Bern University of Applied Sciences

Project 1 — ClockAlarm

Requirements (version 1.6.0)

Loïc Charrière, Samuel Gauthier

supervised by
Claude Fuhrer

June 16, 2017

Contents

Chapter 1

Introduction

This document covers the development of the ClockAlarm application, a project for the module *BTI7301 Projet 1* at the Bern University of Applied Sciences.

Chapter 2 Project Description gives an overview of the project and describes its vision, goals and general requirements.

Chapter 3 Installation explains how to install the ClockAlarm application, how to run the tests and build the api documentation.

Chapter 4 System and Context Boundaries presents the general boundaries (system and context) including the stakeholders and actors.

Chapter 5 Requirements talks about the project requirements (technical and quality requirements) such as the user stories and use cases. It also contains the domain model and the time planning.

Chapter 6 Diagrams contains all the necessary development diagrams (class and sequence diagrams).

Chapter 7 Development decisions lists all our development decisions (programs, packages, etc.) and problems that we encountered.

Appendix A Use Cases contains all the detailed use cases.

Appendix B Class Diagrams has all the class diagrams.

Appendix C Sequence Diagrams has all the sequence diagrams.

Appendix D Minutes contains all our meeting minutes

Chapter 2

Project Description

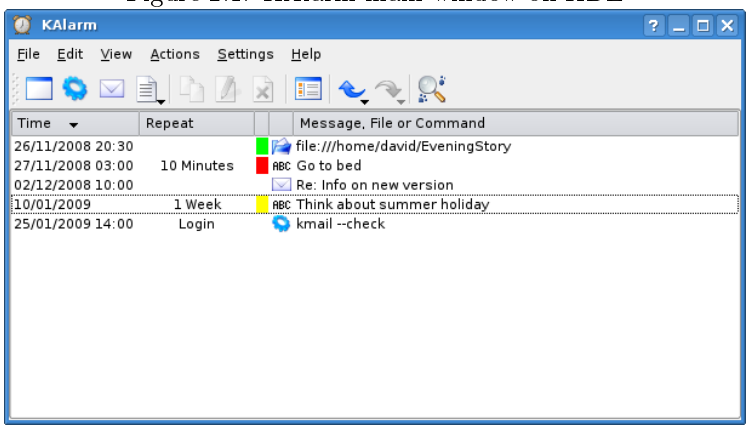
2.1 Project Vision

KAlarm on KDE

"KAlarm is a personal alarm message, command and email scheduler for Linux and UNIX. You can schedule alarm messages to pop up on the screen (with sound if desired), or you can schedule audio to play, commands to execute or emails to send." [5]

Aside from control through the graphical interface, KAlarm can also be operated using the command line.

Figure 2.1: KAlarm main window on KDE



The ClockAlarm Project

KAlarm was officially launched in 2001. The latest update is dated 13 October 2015. Despite the recent updates, the program is aging, and is only available on Linux and UNIX platforms.

The goal of the mandate is to develop a new version of kAlarm, called ClockAlarm. The new version is intended to replace kAlarm by offering the user the same services regardless of the platform used. The desired product therefore consists of a cross-platform program of persistent management of customizable reminders/alerts.

Depending on the progress of the project, it would be good to bring freshness in terms of design and user interaction compared to the version on KDE.

2.2 Project Goals

The main goal of the ClockAlarm project is to improve the user's life organization. By allowing him to define alerts, he will never miss events again.

2.3 General Requirements

- In order to reach a maximum amount of users, the application has to be cross platform. That is, run on Windows, Linux and macOS.
- To avoid data corruption in case of a system failure, the database must not be stored in a binary format.
- To provide portability, the configuration and the alerts have to be easily transferable from one computer to another.
- The user should be given the option to customise the layout of his alerts.
- Alerts should be classifiable so that the user does not lose sight of them.
- The user can schedule alerts so that he is notified at the given date and time.
- Alerts should have the possibility to be recurrent so that the user does not have to create exact same ones every time.
- The user should be able to delay alerts.
- The user should be able to snooze alerts when he does not want to be disturbed.
- The user should be able to schedule an email to be delivered at a given date and time.

Chapter 3

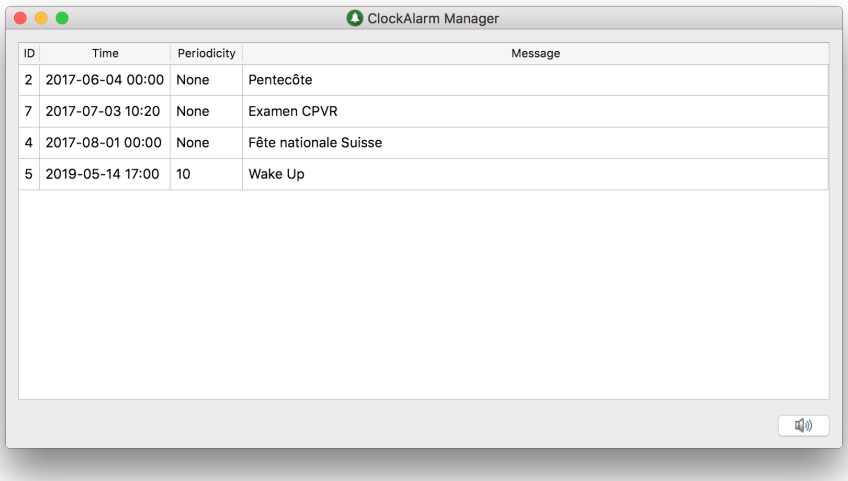
Installation

Before installing ClockAlarm some requirements have to be fulfilled: the installed Python version should at least be version 3.6 and the following packages have to be installed:

- pygame >= 1.9.3
- PyQt5 >= 5.8.2
- sip >= 4.19.2
- tinydb >= 3.2.2

This can be done with one command: "`pip3 install pygame pyqt5 sip tinydb`". The repository can be cloned: "`git clone https://github.com/BFH-BTI7301-project1/ClockAlarm.git`". The program can finally be run using: "`python3 bin/clockalarm`".

Figure 3.1: ClockAlarm main window



To run the tests these packages have to be installed:

- coverage >= 4.4.1
- pytest >= 3.0.7
- pytest-cov >= 2.5.1
- pytest-qt >= 2.1.0

- pytest-catchlog >= 1.2.2
- coveralls >= 1.1

Then the following command should be executed in the project directory: "`py.test -cov-report term -cov=. _clockalarm/_tests`". This will run the tests and output a coverage report in the terminal. To get an html report, change the argument `term` to `html`. The generated html can be found under `ClockAlarm/htmlcov/`. Open the `index.html` to get the coverage report.

To generate the documentation make sure that Sphinx (>=1.6.1) is installed. Use "`make html`" in the directory `ClockAlarm/docs/` to generate the documentation in html format. The output is located under `ClockAlarm/docs/_build/html`. It is possible to change the format, to see which are available execute only "`make`".

Chapter 4

System and Context Boundaries

4.1 Stakeholder

1. kAlarm users

   Users who previously used kAlarm on linux, and wishing to find a similar but remastered and cross-platform software.

2. Computer users

   Those who tend to forget many things.

3. Investors

   Investors in the project. For example advertisers looking for visibility in the software.

4. The GNU Project

   The GNU Project collective could be pleased to see that an updated version of kAlarm is proposed.

4.2 Actors

1. Software User (Primary Actor)

   The peoples who will use the software. They are the first concerned by the product.

2. Administrator (Primary Actor)

   Super users with special authorizations. May be able to manipulate the data and the other users.
   It is likely that the software does not need an administrator.

3. System Time (Supporting Actor)

   The service specific to the OS and providing the exact international time.

4. Messaging Software (Supporting Actor)

   Allows ClockAlarm to send Emails.

5. OS Notification Services (Supporting Actor)

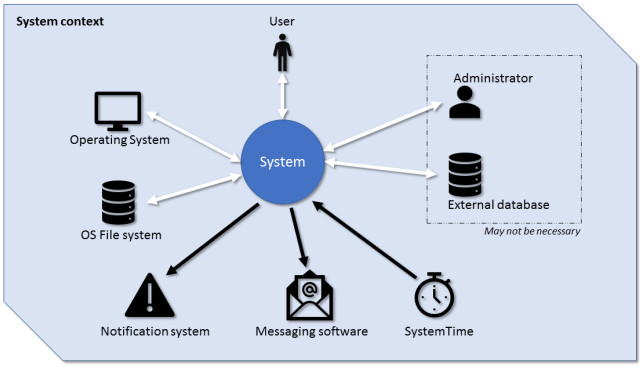
   OS specific notification center or notification service.

6. OS File System (Supporting Actor)

   Allows ClockAlarm to load configuration and alert files.

4.3 System context

Figure 4.1: System context



4.4 System boundary

In addition to the elements above, special attention should be given to the following points.

- The must-have goal is not to add functionality to the existing software kAlarm, but to create a similar and cross-platform product.
- The existence of a database and an administrator could be dropped and go out of context.
- The software is not meant to be an alternative to an agenda. It behaves like a list of tasks and reminders.

4.5 Context boundary

In addition to the logical considerations and those stated above, the following points should be taken into account.

- The software is responsible for asking the messaging software to send to emails. This means that the way emails are sent is not part of the environment.
- The software bases its alarms on the internal time. The accuracy of the time and of the time zone with respect to the geographical position of the machine is not part of the environment.

Chapter 5

Requirements

5.1 User Stories

As an [actor] I want [action] so that [achievement].

As a User

Alerts functionality

1. I want to registrate some alerts so that the software warns me when they occure
2. I want to be able to chose a different color, font and sound for every alert
3. I want to edit the existing alerts when needed
4. I want to delete the existing alerts when needed
5. I want to postpone an alarm so that I am notified again later (snooze)
6. I want to mute an alert so that I am not notified

Managing alerts

7. I want to create categories of alerts, so that I can sort my alerts
8. I want to edit my categories, in a way my alerts remain categorised
9. I want to delete categories of alerts, so that my list of categories stay concise
10. I want to assign color identity to my different categories, some that I can easily find them
11. I want to personalize the default color, font and sound of my alarms in the software configurations
12. I want to mute an alert catogory so that I am not notified by any alert in this category

Persistence

13. I want to retrieve the correct state of my alerts after turning my computer down and back on
14. I want to be alerted at any times, as soon as I am loged on my computer session

Exportability

15. I want to export my alerts so that I cannot import then on an other computer with ClockAlarm installed
16. I want to import an alerts file, so that I can retrieve previously exported alerts
17. I want to load configuration files, so that I can use predifined color and sound themes

Pivacy

18. I want my alerts to remain private, so that nobody except me knows about them

5.2 Domain model

The conceptual model including system behavior and data can be found in Figure 5.1.

5.3 Use Cases

All the Use Cases can be found under Appendix A.

5.4 Priorities and planning

Description of the various stages of development of the clock alarm product. The versions correspond to the fixed milestones. For each element, a priority is associated.
Priorities: Must Have (MH), First Piority (P1), Second Piority (P2), Nice To Have (NTH)

Table 5.1: Priorities Table

Vers.	To Implement	Prio	Status	Description
0.2.0	Time System and Notifications	MH	Done	The program is able to display a hard coded alert, based on the operating system time
0.2.1	A.5 Launch ClockAlarm manager (no login)	P2	Done	The scheduled alert is visible in a GUI before it occures
0.3.0	Persistence & Start at Runtime		Partially done	At run-time, the program is launched and starts popping alerts. The previously saved alerts are retrieved
	A.1 Program Auto Start	NTH	Not done[1]	
1.0.0	Add/Delete/Edit Alerts		Partially done	
	A.7 Setup a new alert: Simple alert	MH	Done	
	A.12 Delete an alert: Simple and Periodic alert	MH	Done	
	A.10 Edit an alert: Simple and Periodic alert	MH	Done	
	A.8 Setup a new alert: Periodic alert	P1	Done	
	A.9 Setup a new alert: E-mail sender	NTH	Not done[2]	

Continued on next page

[1] See 7.15 Unimplemented Requirements, *Auto start*
[2] See 7.15 Unimplemented Requirements, *Email alert*

Table 5.1 – *Continued from previous page*

Vers.	To Implement	Prio	Status	Description
	A.13 Delete an alert: E-mail sender	NTH	Not done[2]	
1.0.1	A.11 Edit an alert: E-mail sender	NTH	Not done[2]	
1.1.0	Default Configurations & Alerts Management		Done	The user can change the default settings. The user can import and export lists of alerts
	A.4 Edit Default Configuration	MH	Done	
	A.17 Import Alerts	P1	Done	
1.1.1	A.18 Export Alerts	P1	Done	
1.2.0	Categories		Not done[3]	The user can create, edit and delete categories
	A.14 Setup a new alert category	MH	Not done[3]	
	A.16 Delete an alert category	MH	Not done[3]	
1.2.1	A.15 Edit an alert category	P2	Not done[3]	
1.3.0	Import/Export Configuration File		Done	
	A.2 Import Configuration	MH	Done	
	A.3 Export Configuration	MH	Done	
1.4.0	Disable Notifications		Partially done	The user is able to prevent certain alerts from taking place.
	A.19 Silent mode ON	MH	Done	
	A.20 Silent mode OFF	MH	Done	
	A.21 Mute a category	P2	Not done[3]	
1.4.1	A.22 Unmute a category	P2	Not done[3]	
1.5.0	Snooze Alert	NTH	Not done[4]	The user can ask an alert to ring again in the future.
	A.23 Snooze Alert	NTH	Not done[4]	
1.6.0	Handback the Code	MH	Waiting	The deadline is 2 June 2017

5.5 Gantt Chart

Please visit `https://bfh-bti7301-project1.github.io/ClockAlarm/gantt/Overview.html` to get the full Gantt chart of the project.

5.6 Technical Requirements

The priorities are the same as in 5.4, i.e., Must Have (MH), First Priority (P1), Second Priority (P2), Nice To Have (NTH).

Table 5.2: Technical Requirements Table

ID	Status	Prio	Description
T.1	Done	MH	The application shall be written using the python programming language version 3 or greater.

Continued on next page

[3] See 7.15 Unimplemented Requirements, *Alert categories*
[4] See 7.15 Unimplemented Requirements, *Snooze alert*

ID	Status	Prio	Description
T.2	Done	MH	The application shall run on Windows, Linux and macOS.
T.3	Done	MH	The database shall be stored in a non binary format.
T.4	Not done	MH	The application shall still be running after a system reboot.
T.5	Done	MH	The configuration file shall be OS independent.

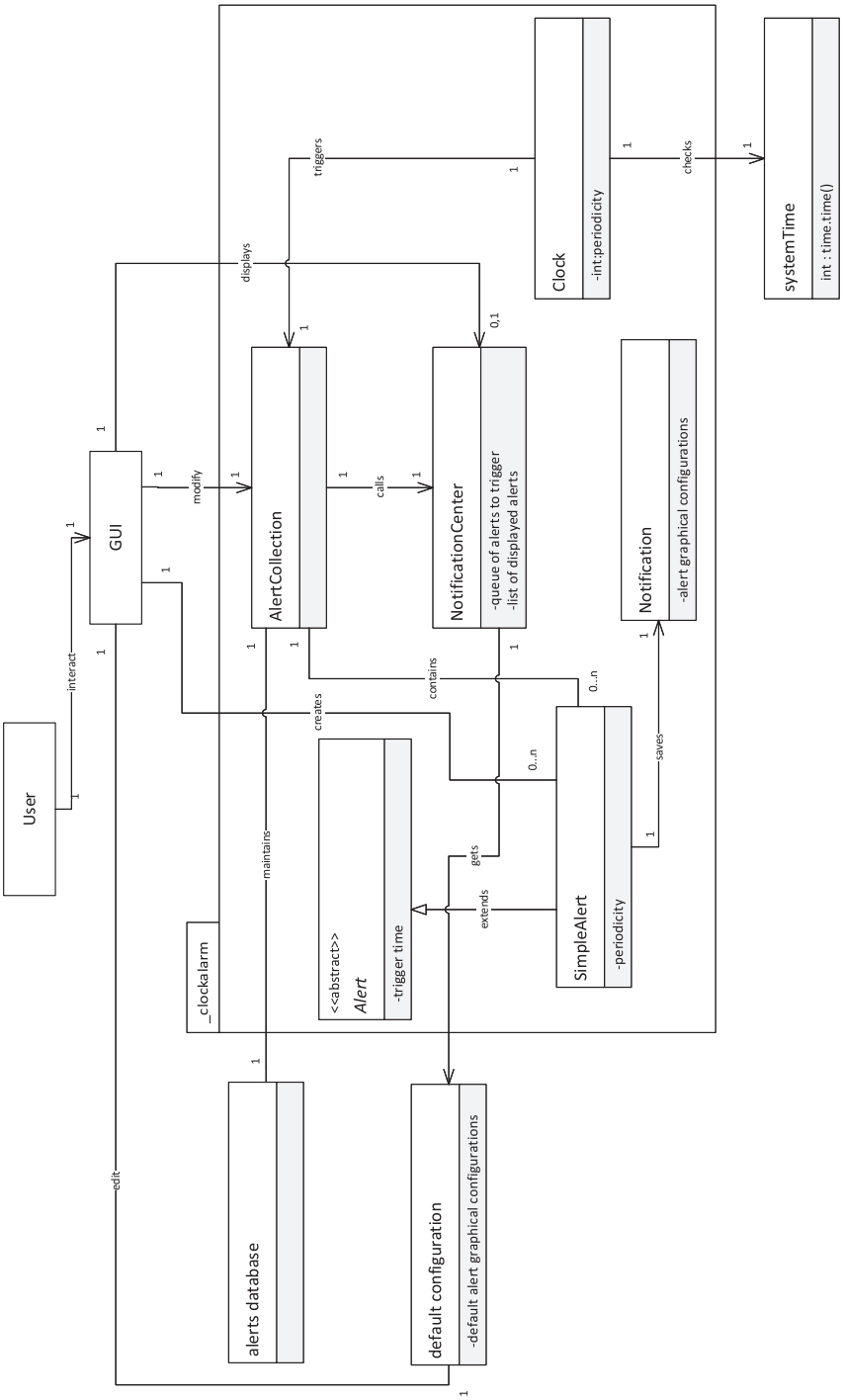
5.7 Quality Requirements

The priorities are again the same as in 5.4 and 5.6.

Table 5.3: Quality Requirements Table

ID	Status	Prio	Description
Q.1	Done	MH	The application shall display an alert on the screen within a second at the specified date and time of the alert.
Q.2	Partially Done	P1	User Interface.
Q.2.1	Done	P1	The main user interface shall contain buttons to create, edit, delete alerts.
Q.2.2	Not done	P1	The main user interface shall contain buttons to create, edit, delete categories.
Q.2.3	Done	P1	The main user interface shall contain a button to access the preferences.
Q.2.4	Done	P1	The main user interface shall contain a button to snooze the alerts appearing on the screen.
Q.2.5	Done	P1	The main user interface shall contain an area where all the alerts are listed.
Q.2.6	Partially Done	P2	The user shall be able to sort the alerts by category, by due date and alphabetically.

Figure 5.1: Domain model



Chapter 6

Diagrams

6.1 Class Diagrams

All the class diagrams are in the appendix B Class Diagrams.

6.2 Sequence Diagrams

All the sequence diagrams are in the appendix C Sequence Diagrams.

Chapter 7

Development decisions

This chapter presents all the development decisions we took during this project. It includes software choices as well as programming decisions.

The structure of the development process of this project is based on the book 'Requirements Engineering Fundamentals'[2].

7.1 Word Processor

In order to write this document, we had to choose between several word processors. We required that the file could be used with a version control system so that the documentation and code could live in the same place. Also, we didn't want to loose time if there was a need to change the layout. After discussion, we came up with the following list:

1. Microsoft Word
2. Google Docs
3. LibreOffice
4. Scribus
5. TEX

The main problem with number 1 and 3 is that the output of these programs are binary files. A built in versioning functionality exits in Microsoft Word but it adds another version control layer to the workflow. The .docx and .dt file types are in fact ZIP archives containing XML documents which can be uncompressed. They could be stored in the uncompressed state to be compatible with the chosen version control system [3]. Or they could be converted into another format. Clearly this is not convenient. Therefore we decided that they were not compatible with our first requirement stated above.
Using a Google Doc means that the document will be stored on Google's servers and not within the folder containing the source code. I thus also violates our first requirement.
Scribus is a free alternative to InDesign and doesn't produce binary files. It allows a fine-grained control of the layout, frames and styles [4].
Finally there is TEX, a typesetting system created by Knuth. It provides a low-level language that is not directly used when writing documents. Instead there exist higher level formats such as LATEX or Plain TEX (much more lower level than LATEX) which provide a large set of macros [6].
Our supervisor strongly recommended us the use of LATEX, that is why we ended up using it.

7.2 Version Control

Out of the multitude of version control systems (VCSs) which exist, the three following are mostly used [19]:

1. Git
2. Mercurial
3. Apache Subversion

All the three offer equivalent features and are equally well suited for our project [1]. We decided to use Git with GitHub mainly because we were already familiar with them. GitHub allows us to easily manage issues and milestones (an equivalent service for mercurial is Bitbucket).

7.3 Python

As two teams separately developed a similar KAlarm project, the decision was made to realize our project in python.
Python is a relatively old programming language, as it appeared in 1991. This makes the Python's libraries extremely numerous and diversified. However, many of them are no longer updated. Python has a simple easy to learn syntax. It is object-oriented and cross-platform. It was mainly disigned for prototyping, which means that it is possible to obtain a functional and efficient code in a minimum amound of time.
Python's detractors reproach it his lack of security mechanisms compared with traditional languages. Indeed, Python follows the philosophy of Linux, which count on a solid OS capable of protecting the code it uses, not a code that protects itself.

7.4 Unit Testing

Python comes with a couple of built-in modules for testing the source code:

1. Unittest
2. Doctest
3. Unittest.mock (Only for Python 3.3)

The Unittest framework is similar to the JUnit one: "It supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework." [8]

Doctests are put in comments and are formatted in form of an interactive Python session. They are usually very simple and give informative examples of the usage of the class or function to the reader. [7]

Unittest.mock allows to replace parts of the system under test with mock objects and make assertions about how they have been used. [9]

Other alternatives include: pytest, nose2 and tox.

Pytest makes it easy to write tests: only plain assert statements are used, which makes the code look very clean and neat. It can run unittest and nose test suites without any additional configuration nor modification. [14]

Nose2 is the successor of nose. Its goals are to extend unittest and make testing easier to understand. [12]

Tox is a test framework and virtualenv management tool that allows to run tests in multiple environments. [18]

We chose to use pytest because as already mentioned, the code is cleaner and more readable. Less work is also required to achieve the same result as with the other solutions. We don't use virtual environments so tox is of no use to us.

7.5 Code Coverage

There is at the moment only one maintained mature tool for measuring the code coverage in Python: Coverage.py.[1] "It monitors your program, noting which parts of the code have been executed, then analyzes the source to identify code that could have been executed but was not."[11] It also provides clean and readable annotated html reports.

7.6 Continuous Integration

We chose to use Travis CI [2], the Continuous Integration service. Every time a commit is made, the entire project is build and the tests are run. Then the report is sent to Coveralls[3]. This approach allows us to have a better development workflow.

7.7 Documentation

For documenting our project we use Sphinx [4] combined with Read The Docs[5] to host the documentation. Read The Docs builds the documentation every time there is a new commit (A git hook has to be set up beforehand for it to work).

We choose Sphinx because it is the most popular tool used in the python community and it is officially recommended (The official Python documentation is created using Sphinx).

During development we realised thanks to Sphinx that our project had cycles. See Section 7.13 for additional details.

After retrospection, this document could have been written using Sphinx. It has the same capabilities than LaTeX and can generate the document in multiple formats: pdf, html, etc..

7.8 Management

To generate the Gantt Chart we used TaskJuggler[6] because it free and cross platform compared to Microsoft Project. We considered other alternatives like GanttProject[7] or ProjectLibre[8] but we were not really pleased with the user interface and user friendliness.

7.9 System Notifications

We first tried to use the OS specific system notification API.With this solution we didn't have to care were the notifications would appear on the screen. They would always be at the correct position on every platform. There is however one drawback: the notifications can't be fully customized. Only the icon can be changed but the typeface and the sound cannot. This is why we choose to develop our own notification system, `Notification.py`, where notifications can be fully customized.

[1] Nose provides code coverage but it is deprecated and being replaced by nose2, still under development.
[2] `https://travis-ci.org/BFH-BTI7301-project1/ClockAlarm`
[3] `https://coveralls.io/github/BFH-BTI7301-project1/ClockAlarm?branch=master`
[4] `http://www.sphinx-doc.org`
[5] `https://readthedocs.org`
[6] `http://taskjuggler.org`
[7] `http://www.ganttproject.biz`
[8] `http://www.projectlibre.com`

7.10 GUI

As our application has to run on multiple platforms, the GUI framework should also support multiple platforms. The framework should also support Python 3. The Python wiki lists over 35 frameworks, many of those target Qt. We have chosen only three that were recommended by the *The Hitchhiker's Guide to Python* [15]:

1. PySide
2. PyQt
3. Tk

PySide, as PyQt, provides a binding to the Qt framework. The main difference between the two is the license: PySide uses the Lesser General Public License (LGPL) whereas PyQt uses the General Public License (GPL). Which means that source code licensed under the GPL must be opensource. On the other hand the LGPL is more suited for closed source projects. In addition PyQt supports Python 3. Tk provides bindings to the Tk framework and is licensed under a Berkeley Software Distribution (BSD) license.

We chose the PyQt framework out of personal preference. All three frameworks are equally well suited for a project like this one.

7.11 Database

We planned to have three alert types:

1. Simple non recurring text alerts
2. Recurring simple alerts
3. Alerts that send emails

All three types are independent of each other, that is, there is no logical link between them. The only "link" that they have in common is that they are extending the abstract class `Alert`. Therefore we don't need a relational database, only a very lightweight one.

The requirement for the file storing the database is that it should be humanly readable, i.e., it should be in a non binary format. We came up with four possibilities for the file format:

- XML
- JSON
- YAML
- Plaintext

BaseX[9] handles both JSON and XML databases. Another alternative is TinyDB [10], it handles only databases in the JSON format. PyXAML [11] handles databases stored in the YAML format. And for a plaintext database, i.e., a simple text file, everything has to be handled with the Python `open(...)` and `write(...)` commands.

We choose to use TinyDB and the JSON file format because our supervisor encouraged us to do so and because the syntax is very simple and follows the Pythonic way.

[9] For python it is the BaseXClient: `http://basex.org`
[10] `http://tinydb.readthedocs.io`
[11] `http://pyyaml.org`

7.12 Sound

We had an issue on macOS and linux when using PyQt's `QSound` class. `QSound` uses the default OS specific sound library to play the sound (NSSound for macOS and NAS for X11 used on linux). The `wav` file only played on Windows but not on macOS nor linux.

An easier and much cleaner alternative was found using pygame's `mixer`.

7.13 Cycles

As already mentioned in Section 7.7 Sphinx made us aware of cycles in our code:

1. `SimpleAlert` ⇝ `main` ⇝ `AlertCollction` ⇝ `SimpleAlert`

   The `SimpleAlert` class was directly accessing the `AlertCollection` inside of `main` to delete `SimpleAlerts` or update periodic `SimpleAlerts`. The code doing the `SimpleAlert` update and delete has been moved to the `check_timers(self, trig_time)` method inside of `AlertCollection`.

2. `AlertCollection` ⇝ `main` ⇝ `AlertCollection`

   The `display(self)` method in `AlertCollection` triggered the actualization method of the `UI.AlertListWidget` from the `App` in `main`. The method now calls the parent and actualizes the `AlertListWidgets`.

7.14 Metaclass

A Metaclass issue was discovered when we tried to apply the new python 3 style for abstract classes. In our case, the `Alert` class is an abstract class that extends `QObject`. As `QObject` has its metaclass, pyhton could not resolve which metaclass `Alert` should inherit. Therefore a special metaclass `AlertMeta` had to be created. It inherits `QObject`'s metaclass and is abstract. The `Alert` class then has the metaclass `AlertMeta` and extends `QObject`. [13, 16]

The difference is shown in Listings 7.1 and 7.2.

Listing 7.1: Abstract class, Python 2.7

```
...
class Alert(PyQt5.QtCore.QObject):
    __metaclass__ = abc.ABCMeta
    ...
```

Listing 7.2: Abstract class, Python 3.6

```
...
class AlertMeta(type(PyQt5.QtCore.QObject), abc.ABC):
    pass

class Alert(pyqt.QObject, metaclass=AlertMeta):
    ...
```

7.15 Unimplemented Requirements

Due to the time constraint the following requirements could not be fulfilled:

- **Alert categories:** A.14 Setup a new alert category, A.16 Delete an alert category, A.15 Edit an alert category, A.21 Mute a category, A.22 Unmute a category, Q.2.2 The main user interface shall contain buttons to create, edit, delete categories, Q.2.6 The user shall be able to sort the alerts by category, by due date and alphabetically.

- **Email alert:** A.9 Setup a new alert: E-mail sender, A.13 Delete an alert: E-mail sender, A.11 Edit an alert: E-mail sender. These requirements do not really make sense because they allready exist in Outlook[12] and Thunderbird[13]
- **Snooze alert:** A.23 Snooze Alert
- **Auto start:** A.1 Program Auto Start, T.4 The application shall still be running after a system reboot.

SimpleAlert class

The existence of a `SimpleAlert` and `Alert` class can seem to be unoptimized at first glance. The two classes could have been merged together. But in our initial development decisions, we planned to add another class extending `Alert`, namely `EmailAlert`. We decided to leave the design like it is now, so that it is easier to create a custom alert class in the future.

[12] `https://support.office.com/en-us/article/Delay-or-schedule-sending-email-messages-026af69f-c287-490a-a72f-6c65793744ba`

[13] `https://addons.mozilla.org/en-US/thunderbird/addon/send-later-3/?src=search`

Chapter 8

Conclusion

As a reminder, the general requirements were that the application should be cross-platform, the database stored in a non binary format, the configuration and alerts easily transferable, the alerts customisable, classifiable into categories, recurrent, delayable, snoozable. Moreover emails should be able to be scheduled and sent at a given time.

We did not implement categories nor email that can be scheduled. The former because of time constraints and the latter because it already exists as a plugin in Mozilla Thunderbird and Outlook.

Looking back on choosing Python to develop this project we found out that with this programming language, it was easy to quickly get to a reasonable result. Many packages in the Python Package Index provide solutions to encountered problems. Such as in our case, the sound problem[1]. One drawback with so many existing packages is that a lot are not actively maintained. This leads to problems when using a newer version of Python.

Our biggest development challenge was Qt. We found it difficult to work with because of its nature: slots and connections with signals have to be set up, two subjects that are not easily understandable. Most of the time the documentation for Python is not very helpful. Generally we were guided to the original C++ documentation which provided a far better help.

The actual state of the project allows users to extend the `Alert` class so that new alert types can be implemented. In our view, adding categories would be the next most important feature. Improving the console support and providing a better installation process (setup.py) could lead this application to get into the Python Package Index.

Our final impression is that it is clearly feasible to develop a better, free and open source alternative to KAlarm that works on all major platforms.

[1] See Section 7.12 for more information

Appendices

Appendix A

Use Cases

A.1 Program Auto Start

Program starts at OS start.

Scope

A computer running any common operating system (Windows, macOS, Linux).

Primary actor

User or Administrator

Precondition

The computer is on. The ClockAlarm application is correctly installed.

Postcondition

ClockAlarm is running and the manager window is hidden in the system tray. The user is logged on his personnal ClockAlarm session.

Main success scenario

1. The system launches the ClockAlarm application.
2. The application starts and loads configurations and alarms from the user's personal files folder.
3. The manager window is hidden in the system tray.

Extension

2. Configurations or alerts can not be loaded
    i The application creates a new default configuration file and an empty alert file.

A.2 Import Configuration

Import a ClockAlarm application configuration from an external file.

Scope

ClockAlarm application configuration phase.

Primary actor

User

Precondition

The application has to be started.

Postcondition

The application configuration is replaced with the one in the given file.

Main success scenario

1. The User selects a file to be imported.
2. The application checks if the file is a valid configuration file.
3. The application replaces the configuration with the one specified in the file.
4. The application saves the new configuration.
5. The application notifies the User that the configuration has been imported successfully.

Extension

2. The application detects that the file is invalid.
   i The application notifies the User that the file is invalid.
   ii The User chooses if he wants to select another file.

A.3 Export Configuration

Export a ClockAlarm application configuration to a file.

Scope

ClockAlarm application configuration phase.

Primary actor

User

Precondition

The application has to be started.

Postcondition

The application configuration is exported to a file whose path was chosen by the User.

Main success scenario

1. The User is asked where he wants to save the file.
2. The application exports the configuration to the file.
3. The application notifies the User that the configuration has been exported successfully.

Extension

1. The application detects that it has not enough privileges to write to the specified folder.
   i The application notifies the User that it has not enough privileges to write to the specified folder.
   ii The User chooses if he wants to select another folder.

A.4 Edit Default Configuration

Edit the default configuration of the ClockAlarm application.

Scope

ClockAlarm application configuration phase.

Primary actor

User

Precondition

The application has to be started.

Postcondition

The application configuration is edited.

Main success scenario

1. The User edits the configuration
2. The application checks if the User entered configuration is valid
3. The application saves the configuration.

Extension

2. The application detects that the User entered configuration is invalid
   i The application notifies the User that the entered configuration is invalid.
   ii The User chooses if he wants to edit the configuration.

A.5 Launch ClockAlarm manager (no login)

In the case of an application without shared database and user accounts.

Scope

A computer running any common operating system (Windows, Mac OS, Linux).

Primary actor

User or Administrator

Precondition

The computer is on and the user is logged on his computer user session. The ClockAlarm application is correctly installed.

Postcondition

ClockAlarm is running and the manager window is open. The user is logged on his personnal ClockAlarm session.

Main success scenario

1. The user launches the ClockAlarm application.
2. The application starts and loads configurations and alarms from the user's personal files folder.
3. The manager window opens and displays the main window.

Extension

2 The application il allready running in background.

i Goto point 3.

2 Configurations or alerts can not be loaded (e.g. first use of the program).

i The application creates a new default configuration file and an empty alert file.

ii Goto point 2.

A.6 Launch ClockAlarm manager (with login)

In the case of an application with a shared database and user accounts.
This solution is unlikely to be kept.
For this reason, not all scenarios, especially those related to the connection with the server, will be treated here.

Scope

A computer running any common operating system (Windows, Mac OS, Linux).

Primary actor

User or Administrator

Precondition

The computer is on and the user is logged on his computer user session. The ClockAlarm application is correctly installed.
The user has a registered user account on the server.

Postcondition

ClockAlarm is running and the manager window is open. The user is logged on his personnal ClockAlarm session.

Main success scenario

1. The user launches the ClockAlarm application.
2. The application starts.
3. The user is asked to enter his credentials and the program tries to check these.
4. The application is connected to the server.
5. The application loads configurations and alarms from the server database.
6. The manager window opens and displays the main window.

Extension

2 The application il allready running in background.

i Goto point 6.

3 The credentials are incorrect.

i The connection to the server is denied.
ii The user is asked to give his credentials again.
iii Goto point 3.

3 The user isn't registered.

i The connection to the server is denied.
ii The user is redirected to a page to an account creation page.
iii The user creates a new account on the server.
iv Goto point 3.

5 Configurations or alerts can not be loaded (e.g. first use of the account).

i The application creates and upload a new default configuration file and an empty alert file.
ii Goto point 2.

A.7 Setup a new alert: Simple alert

A simple alert displays a message to the user at the requested time.

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.

Postcondition

A new simple alert is added to the alert list and is ready to warn the user at the right time.

Main success scenario

1. The user browse the menu and selects "File > New Alert > Simple Alert". The simple alert creation window is displayed.
2. The user sets his alert. The message to be displayed as well as the time of display are mandatory.
   The user can, if desired, assign a category to the alert, or set the color, sound and font of the alert.
3. The user validates his alert. The dialog box closes and the manager window is updated.

Extension

3 The user does not complete the category and settings.

  i The default settings are used and the alert does not belong to any category.

3 The user complete the category, but not the settings.

  i The parameters of the category are used.

3 One of the parameter is invalid (for example, the time entered is earlier than the current time).

  i The user is asked to check his entries. Back to point 2.

A.8 Setup a new alert: Periodic alert

A periodic alert displays a message to the user at the requested periodic time.

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.

Postcondition

A new periodic alert is added to the alert list and is ready to warn the user whenever it occures.

Main success scenario

1. The user browse the menu and selects "File > New Alert > Periodic Alert". The periodic alert creation window is displayed.
2. The user sets his alert. The message to be displayed, the time of display as well as the periodicity are mandatory.
   The user can, if desired, assign a category to the alert, or set the color, sound and font of the alert.
3. The user validates his alert. The dialog box closes and the manager window is updated.

Extension

3 The user does not complete the category and settings.

  i The default settings are used and the alert does not belong to any category.

3 The user complete the category, but not the settings.

  i The parameters of the category are used.

3 One of the parameter is invalid (for example, the time entered is earlier than the current time).

  i The user is asked to check his entries. Back to point 2.

A.9 Setup a new alert: E-mail sender

An e-mail sender send an e-mail at the requested time.

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.

Postcondition

An e-mail is configured and ready to be sent at the scheduled time.

Main success scenario

1. The user browse the menu and selects "File > New Alert > E-mail Sender". The e-mail sender creation window is displayed.
2. The user sets his e-mail. He enters the recipient, subject, and body of the message. He also sets the time of sending.
   The user can, if he wants to, enter the path to an attachment.
3. The user validates his e-mail. The dialog box closes and the manager window is updated.

Extension

3 One of the parameter is invalid (time entered earlier than current time, invalid recipient e-mail address, empty object).

  i The user is asked to check his entries. Back to point 2.

A.10 Edit an alert: Simple and Periodic alert

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.
A simple(resp. periodic) alert is already set.

Postcondition

The selected alert is updated and ready to alert the user at the chosen time.

Main success scenario

1. The user selects the alert he wishes to modify.
2. The user browses the menu and selects "File > Edit Alert". The simple (resp. periodic) alert edition window is displayed.
3. The current settings are displayed. The user edits his alert. The message to be displayed as well as the time of display (and the periodicity for the periodic alert) are mandatory.
   The user can, if desired, assign or reassign a category, color, sound and font to the alert.
4. The user validates his modifications. The edition window closes and the manager window is updated.

Extension

2 No alert is selected.

  i Nothing happens.

3 The user changes the alert category.

  i The user is asked if he also wants to use the parameters of the category.

4 The user does not complete the category and settings.

  i The default settings are used and the alert does not belong to any category.

4 The user complete the category, but not the settings.

  i The parameters of the category are used.

4 One of the parameter is invalid (for example, the time entered is earlier than the current time).

  i The user is asked to check his entries. Back to point 3.

A.11 Edit an alert: E-mail sender

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.
An e-mail sender is already set.

Postcondition

The selected e-mail sender is updated and ready to send an e-mail at the scheduled time.

Main success scenario

1. The user selects the e-mail sender he wishes to modify.
2. he user browses the menu and selects " File > Edit Alert". The e-mail sender edition window is displayed.
3. The current settings are displayed. The user edits his e-mail. He can edit the recipient, subject, and body of the message. He can also edit the time of sending and the path to an attachment.
4. The user validates his modifications. The edition window closes and the manager window is updated.

Extension

4 One of the parameters is invalid (time entered earlier than current time, invalid recipient e-mail address, empty object).

i The user is asked to check his entries. Back to point 3.

A.12 Delete an alert: Simple and Periodic alert

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.
A simple(resp. periodic) alert is already set.

Postcondition

The selected alert is deleted won't alert the user in the futur.

Main success scenario

1. The user selects the alert he wishes to delete.
2. The user browses the menu and selects "File > Delete Alert".
3. A dialog asks the user to validate his decision.
4. The alert is deleted. The dialog box closes and the manager window is updated.

Extension

2 No alert is selected.

i Nothing happens.

3 The user refuses or exits the dialog box.

i The alert isn't deleted and the dialog box closes.

A.13 Delete an alert: E-mail sender

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.
An e-mail sender is already set.

Postcondition

The selected e-mail sender is deleted and won't send any e-mail in the futur.

Main success scenario

1. The user selects the e-mail sender he wishes to delete.
2. The user browses the menu and selects "File > Delete Alert".
3. A dialog asks the user to validate his decision.
4. The email-sender is deleted. The dialog box closes and the manager window is updated.

Extension

2 No alert is selected.

i Nothing happens.

3 The user refuses or exits the dialog box.

i The email-sender isn't deleted and the dialog box closes.

A.14 Setup a new alert category

Define a category to sort alerts.

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.

Postcondition

A new alert category is created. It defines the parameters of the alerts belonging to it.

Main success scenario

1. The user browses the menu and selects "Edit > Categories". The categories manager window is displayed.
2. The user clicks on the "New category" button. The add category dialog is displayed.
3. The user defines the new category. The name is mandatory.
   The user can define a font and a color for the alerts and the sound to be used.
4. The user validates the category. The category is created
5. The categories manager window is updated.
6. The creation window closes.

Extension

4 A non-mandatory field is not filled.

   i The default settings are used.

   ii The category is created.

   iii The categories manager window is updated and the creation window closes.

4 One of the parameter is invalid.

   i The user is asked to check his entries. Back to point 3.

A.15 Edit an alert category

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.
A category allready exist.

Postcondition

The selected category is updated as wanted. It defines the parameters of the alerts belonging to it.
If alerts are defined by this category, they must be updated too.

Main success scenario

1. The user browse the menu and selects "Edit > Categories". The categories manager window is displayed.
2. The user selects from the list the category he wishes to modify.
3. The user clicks on the "Edit category" button. The edit category dialog is displayed.
4. The user update the selected category. The name is mandatory.
   The user can define or redifine a font and a color for the alerts and the sound to be used.
5. The user validates the category. The category is updated, as well as all the alerts it defines.
6. The categories manager window is updated.
7. The creation window closes.

Extension

5 A non-mandatory field is not filled.

- i The initial setting of this parameter is used.
- ii The category is updated.
- iii The categories manager window is updated and the dialog box closes.

5 One of the parameter is invalid.

- i The user is asked to check his entries. Back to point 4.

A.16 Delete an alert category

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.
A category allready exist.

Postcondition

The selected category is deleted.
The alerts previously defined in this category resume the default configuration.

Main success scenario

1. The user browse the menu and selects "Edit > Categories". The categories manager window is displayed.
2. The user selects from the list the category he wishes to delete.
3. The user clicks on the "Delete category" button.
4. The user is informed that the alerts previously defined by this category will now have the default configuration. The user must accept in order to delete the category.
5. The category is deleted. The related alerts are updated. The category manager window is updated.
6. The dialog box closes.

Extension

4 The user refuses or closes the dialog box.

i The category is not deleted.

ii The dialog box closes.

A.17 Import Alerts

Load ClockAlarm alerts from an external file.

Scope

ClockAlarm application user interface.

Primary actor

User

Precondition

The application has to be started.

Postcondition

New alerts from the given file are added to the alert list.

Main success scenario

1. The User selects a file to import.
2. The application checks if the file is a valid alert file.
3. The application asks the User which alerts he wants to import.
4. The application adds the alerts to the User alert list.
5. The application saves the alert list.
6. The application notifies the user that the alerts have been imported successfully.

Extension

2. The application detects that the file is invalid.
   i The application notifies the user that the file is invalid.
   ii The user chooses if he wants to select another file.
3. The application detects that the User has chosen no alerts.
   i The application notifies the User that no alerts were chosen to import.
   ii The User chooses if he wants to select alerts to be imported.

A.18 Export Alerts

Export ClockAlarm alerts to a file.

Scope

ClockAlarm application user interface.

Primary actor

User

Precondition

The application has to be started.

Postcondition

Alerts are saved to a file whose path was chosen by the User.

Main success scenario

1. The User is asked where he wants to save the file.
2. The User selects the alerts he wants to export.
3. The application exports the alerts to the file.
4. The application notifies the User that the alerts have been exported successfully.

Extension

1. The application detects that it has not enough privileges to write to the specified folder.
   i The application notifies the User that it has not enough privileges to write to the specified folder.
   ii The User chooses if he wants to select another folder.
2. The application detects that the User has chosen no alerts.
   i The application notifies the User that no alerts were chosen to export.
   ii The User chooses if he wants to select alerts to be exported.

A.19 Silent mode ON

Mute all the alertes.

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.
Silent mode is disabled.

Postcondition

The system is in silent mode. The user will no longer be disturbed by alerts.

Main success scenario

1. The user browses the menu and selects "Tool > Silent mode".
2. In the displayed drop-down menu, the user selects the duration of the mute time.
3. All alerts are silent for the requested duration.

Note

Email alerts can't be muted.

A.20 Silent mode OFF

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.
Silent mode is enabled.

Postcondition

The system is in nomral mode. The user will be alerted on time.

Main success scenario

1. The user browses the menu and selects "Tool > Silent mode".
2. All alarms, except muted ones, are activated again.

A.21 Mute a category

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.
There is a parameterized and unmuted category.

Postcondition

The category is muted. The user will no longer be disturbed by alerts from this category.

Main success scenario

1. The user browse the menu and selects "Edit > Categories". The categories manager window is displayed.
2. The user selects from the list the category he wishes to mute.
3. The user browses the menu and selects "Tool > Mute".
4. In the displayed drop-down menu, the user selects the duration of the mute time.
5. All alerts from this category are silent for the requested duration.

Extension

3 No category is selected.

i Option "Tool > Mutes" is not available.

A.22 Unmute a category

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.
There is a parameterized and muted category.

Postcondition

The category is no longer muted. The user will again be alerted on time.

Main success scenario

1. The user browse the menu and selects "Edit > Categories". The categories manager window is displayed.
2. The user selects from the list the muted category he wishes to unmute.
3. The user browses the menu and selects "Tool > Unmute".
4. All alerts from this category are no longer silent and will alert the user on time.

Extension

3 No category is selected.

i Option "Tool > Unmute is not available.

A.23 Snooze Alert

Snooze the alert on the screen.

Scope

ClockAlarm application user interface

Primary actor

User

Precondition

The application has to be started. A default configuration file exists.

Postcondition

The alert is snoozed for a certain amount of time.

Main success scenario

1. The User snoozes the alert.
2. The application changes the alert time of the specific alert and adds an amount of time defined by the configuration.

Extension

2. The application can't find the configuration.

i The application notifies the User that the configuration has not been found.

ii The application uses the time defined in the default configuration.

Appendix B

Class Diagrams

For each diagram are indicated the classes, variables and methods useful for the action described.

1. B.1 Add Simple Alert Describes the addition of an alert to the system.
2. B.2 Edit Simple Alert Describes the modification of an alert.
3. B.3 Delete Simple Alert Describes the deletion of an alert.
4. B.4 Export Alert Describes how to export a JSON alert file to the file system.
5. B.5 Import Alert Describes how to import a JSON alert file from the file system.
6. B.6 Silent Mode ON / OFF Describes switching to silent mode.
7. B.7 Simple Alert Notification Describes the mechanism for triggering and displaying notifications.
8. B.8 Exit Describes the closing of the program.

Figure B.1: Add Simple Alert

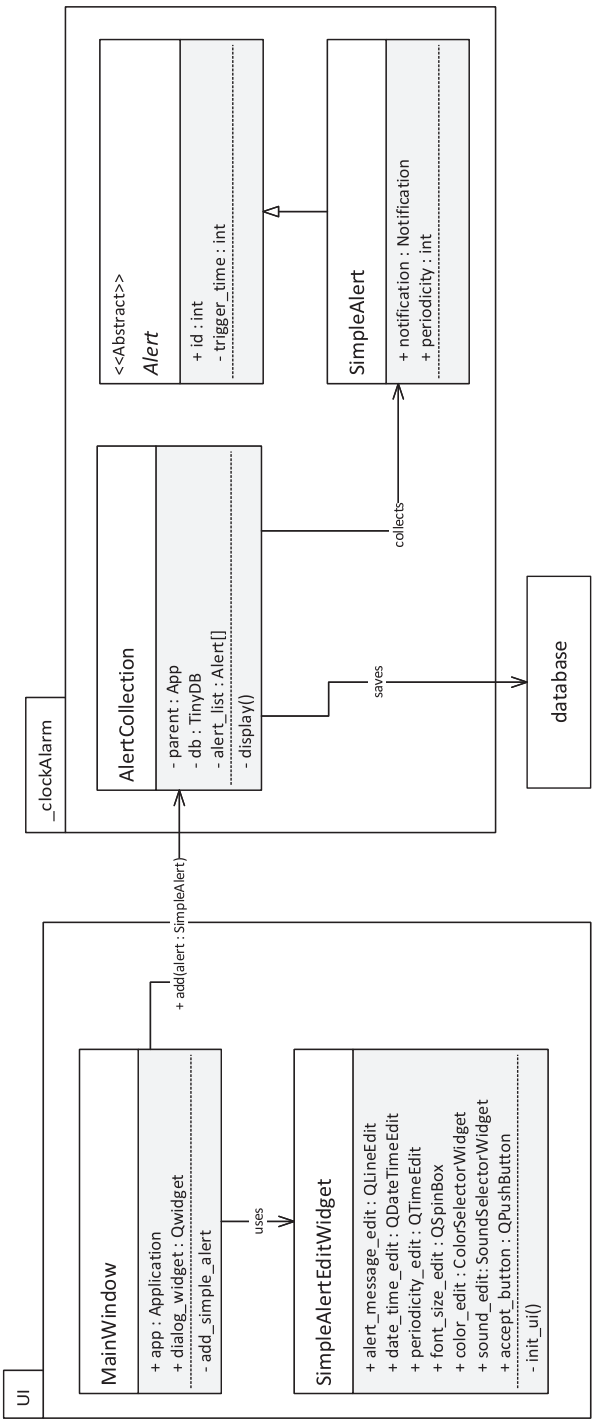


Figure B.2: Edit Simple Alert

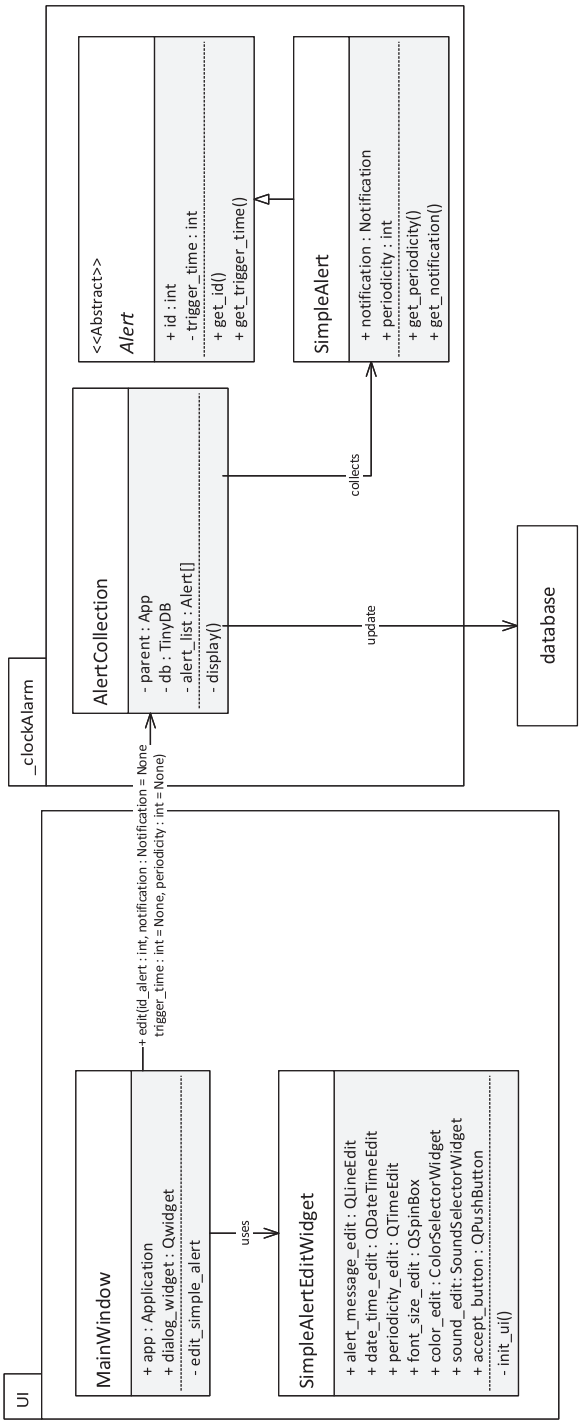


Figure B.3: Delete Simple Alert

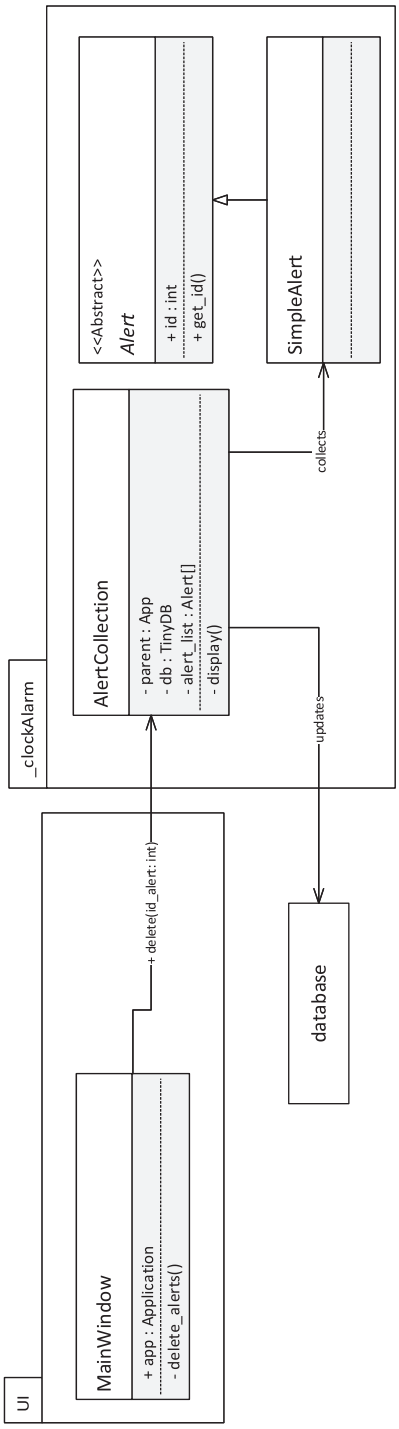


Figure B.4: Export Alert

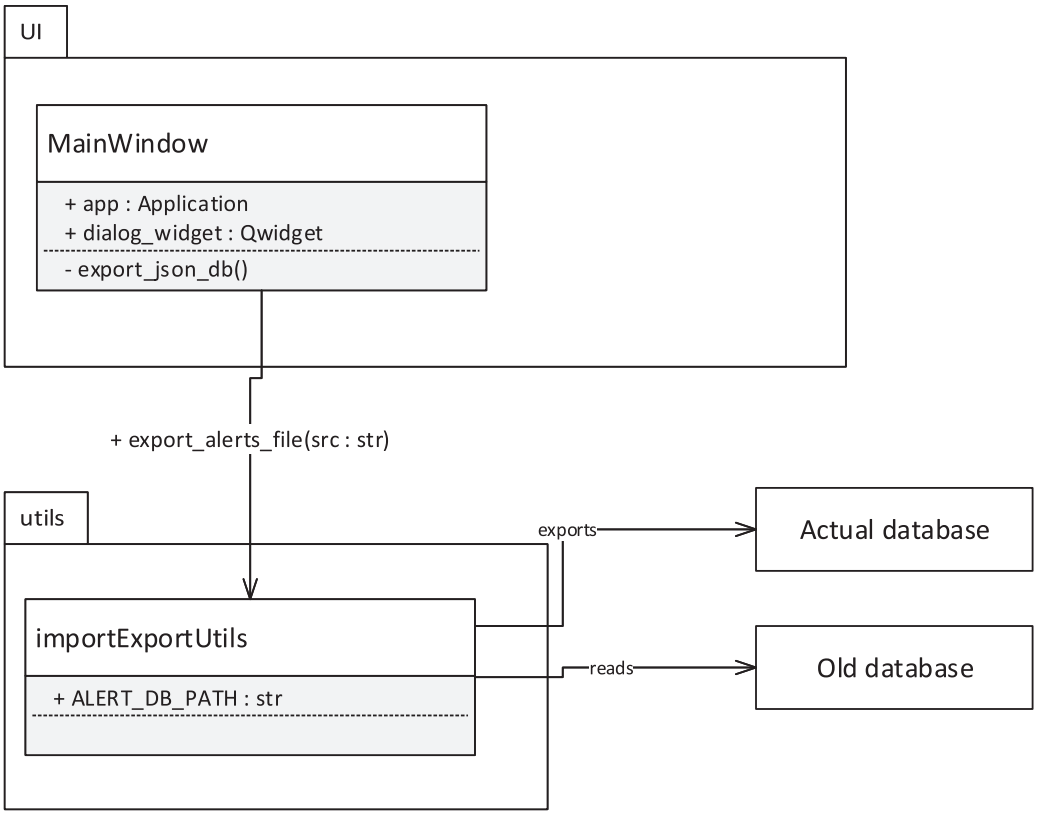


Figure B.5: Import Alert

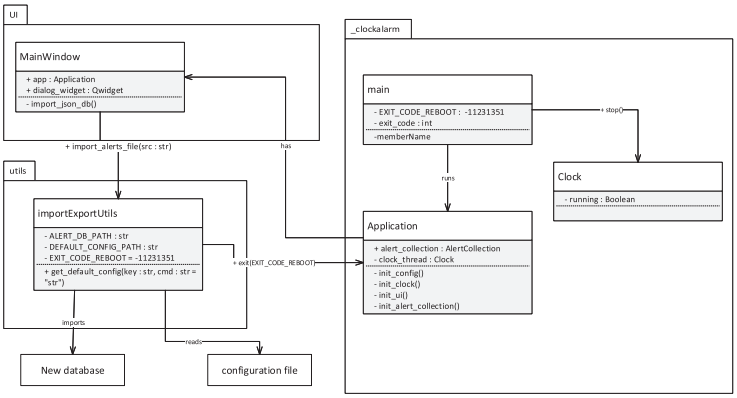


Figure B.6: Silent Mode ON / OFF

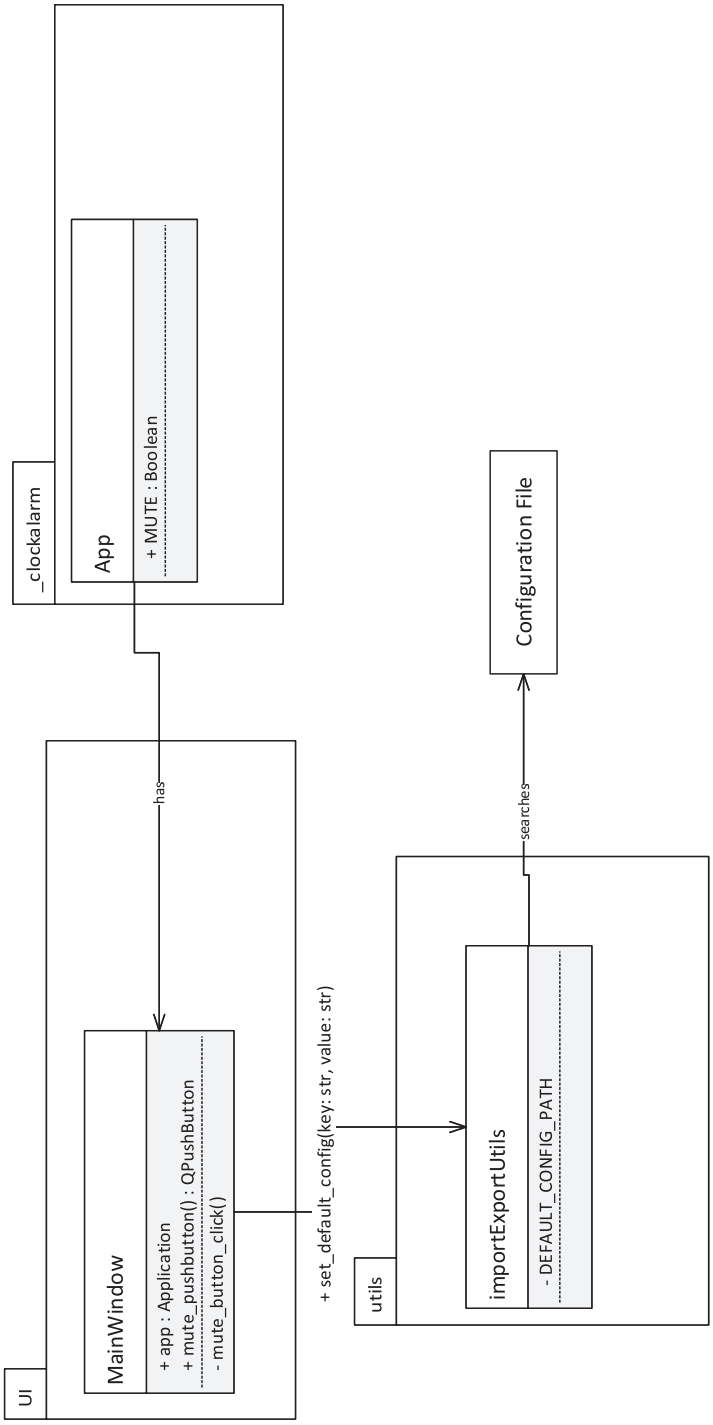


Figure B.7: Simple Alert Notification

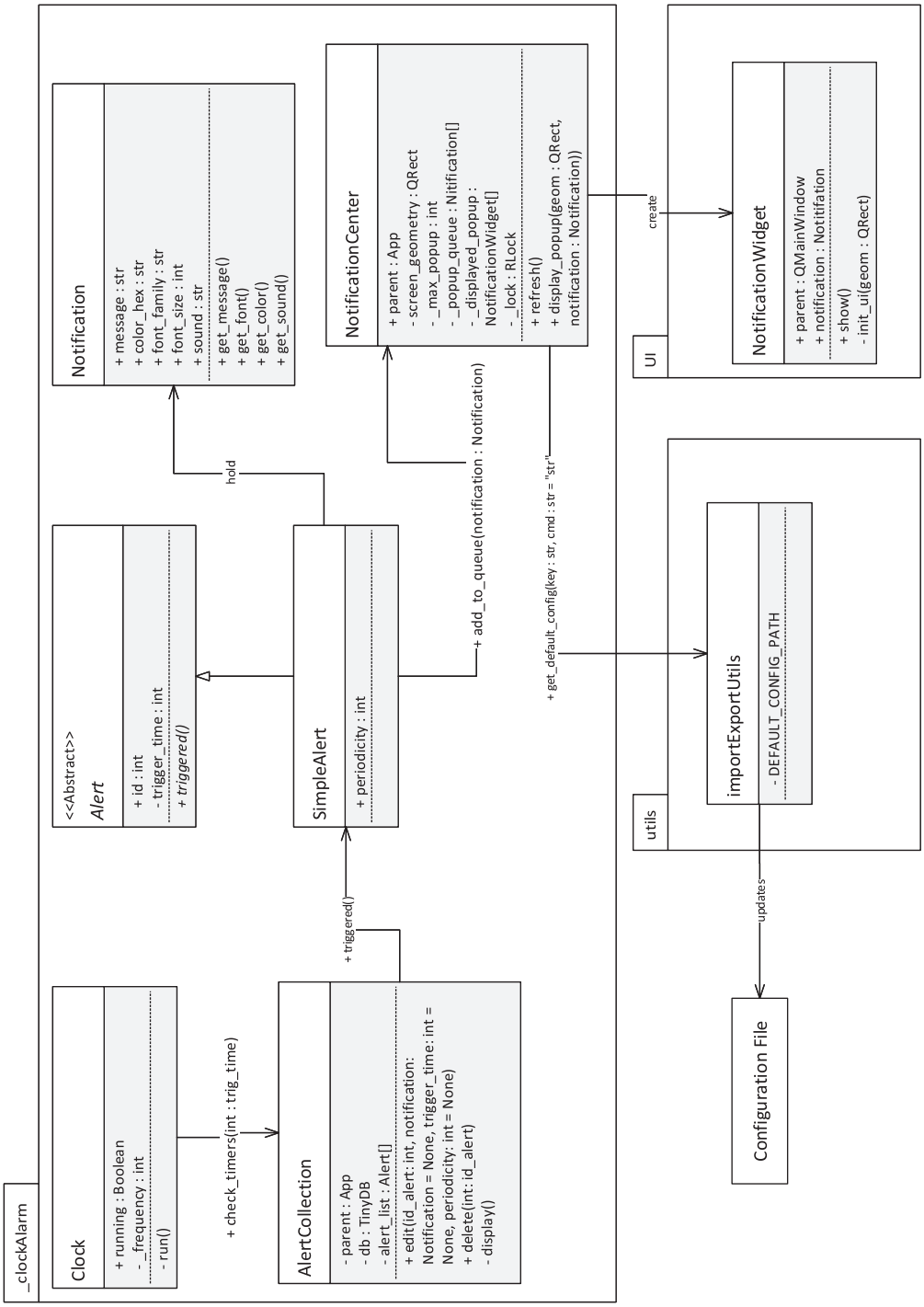
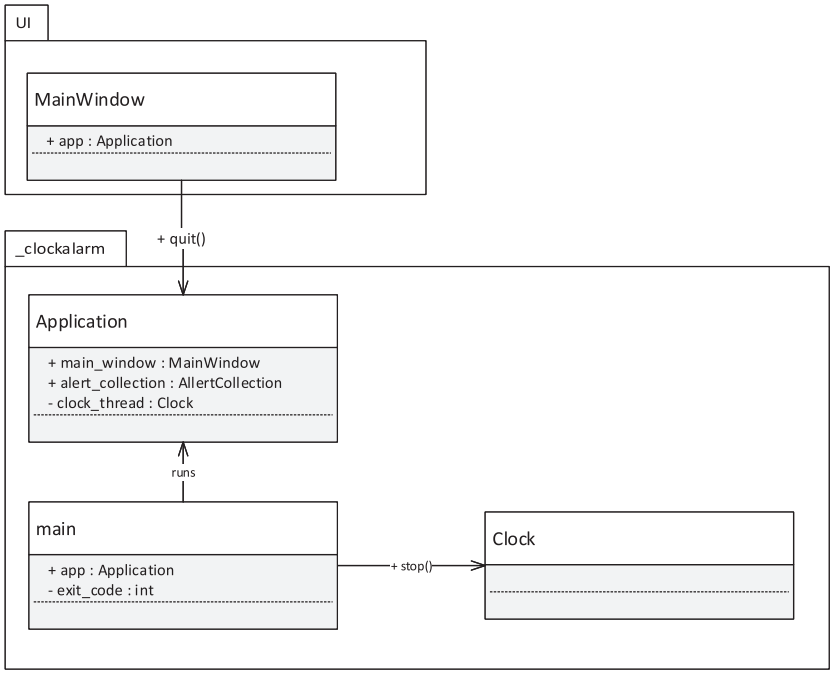


Figure B.8: Exit



Appendix C

Sequence Diagrams

Each of the following sequence diagrams describes the interactions between the different components for a given action.

1. C.1 Add Simple Alert Details the addition of an alert to the system.
2. C.2 Edit Simple Alert Details the modification of an alert.
3. C.3 Delete Simple Alert Details the deletion of an alert.
4. C.4 Export Alert File Details how to export a JSON alert file to the file system.
5. C.5 Import Alert File Details how to import a JSON alert file from the file system.
6. C.6 Silent Mode ON / OFF Details switching to silent mode.
7. C.7 Notification Details the mechanism for triggering and displaying notifications.
8. C.8 Reduce In Tray Details the reduction of the program in the system tray when a closeEvent event occures.
9. C.9 Exit Details the closing of the program.

Figure C.1: Add Simple Alert

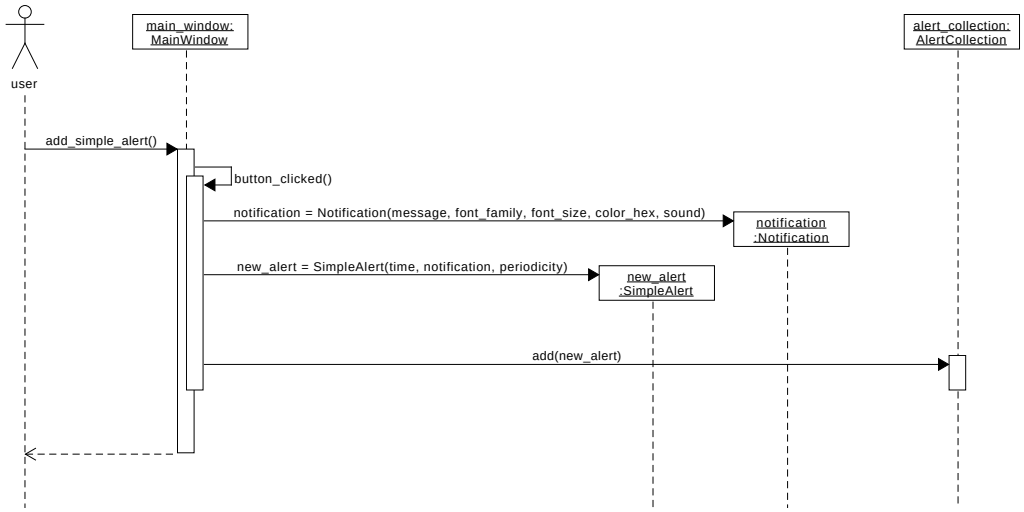


Figure C.2: Edit Simple Alert

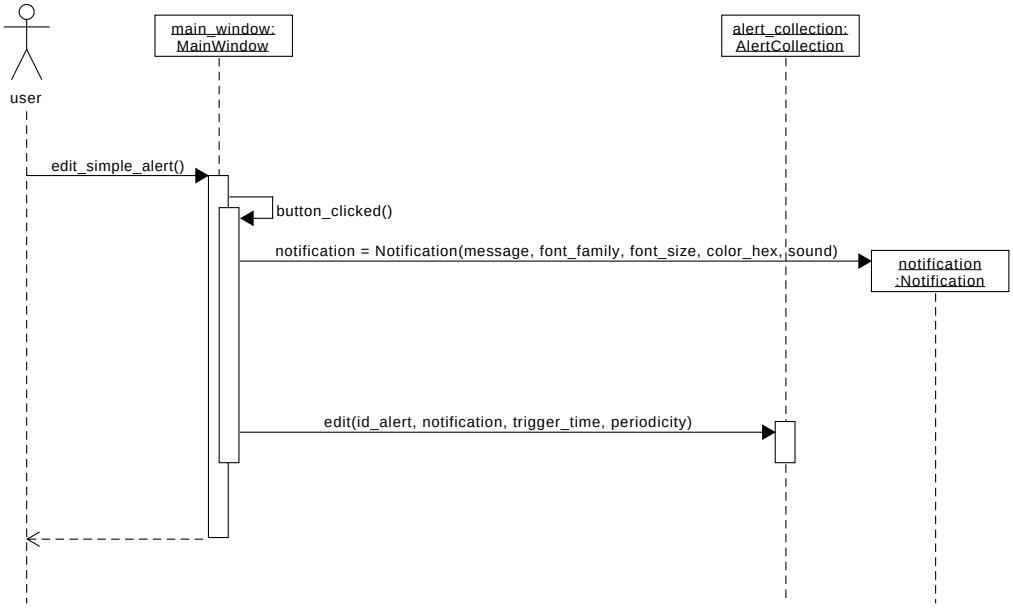


Figure C.3: Delete Simple Alert

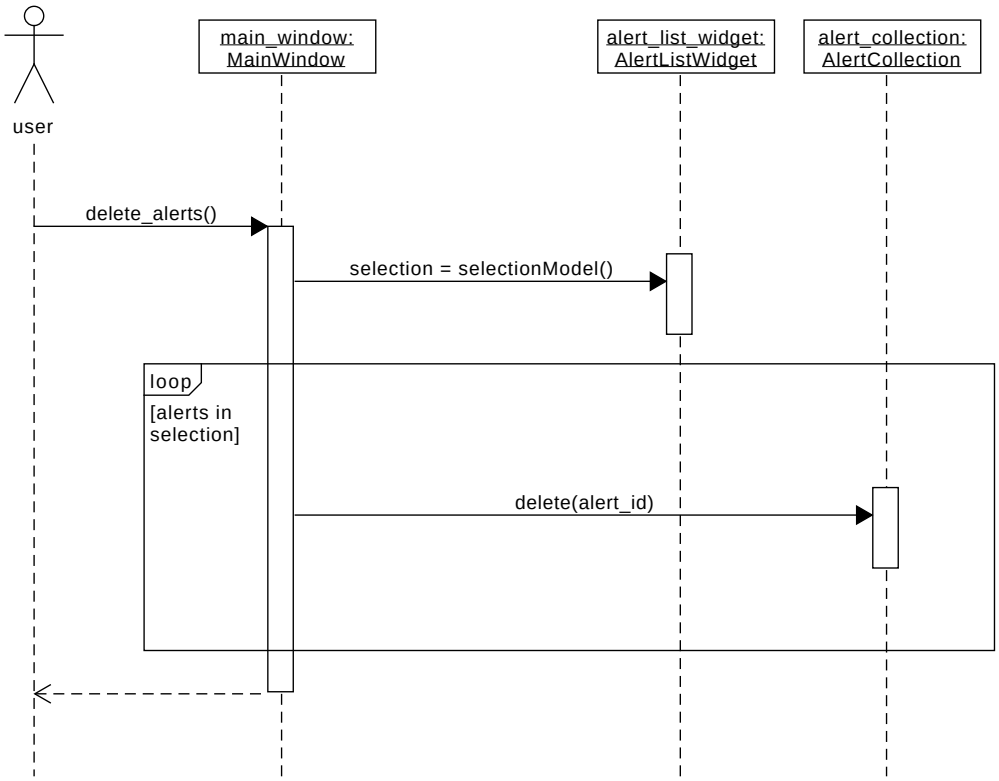


Figure C.4: Export Alert File

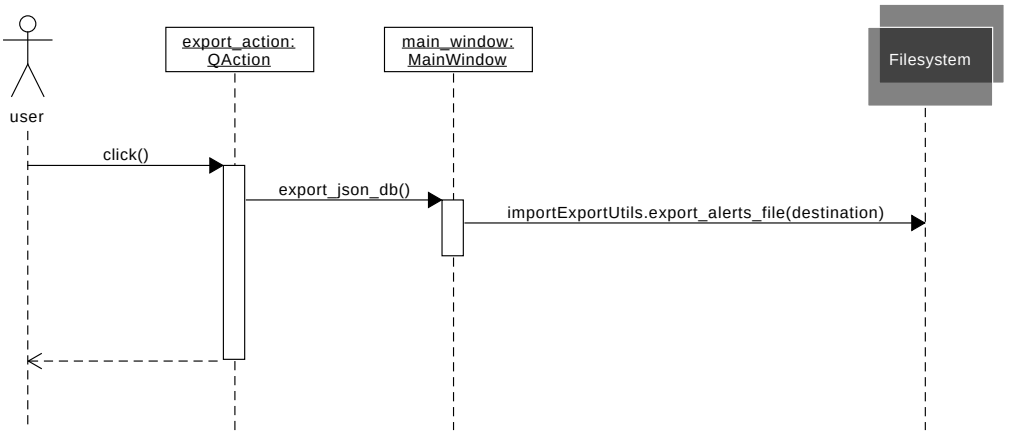


Figure C.5: Import Alert File

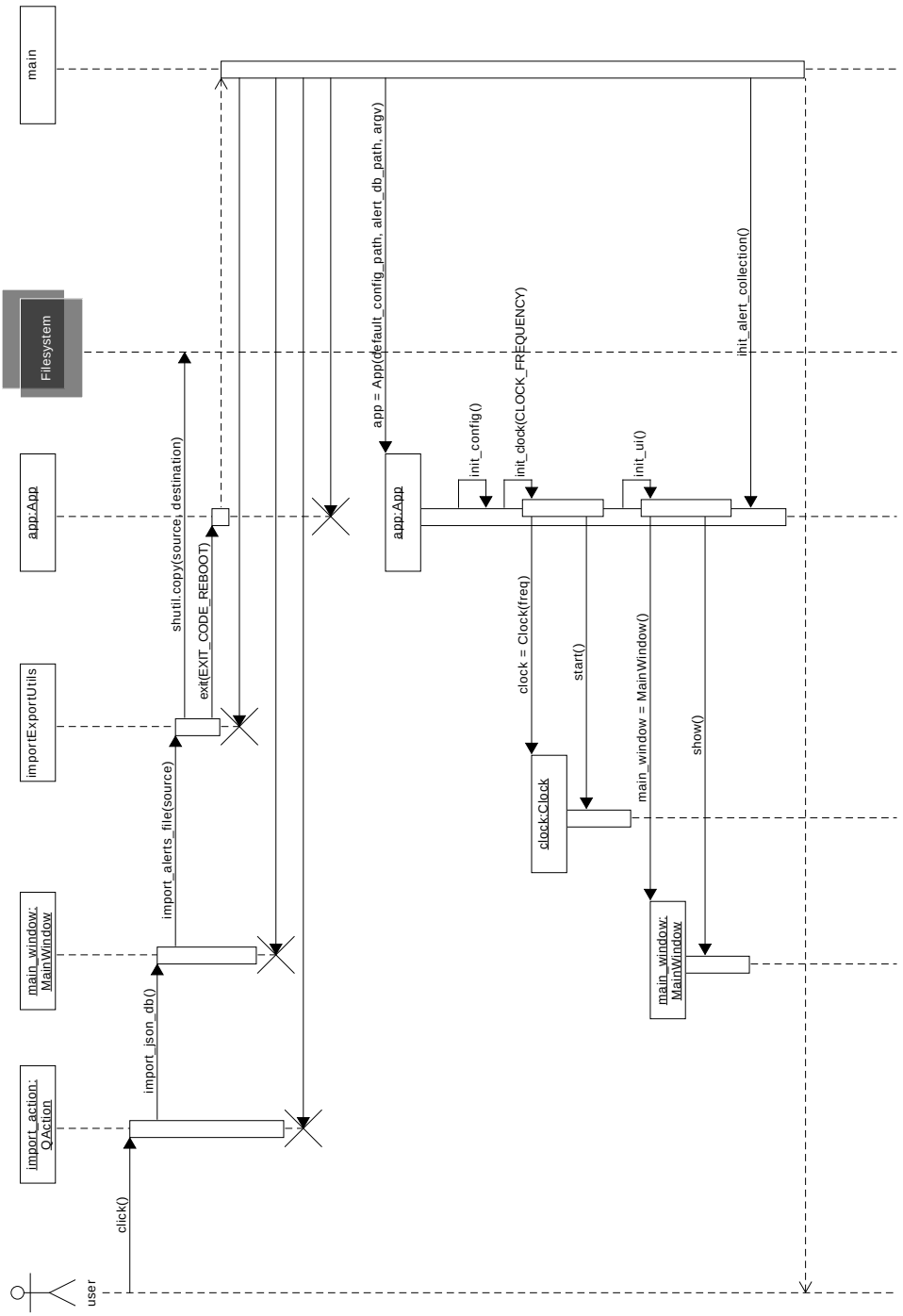


Figure C.6: Silent Mode ON / OFF

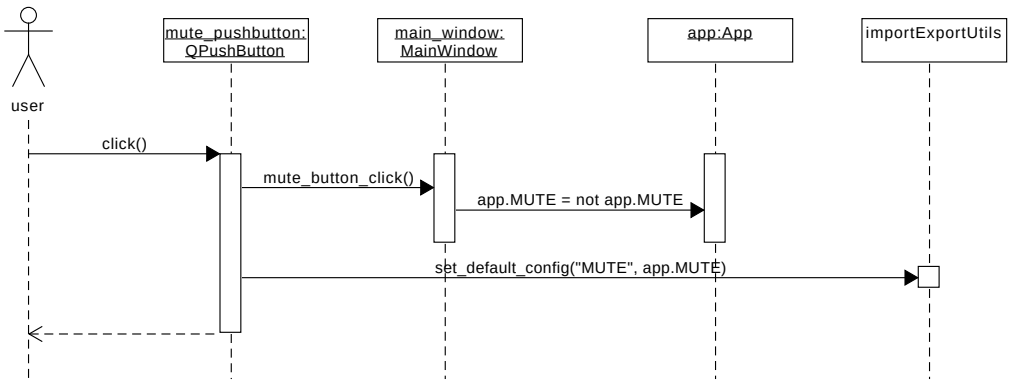


Figure C.7: Notification

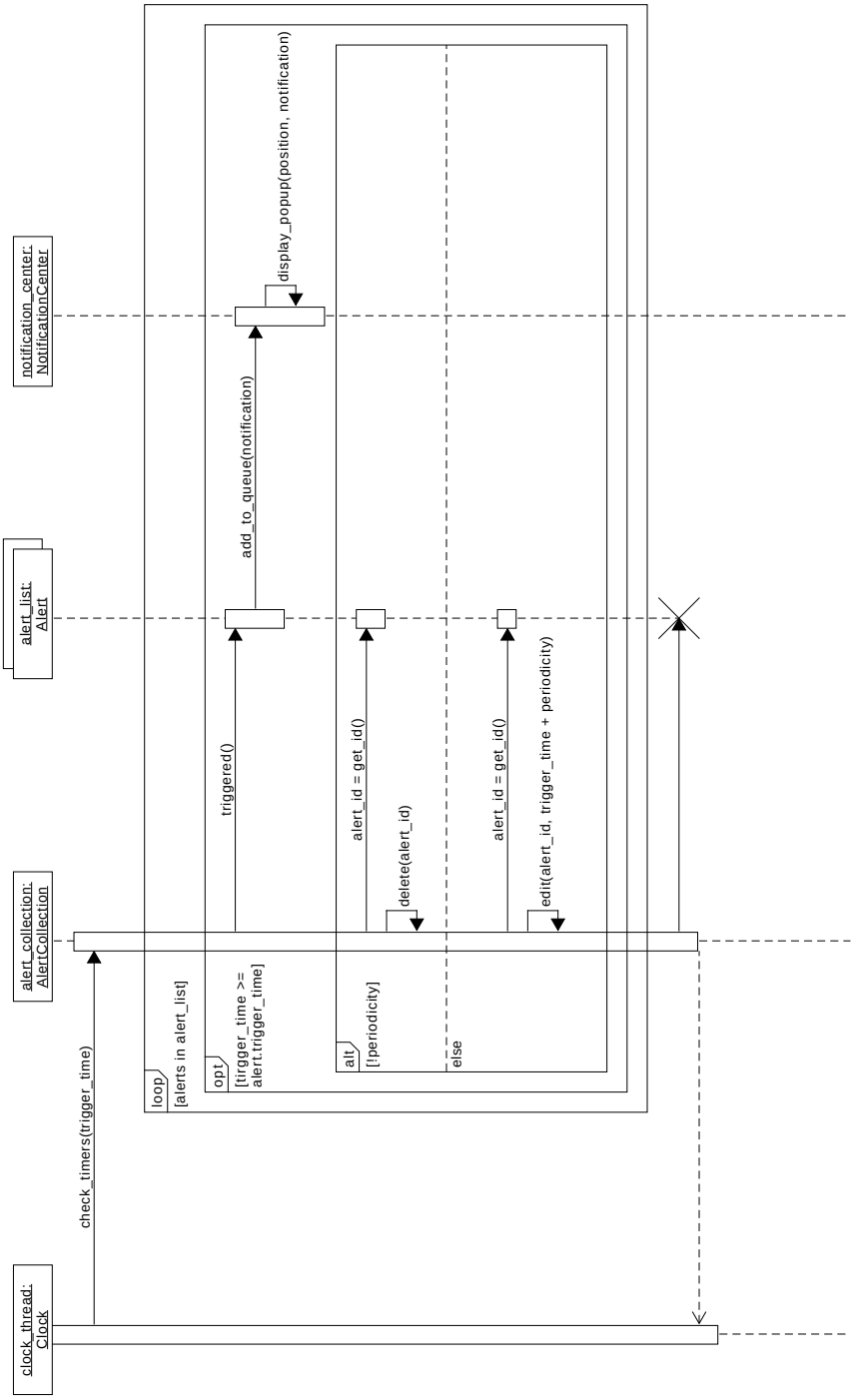


Figure C.8: Reduce In Tray

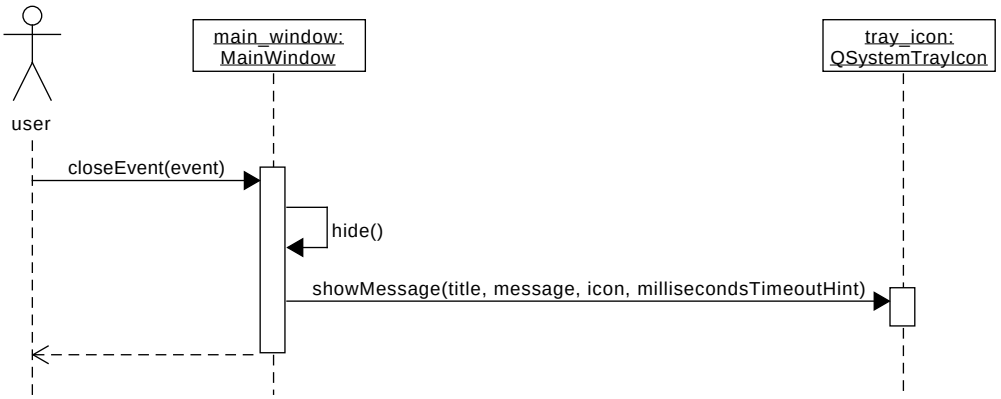
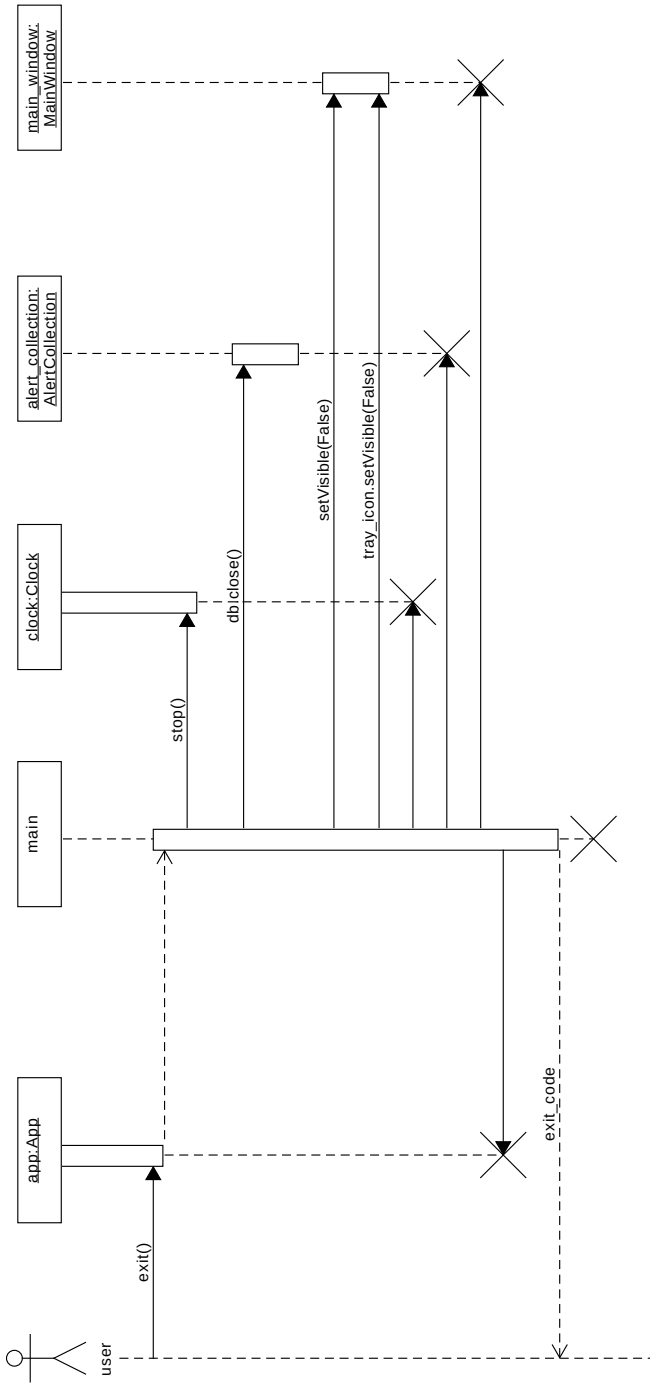


Figure C.9: Exit



Appendix D

Minutes

D.1 Meeting 1: March 1, 2017

Present at meeting

Samuel Gauthier, Loïc Charrière, Claude Fuhrer

Agenda

- Present the Use Cases and User stories
- Discussion about the usage of Python

Notes

Use Cases and User stories

- Vision&Goals: The vision and goals of the project have to be enhanced and not taken "as is".
- If a choice is made during the project, we should write down why it has been made this way. Everything that hasn't been documented is considered not been done.
- If we go for a specific product (such as DBMS, module, etc.) we should at least try out other alternatives (2–3) and explain why we made the choice of using this specific product.
- Split the user story "reate/delete categories" in two.
- Sort the uses cases by topic.
- Alarms could be snoozed. (for example, if the user is at a meeting snooze all alarms so that they don't disturb the meeting)
- Alarms should be user specific, i.e., they should belong to the user connected on the computer and not be shared across all the users of the machine.
- Book recommendation (chapter about use cases) -> Applying UML and Patterns (Larman).

Usage of Python

- Strongly discouraged to share python code via e-mail (General life advice)
- Code coverage 70–80% is ok, 30% is not

Tasks

- Improve vision&goals
- Create use cases / user stories / actors correctly
- Read Python doc

Next Meeting

~~TBD~~ (edit) March 17, 2017

D.2 Meeting 2: March 17, 2017

Present at meeting

Samuel Gauthier, Loïc Charrière, Claude Fuhrer

Agenda

- System and Context Boundaries
- System diagrams
- System description

Notes

The Project Goals should not be written down as a list. We need to create full sentences in order to help the reader's comprehension.
Question: "Should this document contain snippets of code?"
Answer: It depends on the target audience. But here we should only show snippets of pseudo code for complicated parts of our code.

System and Context Boundaries

- Some of the Stakeholders will not influence our project. They are only there so that our documentation is complete.

System diagrams

- Good domain model.
- The domain model should contain types only if they are very specific and part of the DM.
- It is not useful to specify interfaces and abstract classes.

System description

- We need to link the User Stories and Use Cases.

Other

Question: "How should our workflow be? Should we write the documentation and the code at the same time?"
Answer: We shouldn't waste too much time on writing documentation. Basically we should use the Scrum method. All the Use Cases (or the most important ones) have to be written down. Then we can decide what has to be included in each sprints. For each sprint we must have a working version of our application. Start with a small version having the alerts date hard coded in the code, with no GUI. Then add a new feature with each sprints.

Tasks

- Rewrite the Project Goals
- Prepare sprints

Next Meeting

TBD

D.3 Meeting 3: April 7, 2017

Present at meeting

Samuel Gauthier, Loïc Charrière, Claude Fuhrer

Agenda

- Show version with a simple notification

Notes

The `setup.py` is useful for bigger projects but in our case it is not. If we have time at the end then we should implement it.

The autostart function should be cared of only at the very end of the project. Users can do it themselves at the moment. More important are the persistence and customization features.

Question: Should we create one thread for every alarm or one thread for the hole program?

Answer: The best solution is to create one thread for the hole program that checks the due time ever 30 seconds or every minute.

Logging is useful and should be preferred instead of simple prints because it is possible to disable it completely for a project.

Tasks

- Document Use Case "Notification Center".
- Write the encountered development problems in the documentation.

Next Meeting

TBD

D.4 Meeting 4: May 19, 2017

Present at meeting

Samuel Gauthier, Loïc Charrière, Claude Fuhrer

Agenda

- Show version with customizable notifications including sound, font, color selection and recurrence. Import / export notifications file functionality. (state: commit 301822f)

Notes

As we reach the end of the project we should freeze the functionalities as they are now. The only modification that should be made is redesign the pop up because it is ugly.

Question: Now when the user selects a custom sound it is copied into the sound folder. If a sound file with the same name is present it is overridden. Is this behavior ok?

Answer: Very reasonable behavior but we should document it.

The functionalities that we didn't have time to implement and the bugs should be documented.

The Use Cases are too long, move them to the appendix. The documentation should not exceed 30 to 40 pages. Otherwise, it is not read by the expert.

Document the basic Sphinx usage.

The docstrings should contain either the preconditions or the raised exceptions of the function. The information put into the docstrings is information that can't be found in the code.

Document the python requirements and how to launch the program.

Question: How deep should our explanations go during the final presentation?

Answer: We should direct the presentation so that our class colleagues can understand what we did. They do not know our project so for the very complex parts of the code, show an expert or pseudo code.

Tasks

- Write the tests and the appropriate documentation
- Rewrite the readme and add the installation instructions with python requirements.

Next Meeting

TBD

Glossary

LaTeX Is a mark up language specially suited for scientific documents. 17

TeX Is a typesetting system developed by Donald Knuth. 17

Apache Subversion Is a software versioning and revision control system developed by the The Scribus Team. 18

Bitbucket Is a web-based Git and Mercurial repository and Internet hosting service owned by Atlassian. 18

BSD Berkeley Software Distribution. 20

CI Continuous Integration. 62

Continuous Integration Continuous Integration is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early [17]. 19

DTP desktop publishing. 63

Git Is a software versioning and revision control system originally created by Linus Torvalds and now developed by Junio Hamano. 18, 62

GitHub Is a web-based Git repository and Internet hosting service. 18

Google Docs Is an online word processor created and developed by Google. 17

GPL General Public License. 20

InDesign Is a desktop publishing software developed by Adobe Systems Incorporated. 17

KAlarm Is a personal alarm scheduler developed by David Jarvie. 5, 18

LGPL Lesser General Public License. 20

LibreOffice Is an open source office suite created and developed by The Document Foundation. 17

Linux Linux is a Unix-like computer operating system assembled under the model of free and open-source software development and distribution. 5

Mercurial Is a software versioning and revision control system developed by Matt Mackall. 18, 62

Metaclass Class whose instances are classes. 21

MH Must Have. 12

Microsoft Word Is a word processor created and developed by Microsoft. 17

NAS The Network Audio System is a network transparent, client/server audio transport system. It can be described as the audio equivalent of an X server.. 21

NAS Network Audio System. 63

NTH Nice To Have. 12

P1 First Piority. 12

P2 Second Piority. 12

Python Python is a widely used high-level programming language. Python has a design philosophy which emphasizes code readability, and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java.. 18

Qt Qt is a cross-platform application development framework for desktop, embedded and mobile [10]. 20

Scribus Is a is a desktop publishing application developed by The Scribus Team. 17

TDF The Document Foundation. 62

Travis CI Free continuous integration platform for GitHub projects. 19

TST The Scribus Team. 62, 63

UNIX Unix is a family of multitasking, multiuser computer operating systems that derive from the original AT&T Unix, developed starting in the 1970s at the Bell Labs research center by Ken Thompson, Dennis Ritchie, and others.. 5

VCS version control system. 18

Bibliography

[1] Patrick Thomson. *Git vs. Mercurial: Please Relax.* Aug. 2008. URL: https://importantshock.wordpress.com/2008/08/07/git-vs-mercurial/ (visited on 03/12/2017).

[2] Klaus Pohl and Chris Rupp. *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB compliant.* English. 1 edition. Santa Barbara, CA : Sebastopol, CA: Rocky Nook, Apr. 2011. ISBN: 978-1-933952-81-9.

[3] mpm. *ZipdocExtension - Mercurial.* Nov. 2012. URL: https://www.mercurial-scm.org/wiki/ZipdocExtension (visited on 03/12/2017).

[4] William von Hagen. *Open source desktop publishing with Scribus.* Apr. 2013. URL: http://www.ibm.com/developerworks/library/os-scribus/index.html (visited on 03/12/2017).

[5] David Jarvie. *The KAlarm Handbook.* 2016. URL: https://docs.kde.org/trunk5/en/pim/kalarm/index.html.

[6] *Levels of TeX - TeX Users Group.* Jan. 2017. URL: http://tug.org/levels.html (visited on 03/12/2017).

[7] *26.3. doctest — Test interactive Python examples — Python 3.6.1 documentation.* URL: https://docs.python.org/3.6/library/doctest.html#module-doctest (visited on 04/06/2017).

[8] *26.4. unittest — Unit testing framework — Python 3.6.1 documentation.* URL: https://docs.python.org/3/library/unittest.html (visited on 04/06/2017).

[9] *26.5. unittest.mock — mock object library — Python 3.7.0a0 documentation.* URL: https://docs.python.org/dev/library/unittest.mock.html (visited on 04/06/2017).

[10] *About Qt - Qt Wiki.* URL: https://wiki.qt.io/About_Qt (visited on 06/06/2017).

[11] *Coverage.py — Coverage.py 4.4b1 documentation.* URL: http://coverage.readthedocs.io/en/latest/ (visited on 04/06/2017).

[12] *Differences: nose2 vs nose vs unittest2 — nose2 0.6.0 documentation.* URL: https://nose2.readthedocs.io/en/latest/differences.html#differences-nose2-vs-nose-vs-unittest2 (visited on 04/06/2017).

[13] Ionel Cristian Mărieș. *Understanding Python metaclasses | ionel's codelog.* URL: https://blog.ionelmc.ro/2015/02/09/understanding-python-metaclasses/ (visited on 05/16/2017).

[14] *pytest: helps you write better programs — pytest documentation.* URL: https://docs.pytest.org/en/latest/index.html (visited on 04/06/2017).

[15] Kenneth Reitz Schlusser Tanya. *The Hitchhiker's Guide to Python.* ISBN: 978-1-4919-3317-6. URL: http://shop.oreilly.com/product/0636920042921.do (visited on 04/25/2017).

[16] Michele Simionato. *SOLVING THE METACLASS CONFLICT.* URL: http://www.phyast.pitt.edu/~micheles/python/metatype.html (visited on 05/14/2017).

[17] *ThoughtWorks | Creative technology consultants.* URL: https://www.thoughtworks.com/continuous-integration (visited on 06/06/2017).

[18] *tox 2.7.0 documentation.* URL: https://tox.readthedocs.io/en/latest/ (visited on 04/06/2017).

[19] *Version Control Systems Popularity in 2016.* URL: https://rhodecode.com/insights/version-control-systems-2016 (visited on 03/12/2017).