# Assignment 1: Jodel Alert Design Document

Software Engineering - Design
*- 2DV603*

*Name:* Patrik Hermansson
*Email:* ph222md@student.lnu.se
*Name:* Michael Wagnberg
*Email:* mw222uu@student.lnu.se
*Name:* Benjamin Svärd
*Email:* bs222et@student.lnu.se
*Name:* Christofer Nguyen
*Email:* cn222hn@student.lnu.se
*Name:* Jonathan Walkden
*Email:* jw222qi@student.lnu.se

## Contents

| Revision | Author | Date | Changes |
|---|---|---|---|
| 1 | Patrik Hermansson | 2017-04-07 | Created the document |
| 1.1 | Patrik Hermansson | 2017-04-10 | Outline of Purpose and General Priorities |
| 1.2 | Patrik Hermansson | 2017-04-18 | Rewrote some of the purpose with revised requirements |
| 1.3 | Patrik Hermansson | 2017-04-24 | Design issues started to show, updated document |
| 2 | Benjamin Kangur Swärd | 2017-04-26 | Added Class diagram |
| 2.1 | Benjamin Kangur Swärd | 2017-04-26 | Added Component diagram |
| 2.2 | Benjamin Kangur Swärd | 2017-04-29 | Added security and GUI to class diagram |
| 3 | Michael Wagnberg | 2017-05-05 | Added Design Space according to our design options |
| 3.1 | Christofer Nguyen | 2017-05-05 | Added Activity Diagram for modifying database |
| 3.2 | Christofer Nguyen | 2017-05-05 | Added Activity Diagram for handling jodel post |
| 3.3 | Michael Wagnberg | 2017-05-08 | Added GUI to Design Space |
| 4 | Jonathan Walkden | 2017-05-10 | Added Sequence Diagram for main post alert sequence |
| 4.1 | Jonathan Walkden | 2017-05-13 | Added Sequence Diagram for keyword creation |
| 4.2 | Patrik Hermansson | 2017-05-13 | Wrote design options |
| 4.3 | Patrik Hermansson | 2017-05-15 | Developers desiderata and after release |
| 5 | Michael Wagnberg | 2017-05-19 | MVC vs Pipe-and-Filter |

# 1 Design Document

## 1.1 Purpose

This document will describe the entire design and decisions as well as rationales regarding Jodel application API, work-arounds, client-server and core functionality and vital parts of Jodel Alert.

The Jodel Alert Application will scan Jodel post feed, derive keywords desiderata and notify an email recipient(requirement ID 1). It will do this every 15 minutes (requirement ID 2). Possible failure conditions can be that Jodel Alert will not get an authenticated token, the server which Jodel Alert runs upon is down, or wrong format of email recipient. This will be handled by error messages to the user.

The Jodel Alert Application user will be able to add and change email recipient as well as keywords(requirement ID 4,5,6,7)

The Jodel Alert Application admin will be able to remove email recipient and keywords(requirement 3,8) as well as the ordinary user actions.

## 1.2 General priorities

One of our highest priorities is simplicity. The core problem we are trying to solve, the project idéa, is in itself simple and small, which makes it reasonable to design everything with simplicity in our minds.

The alert that raises awareness of a post that Linnéstudenterna is interested in is the important result. There are several ways to accomplish that, you can have an advanced GUI for the customer, or different ways to change the properties of keywords and email.

We have designed everything to just solve the core problem, the alert. The Jodel Alert will run in the background on a server, and send alerts (emails) when keywords found.

We have also focused on reuseability as a priority in our design, this is merely because of our projects uniqueness on the Jodel application.

Everything will be general and not suited just for our customer, this makes other companies potential customers.

To get our application to have high scalability, we decided to make a user and administrator mode. With the case of just a generic user that can change, add and remove keywords and email recipients works fine for a smaller business with maybe 5-10 employees. To ensure scalability we decided that we had to implement an administrator and a user system. This means that a bigger company can have a few administrators that have more privileges than the other users, which in turn results in a better maintained database of keywords and emails.

We also wanted to design for portability. The application will be written and developed in Java. This makes it versatile, supported by multiple vendors and can work on all platforms needed.

## 1.3 Design issues

One of the first things that had to be resolved was of course the Jodel API. We did not know if that API was open or not accessible for the general public. We planned accordingly to cover both situations, so we were ready to design with the API or without.

After several tries of contacting the developers of Jodel we understood that no answer would arrive. Instead we moved towards a program called mitmproxy, that deals with listening to the traffic from Jodel by imitating an ordinary client cellphone. We can then extract that data and use it in our favour.

Choosing database was another possible constraint or issue. Regarding to the technical aspect, maybe no one in the group have competence to use a powerful database (like MySQL) in collaboration with the never used program mitmproxy. Another aspect is budget and time for implementing MySQL as a database.

We have choosen a design that fits a "database", and we can easily choose between a commercial database or just a plane text file, depending on difficulties and other constraints that may show up.

### 1.4 Outline of the design

We use a top-down design where we at a very high level develop the design of the system in order to ensure that we actually conform to the requirements given by the customer in order to solve it's problems.
We have decided about an MVC architecture and using MySQL as a database.

Very early in to the design phase we decided that we are going to separate the user interface from the rest of the system, which ensures a flexible and maintainable design.

## 2 Design details

The UML use cases describe the systems behaviour and what each case intends to achieve. This UML design phase is implemented to achieve for both stakeholders and project members an understanding for how the applications functions will work together with the end-user.

Following in this section there will also be a graphical illustration of the use cases depict in workflows that will be intended to describe the operational workflows of the system. To view activity diagram look on appendix 1.2.

All key components for this application is listed here.

### 2.1 Activity diagram - Handle Jodel Post

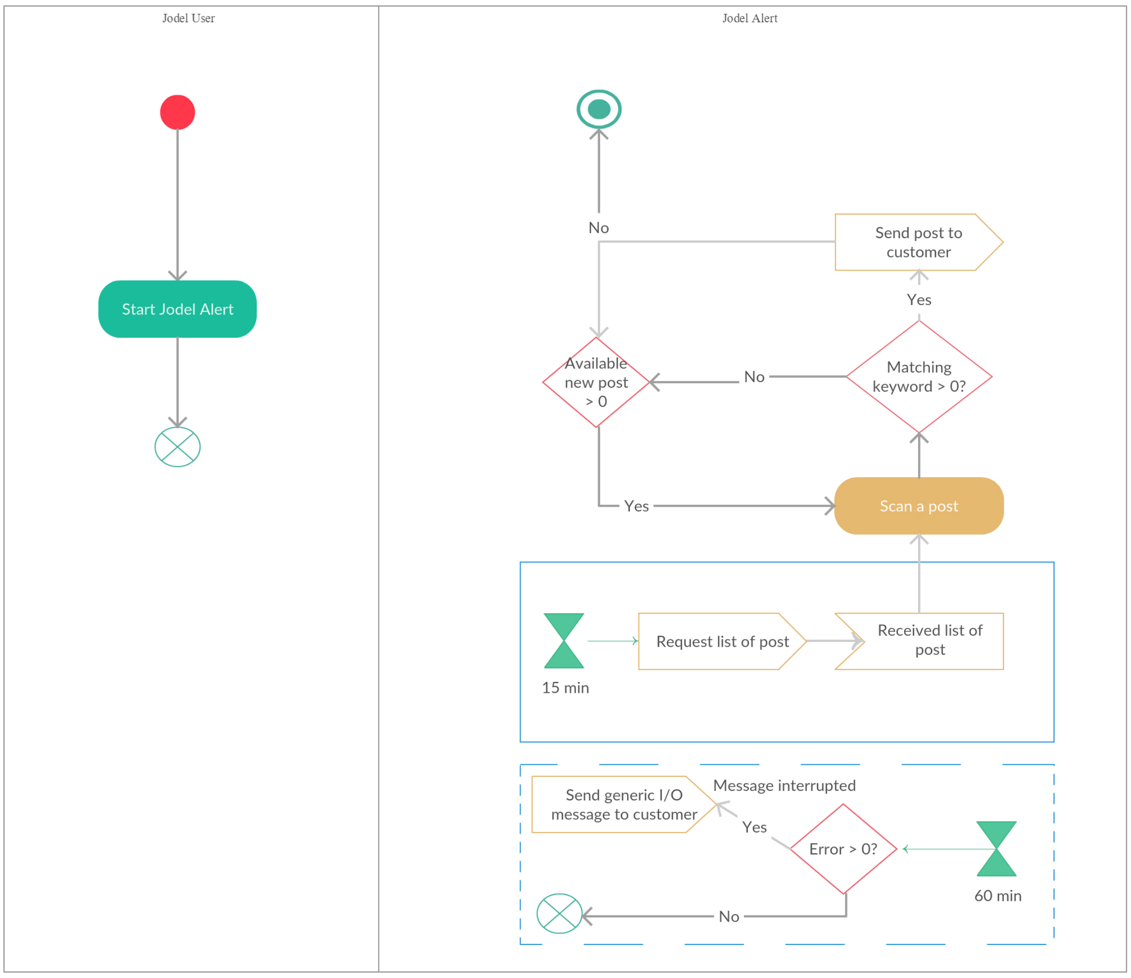

Figure 2.1: Jodel Alert Activity Diagram Handling Jodel Post

## 2.2 Activity diagram - Modifying Database

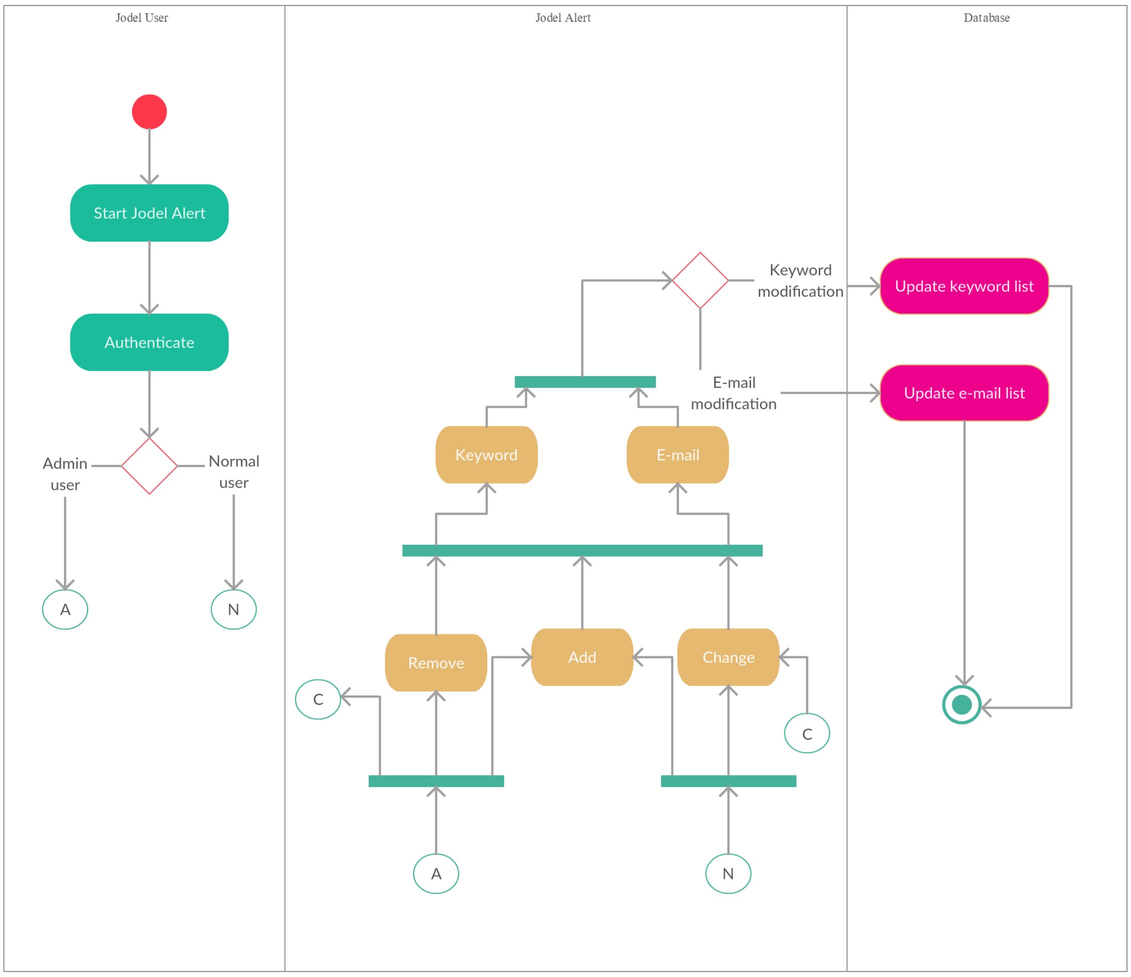

Figure 2.2: Jodel Alert Activity Diagram Modifying Database

## 2.3 Constraints

(Probably not in the finished version)?

## 2.4 Design options

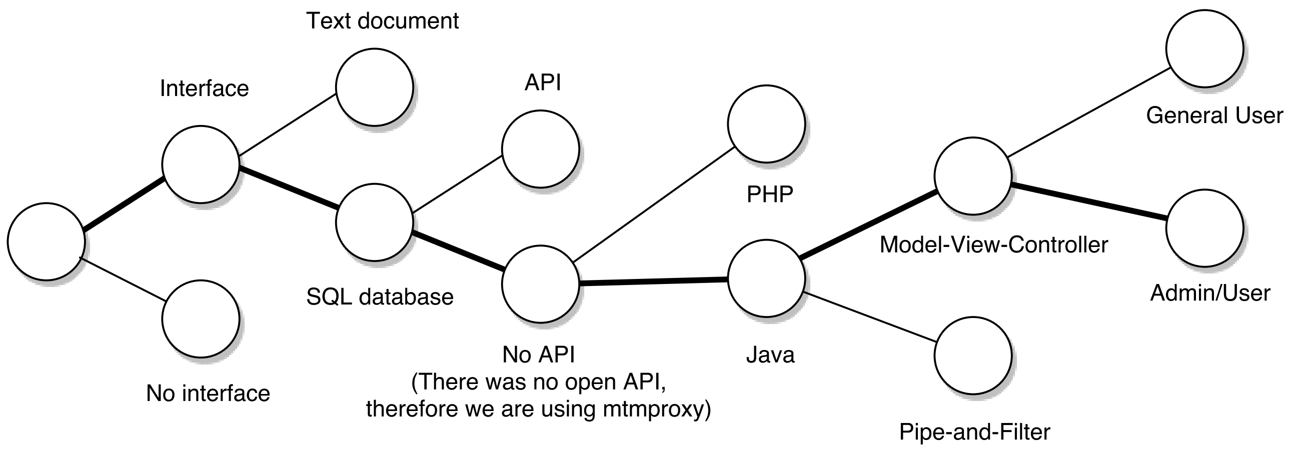

Figure 2.3: Jodel Alert Design Space

User interface vs no user interface is a big difference. As we stated in requirement ID 13 the user interface should be easy to use and simple. If going for no user interface the users are forced to have a higher level of knowledge to operate the application correctly. Adding a simple keyword can mean knowledge of how databases works and maybe require other programs. With the user interface the user can add a simple keyword without knowing about the underlying backbone of our system. This gives the application good usability.

We choose an SQL database which is easier to maintain, than a plane text file. We also get the properties of using Jodel Alert Application on several machines on work and at home, which does not work if using a text file locally on the machine running the application. Scalability is also reached with using an SQL database.

One of the design issues mentioned before was the use of the Jodel API. This was not open and accessible for the general public, and we did not get any response from the Jodel company. This forced us to choose the path of no API at all.

### 2.5 Failure conditions

Some of the possible failure conditions that can arise is the following:

- Not receiving authorized token from Jodel
- Server at Linnéstudenterna is down
- Could not retrieve new posts

Whenever some of these failure conditions occur, the system prompts the user with a message in form of an email. No other actions will be taken after a misevent than only informing the recipient what has gone wrong.

### 2.6 One-time operation

After installing the Jodel Alert Application, the customer must start the application. This is a one-time operation in order to run the application on the server, which means it will begin scanning and sending emails to the recipient if a keyword is matched.

### 2.7 Limitations

If a user creates an entry, there are for now not any limitations on how many keywords or email recipients that can be handled by the database. (Probably not in the finished version)?

## 3 Class diagram

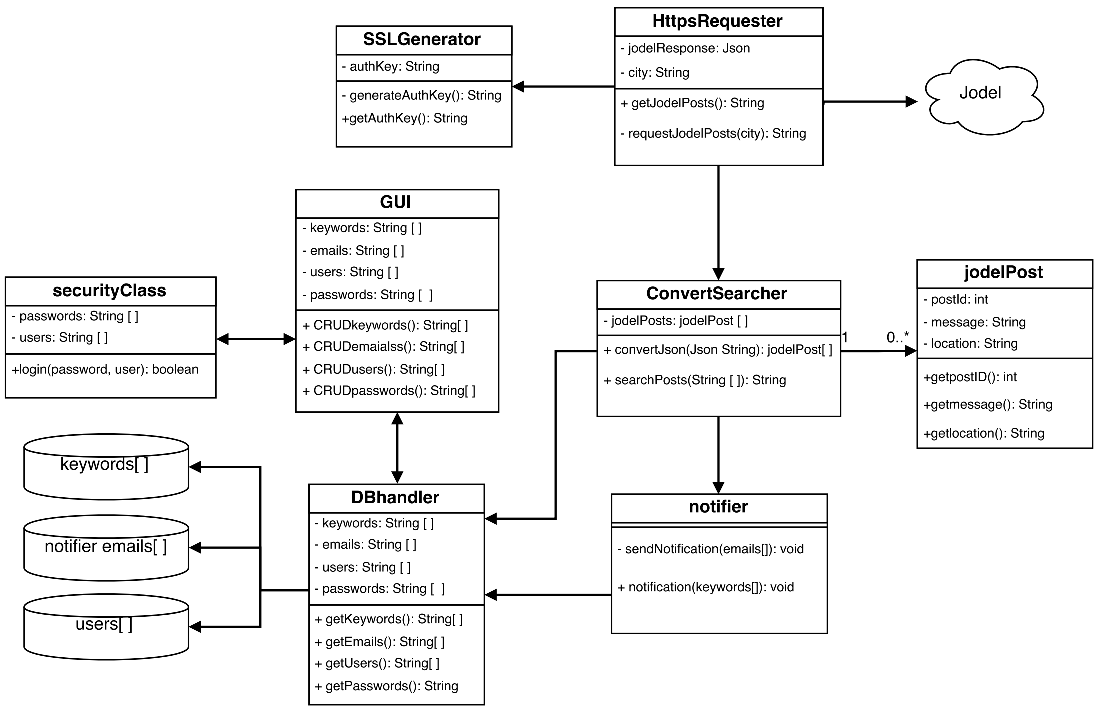

Figure 3.4: Jodel Alert Class Diagram

## 4 Component Diagram

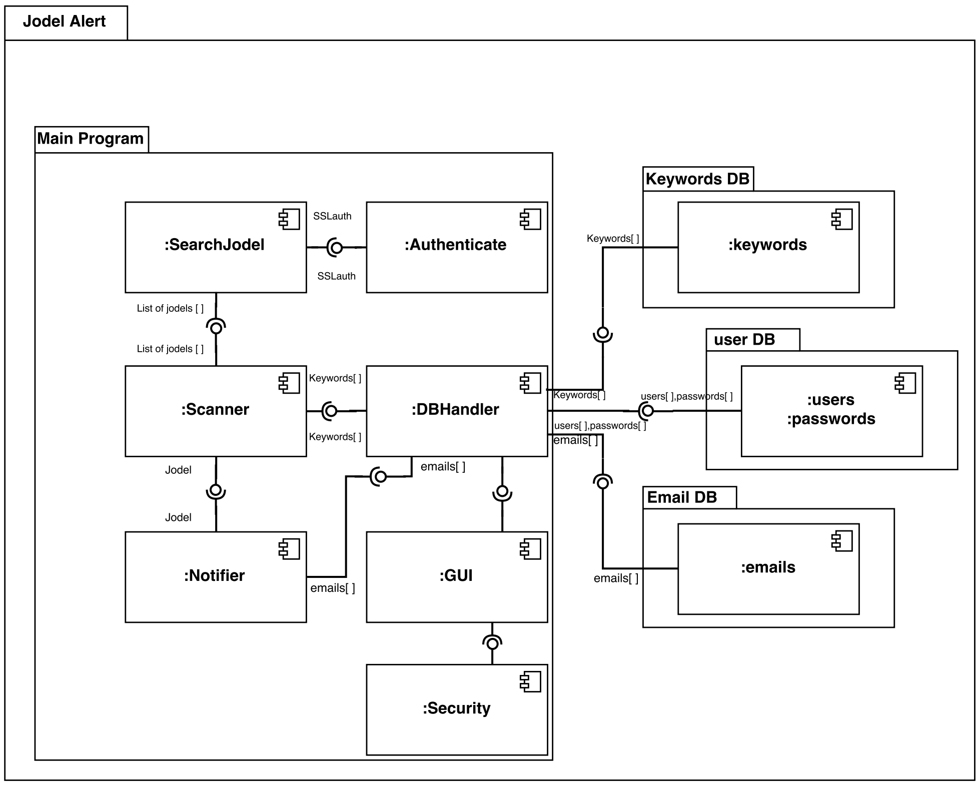

Figure 4.5: Jodel Alert Component Diagram

## 5 Sequence Diagram

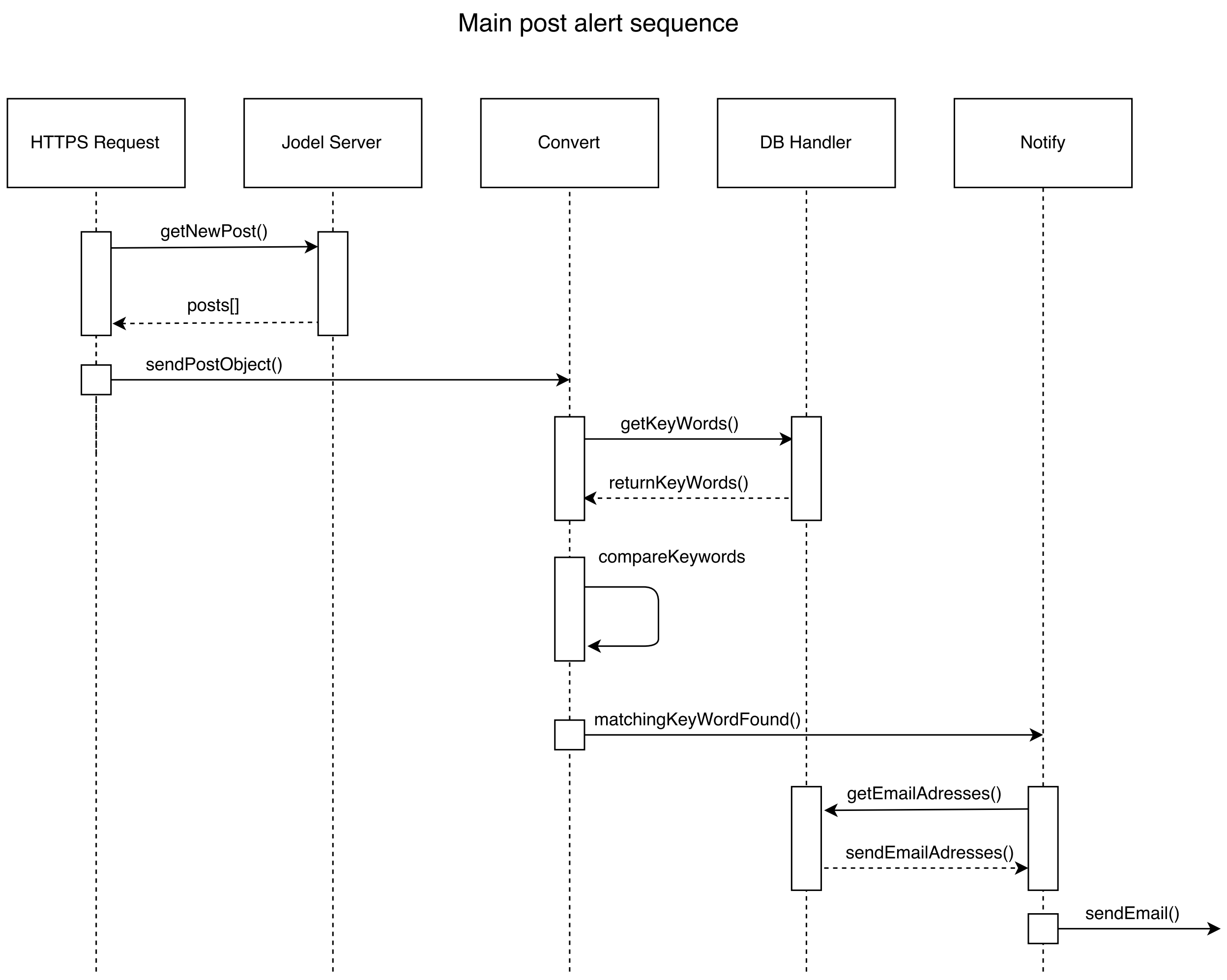

Figure 5.6: Jodel Alert Sequence Diagram

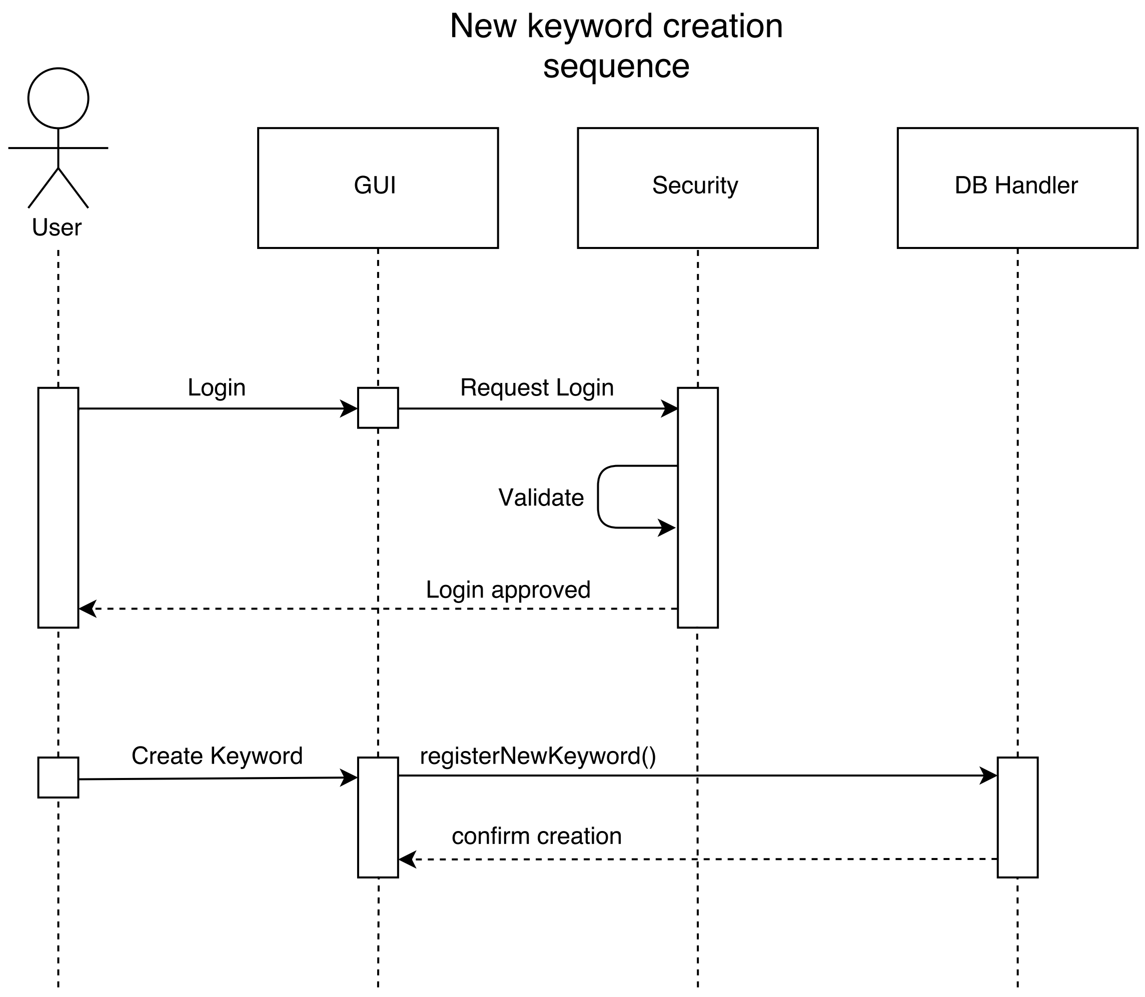

Figure 5.7: Jodel Alert Sequence Diagram

## 6 After release - Nongoals

Some things is always forced to be omitted because of time, budget and other constraints.

In version 2.0 of our application we would like to develop the following features:

- Ability to post answers to Jodel directly from the GUI of Jodel Alert
- Each user can have their own customized keywords

Figure 6.8: Jodel logo