Designing a user interface for diploma configurations in RWTH Online

Bachelor's Thesis
submitted to the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Valentin Engelke

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof Dr. Ulrich Schroeder

Registration date: 05.02.2020
Submission date: 31.03.2020

Eidesstattliche Versicherung

Engelke, Valentin	358096
Name, Vorname	Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende ~~Arbeit~~/Bachelorarbeit/ ~~Masterarbeit~~* mit dem Titel

Designing a user interface for diploma configurations in RWTH Online

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen, 31.03.2020	V. Engelke
Ort, Datum	Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Aachen, 31.03.2020	V. Engelke
Ort, Datum	Unterschrift

Contents

List of Figures

List of Tables

Abstract

Due to the growing complexity involved in administrating a university, universities rely on campus management systems, that help with processes such as application management, course management and exam management. RWTHOnline is one such system introduced by RWTH Aachen University in 2018.

While RWTHOnline is a useful tool for the university administration, some parts of the system have usability problems. In this thesis, we take a look at the process involved in changing the diploma configuration table, that determine how the modules of a course are displayed on diplomas given out by the university. Currently the user interface does not support this process and the users have manually design the tables without being able to preview the resulting diplomas.

In this thesis we develop and evaluate an interface for diploma configurations in RWTHOnline. We implement the interface as a python web app using the framework Flask. The resulting software is then evaluated using both heuristic evaluation and two user evaluations.

Überblick

Auf Grund der zunehmenden Komplexität der Verwaltung von Universitäten, benötigen Universitäten Campus-Management-Software, die bei der Verwaltung von Bewerbungen, Kursen oder Prüfungen helfen. RWTH Online ist eines solcher Systeme, das von der RWTH Aachen im Jahr 2018 eingeführt wurde.

RWTHOnline ist zwar für die Universitätsverwaltung ein nützliches Tool, einige Teile des Systems haben aber Defizite bei der Benutzerfreundlichkeit. In dieser Bachelorarbeit werfen wir einen Blick auf den Prozess beim Ändern einer Report-Konfigurationstabelle, die festlegen wie Module eines Kurses auf Zeugnissen, die von der Universität ausgegeben werden, dargestellt werden. Im Moment hat die Benutzeroberfläche keine Funktion dafür und die Nutzer müssen die Tabelle manuell erstellen ohne die Möglichkeit zu haben, die Zeugnisse die aus den Tabellen entstehen, vorher anschauen zu können.

In dieser Bachelorarbeit entwickeln und evaluieren wir eine Benutzeroberfläche für die Konfiguration von Report-Konfigurationen in RWTHOnline. Wir implementieren die Benutzeroberfläche als Python Web App unter Benutzung des Frameworks Flask. Um die Software zu evaluieren nutzen wir sowohl eine heuristische Evaluation als auch zwei Evaluationen mit Nutzern.

Acknowledgements

First, i want to thank Simon Völker for being my supervisor.

Furthermore, i want to thank Prof. Dr. Borchers and Prof. Dr. Schroeder for examining this thesis.

Chapter 1

Introduction

1.1 RWTHonline

RWTHonline is the campus management software used by RWTH Aachen University. The software is used in ”application and admission management, student and fee management, course and classroom management as well as exam management” [1], both by students as well as the university administration. For each examination regulation the system contains a tree of nodes, where each node corresponds to a module area, module or exam within that examination regulation. Each of these nodes contains information such as the number of ECTS credits a student receives. We focus on a table stored in each node that determines how information about the node is displayed on different types of reports, for example the transcript of records.

Currently the person responsible for each module has to manually design the table for each node and send it to the university administration. There is no way to preview how the reports resulting from the configuration of the report display look like before changes are processed within the

[1] ttps://www.rwth-aachen.de/cms/root/Die-RWTH/Einrichtungen/Verwaltung/Dezernate/Akademische-und-studentische-Angelegenhe/ dtsuc/Abteilung-1-6-Student-Lifecycle-Manage/?lidx=1

administration and the next reports are printed, which can take many months.

1.2 Goals

The goal is to build an application, that allows users to download the course data from RWTHOnline, locally make changes to the configuration of the report display while previewing the reports resulting from the report configuration, and finally export the changes made in an easy to read format.

1.3 Outline

We first analyse the current situation and speak to users of RWTHonline to discover the problems resulting from the present system. We then design multiple iterations of paper and, in later stages, software prototypes. Next we discuss the decisions we made when implementing our software solution and the technologies used. Finally we conclude with a user study to test wether our solution solved the users problems.

Chapter 2

Related work

There are a lot of works related to usability of software. An early book that gives an overview of the area is "Usability Engineering" by Jakob Nielsen [Nielsen, 1994]. While technologies have changed since then, the book presents fundamental principles for the development process that are still relevant:

- Know the User

  Study how the user currently solves their tasks, interview them, keep in mind their previous experiences and individual abilities

- Create prototypes to quickly be able to evaluate interfaces with users

- Continually evaluate and adapt the software based on user studies

Furthermore, the book features advice on high-level features of user interfaces such as:

- minimizing the memory load of the user when using the software

- be consistent across different parts and versions of the software

- provided the user with clear and fast feedback

Finally the book also contains information on preparing and conducting user studies.

Chapter 3

Background

3.1 University Reforms

In 1999 education ministers from 29 european countries signed the Bologna declaration, in which they pledge their support to creating a unified ”European Higher Education Area” (EHEA). The declaration outlines the goals of the signatories:

- the degress awarded by universities in the EHEA should be comparable
- the system should be split in two cycles, Bachelor and Master
- introduction of the credit point system ECTS
- allow for mobility of both students and university personell
- support intra-european cooperation

The necessary changes to achieve these goals were, and are continuing to be, implemented by each country with respect to their respective laws and their current national education system, while still working together with the other countries.

3.2 Modularisation

A consequence of the Bologna process is the modularisation of degrees, to allow both for more transparency and comparability, aswell as for students to be able to move during their bachelor or master degree (e.g. Erasmus).

Implementations of the reforms vary by country, and in Germany also by state, however we will take a closer look at the situation in North-Rhine Westhphalia, where RWTH Aachen is located.

The "Studienakkreditierungsverordnung" (StudakVO) lays out the requirements for a university degree to be accredited in the state. According to §7 degrees have to be split into modules by topic and in certain time limits.

The description for each module has to contain informatiom about:

- the topic and goals
- form of learning (lecture, seminar etc.)
- requirements for students to participate
- applicability of the module
- requirements for ECTS credit points to be awarded
- ECTS credit points awarded and grading rules
- how often the module is offered
- amount of work
- duration

3.3 Campus Management Systems

From these conditions arises the need for universities to have a system that can help with the processes involved in

university administration, while complying with all laws and ordinances. The systems used at RWTH Aachen prior to the introduction of RWTHOnline were not optimal in fulffilling the challenges posed by the Bologna process [Grzemski and Decker, 2018]. Aditionally the change to RWTHOnline provided an opportunity to unify all parts of student-life-cycle management in one system, while previously the exam registration and the organisation of events where split into multiple systems developed by different companies.

RWTHOnline is a system originally developed by TU Graz in Austria and adapted for use by RWTH Aachen. It was introduced university-wide in WS2018/19. Among other parts, it integrates application management, exam registration and management of courses of studies.

Chapter 4

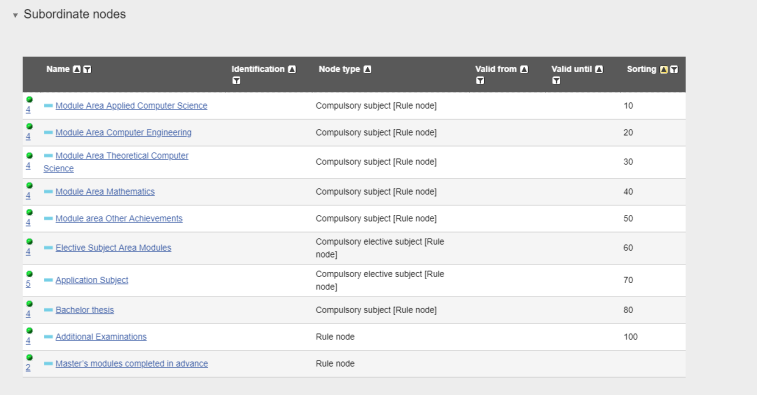
Record Table Management

In the following chapter we present the current functionality of record table management in RWTH Online and the problems that come with it.

4.1 SPO Management

A large part of RWTH Online is SPO (Studien- und Prüfungsordnung, engl. Academic and Examination Regulations) Management. It contains information about each course offered by the university. Each course is represented as a tree of nodes, where each node can for example represent a module area, module or exam. These tree structures can be very large, the bachelor of computer science course tree contains more than 3000 nodes.

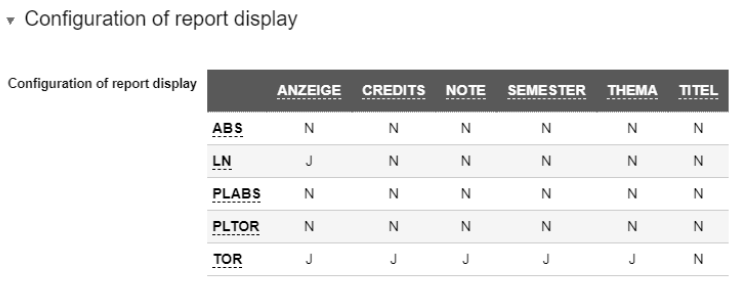
Each node contains a variety of information, including the name and the amount of credit point awarded. We are most interested in the "configuration of report display" table. This table, which is part of each node, determines how the corresponding module or module area is displayed on different types of diplomas.



Subordinate nodes

	Name	Identification	Node type	Valid from	Valid until	Sorting
4	Module Area Applied Computer Science		Compulsory subject [Rule node]			10
4	Module Area Computer Engineering		Compulsory subject [Rule node]			20
4	Module Area Theoretical Computer Science		Compulsory subject [Rule node]			30
4	Module Area Mathematics		Compulsory subject [Rule node]			40
4	Module area Other Achievements		Compulsory subject [Rule node]			50
4	Elective Subject Area Modules		Compulsory elective subject [Rule node]			60
5	Application Subject		Compulsory elective subject [Rule node]			70
4	Bachelor thesis		Compulsory subject [Rule node]			80
4	Additional Examinations		Rule node			100
2	Master's modules completed in advance		Rule node			

Figure 4.1: The list of subordinate nodes of the root node of the bachelor of computer science course



Configuration of report display

Configuration of report display

	ANZEIGE	CREDITS	NOTE	SEMESTER	THEMA	TITEL
ABS	N	N	N	N	N	N
LN	J	N	N	N	N	N
PLABS	N	N	N	N	N	N
PLTOR	N	N	N	N	N	N
TOR	J	J	J	J	J	N

Figure 4.2: An example of the ”configuration of report display” table

Each row represents a different type of diploma (e.g. final diploma, certificate of academic achievement), whereas each column represent a certain information (credit points, grade, semester, topic etc.). Each field of the table contains either a ”J” (standing for ”Ja”, german for yes) or an ”N” (standing for ”Nein”, german for no). This boolean value determines wether the information specified by the column is displayed on document corresponding to the row of the field.

This allows a variety of different configuration options. For example, the university might want to display the topic of a bachelor thesis on a final diploma, but not the topic of every seminar the student has taken.

4.2 Current Situation

Currently, changes to the tables can only be made by certain divisions (Divisions 1.5 and 1.6) of the university administration. This means, that if a person in a university faculty wants changes to be made to a course belonging to the department, they can't make the changes themselves, but have to first send those to the divisions responsible. This involves manually creating the tables without having any way preview the effects of those changes until the next time diplomas are printed, which can be several months.

Then, those changes have to manually entered back into the system by the university administration. All in all, this process is very inefficient and creates a lot of work for everyone involved.

4.3 Stakeholders

Multiple parties currently hold stakes in the process:

- Members of university faculties

  need to decide on the record table configurations for nodes of courses belonging to their area

- Division 1.6 (Student Life Cycle Management) of the university administration

  The department is responsible for developing and supporting software including RWTHOnline

- Modeling Team of Division 1.5 (Examination and Statutory Law) of the university administration

  The team is responsible for modeling the examination regulations, and therefore also the contact if faculties want to changes the record configuration tables.

Chapter 5

Prototyping

To design the interface of the applicaton we went through multiple iterative stages of prototyping.

5.1 Paper Prototypes

We started of by creating storyboard-like paper prototypes of the different parts of our user interface. Paper prototypes are useful in early stages of prototyping, since they allow for quick evaluation of high-level design choices.

We evaluated these protoypes with a user who is experienced with the topic and knows the process currently involved in changing the diploma configurations. The user was happy with the general direction of our design approach and we made some changes based on the user feedback.

The resulting design from those prototypes is very similar to the interface we actually ended up implementing in the software.

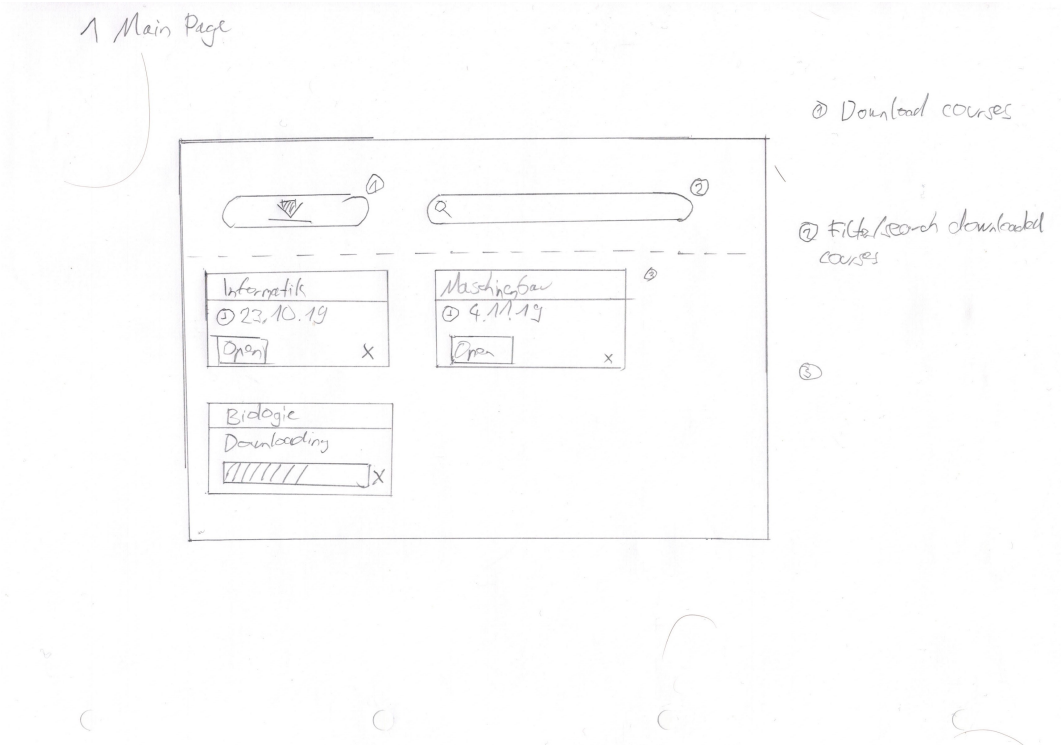


Figure 5.1: Prototype of the start page

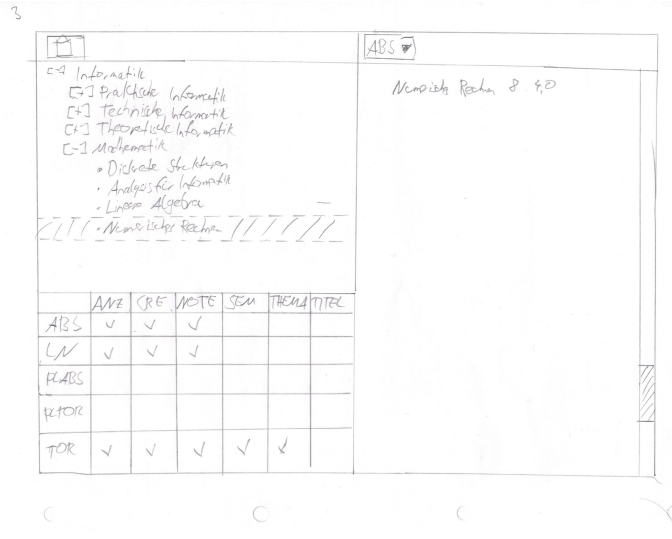


Figure 5.2: Prototype of the edit page

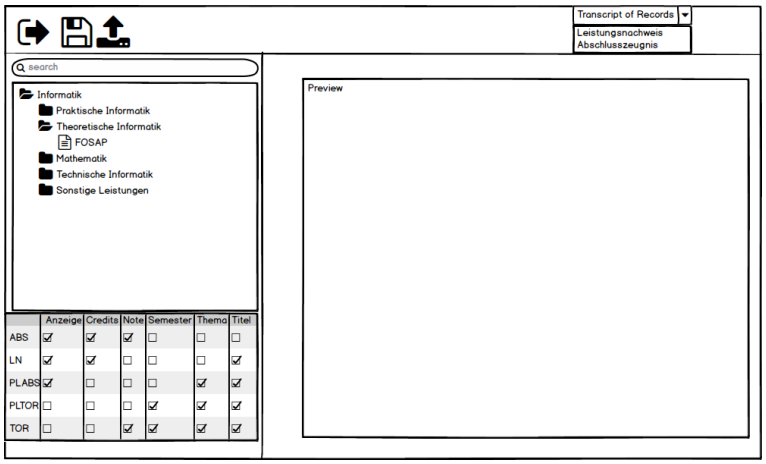


Figure 5.3: Mockup in balsamiq

5.2 Further Prototypes

Furthermore, in later stages of prototyping, we also created a higher fidelity mockup using the design tool balsamiq.

Chapter 6

Implementation

This chapter provides an overview on the design of and the technologies used in the implementation of our user interface for diploma configurations.

The source code can be found on Github[1]

6.1 Requirements

Our prospective users are using a variety of different platforms. For example, Windows is used in university administration, while the Media Computing Group almost exclusively uses macOS. We therefore want our software to be available across different platforms.

Because our user base consists of a small number of people who are already familiar with the domain of the application, we don't need to explain the basic concepts of diploma configurations and the different types of diplomas in detail but just provide an easy to use interface that the users can quickly learn.

Our program has to allow the user to access the course data and existing diploma configuration tables from RWTHOn-

[1] https://github.com/VEngelke/rwthonline-diplomaconfig

line, edit them locally, preview the resulting diplomas, and finally export the changes made in a practical format.

6.2 Python

We decided to use Python to develop the application. Python allows us to run the same program on Linux, Windows and MacOS devices. Additionally familiarity with the language was a factor

Flask

We used the web framework Flask for the front end of our application. This allowed us flexibility to deploy our application either as a desktop app with the server running on the same machine as the client as well as host the application on a server, to allow our users to access it with their web browsers. Also, using a html based interface allows for easy cross-platform usage. We decided to focus on the desktop app approach, but only small changes are required to make the software usable when hosted on a remote server.

We also considered other options for developing the user interface. A popular alternative to Flask in the realm of python web frameworks is Django. While there are many similarities between the two, the philosophies behind them diverge. Flask considers itself to be a so called "microframework", meaning that the aim is to have a small core of the framework that can be extended easily by the user to achieve the functionality the user needs. In contrast, Django by default already has many features like a database or a admin panel. For our purposes, we decided that we did not need most of the features that Django possesses and therefore went with Flask.

Another option would be to go with a classic GUI toolkit like Tkinter or PyQt. However we saw multiple advantages in going for a web based approach compared to the GUI way:

- Ability to host the program on a server, no installation

on the users side required

- more easy to use cross-platform
- many GUI frameworks look quite dated
- ability to have a similar design to RWTHOnline

6.3 Web Scraping

There is no public API to access information from RWTHOnline. We therefore have to manually scrape all the course data we need.

Each node, corresponding to a module or module area, has a unique identifier named "pStpKnotenNr". Requests to the URL "https://qs-online.rwth-aachen.de/RWTHonline/pl/ui/$ctx;lang=de/wbSPO.cbSPOContent?pStpKnotenNr=3197" return the information belonging to the node with the ID 3197, in this case the root node of the bachelor of computer science course. The response is a XML File containing a mixture of HTML and Javascript.

Due to the complex structure of the XML response we decided against parsing XML and instead opted for using regex to extract the information we needed. For each node we were interested in:

- the name of the node
- the amount of credits the module the node belongs to awards
- the record table configuration of the node
- the validity dates (start and expiry date) of the node
- the identification number of the node (differs from the ID used to request the node)
- the IDs of all child nodes

We then recursively fetch the information on all child nodes until we have the complete tree of nodes belonging to the course. In some cases the number of child nodes a node has, means that not all child nodes are displayed at once, requiring more requests to get all.

Since courses can contain many nodes, for example the computer science bachelor contains around 3000, and the response to each request is quite big, generally between 30kB and 150kB, we need to locally cache a course, before our users can edit the record table configurations and see a preview of documents. Since we are only interested in a small amount of information from each node, the file size of the saved courses are tiny compared to the total size of the responses to our requests.

6.4 Features

Start Page

The start page features a list of courses, that the user has dowlnloaded previously and a search bar to filter those. Each course is represented by a ”card”, that features information about the course such as the name and the date it was downloaded, as well as a button for opening the course for editing and deleting a saved course. Additionally there is a ”download” button that the user can press to start the process of downloading another course.

Download courses

To download a course from RWTH Online the user first has to log in. After successfully logging in, the user is redirected to a page where they can search through the available courses. The user can click on each result to start downloading a course.

Celery

Since the download can take quite a long time (up to

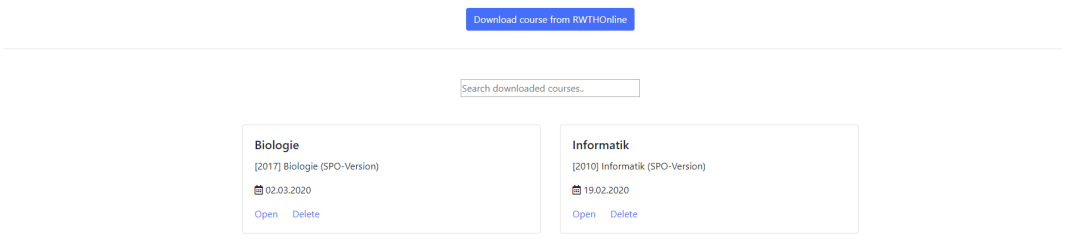


Figure 6.1: Start page

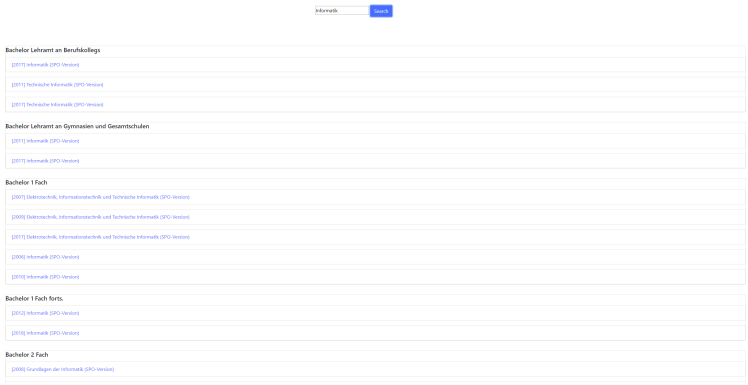


Figure 6.2: The user can enter a search term in the text field and gets a list of all courses matching the search as a result

around 15 minutes for large courses), it is necessary to do the download in a separate process to the flask server. To achieve this we use Celery, a asynchronous task queue library for python. When starting the flask server, we also start a Celery worker in the background that monitors the task queue and spawns the processes that actually perform the download. This means, that during the download duration, the main flask thread is free to respond to other requests.

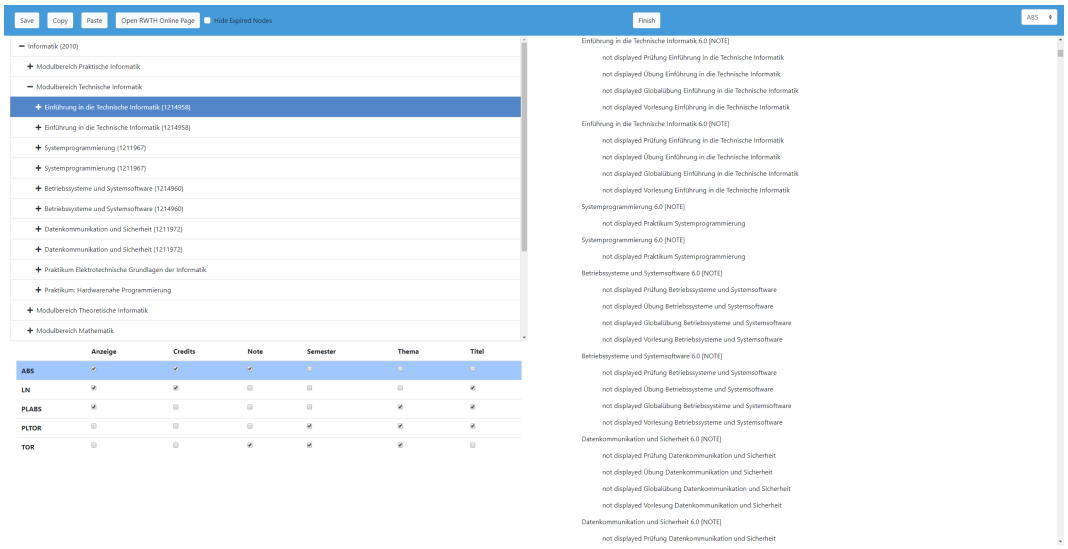


Figure 6.3: Edit page

Edit page

The edit page is the most important part of the interface. It consists of three main elements.

On the top left, there is a tree view containing each node of the course the user has opened. Each node can be expanded to view the subordinate nodes by clicking on the ”plus”-sign next to the name. The user can select a node by clicking on it.

On the bottom left, the diploma configuration table of the selected node is displayed. The user can change the configuration by clicking on the checkboxes in the table.

On the right half of the interface a preview shows the user what the diploma resulting from the current configuration of the tables looks like. The preview automatically scrolls to the corresponding row when a user selects a different node. The toolbar contains a dropdown where the document to be previewed can be selected, but the selection also automatically changes to the corresponding document type when the user clicks on a checkbox in the diploma configuration table.

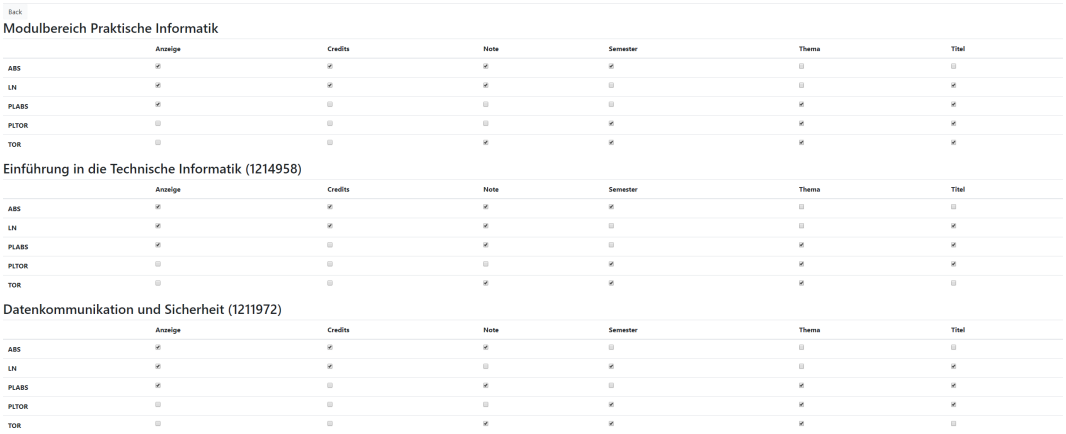


Figure 6.4: Overview page

Copy/Paste

The toolbar also contains two buttons for copying and pasting record tables, since analysis of the courses has shown that they contain many identical tables. To use this functionality, the user selects the node, whose table they want to copy, press the copy button, then select the node they want to copy to and finally press the paste button.

Overview

When the user has finished their edits, they can press the finish button to get an overview of the changes they made. This overview page can then easily be saved as a pdf using the browsers inbuilt tool.

We considered adding the option to export the changes in some kind of standardised machine-readable format, however no matter what, the changes would in the end still have to be added into the system by humans, so we did not see an advantage by adding this feature at the present point in time.

Chapter 7

Evaluation

We used two methods to evaluate our software and discover any remaining usability problems. We decided on first conducting a heuristic evaluation before

7.1 Heuristic Evaluation

A heuristic evaluation consists of one or multiple experts looking at a user interface and judging it by usability heuristics ([Nielsen, 1994]). These heuristics include general heuristics (i.e. the ten basic heuristic listed below), but can also include heuristics specific to the demain of the system ([Nielsen, 1995]).

7.1.1 Advantages and Disadvantages

Heuristic evaluations don't require user participation, which makes them an attractive option in the early stages of development. They only require one expert that is familiar with design heuristics and how to apply them.

Heuristic evaluation is an easy method for evaluating a user interface

However an experiment presented in [Nielsen and Molich, 1990] shows, that experts, when evaluating a user interface, find a varying number of usability problems, but experts,

Use multiple experts to find more usability problems

that overall perform worse, could also discover problems that some better experts did not. Therefore, the authors conclude, that to get better results from heuristic evaluations, multiple experts should independently conduct the evaluation after which the individual results should be consolidated. While, for each of the 4 user interfaces evaluated in the experiment, individual experts on average found only found between 20% and 51% of usability problems, on average aggregating the results of 5 evaluators resulted in a much increased proportion of found problems, between 55% and 90%.

7.1.2 Heuristics Used

We use the ten basic heuristics presented in [Nielsen, 1994].

- Simple and Natural Dialogue

  User interfaces should only be as complex as absolutely necessary. The gestalt rules should be followed and the user task should be mapped to the interface in a natural way.

- Speak the Users' Language

  The terminology used in the user interface should match the user's language. Don't use technical terms the user is not familiar with.

- Minimize User Memory Load

  The user should not be required to remember a lot of information to use the system.

- Consistency

  The interface should be consistent, both within different parts of the application (e.g. the same information should be presented in the same way and the same position across different parts of the application), as well as across time (the same action should always produce the same results).

- Feedback

The system should always give clear feedback on what it is doing and what the effects of a users action were. This feedback has to be fast enough (ideally less than 0.1 seconds) for the system to feel responsive.

- Clearly Marked Exits

  Users should be able to cancel actions and be able to get from any state to a previous state (e.g. back button or undo button).

- Shortcuts

  The system should include shortcuts that make it easier and faster to use

- Good Error Messages

  Error messages should be easily understandable for the user without having to use a manual. They should help the user in solving the problem.

- Prevent Errors

  The user interface should support the user in avoiding errors (e.g. in a command-line interface provide the user with a list of possible commands).

- Help and Documentation

  While the goal should be for the user to be able to use the interface without consulting a manual, the information the user requires should be accessible quickly if the necessity arises.

7.1.3 Usability Problems Found

1. Layout of Course Overview Screen

The text field for searching through already downloaded courses is placed right next to the ”Download” button that starts the process of downloading a new course.

Due to the gestalt law of proximity the user might think, that the two interface elements are related to each other and therefore, for example, they have to enter the name of the

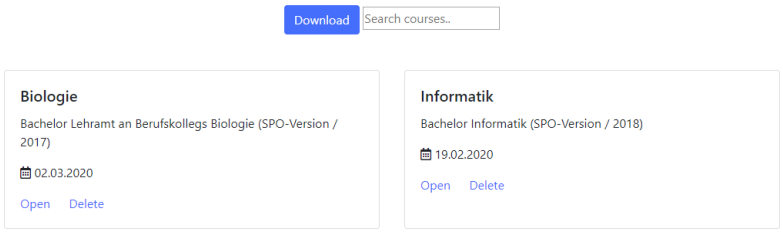


Figure 7.1: Start page before

course they want to download into the text field. The placeholder text also does not help to clarify the function of the text box.

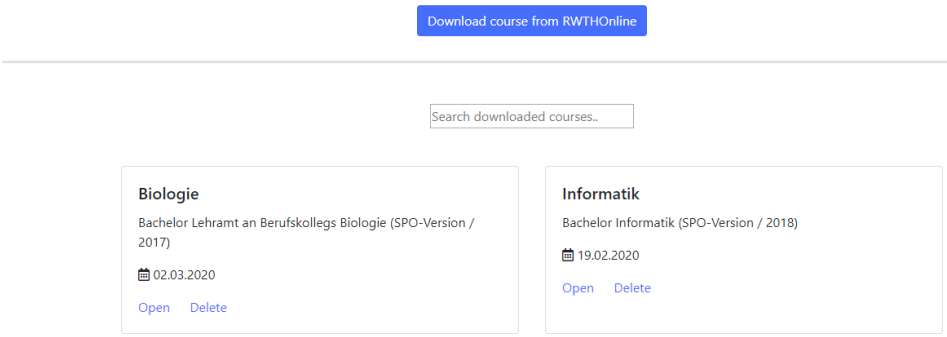


Figure 7.2: Start page after

We therefore moved the text field closer to the list of downloaded courses that it interacts with. We also added a border between the area of the "Download" button and the area below to emphasize the separation. Finally, we changed the placeholder text to make clear, that the text field is used to search through already downloaded courses.

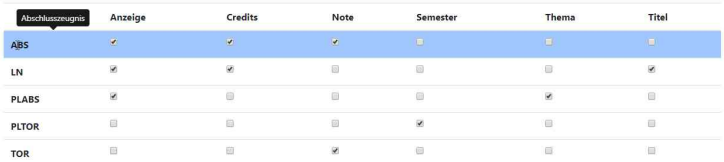


Figure 7.3: The full document type now shows on hover

2. Full Names of Document Types on Edit Page

The report configuration table on the edit page only features the abbreviations of document types as labels (e.g. ”ABS” instead of ”Abschlusszeugnis”). While these abbreviations are also used in RWTHOnline, the full names should also be available to the user (heuristic ”Speak the Users’ Language”).

We added the ability for the user to hover over the abbreviation to see the full name.

3. Back button from overview screen

There is no ”back” button from the overview screen after the user has made the changes they wanted. If they use the browser button to go back they lose all the changes they have made to the report table configurations. This is not in compliance with the heuristic of always providing users with clearly marked exits.

We added a ”back” button to the overview screen which takes the user back to the edit page with all the changes they have made retained, so they can continue editing.

In addition to that, we added a button that takes the user back to the course overview screen.

7.2 Evaluation with users

Finally, we conducted user evaluations with two different users.

In both cases, the users were tasked with downloading a course from RWTH Online, making specific changes to the diploma configurations and reviewing the changes made on the overview page.

First user evaluation

The first user study was conducted before the changes we had decided to make after the heuristic evaluation were implemented. The user was not familiar with the domain of university administration and RWTHOnline specifically, but very experienced with using web interfaces in general.

We first gave the user a small introduction to the background of the system and their task but did not explain the interface to them. After that we just observed the user.

To download the course, the user first tried to enter the name of the course in the filter text field on the start page and then press the ”download” button. This clearly showed us, that this issue we discovered using the heuristic evaluation was not purely of theoretical nature but needed to be fixed .

The user nevertheless then managed to successfully navigate the user interface and download the course.

The user completed the other tasks without any issues.

In the interview after the user study, the user critisised, that the changes they made were gone after going back from the overview screen. This was an issue we also already discovered in the heuristic evaluation and fixed after the user evaluations.

Second user evaluation

The second user study was conducted with a user who has used RWTHOnline a lot and has experience with the diploma configuration feature specifically. The aim of this evaluation was to see whether our result matched their expectation and needs.

The user had the same task as in the first evaluation.

The user completed the task without any problems and was happy with the application we developed.

Chapter 8

Summary and future work

In this final chapter we give a summary of the thesis and an outlook on what future work could be done.

8.1 Summary

In this thesis, we started out by taking a look at RWTHOnline and particularly the diploma configuration feature in SPO-management. We analysed the problems in the current process involved in changing it and set out to develop a user interface that would support users in this process. Subsequently we went through multiple stages of prototyping before implementing our final design using python and flask. Finally we evaluated our end product using multiple evaluation methods. We concluded that our interface matched the users expectations and helped them in their task of changing diploma configurations.

8.2 Future work

We would like to evaluate the current version with more users. We only tested it with one user that was part of the target audience. Due to scheduling issues we were not able to evaluate it with people that are in the department responsible for developing RWTHOnline (Department 1.6) and get their feedback on our ideas.

Currently our system is configured to be run directly on a users computer. Hosting it on a central server would make it more easily accessible to our intended users. This however requires some additional features, most importantly a user account feature that ensures that users can only view courses they have permission for in RWTHOnline.

The user interface we developed, or a system similar to it could be directly integrated into RWTHOnline. This would provide an opportunity to develop a protocol to directly transmit the changes a user made to the report table configures, without them having to be manually be entered back into the system.

Additionally, more features than just the record table configuration could be integrated into the tool. In RWTHOnline each node contains a lot more information than just this table, and a more general version of this tool that also allows quick access to those informations might be useful for some users.

Bibliography

Sarah Grzemski and Bernd Decker. Challenges of the change of decentralized support structures in combination with digitization processes in the student life cycle. rwthonline the new campus management system of rwth aachen university. In *Proceedings of the 2018 ACM SIGUCCS Annual Conference*, SIGUCCS '18, page 91–97, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355827. doi: 10.1145/3235715.3235719. URL https://doi.org/10.1145/3235715.3235719.

Jakob Nielsen. *Usability engineering*. Morgan Kaufmann, 1994.

Jakob Nielsen. How to conduct a heuristic evaluation. *Nielsen Norman Group*, 1:1–8, 1995.

Jakob Nielsen and Rolf Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256, 1990.

Typeset March 31, 2020