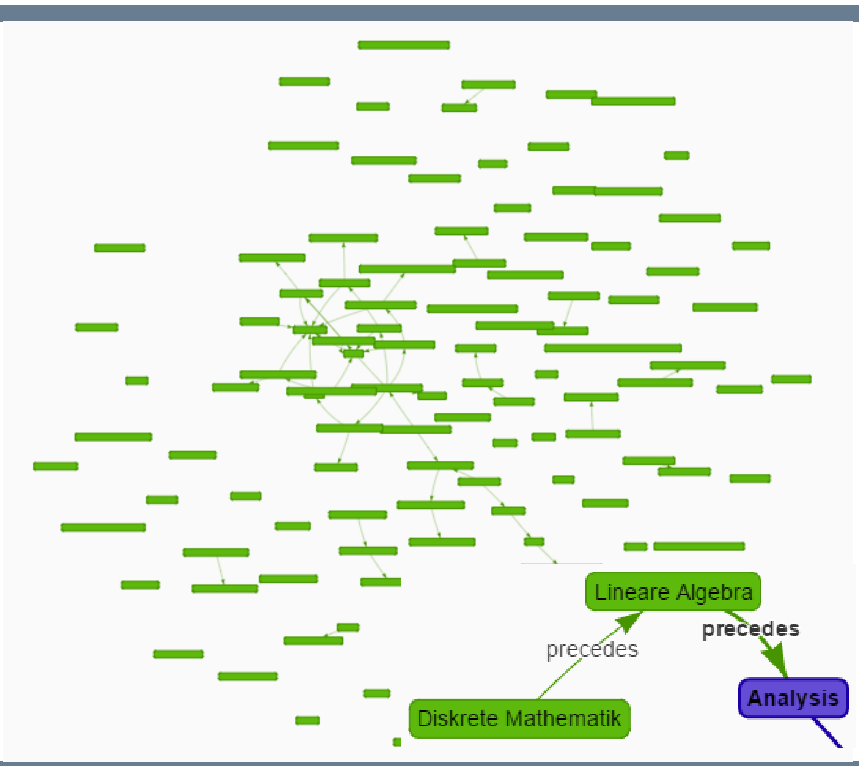

StudGraph (Neo4j)

Project report

Module BTI7504p Semantic Web

Course of studies:	Computer Science
Authors:	Steven Cardini, Sebastian Häni
Supervisor:	Prof. Dr. Olivier Biberstein
Date:	17.06.2016

1 Introduction

The assessment of module *BTI7504p Semantic Web* consists of two parts: a final exam as well as a project. In this report we describe the results of our project *StudGraph*.

The suggested task was to *study Neo4j, especially the links between Neo4j and RDF (storage, exportation, ...)*. We were however free to adapt the project goals to our individual ideas.

Since we were both new to Neo4j we started by learning the basic functionality of what is today *the world's leading graph database* according to the makers' own statements [http://neo4j.com]. As we experimented with different examples, the idea of what we wanted to achieve in this project became clear. We decided neglecting to investigate the link between Neo4j and RDF and instead study the link between a web application and a Neo4j database.

The goals of this project were

1. to implement a Neo4j database containing information about the modules of the computer science degree course at BFH as well as some relationships among them such as information about precedence, exclusiveness and hierarchy, and

2. to develop a web application that queries this database and presents some information about the modules to the user.

2 Implementation

Since we were a group of two students, we had to split tasks. We used Trello [https://trello.com] as our project management tool. Trello supports a Kanban workflow [https://www.atlassian.com/agile/kanban]. This way, we had a standard task board with columns. Even though this project is rather small, it was still helpful to see the progress of the other team member. It was also very fast to set up.

We started by defining what information our database should store. We decided that it should contain the following information:

- all modules of the computer science degree course at BFH, including:
    - the id (e.g. *BTI7054*)
    - the German name
    - the French name
    - the number of ECTS points

- hierarchy (module types A, B, C, D)

- module types (compulsory or elective modules)
- assessment types (E, Pa or Pb)
- dependency of modules among each other (some modules have to be completed before another can be started)
- XOR-relationships (students sometimes have to choose one or another module)

The statements to insert this data into a Neo4j database (and some example queries) can be found in folder */cql* of our git repository.

After filling our database with the corresponding data, we developed a web application that is able to display some of this information to the user.

Our web application consists of three layers:

1. a web frontend:
    - Implemented with JavaScript ES2015 and CSS.
    - We used the following frameworks to visualize the graphs:
        - react [https://facebook.github.io/react],
        - react-redux [https://github.com/reactjs/react-redux],
        - redux-saga [https://github.com/yelouafi/redux-saga],
        - webpack [https://webpack.github.io],
        - VisJS [http://visjs.org].
    - The following utilities are used to lint the code:
        - eslint [http://eslint.org],
        - jscs [http://jscs.info],
        - stylelint [https://github.com/stylelint/stylelint].
    - The frontend was built with webpack.
    - We ran unit tests using karma [https://karma-runner.github.io].

2. a PHP middleware:
    - The middleware takes requests from the frontend, forwards them to the Neo4j server, transforms it into a useful and friendly format and returns it to the frontend.

3. a Neo4j database

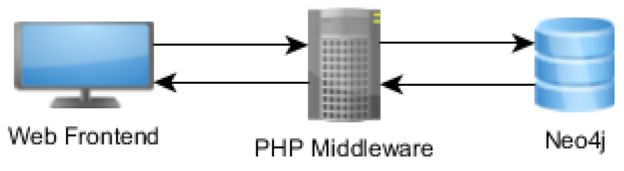

Figure 1 | This illustration shows the technology stack we used for our web application

Our final application features the following functionalities:

- Free text search for modules
- Module details
- Dependency graph
- Hierarchical graph
- Overview graph

3 Installation

The easiest way to test our application is to browse to our demo site
http://studgraph.herokuapp.com/

We deployed the frontend as well as the middleware of our application to the cloud application platform Heroku [https://www.heroku.com]. Our Neo4j server runs on the Neo4j cloud hosting platform GrapheneDB [http://www.graphenedb.com].

Please note that we use free web hosting and the performance may be poor upon the first requests (the server hibernates; there can be some error messages).

To install a local version of our web application, the following prerequisites need to be met:

- Neo4j
- PHP
- NodeJS

At first, the Neo4j database should be filled with our data from the file */cql/01_insert-data.cql*. The password of the database should be *1234*.

The frontend can then be installed using the following commands in the root directory of the git project:

```
$ npm install
$ npm start
```

Finally, the middleware can be initiated using the statements:

```
$ cd app/
$ php -S localhost:8000
```

4 Conclusion

We learned a lot about Neo4j databases, which to us represent an interesting alternative to relational databases in order to store information.

Our web application is built in a way that can be easily extended in the future. One interesting possibility of further development is to include a functionality to automatically generate a schedule over all semesters for a student, taking into account his individual preferences.