



DDIARY
THE DIGITAL DIARY

Technologische Fachoberschule
„Max Valier“ Bozen

Alex Micheli & Daniel Bertagnolli

2015 - 2016

Tutor: Prof. Dr. Michael Wild

Contents

1	Introduction	6
1.1	The problem	6
1.2	The solution	6
1.3	The motivation	6
1.4	The goal	6
2	DDiary	7
2.1	Events	8
2.2	Calendar interface	9
2.2.1	Month	10
2.2.2	Week	10
2.2.3	Day	11
2.3	Posts	12
2.4	Commenting System	13
2.5	Changelog	13
2.6	Administrator - Normal user	13
2.7	Statistics	14
3	Technologies	15
3.1	What is a single-page application?	15
3.2	Why single-page application?	15
3.2.1	MVC	15
3.3	jQuery	16
3.3.1	Why not jQuery?	16
3.4	Angular 2	17
3.5	Why Angular 2 and not AngularJS?	17
3.6	REST?	17
3.7	Java vs PHP vs Ruby	18
3.8	Was Java a good decision?	18
3.9	ES5 versus ES6	18
3.10	Typescript	18
3.11	Deployment for development	19
3.12	Deployment for production	19
3.13	Tomcat	19
3.14	Minification and bundling - WebPack	20
3.15	How our application works	20
3.15.1	PostComponent	22

4	Security	24
4.1	Authentication	24
4.2	Forgot your password?	25
4.3	SQL Injection	26
4.4	Cross Site Scripting - XSS	26
4.5	Form validation	27
5	Database	28
5.1	Sql2o	28
5.2	MariaDB	28
5.3	Event list	28
6	Design	32
6.1	UX - User Experience	32
6.2	One design language	32
6.3	Gravatar	33
7	Development	35
7.1	Debugging	35
7.2	Browser compatibility	35
7.3	Versioning / Git / Bitbucket	36
7.4	Dropbox	37
7.5	Tasker	37
7.6	Eclipse	37
7.7	Atom	37
7.8	NPM - Node Packet Manager	37
8	Mobile App Development	38
8.1	Hybrid apps	38
8.2	Why Ionic 2?	38
	8.2.1 Advantages	38
	8.2.2 Disadvantages	39
8.3	Deployment	39
9	Appendix	41
9.1	Future prospects	41

List of Figures

2.1	Use cases	7
2.2	List of future events	8
2.3	Dialogue box to create a new event	8
2.4	Calendar viewed from the browser	9
2.5	Week view of the calendar	10
2.6	Week view of the calendar	11
2.7	Day view of the calendar	12
2.8	A post with a comment	12
2.9	Comments about a future event	13
2.10	Changelog	14
2.11	Administration section	14
3.1	https://angular.io/resources/images/logos/standard/shield-large.png http://code-maven.com/img/angularjs.png	17
3.2	Tomcat Logo	19
3.3	Class diagram of posts	21
3.4	Sourcecode of the PostlisteComponent	22
4.1	JWT Token consisting of 3 parts	24
4.2	Header of JWT	24
4.3	Payload of JWT	24
4.4	Signature of JWT	24
4.5	Forgot password page	26
4.6	Example of form validation on the sign-up page	27
5.1	Method for requesting the database to retrieve an event list	29
5.2	Structure of the database	30
5.3	Database design	31
6.1	Information bubble	32
6.2	Private Badge	33
6.3	Date format	33
6.4	Number of events in that lesson	33
6.5	https://upload.wikimedia.org/wikipedia/commons/b/b2/LogoGravatar.png .	34
6.6	Random profile pictures	34
6.7	Random profile pictures	34
7.1	Firefox clock picker	36

7.2	Chrome clock picker	36
8.1	http://ionicframework.com/img/ionic-logo-blog.png	38
8.2	https://upload.wikimedia.org/wikipedia/commons/d/d7/Android_robot.svg ; http : //logok.org/wp-content/uploads/2014/04/Apple-logo-grey-880x625.png ; https : //upload.wikimedia.org/wikipedia/commons/thumb/5/5f/Windows_logo-2012.svg/2000px- Windows_logo-2012.svg.png	40

1 Introduction

1.1 The problem

We had the particular problem that we always forgot to do our homework because we were too lazy to take down the assignments or we just didn't take notice what the teachers were saying. Moreover, absent students often do not get informed well enough by their classmates about assignments and sometimes appointment change or unclear assignments also lead to confusion.

1.2 The solution

Our solution to all these annoying occurrences was to create a platform where we cannot only register our own assignments, but further share them with the whole class. The advantages compared to the classic paper diary are obvious. You can access all your assignments from anywhere with your smartphone. Not every student has to write down the assignments, but instead only one student does and can then share it with his classmates. A built-in comment system would also make it easier to discuss assignments with classmates.

1.3 The motivation

There are some applications developed for taking down assignments but none of them allows students to share their assignments in a comfortable way nor do they perfectly fit students needs. So the only solution was to develop our own system, which would also allow us to integrate some special features such as an exam-list-tool.

1.4 The goal

We wanted to create a platform that simplifies and improves the whole process of writing down assignments by providing our users a web and a mobile application with very good user experience. We have called this platform DDiary (Digital Diary).

2 DDiary

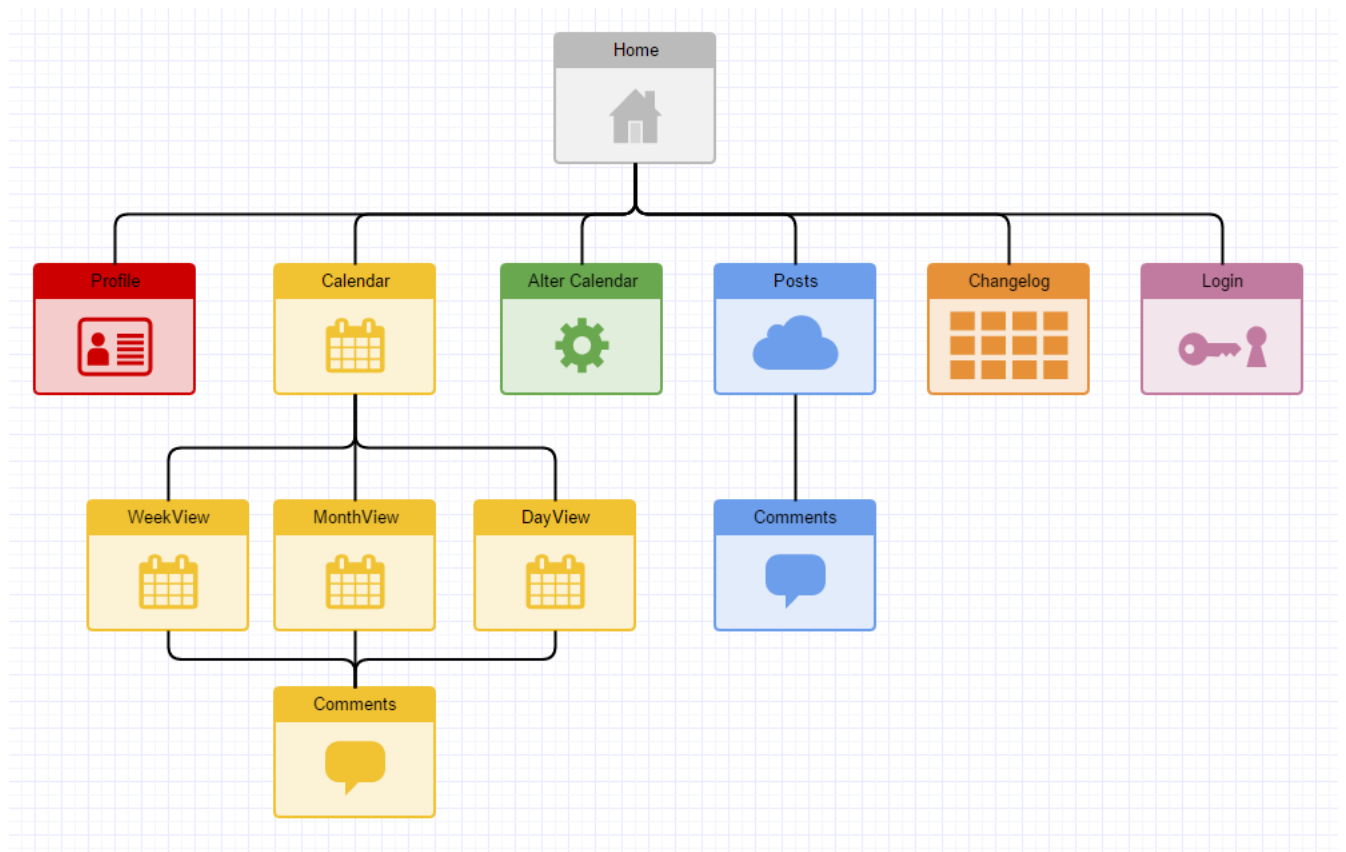


Figure 2.1: Use cases

2.1 Events

Events are one of the most important components in DDiary. A user can create events which are bound to a specific lesson, for example, an assignment, or events that have nothing to do with the timetable, a meeting in the afternoon with an old friend for example. All classmates can see the events of their colleagues unless if they mark them as "private". In this case only the creator can see them. An example for a private event would be an examination of a single person. Events have a title, description, date, start time, end time and a flag that says if it is private or not. Events are not directly bound to the timetable, but if an event coincides with a lesson it seems as if they are bound to each other. The only connection they have is the time. When querying the events and the lessons from the database, an algorithm checks if the events start and end in a lesson, if this is the case they will be put together in a common object. On the application however they will be displayed in different ways. Every class member can alter or delete public events.

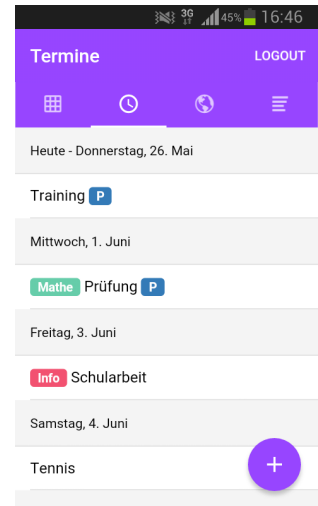


Figure 2.2: List of future events

Figure 2.3: Dialogue box to create a new event

2.2 Calendar interface

The calendar is the heart of DDiary. Nearly everything revolves around it. Accordingly we spent most of the time developing and optimizing it. We wanted to provide the best user experience possible and at the same time keep it simple but powerful.

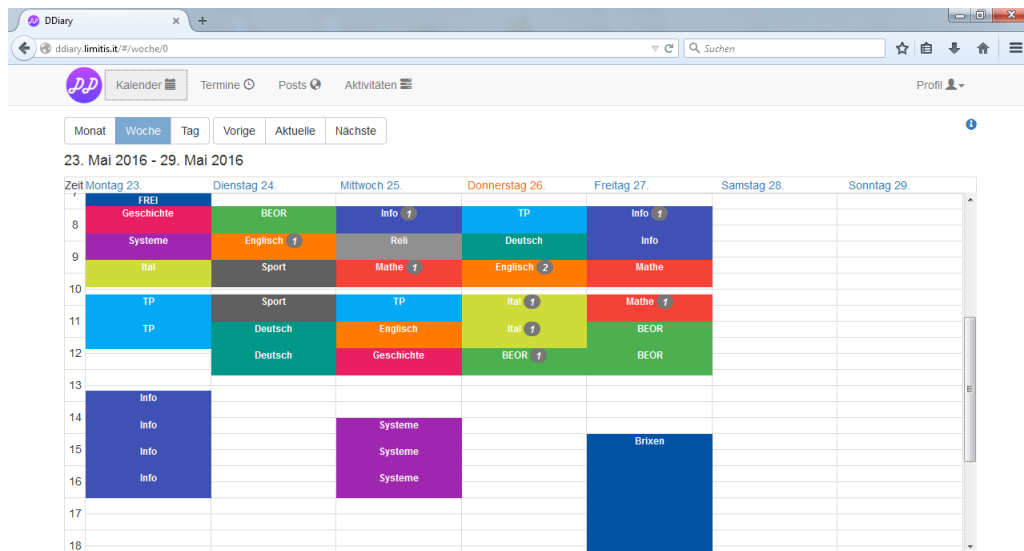


Figure 2.4: Calendar viewed from the browser

The whole calendar is divided into 3 parts: the month view, the week view and the day view.

2.2.1 Month

The month view displays one month. Days that have an event are marked with a number that represents the number of events that day. When the user clicks on a day he/she is redirected to the day view of that day. With two buttons the user can navigate through the months to spot the day of desire fast. The month view consists of only one table that is generated dynamically by data. The data is gathered by an Ajax call to the back-end server and consists of two number arrays where the first array contains every day of the month and the second array contains the number of events that occur on that day.

Juni 2016

Montag	Dienstag	Mittwoch	Donnerstag	Freitag	Samstag	Sonntag
		1	2 1	3 1	4	5
6	7 1	8	9	10	11 1	12
13	14	15	16 1	17	18	19
20	21	22 1	23 1	24	25	26
27 1	28 1	29	30			

Figure 2.5: Week view of the calendar

2.2.2 Week

The week view is the most complex part of the calendar. It displays the timetable, the events of the selected week and the private events as well.

The week is structured with multiple nested tables. The main-table dictates its structure. At the top a table contains information about the days of the selected week. By clicking on them the user gets redirected to the day view. For a better user-experience the current day is highlighted. On the left hand side another table represents all 24 hours of a day and the rest is the timetable itself. The background of the table is a grid which divides the day in "half-hour cells". By clicking on one of them a dialog opens and the user can create a new event, which is not bound to any lesson, by filling in and submitting the form. All lessons and events are displayed at the top of the grid.

The lessons and events are positioned exactly at the time they were previously set. Even very slight changes, like 5 minutes, are visible. We achieved this by calculating the pixels of every event and lesson with the following two formulas.

Formula 1:

calculates the distance from the top of the table to the start of the lesson/event.

`Math.floor(begin / SIZE_CONSTANT)`

Formula 2:



Figure 2.6: Week view of the calendar

calculates the height of the lesson/event.

`Math.ceil((end - begin) / SIZE_CONSTANT)`

Example:

An event starts at 07:00 am and ends at 10:00 am. The size constant is set to 1.5. 07:00 am equals 420 minutes and 10:00 am equals 600.

After using the first formula the top-pixels would equal 280 (`Math.floor(420 / 1.5) = 280`). `Math.floor()` rounds the result to the next lower integer and is needed avoiding strange pixel-spaces between the components since fractional digits cannot be displayed as pixels.

The second formula would return 120 (`Math.ceil((end - begin) / SIZE_CONSTANT) = 120`). `Math.ceil()` rounds the result up to the next integer, this is needed for the same reason as `Math.floor()` is needed for the first formula.

2.2.3 Day

The day view is basically identical with the week view except that only one day is displayed. Through the day view the user can get a better overview of a single day and can focus only on the desired day without having the whole screen covered with overhead. The functions of the day view are the same as in the week view, thus a user can create events for a certain lesson or time interval.

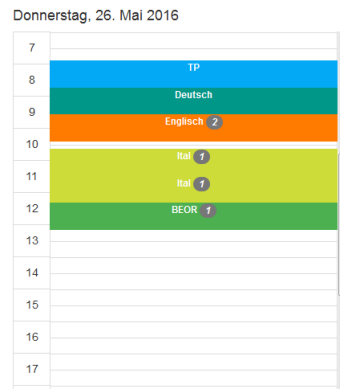


Figure 2.7: Day view of the calendar

2.3 Posts

Sometimes a student wants to ask or share messages and information with his classmates. Therefore we created the post functionality. Students can create posts that are shared with their classmates and the classmates can comment on them. A post consists of text that cannot be altered after it has been submitted, however it can be deleted. All posts of a class can be viewed in the post section of the website in chronological order.

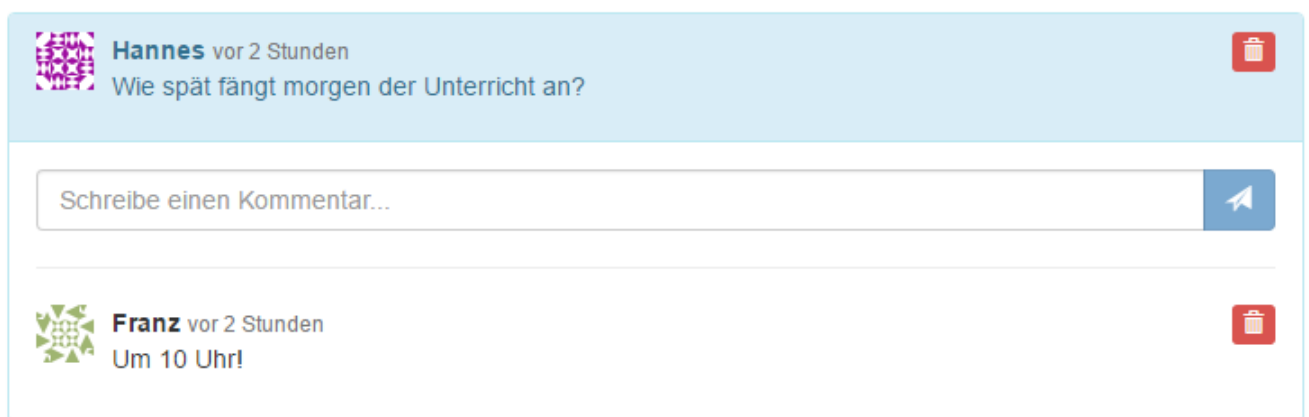


Figure 2.8: A post with a comment

2.4 Commenting System

The users need to have the possibility to comment on posts, events, exchange ideas or simply ask for information.

For example, if there is a homework assignment for next week and someone wants to know where he can find the assignment, he could just put a comment under it and wait for an answer from his classmates. The same applies for posts, where someone could have asked a question and needs an answer. The commenting system is an Angular component that represents a section which displays comments that are already made and a possibility to create new ones. As input it gets an identification number and the type that can either be "post" or "event" because it is possible to comment posts or events. The component then makes a request to the back-end server for the comments and the back-end server expects two parameters: the number of comments it wants to fetch and the offset. The server then returns that number of requested comments if there are any and orders them by date, so that the latest comments come first. A comment consist of a comment text, the profile picture of the creator, the name of the creator and the date when the comment was created. A comment can not be altered but it can be deleted by the creator and all administrators of that class.

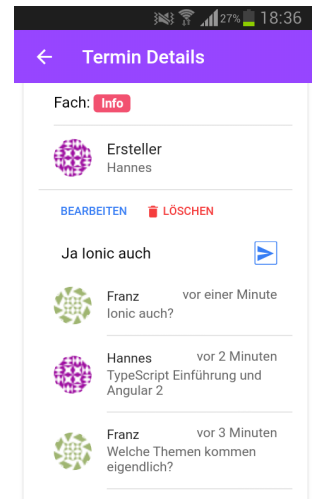


Figure 2.9: Comments about a future event

2.5 Changelog

We created the changelog function because of the necessity that students want to know what their classmates do in DDiary. We got inspired by Facebook's notification system where a user is notified when something related to them happens on Facebook, but contrary to Facebook all students of a class have the same interests related to DDiary. Therefore we decided that it would make sense to show all classmates the same notifications. A notification is created when a comment, event or post is created or altered except for private events where only the creator of the event is notified.

A changelog consists of the profile picture of the user who did the action, a recognizable icon, a short message describing the action and whenever possible a link to that action.

2.6 Administrator - Normal user

For managing the classes we introduced a user system with both normal users and administrators. A normal user can make posts, create and alter events and comment every entry model. An administrator has the same rights as a normal user but in addition to that he can delete all posts, comments and events, create new subjects, alter or delete them, change the timetable and manage the members of the class. He can also give administration privileges

Aktivitäten

Filtern nach:



Figure 2.10: Changelog

to his classmates or take them. The administrator is even able to kick classmates out of the class or to change the password of the class. A user becomes administrator when he creates a class or when he is nominated by another administrator.

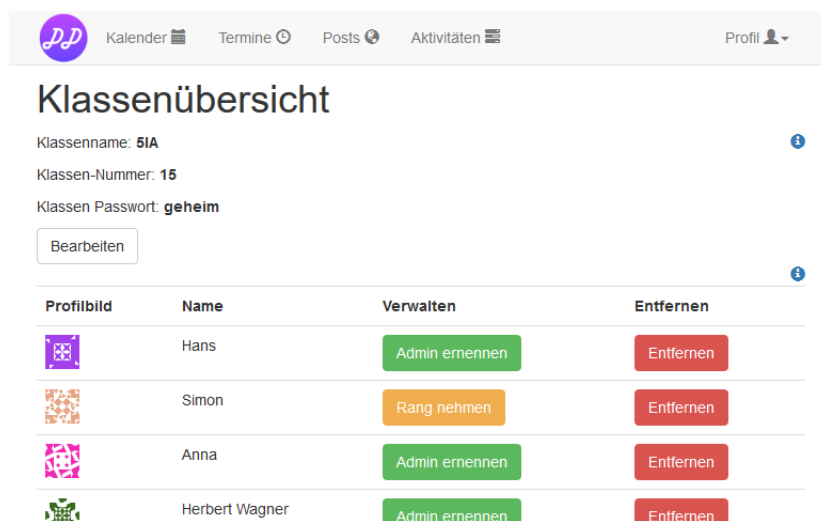


Figure 2.11: Administration section

2.7 Statistics

We implemented a simple statistics page that is only accessible by the developers. For this purpose we added a time stamp field to almost every table in the database so that we can track the activity on our application over time. Changing the database led to some problems since some tables already had a time stamp field and because MySQL only updates/fills the first time stamp field in a table. We had to manually adjust this by changing the default behaviour of the time stamps.

3 Technologies

3.1 What is a single-page application?

A single-page application consists of only one webpage, the so called single-page. This single-page can then load other resources from the server by making Ajax requests. One big advantage of single-page applications is that they feel more native and the pages load faster after the first loading.

3.2 Why single-page application?

Nowadays most users access the Internet via their smart phones, because of that we try to provide our users the best experience possible, by providing them a "mobile-first application". For our frequent users we also wanted to create a mobile app to ensure them an even better experience.

For this purpose we invested a lot of time in researching the best technologies to achieve this. Starting from a PHP, JSP background we realized that these frameworks did not work well to achieve our objectives. Because of our project's complexity we considered using JSF that builds on MVC model.

3.2.1 MVC

MVC stands for Model View Control and divides the application layers into three parts. The Model manages the data, logic and rules of the application. The View represents presentation of the data. Finally, the Controller interconnects the Model with the View and the application flow.

After prototyping our application with JSF we were afraid that the mobile user experience would not be seamless enough to use it on a daily basis. For this reason we decided not to use JSF.

After we realized that if we also wanted to provide a mobile app to our users, we would have to write the application twice or even more times considering different platforms, we were also confronted with the issue of data synchronization between the mobile and the web versions. Having experience with data synchronization (Tasker) already, we knew that it would require to much effort to implement this. After some research we bumped into web services. With this tool we would have to write the system's logic only once and both web-app and mobile app could fetch data from it.

3.3 jQuery

jQuery is a very popular JavaScript library. It allows the developer to manipulate the DOM by just calling some functions. Tons of plug-ins like datepicker, timepicker, accordions, etc. are built with jQuery. It has also the ability to make Ajax calls which are essential for single page applications. So this sounds like a lot of advantages, but we decided not to use jQuery for building our application.

3.3.1 Why not jQuery?

The main reason we are not using jQuery is that writing an application of this scale is almost impossible just with jQuery without the help of a framework.

Instead we have used a framework called Angular 2. JQuery should not be used in combination with Angular as they work on different levels. JQuery is mainly used for doing DOM manipulation, on the other hand Angular is trying to get away from DOM manipulation. So it is bad practice to use jQuery in addition to Angular, but to be honest we are using jQuery for some small tasks because Bootstrap relies on it and because we use some plug-ins like timepickers written in jQuery as at the moment of developing there were almost no plug-ins for Angular. As ng-bootstrap (Bootstrap written in Angular) is in development we want to switch to that just when it is available in a stable release.

Other technology

There are many single-page application frameworks such as the React framework from Facebook or Backbone, Ember, Knockout, Meteor, etc. this was a bit overwhelming so we stuck with the most popular, Angular 2, because hopefully it has the best documentation and support.

3.4 Angular 2

Angular 2 is the second version of the AngularJS web framework for building mobile and desktop web applications. It is open source and mainly maintained by Google and the community.



Figure 3.1: AngularJS vs Angular 2

3.5 Why Angular 2 and not AngularJS?

Before we started with the development we wrote a little test website with AngularJS. The first thing we noticed was that the way of programming was really different from how we had known it until then. The code seemed unreadable to us and we thought: if the code is unreadable now, what will it be like in a few months when our code grows up to thousands of lines?

The test website written in Angular 2 looked a lot more organized and structured. We could code using Typescript which looks a lot more like the programming languages we knew.

Obviously this was not the only reason we chose Angular 2 over AngularJS.

As Angular 2 was only still Alpha, we knew that development would not be easy at all, since every new version would bring breaking changes with it and this would mean changing the code and maybe spending hours in searching for bugs which were actually not introduced by us but by Angular and we would have to wait weeks for bug fixes.

Other motivations for using Angular 2 were that it is mobile oriented and AngularJS is not. The second version also promised a much better performance than AngularJS by the use of newest technologies.

3.6 REST?

Representational state transfer, also known as REST is an architecture style for designing networked applications. It is used for web services, in our case to let the front-end Angular application and the server communicate with each other. Applications using REST use HTTP-requests to post, delete and query data, which is exchanged using JSON.

3.7 Java vs PHP vs Ruby

It is possible to write REST API's in many languages. The most appropriate languages for web development seemed Java to us, which is very well supported in the enterprise development, PHP is also well known for its use in web development and Ruby with its concise and beautiful syntax and the very popular Ruby on Rails framework. Other languages could be mentioned, C-Sharp, for example, but we decided to stick with these three languages. One big plus point for Java was that we already knew it very well, we also knew PHP and its pitfalls. Ruby was new to us and we were highly interested in its framework Ruby on Rails, we gave it some time to convince us but in the end we favoured Java.

3.8 Was Java a good decision?

Java is a friendly language, it is verbose, very stable and it has a rich eco-system. On the other hand it requires more resources to be set-up than PHP and that is one reason why web hosts offer much less Java hosting. This was a big problem for us, because we could merely find an appropriate offer. Maybe next time we would write the project in PHP and probably facing other problems.

3.9 ES5 versus ES6

ES stands for ECMA Script and is the standardized JavaScript by the ECMA International organization. ES5 is the standardized fifth version of JavaScript and has been available since 2009. In 2015 the sixth version ES6 was released. The reason why many developers still stick with ES5 is that ES6 has not been implemented by all browsers yet. With the help of some transpilers ES6 could be used nonetheless as it brings many improvements. For example, it introduces classes and modules, promises, better block scope, default parameters, multi-line strings, iterators, arrow functions and many other enhancements. Nevertheless, we do not use ES6 directly but instead we use it combined with TypeScript.

3.10 Typescript

TypeScript is a programming language developed by Microsoft and released in 2012. It is a superset of JavaScript, which means that every JavaScript code is valid TypeScript. The main difference is that with TypeScript you can add types to your variables. And that is a huge advantage as IDEs can spot errors more easily and they can also make better code suggestions because of type inference. On small projects this does not matter but as the projects get bigger and more complex it helps a lot to write stable software. TypeScript also offers interfaces, enums, generics and much more. All these features convinced us to use TypeScript instead of JavaScript. So we currently use one server for backend and frontend.

3.11 Deployment for development

First we wanted to create two independent projects, one for our REST API and one for the front-end but we found out that the browsers did not allow us to access our back-end server because of SOP, which stands for Same-Origin-Policy and is a security feature that blocks such requests. To make server requests regardless of SOP we had to enable CORS, which stands for Cross-Origin Resource Sharing. We were not able to activate CORS on our Tomcat server, an alternative to CORS could have been JSONP, what stands for JSON with Padding, but we did not use it since the Angular Http library did not rely on it.

3.12 Deployment for production

The deployment was difficult, because we had an exotic set-up and it was our first time deploying an application of this dimension. At first we tried to find a good hosting service that would fulfil our needs.

Nowadays it is easy to get a free PHP hosting service because it is relatively cheap and easy for the host to set the server up, but since we are using Java for our back-end server it was not that easy to find a free host and the few that offered Java hosting charged for the database. We considered using Google App Engine and Heroku but both charged for the database.

Luckily we met a web host called Limitis based in South Tyrol, that offered us to host our application for free. We created a .war-file out of our Java web project and uploaded it to the server but it did not execute because the server ran an older Java version.

After fixing that problem by asking the web host to update their Java version we ran into our next problem. We were not able to connect to the database. The reason was that the database was running on another machine than the Java server and because of the security setting of the database we were not allowed to connect to it from the Java server. We fixed that problem by also asking our sponsor to set up a database on the same machine as the Java server, or allowing our application to connect to the database from a different computer.

3.13 Tomcat



Figure 3.2: Tomcat Logo

As mentioned earlier, we wanted to divide our whole project into two smaller projects, one for the back-end REST API and the other one for the web front-end, which would have made deployment of the application possible on two different servers.

We could have used a static file server for the whole front-end part, as there is nothing that has to be computed before it is sent to the client and for the back-end we would have used the Tomcat server, which is able to execute Java. Now we are stuck with deploying everything

on the Tomcat server because of the Same-Origin-Policy. The disadvantage is that every time something either on the front-end or back-end code is changed we are forced to redeploy the whole application.

3.14 Minification and bundling - WebPack

After having deployed our application we noticed that the web-app took a long time to load. For new users this loading time was even longer. This problem was caused by multiple reasons:

- Our JavaScript code was not minified
- The app had to make over 90 requests to the server
- First-time users have no cached code in their browser

For the development it is no problem not to minify the code, because the application runs on localhost and the files are transferred within milliseconds. The down-side of not minifying is that the files are very big and if the internet connection is slow the transfer is also slow, which results in a slowly loading web-page. For instance, Angular's code is bigger than 1 MB, which adds up to nearly 3 MB with our code and all other libraries we use. On smartphones the web-app was loading twice as slowly as in the browser. This was caused by the huge amount of requests that had to be dealt with. A PC can easily manage 100 requests, but phone browsers are not so powerful and the high request-amount makes the initial loading very slow.

Since Angular 2 was still in Beta it did not provide a built in feature of deploying a production-ready project. After searching for something that would manage this for us we bumped into Webpack.

Webpack is a module bundler that allows us to bundle our code and all the dependencies into one bundle.js file. The advantage of Webpack is that it also takes care of our Typescript files and transpiles them into JavaScript files. It even has the possibility to export development-bundle or production-bundle. The development-bundle contains all code as the developer coded it, however the production-bundle is just a minified version of the development-bundle. Minification means that all unnecessary characters get removed from the file without changing its functionality. New line characters, comments, etc. are removed and most important all variable and function names get a new, much shorter name.

With the help of Webpack we could reduce our code from 3 MB to under 1 MB and from over 90 requests to just 3 requests.

3.15 How our application works

In this section it gets a bit technical, we want to give an overview of how most of our components work. For this we choose the post component, not because it is the biggest or most

complex component, but because it is understandable and a good example for the workflow of our application. We start with the post-front-end. The postlist is an Angular Component

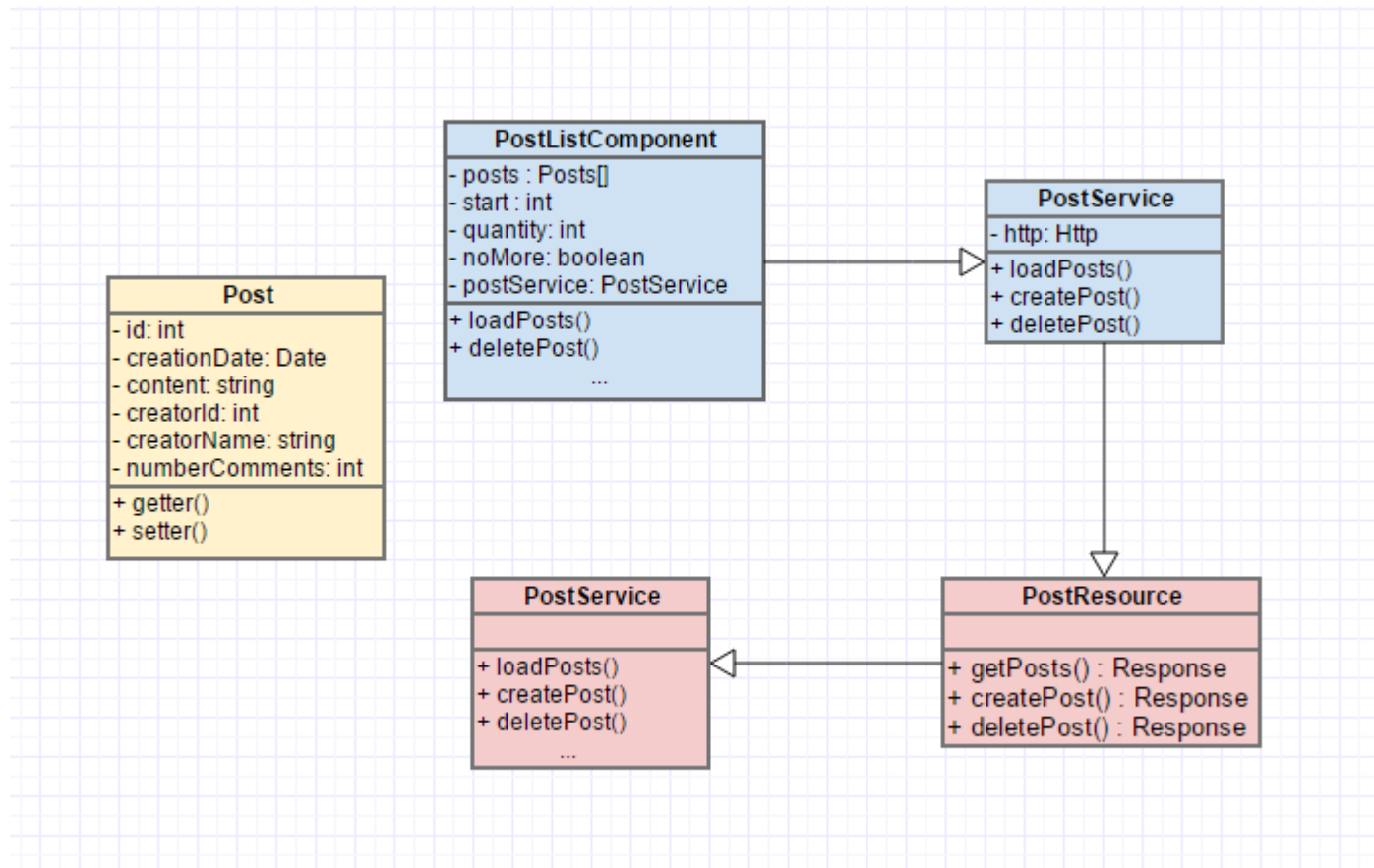


Figure 3.3: Class diagram of posts

which consists of a few sub-components, like the create-post-component or the comment-component, but we will not explain them further.

When users call the post-page in their browser a few things happen before anything is displayed to them:

At first all posts are loaded. For achieving this the post component uses the post-service. The post-service is a class which takes care of every interaction with the back-end-server that has something to do with posts as, for instance, loading posts. In other words, the post-service does the HTTP-Requests for us and returns the data we requested to the component.

When we want to load all posts the `getAllPosts()` method is called on the REST back-end. The servers structure is really simple: every component has its resource class with all CRUD operations and an appertaining service class with all CRUD operations on the database.

The resource takes care of validation and possible error-responses. If all input is valid the service method is called, which does the database query and returns the data to the resource, which again sends it back to the front-end service.

The advantage of this structure is that the resource is only responsible for communicating with the frontend and the service only for the database interactions. This results in well-structured, understandable and easily maintainable code.

With the help of Observables we can easily handle success and error situations on the frontend. If everything goes right, doing the HTTP-request, the success method is called, which displays the posts to the user, otherwise the error method is called, which shows an error-message.

The view of the component is defined in the template file, which is an .html file. It is pure HTML5 with a few Angular key words to bring logic to the code. With a simple for-loop, that is provided by Angular 2, all posts can be displayed dynamically.

3.15.1 PostComponent

```
15  @Component({
16      selector: 'posts',
17      templateUrl: 'postliste.html',
18      directives: [KommentarListe, PostErstellen, SicherModalComponent],
19      providers: [PostlisteService],
20      pipes:[DatumPipe]
21  })
22  export class PostlisteComponent {
23      private posts: Array<Post> = [];
24      // ...
25      constructor(private postService: PostlisteService){
26          this.loadPosts();
27      }
28      loadPosts(){
29          this.postService.loadPosts()
30              .subscribe((data: Array<any>) => this.posts = data,
31                      error => console.error(error),
32                      () => console.log("Fetched posts successfully"));
33      }
34      // ...
35  }
```

Figure 3.4: Sourcecode of the PostlisteComponent

The component annotation says that the following class will be an Angular component. This is new ES6/Typescript syntax. In the following lines we define some properties for the whole component:

- The selector describes how we can call this component in another html file, in this way when we use `<posts></posts>` our component will be inserted in that exact place.
- The `templateUrl` contains the url to the html-file in which the view of the component is specified.
- We need the next line, directives, to declare all subcomponents we want to use in our component. For example, if we want to include the `CommentsComponent`, we wrote previously we need to register it here, otherwise when we write `<comments></comments>` Angular will not know what we mean.
- Providers are used when a service is needed for this component. In this case we need to inject the `PostsService` to use its functions.
- Pipes are used to transform values when displaying them. Angular provides some built-in pipes such as `DatePipes` or `CurrencyPipes`. Pipes are also customizable, we wrote our own `DatePipe` to convert the date in our custom format. Since we want to use it in this component, we need to register it.
- Then we define a class called `PostlisteComponent` and with the `export` keyword we make this class in other files available.
- Then we define a member variable with a type in which we will put all the posts we fetch with the `PostlisteService`
- The constructor shows a neat TypeScript feature: it expects the `PostlisteService` on instantiation and Angular 2 injects this dependency. Next the keyword `private` tells TypeScript that this variable should be set as private member variable. With this you can save a whole line of boiler plate code.
- In the constructor the `loadPosts()` method is called, which is declared in the next line.
- `loadPosts()` calls the `loadPosts()` method of the `postService`, which returns a `Observable<Response>` to which we can subscribe. The `postService` does the http request to our backend server.
- On line 29 we give the subscription method an arrow function that assigns the data it retrieves from the subscription method to the `posts` member variable that is defined in line 23.
- When an error occurs on the fetching the error arrow function will be called that simply prints the error to the console.
- The arrow function defined on line 32 will be called if everything was fetched successfully.

4.1 Authentication

[eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMj0NTY3ODkwIiwiaXNwIjoibWVudDUyOTIyMCIsInZlZXItYWQiOjEsImNsYXNZLWlkIjozMjgsImFkbWluIjp0cnVlfQ.hdPdvRZ5zNLAT0AntwAUagnTzwWPkVL4r16qOFa5hAk](#)

The advantage of this is that there is no need to send user name and password in each request and that the server can exchange information that is signed with the client. JWTs are just Base64 encoded, so the connection between client and server should be secured. A JWT basically consists of three parts, separated by dots (Header, payload and signature). The header contains information about the signature algorithm, the payload contains user information and the signature is used to verify the authenticity of the token.

```
{
  "sub": "1234567890",
  "exp": "1234569220",
  "user-id": 1,
  "class-id": 328,
  "admin": true
}
```

Figure 4.2: Header of JWT

Figure 4.4: Signature of JWT

24

After this, a JWT is created and sent back to the new user.

If the users choose to log in with an existing account, the process is nearly the same: they fill in the login form, send the credentials to the server, the server verifies them and returns the token. The standard expiration time of the token is set to 30 minutes. Every 20 minutes the web-app automatically sends a new refresh-request to the server. In this way a single token is never valid for more than 30 minutes and if the user does not visit the website for more than 30 minutes he/she gets kicked out and needs to authenticate again. There is also the option to not get logged out by checking "remain logged in". In this case the expiration time is set to a whole week. Every time the user visits the page the token gets refreshed and is valid for another week, but if he/she does not do any action for a whole week his/her token is not valid any more.

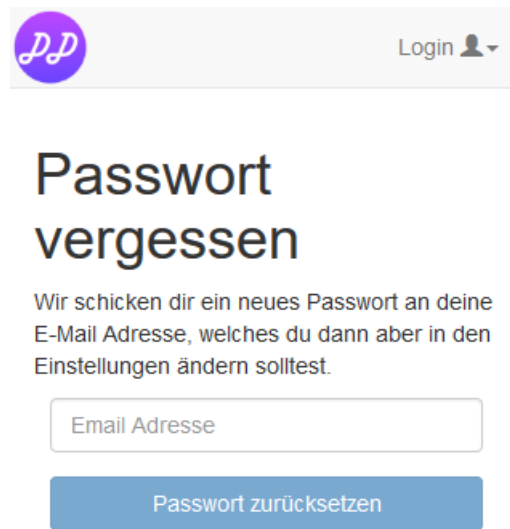
The generated token needs to be sent to the server in every request, in this way the web service knows if the user is authenticated and who is making the request, so he/she can, for example, deny access to a user that has no permission to access certain data.

4.2 Forgot your password?

As more and more classmates signed up on our application we found out that the users often forgot their password. Due to that we considered multiple solutions like "Security questions", "Password restore link" and others, but in the end we stuck to an easy to implement and secure solution that would not increase the application's complexity too much.

Any user can reset the password, the only thing they need for that is their email address. They enter it into a form and submit it. The server then looks up the database for an account with such an email address. If there is no such account the server will abort the process and inform the user that there is no account with such an email address. Otherwise the server generates a new and cryptologically secure password, which means that the password is cryptologically random enough. For the password generation we use the Apache Commons library. Then an email is generated, containing the new password which is sent to the submitted email address. Only if the email sending process is successful, the old password will be replaced by the newly generated one. For sending the emails we use the JavaMail library.

Lastly, the user will be informed that the password reset was successful and prompted to change it in the near future because of security reasons, as the new password is now visible in the email inbox.



The image shows a web interface for a 'Forgot password' page. At the top left is a purple circular logo with white stylized letters 'pp'. At the top right is a 'Login' link with a user icon and a dropdown arrow. The main heading is 'Passwort vergessen' in a large, bold, dark blue font. Below the heading is a paragraph of German text: 'Wir schicken dir ein neues Passwort an deine E-Mail Adresse, welches du dann aber in den Einstellungen ändern solltest.' Underneath this text is a white input field with the placeholder text 'Email Adresse'. Below the input field is a blue button with the text 'Passwort zurücksetzen' in white.

Figure 4.5: Forgot password page

4.3 SQL Injection

SQL injection is one of the three famous web attacks, where the attacker tries to execute harmful SQL queries to obtain access to all kinds of information that is stored in the database. To protect against SQL injection we exclusively use prepared statements that make SQL injections impossible as every user input is marked not to be interpreted and run by the database system. The use of prepared statements instead of plain statements was by no means more complex but provided an elegant protection to this kind of attack.

4.4 Cross Site Scripting - XSS

Cross site scripting, also known as XSS, is one of the three famous web attacks, where the attacker tries to submit inputs with harmful JavaScript code so that the JavaScript code is executed on victims browser. To protect against cross site scripting we use Angular 2, which has a build in shield against this attack, which means that every user input is escaped automatically by Angular 2.

4.5 Form validation

We validate every user input twice. First we validate the user input on the client, in the browser to directly give the user feedback and second we validate the user input on the server, because someone could manipulate the client-side validation, as we follow the motto: "Never trust the client". When the server detects invalid input, which theoretically should never happen since the validation already happened in the browser, it sends a message to the client, so that the user can correct his/her input and redo the request.

The image shows a web form for registration. It consists of four text input fields and one submit button. The first input field is labeled 'Name' and has a red error message below it: 'Name muss gesetzt werden'. The second input field is labeled 'Email' and has a red error message below it: 'Email muss gültig sein'. The third input field is labeled 'Passwort' and the fourth is labeled 'Passwort wiederholen'. The submit button is labeled 'Registrieren'.

Figure 4.6: Example of form validation on the sign-up page

A good front end validation is really important for the user experience, since the user needs to know what he/she is doing right and what wrong. Instant validation is a way to improve this experience. Instant validation checks if the content is correct while the user is typing and gives immediate feedback. In this way the user does not lose time submitting the form and re-checking what he/she did not enter correctly.

Angular 2 comes with various ways of handling client-side form validation. One of them is the FormBuilder, which helps to create forms with predefined, as well as custom-defined validators. The form is specified in the controller with all of its logic and validators and is then bound to the specific HTML-elements in the view where custom warnings can be output if a field is invalid. This increases the readability of the form considerably, since the logic is well separated by the view.

5 Database

5.1 Sql2o

Sql2o is a Java library for accessing the database with more comfort than just using plain JDBC.

Sql2o offers a few useful features, like the functionality to automatically map the result set of a query with a Java class without the need of parsing and initialising an object by the developers. The main reason for choosing Sql2o is that the code is a lot more readable, due to much less code than plain JDBC. With the help of Sql2o we have saved a lot of time and lines of code.

5.2 MariaDB

MariaDB is a fork of the very popular MySQL database and we use it because our server sponsor Limitis provided use with it. As we began developing the application we used MySQL and then the migration to MariaDB was without major problems.

5.3 Event list

The following code represents how we access the database. This example is code from our back-end server. It fetches a number of events that will occur in the future for a specific user. The code uses Sql2o to access the database and retrieves a Java list of events. The most difficult part was to write the SQL query and to find out if a specific event occurs at the same time as a lesson.

```

public static List<Event> getEventListe(int bid, int start, int anzahl){
    Connection con = Sql2oConnectionManager.getConnection();
    String sql =
1  SELECT eid      id,
2      etitel      titel,
3      ebeschreibung beschreibung,
4      estart      start,
5      e.kid       klassenId,
6      f.fname     fach,
7      f.ffarbe    farbe
8  FROM  events e
9      LEFT JOIN stunden s
10         ON s.stag = IF(Dayofweek(estart) = 1, 6, Dayofweek(estart) - 2)
11         AND s.sbegin <= Time(e.estart)
12         AND s.sende > Time(e.estart)
13         AND ( e.eende IS NULL
14             OR s.sende >= Time(e.eende) )
15         AND s.kid = (SELECT kid
16                     FROM  benutzer
17                     WHERE bid = :bid)
18      LEFT JOIN faecher f
19         ON s.fid = f.fid
20 WHERE  estart >= Now()
21      AND ( ( e.kid IS NOT NULL
22          AND e.kid = (SELECT kid
23                      FROM  benutzer
24                      WHERE bid = :bid) )
25          OR ( bid = :bid
26              AND e.kid IS NULL ) )
27 ORDER BY estart,
28          e.eid
29 LIMIT  :start, :anzahl;

    return con.createQuery(sql)
        .addParameter("bid", bid)
        .addParameter("start", start)
        .addParameter("anzahl", anzahl)
        .executeAndFetch(Event.class);
}

```

Figure 5.1: Method for requesting the database to retrieve an event list

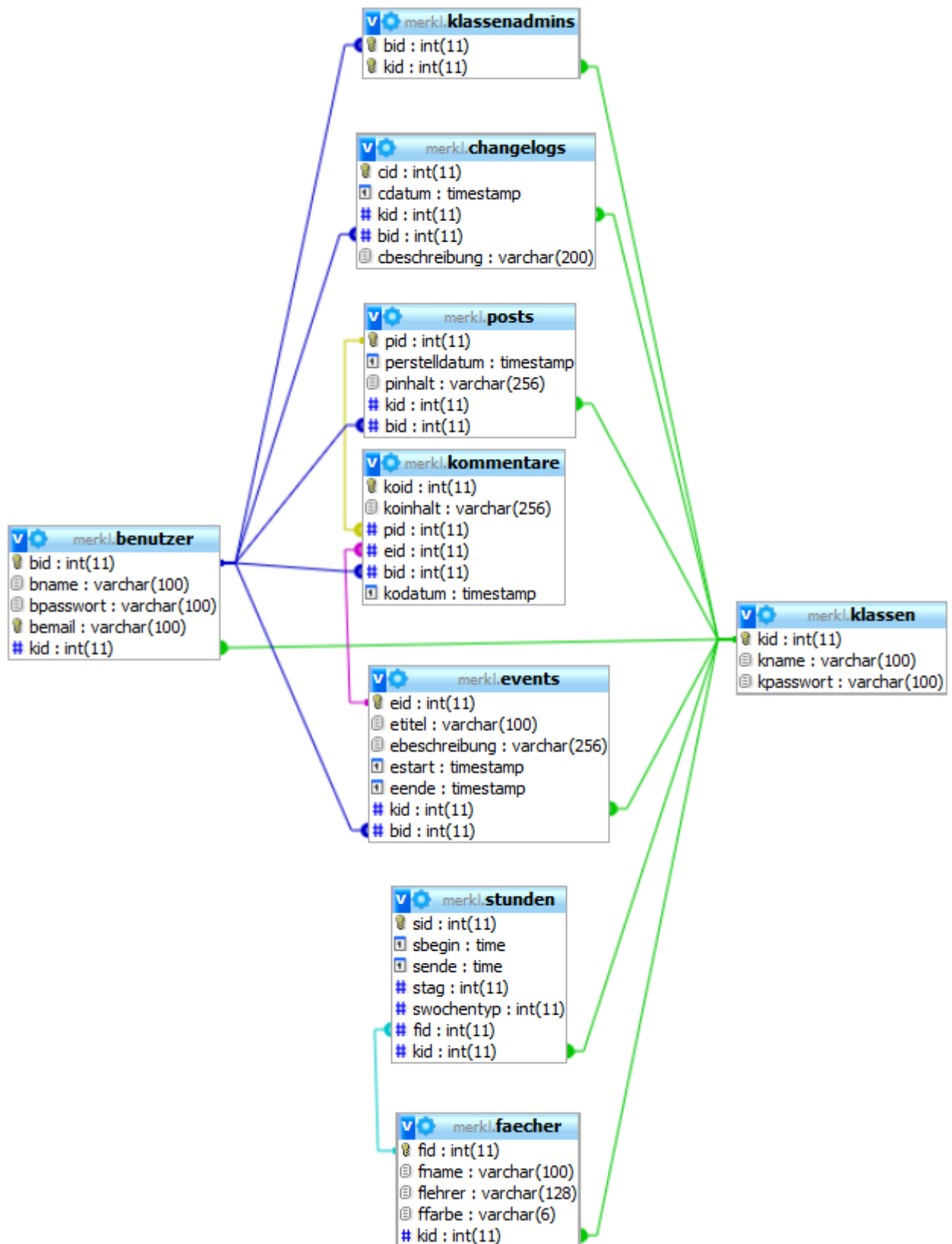


Figure 5.2: Structure of the database

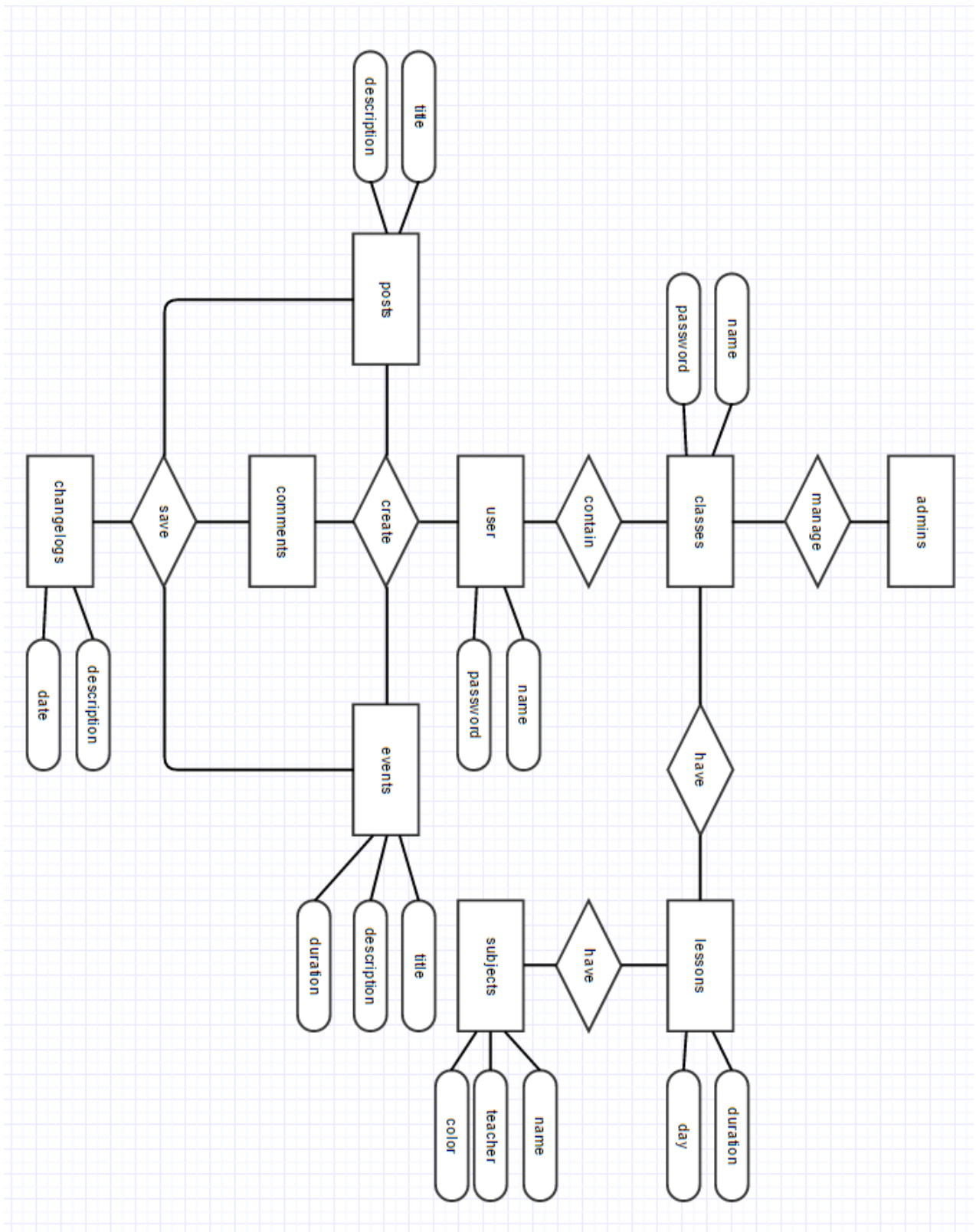


Figure 5.3: Database design

6 Design

6.1 UX - User Experience

Nowadays the design of a website is crucial for its success. We decided to provide the users the best possible UX, accordingly we invested much time optimising the use flow of the application. Our goal was to reduce the amount of clicks that a user has to make to carry out tasks. As today many websites are used from mobile devices we felt the necessity to make our design responsive, which means that the website's appearance is good on every device. As once Josh Clark, the author of "Seven deadly mobile myths" said: "Content is like water. You put water into a cup it becomes the cup. You put water into a bottle it becomes the bottle. You put it into a teapot, it becomes the teapot." Inspired by this quote we set our goal to make a beautiful but functionality rich application. Having gained experience with responsive design in previous projects, we knew that the responsive mobile first css-framework Bootstrap would be the best solution. Bootstrap is an open-source framework for creating responsive websites and web applications. It contains predefined design for forms, buttons and numerous other components, as well as JavaScript extensions, which allow the developer to create components with specific animation and behaviour, like for example an accordion. Setting a class to a HTML component acquires the bootstrap design. To provide a better user experience for new users we created many information bubbles, which give good advice.

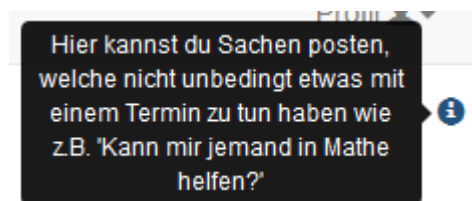


Figure 6.1: Information bubble

6.2 One design language

We tried to be consistent with our design. To achieve that we used the same design language on the website and in the app. One repeatedly appearing element is the "Event number Badge", which we use to show the user directly how many events he/she has in a specific time interval like a school hour or day. We show this Badge in the time table and in the

events list both on the website and in the app.

Another repeatedly appearing element is the "Private Badge", which shows if an event is private so that it only appears to the user or to every classmate. This Badge is also used in the events list and the detail view of the event, both on the website and the app. For showing the current day in the time table or month view we introduced the orange coloured font, so that a user immediately could see the most important elements of the time table.

Other details that most users may not even notice directly, but that their subconscious notices, are for example the button colours. Create and save buttons are always blue, cancel always grey and delete always red.

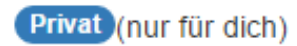


Figure 6.2: Private Badge

Franz vor 5 Stunden

Figure 6.3: Date format

Or just to name another one: the date-format. If something happens within the current day, we did not find it good to just output the plain date. Instead we calculate how much time ago it happened, in this way the user sees items like "just now", "12 minutes ago", or "5 hours ago", which says a lot more than indications like "Friday 23. May 2016 12:20".

We implemented a many other little details like these, and all together result into a much more complete experience for the user.

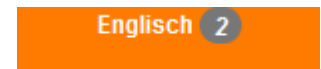


Figure 6.4: Number of events in that lesson

6.3 Gravatar

Gravatar is a web service that provides a picture related to a hashed email address. The picture is generated randomly with help of the hashed email address.

We decided to use Gravatar as a profile picture provider because it is popular, widely spread and additionally we did not want to store all the profile pictures on our server. The reason for that was that then we would have had to manage all these pictures, needed more data storage and would have annoyed the user for the necessity of uploading a picture to our service.

Because of privacy concerns we decided to randomise our users' email address so that a user could not be tracked on our application by Gravatar with the impact that a user could no longer set its profile picture by hand.



Figure 6.5: Gravatar logo



Figure 6.6: Random profile pictures

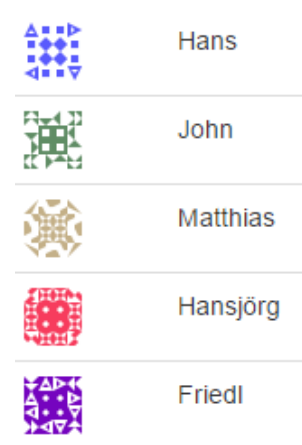


Figure 6.7: Random profile pictures

7 Development

7.1 Debugging

When errors occurred where the cause was not that obvious we used the internet to find out. We searched up the error message which most of the time led us to the website stackoverflow.com, which was very useful as we often found the solution for our very specific problem there. But many times we had to dig deeper as we were using very unstable frameworks with many bugs. To solve these problems we searched in the specific forums of the frameworks like the forum.ionicframework.com, where we could ask our questions and problems directly to the developer of these frameworks. There we often found out that the reason for our problem were bugs in these frameworks.

7.2 Browser compatibility

To get the website and app running on different devices and browsers we had to make a great effort. The first problem was that from the beginning Angular 2 did not support many browsers. The support did just come over time. Second, we often encountered problems with badly supported browser API's. For example, we used the browser function "toLocaleString" that is called on a date object to get a localized string representation of a date. On most browsers it worked except some like the android version of Firefox. Firefox output some bad formatted strings we could not use. So we wrote our own functions to bypass this issue. After some time when reading the changelog of Angular 2 we found out that this was a bug that now has been fixed. Other problems were specialized input fields like the date input field that is defined in the HTML 5 standard. Browsers often implement these standards very differently or not at all. These standardized input fields should provide an input validation and an easier and more appropriate input method depending on the device they are called but instead of enhancing the user experience and speeding up the development they only caused trouble. Additionally when more complex validations were required we had to implement it ourselves anyway. For example, we needed an input field for time. In the HTML 5 standard a time input field is defined, so we used it but it was only supported by some mobile browsers so we ended up with using an external library that provided us a clock picker. On the desktop browsers this solution was satisfactory but on the mobile browsers it did not run smoothly enough. We added the HTML 5 time input field again but now we had the problem that both clock picker opened when clicking into the input field, this was a really bad user experience and to solve that we needed to find out which browsers support the HTML 5 standard. For that, every time the site is called we create a time input field

with JavaScript and enter scrap data into the field, when we then try to read the data that stands in the input field and that data equals the scrap data we entered first then we know that the input field did not validate the entered data, so we know that the browser does not support that type of input field. When that is the case we open the external clock picker otherwise we don't. At the end we tossed out the external clock picker because we were not able to import it properly with webpack. What a sad story.

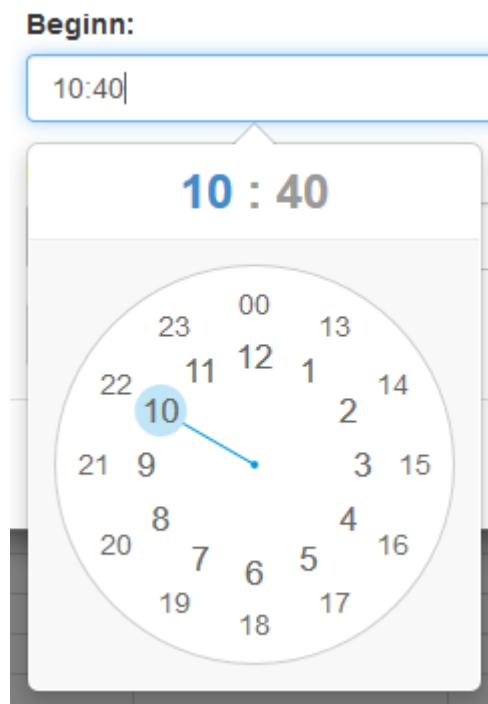


Figure 7.1: Firefox clock picker

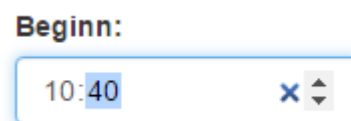


Figure 7.2: Chrome clock picker

7.3 Versioning / Git / Bitbucket

For good collaboration we needed a version control system, which would allow us to code at the same time without having the problem of merging the code into one common folder. We already used Git in previous projects, so it was clear that we would also use it for this project. We created a repository on Bitbucket since Bitbucket allows the developer to create private repositories for teams up to 5 members for free.

Using Git we have the advantage that every change on the code is saved with a specific id in the changelog. This saved us a lot of time finding bugs or performance issues from one version to the next. We only had to compare the old version with the new one and we immediately knew where we had to fix the problem.

7.4 Dropbox

Dropbox is a Cloud service that allows us to share files between computers. We use it to collect all kinds of documents that relate to the project like pictures, text documents or emails.

7.5 Tasker

Tasker is a program, developed by us, that allows programmers to collect and share all kinds of tasks related to a project for example bug-tasks or feature-tasks with their team mates. Since we developed it ourselves we use it a lot, because compared to other programs it fits our needs perfectly with its minimalistic style and merge-feature.

7.6 Eclipse

Eclipse is the most popular IDE, it stands for Integrated Development Environment, for Java development. It is open source and free, constantly improved by the community and extended by add-ons that the community creates. We use it as our main tool in the Java development of our REST API.

7.7 Atom

Atom is an open-source text editor, developed by GitHub, which is very extensible. We use it because the support of the IDE Eclipse for Typescript is poor where Atom instead has a very good support for Typescript.

7.8 NPM - Node Packet Manager

NPM is a program to obtain and share JavaScript code that is wrapped up in so called packages. Many projects use it for distribution like Angular 2. With NPM it is easy to obtain code from other projects and to keep this code up to date.

8 Mobile App Development

8.1 Hybrid apps

After we finished the development of the website we started the development of a mobile app. We decided to create a hybrid app, to reuse our codebase. A hybrid app is an app that is displayed in a webview but can also access native functions of the smart phone, like Storage, Camera or Datepicker. Cordova is such a hybrid mobile app framework. It connects the webview with and can be executed on various devices such as Android, IOS, Windows Phone, FireOS, Black Berry 10 and Ubuntu. Ionic 2 works on top of Cordova. It uses Angular 2 and HTML. Ionic 2 is a framework that allows programmers to develop such apps using HTML, CSS and Javascript. It provides prefabricated components like lists or buttons that look very native and change their appearance depending on the device it is executed.

8.2 Why Ionic 2?



Figure 8.1: Ionic logo

After concluding the website development we had a lot of code in our repository. We were afraid that it would take too long to write a mobile application from scratch, since the only thing that we would not need to rewrite was the web-service. But with these frameworks we could reuse a big part of our code.

8.2.1 Advantages

The fast development and the code reuse are one of the big advantages besides the broad support of devices. It is also very simple to customize the native look of the app.

8.2.2 Disadvantages

When we started with the development Ionic 2 was still in the beta phase. That was a problem as things changed, had bugs or were still unsupported for some devices. Another problem is that hybrid apps tend to be slower than native apps so for time critical apps we would not consider using hybrid frameworks. For Android we had the problem that the app did not run on phones lower than Android 4.3, the reason for that was that in these hybrid apps the default webview is used and that webview can't execute Angular 2 because on older Android versions the webview never gets updated as long as Android itself is not updated. So a normal webview update would not solve the problem but instead we had to use something called Crosswalk. Crosswalk a runtime for HTML applications and uses Google Chromium. In combination with Cordova Corsswalk it is wrapped up in the apk file which increases the size of the apk by 20 MB. Crosswalk allows us to deploy the app on Android 4.1+ devices and to use cutting edge browser technology. It also speeds the app up and makes the behaviour of the webview consistent and predictable.

8.3 Deployment

We deployed our app on the Google Play Store. For that we had to acquire a Google developer account for 25 euro. Then we had to change the config.xml, sign the apk with a cryptological private key, run it with a special alignment software and last we could upload it to the Play Store.

To deploy the app on the Apple App Store we would have needed an Apple Mac and Apple developer account with a yearly charge of 100 Dollar. We were not willing to pay that nor did we have an Apple Mac.

To deploy the app on the Microsoft App Store it was required to run a special build command in the ionic CLI (Command Line Interface) but since this command had not been implemented yet there was no way to deploy it.

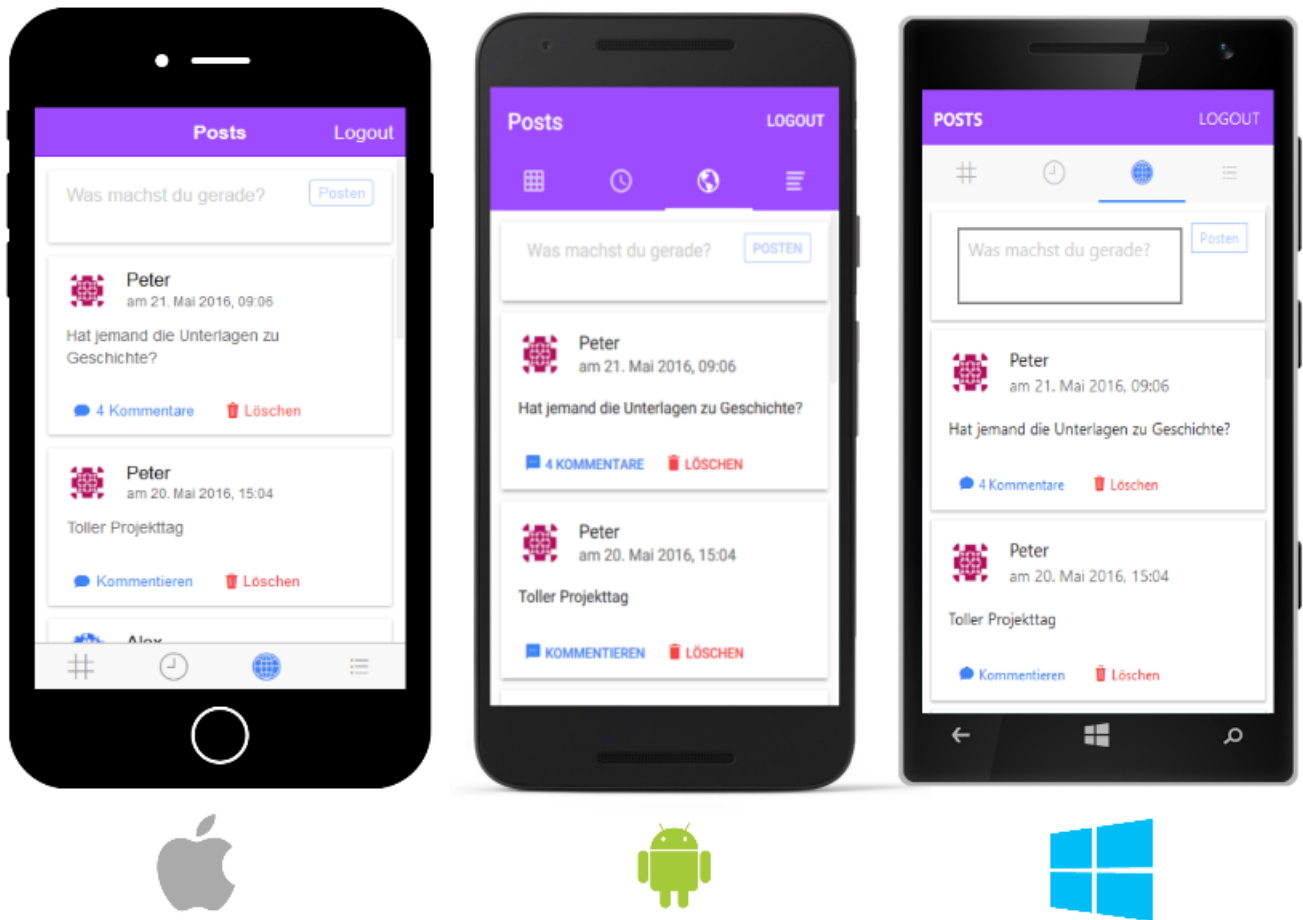


Figure 8.2: App on different devices

9 Appendix

9.1 Future prospects

During development we decided to cut some features: Repeating events, alternating timetables, signing up with Google, mark list, exam list as the implementation of all these features would require more time than we had.

The application does not support multiple languages yet. When focusing on the Italian market we could implement Italian since the application is currently only available in German.