score is a weighted average that depends on:

- The importance of each parameter defined by the user
- The constant values described as above for each parameter
- The actual value of each parameter (number of lights, number of garda stations)
- The total distance of the route

Giving the following formula:

$$\text{score} = 5\left(\frac{10x}{d} * p1 + \frac{1900y}{d} * p2\right)$$

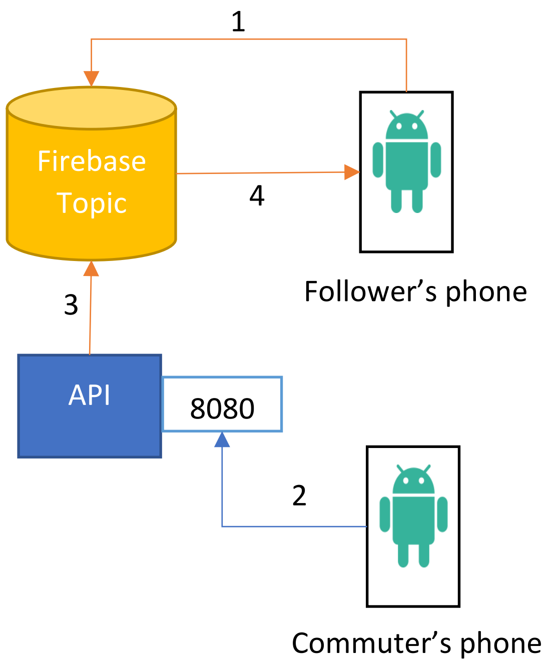
Where:

- **d** is the total distance of the route
- **x** is the number of lights found on the route
- **y** is the number of garda stations found on the route
- **p1** is the user's preference for the weight of lights
- **p2** is the user's preference for the weight of garda stations
- The constant **5** gives a final score out of 5

4.3 Sending a notification

Another core feature in USafe is the ability to send notifications to the commuter's followers whenever a problem is detected. The following schema illustrates how this feature is implemented.

1. When the user login, he subscribes to a Firebase Topic. The Firebase Topic acts as a Queue that sends messages when a new one arrives in the queue. This allows the follower to be notified in real-time.
2. A commuter is about to start a journey and has selected some of his contacts as followers. The commuter starts his journey by selecting a route.
    a. A message is sent to the internal API. The reason why the message is not sent to directly to the Firebase topic is because the Firebase Cloud Messaging API requires a server to send the messages.
3. The API sends the message to the Firebase Topics of each Follower.
4. Once the message is in the topic, the Android client's phone receives the message and displays the notification associated to the message.



5 Test Plan

Action	Input	Expected Output	Actual Output	Test Result
Access application	Click on app's logo	Login screen	Login screen	PASS
Fail login	Enter invalid email or password	Login screen with error message	Login screen with error message	PASS
User not filling all fields to login	Email or password not filled	Error message asking to fill all fields	Error message asking to fill all fields	PASS
Create account page link	Click on "No account, create one"	Create account form	Create account form	PASS
Fail create account	Click create button	Create account form displaying error message	Create account form displaying error message	PASS
Create account	Fill required information in create account form	Login screen with success message	Login screen with success message	PASS
Success login	Enter valid email Enter valid password	Map screen with user's location pin	Map screen with user's location pin	PASS
Add contact page	Click on menu, select add contact option	List of contacts from phone	List of contacts from phone	PASS
Add a contact	Click on the add button on the right side of the contact	Contact added to application's contact list	Contact added to application's contact list	PASS
Preferences page	From the menu bar, click on preferences	Display preferences page	Display preferences page	PASS
Edit score parameters	Change values of any parameter	Value updates accordingly. Total score should always be 100.	Value updates accordingly. Total score should always be 100.	PASS
Select emergency contact	Click on 'Select a contact' button	Access phone's contact and add the person as emergency contact	Access phone's contact and add the person as emergency contact	PASS

Define emergency keyword	Fill the field with the desired emergency keyword and press save preferences button	The text above the input field gets updated with the new value	The text above the input field gets updated with the new value	PASS
Save preferences	Click on save preferences button	Score and emergency contact values are saved. Values will be the saved one when coming back on the page.	Score and emergency contact values are saved. Values will be the saved one when coming back on the page.	PASS
Access start journey page	Press “Start Navigation” Button	Start journey screen displayed	Start journey screen displayed	PASS
Fail to start journey	Press “Start” button without filling destination	Start journey screen displayed with error message	Start journey screen displayed with error message	PASS
Search for followers	Type the name of a previously added contact in the app in Search bar	Desired contact displayed in search result list	Desired contact displayed in search result list	PASS
Add follower	Press on desired contact name from search result list	The name of the contact appears in the “following” list	The name of the contact appears in the “following” list	PASS
Remove follower from “following” list	Click on the follower remove button.	Follower is removed from the “following” list	Follower is removed from the “following” list	PASS
Start a journey	From previous steps, press Start button	Select journey screen is displayed. List of paths displayed with a safety score	Select journey screen is displayed. List of paths displayed with a safety score	PASS
Select a journey	Press on any path displayed from above step	Map screen displayed with the path to follow. A notification is sent to followers (if any)	Map screen displayed with the path to follow. A notification is sent to followers (if any)	PASS

Follow the user journey	Press "Follow user journey" from App notification	The app opens on the map, the follower can see the path chosen by the commuter and see the commuter's location	The app opens on the map, the follower can see the path chosen by the commuter and see the commuter's location	PASS
Detect if the user is running	During journey, user is suddenly running	Popup message displayed on screen asking if the user is safe	Popup message displayed on screen asking if the user is safe	PASS
User is safe (running detection)	User clicks on "yes" button (popup message)	Popup dismissed	Popup dismissed	PASS
Warn followers that user is running	User not replying to popup message in 20 seconds	Notification sent to followers	Notification sent to followers	PASS
Warn followers that user is deviating from his route	Commuter deviating from his route	Notification sent to followers	Notification sent to followers	PASS
Open phone dialler	Follower click on notification (user running or deviating)	Opens phone dialler with user's phone number	Opens phone dialler with user's phone number	PASS
Record video	Commuter clicks on the video button	Opens phone's camera in video mode	Opens phone's camera in video mode	PASS
Detect commuter is in danger	Commuter is saying 'help'	Calling emergency contact	Calling emergency contact	PASS
Ends journey	On map, press "Stop button"	Notification sent to followers and redirected on the main page	Notification sent to followers and redirected on the main page	PASS
Logout	Click logout from menu bar	Login page	Login page	PASS

6 Installation Manual

This section explains how it is possible to setup and use USafe. All the steps are explained in the subsections and may require some pre-requisites from the user.

It is assumed that USafe's Git repository have already been cloned from Github.

6.1 Install the API

6.1.1 Pre-requisites

Maven will be required in order to build the API's jar. In order to install Maven, follow the instruction described in the following link: https://maven.apache.org/users/index.html.

Docker must be installed in order to run the API. In order to install Docker, follow the instructions described in the following link: https://docs.docker.com/docker-for-windows/install/. Although this link describes the installation for Windows users, it is possible to find a section for Mac and Linux users.

6.1.2 Create a jar of the API

First, the user will need to create an executable jar of the API. In order to do so, go inside the API directory from USafe's main directory.

Execute the following command: mvn clean package

This will download all USafe's dependencies of the API and create an executable jar file.

6.1.3 Start the API Stack

After the executable jar have been created, the user will be able to start USafe's API stack. From the API directory, execute the following commands:

1. docker-compose up mysql

2. docker-compose up elasticsearch

3. docker-compose up –build usafe-api

The first step will download and start a container running a MySQL database. The second step will download and start a container running Elasticsearch. Finally, the third step will build an image of the API and start a container running the jar build in the above section.

All the components of the API are now running. The user can now run USafe's Android Client in order to use the application.

6.2 Install the Android Client

6.2.1 Pre-requisites

In order to install the Android client, the user will have to download and install Android Studio. The following link explains the steps in order to download the software: https://developer.android.com/studio.

The IP of the computer where the API is running must be known.

6.2.2 Running the Android client

From Android Studio, open the android_client folder in Usafe's main folder as a project.

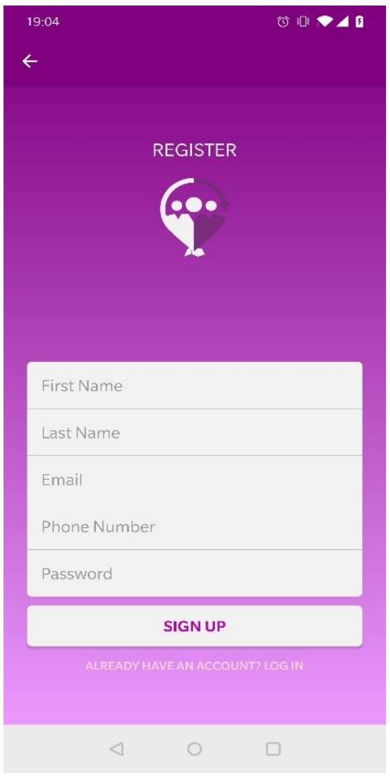
Then, plug the phone to the computer in order to have it available as a device in Android Studio.

Update the content of the property: home_IP inside the strings.xml file. This will also be necessary to be made in the domain attribute in the network_security_config.xml file.

Finally, from Android Studio, click the run button, making sure that your android device selected is yours.

7 User Manual

7.1.1 Create an account

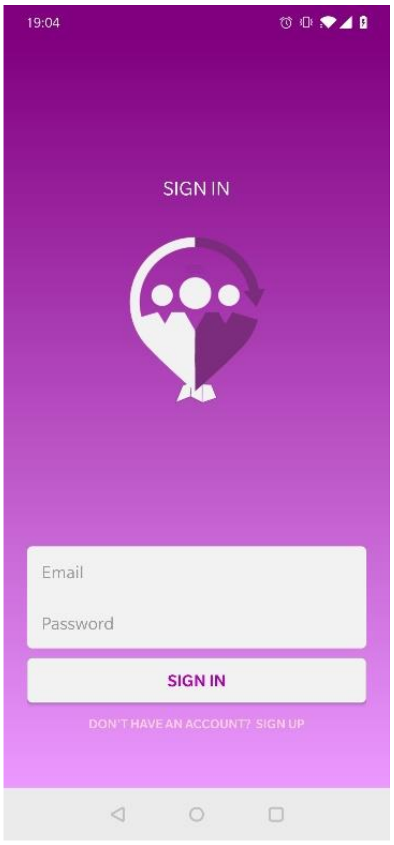


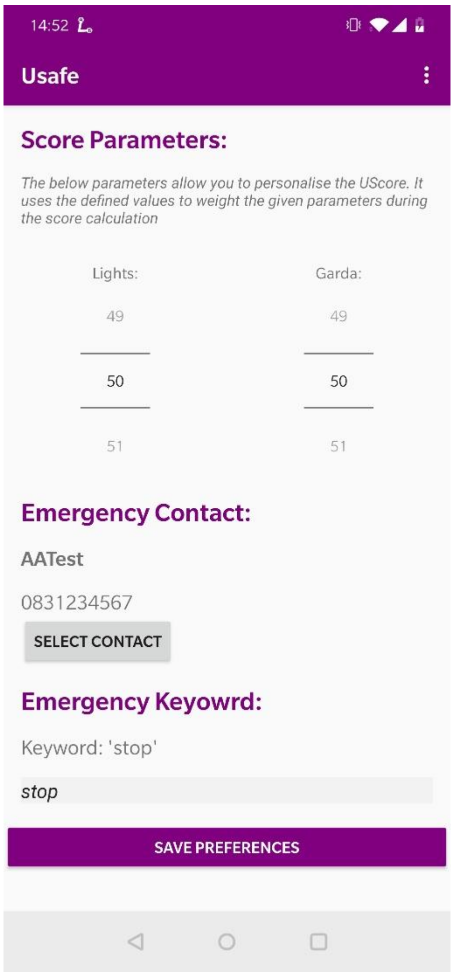
In order to be able to use the application, the user must create an account. To access the registration page when launching Usafe on the login page, the user needs to click on the link "Don't have an account? Sign up". Then, the user must fill in all the fields such as first name, last name, email, phone number and password. The email address and the phone number have to respect a specific format: email address must be of format: email@domain.com, and phone number must not contain country code prefix. Password needs to have minimum 8 characters. Lastly, email address must be unique. If any of those conditions are not fulfilled, a corresponding error message will be displayed and the account will not be created.

7.1.2 Login

To sign in, the user must provide the email address and password he used to register. If any of these information is incorrect, an error message will be displayed.

If login is successful, the user will be redirected to the main screen of the application.




14:52

Usafe

Score Parameters:

The below parameters allow you to personalise the UScore. It uses the defined values to weight the given parameters during the score calculation

Lights:	Garda:
49	49
50	50
51	51

Emergency Contact:

AATest

0831234567

SELECT CONTACT

Emergency Keyowrd:

Keyword: 'stop'

stop

SAVE PREFERENCES

7.1.3 Define user preferences

Score parameters

This section allows the user to personalise the Safety Score. The lights and garda stations values will be used as a weight in the score calculation.

The default values are 50 for lights and 50 for garda stations.

Emergency contacts

Usafe allows the user to select a contact from his phone as a contact to be called in case of an emergency.

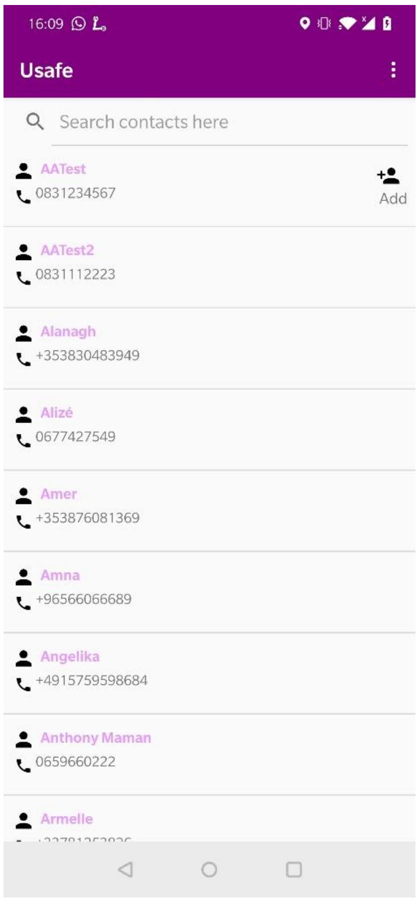
The default emergency contact is 999.

Emergency keyword

The user can choose an emergency keyword. The default one is ‘help’. If the given keyword is detected during a journey (meaning the user would have said it), the application will immediately call the emergency contact.

7.1.4 Add a contact

The user will be prompted for permission to access the phone's contacts. Once the permission granted, phone's contacts will be displayed. The user will be able to add contacts who are also Usafe's users as friends. The user will then be able to add those friends as followers when starting a journey.



Search contacts here

AATest
0831234567

AATest2
0831112223

Alanagh
+353830483949

Alizé
0677427549

Amer
+353876081369

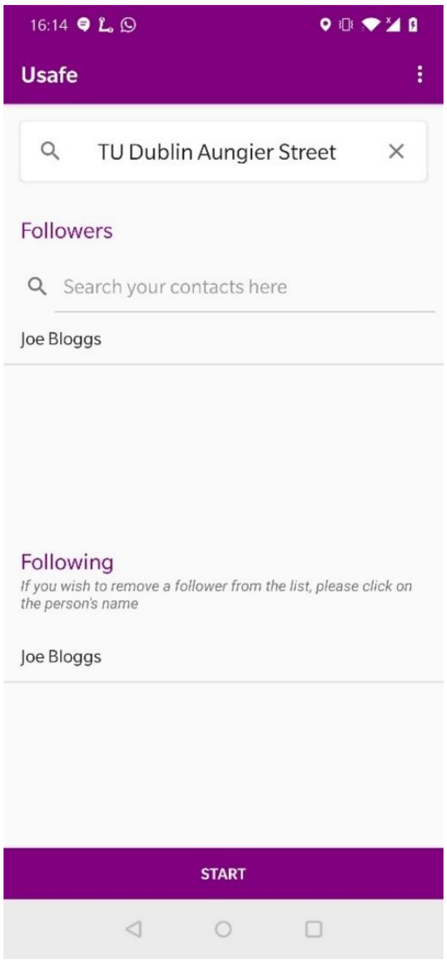
Amna
+96566066689

Angelika
+4915759598684

Anthony Maman
0659660222

Armelle

7.1.5 Start a journey



TU Dublin Aungier Street

Followers

Search your contacts here

Joe Bloggs

Following

If you wish to remove a follower from the list, please click on the person's name

Joe Bloggs

Select your destination

In order to start a journey, the commuter must provide a destination by filling the Google autocomplete field. After typing the first characters of his destination address, Google autocomplete will instantly provide a list of potential matching addresses.

Add some followers

Commuter's friends will be displayed in a list as followers. Commuter will be able to select some of them before starting the journey. Selected ones will then be displayed in the 'Following' list.

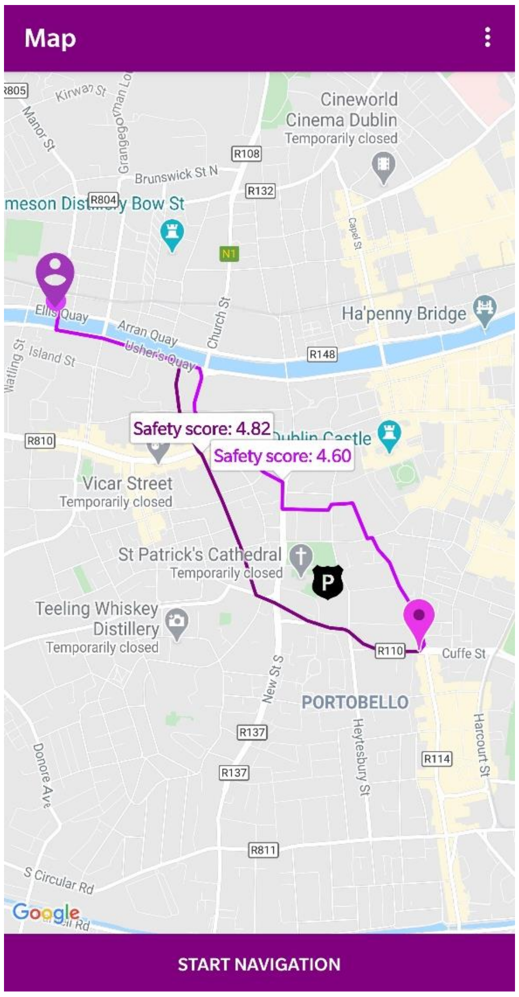
Remove some followers

Commuter can remove some followers from the 'following' list by clicking on the follower's name.

7.1.6 Select your route

Once the destination and the followers are chosen, a set of routes from commuter's location to the final destination will be displayed on the map.

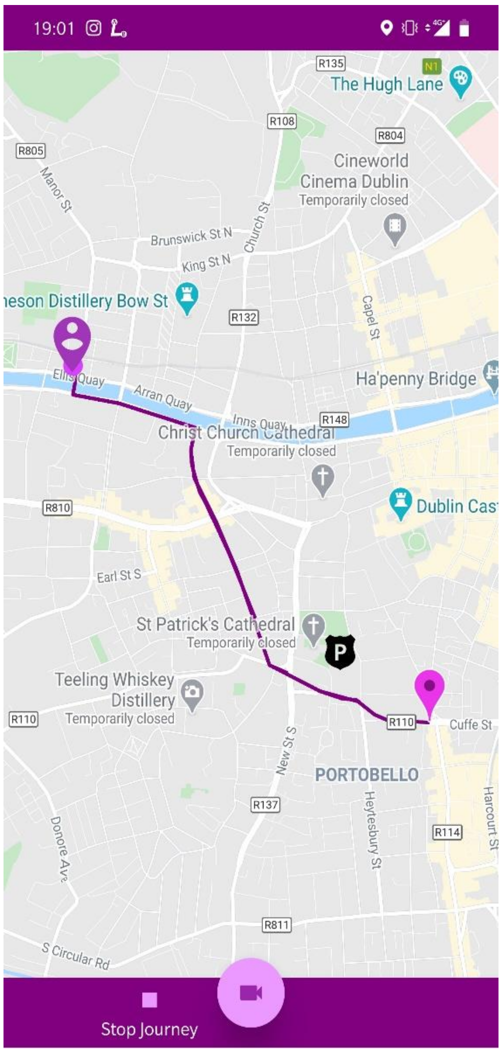
The commuter will be able to select one of the proposed routes. Each route will have a safety score (out of 5) that represents how safe the route is based on his preferences.



7.1.7 Journey features

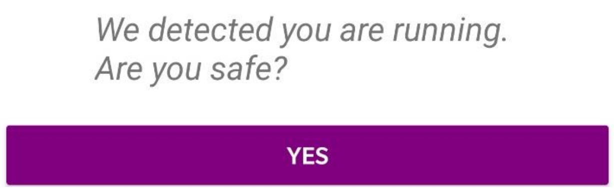
Route deviation

During the journey, the application can identify if the commuter is not following the route. In that case, a notification will be sent to the followers indicating that the commuter is deviating from his route. If the commuter has no followers, it will instead call the emergency contact set in the profile section.



End-user running

USafe has also the feature to detect if the commuter is running during his journey. As a result, a popup notification will be displayed on the commuter's phone asking if everything is alright. If the commuter is not pressing the 'yes' button after 20 seconds, followers will be notified. If the commuter has no followers, it will call instead the emergency contact.


We detected you are running.
Are you safe?

YES

Calling for help

In a case that the emergency keyword is detected by the speech recognition, the application will immediately call the emergency contact.

Take a snapshot video

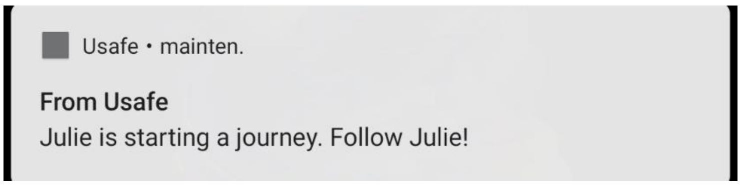
In the event of potential threat, the commuter can click on the 'SOS' button. By doing so, the application will open the phone's camera and the user will be able to record a video.

Stop journey

Once the journey completed, pressing the stop button will notify the followers that the journey is finished.

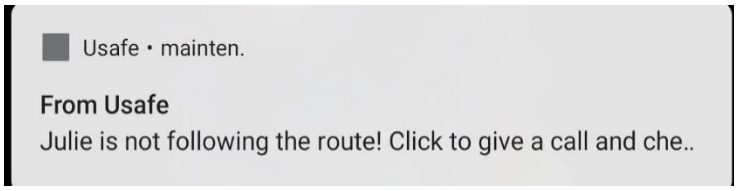
7.1.8 Followers features

Starting journey notification


Usafe • mainten.
From Usafe
Julie is starting a journey. Follow Julie!

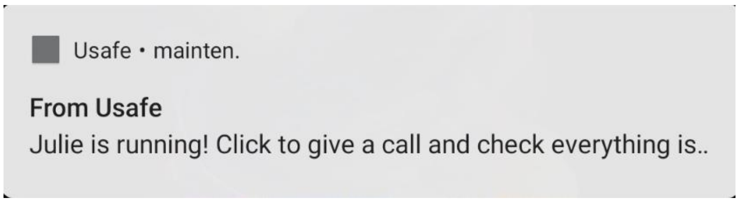
As soon as the commuter is starting the journey, followers will be notified. By clicking on the notification, followers will be able to see the chosen route and follow the commuter’s journey.

Route deviation notification


Usafe • mainten.
From Usafe
Julie is not following the route! Click to give a call and che..

By clicking on the notification, the application will open the phone dialler with the commuter’s phone number. This way it makes it easier for the follower to call the user.

Running detection notification


Usafe • mainten.
From Usafe
Julie is running! Click to give a call and check everything is..

Same behaviour as the route deviation notification, clicking will open the phone dialler with the commuter’s phone number.

8 Conclusion

The idea of implementing this project was really dear to me. I have spent some time prior starting its implementation to think about what features could be added and what technologies could be used. I have been very happy to receive good features ideas coming from my supervisor and second reader that helped me pushing the original idea of the project. Making it feature-rich product that solves the problem it was built for.

I have been very happy to discover, learn and use a large range of technologies and APIs that contribute to build my technical knowledge. This project have challenged me a lot in different areas and I have learnt a lot from building it.

I wish to continue contributing to my project after completing my degree. This, by adding new features and datasets that would enrich the safety score provided by the application.

Leading eventually to a release on the market.

9 References

What is open data? Retrieved from Open Data Handbook https://opendatahandbook.org/guide/en/what-is-open-data/

Public Lighting Locations. Retrieved from DATA.GOV.IE https://data.gov.ie/dataset/public-lighting-locations

Crimes at Garda Stations Level 2010-2016. Retrieved from DATA.GOV.IE https://data.gov.ie/dataset/crimes-at-garda-stations-level-2010-2016

Saul McLeod (2020, March 20). *Maslow's Hierarchy of Needs*. Retrieved from Simply Psychology https://www.simplypsychology.org/maslow.html

Safe Cities Index 2019. Retrieved from The Economist https://safecities.economist.com/safe-cities-index-2019/

(2019, January 24) Crime Figures : Violent crime recorded by police rises by 19%. Retrieved from BBC News https://www.bbc.com/news/uk-46984559

Statistics. Retrieved from Stop Street Harassment http://www.stopstreetharassment.org/resources/statistics/

2018 Study on Sexual Harassment and Assault. Retrieved from Stop Street Harassment http://www.stopstreetharassment.org/our-work/nationalstudy/2018-national-sexual-abuse-report/

Crime drop. Retrieved from Wikipedia https://en.wikipedia.org/wiki/Crime_drop

The Conversation (2018, October 24)What makes people feel safe at night? On the science of street lights. Retrieved from City Metric https://www.citymetric.com/fabric/what-makes-people-feel-safe-night-science-street-lights-4251

Extract, Transform, Load. Retrieved from Wikipedia https://en.wikipedia.org/wiki/Extract,_transform,_load

Manage data in Docker. Retrieved from Docker Docs https://docs.docker.com/storage/

Activity Recognition API. Retrieved from https://developers.google.com/location-context/activity-recognition

Lighting Design Guidance. Retrieved from Global Designing Cities Initiative https://globaldesigningcities.org/publication/global-street-design-guide/utilities-and-infrastructure/lighting-and-technology/lighting-design-guidance/

Gotev speech recognition. Retrieved from https://github.com/gotev/android-speech

(2019, June 17). What is Geodata? A guide to geospatial data. Retrieved from GIS Geography https://gisgeography.com/what-is-geodata-geospatial-data/

(2014, July 8). What is Geographic Information Systems? Retrieved from GIS Geography https://gisgeography.com/what-gis-geographic-information-systems/

1854 Broad Street cholera outbreak. Retrieved from Wikipedia https://en.wikipedia.org/wiki/1854_Broad_Street_cholera_outbreak

(2015, September 29). Vector vs Raster: What's the difference between GIS Spatial Data Types? Retrieved from GIS Geography https://gisgeography.com/spatial-data-types-vector-raster/

Caitlin Dempsey (2017, May 1). Types of GIS Data Explored: Vector and Raster. Retrieved from GIS Lounge https://www.gislounge.com/geodatabases-explored-vector-and-raster-data/

Advantages and disadvantages of Raster & Vector Data. Retrieved from GIS RS GPS https://funwithvertices.blogspot.com/2018/08/advantages-and-disadvantages-of-raster.html

Spatial reference system. Retrieved from Wikipedia https://en.wikipedia.org/wiki/Spatial_reference_system

Irish grid reference system. Retrieved from Wikipedia https://en.wikipedia.org/wiki/Irish_grid_reference_system

Measuring Distance to Everyday Services in Ireland. Retrieved from Central Statistics Office https://www.cso.ie/en/releasesandpublications/ep/p-mdsi/measuringdistancetoeverydayservicesinireland/generalresults/

10 Lexical

10.1 ETL

ETL, in Data Warehousing means Extract Transform Load. It is the general procedure of copying data from one or more sources into a destination system which represents the data differently from the source(s) or in a different context than the source(s).