

# Coursework 2

---

## Web-app: Napier Public Key Server

SET09103 - Advanced Web Technologies

Module Leader: Mr. Simon Wells

29.11.2015

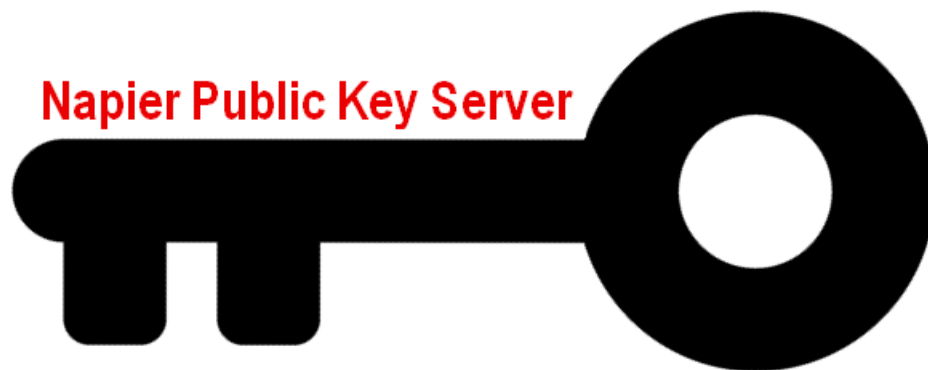


Figure 1: Logo

Student-ID 40219221

Exchange Student

## Abstract

In the module SET09103 “Advanced Web Technologies” by the module leader Dr. Simon Wells, each participant student has to work out two coursework. The following assignment represents the second coursework and adapted the development of a web application about a personal project. In this following work the chosen topic is the development of a Public Key Server called “Napier Public Key Server”.

The requirements and constraints on this web-app are the using of Python Flask, routing, static files, requests, redirects, templates, sessions, logging, testing, CSS, JavaScript, multiple users, and data storage. In addition to that there has to be designed a functional URL hierarchy. Therefore the source code has to run in the Levinux environment and it has to be committed in GitHub.

Summarized this coursework dealt with the implementation and development of a web-app about a Public Key Server, which afford users a secret and safety message service.

## Content

|                                    |    |
|------------------------------------|----|
| 1. Introduction.....               | 1  |
| 2. Description of the web-app..... | 2  |
| 3. Design section .....            | 5  |
| 4. Enhancements .....              | 9  |
| 5. Critical evaluation.....        | 10 |
| 6. Personal evaluation.....        | 11 |
| 7. References.....                 | 12 |
| 8. Appendix.....                   | 14 |
| 8.1 GitHub-Commits.....            | 14 |

## List of Figures

|   |    |
|---|----|
| Figure 1: Logo .....                    | I  |
| Figure 2: Secret Key Crypto .....       | 2  |
| Figure 3: Public-key Crypto .....       | 3  |
| Figure 4: Problematic Git comment ..... | 11 |

## List of Tables

|                                |   |
|--------------------------------|---|
| Tabelle 1: URL history.....    | 6 |
| Tabelle 2: File structure..... | 7 |

## 1. Introduction

Privacy and Security – these are the two items, which come to our mind when we think about communication in the World Wide Web. Sometimes we want to communicate with a friend about secrets by social media for example Facebook - only to mention one possibility of the potential internet services. Of course these secrets should not know by others, other ways they would not be secrets. But how can we be sure that no one except the receiver or receivers read the messages?

Consequently the developed web-app represents a Public Key Server-app which is called “Napier Public Key Server”. This web-app enables the users to communicate safely. Therefore every user has a pair of keys – a public and a private key. The public key is for the sender to encrypt the message and the private key is for the receiver to decrypt the message to finally read it.

So first of all this coursework explains some basic inputs about encryption. Then a description of the web-app is shown. Consequently, after explaining how this web-app is architected, there are some enhancements mentioned. Finally the last two categories deal with a critical and personal evaluation.

## 2. Description of the web-app

Before the developed web-app is described, there has to be explained some basics. First of all there are shown the two different encryption methods, the “Secret Key Encryption” and the “Public Key Encryption”. Afterwards some information about databases are explained.

### Secret Key Encryption

This Encryption involves applying an operation or better said an algorithm, which use a private key to make the transferred data unreadable (CCM Benchmark Group, 2015). Only under the premise that the used key is known only by both, sender and receiver, the data can encrypted and later decrypted to read. As shown in the picture below, Figure 2, there is Alice as sender and Bob as receiver. Alice wants to send Bob a secret message, which is represented by the “M”. After encrypting the message with the secret key, there is only an unreadable text left, which is represented with the “C”, which means cipher text. Only the person who owns the private key can decrypt the cipher text. So Bob, who owns the key, can decrypt the cipher text and can read it.

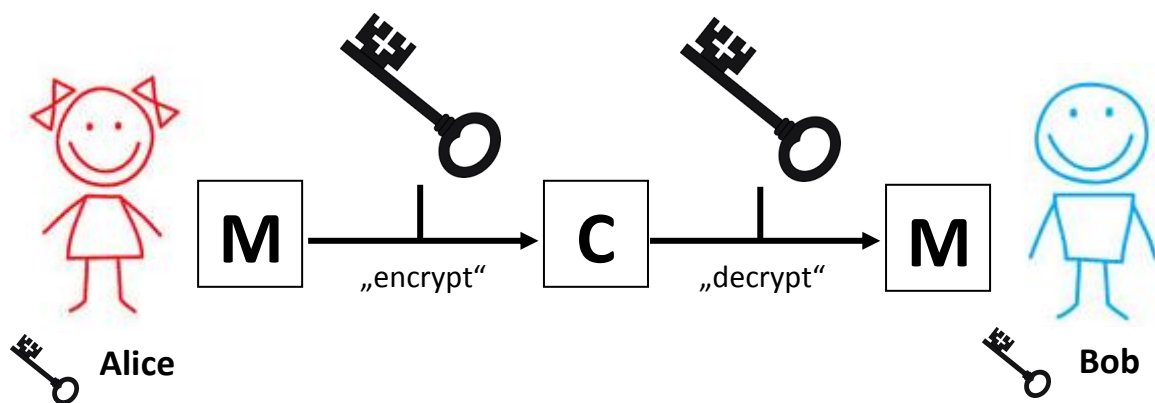


Figure 2: Secret Key Crypto

So this method of encryption is simple. The problem is to handle with the secret keys. Either there will be many different keys for every communication between two groups, which is difficult to handle, or there is only one key, which means that it is not secret anymore because everyone, who owns the key, can decrypt the message although the message is not addressed to him.

## Public Key Encryption

An alternative is the public-key encryption. For this encryption method is needed two keys that mean a pair of keys. One key, the public key, is for encryption and another key, the secret key, is for decrypt the transferred message. In a public-key encryption system, users generate a keypair consisting of a private key that only they know and a corresponding public key. Users exchange this public key over an insecure channel (CCM Benchmark Group, 2015).

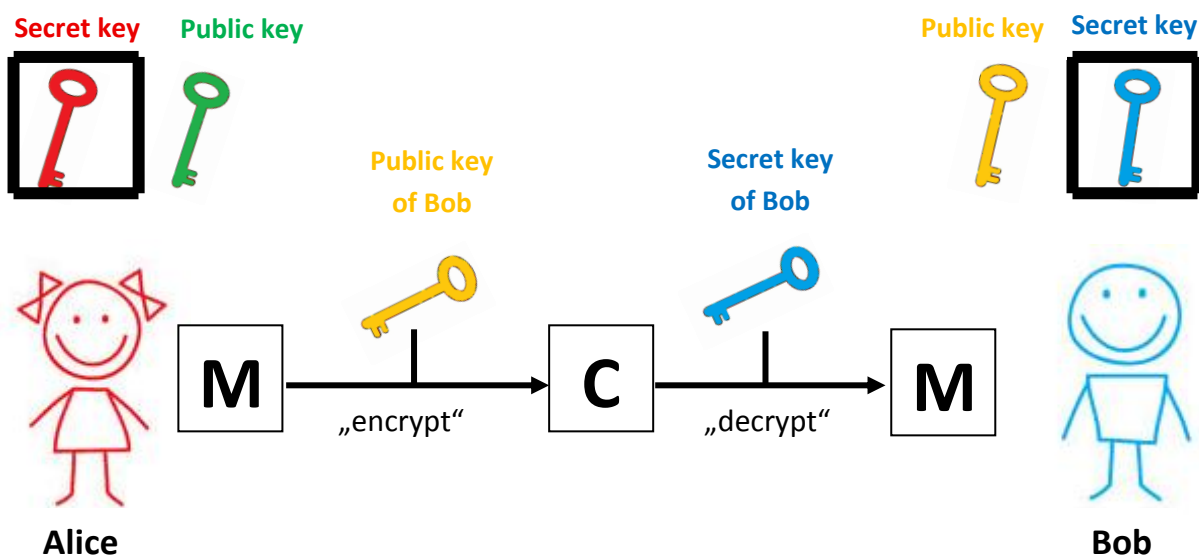


Figure 3: Public-key Crypto

Again, Alice wants to send Bob a secret message. In the system of public-key encryption Alice uses the orange key, public key, of Bob and encrypt the message. Now this encrypted text is not readable anymore for other users. Then Bob take his blue key, the secret key, which only he knows and own, and decrypt the message. The message cannot be decrypted with the public key. Finally he can read what Alice sends him. This is possible because of the algorithm of the key-pairs as mentioned above.

The advantage is that the user only has to keep one key, his own key pair. By using the Secret Key Crypto the users have to keep all keys secret what means more effort.

## Databases

A database is a collection of information that is organized so that it can easily be accessed, managed, and updated (TeachTarget, 2015). So a database serves as a structured storage for many data.

There are some important and relevant guidelines by develop a database. The following three points represent the most important ones:

- Clarity – The sense of databases is to store many data, which the user wants to find and get easily and fast.
- No duplication – Data storage space is expensive moreover the arising of mistakes by changes is significant higher.
- Empty cells or fields – Unused storage places are expensive because they were provided and were filled with NULL, what means that there is no information lodged but kept free for some data.

In addition further requirements for a database are persistence, integration of data, data structuring / data design, request / reading and writing of data, data and database consistency, time sharing / coordination, protection, redundancy-free, performance / scalability, and resistance (Scheuermann, 2015).

Summarized the developed web-app is a public key a public key server. It is comparable with a phone book, where public keys are stored. So every user can search and find public keys of other users, which he wants to contact or send a secure message to. In addition to that it is possible to add users as friends. Consequently all added friends were presented on the right cell of the “My profile” page. At the same time, after adding someone as friend, there arises a short message under the profile picture, which tells users if you are friends. Therefore each user can send other users secure messages. By going on the wished receiver’s profile there is a message panel watchable. There everyone can send this user a secure message. In addition to that every user can see on his own profile page his messages. It is possible to encrypt all the arrived messages on the profile page. That means that no one can read messages when the user locked and encrypted his messages. Only after entering the password the messages get decrypted and readable again.

Finally the “Napier Public Key Server” represents a secret message service with some special features.



### 3. Design section

#### Procedure of the development

First of all there has to be made a decision about the topic. After a long period of researching the chosen topic is the development of a Public Key Server web-app.

The next step was doing a rough and temporary layout, structure and design. So that means to look and think about the URL hierarchy. While this hierarchy was built, there was made some comparisons with other Public Key Server Services, like for example (Keybase, 2015). At the same time there were worked out some basic functions, too. After the drawings and paintings on the paper, the design was adapted to the tutorial website of w3schools to define the user surface by bootstrap.

After creating the design drawings and the bootstrap code, there had to be made some ideas and researches about the database and its schema. So with the help of some tutorials, for example (Ronacher, 2014), (SQLite, 2015) and lecture notes of database systems (Scheuermann, 2015), there was developed the design of the database and some important functions. For the web-app there was used and imported the SQLite3, which is compatible with Flask. "SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed database engine in the world" (SQLite, 2015).

Then there has to be made some thoughts about nearly the most important part of this coursework the encryption of the private and public key. Therefore it was used some predesigned encryption items. "Cryptography is an actively developed library that provides cryptographic recipes and primitives. It supports Python 2.6-2.7, [...]. Cryptography is divided into two layers of recipes and hazardous materials (hazmat). The recipes layer provides simple API for proper symmetric encryption and the hazmat layer provides low-level cryptographic primitives" (Reitz, 2014). In addition to this reference there were some really helpful and suitable questions and answers in the stackoverflow.com-forum. For example when you want to have a look at simple possibilities to encode passwords, there are many different comments, which helped me a lot.

So the table below, Table 1, represents an overview of the URL history.

| URL history |           |                   | Short comment                     |  |
|-------------|-----------|-------------------|-----------------------------------|--|
| /           |           |                   | Start page, show mainpage.html    |  |
|             | /register |                   | Someone wants to sign up.         |  |
|             | /login    |                   | On this site user can login       |  |
|             | /logout   |                   | After login has to get logged out |  |
|             | /profile  |                   | After login, see own profile      |  |
|             | "         | /lock             | Message encrypted                 |  |
|             | "         | /unlock           | Message decrypted                 |  |
|             | /user     | /<username>       |                                   | Show profile page of other user and show if they are friends                               |
|             | "         | /<username> /add  |                                   | This route enable users to be friends  |
|             | "         | /<username> /send |                                   | Possibility to send other users secure messages  |
|             | /info     |                   |                                   | By clicking on terms and conditions, there arises a info page                              |
|             | /search   |                   |                                   | The navigation bar contents a search function, where users can search and find other users |

Tabelle 1: URL history

Then the corresponding file-structure is shown by the following table, Table 2.

| File structure: coursework2: |                        | Short explanation   |
|------------------------------|------------------------|---|
| <b>data</b>                  |                        | Directory for storing the database  |
|                              | README                 |   |
| <b>exampledata</b>           |                        |   |
|                              | friends.example        | Examples for friendrelationships between users                            |
|                              | user.example           | Users which are registered  |
|                              | messages.example       | Store example messages to show encryption and decryption                  |
| <b>static</b>                |                        | This file contains all static files like bootstrap, pictures, ...         |
|                              | Alice.png              | pictures  |
|                              | Bob.png                |   |
|                              | logo.png               |   |
|                              | startpic.png           |   |
|                              | css                    | Bootstrap files   |
|                              | fonts                  |   |
|                              | js                     |   |
|                              |                        |   |
| <b>templates</b>             |                        | Contains all HTML code  |
|                              | info.html              | Different HTML codes  |
|                              | left_middle-right.html |   |
|                              | login.html             |   |
|                              | mainpage.html          |   |
|                              | navbar.html            |   |
|                              | profile.html           |   |
|                              | register.html          |   |
| .gitignore                   |                        | File which includes all data, which should not be updated in git          |
| README                       |                        | Instruction for web-app users   |
| appconfig                    |                        | Outsourced configuration  |
| create_db.py                 |                        | Tool to create with the exampledata the database                          |
| keyserver.py                 |                        | Main source code  |
| <b>run_server.sh</b>         |                        | Includes all executing files, which has to get started to run the web-app |
| schema.sql                   |                        | Definition of the database schema   |

Tabelle 2: File structure

Finally there are some comments and explanations to the mentioned files above.

The “coursework2” directory contains the whole source code of the web-app.

The “data” directory is the place where the database is going to be located. A testing database with data testing directory can be created with the create\_db.py script.

“SQLite3” is the used database engine. “schema.sql” is the schema, which consists all important information about each user. The username is the primary key because each username exists only one time.

Afterwards there are some information about the “static” directory. This file contains all static data, for example bootstrap and pictures, that have to be stored in the file system.

The “templates” folder contents HTML codes, which describe the basic layout of the page. Here Python uses the external, already installed templates engine Jinja2.

„gitignore” specifies intentionally untracked files to ignore. In this case git should ignore the database.

The “keyserver.py” is written in Python and uses the Flask microframework and contains all the routes of the web-app and their functionality.

Run\_server.sh, by the way .sh means that this is a executing data, contains all files, which has to be run with Python. In comparison with my last web-app from my first coursework, I had to run the create\_db.py (creating the database) first and only afterwards I could run the main.py, which starts the actual web-app. So by adding this tool, it is for the web-app user a lot handier and easier to run the web-app, because the user has only to enter ./run\_server.py, which has a fixed order of executing files.

## 4. Enhancements

After visiting the web-app “Napier Public Key Server” for the first time, perhaps some users noticed that there can be added some practical tools. The following paragraphs show some nice and suitable enhancements.

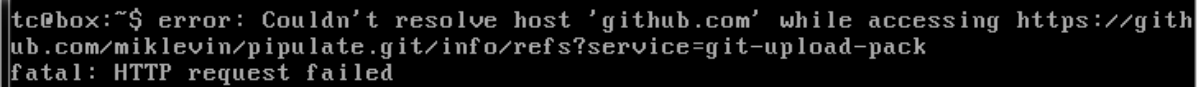
- When a user want to register, the web-app checks if the entered username is already taken. In the case that the wished username is not available anymore the web-app can suggest similar usernames, which are still available. The reason of this checking is that each username exists only one time because the username is like an identification number for each user.
- Many of these users are registered on other web services. Only to explain some possible examples, there are social networks like Facebook, Twitter, or GitHub and many other networks. Consequently it is possible to add the usernames of these networks, which each user is registered in and present is on the user profile page.
- There were made many layout improvements. But in some cases there are some anaesthetic pages on which can made some improvements. For example the startpic.png should varied by the size of the window. Therefore it is not possible to upload a own profile picture.
- By searching users, there were not shown any similar and alternative users.
- By adding a user as a friend you do not have to agree or disagree. So in case that someone do not want to be friends it is not possible to change it.
- Furthermore it would be nice if there was added a sorting function for the messages. After adding a timestamp and datum it is possible to order the messages from newest to oldest message. Or another possibility to sort the messages by senders.

## 5. Critical evaluation

- Uploading own profile pictures is not possible.
- There is missing an edit button, which enables to change some user information. For example the user wants to change his surname because of a marriage.
- Used usernames can only take once. That means when a new user wants to register, it is possible that he has to test many different usernames to find one, which is not used before.
- The search-function only shows users, which matched exactly to the searched term. So there should be paid attention to the case sensitively. In addition to that there is not included a function, which shows similar called users.

## 6. Personal evaluation

It is a pity that my Levinix did not get any internet connection. Also after talking with my tutors we could not find any solution.



```
tc@box:~$ error: Couldn't resolve host 'github.com' while accessing https://github.com/miklewin/pipulate.git/info/refs?service=git-upload-pack
fatal: HTTP request failed
```

Figure 4: Problematic Git comment

To handle that problem I installed a virtual machine, which is called “Ubuntu”. Of course that means many additional effort to install and to get to know this system but it is much easier and of course more flexible regarding the working on my second coursework. The used guideline to install the VM “Ubuntu” was an online instruction (wikiHow, 2015).

The suggested topics, which are suitable for the second coursework, represented a challenge for me. On the one hand I have to get to know what these subjects contain and represent, on the other hand to know what is important and what has to be considered by the development.

Furthermore I used many forums and tutorials, which helped me to develop my web-app “Napier Public Key Server”. Certainly I have to understand the shown and explained tools to transfer them to my web-app. That means spending a lot of time by reading and understanding because I do not attend the basic web technologies module. In chapter seven there are nearly all websites shown, from where I learned much.

Adding pictures is easier to handle because by the installation of Ubuntu there was created a directory, which is connected with windows and Ubuntu. So after copying for example pictures into this file, it is easier to upload the wished data into the static file.

By developing this challenged web-app I have written the report most time parallel. Consequently it is better structured what means that the reader can understand the way how I developed the web-app. In addition to that I tried to improve my git-commits-comments.

## 7. References

CCM Benchmark Group, 2015. *CCM*. [Online]

Available at: <http://ccm.net/contents/130-private-key-or-secret-key-cryptography>  
[Accessed 14 November 2015].

Reitz, K., 2014. *Python Guide. - Cryptography*. [Online]

Available at: <http://docs.python-guide.org/en/latest/scenarios/crypto/>  
[Accessed November 2015].

Ronacher, A., 2014. *Flask - web development, one drop at the time*. [Online]

Available at: <http://flask.pocoo.org/>  
[Accessed November 2015].

Scheuermann, B., 2015. *Databasesystems (lecture notes)*, Karlsruhe: s.n.

SQLite, 2015. *SQLite*. [Online]

Available at: <https://www.sqlite.org/>  
[Accessed November 2015].

TeachTarget, 2015. *SearchSQLServer*. [Online]

Available at: <http://searchsqlserver.techtarget.com/definition/database>  
[Accessed 15 November 2015].

wikiHow, 2015. *wikiHow*. [Online]

Available at: <http://de.wikihow.com/Ubuntu-in-VirtualBox-installieren>  
[Accessed 5 November 2015].



## References used by developing the web-app

Dr. Braun, S., 2015. *Webtechnologies (Lecture notes)*, Karlsruhe: s.n.

Dr. Wells, S., 2015. *Advanced Web Technologies (Lecture notes: SET09103)*, Edinburgh: s.n.

GitHowTo, 2015. *GitHowTo*. [Online]  
Available at: <http://githowto.com/>  
[Accessed November 2015].

Keybase, 2015. *Keybase*. [Online]  
Available at: <https://keybase.io/>  
[Accessed November 2015].

nixCraft, 2015. *nixCraft*. [Online]  
Available at <http://www.cyberciti.biz/>  
[Accessed November 2015].

Reitz, K., 2014. *Python Guide. - Cryptography*. [Online]  
Available at: <http://docs.python-guide.org/en/latest/scenarios/crypto/>  
[Accessed November 2015].

Ronacher, A., 2014. *Flask - web development, one drop at the time*. [Online]  
Available at: <http://flask.pocoo.org/>  
[Accessed November 2015].

Slack Exchange Inc., 2015. *slackoverflow*. [Online]  
Available at: <http://stackoverflow.com/>  
[Accessed November 2015].

SQLite, 2015. *SQLite*. [Online]  
Available at: <https://www.sqlite.org/>  
[Accessed November 2015].

w3schools.com, 2015. *w3schools.com*. [Online]  
Available at: <http://w3schools.com/css/default.asp>  
[Accessed November 2015].

## Bibliography

Chen, Y.-C., 2014. SPEKS: Secure Server-Designation - Public Key Encryption with Keyword - Search against Keyword - Guessing Attacks. *The British Computer Society*, 5 March.

Katz, J., 2015. *Public-Key Cryptography PKC 2015*. London: Springer.

## 8. Appendix

### 8.1 GitHub-Commits

By the developing of my biggest web-app ever, I tried to add a little more structure. The following paragraphs show the commits and give short explanations to each commit, why or how I change the code.

#### **1 initial commit:**

- clone the coursework file and found the git-directory

#### **2. Login and Logout function**

- defined the basic functions, which my web-app should have.

#### **3. Added errorhandler and appconfig**

- errorhandler shows users errors e.g. by entering a URL, which is not available.
- created an outsourced appconfig

#### **4. installed bootstrap**

- Adding design is a lot easier and handier than writing CSS and HTML code. In addition to that there are existing many different tutorials and examples how you can handle it.

#### **5. Loaded appconfig**

- "For small applications it's a possibility to drop the configuration directly into the module which we will be doing here. However a cleaner solution would be to create a separate *.ini* or *.py* file and load that or import the values from there" (Ronacher, 2014).
- Here I write the config into a separate file.

#### **6. updated config with port number**

#### **7. created database schema**

- By the developing of the web-app of the first coursework, I added and used a database, too. Consequently I had a tiny knowledge about how to build one. But the problem was not only to add a database into the web-app but also how it should looks like.

#### **8. created .gitignore**

- „gitignore" specifies intentionally untracked files to ignore. In this case git should ignore the database.

- In this folder I added all the files and data, which should not upload to gitHub.com. In case I did not do that, my account would “explode” because of the data of the database, which would be uploaded every time I push data on it.

### **9. added example data for the database**

- Created exampledata file, where I add three examples. These three users helped me by developing the web-app.

### **10. implemented methods for database creating, opening and querying**

### **11. Adapted run\_server.sh script to initialize the database before starting**

- In my last coursework - the developing of a music catalogue - I had to run the database.py first and afterwards I could start the main.py.

- This time I looked for a better solution because I was not sure how many databases I will need. Then I found the idea to create an executed file, which is called run\_server.sh.

### **12. store the hashed user password in the database**

- pycrypto „is a collection of both secure hash functions (such as SHA256 and RIPEMD160), and various encryption algorithms (AES, DES, RSA, ElGamal, etc.)” (Python Software Foundation, 2015).

- Hash-functions are encryption (hashes sind nicht zum verschlüsseln → also das wort einfach löschen) procedures, which make information safety. By mathematic algorithms it is possible to create a one way function, which is not reconstructable. So this function – a one-way-function - changes the password in an appropriate code, which changed by the littlest variation a completely different code.

- By adding the encryption code to my web-app, the workbook was very helpful.

### **13. store private key encrypted**

- By the help of Crypto.Cipher and base64, which I copied from the workbook, it is possible to store the user private key encrypted (Wells, 2015).

### **14. changed database schema of friends table**

- Here was changed the database schema. Now the user can see on his profile with who he is connected. That means to show the relationship status, friends or not.

### **15. basic page layout incl navbar**

- Created the basic page-layout with bootstrap. For me it was easier to write the bootstrap code on the w3school.com tutorial website. There it was easier for me to have and get an overview.

#### **16. added login**

- In this commit the login.html is comparable to the "old" one from the first homepage.

#### **17. added logo**

- By using Ubuntu as virtual machine it was a lot easier to add and upload pictures than with Levinux.

#### **18. implemented login**

- Here was not only added the login but also the left-middle-right.html. This HTML page represents a design change because it suits better to idea how this web-app should work and look like.

#### **19. Added terms and conditions of use**

- Nearly every web-app, where you have to register, the user have to read the terms and conditions before he can be registered. So in this step I added a info.html, which opens when the user click on the terms and condition text in the footer.

#### **20. Created register form**

- This commit contains the adding of the register.html and that means a lot of code because there has to be made some ideas about how the register procedure should look like.

#### **21. improved layout**

- Renew some HTML files by adding the left-middle-right.html.

#### **22. return to mainpage when clicking on logo**

- By clicking on the logo the user gets back to the start page, which is called mainpage.html. That means that the logo.png is linked with the mainpage.html.

#### **23. further layout improvements**

- Changed and improved the layout

#### **24. Mainpage**

- Here was added the startpic.png to the mainpage.html and in addition to that there was made some improvements on the mainpage.html.

#### **25. Implemented registration process**

- This commit was a little challenge because there has to be added some encryption data. Not to forget are the different possibilities by registering. For example entering wrong or already taken usernames or forget to accept the terms and conditions.

#### **26. store name of logged in user in session**

- Is needed for displaying the right content on the webpage.

#### **27. show first information on user's profile page**

- This commit contains the showing of some and basic user information.

#### **28. check if password only contains valid letters**

- After testing the previous web-app, I noticed that some letters cannot be used. In this step there were checked if the entered password contains any of these unsuitable letters.

#### **29. show the profiles of friends and other users**

- Added in `"/logout"` a definition and created a new `@app.route`, which is called `"/user/<username>"`.

#### **30. show a gender-specific profile picture**

- After adding the gender for each user, there were uploaded two specific user profile pictures.

#### **31. insert navigation to profile into navbar**

- Here was added the missing `"My profile"` button, which was missing.

#### **32. added more example data**

- This commit was made because of testing and layout reason.

#### **33. added possibility to add users as friends**

- After adding a variable, which represents the relationship between the users, it is possible to add a user as friend in case that they had not be friends before.

#### **33. improvement of profile layout**

- Changed some sizes and entered some nice layout tools.

#### **34. search for users**

- Here search function for users was added. It is similar to the one, which had used in the coursework before.

#### **35. added for form for sending secure messages to a user**

- This commit creates a message panel on the user profiles.

### **36. Possibility to lock profile so that messages are only shown encrypted**

- By adding two routes “/profile/lock” and “/profile/unlock” the user can encrypt the messages. The reason why this is handy is that when a user has to leave his computer or laptop alone. Consequently no one else can read the secure message. Only after unlock the profile the user can read it again.

### **37. display encrypted messages**

- To add and show encrypted messages there were needed some messages. Therefore there was created further exampledata, which is called messages.example. At the same time there has to be made many changes like for example in the database schema.

### **38. show decrypted messages after entering the password again**

- adding the function that the user can encrypt and decrypt his messages on his profile.

### **39. Fixed error**

### **40. Fixed error with missing messages**

### **41. Send secure messages to other users**

- Adding the new route “/user/<username>/send” to enable the users to send secure messages to other users.

### **42. Fixed error when sending special characters**