Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Projektarbeit IT

HS 2018

ZHAWo

—

Platform Independent Timetable App

Authors
Bachmann Dominik
Visser Julian

Supervisor
Meier Andreas

December 21, 2018

Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Erklärung betreffend das selbständige Verfassen einer Projektarbeit an der School of Engineering

Mit der Abgabe dieser Projektarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Projektarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:

Winterthur, 20.12.2018

Unterschriften:

Das Original dieses Formulars ist bei der ZHAW-Version aller abgegebenen Projektarbeiten zu Beginn der Dokumentation nach dem Titelblatt mit Original-Unterschriften und -Datum (keine Kopie) einzufügen.

Contents

1 Introduction **2**
1.1 Goals . . . 2
1.2 Primary Functions . . . 3

2 Development **4**
2.1 Agile Approach . . . 4
2.2 Continuous Integration & Deployment . . . 4
2.3 Product Backlog . . . 5

3 Implementation **7**
3.1 Architecture . . . 7
3.2 Backend . . . 7
3.2.1 Express server application . . . 8
3.2.2 Persistence layer . . . 8
3.2.3 REST API adapter . . . 8
3.2.4 RSS feed adapter . . . 8
3.3 Frontend . . . 9
3.3.1 React web application . . . 9
3.3.2 REST API adapter . . . 10
3.3.3 Service worker . . . 10
3.4 Test Coverage . . . 10
3.5 User Interface . . . 11

4 Discussion **12**

5 Appendix **13**

Abstract

With ZHAWo we want to provide students of the Zurich University of Applied Sciences (ZHAW) with a single application where they can complete the most important tasks in their daily studies. Students need to be able to quickly look up their timetable, what lunch menus are available at their campus mensa and where a certain building or room can be found. While these functions are provided by the university's official app, we provide a consistent user experience among different platforms using Progressive Web App technologies. Additionally, with ZHAWo we provide students with an interface to search for free rooms. This function was previously provided by an app that is no longer available and was extensively used to find rooms for projects or study groups to work in environment less busy than the usually crowded official work spaces.
We also integrated news and events from the Verein Studierende ZHAW (vszhaw) into ZHAWo to provide students with easier access to information about events and services.
Using the JavaScript frameworks React for the frontend and Express for the backend in combination with Progressive Web App enables us to apply an agile development approach with fast prototyping of new features while still providing an application with the functionalities and feel that users expect from a native app. With this approach, we also avoid having to maintain multiple code bases for different platforms and mixing different technologies for frontend and backend. As a consequence, extending and improving the application's functionality can be done at a high velocity.

1 Introduction

1.1 Goals

Students of the Zurich University of Applied Sciences (ZHAW) have to visit multiple websites or use different applications on a daily basis in order to get information about their schedule, the offered menus in their campus mensa and events organized by the Verein Studierende ZHAW (vszhaw).
For their schedule, a student can either visit the official site [1] or use the official application for either Android [2] or iOS [3]. The official site was designed for use on desktop browsers and is not optimised for responsiveness and display on phones. And while the official Android application is well maintained and offers a lot of additional features - such as direct access to public transport timetables and mensa menu plans - its iOS counterpart lacks many of these additional features and looks and feels rather unpolished in comparison. This difference in quality and features is a common occurrence because the development of native applications requires two separate code bases for Android and iOS. The most glaring issue with the iOS application is the lack of offline functionality. When a user's network cuts out, even the schedule information that was previously loaded can no longer be accessed after navigating away.
When students want to check the mensa menus of their campus, they have the option of visiting the SV groups site or to use the official Android or iOS application. These options suffer from the same issues as previously explained for the schedule.
With ZHAWo, our goal is to provide students with an improved application to have access to their schedule and mensa menus in one single cross-platform application. Additionally, with ZHAWo students can look for unoccupied rooms. This functionality was - until about two years ago - provided by an Android application that was no longer maintained and eventually disappeared from the Google Play Store. While the study rooms at, for example, the Technikum campus offer a space for both quiet work as well as group projects, in our experience as students it was very convenient to have a service to quickly look up free rooms without having to walk from room to room. Another feature ZHAWo provides is the integration of news and events of the vszhaw directly into the application. We aim to reduce the effort that is needed to stay up to date with the vszhaw and hope that both students and the vszhaw can profit from this.
By using Progressive Web Application (PWA) technologies [4] in combination with JavaScript frameworks for both front- and backend, we achieve a consistent user experience on desktop, Android and iOS devices. We eliminate the issue of having to maintain separate code bases for different platforms while still being able to provide a native feel and functionality. PWA features such as offline caching of HTTP requests allow us to overcome previously mentioned issues with reliability on spotty networks.
An additional advantage we gain by using PWA technologies and the same programming language across the full stack is fast prototyping in an agile development process. Development of additional features and functionality can be achieved at a much faster rate with a single code base across all platforms.

1.2 Primary Functions

While our goal is to develop a feature rich application tailored to students of the ZHAW, we established the following four primary functions ZHAWo should provide in its prototype. Using an agile development approach, the scope of this project was to implement these primary functions in a prototype and adjust them to student feedback.

- **Timetable**: A user can access their schedule and look up schedules of lecturers, classes, courses and rooms.
- **Menu plans**: A user can access menu plans of the different campus mensas across the ZHAW.
- **Room search**: A user can look for and find free rooms for a specific time-frame and location.
- **Student events**: A user can access vszhaw news and events to bring more attention to student parties and events.

After having established the core functionalities and a prototypical version of ZHAWo in the scope of this project, the features will be iteratively expanded and improved to end up with a production-ready application. For a full product backlog please refer to section 2.3.

2 Development

2.1 Agile Approach

For development of the ZHAWo prototype we used an iterative agile approach. The code base was managed on a single GitHub repository [5] for both front- and backend. User stories were tracked using GitHub issues and the sprints were organised using GitHub project boards. Over the course of the project, we structured the development into six sprints of 2 weeks. We regularly considered feedback of other students and our own review of our practices and used technologies in our planning. The flexibility of using only JavaScript and PWA technologies enabled us to rapidly prototype new ideas without the hassle of having to maintain two different code bases. User stories were implemented and tested in feature branches and merged into the master branch as part of the sprint reviews. This practice ensured that we had - for the most part - stable new iterations of the application to deploy for user feedback.

2.2 Continuous Integration & Deployment

For continuous integration we used Travis CI [6] in combination with Codecov [7] to track test coverage. With good integration of both these tools into GitHub - for example reporting of build status and test coverage changes in GitHub comments on each pull request - we achieved good code and test quality across feature implementations and sprints. While the goal is to eventually host our application on a ZHAW server, in the scope of this project the prototype was deployed to Heroku [8] and shared amongst students for quick feedback on new features.

2.3 Product Backlog

The following planned user stories have been implemented in the scope of this project. We put the focus on implementing the schedule context of ZHAWo in detail and provide prototypical functionality for the other primary functionalities. For implementation details please refer to section 3.

- **US01**: As a user I want to save my credentials/username
- **US02**: As a user I want the app to work even when I don't have network connection
- **US03**: As a user I want to switch between contexts (schedule, menus, room search, vszhaw)
- **US10**: As a user I want to view my timetable/schedule for a day
- **US11**: As a user I want to view my timetable for a week
- **US12**: As a user I want to navigate to the current day
- **US13**: As a user I want to navigate between days when using the day view (timetable)
- **US14**: As a user I want to navigate between weeks (timetable)
- **US15**: As a user I want to navigate to a specific date in a month view (timetable)
- **US16**: As a user I want to view a specific room's timetable
- **US17**: As a user I want to view a specific class's timetable
- **US18**: As a user I want to view a specific course's timetable
- **US19**: As a user I want to view a specific person's timetable
- **US20**: As a user I want to have a detailed view of my events
- **US30**: As a user I want to find currently unoccupied rooms
- **US56**: As a user I want to see prices for all menus
- **US58**: As a user I want to view a specific mensa menu plan
- **US70**: As a user I want to view vszhaw blog posts/event announcements

The following user stories are planned to be implemented in the finished application. This list will of course be extended and adjusted based on user feedback and our own review.

- **US31**: As a user I want an overview of my campus with highlighted buildings where there are unoccupied rooms
- **US32**: As a user I want a floor plan of each floor per building with highlighted unoccupied rooms
- **US33**: As a user I want to navigate between buildings through the overview of my campus
- **US34**: As a user I want to navigate between floor plans of a building
- **US35**: As a user I want to see until when a room is unoccupied
- **US36**: As a user I want to filter my search to only show rooms that are unoccupied for at least x hours/minutes
- **US50**: As a user I want to view the mensa menu for my campus for a day
- **US51**: As a user I want to view the mensa menu for my campus for a week
- **US52**: As a user I want to navigate to the mensa menu of the current day
- **US53**: As a user I want to navigate between days when using the mensa menu day view
- **US54**: As a user I want to navigate between weeks when using the mensa menu week view
- **US55**: As a user I want to navigate to a specific date in a month view (mensa menu)
- **US57**: As a user I want to navigate between menus of different days

3 Implementation

3.1 Architecture

ZHAWo consists of a frontend React [9] web application and a backend server based on the Node.js [10] package Express [11]. The frontend communicates with the backend through a REST API. The backend server itself - apart from providing the API for the frontend - communicates with the CampusInfo REST API provided by the ZHAW and the RSS feed of the vszhaw homepage. The CampusInfo API provides timetable and menu plan information.

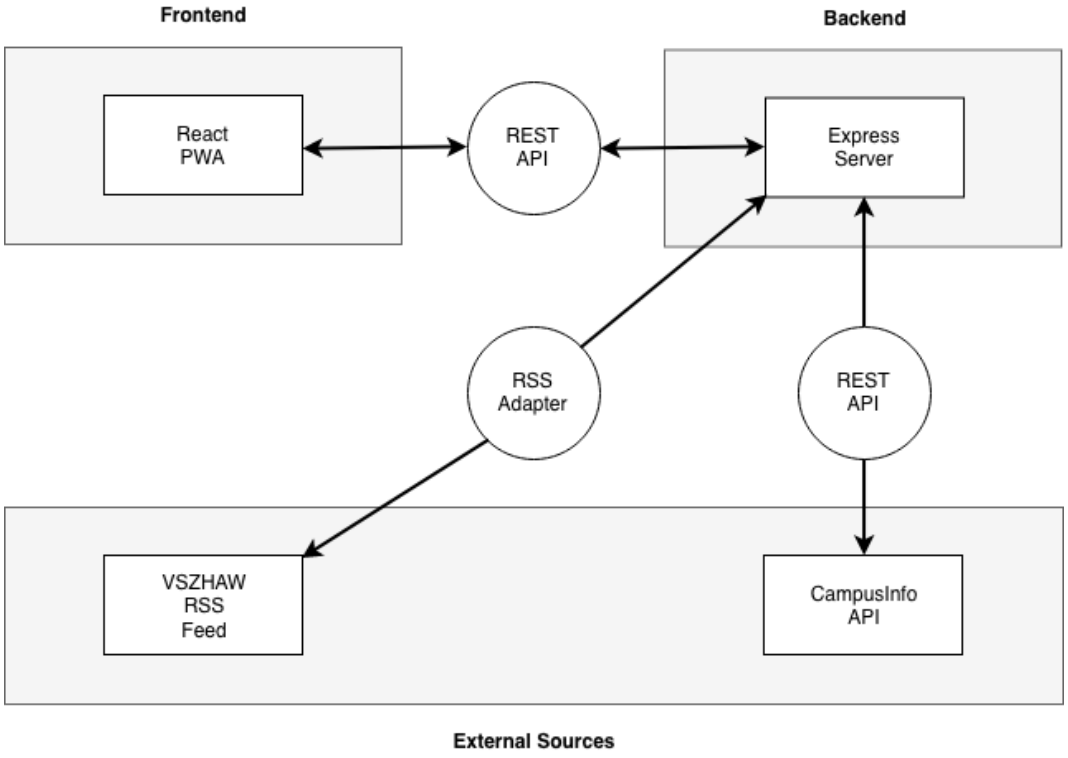

Figure 1: Application Architecture Diagram

3.2 Backend

The backend consists of the following modules:

- **Express server application**: Handles HTTP requests from frontend and redirection to persistance layer or third-party adapters.
- **Persistence layer**: Handles caching of more resource intensive requests such as room search. Implemented with basic file system using JSON file in the prototype. Can be extended by database if functionality requires.
- **REST API adapter**: Handles HTTP fetch requests to the CampusInfo API.
- **RSS feed adapter**: Handles fetching of RSS feed from vszhaw.

3.2.1 Express server application

For the backend server application, we have chosen to use the framework Express [11] based on the Node.js [10] technology. The server provides access to the React web application and also offers a REST API. The primary responsibility of the server is to redirect user request from the frontend to either persistence layer or the different API adapters and then serve the data. We have chosen to implement the backend with a JavaScript based framework so there is no technology difference between front- and backend. In the context of agile development in a small team, this allows for a fast implementation of features. A new feature often requires changes to both front- and backend, and by removing the need to switch context between different programming languages, we were able to maintain a good velocity throughout our sprints.

3.2.2 Persistence layer

In the scope of developing this prototype, there was no need to implement a full database for data persistence. The data for more resource intensive requests such as the search for free rooms is currently stored in JSON format directly in the servers file system. Less intensive requests such as timetables are delivered from CampusInfo in a format that only needs minimal modifications and are directly sent to the frontend without persistence.
However, should the need for a database later arise through new functionality, integration into the Express server application is already prepared.

3.2.3 REST API adapter

Most of the application data for ZHAWo is provided by the CampusInfo API. Since this is a third-party API provided by the ZHAW, we decided to implement an additional layer between our backend application core and the API. This insures that we are flexible to changes to CampusInfo.

3.2.4 RSS feed adapter

Similarly to the CampusInfo API adapter, the RSS feed adapter provides an additional layer between our application and the RSS feed of the vszhaw. This ensures that should the data received by this third-party feed change, the changes needed to our application will be isolated to the adapter.

3.3 Frontend

The frontend is made up of three main parts:

- **React web application**: Handles presentation of data and user interaction.
- **REST API adapter**: Handles HTTP fetch requests to the backend.
- **Service worker**: Handles PWA functionality such as caching of data for offline use and installation to desktop or phone homescreen.

3.3.1 React web application

For the presentation of the web application to the user, we used the React framework [9]. React was originaly developed by Facebook and is one of the most popular UI libraries in web development. It is based on reusable components built with JSX, a syntax extension to JavaScript. We decided to use React because of its component based modularity, which works well with an agile development approach where multiple features need to be implemented simultaniously with little interference.
To handle the application data we chose to use the Flux design pattern [12]. Using the Flux pattern ensures a unidirectional data flow from view components through actions into a single dispatcher into data stores, where the application data such as timetables and menu plans is handled. In Flux, the dispatcher is a singleton that directs the flow of data to ensure that updates do not cascade, which would lead to unpredictable behaviour. When a user interacts with a React view, the view sends an action through the dispatcher, which notifies the stores that hold the application's data. When the stores change state, the view gets notified and changes accordingly [12].

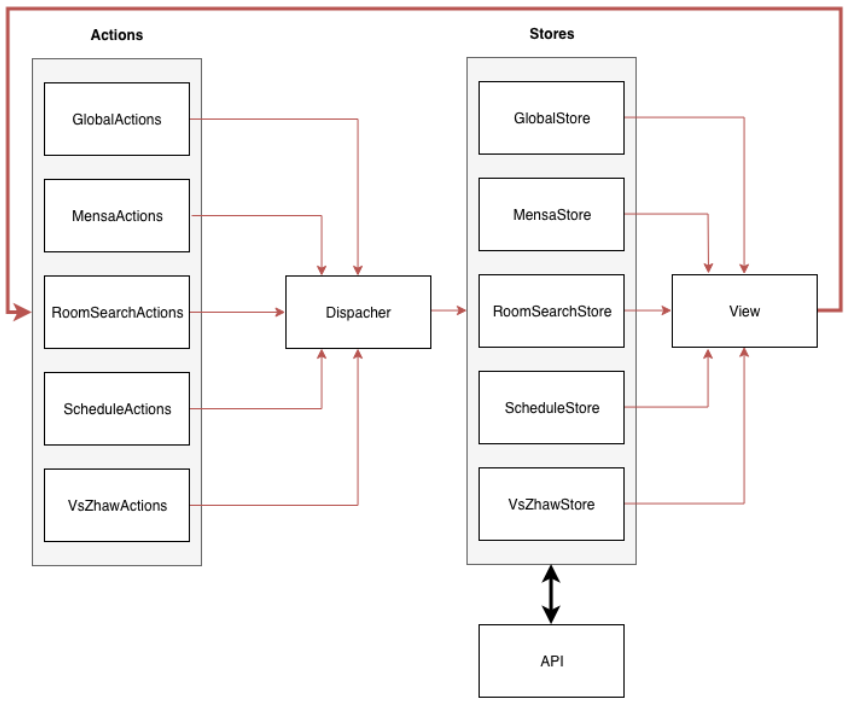

Figure 2: Flux Pattern Diagram

3.3.2 REST API adapter

The data such as timetables, menu plans or lists of free rooms are provided by the backend REST API. To ensure modularity between front- and backend, we implemented an adapter module that handles all data requests to the backend. This practice allows easier adaptations to changes in API, as only the adapter would have to be changed, while the other parts such as the React application and service worker do not need to change.
The modularity this design choice provides - similarly to the modularity of React - goes well with the agile principle of fast prototyping of new ideas and features.

3.3.3 Service worker

The data for the frontend is provided by the backend REST API. All data received from HTTP fetch requests is cached using a service worker [13]. By caching the application data, we ensure that the user can still access all the information that was already loaded once. It also improves the loading speed of ZHAWo, since resources that have been cached are first served from cache, before the data is requested by the backend. After the fetch request has completed the cache is updated. With regard to ,for example, a students schedule information, this practice makes a lot of sense since the actual data does not change often during the course of a semester.
In addition to offline caching, the service worker also allows users to install ZHAWo as a Progressive Web App (PWA) [14] to either their desktop or their phones homescreen.
These two features enables us to offer the experience of a native application while we avoid having to maintain separate code bases for each platform.
Google provides the automated tool Lighthouse [15] to audit and rate PWAs. Lighthouse various PWA aspects, including if the application still responds without network connection. ZHAWo achieved a score of 92 out of 100. A detailed report can be found on the Github Repository [5].

3.4 Test Coverage

For testing we used the framework Jest [16] for both front- and backend. To test functionality and output of React components, we used the GUI testing framework [17] in addition to Jest. For the express server application, we achieved a test coverage of 62%. For the frontend React application 53% of the code was covered, resulting in an overall project test coverage of about 55% [18]. The coverage is lower than we had planned and is something we aim to improve when going from application prototype to production ready application. This can be in part attributed to the prototypical implementation of the primary functions menu plan, room search and student events. Another aspect is that the current JavaScript framework environment is very dynamic and fast paced, which on one hand offers many advantages, on the other hand, best testing practices have not been established. For example, it was challenging to isolate and test the different components of the Flux architecture in the frontend. Therefore, we decided to put a lot of focus on establishing better testing practices in the next stage of this project.

3.5 User Interface

The user interface was designed to provide students with the familiar look and feel of a native mobile application. We chose a minimalistic design approach. The focus was on displaying relevant information without any distracting noise or clutter.
The mockups were made so that both of us could work independently and no design decisions had to be made on the go (see Figure 3). It was important to us to have a consistent design throughout the application.

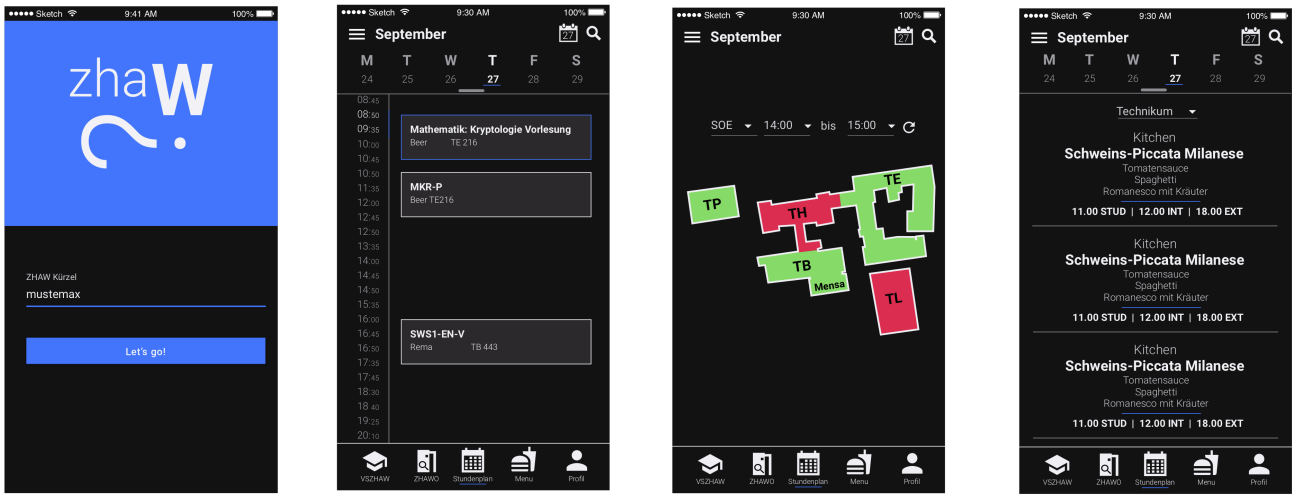

Figure 3: Mobile Mockups

The initial mockups were made with a dark theme. Our first user feedback on the design was very positive, but we learned that most users preferred a light theme over our dark theme. We decided to changed the default theme, whilst still leaving the option for users to change to the dark theme if preferred.

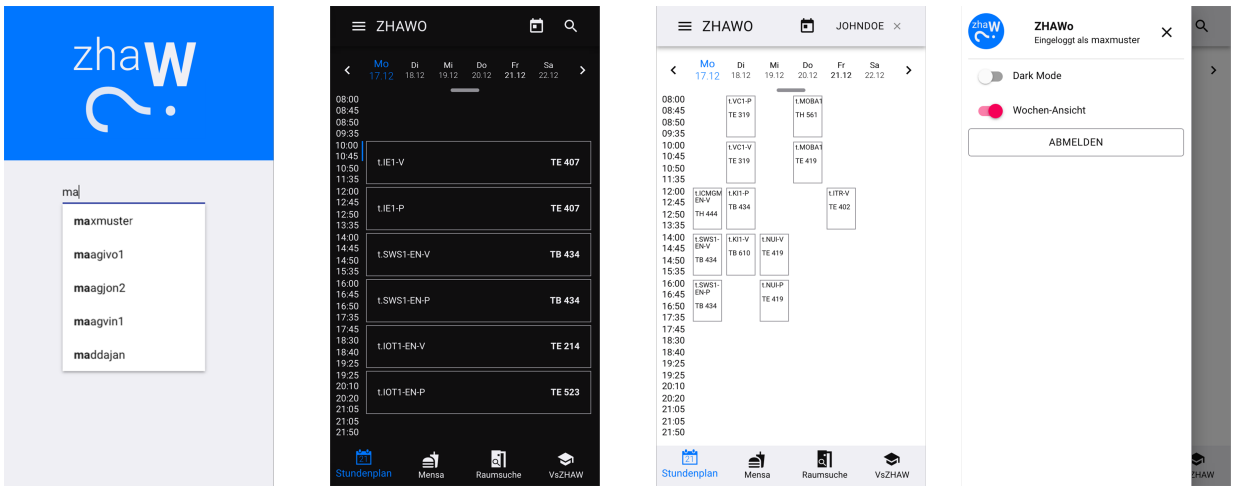

Figure 4: Mobile Screenshots

4 Discussion

We were able to implement a prototype of ZHAWo as a Progressive Web App which covers all primary functions that were planned. Students can access their schedule, mensa plans and the news and event feed from vszhaw. Additionally, they can get a list of free rooms that can then be used as a distraction free work space. We were able to share the prototype with a small group of students and user feedback for our application was positive.
Using a full JavaScript technology stack for front- and backend and Progressive Web Apps has proven to work well with an agile development approach. While developing native applications, having to maintain different code bases can be awkward in combination with agile principles. The strength of the agile development approach that we have chosen is that we can prototype new features rapidly and can receive user feedback from both Android and iOS users immediately. In addition to that, not having to stay up to date with best practices, libraries and frameworks for different languages such as Java/Kotlin and Swift/Objective C for Android and iOS development respectively allowed us to implement new features at a faster pace.
While there are obvious advantages to building ZHAWo as a cross platform PWA, we have also identified some weaknesses. Installing an application not through the App Store or the Google Play Store is unexpected for most users. There are also some inconsistencies in support offered by different platforms. We attribute most of our issues to PWA technologies being relatively new and expect both support and users familiarity with the concept of PWAs to increase in the near future.
In a second part of this project, we aim to improve the user experience further and extend ZHAWo with more features.

5 Appendix

List of Figures

1 Application Architecture Diagram 7
2 Flux Pattern Diagram 9
3 Mobile Mockups 11
4 Mobile Screenshots 11

References

[1] ZHAW. (2018). ZHAW Stundenplan, [Online]. Available: http://stundenplan.zhaw.ch (visited on 12/17/2018).

[2] ——, (2018). ZHAW Engineering CampusInfo Android App, [Online]. Available: https://play.google.com/store/apps/details?id=ch.zhaw.init.android.campusinfo (visited on 12/17/2018).

[3] ——, (2018). ZHAW Engineering CampusInfo iOS App, [Online]. Available: https://itunes.apple.com/ch/app/zhaw-engineering-campusinfo/id715684381 (visited on 12/17/2018).

[4] Google. (2018). Progressive Web Apps, [Online]. Available: https://developers.google.com/web/progressive-web-apps/ (visited on 12/12/2018).

[5] Dominik Bachmann, Julian Visser. (2018). ZHAWo Github repository, [Online]. Available: https://github.com/zhaw-timetable/zhawo (visited on 12/17/2018).

[6] travis ci.org. (2018). Travis CI, [Online]. Available: https://travis-ci.org/ (visited on 12/17/2018).

[7] codecov.io. (2018). Codecov, [Online]. Available: https://codecov.io/ (visited on 12/17/2018).

[8] heroku.com. (2018). Heroku, [Online]. Available: https://heroku.com/ (visited on 12/17/2018).

[9] reactjs.org. (2018). ReactJS, [Online]. Available: https://reactjs.org (visited on 12/12/2018).

[10] nodejs.org. (2018). NodeJS, [Online]. Available: https://nodejs.org/en/about/ (visited on 12/12/2018).

[11] expressjs.com. (2018). ExpressJS, [Online]. Available: https://expressjs.com/ (visited on 12/12/2018).

[12] Facebook. (2014). Flux, [Online]. Available: http://facebook.github.io/flux/ (visited on 12/12/2018).

[13] M. Gaunt. (2018). Service Workers: an Introduction, [Online]. Available: https://developers.google.com/web/fundamentals/primers/service-workers/ (visited on 12/13/2018).

[14] P. LePage. (2018). Your First Progressive Web App, [Online]. Available: https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/ (visited on 12/13/2018).

[15] Google. (2018). Lighthouse, [Online]. Available: https://developers.google.com/web/tools/lighthouse/ (visited on 12/20/2018).

[16] Facebook. (2018). Jest, [Online]. Available: https://jestjs.io/ (visited on 12/13/2018).

[17] Airbnb. (2018). Enzyme, [Online]. Available: https://airbnb.io/enzyme/ (visited on 12/13/2018).

[18] Dominik Bachmann, Julian Visser. (2018). ZHAWo Codecov project, [Online]. Available: https://codecov.io/gh/zhaw-timetable/zhawo/ (visited on 12/17/2018).