

Universidad de Alcalá

Escuela Politécnica Superior

**Grado en Ingeniería en Tecnologías de la
Telecomunicación**

Trabajo Fin de Grado

Implementation of an Alert Management Application with
Support for Geographical Visualization

ESCUELA POLITECNICA
SUPERIOR

Author: Gustavo Martín Alcalde

Advisors: Javier Tejedor Noguerales y Javier Macías Guarasa

2014

UNIVERSIDAD DE ALCALÁ
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería en Tecnologías de la Telecomunicación

Trabajo Fin de Grado

Implementation of an Alert Management Application with
Support for Geographical Visualization

Author: Gustavo Martín Alcalde

Advisors: Javier Tejedor Noguerales y Javier Macías Guarasa

Tribunal:

President: Sira Elena Palazuelos Cagigas

1st Vocal: José Luis Martín Sánchez

2nd Vocal: Ernesto Martín Gorostiza

Calification:

Date:

To my parents.

Resumen

Hoy en día, las tuberías de distribución del gas natural ofrecen unas características de seguridad altas. Sin embargo, se han dado casos de accidentes con pérdidas humanas y materiales, debido a fallos humanos evitables. Para dar respuesta a esta demanda extra de seguridad, se creó el proyecto PITSTOP, que consiste en detectar posibles amenazas a las tuberías de gas. Este Trabajo Fin de Grado aborda una parte de ese proyecto relativa a la gestión de las alarmas generadas por el sistema de detección mediante el desarrollo de una aplicación con una interfaz intuitiva encargada de distintos procesos, incluyendo soporte para visualización geográfica de dichas alarmas basada en Google Maps.

Palabras clave: Gestión de alertas, información geográfica, google maps, programación con qt.

Abstract

Nowadays, natural gas pipes provide high safety features. However, there have been some accidents with human casualties and economical loses, due to third party works nearby. The PITSTOP project is the answer to this problem by detecting possible threats on the gas pipes. This bachelor thesis covers a part of the whole project regarding the management of the threats (so-called alarms) generated by the detection system through the development of an application with a intuitive interface, including support for geographical visualization of the alarms based on Google Maps.

Keywords: Alert handling, geographical information, google maps, qt programming library.

Extended Abstract

Nowadays, natural gas pipes provide high safety features. However, there have been some accidents with human casualties and economical losses due to third party works nearby that occasioned injuries in the pipes and gas leaks. The Project for the early detection of Pipeline Integrity Threats using a Smart Fiber-Optic Surveillance System (PITSTOP) is the answer to this problem by prematurely detecting possible threats on the gas pipes thanks to fiber technology, which allows to listen and classify the noise produced by heavy machinery works.

This bachelor thesis covers a part of the overall project regarding the alarm management through the development of an application (TRS-UI) with an intuitive interface and a visual map to simplify the location and handling of the generated alarms. The system is distributed and allows multiple users running the application at the same time, which provides the possibility to have different user roles at the same time with a specific task each.

The TRS-UI application needs the Threat Recognition System (TRS) and the database to be able to work properly. It will receive the generated alarms from the TRS and insert them automatically into the database. All users will have specific permissions to access and view certain parts of the application. It is possible to view the events, exclusions and create or modify them. The exclusions are areas in which if an event occurs, the alarm will not sound because of it.

The connection between the TRS-UI and TRS is performed by standard TCP/IP communication protocol. Both applications have one master module and one slave module to communicate with the slave and master of the other system respectively. The master will send command and actions requests to the slave of the other system, such as parameter changes, events and configuration options.

The database, whose design is described here, contains several tables and fields to store the necessary information for the application to work. This includes a list of user accounts, events, exclusions, application logs of recorded actions made by the users, configuration options and a geographical mapping between geographical coordinates and fiber optic positions that allows to translate them from one format to another.

This application has been developed in Qt [1], a cross-platform application framework used for developing software that can be run on different software and hardware platforms, such as Linux or Windows. The coding and the design of the graphical user interface has been carried out with the Qt Creator [2]. This application is split in several modules, which provides features or jobs. Each module is made of certain objects or classes. The objects are managed by one special object called Core, which is responsible of the application flow. This object is capable of starting different modules and objects, and making connections between them using the Qt Signals & Slots technology, in order to achieve a communication and data transport, such as data structures, between objects.

The evaluation tests have been carried out both during the developing process and at the end of it, to ensure the reliability of this software application. Even though the objectives of the project have

been reached, this software application is still a prototype and have not been tested in a complete real scenario. The PRS-UI application provides many features and advantages, although there are some suggested improvements for future work.

Contents

Resumen	vii
Abstract	ix
Extended Abstract	xi
Contents	xiii
List of Figures	xvii
1 Introduction	1
1.1 Motivation, Aims and Objectives	1
1.2 Overview of the Alert Management Application	1
1.3 Thesis Outline	2
2 System Description	5
2.1 Introduction	5
2.2 General Description of the PIT-STOP Working Context	5
2.3 System Specification	6
2.3.1 Generalities	6
2.3.2 Detailed Specification	6
2.4 System Functionality	14
2.5 Hardware Architecture	14
2.6 Software Architecture	15
2.7 Communication Protocol	16
2.8 Conclusions	16
3 Database Design	19
3.1 Introduction	19
3.2 Database Structure	20
3.2.1 Users	20
3.2.2 Events	21

3.2.3	Exclusions	21
3.2.4	Threat List	22
3.2.5	Geographical Mapping	22
3.2.6	Application Activity List	22
3.2.7	Application Log	23
3.2.8	Configuration	24
3.3	Conclusions	25
4	Implementation	27
4.1	Introduction	27
4.2	Main Modules	27
4.2.1	Core Module	27
4.2.2	User Login Module	28
4.2.3	Event Management Module	29
4.2.4	Exclusions Management Module	31
4.2.5	Alarm Handling Module	33
4.2.6	Application Log Management Module	35
4.2.7	Configuration Module	36
4.2.8	Database Daemon Module	37
4.2.9	Exiting the Application	38
4.3	Supporting Modules	38
4.3.1	Database Support Module	38
4.3.2	Alarm Handler Support Module	39
4.3.3	TRS Communication Support Module	39
4.3.4	Application Log Support Module	40
4.3.5	User Agent and Account Permissions System Support Module	40
4.4	Conclusions	41
5	Evaluation	43
5.1	Introduction	43
5.2	Evaluation Procedures and Results	43
5.2.1	Application access	43
5.2.2	Event Management Module	43
5.2.3	Application Log Module	43
5.2.4	Alarm Handler	44
5.2.5	Exclusion Management Module	44
5.2.6	Configuration	44
5.2.7	Daemon Service	44

5.2.8	Status Bar	44
5.3	Conclusions	45
6	Conclusions and Future Work	47
6.1	Conclusions	47
6.2	Future Work	48
6.2.1	Introduction	48
6.2.2	Database access password management	48
6.2.3	Add Daemon status in the status bar	48
6.2.4	Implementing TRS Dynamic Training Procedures	48
6.2.5	Automatic port configuration	48
	Bibliography	49
A	User Manual	51
A.1	Introduction	51
A.2	General description	51
A.2.1	PIT-STOP Working Context	51
A.2.2	System Architecture	52
A.2.3	User Roles and Permissions	52
A.3	Functionality	53
A.3.1	Accessing the Application	53
A.3.2	Event Management Module	53
A.3.2.1	Filtering events	53
A.3.2.2	Seeing the information of an event	54
A.3.2.3	Modifying an event	54
A.3.2.4	Adding an event	54
A.3.2.5	Deleting an event	54
A.3.2.6	Showing the logs related to an event/s	54
A.3.2.7	Showing the heat map	55
A.3.3	Exclusion Management Module	56
A.3.3.1	Filtering exclusions	56
A.3.3.2	Seeing the information of an exclusion	57
A.3.3.3	Modifying an exclusion	57
A.3.3.4	Adding an exclusion	57
A.3.3.5	Setting the geographical position of an exclusion	58
A.3.3.6	Deleting an exclusion	58
A.3.3.7	Showing the logs involving an exclusion	58

A.3.4	Application Log Management Module	59
A.3.4.1	Filtering the application log entries	59
A.3.4.2	Showing the events related to a log	59
A.3.4.3	Showing the exclusions related to a log	59
A.3.5	Alarm Handling Module	59
A.3.5.1	Seeing the information of an alarm	60
A.3.5.2	Silencing an alarm	60
A.3.5.3	ACKing an alarm	60
A.3.5.4	Reset an alarm	60
A.3.6	Configuration Module	60
A.3.7	Setting up the Database Daemon	63
A.3.8	Login out of the Application	63
A.3.9	Exiting the Application	64

List of Figures

- 2.1 PITSTOP Project. General Architecture. 5
- 2.2 Login window. 6
- 2.3 Exclusion viewer. 7
- 2.4 Event viewer. 8
- 2.5 Application Log viewer. 9
- 2.6 Alarm Handler. 10
- 2.7 User account viewer in configuration window. 11
- 2.8 User details dialog. 12
- 2.9 Directory configuration. 12
- 2.10 Network configuration. 13
- 2.11 FINDAS configuration. 13
- 2.12 PITSTOP Project. Example of hardware architecture. 15
- 2.13 Software architecture diagram. 16
- 2.14 Communication Module diagram. 17

- 3.1 Database relationship. 19

- 4.1 Software architecture diagram. 28
- 4.2 User Login Module diagram. 29
- 4.3 Event Management Module diagram. 30
- 4.4 Opening event dialog operation diagram. 31
- 4.5 Exclusion Management Module diagram. 32
- 4.6 Opening exclusion dialog operation diagram. 33
- 4.7 Alarm Handler Module diagram. 34
- 4.8 Application Log Management Module diagram. 35
- 4.9 Configuration Module diagram. 36
- 4.10 Opening user dialog operation diagram. 37
- 4.11 Database Daemon diagram. 38
- 4.12 Communication Module diagram. 40

A.1	PITSTOP Project. General Architecture.	51
A.2	Login dialog.	53
A.3	Event viewer.	54
A.4	Event modification dialog.	55
A.5	New event dialog.	55
A.6	Heat map example.	56
A.7	Exclusion viewer.	56
A.8	Exclusion modification dialog.	57
A.9	New exclusion dialog.	58
A.10	Application Log viewer.	59
A.11	Alarm viewer.	60
A.12	User account viewer in configuration window.	61
A.13	User details dialog.	61
A.14	Directory configuration.	62
A.15	Network configuration.	62
A.16	FINDAS configuration.	63
A.17	Database Daemon interface.	63

Chapter 1

Introduction

1.1 Motivation, Aims and Objectives

Nowadays, the natural gas transmission pipeline system is by far one of the safest ways to transport this kind of fuel through long distances and urban areas. In spite of the fact that several strong security measures are used in all natural gas pipelines, such as regularly leak inspections, foot patrols, helicopter flights and (un)announced activities, the risk of damage cannot be completely eliminated. This is the main motivation of this project, to add an extra measure to increase the security of the pipeline transmission.

It is important to consider that most incidents involving natural gas transmission are mainly occasioned because of an external agent. More specifically, as a relevant research reported, the 50% of the accidents [3] were caused due to third party works in the pipeline proximity with a significant difference from the second main cause (construction defect or material failure). These accidents resulted in the loss of human lives, environmental damage and high economic losses (due to supply disruptions).

The situation described above demands additional preventive measures with high performance and economically effective solutions for monitoring potential threats to the integrity of the pipelines. The main objective is to increase the detection of possible threats and lower the response time. Considering that all gas pipelines have fiber optics deployed along all the infrastructure, the vibration-based sensing technology that optic fiber supports to observe the activities near the pipeline has become a very interesting solution to recognize possible threats and carry out a preventive action. The Project for the early detection of Pipeline Integrity Threats using a SmarT Fiber-Optic Surveillance System (PITSTOP) [4] is the possible answer to the solution that is being pursued.

This thesis is an integral part of the whole PITSTOP project. More specifically, the role of this thesis is the development of an application that receives the information of the alarms generated by the PITSTOP system and gives the possibility of managing those alarms. In addition, this application also manages several configuration options as it will be explained in the next sections.

1.2 Overview of the Alert Management Application

This bachelor thesis offers the user an intuitive, sophisticated and versatile user interface, which can be used to:

- Access and manage the information related to all events generated by the Threat Recognition System (TRS).

- Handle alarms and report them.
- Create exclusion areas.
- Change the TRS and the Distributed Acoustic Sensing System by FOCUS (FINDAS) configuration remotely.
- Access and manage the application log reports of all users enrolled in the system.

To do so, a User Interface Application to handle the Threat Recognition System (TRS-UI) has been developed in Qt [1], a cross-platform application framework used for developing software that can be run on different software and hardware platforms, such as Linux or Windows. The coding and the design of the graphical user interface has been carried out with the Qt Creator [2]. This application is split in several modules, which provides features or jobs. Each module is made of certain objects or classes. The objects are managed by one special object called Core, which is responsible of the application flow. This object is capable of starting different modules, objects and making connections between them using the Qt Signals & Slots technology, in order to achieve a communication and data transport, such as data structures, between objects. The interface counts with Google Maps support in order to accurately show the events, alarms, exclusions, etc and Qt library [1] widgets, like tables, menus and windows.

The program has a database connection to store all data, and read it when needed. The database, whose design is described here, contains several tables and fields to store the necessary information for the application to work. This includes a list of user accounts, events, exclusions, application logs of recorded actions made by the users, configuration options and a geographical mapping between geographical coordinates and fiber optic positions that allows to translate them from one format to another.

It also employs TCP/IP socket communication to connect to the TRS and communicate to it with a robust custom protocol. The TRS-UI allows TRS configuration data to be requested by this same communication mean. Both TRS-UI and TRS applications have one master module and one slave module to communicate with the slave and master of the other system respectively. The master will send command and actions requests to the slave of the other system, such as parameter changes, events and configuration options. To control these connections, TRS-UI provides a status label in the interface to know in real time whether TRS, FINDAS and Database are connected at each time.

To sum up, TRS-UI offers a sophisticated solution to the integral management of some parts of the PITSTOP system in a high level way to the end user.

1.3 Thesis Outline

This thesis is structured in six chapters. This chapter briefly introduced the alert management application description, and the motivation why the application was created for.

The second chapter describes the system functionality, giving a high level view of the software and hardware architecture. It explains the application features and the variety of possible actions available for the users. Furthermore, the requirements of the system are enumerated in terms of hardware and software elements. Last but not least, a general introduction to the software design is given.

In the third chapter, the database design is presented in detail with the description of the tables and the fields that it consists of, explaining the functionality of each.

The fourth chapter is an extensive description of the software design strategy carried out in the application development and how the system is implemented. There is an exhaustive explanation of how the program works and the different modules are inter-connected.

The fifth chapter presents a set of evaluations performed and its results to demonstrate the software functionality and reliability.

In the sixth chapter, a conclusion of the thesis is given along with some suggested improvements for future work.

After the bibliography, this bachelor thesis concludes with two appendices for the user manual and the reference manual of the TRS-UI application, with a detailed description of how to work with the interface and how to understand and reuse the software code.

A detailed documentation of the software code in a low level approach has been provided in an external document generated by Doxygen [5].

Chapter 2

System Description

2.1 Introduction

The TRS User Interface (TRS-UI) is an integral part of the PITSTOP project. It provides an intuitive interface to control the configuration parameters of the TRS system and manage all relevant aspects of the PITSTOP system. It also carries out the database management tasks that have taken place in the TRS.

This chapter describes the system architecture and functionality to show what the TRS-UI is able to do and what are the requirements of the system. The application provides a set of error handlers that alert the user about an external error in the database.

2.2 General Description of the PIT-STOP Working Context

The PIT-STOP general architecture, describing a very high level of interactions, is shown in Figure A.1 [6]. We consider that, at least, two user roles will be required: one in charge of getting the alarm notifications and launching the actions to be taken (alarm controller), and the other in charge of managing the TRS core system and the FINDAS, including the handling of the signals associated to actual alarms for improving the system performance (system controller). The TRS system controller role could be further divided in two roles: managing the core system (programming), and managing actual alarms for system improvement.

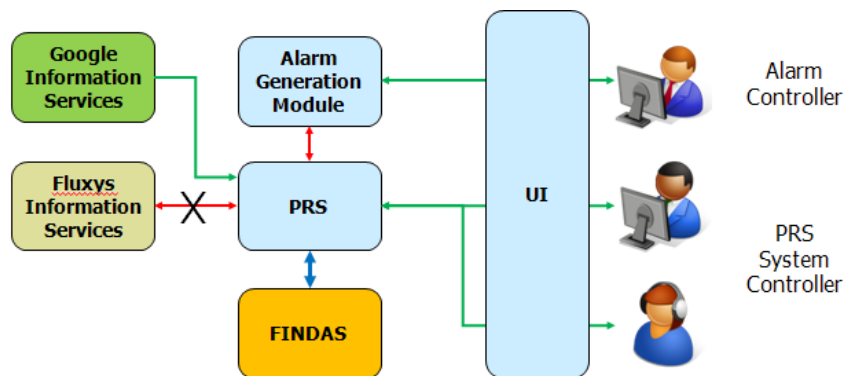


Figure 2.1: PITSTOP Project. General Arquitecture.

2.3 System Specification

2.3.1 Generalities

The user interface for the TRS application will allow (depending on the user role and permissions)[6]:

- Managing the user information.
- Managing the information in the exclusion table.
- Managing the information in the event table.
- Managing the information in the application log.
- Managing alarm events. This also includes communication problems or system malfunction.
- Visualizing the geographical information and details of any system activity (related to events and/or exclusions).
- Configuring the system (including, as required, the TRS and/or the FINDAS system).

2.3.2 Detailed Specification

The main functionality for each of the defined modules is described below [6]:

- Log in procedure: Will open a dialog box requesting the login name and password for the user willing to access to the system. After verification, the system will allow or deny the user access, showing an adequate information message if needed (e.g., with the reason for access denial). In the current version of the system, when the application starts it automatically invokes the user logging dialog box, as shown in the Figure 2.2.

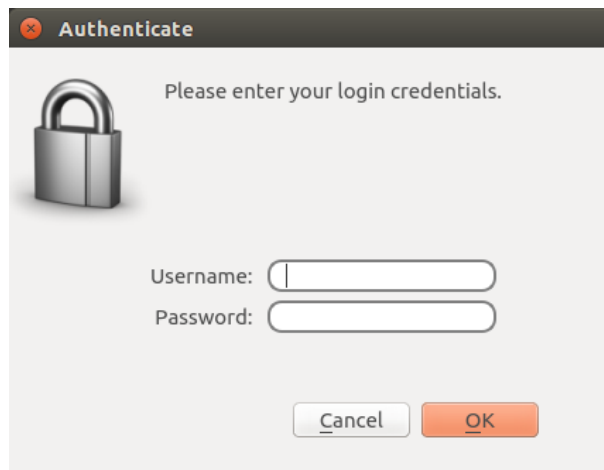


Figure 2.2: Login window.

- Exclusion management: Will activate the exclusion management functionality. It will display the default map (with a given center and zoom factor), the list of current and future exclusion entries, and a region to filter the list allowing to modify the initial and end dates of the list entries. The list includes the initial and end timestamp, the exclusion name, and an icon showing the exclusion state

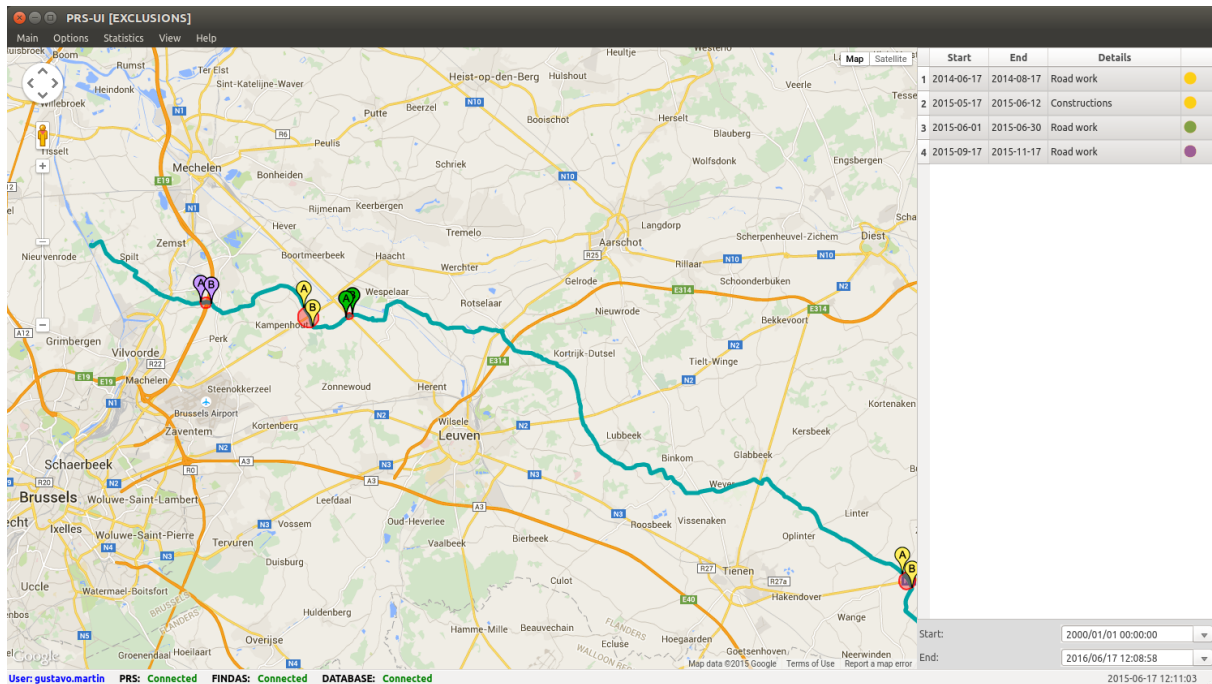


Figure 2.3: Exclusion viewer.

(green: currently active, purple: future exclusion, yellow: past exclusion). A sample screenshot from the current implementation is shown in the Figure 2.3.

The user will be able to:

- Place the cursor on an entry: This will open a pop-up box with all the information of the selected entry.
- Click in a list entry: This will center the map in the location corresponding to the clicked entry. Multiple selection is possible.
- Double click in a list entry: This will center the map in the location corresponding to the clicked entry, and open a dialog box with all the information, with the possibility of editing any field. If any modification is done, it will create a log entry in the application log table.
- Right click in a list entry: will pop up a menu allowing to add, delete and modify the selected entry:
 - Add: Will open a dialog box with all available information to be filled in by the user. The definition of the actual position of the exclusion will be done either by introducing the optic fiber position, the geographical coordinates, or by direct selection in the map display. It will create a log entry in the application log table. For the Add operation, it is not necessary to click in a certain list entry, but in some place of the list.
 - Delete: Will open a confirmation box before deleting the given exclusion.
 - Modify: Same action when double clicking an entry.
 - Modify the start and end dates of the listed exclusions, acting on the given fields on screen.
 - Quick select exclusion entries from current day, last week, last month, all, past ones, active ones and future ones.
- Event management: Will activate the event management functionality. It will display the default map (with a given center and zoom factor), the list of the latest event entries, and a region to

filter the list allowing to modify the initial and end dates of the list entries. The list includes the timestamp, the event name, and an icon showing the event state (green: possible threat, detected by the TRS as no threat, yellow: event detected as a low level threat, red: high level threat detected). A sample screenshot from the current implementation is shown in the Figure 2.4.

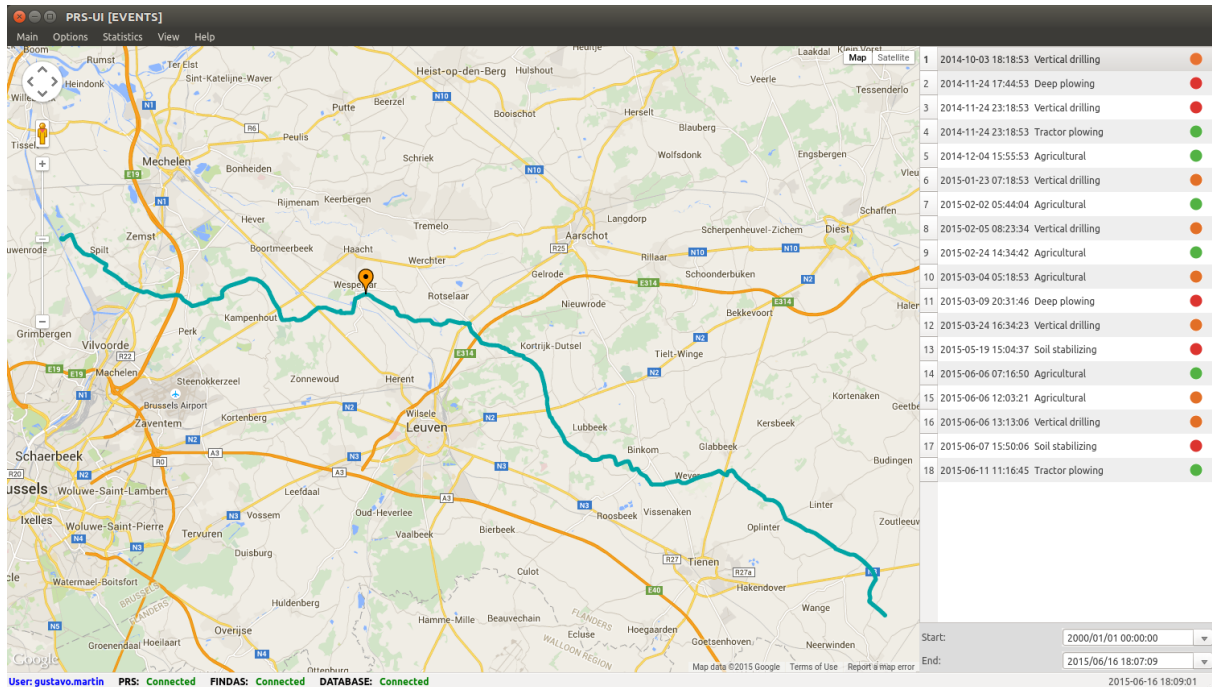


Figure 2.4: Event viewer.

The user will be able to:

- Place the cursor on an entry: This will open a pop-up box with all the information of the selected entry.
- Click in a list entry: This will center the map in the location corresponding to the clicked entry. Multiple selection is also possible.
- Right click in a list entry: will pop-up a menu allowing to add, delete and modify the selected entry. Only certain fields are able to modify by the user.
- Add: Will open a dialog box with all the available information to be filled in by the user. Will create a log entry in the application log table. For the Add operation, it is not necessary to click in a certain list entry, but in some place of the list.
- Delete: Will open a confirmation box.
- Modify: Same action that when double clicking an entry.
- Show logs: Will change to the application log view listing all entries related to the selected events.
- Modify the dates of the listed events.
- Quick select events from current day, last week, last month and all.
- There is a link between the event database and the log database, so one can access the log database from an event entry.
- Statistics: Can create a heat map in the map with the events in the table.

- Application log: Will activate the application log management functionality. It will display the list of the latest application log entries, and a region to filter the list allowing modifying the initial and end dates of the list entries. The list includes the timestamp, the activity name, the user responsible of that action and the activity details. A sample screenshot from the current implementation is shown in the Figure 2.5.

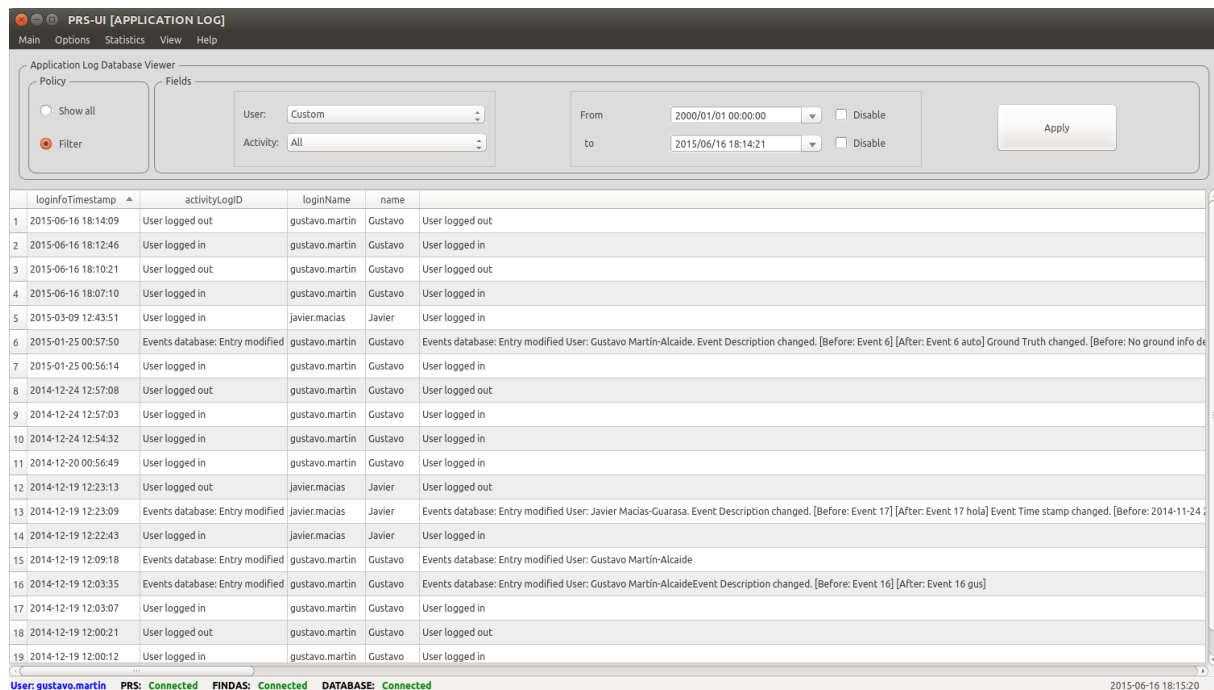


Figure 2.5: Application Log viewer.

The user will be able to:

- Filter the events to be shown by also selecting users and event types.
 - Quick select log entries from current day, last week, last month and all.
 - There is a link between the event database, exclusion database and the log database, so one can access the events or exclusions entries from a log entry referring to that event or exclusion.
- Alarm handling: Will activate the alarm handling interface. A sample screenshot from the current implementation is shown in the Figure 2.6. It will display the default map (with a given center and zoom factor), the list of the latest event entries, a region showing the alarm handling action and information items:
 - Two colored areas (led/square type, red and orange) that will blink when an alarm is raised. If an alarm is selected, the icon will be square type. Otherwise it will be a circle icon by default.
 - Place the cursor on an entry: This will open a pop-up box with all the information of the selected alarm.
 - Silence button: to silence the acoustic beeper. If pressed, the current timestamp will be recorded and beeper is silenced.
 - Ack button: to state the user acknowledgement of the alarm. If pressed, the current timestamp will be recorded.

- Reset button: to state that the user has proceeded with the required corrective action. If pressed, the current timestamp will be recorded. Pressing the ack button before pressing the reset button is required.
- ACK and Reset text report: to be filled in by the user.
- When an alarm is raised, this is a possible implementation for the procedure:
 1. An alarm is raised: there is a blinking colored area on the Graphical User Interface (GUI) and an acoustic beeper is started.
 2. The user may press the silence button to stop the acoustic beeper.
 3. The user may write the action taken in the ACK text report and click the acknowledge button to state that an action was taken.
 4. The user may press the reset button to state that a corrective action has been taken. This requires the Reset text report to be non-empty.

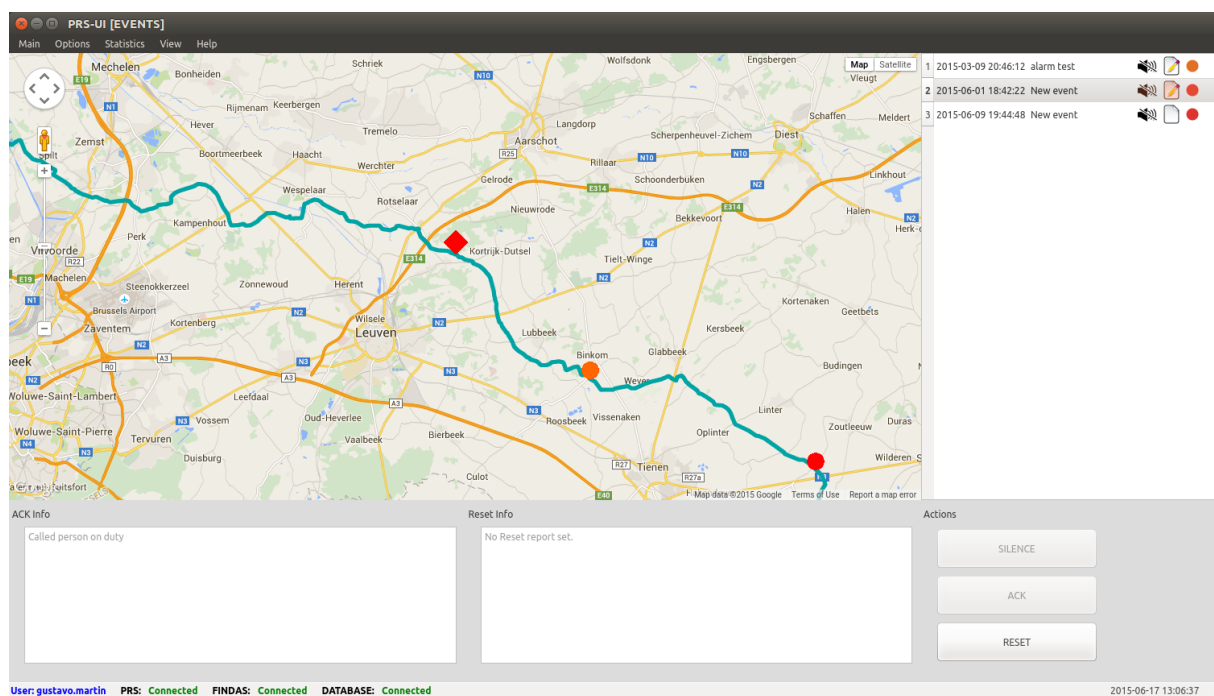


Figure 2.6: Alarm Handler.

- Log out: Will open a dialog box requesting confirmation for logging the user out. Will create a log entry in the application log table.
- Configuration: Will open the configuration dialog box, with the following tabs:
 - Users: Will show a list of the users in the database. A sample screenshot from the current implementation is shown in the Figure 2.7 and Figure 2.8. The user will be able to:
 - * Click an entry: This will select it.
 - * Double click in a list entry: This will open a dialog box with all the information, with the possibility of editing any field. If modification is done, it will create a log entry in the application log table.
 - * Add button: Will open a dialog box with all the available information to be filled in by the user. Will create a log entry in the application log table. For the Add operation, it is not necessary to click in a certain list entry, but in some place of the list.

- * Delete: Will open a confirmation box.
- * Modify: Same action that when double clicking an entry.
- Communication: Will show a list of the relevant communication parameters between the different subsystems (TRS, FINDAS and User Interface). A sample screenshot from the current implementation is shown in the Figure 2.10.
- Directories: Will show a list of relevant directories to be used. The user will be able to modify any of them. A sample screenshot from the current implementation is shown in the Figure 2.9.
- FINDAS: Will show a list of configuration variables related to the FINDAS box. The user will be able to modify any of them. A sample screenshot from the current implementation is shown in the Figure 2.11.

Additional details:

- All lists will be scrollable if needed.
- All the information with associated geographical references (events and exclusions), will be graphically represented in the main window of the application, using the Google Maps API. The pipe location will be superimposed to the main map.
- The application should rescale all the components after any change in the application window size (i.e., maximize, minimize, etc).

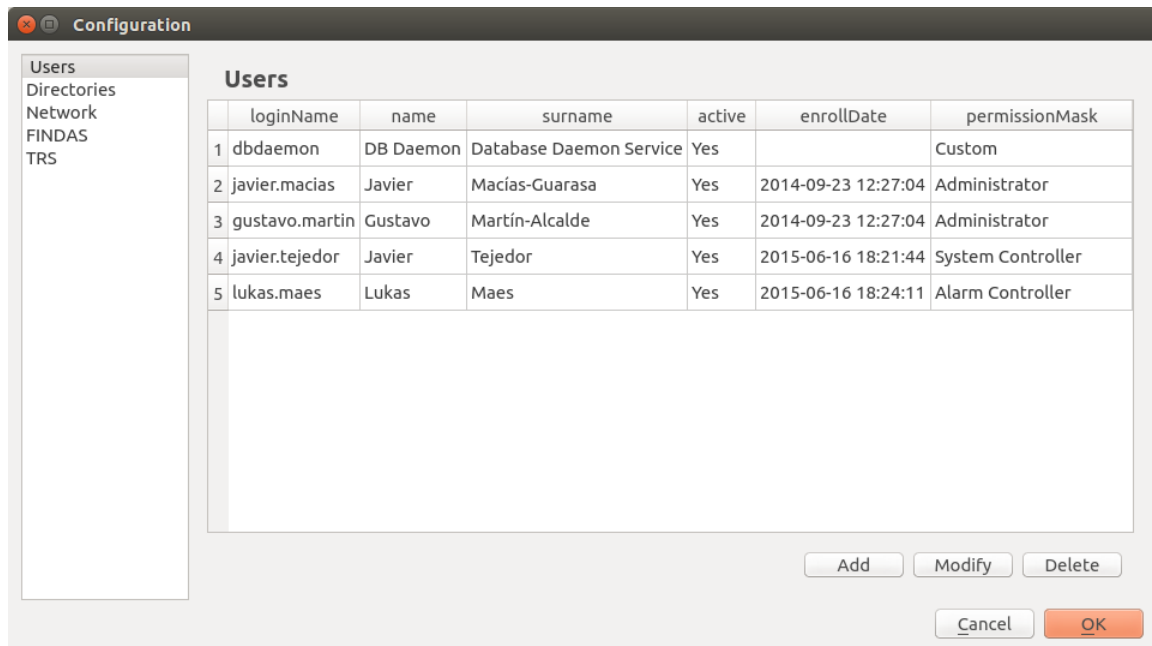


Figure 2.7: User account viewer in configuration window.

Modify user

Name: Lukas

Surname: Maes

Login name: lukas.maes

Status: Active

Password: (unchanged)

Confirm password:

Show password

Account type: Alarm controller

Permissions:

- Alarms management
- Events (View)
- Exclusions (View)
- Users (View)
- Directories
- Network
- FINDAS
- TRS

Cancel OK

Figure 2.8: User details dialog.

Configuration

Users
Directories
Network
FINDAS
TRS

Directories

Directory 1: /FINDAS/dir1 ...

Directory 2: /FINDAS/dir2 ...

Directory 3: /FINDAS/dir3 ...

Directory 4: /FINDAS/dir4 ...

Directory 5: /FINDAS/dir5 ...

Directory 6: /FINDAS/dir6 ...

Directory 7: /FINDAS/dir7 ...

Directory 8: /FINDAS/dir8 ...

Directory 9: /FINDAS/dir9 ...

Directory 10: /FINDAS/dir10 ...

Cancel OK

Figure 2.9: Directory configuration.

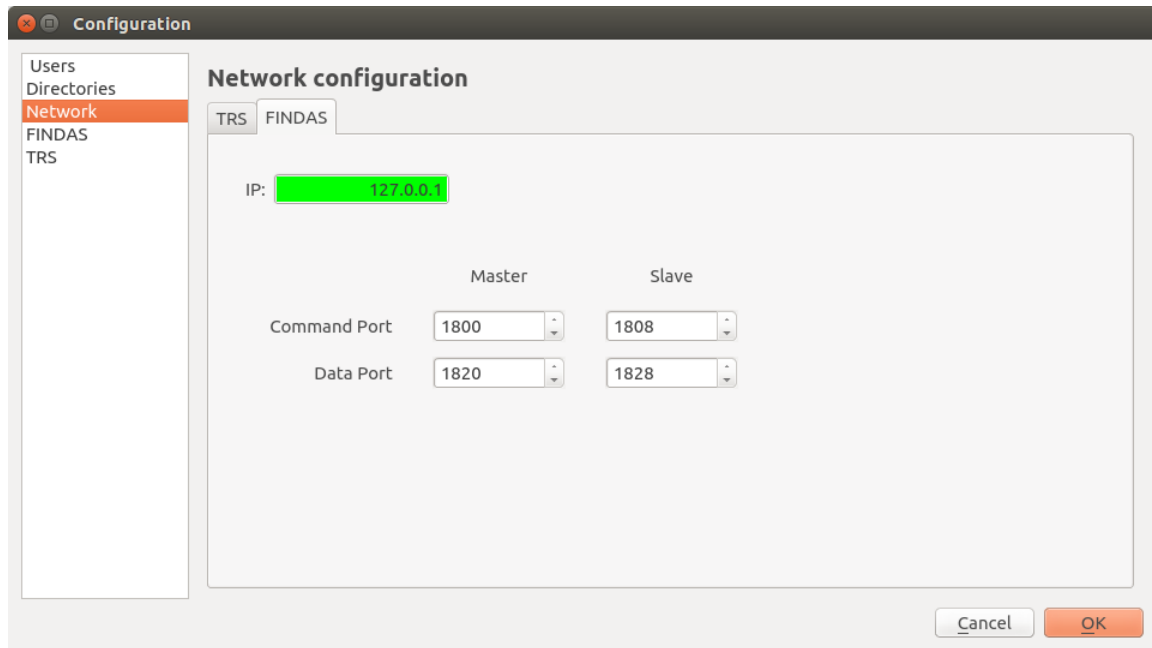


Figure 2.10: Network configuration.

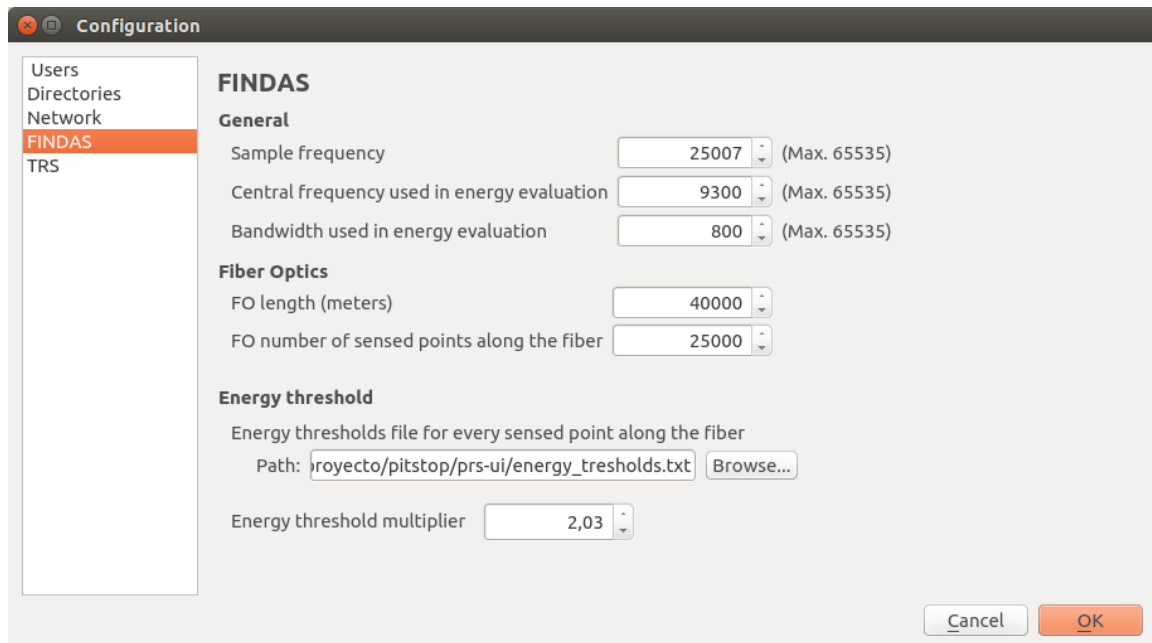


Figure 2.11: FINDAS configuration.

2.4 System Functionality

This software application consists of several modules, where the user can interact with the database and the TRS. To have access to the application, the user must init a session with a user account and the password. The TRS-UI allows multiple accounts with custom permissions, although they can be set automatically with account profiles, providing a reliable management of what the users can do.

One of the main task of the TRS-UI software is to make the TRS system communicate with the database. It must receive the new events from the TRS and insert them in the database. Due to likely concurrent problems, since there may be several TRS-UI running in different computers at the same time, there will be just one special user account responsible of recording the new incoming events into the database.

The TRS-UI provides a map from Google Maps API [7], in which events, alarms and exclusions can be displayed and several tables where these are listed. This makes quite easy to the user to know where the elements are placed and find them quickly due to the filter menu. Events and exclusions can be created, modified or deleted (if the user has permission to it).

Another important section in the program is the alarm handler interface that shows a map and a list with the new incoming events from the TRS. This section notices the user when a new alarm arrived by making an alarm sound, and allows to silence, acknowledge and reset it, by typing the corresponding report to the application of the actions that were taken in response to that.

Exclusions play an important role in the application. When an event occurs inside an exclusion, the event is automatically silenced and does not go to alarm section but to event section. In other words, the event is excluded and therefore it does not need to be handled by the user.

This application also has a configuration menu for the user accounts that have access to it, and for the network and FINDAS configuration, which allows to easily change FINDAS parameters. These FINDAS parameters will be automatically sent to the application from the TRS with a specific communication protocol.

All actions or changes done in the application are recorded in the application log to track whatever is made by users.

2.5 Hardware Architecture

The system architecture, as Figure 2.12 shows, is made by a series of TRS-UI running in different computers that are used by the corresponding users, and one TRS-UI running in Database Daemon mode, in addition to the TRS and FINDAS. All TRS-UI must be connected with the Database and the TRS at the same time.

The TRS will be connected with FINDAS as well. FINDAS is responsible of listening the optic fiber along the gas pipes and figure out if there is a potential threat. In that case, it will be sent to the TRS to recognize the actual threat type and propagated throughout all the TRS-UI currently connected. The Database Daemon will automatically insert the new alarms generated by the TRS in the database and therefore, does not need to be manipulated by a human.

It is also indispensable to have the TRS and FINDAS connected, although FINDAS is in the same system network, the communication and the configuration changes towards FINDAS from the UI is done through TRS, so FINDAS is transparent to TRS-UI in the communication matter. Normally, the Database Daemon will be running in the same computer as TRS to simplify the system architecture. Last but not

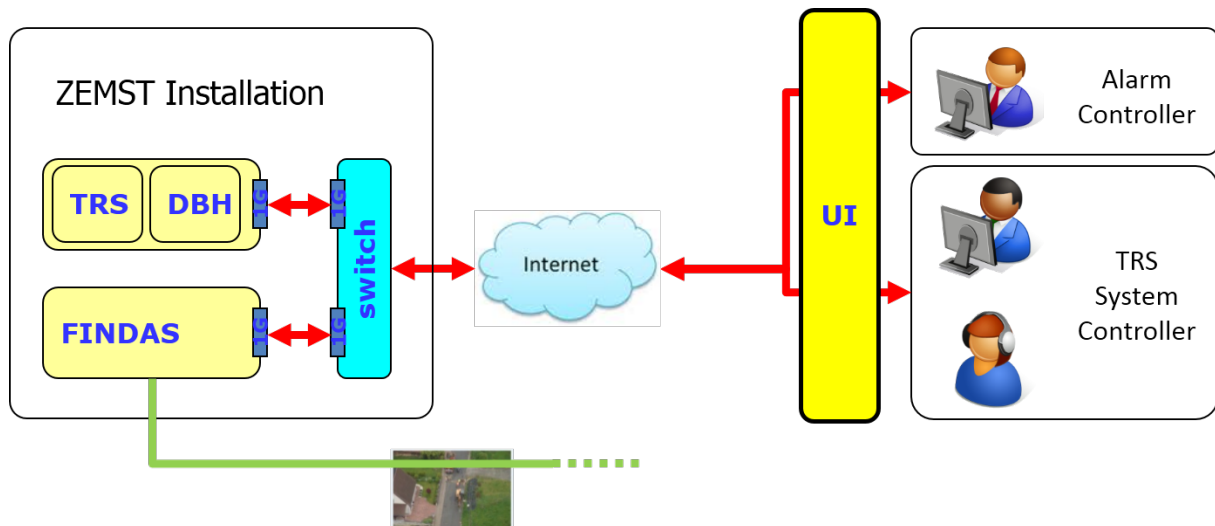


Figure 2.12: PITSTOP Project. Example of hardware architecture.

least, a Database is crucial in the system. The database is only accessed by the TRS–UI, and hence when TRS needs one or more configuration parameters, or just records a new event, the TRS–UI acts as an intermediary.

This architecture configuration leads to the need of having a reliable network which permits to set up all the connections that are necessary. The network connection must be configured in the TRS–UI, by setting the TRS IP, FINDAS IP and their ports, so that the UI applications connect automatically by reading the network configuration in the database.

2.6 Software Architecture

The software design of this application has been carried out designed by modules. A schematic diagram is shown in Figure 2.13. There are also two important files which contain the definitions of structures, lists, variables, etc. One of them is for communication definitions and it is shared with the TRS software that allows the communication between the UI and the TRS.

The other definition file contains declarations for the TRS–UI application. This software application can be split in several modules where different tasks are performed. These modules are composed of one or more objects. For instance, the event module is formed by the main window graphical interface object, which interacts with the user and receives the user actions, the event dialog window, which opens a window interface to allow the user to graphically modify or create an event, and the Core part. The last one, is the heart of the application, and it is where all the logic and program flow is carried out. In our example, the Core part dedicated to the event module would be responsible of getting the event parameters from the application, giving them to the graphical modules, handling the modification or creation of an event and so on.

The Core is the main part of this software and interacts with other objects to build a module of the application. The communication with the different objects is mainly made with signals and slots, taking advantage of one of the most important features of the Qt library. The interface design has been created with the Qt Creator [2], due to its simplicity, power and versatility.

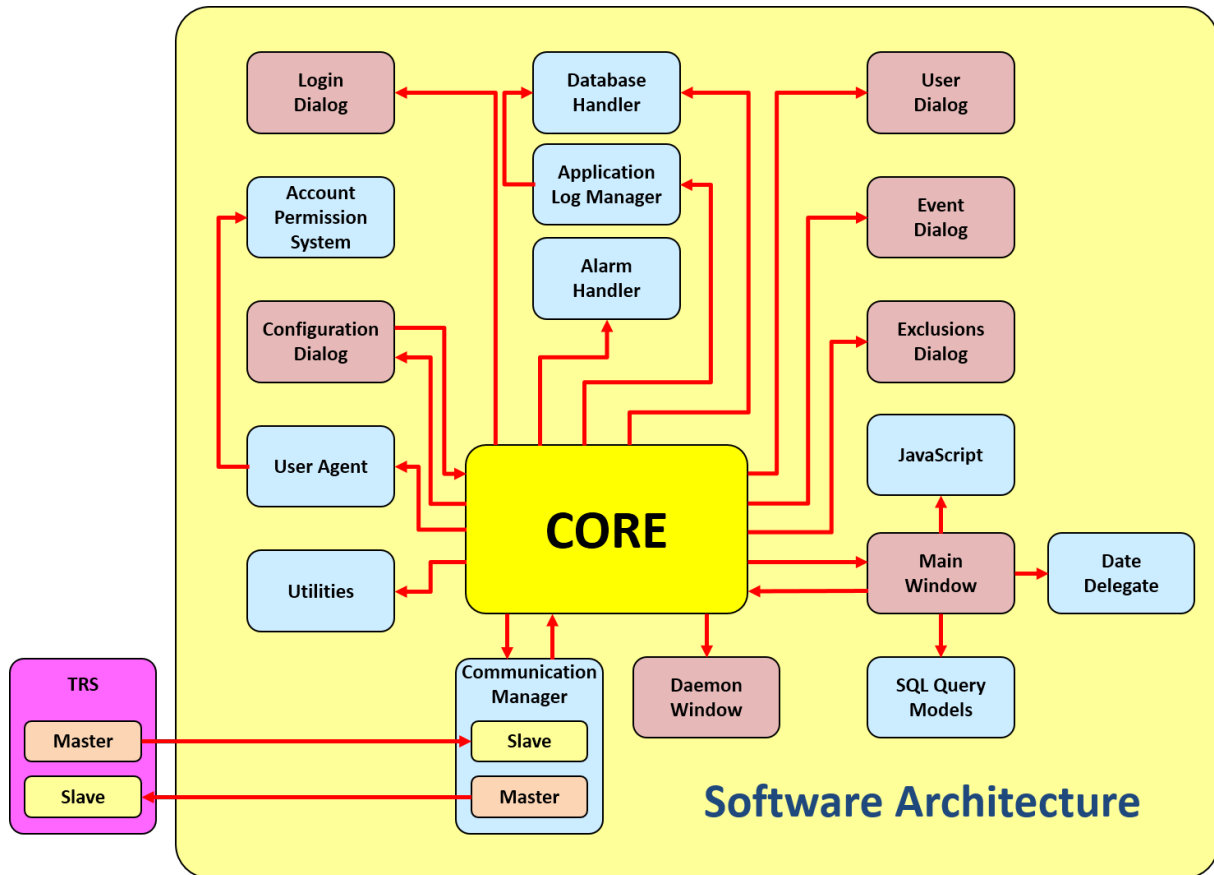


Figure 2.13: Software architecture diagram.

2.7 Communication Protocol

The communication with the TRS is performed with standard TCP/IP communication via sockets and the TRS communication protocol. Both in UI and TRS, there is a pair of Master and Slave modules, with a Communication and Data port each. The Figure 2.14 shows the connection of the Modules and ports. The communication protocol consists of two packet types, Commands and Parameters. When either TRS or TRS-UI wants to carry out an action on the other, a command packet is sent, with the corresponding command ID. After, the receiver will send back an ACK Command. If the configuration parameter are requested by the command, a parameters packet will be sent after the ACK.

To execute the application it is necessary a computer with Linux operative system, although the software can be easily ported to Windows thanks to Qt library. The computer must have the Qt drivers installed, including the Qt MySQL drivers and a working network connection.

2.8 Conclusions

This chapter has covered the hardware and software system architecture, to give a overall idea of how this software has been performed in a technical way. It introduces the context that this software application is in, regarding the additional systems or parts this application needs to work with, and briefly explains how the developing strategy, based on modules, has been carried out to build the TRS-UI software.

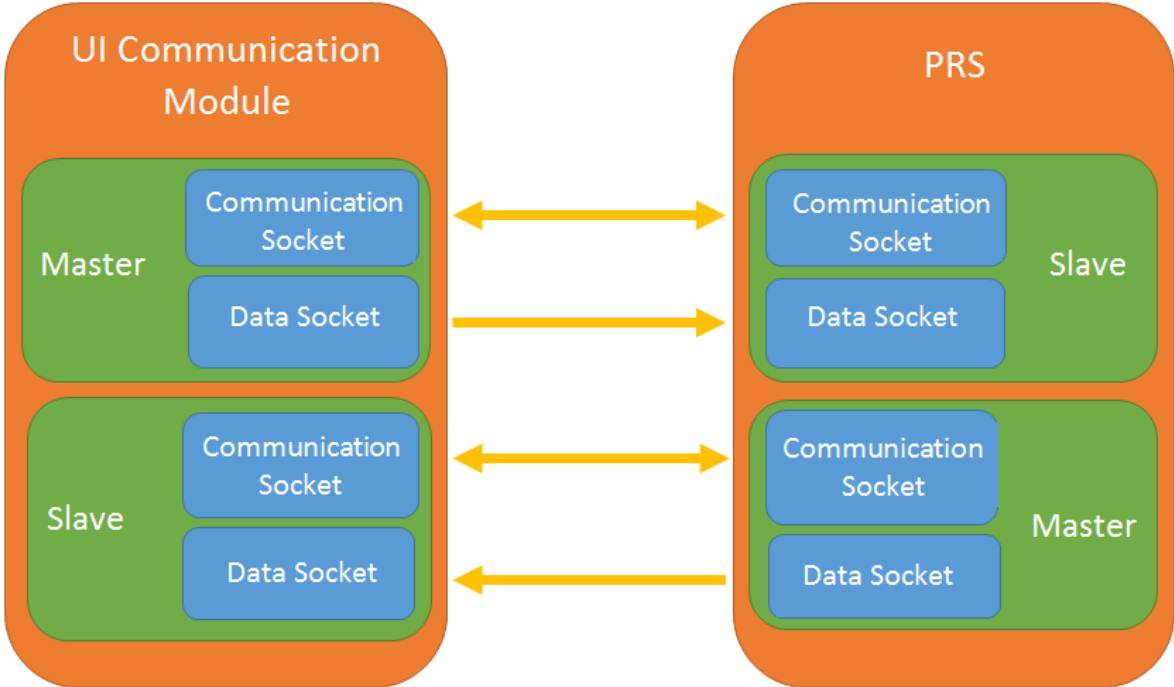


Figure 2.14: Communication Module diagram.

Chapter 3

Database Design

3.1 Introduction

This chapter describes the Database structure, its tables, fields and the relationship between them. The design responded to the structure and needs of the application because the database is needed to store all the relevant information. The database design contains important data for the TRS and FINDAS, such as the configuration parameters, and for the whole system, which includes the user accounts list, events, exclusions, threats, application logs and a geographical mapping. Some fields tables are linked to others, which makes a relationship that is shown in the Figure 3.1 below.

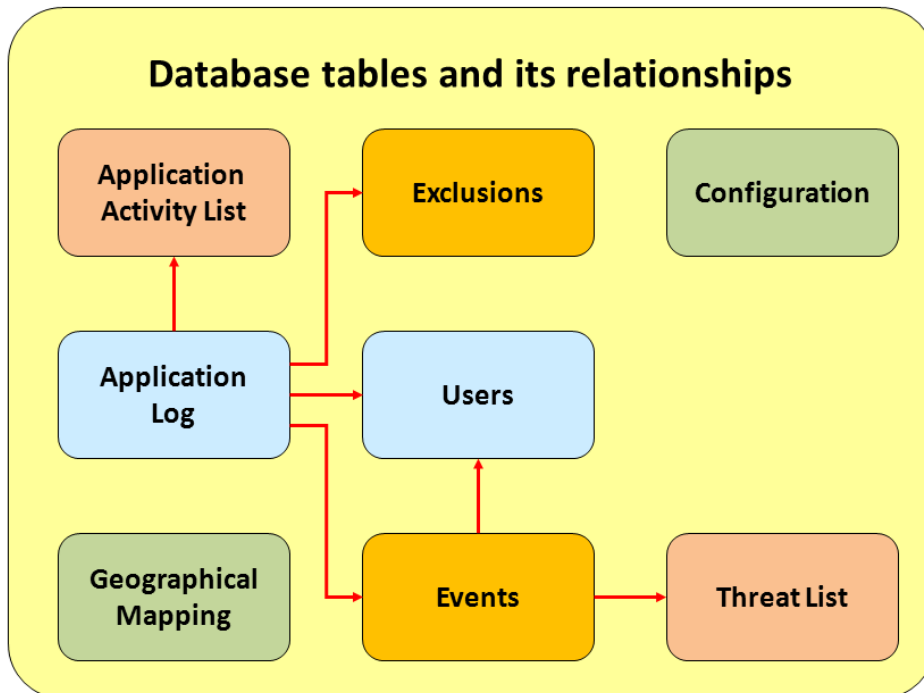


Figure 3.1: Database relationship.

3.2 Database Structure

3.2.1 Users

This table contains a list of user accounts enrolled to the TRS User interface management, with the details belonging to themselves that are defined as follows:

- `userID`: User Identification number. It is unique and cannot be changed. ID number equal to -1 is reserved for Daemon user.
- `loginName`: User login name. Used to log into the application.
- `name`: User first name.
- `surname`: User surname.
- `password`: Encrypted password with SHA-1 algorithm.
- `active`: User status. The program requires the user to be active (set to 1) to be allowed to enter in the application.
- `currentlyLogged`: Only used for Daemon user. Keeps record if Daemon is currently up and working.
- `enrollDate`: Sign up date of the user account.
- `permissionMask`: Permission mask. Integer number whose bits correspond to a certain permission in the application. The permissions that the application has are the following:
 - Alarm: Alarm viewer access.
 - Events: Event viewer access.
 - Events Add: Able to add a new event.
 - Events Modify: Able to modify events.
 - Events Delete: Able to delete events.
 - Exclusions: Exclusion viewer access.
 - Exclusions Add: Able to add a new exclusion.
 - Exclusions Modify: Able to modify exclusions.
 - Exclusions Delete: Able to delete exclusions.
 - Users: User list access.
 - Users Add: Able to add a new user account.
 - Users Modify: Able to modify a user account.
 - Users Delete: Able to delete a user account.
 - Directories: Able to view and modify directories.
 - Network: Able to view and modify network configuration.
 - FINDAS: Able to view and modify FINDAS configuration.
 - TRS: Able to view and modify TRS configuration.
 - Application Log: Application log viewer access.

3.2.2 Events

This table contains all the events that have been analyzed by the system. An event is associated to a location for which an acoustic trace has been received from the FINDAS system, so that the TRS system can decide if this is an actual threat. The main parameters to store are:

- `eventID`: Event Identification number. It is unique and cannot be changed.
- `eventDescription`: Brief description of the event (maximum 128 characters).
- `eventTimestamp`: Timestamp of when the event arises.
- `FOposition`: FO (Fiber Optics) position in meters where the event took place.
- `coordinates`: Geographical coordinates of the event (latitude and longitude).
- `acousticTracePointer`: Pointer to the stored acoustic trace file recorded for the event.
- `thresholdUsed`: Threshold value used for the given geographical location as a reference value that decides if an alarm is sound or not.
- `classificationInfo`: Classification information generated by the TRS (for each label considered, the probability/likelihood value, and possibly the associated confidence value).
- `threatID`: Threat Identification number associated with the event. This field links with the corresponding entry in the Threat List table, described in section 3.2.4.
- `silenceTimestamp`: Records the timestamp when the silence button was pressed for the event.
- `ackTimestamp`: Records the timestamp when the ACK button was pressed for the event.
- `resetTimestamp`: Records the timestamp when the Reset button was pressed for the event.
- `groundTruthInfo`: Indicates the true values for each of the labels considered (these are set by the TRS System Controller in charge of evaluating the events).
- `ackInfo`: ACK report entered by the user when an event is acknowledged.
- `resetInfo`: Reset report entered by the user when an event is reset.
- `Silence_userID`: User ID that has silenced the event. This field links with the corresponding entry in the Users table, described in section A.12.
- `ACK_userID`: User ID that has acknowledged the event. This field links with the corresponding entry in the Users table, described in section A.12.
- `Reset_userID`: User ID that has reset the event. This field links with the corresponding entry in the Users table, described in section A.12.

3.2.3 Exclusions

It contains all the details related to pipeline sections that should be excluded from inspection by the TRS system (such as permitted works), thus events arised on an exclusion will not fire an alarm. The main data fields to store are:

- `exclusionID`: Exclusion identification number. It is unique and cannot be changed.

- `startDateTime`: Start date and time of the exclusion.
- `endDateTime`: End date and time of the exclusion.
- `initialDailyTime`: Initial time when the exclusion activates every day.
- `endDailyTime`: End time when the exclusion is not active until the next working period.
- `locationContactInfo`: Contact information of the exclusion location.
- `startGeographicalPosition`: Start geographical position point.
- `endGeographicalPosition`: End geographical position point.
- `startFOPosition`: Start Fiber optics position point.
- `endFOPosition`: End Fiber optics position point.
- `exclusionDetails`: Exclusion additional details.

3.2.4 Threat List

Contains every threat that have been identified and recognized by the TRS system with the necessary information regarding to it. The main data fields to store are:

- `threatID`: Threat identification number.
- `raisedAlarmCondition`: Alarm type which differentiates the threat level. There are only three types of alarm: Red, orange and green.
- `threatDescription`: Brief description of the threat.
- `additionalDetails`: Additional details of the threat.

3.2.5 Geographical Mapping

Contains the geographical mapping between the Fiber optics position (in meters) and the corresponding latitude and longitude for that position. In other words, each entry is a fiber optics point given in two ways, the meter of the fiber and the coordinates. The stored data fields are:

- `FOposition`: Fiber optics position in meters.
- `Latitude`: Latitude of the point.
- `Longitude`: Longitude of the point.

3.2.6 Application Activity List

List of user activities that are recorded in the application log. The main data fields to store are:

- `activityLogID`: Activity identification number.
- `activityDescription`: Activity description.

The list of available activities is the following:

- User logged in
- User logged out
- Events database: Entry added
- Events database: Entry modified
- Events database: Entry deleted
- Exclusions database: Entry added
- Exclusions database: Entry modified
- Exclusions database: Entry deleted
- Alarm database: Entry added
- Alarm database: Entry modified
- Alarm database: Entry deleted
- Alarm database: Entry Silenced
- Alarm database: Entry acknowledged
- Alarm database: Entry reset
- Users database: Entry added
- Users database: Entry modified
- Users database: Entry deleted
- Configuration database: Entry changed

3.2.7 Application Log

Contains all the activity records that happened in the system. This table is filled up automatically by the program when the users make actions that are actual application activities. The main data fields to store are:

- `logID`: Log entry identification number.
- `loginfoTimestamp`: Timestamp when the log was recorded.
- `activityLogID`: Activity log ID associated to the log. This field links with the corresponding entry in the Application Activity List table, described in section 3.2.6.
- `userID`: User ID responsible for the action to be recorded in the log. This field links with the corresponding entry in the User table, described in section A.12.
- `activityDetails`: Additional details and explanations of the activity. Generated automatically by the application.
- `eventID`: Event ID associated with the log (if it is not related to an event, set to -1). This field links with the corresponding entry in the Event table, described in section 3.2.2.
- `exclusionID`: Exclusion ID associated with the log (if it is not related to an exclusion, set to -1). This field links with the corresponding entry in the Exclusion table, described in section A.7.

3.2.8 Configuration

It contains the list of configuration options for the TRS and FINDAS, including technical parameters, directories and network configuration parameters like IP's and communication ports.

- `confID`: Configuration entry identification number. There is only one configuration entry, but this will be used just in case a future version of the application supports more than one configuration entries at the same time.
- `ipFINDAS`: Current FINDAS IP address.
- `FINDASslavePortC`: FINDAS slave module communication port.
- `FINDASmasterPortC`: FINDAS master module communication port.
- `FINDASslavePortD`: FINDAS slave module data port.
- `FINDASmasterPortD`: FINDAS master module data port.
- `ipPRS`: Current TRS IP address.
- `PRSSlaveToUIportC`: TRS slave communication port used with the UI.
- `PRSMasterToUIportC`: TRS master communication port used with the UI.
- `PRSSlaveToUIportD`: TRS slave data port used with the UI.
- `PRSMasterToUIportD`: TRS master data port used with the UI.
- `directory1`: Directory 1.
- `directory2`: Directory 2.
- `directory3`: Directory 3.
- `directory4`: Directory 4.
- `directory5`: Directory 5.
- `directory6`: Directory 6.
- `directory7`: Directory 7.
- `directory8`: Directory 8.
- `directory9`: Directory 9.
- `directory10`: Directory 10.
- `FINDASfs`: Frequency sample used by FINDAS.
- `FINDASfoLengthMeters`: Fiber optics length in meters used by FINDAS.
- `FINDASfoNumberOfPoints`: Points of the Fiber Optics.
- `FINDAScentralFreqEnergyHz`: Central Frequency (Hz) used by FINDAS.
- `FINDASbandwidthEnergyHz`: Bandwidth (Hz) used by FINDAS.
- `FINDASenergyThreshold`: Energy thresholds used by FINDAS.
- `FINDASenergyThresholdPATH`: Path to the file containing the energy thresholds.
- `FINDASenergyThresholdMultiplier`: Threshold multiplier used by FINDAS.

3.3 Conclusions

This chapter described the database tables and their respective fields to provide to the reader a necessary understanding of this crucial part of the application. The database of the PITSTOP project stores all events, exclusions, configuration options and logs. This leads the database to be a crucial part of the project and its understanding and knowledge is very important to understand how the system and therefore the TRS-UI application works.

Chapter 4

Implementation

4.1 Introduction

The goal of this chapter is to understand how the application software has been carried out and to explain the development strategy followed to construct it.

The implementation of this application software is broken down in several modules which are analysed in detail next.

4.2 Main Modules

4.2.1 Core Module

The `Core` module is the heart of the application. Every module revolves around it. It is responsible for the application flow, talking with the rest of the modules and connecting them to perform the operations. It also carries out important operations such as updating and inserting entries in the database. The details of the different roles in the overall application are specified in every application module. The schematic of the different parts of the software governed by the `Core` is shown in the Figure 4.1.

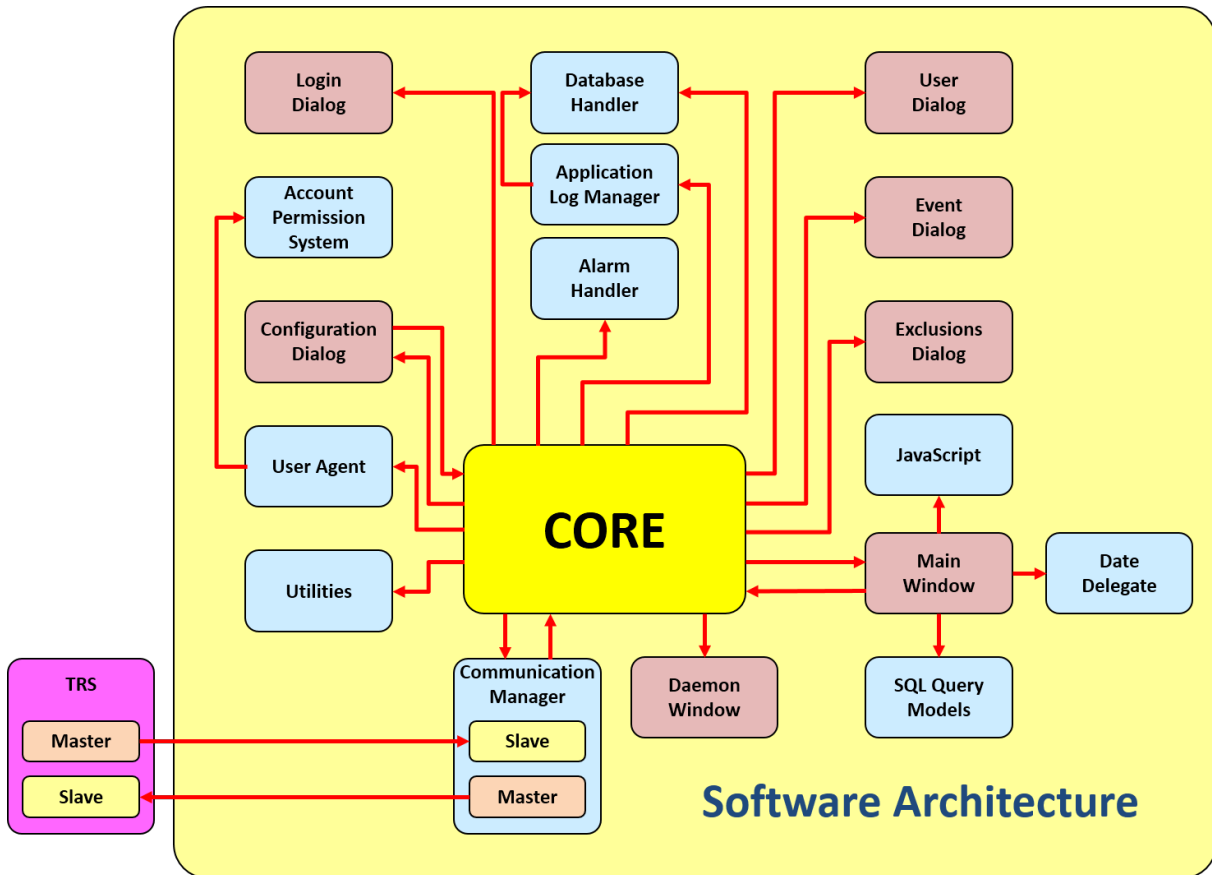


Figure 4.1: Software architecture diagram.

4.2.2 User Login Module

This module is compound of the Core, the Login_Dialog and other additional parts, as shown in Figure 4.2. It starts the login procedure to permit the end user starts a session in the application with a given user name and password. This window has been designed with Qt Creator, and provides a simple interface to the user. It has two input text boxes for the user name and password and a pair of buttons to accept or reject the procedure.

When the application starts, the Core creates an object of Login_Dialog with a User_Struct pointer passed by argument and it shows the Login_Dialog window and saves a copy of the pointer. Just after the object is created, the Core makes the appropriate signal and slots connections. When the user presses the button Accept, a signal emitted from the Login_Dialog object is sent to the core to execute the corresponding function to handle the operation (i.e check the table of users if the login name and its password match to some table entry and the user attempting to login is currently active and allowed to get into the application).

If the login succeeds, a signal is emitted back to the login dialog object from the application Core to close the login window. On the other hand, if the login fails for any reason, a signal is emitted back as well to the login dialog object, showing an information message.

If the user presses the cancel button, the corresponding memory is deallocated and the login dialog and the application close.

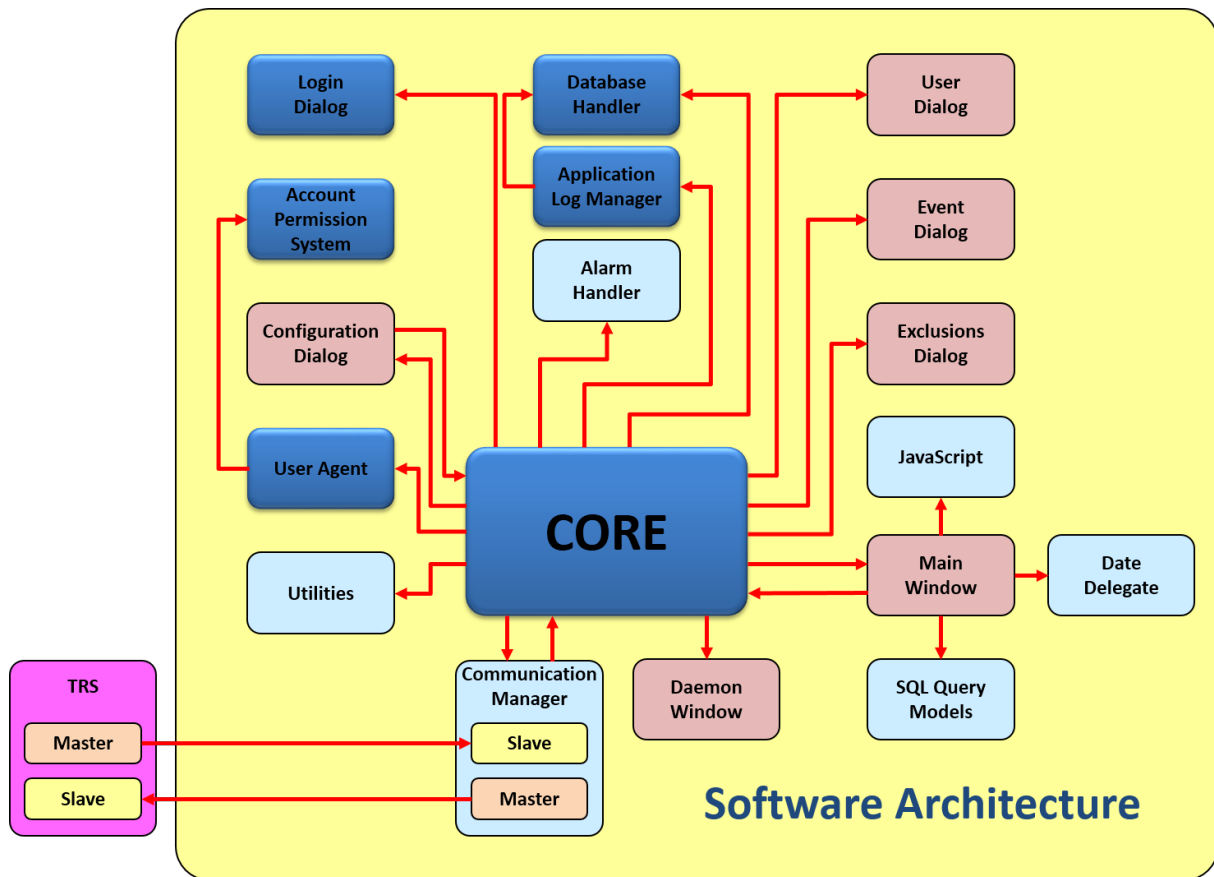


Figure 4.2: User Login Module diagram.

4.2.3 Event Management Module

The Event Management Module is composed of the Core, the Main window interface, the Event Dialog interface, the Application Log Module and some other parts, which correspond to object classes, as shown in Figure 4.3. They are inter-connected with signals and slots, providing the user to be able to view the event list, to filter events, the location in the map, the event parameters, to modify or create an event and to insert log entries related to event actions.

The MainWindow class is responsible for setting up the Event module part (and the other modules as well). The event management module, as the other module interfaces too, has been created and designed with the QtCreator tool. This tool provides a simple way to create interfaces without code and therefore to save time.

The event table, where the events are listed, is fulfilled taking them from the database and putting them in a SQL Query Model, which is attached to the event table and inherited from QSqlQueryModel. This keeps the information of the event in the model. The function to draw in the table of the model has been overridden to customize it, so it is possible to represent the event timestamp with a specific format and to indicate the type of the threat with color icons. In other words, the custom SQL Query Model for the event table stores the events and draws the items in the table properly.

In the case of the icons, it checks the threat ID associated with the event, gets the alarm type and shows an icon with the corresponding color. The tool-tip that appears when the mouse is over an event entry is created with the custom SQL Query Model and gets all the event parameters by calling a helping function from the Utilities class.

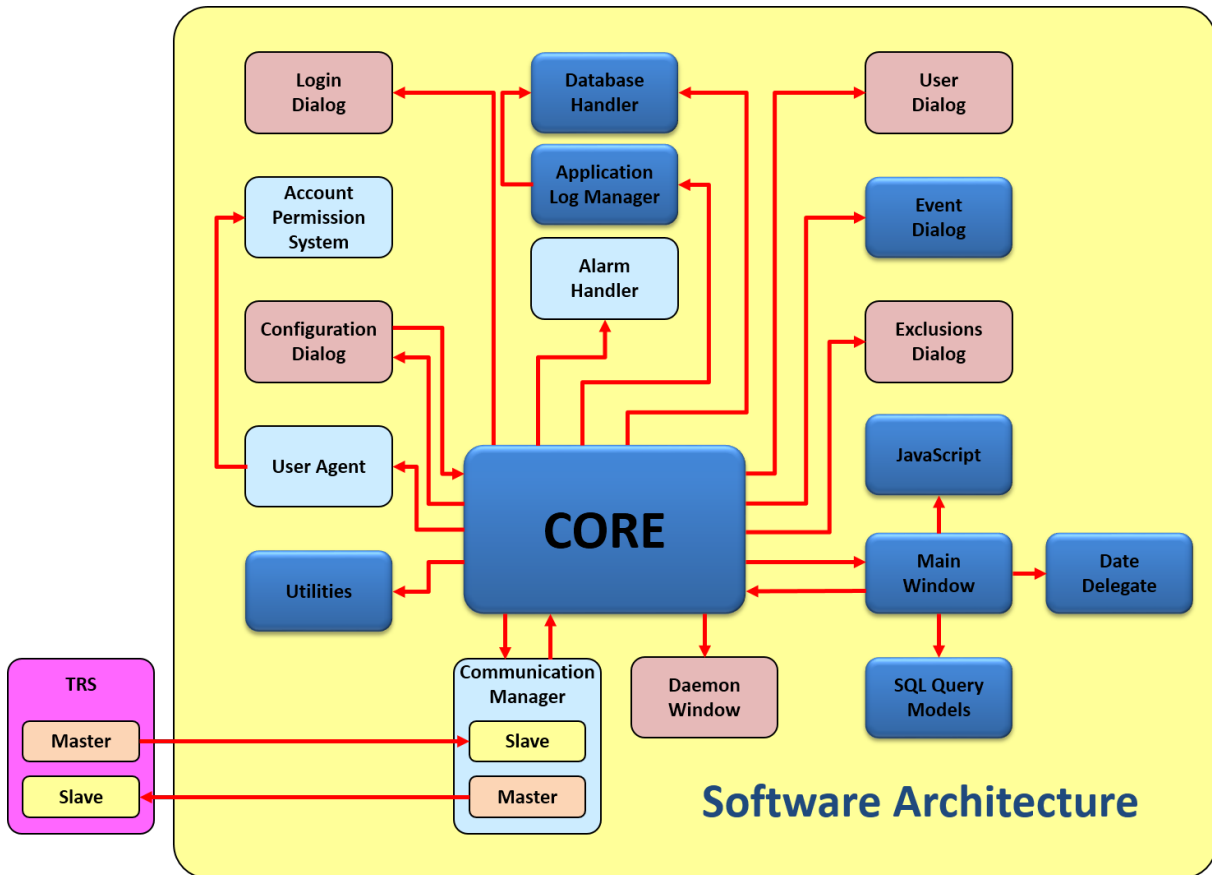


Figure 4.3: Event Management Module diagram.

The filter, either the one below the event table or the one placed in the top menu bar, calls a refresh function and makes a query with the given filter option to put the events in the event table. The pop-up menu, which appears when right clicking on an event entry, is made by customizing the mouse press event in the table object inherited from `QTableView`. This allows to control the pop-up buttons to enabling and disabling when it is necessary. To give functionality to these buttons, the button press signals have been connected with `Core` methods that do the corresponding work the button needs.

The map is built with a `QWebView` object of the Qt library, loading an HTML file with the Google Map API web. The pipeline is drawn with a KML layer library that parses the KML file. The html file also contains several JavaScript functions to add the markers in the position of the events when these are selected by the user in the table. When one or more events are clicked or selected, the program calls the JavaScript function to show all the markers. The location and the text information are passed through the arguments, so the marker is drawn in the map in the position of the event, storing the event information. Therefore, when it is clicked it shows a small box with the event parameters. The heat map feature available in statistic options, is made by calling another JavaScript function from C++ code when the button is pressed in the interface. This function does a loop through all current events in the event table and injects all the coordinates of the events in a JavaScript array. After that, the heat map constructor from the Google Maps API is called to set it up.

This module is also responsible for providing an interface in which the user can create a new event or modify an old one. The dialog box that this module can create with the `Event_Dialog` object is a window with all event parameters and text boxes. When the `Core` creates an `Event_Dialog` object, an event structure and the dialog mode are required to pass through the arguments of the constructor. First of all, this checks the opening mode. If it is ADD mode, this module will open the event dialog

window with all parameters empty, which must be filled in by the user. When the user accepts, all event information is written in the event structure given by the Core, and next, the Core will insert it properly in the database. On the other hand, if the user is modifying an event, the procedure would be the same, apart from the fact that the Core sends the Event struct filled with the current event data and the Event Management Module shows it in the parameter fields, allowing the user to see the actual event data. This process is explained in the diagram shown by the Figure 4.4.

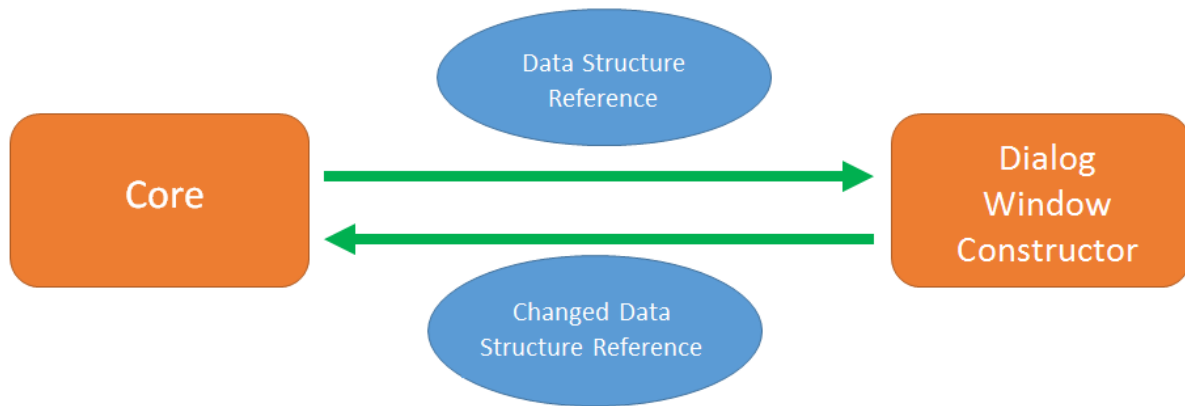


Figure 4.4: Opening event dialog operation diagram.

On the other hand, if the user cancels the modification or the creation of an event, nothing is changed nor added in the event structure and therefore, when its returned back to the Core, no modifications are carried out in the event database and no logs are recorded.

The event management module has access to the database and the Application log module through the Core. This allows to insert the corresponding log entry in the application log when is needed.

4.2.4 Exclusions Management Module

The Exclusion Management Module is composed of the Core, the MainWindow interface, the Event Dialog interface and some other parts, which correspond to object classes, as shown in Figure 4.5. They are inter-connected with signals and slots, although the Core part is the one which controls the flow logic. This module allows the user to view the exclusion list, to filter exclusions, place them on the map, modify and create them. In the same way, new log entries are added to the database when an important action is taken.

The MainWindow class is responsible for setting up the Exclusion Management Module part (and the other modules as well), which involves the exclusion table, map and filter. The Exclusion Management Module, as the other interface modules, has been created and designed with the QtCreator tool. This tool provides a simple way to create interfaces without code and therefore, to save time.

The exclusion table, where exclusions are listed, is fulfilled taking them from the database and putting them in a SQL Query Model, attached to the exclusion table and inherited from QSqlQueryModel, which keeps the information of the exclusion in the model. The function to draw in the table of the model has been overridden to customize it, so it is possible to represent the exclusion timestamps with a specific format and to indicate the type of the exclusion with color icons (past, currently active or future exclusion). In other words, the custom SQL Query Model for the exclusion table stores the exclusions and draws the items in the table properly. In the case of the icons, it checks the timestamp associated

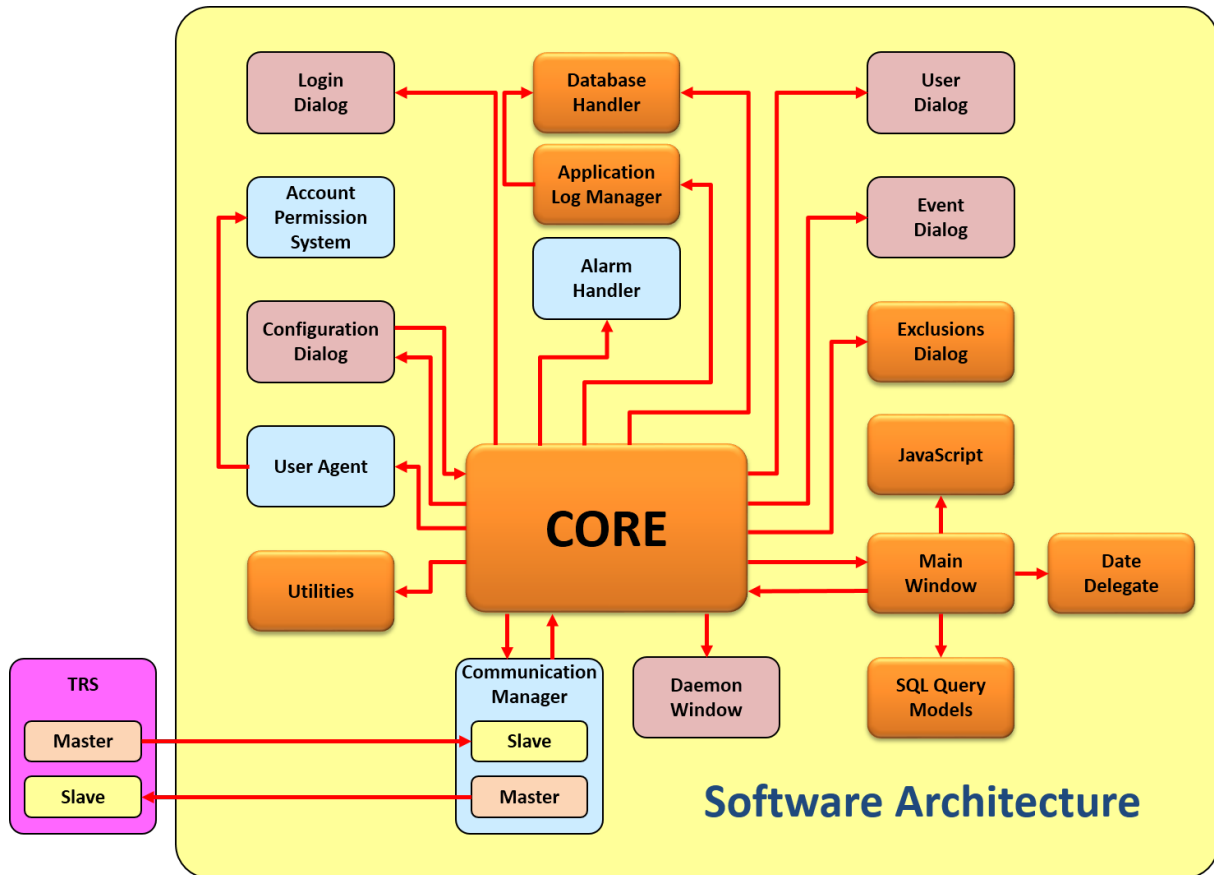


Figure 4.5: Exclusion Management Module diagram.

with the exclusion, gets the exclusion type (according with the current date and time) and shows a icon with the corresponding color. The tool-tip that appears when the mouse is over an exclusion entry is created with the custom SQL Query Model and gets all the exclusion parameters by calling a helper function from the `Utilities` class.

The filter, either the one below the event table or the one placed in the top menu bar, calls a refresh function, makes a query with the given filter option and puts the exclusion in the event table. The pop-up menu, which appears when right clicking on an event entry, is made by customizing the mouse press exclusion in the table object inherited from `QTableView`. This allows to control the pop-up menu buttons by enabling and disabling them when it is necessary. To give functionality to these buttons, the button press signals have been connected with `Core` methods that do the corresponding work the button needs.

The exclusion map is built with a `QWebView` object of the Qt library, loading a HTML file with the Google Map API web. The pipeline is drawn with a KML layer library that parses the KML file. The HTML file also contains several JavaScript functions to add the markers in the position of the exclusions when these are selected by the user in the table. When one or more exclusions are clicked or selected, the program calls the JavaScript function to show all the markers. The location and the text information is passed in the arguments, so the marker is drawn in the map in the position of the exclusion and stores the exclusion information. Therefore, when this is clicked it shows a small box with the exclusion parameters. Apart from the two marker points (i.e., start and end of the exclusion), the exclusion area is also painted, so the user can know exactly which area is excluded. This is done with a JavaScript function called when placing an exclusion on the map, which calculates a circle between the two points, and draws it with the Google Map API.

This module is also responsible for providing an interface in which the user can create a new exclusion or modify an old one. The dialog box that this module creates with the `Exclusion_Dialog` object is a window with all the exclusion parameters and text boxes. When the `Core` creates a `Exclusion_Dialog` object, an exclusion structure and the dialog mode are required to pass through the arguments to the constructor.

First of all, it checks the opening mode. If it is Add mode, this module will open the exclusion dialog window with all parameters empty to be filled in by the user. When the user accepts, all event information is written in the exclusion structure given by the `Core` and next, the `Core` will insert it properly in the database. On the other hand, if the user is modifying an exclusion, the procedure would be the same, with the only difference being that the `Core` gives the `Exclusion` struct filled with the current event data and the `Exclusion` module shows it in the parameter fields, thus the user can see the actual exclusion data. On the other hand, if the user cancels the modification or the creation of an event, nothing is changed nor added in the event structure and therefore, when it is returned back to the `Core`, no modifications are carried out in the event database and no logs are recorded. This process is explained in the diagram shown by the Figure 4.6.

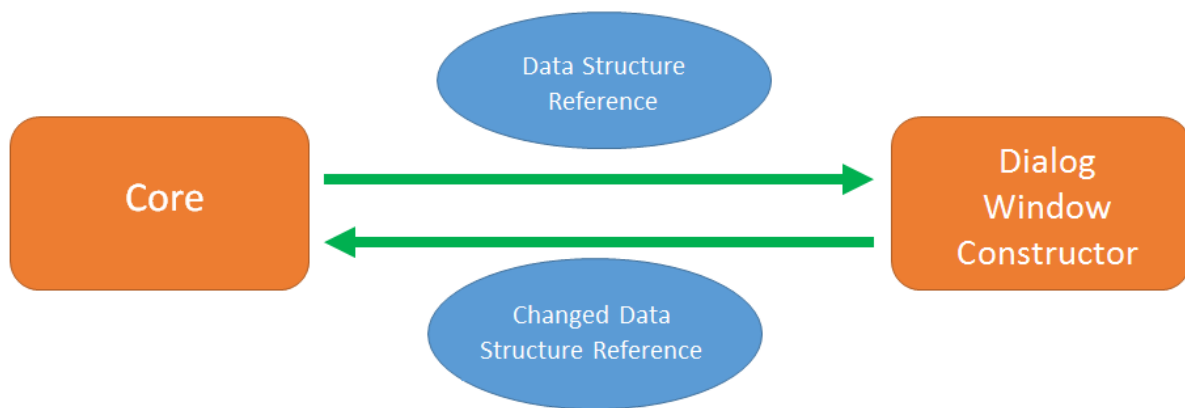


Figure 4.6: Opening exclusion dialog operation diagram.

The `Exclusion_Dialog` has a special feature related to the exclusion position input. It permits the user to enter either coordinates (latitude or longitude) or Fiber Optics position. Both can be set by writing directly in the window, but there is also the possibility of selecting the coordinates on the map. A special button is placed on the `Exclusion` dialog to do that. When it is pressed, the exclusion dialog window (in add or modify mode) hides and emits a signal to `Core` that will execute a JavaScript function to enable capturing the user clicks on the pipeline. This allows to get the coordinates in the next two clicks, which will be returned to the `Exclusion` dialog. This is made with Qt library function by adding a C++ object to the JavaScript platform. In other words, the C++ object can be accessed from the JavaScript part, making possible to execute C++ function and therefore, setting the coordinates obtained from the pipeline clicks. Finally, the `Exclusion` dialog window shows back, with the selected start and end exclusion coordinates in the `Location` fields.

The exclusion management module has access to the database and the Application log module through the `Core`. This allows to insert the corresponding log entry in the application log when needed.

4.2.5 Alarm Handling Module

The Alarm Handling Module is composed of the `Core`, the `MainWindow` interface, including the map, table and the handler buttons, plus some other parts, which correspond to object classes, as shown in

Figure 4.7. They are inter-connected with signals and slots, although the `Core` part is the one which controls the logic flow. This module provides the user to be able to view the alarm list, see them on the map and handle them with the corresponding report of the action taken, besides inserting log entries when an important action is taken.

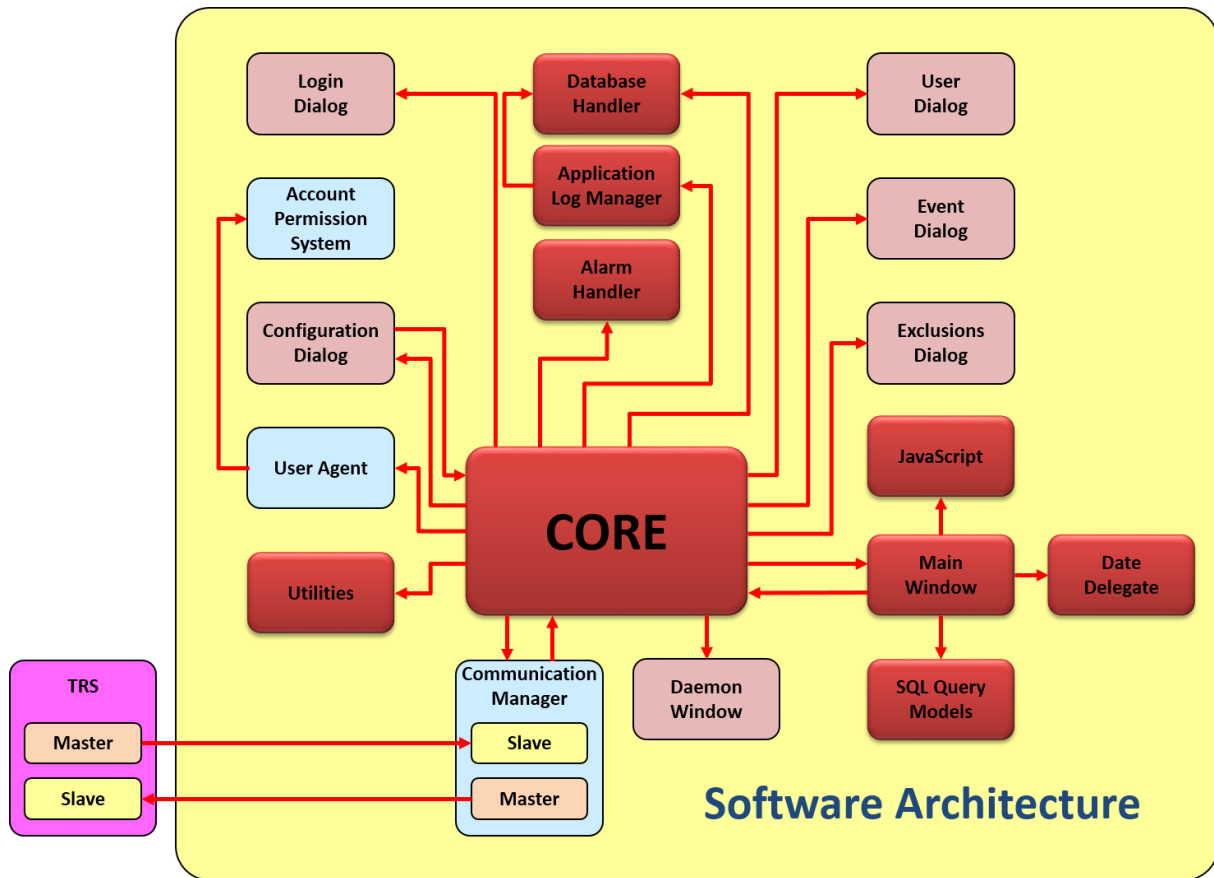


Figure 4.7: Alarm Handler Module diagram.

The `MainWindow` class is responsible for setting up the Alarm module part that involves the Alarm table, map and handler buttons. The Alarm Handling Module, as the other module interfaces too, has been created and designed with the `QtCreator` tool. The alarm table, where alarms are listed, is fulfilled taking them from the database and putting them in a `SQL Query Model`, attached to the exclusion table and inherited from `SQL Query Model`. This keeps the information of the alarm in the model. The function to draw in the table of the model has been overridden to customize it, so it is possible to represent the alarm timestamp with a specific format and to indicate the threat type of the alarm with color icons (Orange and red). In other words, the custom `SQL Query Model` for the alarm table stores the alarm and draws the items in the table properly. In the case of the icons, it checks the threat associated with the alarm, gets the threat type and shows an icon with the corresponding color. The tooltip that appears when the mouse is over an alarm entry is created with the custom `SQL Query Model`, and gets all the alarm parameters by calling a helper function from the `Utilities` class.

The three control buttons, `Silence`, `ACK` and `Reset`, when pressed, they emit a signal to the `Core` and send the alarm ID and the report text (except when the button pressed is `Silence`) that will execute a method in which the alarm handler module is called to handle it and update the alarm in the database (writing the timestamp and report text). In addition, the corresponding log entry will be inserted for the action that was taken.

The alarm map is built with a `QWebView` object of the `Qt` library, loading a `HTML` file with the

Google Map API web. The pipeline is drawn with a KML layer library that parses the KML file. The HTML file also contains several JavaScript functions to add the markers in the position of the alarms when these are selected by the user in the table. When one alarm is clicked or selected, the program calls a JavaScript function to change the alarm marker icons in order to put a special marker shape to the one selected, and normal marker shape to the rest, so that the user knows which one is selected. The location and the text information is passed through arguments, so the marker is drawn in the map in the position of the alarm storing the alarm information. Therefore, when this is clicked it shows a small box with the alarm parameters. The alarm markers are blinking all the time, made by using an animated GIF for the marker images.

4.2.6 Application Log Management Module

The Application Log Management Module, as shown in Figure 4.8, is composed of the MainWindow interface, the Database Handler and the Core which sets it up when the application starts and enables or disables the corresponding buttons depending on whether the user has access or not to the different application parts. The MainWindow interface contains only a table and a filter menu. The user can see all the application logs that are collected from the database. The table to show the logs is made automatically by inserting a custom SQL Query Model for the Application Log table, which controls the output to the table. In this way, the date and time format of the log timestamp can be changed.

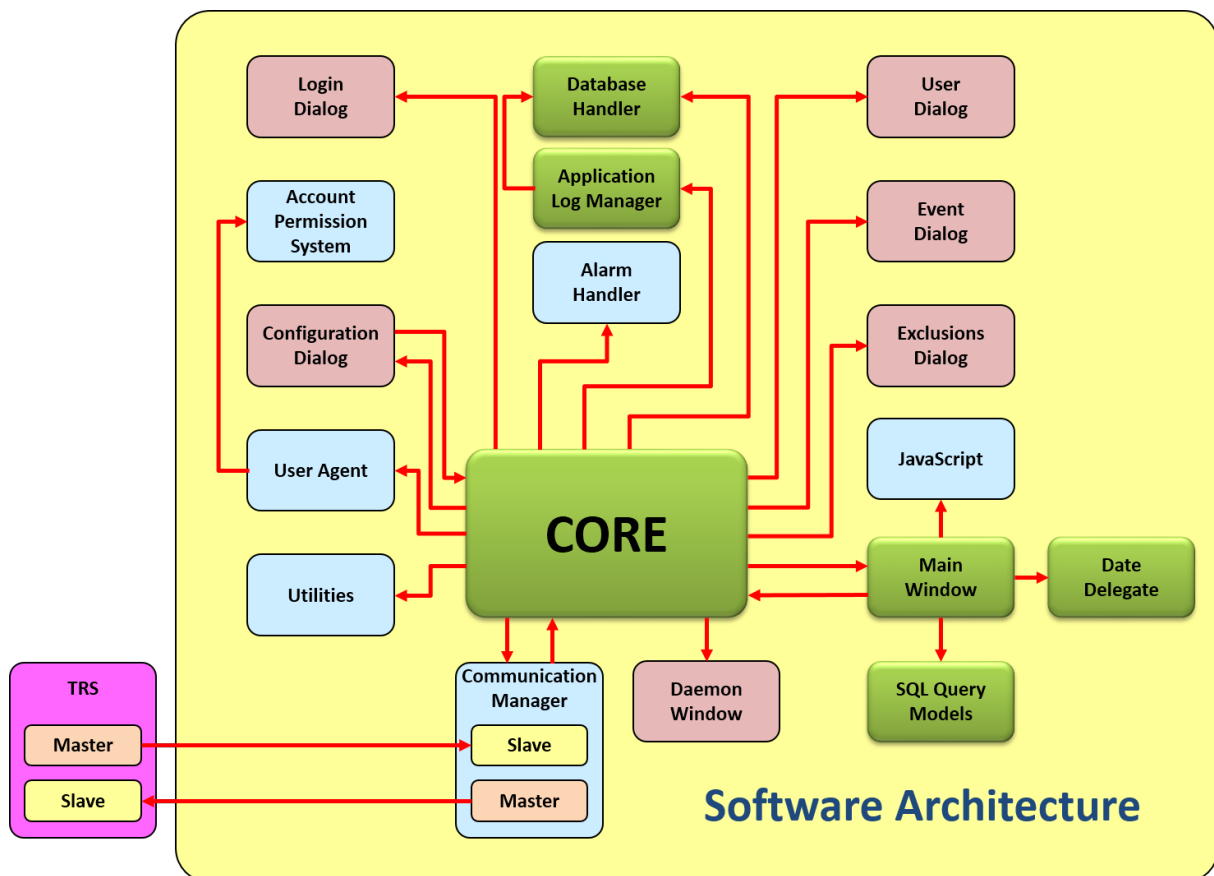


Figure 4.8: Application Log Management Module diagram.

The filter works with the accept button handler. When it is pressed, it reads the configuration of the filter options and prepares a specific query to satisfy that request. After executing the query, the query

model is set again to the application log table which refreshes it with the filter options that the user introduces.

4.2.7 Configuration Module

The Configuration Module is composed of the Configuration Dialog, the Core and some other parts, which correspond to object classes, as shown in Figure 4.9. When the Core receives a signal from the Main window that the configuration button was pressed, it prepares a configuration structure taken from the database, which contains all the configuration parameters and creates a configuration dialog. This structure and the current user permissions are passed to the Configuration Dialog interface through its constructor. In this way, the configuration interface has all the current parameters loaded in its fields and disables the access to the parts that the current user is not allowed to see or modify. This way, the user knows the current configuration. When the user presses accept, all the configuration fields are overwritten in the structure and the configuration dialog closes. The Core takes the control of the application and is responsible for updating the new structure into the database. In addition, it will also call the proper Application Log Management Module method to insert a log with the actual changes.

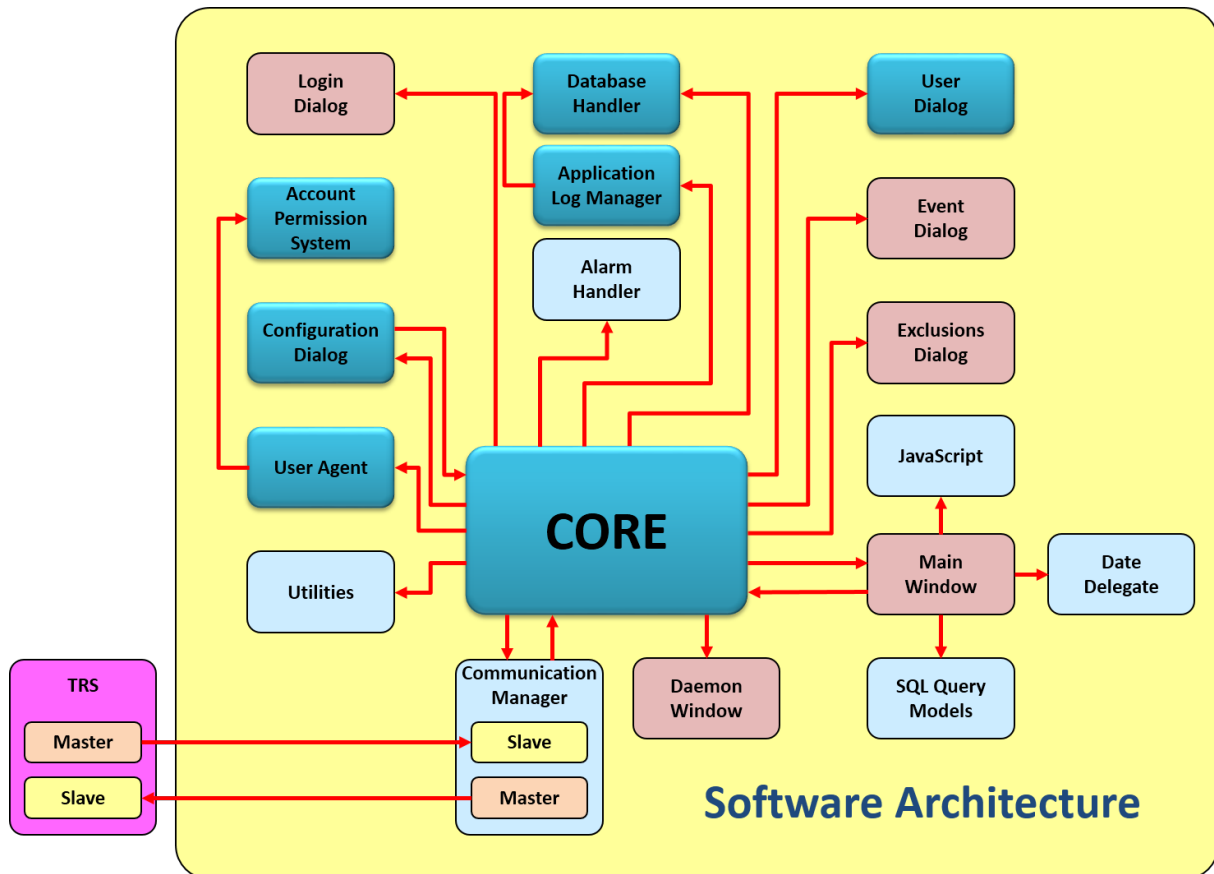


Figure 4.9: Configuration Module diagram.

The configuration interface is built with Qt Creator tool and to ensure that the IP parameters are entered correctly, a validator has been set. The validator checks the IP and if it matches with an IP format, the text box turns into green. Otherwise it turns into red and the user cannot submit the configuration changes.

The user account management is also integrated here. The configuration window interface has a user account list to show the users enrolled in the application. The table list is made as the other tables in

the rest of the program. The user accounts are taken from the database and inserted in a custom query model, which is assigned to the user table and shows the users automatically. To do so, the function which paints the data is modified so in the column of the permissions, it translates the permission bit mask to the corresponding user permission profile (Alarm handler, system controller, Admin or Custom). In this way, the user can actually identify the kind of permissions the accounts have, instead of reading the bit mask, which would be awkward.

The modify, add and deleted actions in the user account table are carried out at real time and it is not necessary to accept the configuration dialog. When the user does those actions, a signal is sent with the target user account ID to the Core, which will perform the account update, insert or remove depending on the required action. The configuration interface is responsible for checking if the user account attempting to be deleted is not the Daemon account, since it is not allowed to be deleted.

Modifying and creating a user account is performed in another window dialog, which is created and opened from the Configuration interface. This user account dialog allows to see, create or modify the user account details, such as full name, login name, permission, account status and password. This is made by passing a user account structure from the configuration interface to the user account interface. In case of modifying a user account, that structure would be filled with the current information of the target user. When accepting the user account dialog, the parameters are fulfilled in the user account structure, so that when the program flow returns to the configuration interface, it can manage that updated account by sending it to the Core through a signal. The Core receives it and is responsible for updating or inserting the account and calls the application log module to create the corresponding application log entry. This process is explained in the diagram shown by the Figure 4.10

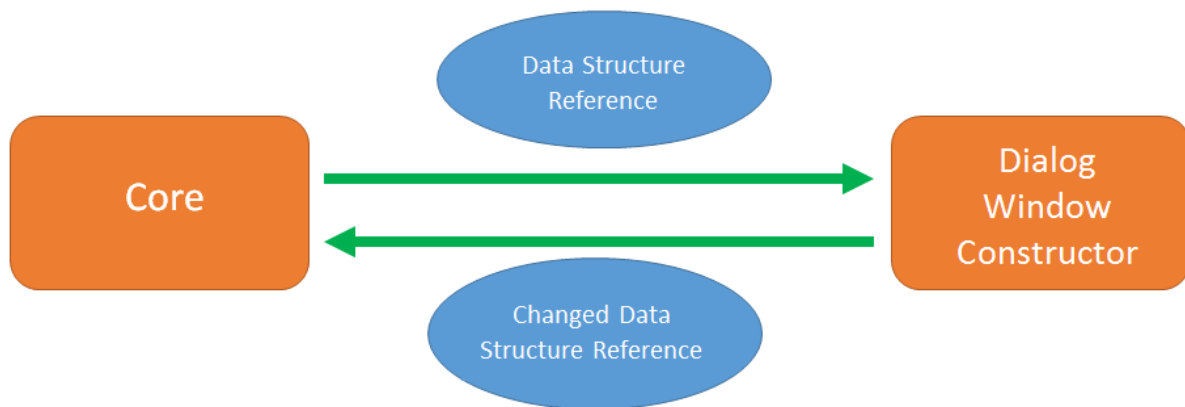


Figure 4.10: Opening user dialog operation diagram.

4.2.8 Database Daemon Module

The Database Daemon Module is composed of the Core, the Daemon window interface, the TRS Communication Module and the Application Log Module, as shown in Figure 4.11. The Daemon user account uses the same modules as the rest of the users. However, it is the only one that has a method to insert new events received from the TRS. It also has a status bar which works exactly the same way as in a normal application access. The Application Log Manager is also used to create a log entry when a new event is recorded in the database.

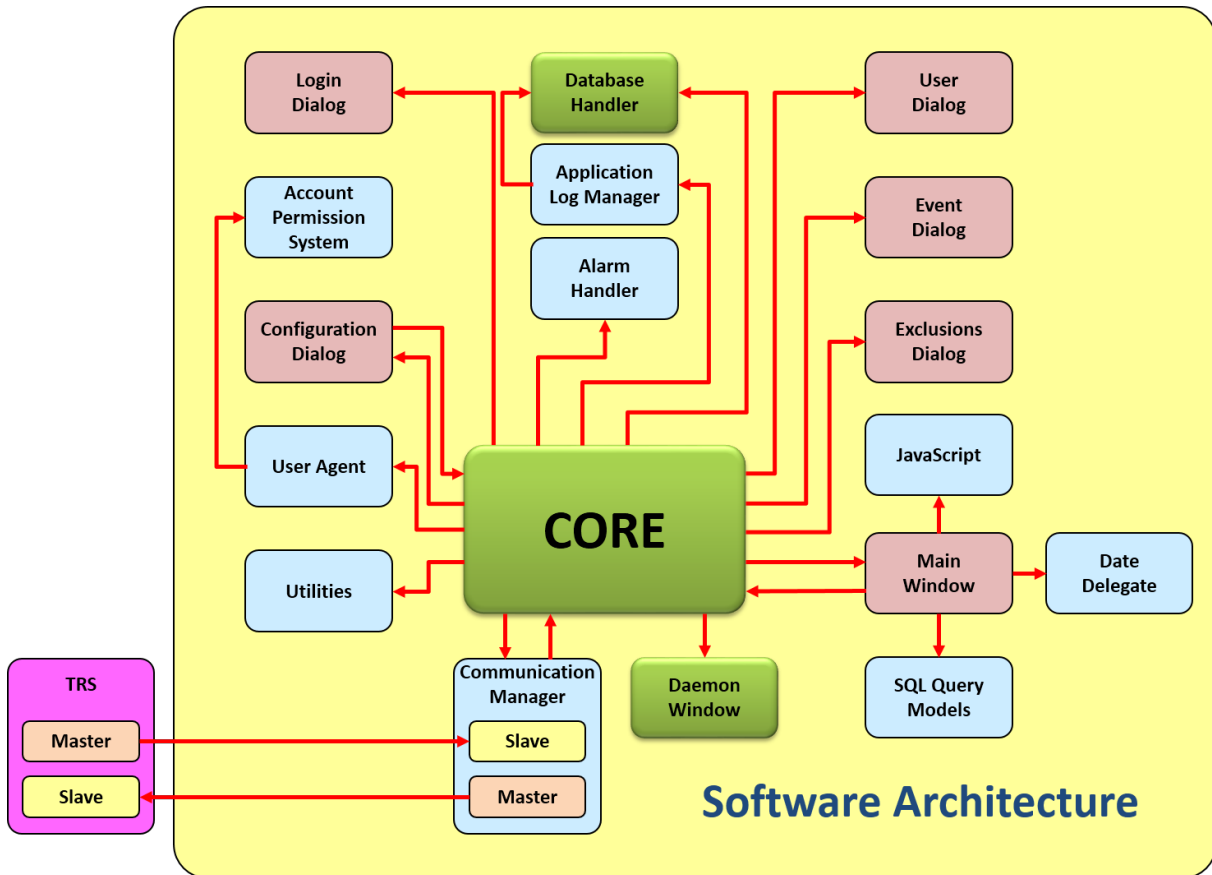


Figure 4.11: Database Daemon diagram.

4.2.9 Exiting the Application

Both exiting and login out from the application are carried out by the Core, the main window interface (buttons pressed) and the application log. Login out from the application frees all resources allocated for the interfaces and modules, except the database module and the application log module. The main interface window closes but the Core is still working, so it opens a login dialog again and therefore, the application is still running and it is possible to access again. However, when the user exits from the application all resources are cleaned up, including the Core. This leads to the end of the application.

Log out and exit buttons, emit a signal that is connected with a handler situated in the Core. This handler carries out what was explained above after showing a confirmation message box.

4.3 Supporting Modules

4.3.1 Database Support Module

The Database Handler Module is used by the Core to create two database connections. One allows to full access and the allows to read only. The full access one is used mainly by the Core, and the read only connection is used by the graphical interface modules, those which do not need to insert nor update into the database. The database error management is carried out in this module as well. It contains a method that is called when the rest of the modules access the database, and checks for errors, showing them in an informative message box with the error text and the native error code generated. This module is also responsible for ensuring that the user knows in real time when the database is connected. To perform

that, the module contains a timer that executes every seven seconds a routine to check the connection, keeping running until application exits. Both when the database is connected or disconnected, the proper signal is emitted from the database module to the Main window to update the status bar.

4.3.2 Alarm Handler Support Module

The management and the alarm sound are carried out by this module which is used by the Core. It has three lists for each alarm status(unhandled or not silenced alarms, silenced alarms, and ACK'ed alarms). The aim of this module is to manage this list. When an alarm is silenced, it goes from the unhandled alarm to the silenced alarm list. When an alarm is ACK'ed, it goes from silenced alarm to ACK'ed alarm. And, finally, when an alarm is reset it gets out from the ACK'ed alarms and therefore from the alarm handler. Therefore, the corresponding alarm is now considered handled. This module performs the alarm management logic. For instance, a silenced alarm cannot be silenced again. It is also responsible for the alarm buzzer. When there is a change in the alarm list, the unhandled alarm list is checked out and if it is non empty the alarm sound continues playing. The buzzer stops when that list is empty.

4.3.3 TRS Communication Support Module

The TRS Communication module is composed of the Core, PRS Communication Manager and some other parts, as shown in Figure 4.12, to send and receive commands from the TRS. It is split in two other modules, slave and master. The slave connects to the master of the TRS and the master does with the slave of the TRS. The slave is responsible for carrying out the tasks that TRS orders, such as setting or getting parameters. However, the master sends commands to the TRS to change parameters on it. When a command is sent with a request (either getting or setting a parameter), an ACK command must be sent back through the command port. The data port is used to send the parameters in a special packet when it is needed.

The Communication module has the feature of reconnecting automatically when the TRS is down. It has a routine that initializes the master and the slave and their sockets. When the slave socket receives an error signal from the QSocket library and the communication is closed, the error handler starts a timer that calls, after seven seconds, the routine to connect again to the TRS. This code repeats automatically until the TRS is up and the UI can connect it again. Since a QTimer object is used, this process is done in background which leads to non blocking routine.

When the Slave receives a command, it sends the command through a signal that will be handled in the Communication manager by a Slot. This handler method contains a list of the possible commands and calls the corresponding functions depending on the command. The communication manager has some special functions to insert and get the configuration options from the database, so it is not needed to send a signal to the Core. However there is one exception regarding the new incoming events from the TRS. The Communication manager sends a signal when it receives a new event, and the Core handles it in one way or another depending on the current user. If it is a normal user account which gets the new event, it will only refresh the alarm table. On the contrary, if it is a Daemon special account, it will insert the new event in the database.

The Master contains methods to send commands with some informative data. For instance, when a change is made in the configuration options in the UI, the Core uses this module to send the commands with the updated configuration parameters to the TRS.

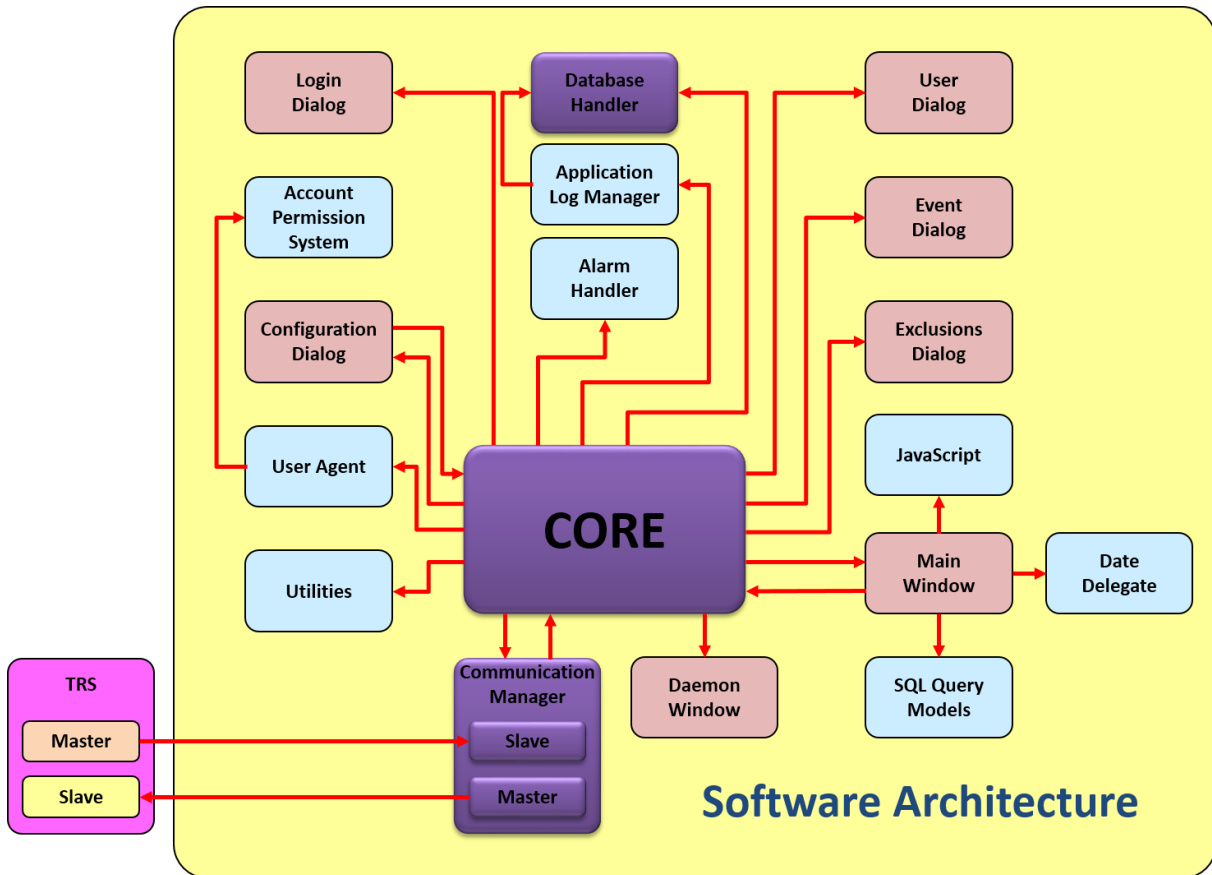


Figure 4.12: Communication Module diagram.

4.3.4 Application Log Support Module

The `Application Log Module` is responsible for registering all activities of the user does in the application. It has full access to the database. It contains a method to insert an application log for every kind of activity. All the methods are called from `Core`, when an action is carried out. This module also has several helper functions to, in some cases, check exactly the parameters that were changed or modified. In the definition header of the software there are all the structures and every parameter that will correspond to a bit in the change mask. In other words, there is a change mask for every structure (event, exclusion, configuration, etc.) in which bits represent a specific parameter. Therefore, when a log entry is about to be inserted, this module calls some of its methods with the old and the updated structure (by the user who modified it) with the corresponding arguments and performs a check of every parameter with the updated one to make the structure with the information of changes.

The additional information of a log entry is created by other helper functions inside the module. However, when an event, exclusion or the configuration is modified, the mask containing the user changes is given to a specific function to create the text corresponding to the additional details and including only the information of the modified fields, writing the old and the updated parameter. This way there is an accurate record of the changes done by the user.

4.3.5 User Agent and Account Permissions System Support Module

The `User Agent` class is made for holding the user account that is actually logged in the application. It contains a `User` structure with all the user information. The class inherits from the `Account permission`

system class, which provides a permission management. This class has a set of functions which help in the permission checking when the `Core` requires to ensure that the current logged user has the necessary privileges. The `Core` is responsible for enabling or disabling the buttons that provide access to the different areas in the application.

When a log in succeeds, the `Core` creates a user agent object with the user information, so that it has the user parameters when the core needs. For instance, when inserting an application log entry, the user agent object is crucial to know the information of the actual user that is doing a certain action.

4.4 Conclusions

The implementation described in this chapter provides the reader a more detailed description of the development of this software application. As the reader can see, the application is divided into several modules formed by different objects. The connection between them describes how the software code works, which is very important for making future improvements. A more detailed low level explanation is given in the documentation of the code, although the implementation explanation of this chapter is highly recommended to read and understand before going through the specific documentation of the code.

Chapter 5

Evaluation

5.1 Introduction

This section covers the reliability and evaluation procedures of the TRS-UI software. The evaluation tests have not been carried out only during the development process, but also at the end of it. The software used to perform the tests are the PRS Simulator and a MySQL Database, on Ubuntu 14.04 LTS. The detailed procedures of the application evaluation are described next.

5.2 Evaluation Procedures and Results

5.2.1 Application access

The TRS-UI access has been tested with several accounts set with different account profiles each, to ensure the areas of the program that the user does not have access to, are not enabled. In this way, it is guaranteed that the user will only have access to the allowed parts, according to the account permissions. The tested accounts were modified during the evaluation to check login attempts with wrong passwords, not activated accounts and custom permissions.

5.2.2 Event Management Module

A set of simulated events have been created to provide the test some examples to work with. First of all, the event table is checked to see that the showed events and their parameters match with the same event in the database. The filter is tested changing the filter options and checking the output on the table. On the other hand, the map and the event selection is tested to check that the selected events correspond to the ones showed in the map with the proper marker and characteristics. Multi selection is available here, so it has been evaluated as well. Last but not least, the pop-up menu was tested with several modifications, creations and deletions of event examples, checking in the database the relevant parameters at each time.

5.2.3 Application Log Module

The test of Application Log Module needs a lot of logs examples. These logs have been created during the development process when testing the application. The database was checked during the

process to ensure the correct operation of this module. The filter was tested by analyzing every part of it alone and verifying the output.

5.2.4 Alarm Handler

The way to test the alarm handler has been to try to simulate the normal procedure that a user would do to handle a set of alarms. So, from the TRS simulator, several alarms were fired with different parameters and threats. After checking the correct operation of the alarm icons, including the color and the shape, the alarms were handled one by one, ensuring that this goes through the different states, until is reset. Since this application software has been designed to work in a distributed system (more than one alarm handlers), the test were carried out with several user accounts handling the same events.

5.2.5 Exclusion Management Module

Some simulated exclusions have been created to provide the tests some examples to work with. First of all, the exclusion table is checked to see that the showed exclusion and its parameters matches with the same exclusion in the database. The filter is tested changing the filter options and checking the output on the table. On the other hand, the map and the exclusion selection are tested to check that the selected exclusions correspond to the ones showed in the map with the proper marker and characteristics. Milt selection is available here, so it has been evaluated as well. Last but not least, the pop-up menu was tested with several modifications, creations and deletions of exclusion examples, by checking in the database the relevant parameters at each time. One important feature of the exclusion manager is the capability to set the exclusion coordinate start and end points by clicking on the map, instead of entering the raw coordinates. This procedure has been checked many times and can be a bit slow sometimes.

5.2.6 Configuration

The configuration module has been tested with the Database and TRS Simulator. Several configuration parameter changes, including user account creation, modification and deletion, have been carried out for this test. To ensure that the configuration changes were successfully committed, the proper database fields, and TRS Simulator output were checked.

5.2.7 Daemon Service

The Daemon Service has been tested with the TRS Simulator, sending a set of new events with different parameters and alarm threats. The database shows that every event is correctly inserted on it and on the alarm handler part, the events with green threats are not put on it.

5.2.8 Status Bar

The Status Bar tests requires connections and disconnections of the network between the TRS-UI, TRS and Database. Due to both TRS and Database are in the same local host during the evaluation procedures, the disconnections and connections have been carried out by changing the access password to the database, the database path and killing the PRS Simulator process. These tests simulate several disconnections that the TRS and Database could have due to technical problems, and permitted to ensure that the Status Bar was working correctly, showing the proper connection status of them.

5.3 Conclusions

The results are a success and prove the correct functionality of the application, although it would be much better to carry out the test in a real simulation scenario, which means, several computers forming a distributed system.

It is important to mention that this software application is a part of the prototype PITSTOP project. This means that this application has not been tested in a real situation and therefore can be susceptible to changes.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The TRS–UI software application was developed following specific guidelines in order to meet the desired requirements of the PITSTOP system. This leads to a unique application to manage the alarms generated from FINDAS and to control the configuration of the different systems involved in the PITSTOP project. The TRS–UI application has been carried out through several meetings with PITSTOP project members, in which some specific details and features of the application were discussed and defined. Due to the PITSTOP project is nowadays under construction, it is a prototype that is built to answer to a need of increase the security regarding gas pipelines.

This bachelor thesis offers the user an intuitive, sophisticated and versatile user interface, which can be used to:

- Access and manage the information related to all events generated by the TRS: view the events in the map, create a heat map, modify their parameters, create and delete them.
- Handle alarms and report them.
- Create exclusion areas in the map with a specific start and end time, where events will not sound.
- Change the TRS and FINDAS configuration remotely.
- Access and manage the application log reports of all users enrolled in the system.
- Set up the Daemon Service to automatically store the received events from the TRS to the Database.
- Manage User accounts: Create, modify and delete accounts with specific characteristics, such as application permissions, which establish the actions that the user is allowed to carry out.

Even though all objectives of the project have been successfully reached, there are several improvements that can be done. The ideas of those improvements occurred during the different stages of the developing process. However, they have not been done so far due to some limitations such as time. Therefore, they are presented in the section below as suggestions for future work, in order to make the current software application more effective and complete.

6.2 Future Work

6.2.1 Introduction

This section gives some improvement ideas about the future of the application. The main goal of this application software is satisfied with the current version although some useful improvements can be implemented. Some ideas were originated by the author of this thesis and some others were proposed in the overall PITSTOP project.

6.2.2 Database access password management

The database access is granted with two specific database user names and passwords hard coded in the application, thus being transparent for the end user. This idea holds to create a small module or option in the configuration to be able to change the database password easily from the application.

6.2.3 Add Daemon status in the status bar

The TRS-UI provides the status information of three important parts of the whole system that are TRS, FINDAS and Database, but no information is known about the Daemon, which is crucial for the system. This idea suggests to place another status label to inform whether there is a Daemon connected to TRS or not.

6.2.4 Implementing TRS Dynamic Training Procedures

The PITSTOP project suggests an extension of the TRS-UI application regarding acoustic traces associated to an event. This refers to visualize, listen, make selection and deleting the acoustic traces. This leads to labelling the acoustic trace with the ground truth information of the same event, so that it can be used for dynamic training.

6.2.5 Automatic port configuration

The TRS-UI uses the configuration options getting them from the Database which prevents another TRS-UI to connect in the same computer. Changing automatically the UI ports when starting the application, will permit to execute more than one TRS-UI in the same computer at the same time. This can be done by incrementing the port value every time a TRS-UI starts.

Bibliography

- [1] “Qt library webpage,” <http://www.qt.io/> [Last accessed 23/june/2015].
- [2] “Qt Creator webpage,” <https://www.qt.io/download-open-source/> [Last accessed 23/june/2015].
- [3] “7th egig report gas pipeline incidents,” European Gas Pipeline Incident Data Group, Tech. Rep., 1970-2007, <http://www.egig.eu/uploads/bestanden/d1244d38-8194-46e8-89f4-6b6258d05f3a>.
- [4] H. Martins, D. Piote, J. Tejedor, J. Macias-Guarasa, J. Pastor-Graells, S. Martin-Lopez, P. Corredera, F. D. Smet, W. Postvoll, C. H. Ahlen, and M. Gonzalez-Herraez, “Early detection of pipeline integrity threats using a smart fiber-optic surveillance system: The pit-stop project,” *International Conference on Optical Fiber Optics Sensors*, September 2015.
- [5] “Doxygen main page,” <http://www.stack.nl/~dimitri/doxygen/> [Last accessed 23/june/2015].
- [6] J. T. Nogueras and J. Macias-Guarasa, “Pit-stop report: Trs ui specification,” FOCUS, S.L., Tech. Rep., 2014.
- [7] “Google Maps JavaScript API webpage,” <https://developers.google.com/maps/documentation/javascript/?hl=eng> [Last accessed 23/june/2015].

Appendix A

User Manual

A.1 Introduction

This manual is a simple guide for the users to understand how to use the application. One important feature of this application is that it is made to be intuitive, but there are some parts that may require some further description.

A.2 General description

A.2.1 PIT-STOP Working Context

The PIT-STOP general architecture, describing a very high level of interactions, is shown in Figure A.1 [6]. We consider that, at least, two user roles will be required: one in charge of getting the alarm notifications and launching the actions to be taken (alarm controller), and the other in charge of managing the TRS core system and the FINDAS, including the handling of the signals associated to actual alarms for improving the system performance (system controller). The TRS system controller role could be further divided in two roles: managing the core system (programming), and managing actual alarms for system improvement.

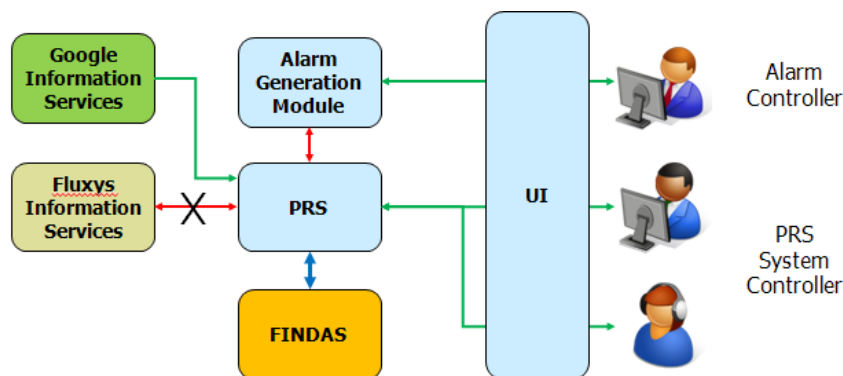


Figure A.1: PITSTOP Project. General Architecture.

A.2.2 System Architecture

The TRS-UI is designed to work with a Database and the TRS. Without these important parts, the TRS-UI program has no point at all. The applications run in one computer each and every user must have a personal account to access. The computers running the TRS-UI need to be connected to the Database and PRS. Therefore, the system architecture must ensure that they are inter-connected and they can reach each other in the network. To do so, the Admin or whoever has access to the configuration menu, must set the PRS and FINDAS network configuration properly which includes IPs and ports, in order to allow the TRS-UI to connect. After the network configuration changes, every TRS-UI should reconnect again to the given network options.

The Database password is hard coded. Therefore, in order to change it, the code must be recompiled again. The database Daemon is crucial for the system to work properly. It inserts the new events to the database, so it is totally necessary to run it in a computer. This is recommended to be done where the PRS is working. Only one Daemon must be running at a time, otherwise it can lead to duplicate events.

A.2.3 User Roles and Permissions

The application has several user roles and permissions. Each permission provides access to a specific part of the application or some kind of action. The user roles are user profiles with a unique set of permissions, but it is possible to set a custom profile of permissions by selecting manually which permissions the user has. The permissions that the application has are the following:

- Alarm: Alarm viewer access.
- Events: Event viewer access.
- Events Add: Able to add a new event.
- Events Modify: Able to modify events.
- Events Delete: Able to delete events.
- Exclusions: Exclusion viewer access.
- Exclusions Add: Able to add a new exclusion.
- Exclusions Modify: Able to modify exclusions.
- Exclusions Delete: Able to delete exclusions.
- Users: User list access.
- Users Add: Able to add a new user account.
- Users Modify: Able to modify a user account.
- Users Delete: Able to delete a user account.
- Directories: Able to view and modify directories.
- Network: Able to view and modify network configuration.
- FINDAS: Able to view and modify FINDAS configuration.
- TRS: Able to view and modify TRS configuration.

- App Log: Application log viewer access.

The user roles or account profiles available in the application are as follows:

- Alarm controller: has access to alarm handler and event viewer.
- System controller: has access to alarm handler, full access to event and exclusion viewer.
- Admin: has full access, except to alarm handler.
- Daemon: has access only to Daemon interface.

A.3 Functionality

A.3.1 Accessing the Application

To access the application the user must execute the program and a login window will appear, as shown in Figure A.2, where the user will enter both user name and password. It is necessary to be enrolled in the system account list and to be set up as an active user. Otherwise, the access will be denied, even though the credentials are correct. When the access is granted, the user will get into the application but will only be able to access in the sections that has permissions to. Whether the credentials are wrong or the user attempting to log in is not active, the access will be denied and an informative message will appear.

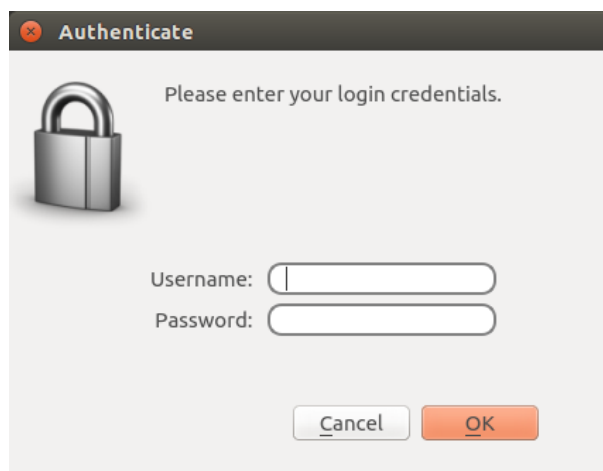


Figure A.2: Login dialog.

A.3.2 Event Management Module

Click on Main button on the top menu and press Events . The event module will appear and provide an interface to see and manage all the events in the application, as shown in Figure A.3. When the application starts, all the events are shown by default in the event table.

A.3.2.1 Filtering events

Filtering the events by time can be done with the filter below the event table. Another way to do it is to go to the top menu and press view, where additional filter options can be found.

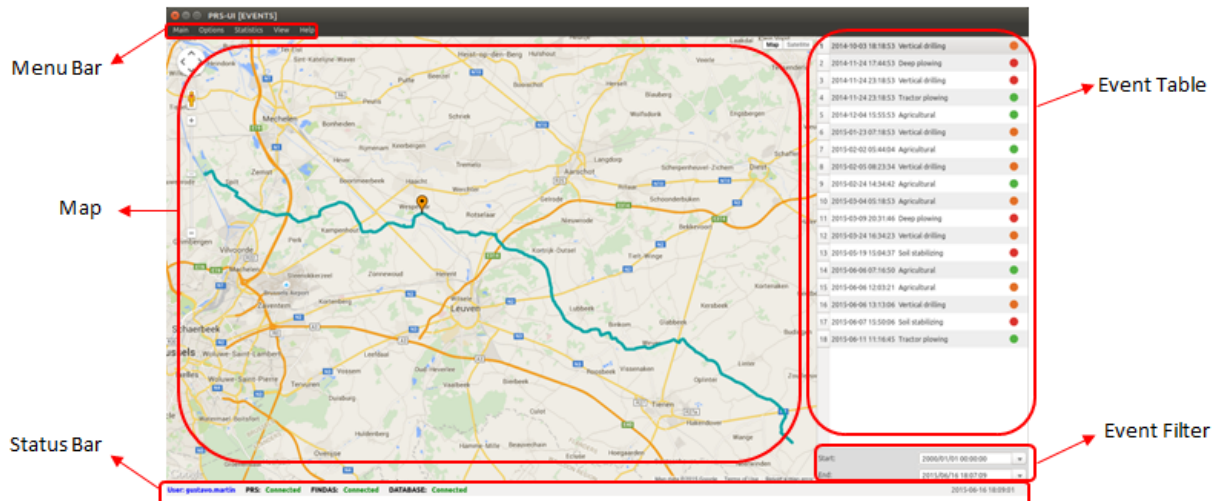


Figure A.3: Event viewer.

A.3.2.2 Seeing the information of an event

Press an event entry in the event table. A marker will appear in the location of this event in the map. The marker color corresponds with the threat color of the event. To see the information of an event, click on the marker on the map or place the cursor on an event in the event table. With both ways, an informative box will appear in a tool-tip. Multiple selection is allowed by holding `CTRL` or `SHIFT`, thus all events selected will be shown in the map with their corresponding markers.

A.3.2.3 Modifying an event

Right-click on the target event and press `Modify` in the pop-up menu. Multiple event modification is not possible. Modify the desired event parameters in the event window that will appear and press `Accept` to save changes. To abort modifying the selected event, press `Cancel` and no change will be saved. An event modification dialog example is shown in Figure A.4.

A.3.2.4 Adding an event

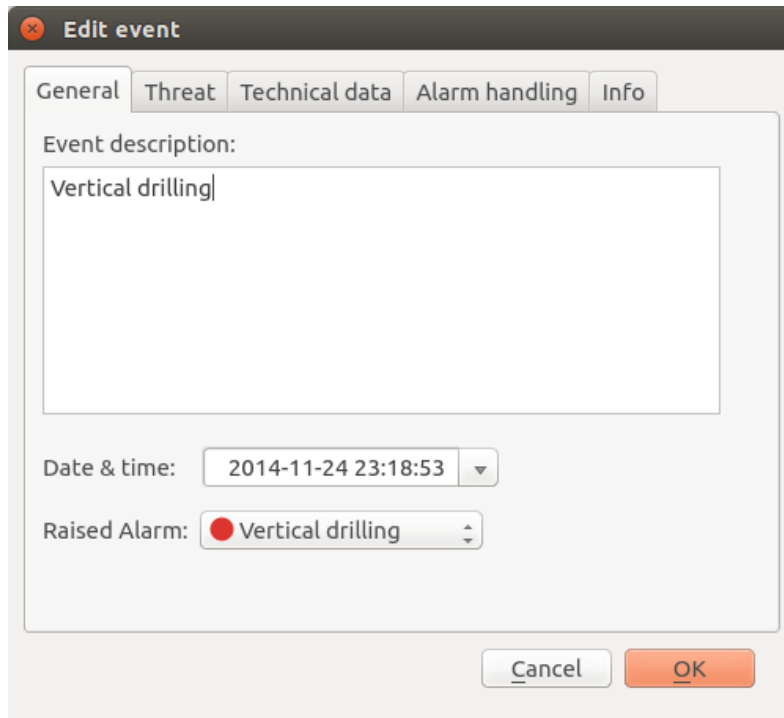
Right-click in the event table and click `Add` in the pop-up menu. Fill in all the event parameters in the event window that will appear and press `Accept` to create the new event. An event add dialog example is shown in Figure A.5.

A.3.2.5 Deleting an event

Right-click in the event to be deleted from the event table and click `Delete` in the pop-up menu. Multiple deleting is possible by selecting the events (holding `CTRL` at the same time) before opening the pop-up menu. A message box will be shown to confirm the operation.

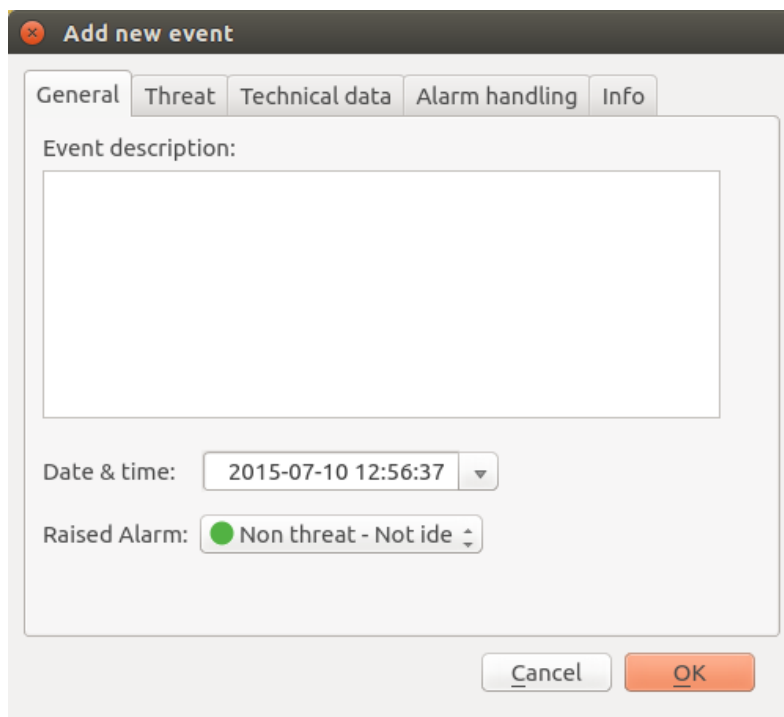
A.3.2.6 Showing the logs related to an event/s

Right-click in the target events to open the pop-up menu and press `Show logs`. All the application logs related to the selected events will be shown in the application log. Multiple selection is allowed.



The 'Edit event' dialog box features a title bar with a close button and the text 'Edit event'. Below the title bar are five tabs: 'General', 'Threat', 'Technical data', 'Alarm handling', and 'Info'. The 'General' tab is active. The main content area is titled 'Event description:' and contains a text input field with the text 'Vertical drilling'. Below this field are two controls: 'Date & time:' with a dropdown menu showing '2014-11-24 23:18:53' and 'Raised Alarm:' with a dropdown menu showing a red circle icon and the text 'Vertical drilling'. At the bottom right of the dialog are two buttons: 'Cancel' and 'OK'.

Figure A.4: Event modification dialog.



The 'Add new event' dialog box features a title bar with a close button and the text 'Add new event'. Below the title bar are five tabs: 'General', 'Threat', 'Technical data', 'Alarm handling', and 'Info'. The 'General' tab is active. The main content area is titled 'Event description:' and contains an empty text input field. Below this field are two controls: 'Date & time:' with a dropdown menu showing '2015-07-10 12:56:37' and 'Raised Alarm:' with a dropdown menu showing a green circle icon and the text 'Non threat - Not ide'. At the bottom right of the dialog are two buttons: 'Cancel' and 'OK'.

Figure A.5: New event dialog.

A.3.2.7 Showing the heat map

Filter the events wanted to be shown with heat map in the table, then open *Statistics* in the top menu of the application and right-click on *Show heat map*. To stop seeing it, just click *Hide heat map*. A heat map example is shown in Figure A.6.

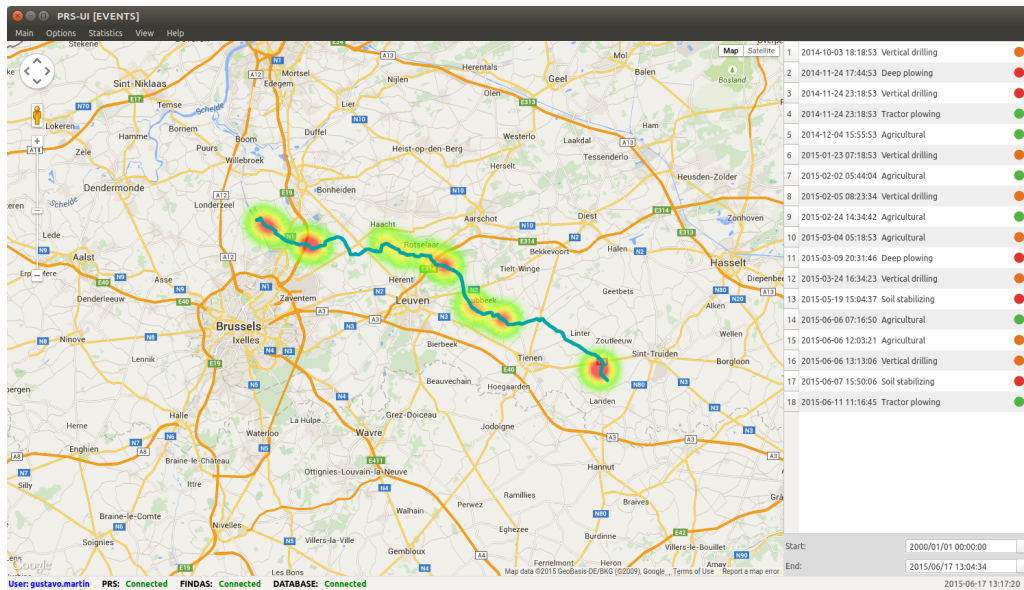


Figure A.6: Heat map example.

A.3.3 Exclusion Management Module

Click on Main button on the top menu and press Exclusions. The exclusion module will appear and provide an interface to see and manage all the exclusions in the application. When the application starts, all the exclusions are shown by default in the exclusion table. An exclusion visualization example is shown in Figure A.7.

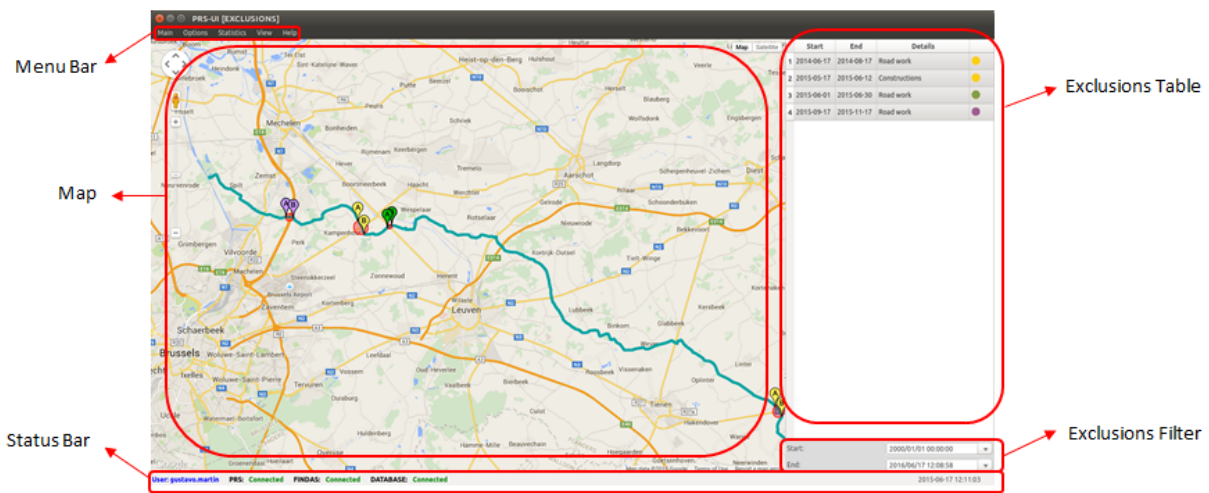


Figure A.7: Exclusion viewer.

A.3.3.1 Filtering exclusions

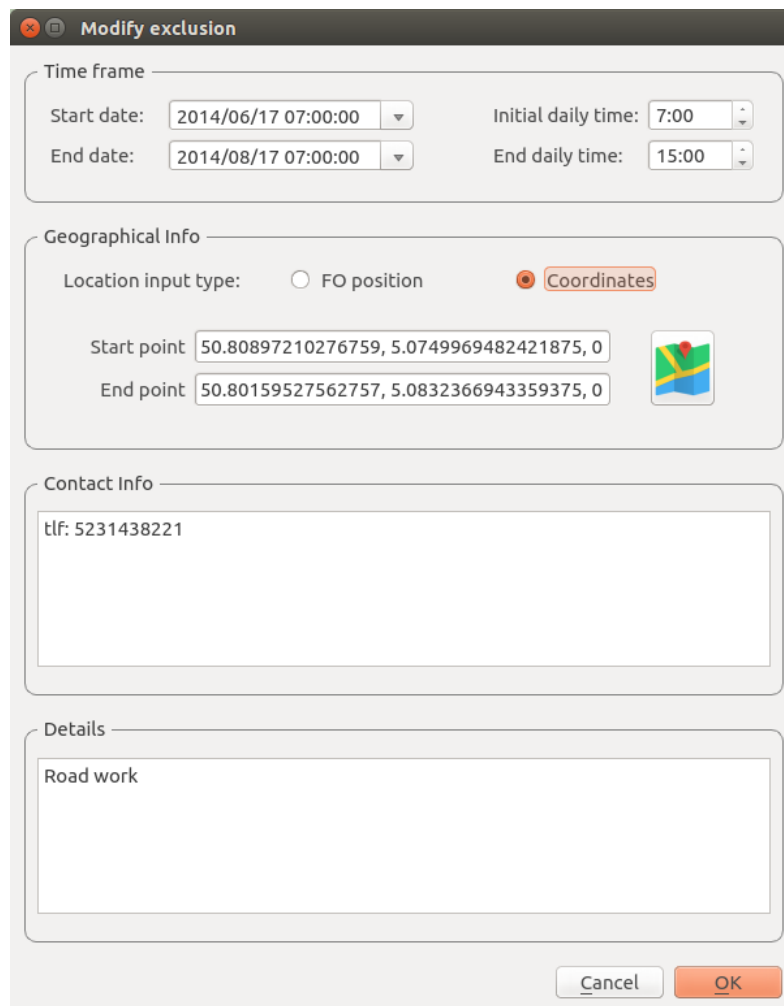
Filtering the exclusions by time can be done with the filter below the exclusion table. Another way to do it is to go to the top menu and press View, where additional filter options can be found. Here, there is another type of filter, which allows to see only the exclusions that are past, present and future.

A.3.3.2 Seeing the information of an exclusion

Press that exclusion entry in the exclusion table. A pair of markers will appear in the location of the start and end exclusion in the map. The marker color corresponds with the status of the exclusion. To see the information of an exclusion, either click on the marker in the map or place the cursor on an exclusion in the exclusion table. With both ways, an informative box will appear. Multiple selection is allowed by holding CTRL or SHIFT, thus all exclusions selected will be shown in the map with their corresponding markers.

A.3.3.3 Modifying an exclusion

Right-click on the target exclusion and press `Modify` in the pop-up menu. Multiple exclusion modification is not possible. Modify the desired exclusion parameters in the exclusion window that will appear and press `Accept` to save changes. To abort modifying the selected exclusion press `Cancel` and no change will be saved. An event add dialog example is shown in Figure A.8.



The image shows a 'Modify exclusion' dialog box with the following fields and options:

- Time frame:**
 - Start date: 2014/06/17 07:00:00
 - End date: 2014/08/17 07:00:00
 - Initial daily time: 7:00
 - End daily time: 15:00
- Geographical Info:**
 - Location input type: FO position, Coordinates
 - Start point: 50.80897210276759, 5.0749969482421875, 0
 - End point: 50.80159527562757, 5.0832366943359375, 0
- Contact Info:**
 - tlf: 5231438221
- Details:**
 - Road work

Buttons: Cancel, OK

Figure A.8: Exclusion modification dialog.

A.3.3.4 Adding an exclusion

Right-click in the exclusion table and click `Add` in the pop-up menu. Fill in all the event parameters in the exclusion window that will appear and press `Accept` to create the new exclusion. An exclusion add

dialog example is shown in Figure A.9.

The dialog box is titled "Create new exclusion area". It is divided into four main sections:

- Time frame:** Contains "Start date" and "End date" dropdown menus, both currently showing "2015/07/10 12:55:49". It also has "Initial daily time" and "End daily time" dropdown menus, both showing "0:00".
- Geographical Info:** Features a "Location input type:" label with two radio buttons: "FO position" (which is selected) and "Coordinates". Below this are "Start point" and "End point" text input fields, and a small map icon to the right.
- Contact Info:** A large, empty text area for entering contact information.
- Details:** Another large, empty text area for providing additional details.

At the bottom right of the dialog, there are two buttons: "Cancel" and "OK".

Figure A.9: New exclusion dialog.

A.3.3.5 Setting the geographical position of an exclusion

When adding or modifying an exclusion, the geographical position of an exclusion can be set, either with coordinates or with FO position, just by selecting the geographical position in input type. If coordinates are entered, the application will automatically translate them to FO position. The same goes if the FO position is entered first.

A.3.3.6 Deleting an exclusion

Right-click in the exclusion to be deleted from the exclusion table and click `Delete` in the pop-up menu. Multiple deleting is possible by selecting the exclusions (holding `CTRL` at the same time), before opening the pop-up menu. A message box will be shown to confirm the operation.

A.3.3.7 Showing the logs involving an exclusion

Right-click in the target exclusions to open the pop-up menu and press `Show logs`. All the application logs related to the selected exclusion will be shown in the application log. Multiple selection is allowed.

A.3.4 Application Log Management Module

Click on `Main` button on the top menu and press `Application Log`. The application log module will be shown and the user will be able to see the application logs. Clicking in the column names sorts the current entries by alphabetical order, in descend or ascend way. A application log module example is shown in Figure A.10.

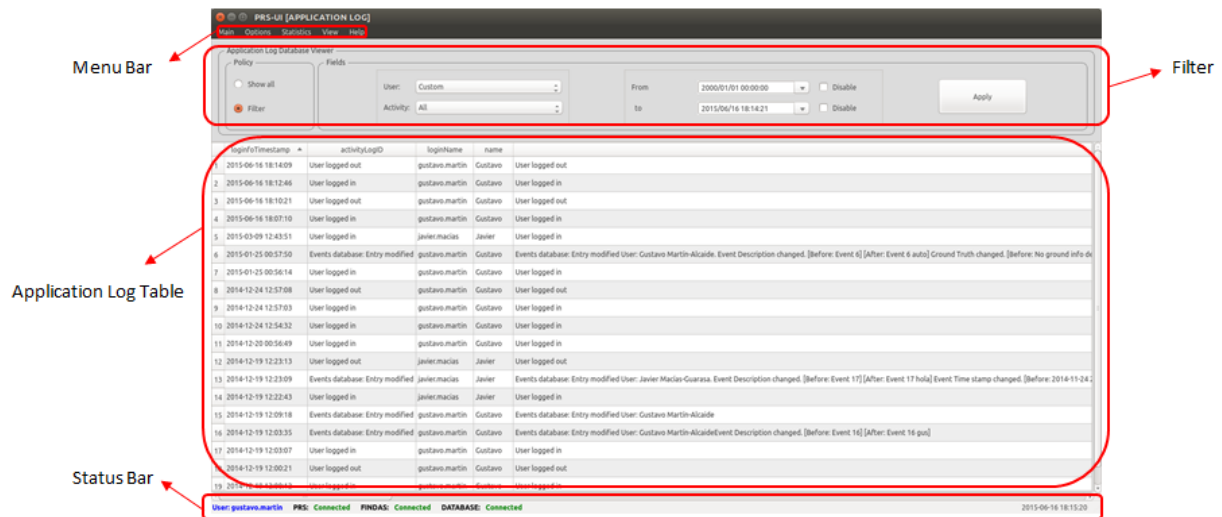


Figure A.10: Application Log viewer.

A.3.4.1 Filtering the application log entries

Above the log table there are filter options. To show all available entries select `Show all` and to set the filter select `Filter`. If the `Filter` button is set, all filter options will be enabled and available to configure the filtering option. To see the results, click `Accept` button.

A.3.4.2 Showing the events related to a log

Right-click in the target logs corresponding to an event. In the pop-up menu, press `Show events`. All the events related to the selected application logs will be shown in the event section automatically. Multiple selection is not available.

A.3.4.3 Showing the exclusions related to a log

Right-click in the target logs corresponding to an exclusion. In the pop-up menu, press `Show events`. All the exclusions related to the selected application logs will be shown in the exclusion section automatically. Multiple selection is not available.

A.3.5 Alarm Handling Module

Click on `Main` button on the top menu and press `Alarm handling`. The `Alarm Handling Module` will appear, as shown in Figure A.11, and provide an interface to see and manage all the alarms in the application. On the right there is a table which contains all unhandled alarms, showing the description, the alarm type with a color icon and three more icons to see if the alarm is silenced, `ACKed` and `Reset`.

When the application starts, the events that are not handled yet will be shown as alarms in the alarm table. The alarms are classified by the threat color, being orange a medium emergency and red a high emergency.

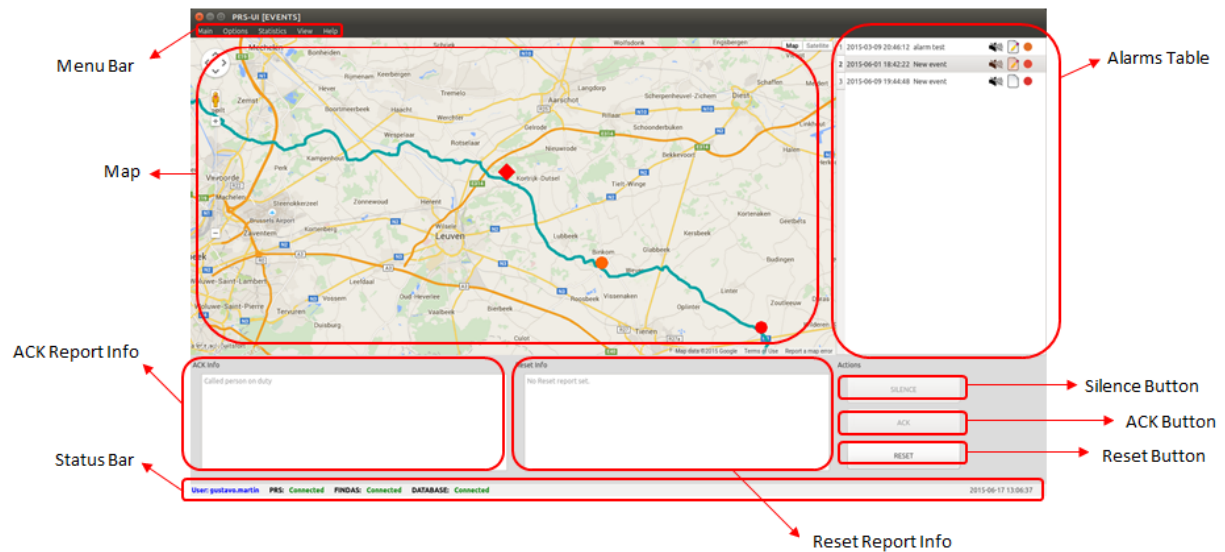


Figure A.11: Alarm viewer.

A.3.5.1 Seeing the information of an alarm

Press the alarm entry in the alarm table. The marker will be shown as selected (square instead of a circle) and will appear in the location of the alarm on the map. The marker color corresponds with the alarm type, which can be red or orange, depending on the emergency of the event. To see the information of an alarm, either click on the marker in the map or place the cursor on an alarm in the alarm table. With both ways, an informative box will appear. Multiple selection is not possible.

A.3.5.2 Silencing an alarm

Select the alarm and click Silence button.

A.3.5.3 ACKing an alarm

Select the alarm, write the ACK report and click the ACK button.

A.3.5.4 Reset an alarm

Select the alarm, write the Reset report and click the Reset button.

A.3.6 Configuration Module

Press Configuration on the top menu. The configuration window will appear and the users will be able to change the parameters. To save the changes press Accept and to abort them press Cancel.

There are several parts in the configuration. The first is the user account viewer, as shown in Figure A.12, where the given account can be created or edited.

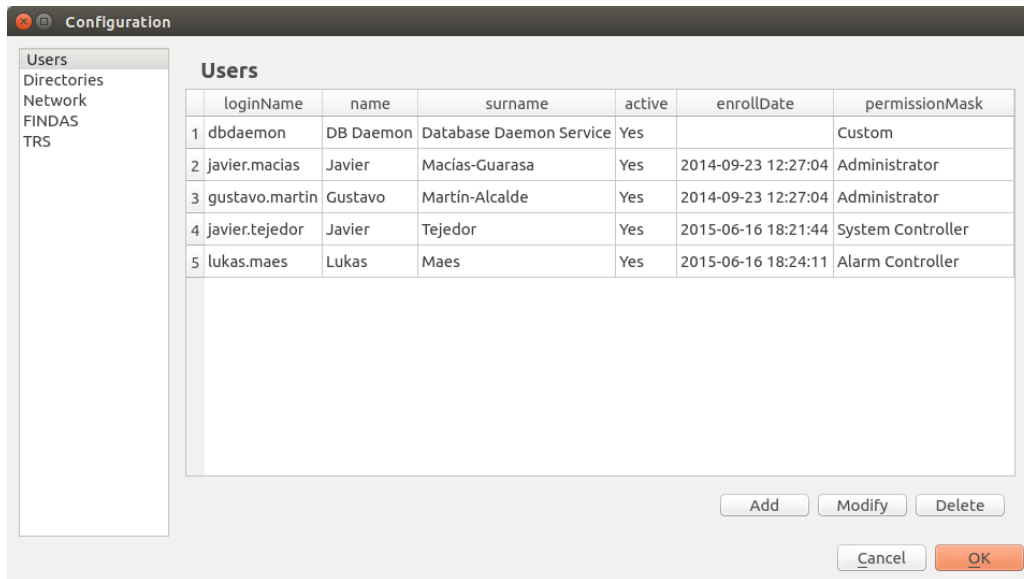


Figure A.12: User account viewer in configuration window.

When clicking add or modify, the user dialog window will appear, as shown in Figure A.13. In this window interface, the user details can be set or modified, except from the Daemon user. This account is an exception, since it cannot be deleted and only login name can be modified.

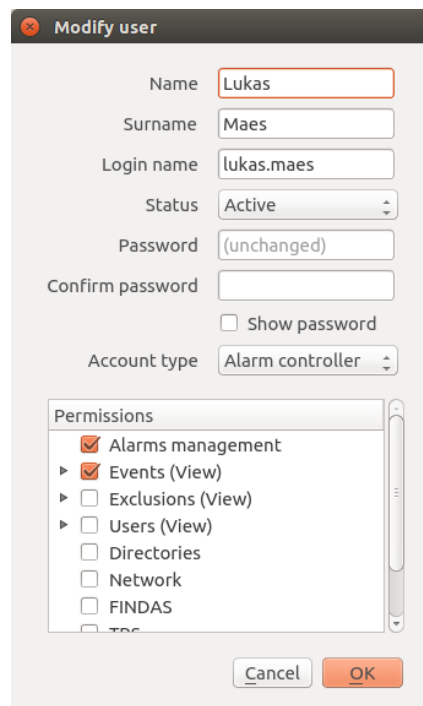


Figure A.13: User details dialog.

The second part comprises ten directories for specific use, as shown in Figure A.14.

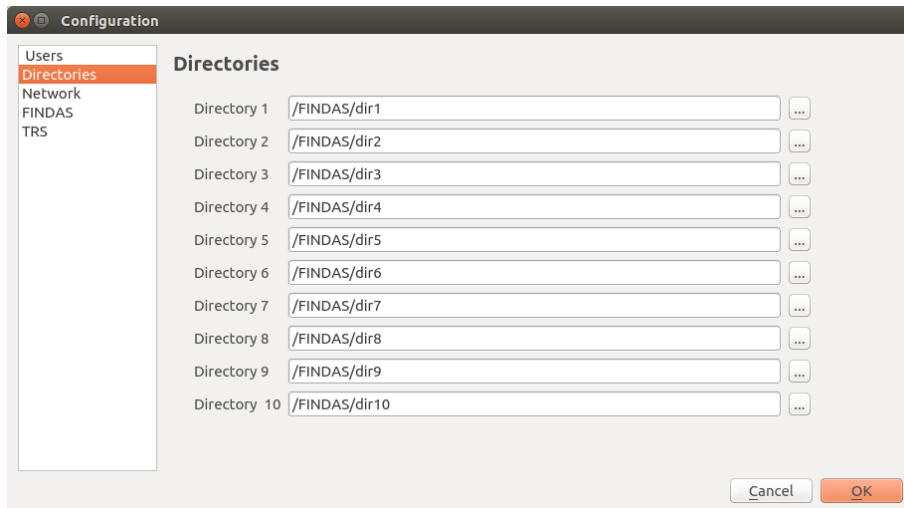


Figure A.14: Directory configuration.

The third part is the network configuration, as shown in Figure A.15, which allows to see and modify the IP and ports of TRS and FINDAS. The IP of TRS is where all the UIs are going to connect when they start running.

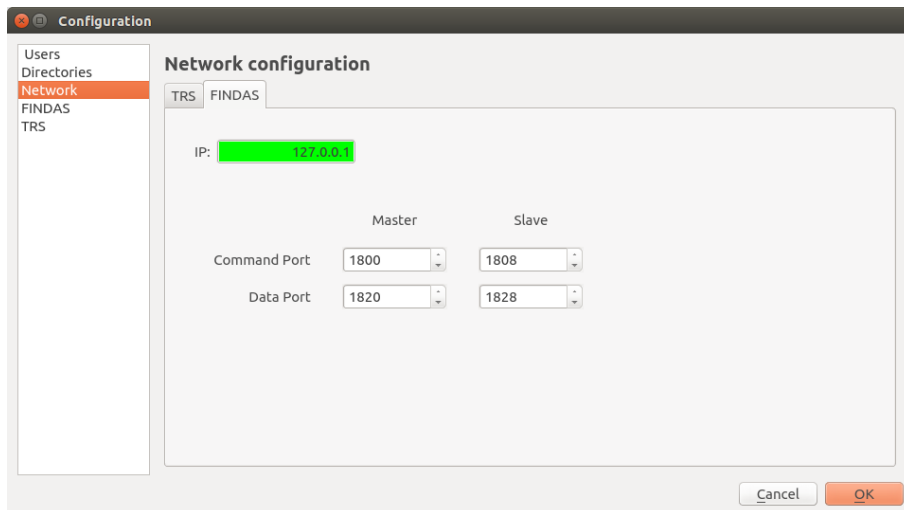


Figure A.15: Network configuration.

The last part corresponds to the FINDAS configuration, as shown in Figure A.16. This interface allows to change the technical FINDAS parameters.

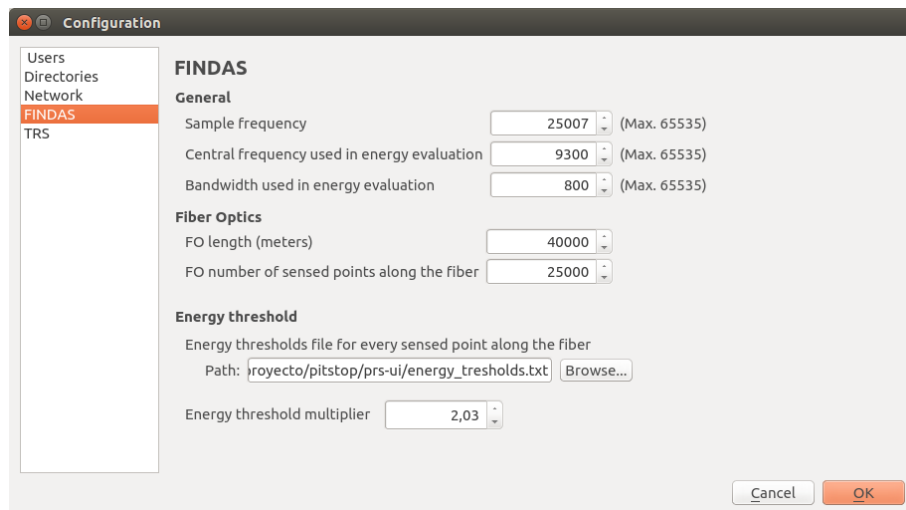


Figure A.16: FINDAS configuration.

A.3.7 Setting up the Database Daemon

The Database Daemon is a member of the main application, despite of it has a different interface, as shown in Figure A.17. This module is in charge of inserting the new incoming events from the TRS to the database. Only one Database Daemon must be opened at a time, otherwise this would lead to duplicated events. To open the Daemon, just execute the TRS-UI application and log in with the Database Daemon credentials. To assess it is working properly, ensure that the Database Daemon has established connection with the TRS and the Database by checking the status bar. To close the Daemon, click on the window close button.

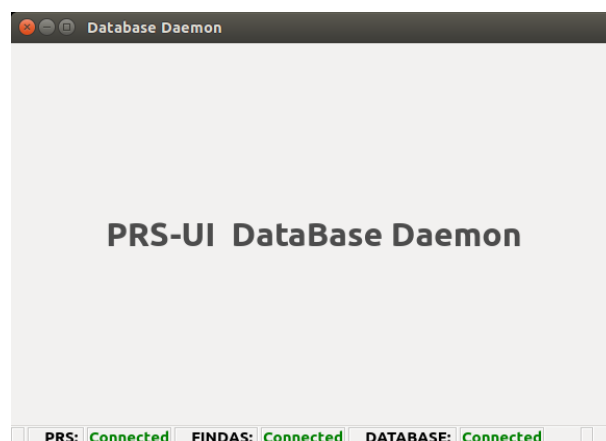


Figure A.17: Database Daemon interface.

A.3.8 Login out of the Application

Click on the main menu on the top of the window and press Log out. A confirmation window will appear. Click Accept to proceed. The user will log out from the application and the login window will appear again.

A.3.9 Exiting the Application

Click on the main menu on the top of the window and press `Exit`. A confirmation window will appear. Click `Accept` to proceed. The user will log out from the application automatically.

Universidad de Alcalá
Escuela Politécnica Superior



ESCUELA POLITECNICA
SUPERIOR



Universidad
de Alcalá