1. INTRODUCTION

Wildfires have a great impact on a considerable number of countries around the world, costing thousands of lives yearly and inflicting large societal and economic problems in the affected region. Being able to escape wildfires is often difficult and dangerous without prior knowledge of the fire propagation and the possible escape routes. The availability of modern technologies, such as smartphones, could significantly improve the provision of real-time information to ordinary citizens in order to help them to evacuate as soon as possible. The goal of EscapeWildFire is therefore to provide a framework for fire departments and governing bodies where they can enter information about wildfire observations, model the future spread of the fires and communicate escape routes to the public. The framework aims to provide this information with high precision and efficiency, based on the predicted spread of the wildfire and the real-time location of the end user.

A. Who manages and develops EscapeWildFire?

The EscapeWildFire framework is developed by RISE in Nicosia, Cyprus. It is open-source and can be adopted by any organization aiming to provide citizens with wildfire escape routes. The framework serves as a basis for an actual implementation, i.e. all components may be used, further developed and/or customized by a local, regional or national organization. Please note that the platform is dependent on three external components: the ForeFire API[1] for the fire propagation simulations, HERE XYZ[2] for the storage of geodata and Windy API[3] for wind speeds. Adopters of EscapeWildFire should therefore take into account that they agree with the terms of use of these products, and request their own personal API keys before the solution can be successfully and fairly implemented for widespread use.

B. Who can use the EscapeWildFire framework?

Anyone may use, modify and/or implement the EscapeWildFire framework. The intended users for the platform are fire departments, fire prevention agencies or governments (either on a local, regional or national level) who use EscapeWildFire as a starting point to build their own system which can provide citizens with real-time escape route information.

When using EscapeWildFire in any way, please remember to credit RISE with the creation and initial development of the framework.

2. OVERVIEW OF FRAMEWORK

The EscapeWildFire platform consists of three main components: the **incident/fire management application**, the **fire propagation simulator** and the **mobile application.** Responsible organizations and governments use the fire

[1] https://firecaster.universita.corsica/
[2] https://www.here.xyz/
[3] https://api.windy.com/point-forecast

management system, which is a web application where wildfires can be added, removed and modified. No prior knowledge about wildfire modeling or programming is required – the ignition point can simply be entered into the fire management application using a GUI, after which the system automatically propagates the fire and communicates this via the mobile application to the end user. In Figure 1, a comprehensive schematic overview of the framework is provided.

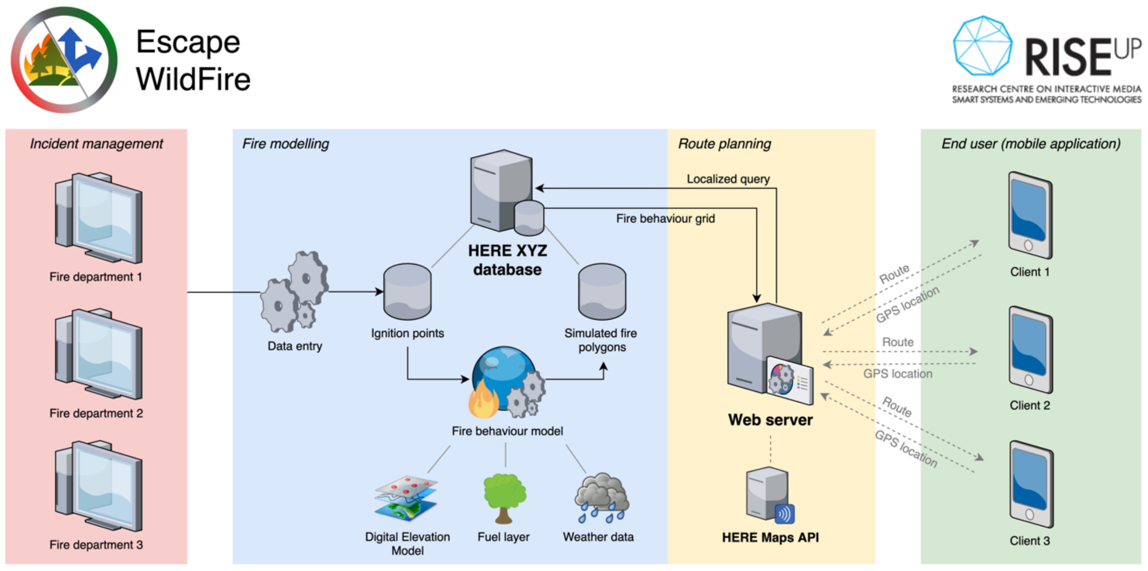


Figure 1. Schematic of the EscapeWildFire framework

3. MOBILE APPLICATION[4]

The mobile application will used by the end users, i.e. citizens who want to be able to find escape routes whenever a wildfire is near. The app is developed in Android Studio, using the Kotlin programming language. It leverages the routing capabilities of the HERE Maps API[5].

A. App structure

The app is divided into two views, the design of which will be discussed later. An activity is associated with each of these views. These activities are MainActivity (associated with activity_main, the home screen) and MapActivity (associated with map_view, which is used to display the map and navigation). These two activities contain most of the code of the app, however there are also some separate classes. The most important of these are the ApiHandler and the FireDataHandler.

i. MainActivity

MainActivity handles everything required to have the app function, such as checking permissions and setting up a cache path for the map, which is required for the map to function. It also attaches listeners to the buttons. These listeners are pieces of code that is run when their respective action has occurred. An

[4] Couwenbergh, W. (2020). *Real-time assistance for people endangered by forest fires*. University of Twente.

[5] https://developer.here.com/

onClickListener for example will run when a button has been clicked. Using these listeners, the actions being the buttons are set. The last thing that MainActivity is responsible for is starting the ApiHandler and having it run every 15 minutes.

ii. ApiHandler

The ApiHandler is used to retrieve the data about the fire from the HERE XYZ web service through the associated REST API. It can request data from the web service by calling a resource method of the API. In this case a get request is used to retrieve the data about the fire. The data about the fire's progression is stored in the form of polygons that describe the fire at its different stages (e.g. now, fifteen minutes into the future, thirty minutes into the future, etc). These polygons are returned by the REST API in the form of a GeoJSON file, which is shaped exactly like a JSON file however has predefined ways to store various different shapes (e.g. lines, polygons, etc.). The ApiHandler then uses the GSON library to convert the (Geo)JSON file to a FireData objects that can be easily used in code. This FireData object contains the polygons to describe the fire and all the coordinates to generate these polygons. Moreover, it contains most of the function needed to retrieve and interact with the fire data. These objects are then passed along to the FireDataHandler.

iii. FireDataHandler

FireDataHandler is used to store the information about the fires retrieved from HERE XYZ and will act as a go between for the MapActivity and the data for the various fires. The FireData is added to the handler through the setFireData function, which is called by the ApiHandler. This function also tells the FireData object to generate the polygons (if the map has been initialized).

iv. MapActivity

MapActivity is where the bulk of the code can be found. This activity is responsible for handling the map and navigation, which also includes generating the destination and route to get to this destination. The way the route is currently being calculated is done as follows. First the user's heading with regards to the fire is calculated. This heading is in degrees, like the degrees on a compass. A heading for the fire itself is also calculated, to ascertain the direction in which the fire is moving. It is then made sure that the user's heading is adjusted such that the difference between it and the fire is at least 45 degrees. This heading together with a set distance is then used to calculate a destination. In the current prototype this distance has been set at 10km, however this was purely for testing purposes. The aim is to have this be set dynamically in the future. Before generating a route, it is also checked if the user is not already within one of the polygons describing the fire. If this is the case, the nearest coordinate outside if that polygon is obtained and set as the first waypoint in the route generation. Then the waypoint for the final destination is added.

Finally, the dynamic penalty is set. The dynamic penalty is what allows the route calculation to avoid a certain region or take into account the traffic when generating a route. At this moment only a banned area is set (the region the route calculation cannot use), however taking traffic into account is planned in future version. The banned area is set to all the polygons that describe the fire before the polygon in which the user is located. If the user is in the outermost polygon, for example, the inner polygons will be set as a banned area. Here the centre of the inner most polygon (labelled now) is the fire as it is currently. The polygons around this labelled +1 through +3 are the polygons that describe the progression of the fire at different times. This is currently not yet set, however each increment could for example be an extra 15 minutes into the future. This could mean that +1 is +15 minutes, +2 is +30 minutes, etc. The green dot indicates the user's location in this illustration. So, in this case the polygons labelled now, +1 and +2 will be set as a banned area since the user is in the area labelled +3.

B. App design

The app consists of two separate views: the *home screen* and the *map view*. The home screen contains two toggle buttons, one to select the mode of transport and one to select the type of navigation. It also contains an image view which gives a short explanation about the two different types of navigation. And finally, it contains the Show Map button, which lead the user to the map view.

The map view is responsible for anything that involves the map, whether that is viewing the fire on the map or navigating. In Figure 2, the different modes of navigation (none, turn-by-turn and directional) are displayed.

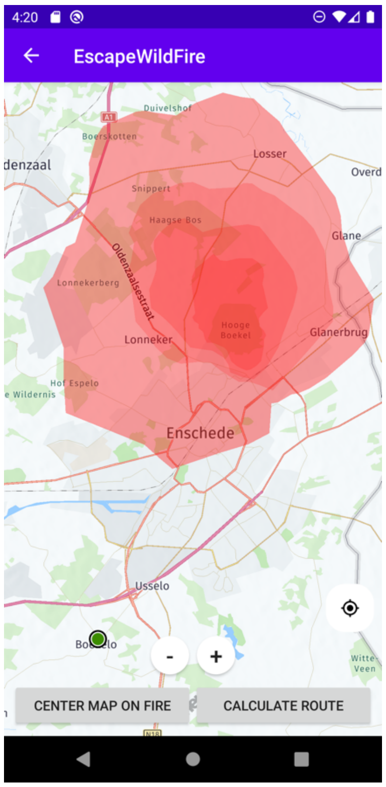


Figure 2a: Screenshot of map view (no navigation)

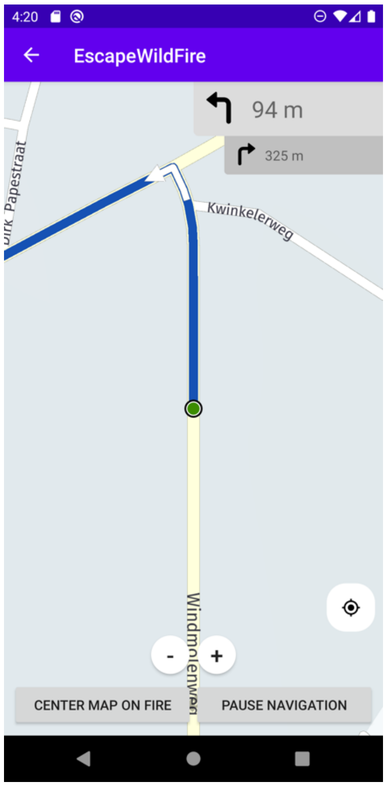


Figure 2b: Screenshot of turn-by-turn navigation

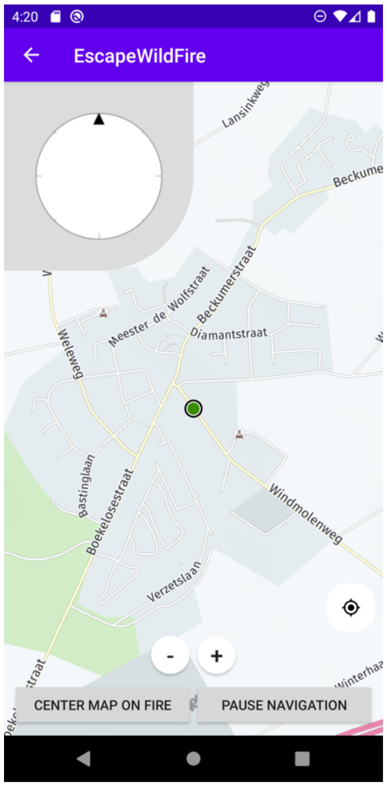


Figure 2c: Screenshot of directional navigation

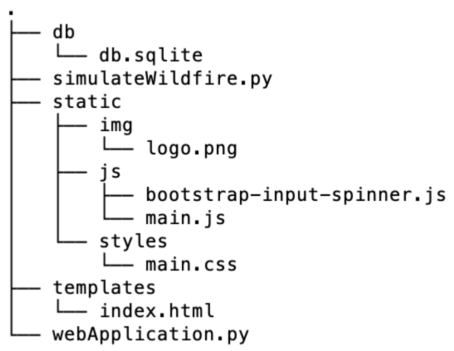
4. FIRE SIMULATION

The ForeFire API[6] (developed at the Università di Corsica Pasquale Paoli) is used to simulate the fire propagation in the near future. After specifying a coordinate pair and wind speed in both X and Y direction, the API returns the coordinates of the fire front at the given timestamp. Please note that, before developing a commercial or large-scale implementation of the EscapeWildFire framework including the ForeFire API, an agreement should be made with the authors. More information can be found here.

To produce an accurate simulation of the wildfire, the ForeFire API needs wind speeds in the X and Y direction. The EscapeWildFire can automatically retrieve this data using its integration with the Windy API[7]. This returns the wind speed in km/h on the surface at a given location (specified using coordinates). A free API key can be obtained, allowing up to 500 requests per day.

5. SYSTEM CORE

This component serves as the main backbone of the system, since it connects the local wildfire database, the HERE XYZ database and the ForeFire API. The web application is targeted at governmental agencies and fire departments, i.e. the organizations who are responsible for management of wildfires within a certain territory. Users can enter or delete wildfires from a local database which will instantiate or terminate a simulation in the ForeFire API. The main scripts for (1) running the web server and (2) instantiating/terminating the fire simulator are both written in Python 3.8. The actual web-based interface was created using Bootstrap with some additional Javascript, and the backend service being run using Flask (a Python library). A local SQLite database called db.sqlite stores all the wildfires which have been created in the management system, as well as their current status. Both Python scripts directly perform SQL operations on the local database. The folder structure of the system core is shown in Figure 3.



```
.
├── db
│   └── db.sqlite
├── simulateWildfire.py
├── static
│   ├── img
│   │   └── logo.png
│   ├── js
│   │   ├── bootstrap-input-spinner.js
│   │   └── main.js
│   └── styles
│       └── main.css
├── templates
│   └── index.html
└── webApplication.py
```

Figure 3. Folder structure containing the system core scripts

The following Python scripts are essential to run the EscapeWildFire system:

A. webApplication.py

This script is the main script for running the entire EscapeWildFire system (except for the mobile application). Executing this script will spawn two separate processes:

[6] http://forefire.univ-corse.fr/

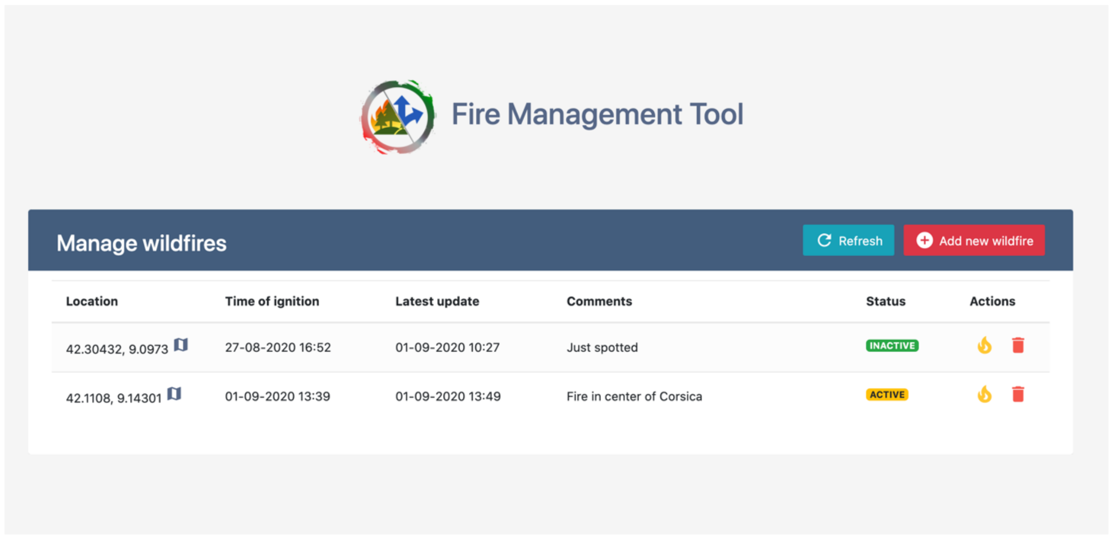
[7] https://api.windy.com/

- A continuous routine (i.e. executed every 10 seconds) of updating all wildfires in the database. This checks whether there are any active fires in the local database that have not been updated within the past 20 minutes. If this is the case, it will run the ForeFire simulator for one cycle, add the new polygons to HERE XYZ and update the local database.
- The Flask web server which runs the Fire Management application that is used by the fire department. The application has a minimalistic design, allowing employees to simply add, delete and (de)activate wildfires in the system. Choosing one of these actions triggers the corresponding function within the simulateWildfire.py script. See Figure 4 for a screenshot of the web application in the browser.

B. simulateWildfire.py

This script contains the dependencies of the main script. The methods in this script provide an interface to the local SQLite database, the ForeFire API and the HERE XYZ API. Therefore, the webApplication.py script makes calls to the functions inside this Python file.

Important: Before the EscapeWildFire system can function correctly, the HERE XYZ and Windy API authorization keys should be assigned to the designated variable (see code). It is therefore necessary to request this API key from HERE and Windy first, respectively.



Location	Time of ignition	Latest update	Comments	Status	Actions
42.30432, 9.0973	27-08-2020 16:52	01-09-2020 10:27	Just spotted	INACTIVE	
42.1108, 9.14301	01-09-2020 13:39	01-09-2020 13:49	Fire in center of Corsica	ACTIVE	

Figure 4. The Fire Management web application