Vuezeit - Implementing a new UI for Wahlzeit

BACHELOR THESIS

Hendrik Wagemann

Submitted on 5 July 2021

Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät, Department Informatik
Professur für Open-Source-Software

Supervisor:
Prof. Dr. Dirk Riehle, M.B.A.

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Versicherung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, 5 July 2021

License

Erlangen, 5 July 2021

Abstract

The Wahlzeit software is a website for sharing photos with other people. It was made by Professor Dr. Riehle. The Wahlzeit User Interface (UI) now has a very outdated look and feel. In addition to that, the way the UI is implemented is not up to date with modern standards. This thesis describes how the new UI for Wahlzeit was implemented. It uses Vue.js 3 together with Bootstrap 5 to accomplish that task. The UI now is a single page application frontend that interacts with a RESTful service backend, which is developed separately. The goals of making an appealing looking interface with an easy to understand implementation were met.

Contents

Acronyms

UI	User Interface
API	Application Programming Interface
CSS	Cascading Style Sheets

1 Introduction

Wahlzeit is an Open Source photo sharing website, made by Professor Dr. Riehle. Users can upload photos and rate photos uploaded by other people.

It is used as lecture material in their computer science university course "Advanced Design and Programming". In this course students work on the Java backend and learn how to design models in a way that is easy to maintain and extend.

The Wahlzeit software now is pretty outdated in its implementation and is undergoing an update. This thesis focuses on the frontend part of this process.

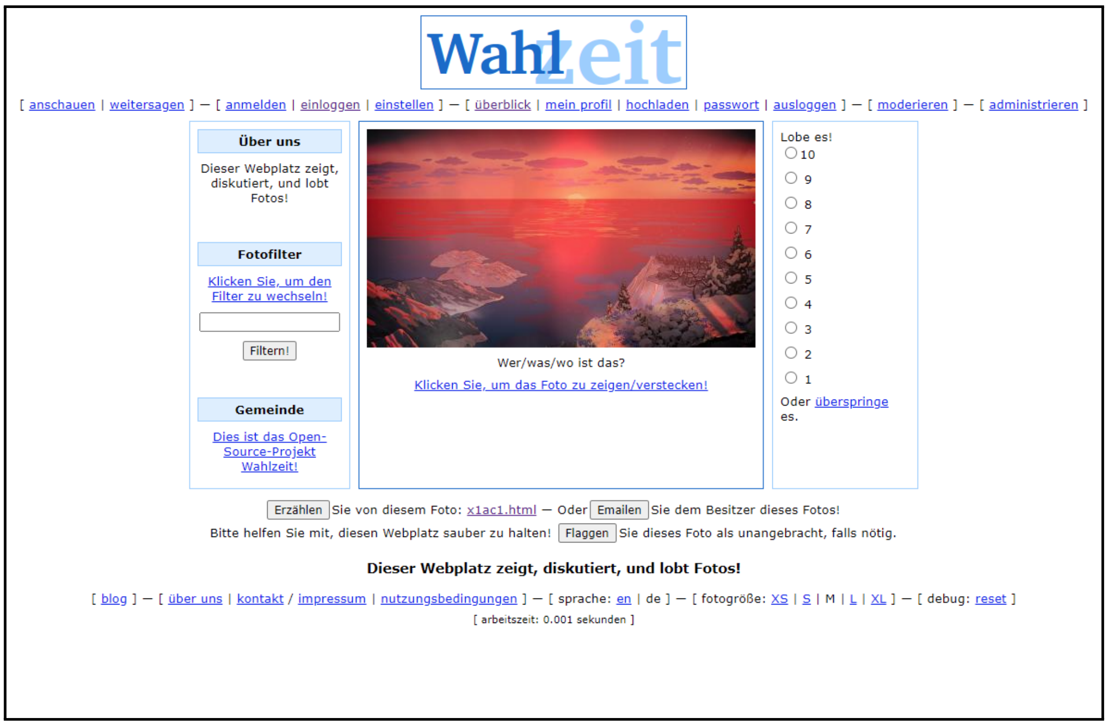
The goal of the new UI was to improve attractiveness and ease of use of the design. To achieve this, modern JavaScript and CSS techniques will be used to create an appealing dynamic and responsive design. The used techniques will be described in the following chapters. At the same time, the code needs to be easy to read and understand for new students every Semester, that have never seen the code before. Because of this, every used external libraries and the usage of their features need to be carefully evaluated to ensure readability of the code for new developers.

After outlining these problems with the old UI and the objectives for the new design in detail, this thesis will go into detail about the implementation. After that the result will be demonstrated and then evaluated in a comparison to the old UI. It will end with a conclusion and some ideas for future continued work.

2 Problem Identification

2.1 Outdated look and feel of current UI

One big problem with the current Wahlzeit UI is, that it just looks antiquated. It is very basic, being made from a lot of individual webpages and a lot of links between them. These many links can overwhelm the user, resulting in the link for the intended action not being found. The overwhelming amount of links in the old UI can be seen in figure 2.1. Every interaction leads to a new page being loaded from the sever. This static, non responsive design interrupts the users experience in using the website. There are none of the dynamic features that modern websites provide like modal windows and asynchronous content loading. This look makes the website a lot less appealing. The design also heavily hinders ease of use.



Wahlzeit

[anschauen | weitersagen] — [anmelden | einloggen | einstellen] — [überblick | mein profil | hochladen | passwort | ausloggen] — [moderieren] — [administrieren]

Über uns

Dieser Webplatz zeigt, diskutiert, und lobt Fotos!

Fotofilter

Klicken Sie, um den Filter zu wechseln!

Filtern!

Gemeinde

Dies ist das Open-Source-Projekt Wahlzeit!

Wer/was/wo ist das?

Klicken Sie, um das Foto zu zeigen/verstecken!

Lobe es!
10
9
8
7
6
5
4
3
2
1
Oder überspringe es.

Erzählen Sie von diesem Foto: x1ac1.html — Oder Emailen Sie dem Besitzer dieses Fotos!

Bitte helfen Sie mit, diesen Webplatz sauber zu halten! Flaggen Sie dieses Foto als unangebracht, falls nötig.

Dieser Webplatz zeigt, diskutiert, und lobt Fotos!

[blog] — [über uns | kontakt / impressum | nutzungsbedingungen] — [sprache: en | de] — [fotogröße: XS | S | M | L | XL] — [debug: reset]

[arbeitszeit: 0.001 sekunden]

Figure 2.1: Old photo page

2.2 Need to improve implementation

Another problem with the current Wahlzeit UI ist the implementation itself. The old implementation has its own simple implementation of a template engine, allowing only for simple format string style text replacement. This falls far behind existing template engines, that allowed to write inline code in a high level language like JavaScript or Python into an html template that gets rendered into the final html page. The old UI implementation also uses string concatenation to piece together html code from string constants and user variables.

```java
public static String asHref(String link, String body) {
    return "<a href=\"" + link + "\" rel=\"nofollow\">" +
           body + "</a>";
}

public static String asImg(String link, int width, int height) {
    return "<img src=\"" + link + "\" width=\"" + width +
           "\" height=\"" + height + "\" />";
}
```

Code 2.1: Example of html code built with string concatenation in the old Wahlzeit

This is tedious for the programmer and hard to read for anyone working on the software in the future. It is also prone to cause errors, especially without proper input sanitization, which is also handled in a custom implementation. It also has the disadvantage, that no development environment can offer proper html syntax highlighting for the code.

3 Objective Definition

3.1 Modern look and feel

The first goal for the design of the new UI was a more attractive look and increased ease of use. A responsive design that does not remind one of the 1990s, would not only make the look and feel of the website more appealing, but also increase the ease of use by a lot.

The new UI will consider intuitive guidance of the user, attractive styling and dynamic and responsive behavior. There are JavaScript and CSS frameworks and libraries available that provide thorough solutions for these problem by applying existing patterns. They for example provide simple solutions for creating appealing and responsive navigation bars and dropdown menus. Using these enhances the user experience and ease of use.

3.2 Use in teaching

The Wahlzeit software is used by Professor Dr. Riehle in their lecture "Advanced Design and Programming" for teaching purposes. This affects how the new UI is implemented. The lecture focuses on the backend part of Wahlzeit, but students looking at the UI should be able to understand that in a short amount of time. Code is read more often than written, especially in this setting where every year there are 30 new students looking at this code for the first time.

The first aspect is to keep the code as simple as possible to allow students to understand it very quickly without spending too much time reading it. This means sometimes not implementing some features that improve security or stability of the application as long as the chosen implementation is secure and stable enough for the purposes of Wahlzeit.

Another aspect to simplifying the UI implementation for teaching purposes is to keep the number of additional libraries used to a minimum. While adding additional libraries may simplify code further or make the application more secure, it would require learning how the library works to understand the code that uses it. So unless the benefits greatly outweigh the increased complexity, no extra dependencies should be added to the project.

4 Solution Design

4.1 Design Description of new UI

The UI design starts with the choice of the right language and tools. To achieve a dynamic and responsive design, a client side JavaScript implementation is necessary. TypeScript was chosen as a language on top of JavaScript to improve the code quality. Vue.js as a UI framework brings a template engine, and component system for easy and structured programming.

The UI is served as a single page, resulting in the entire application being loaded by the browser at once. This allows for a very responsive design and minimal waiting time on server responses.

4.2 Used Frameworks etc.

4.2.1 Preferred Language - TypeScript

JavaScript is known to allow mixing different types in different operations for vastly different, sometimes nonsensical results. This can cause a lot of different problems.

```
> "10"+1
< "101"
> "10"-1
< 9
```

Code 4.1: examples of weird JavaScript expressions and their results

Due to how JavaScript handles interaction of different data types, programming mistakes can ripple throughout an entire codebase. Code example 4.1 shows an example how arithmetic operations on a string and an integer, which, due to the two operands being of completely different types, does not make any sense, in JavaScript lead to vastly different and nonsensical results. This can lead to the mistake to be realized very late, and finding the exact line of code causing the problem can be difficult.

TypeScript is a type checking language, that is based on JavaScript. Any JavaScript code is valid TypeScript code. But TypeScript provides additional features on top of JavaScript to allow for more type safe code. These type checking features can be used to find these type mismatch based programming errors very quickly, and fix them without the need for intensive debugging, tracing nonsensical values to their root.[1] The TypeScript code gets compiled into JavaScript which is executed normally.

For this thesis TypeScript was chosen to reduce the risk of errors and produce simple to read code.

4.2.2 Preferred UI framework - Vue 3

To implement a UI like this, using a framework is necessary to avoid writing huge amounts of code yourself. There are different frameworks for this purpose available. Some of these JavaScript frameworks are Vue.js Angular or React. Vue.js was chosen for this project because of the capability to offer the following features.

Vue supports the use of TypeScript.

Version 3 was chosen over Version 2 to get the latest features.

[1]'Why does TypeScript exist?', 2021.

The new UI is set up using vue-cli, a tool that structures your project in multiple files in Vue components and uses those to build a single page website from that. This allows the programmer to structure different components in multiple files, but the end user only interacts with a single, dynamic website. A Vue component consists of an html template, a JavaScript or TypeScript script and some CSS. The html template can contain inline JavaScript that gets rendered into the final webpage. It also has special tags to add other Vue components. This results well structured and simple to read code.

```
<template>
    <h1 class="heading">{{ theHeading }}</h1>
    <AnotherVueComponent />
</template>

<script lang="ts">
import { Vue, Options } from ;
import AnotherVueComponent from "@/AnotherVueComponent.vue";

@Options{
    components = { AnotherVueComponent }
}
export default class MyComponent extends Vue {
    theHeading = "The Heading";
}
</script>

<style scoped="true">
.heading {
    color: red;
}
</style>
```

Code 4.2: example of a Vue.js Component using TypeScript classes

The code between the double braces in the html template is inline JavaScript or TypeScript code that will be evaluated and the result will be inserted into the html. In this example this results in a simple page with a red heading "The Heading". This heading text is tied to the TypeScript variable theHeading. Because of this, when the heading text needs to change, it is only necessary to assign the new value to that variable. The component AnotherVueComponent will be rendered in the same way and inserted below the heading. The CSS style is scoped, this means the classes inside will only be usable within this Component and not visible to other code in the webpage.

This shows how this component based system allows for easy structuring of different parts of the web application and separation of concerns.

4.2.3 CSS Library - Bootstrap 5

To get an appealing website, it needs to be styled with CSS. To avoid having to write a lot of custom CSS, an existing library with ready to use CSS style components can be used. Bootstrap is such a library that provides a lot of CSS classes to create nice and modern looking websites in a quick and simple way. Examples are navigation bars and containers for layouting components of the page. These components also adapt well to small screen devices like smart phones. It also provides some JavaScript module to provide responsive components like tooltips, modals and many more.

Version 5 of Bootstrap was chosen over version 4 also for access to the newest features. Version 4 also has an additional dependency on the JavaScript library jQuery which would cause conflicts with Vue. This is another reason Bootstrap 5 was chosen.

4.2.4 Wahlzeit API

The Wahlzeit app provides an API over a RESTful HTTP interface using the json format for the data. It provides multiple endpoints for photos, users and moderation. To interact with this API the new UI uses the JavaScript fetch API. The resulting objects will be wrapped using TypeScript for more type safety.

5 Implementation

5.1 The Vue Router and Views

This Application is served as a single page with the previously explained benefits in regards to responsiveness. The Vue Router is a component of the Vue framework that handles the access to the different parts of the website. This makes the single page appear to the user as if they were navigating multiple pages. These perceived different pages are called views. This section will first explain how the Vue Router works, then goes into the individual views.

5.1.1 The Vue Router

The Vue Router uses Views, which are just normal Vue components used in special circumstances. The Router is used to select a component as the view to display, based on the accessed url.

The Vue Router can operate in two modes, hash mode and history mode. To the end user, they behave the same.

```
http://localhost:8080/#/photo/1
http://localhost:8080/photo/1
```

Code 5.1: a hash mode url compared to a history mode url

Hash mode uses the url hash to serve the entire application at one url to prevent website reloading upon a change to the path. This is simple to work with, but results in not so nice urls. History mode uses the html5 history API to manipulate the url without a page reload. This requires further configuration of the webserver to work. But it results in better looking urls, that also match what a user would expect the url to look like. Hash mode also has a negative impact on Search Engine Optimization, this problem also does not exist with history mode.[1] Because of this, the new UI uses history mode.

[1]'Different History modes', 2021.

The router and its views are configured in a dedicated TypeScript file.

```
[
  {
    path: "/",
    name: "Home",
    component: Home
  },
  {
    path: "/photo/:id",
    name: "Photo",
    component: Photo,
    props: true
  }
]
```

Code 5.2: Example configuration for two a Vue router with two views

The above code example shows how to configure views for the Vue Router. The path specifies under what url the view should be reachable. The name can be used to refer to this view from code later on. The component specifies the Vue component that builds the root of the view. The path can contain parameters, for example a photo id. These parameters can the passed to the view component as properties by setting props to true.

The Vue router also provides utilities for generating links and navigating the page. The following code example shows how links can be generated in the html template using the router-link tag.

```
<router-link class="btn" :to="{ name: 'Home' }">
  Link text
</router-link>
<router-link class="btn" :to="{ name: 'Photo', params: { id: photoid } }">
  Another link
</router-link>
```

Code 5.3: Example for router-link

You can set a CSS class just as normal. The second link provides parameters for the properties specified for the view. You can also use the router to navigate the webpage from the TypeScript code. Navigating to the Home view for example would use `await this.$router.push({ name: "Home" })`.

5.1.2 Photo View

The photo view displays a single photo. It uses a few Vue components to structure its content.

The Description component displays the owners user name with a link to owners user view. It also shows information about the photo, like its assigned tags and the current praise score.

The PhotoActions component provides different buttons to interact with the photo. These include telling someone about the photo via email, or message the photos owner the same way. It includes a button to report the photo to the moderation team should it violate the website rules. The photos owner can also delete the photo here. To generate the different buttons for the owner and other users, a Vue feature called conditional rendering is used. This allows to to use conditions inside the html template code, via the v-if directive.

```
<template v-if="own()">
  <button class="btn btn-danger" @click="delet">Delete</button>
</template>
<template v-else>
  <Message btn-class="btn btn-secondary" :photo="photo" :owner="owner" />
  <Report btn-class="btn btn-danger" :photo="photo" />
</template>
```

Code 5.4: Code snippet showing the usage of the v-if directive

To use the v-if directive, the v-if attribute on a html element is set to a JavaScript condition. In the example above, the `own()` method from the TypeScript part of the Vue component is used to determine whether the condition is true. The element with the v-if attribute will only be rendered if the condition is true. A different element can be given the v-else attribute. The element with the v-else must directly follow the element with the v-if and will only be rendered when the condition of the v-if is false. The Vue `<template>` tag can be used to group multiple tags under a single v-if.

The last sub component is the Praise component. It offers the radio buttons used to give the photo a praise score.

The elements of the page are layed out using the Bootstrap grid system. This allows to structure content in rows and columns with a few simple CSS classes.

```
<template>
  <div class="row">
    <div class="col-md-10 border border-3 rounded-3">
      <img class="img-fluid" :src="src" />
    </div>
    <div class="col-md-2 border border-3 rounded-3">
      <Praise :photo-id="id" />
    </div>
  </div>
  <div class="row">
    <div class="col-md-10 border border-3 rounded-3">
      <Description :photo="photo" :user="user" />
    </div>
    <div class="col-md-2 border border-3 rounded-3">
      <PhotoActions :photo="photo" />
    </div>
  </div>
</template>
```

Code 5.5: html part of the PhotoView showing the use of Bootstrap classes

The CSS class `row` groups elements into rows of the grid. The `col-md-n` classes are used to assign individual elements into columns. The n specifies the width of the column. The total width of a row is 12. So an element with class `col-md-10` uses 10/12th of the row space. Bootstrap has built in mechanics that break columns into multiple rows should the viewport width be too small, for example on a smartphone, but also dynamically when resizing the browser window on a desktop computer.

5.1.3 User View

The user page provides information about a particular user, like username and email address. It also displays a list of all the users photos. To list the photos a feature of Vue called list rendering is used. It allows to write a for loop style structure inside the html template code. To use list rendering a v-for attribute is assigned to an html element. The value of the attribute has to be of the form `item in list` where list is a JavaScript/TypeScript array. The element with the v-for attribute will be repeated once for every item in the list.

```
<template v-for="photo in photos" :key="photo.id">
  <PhotoSummary :photo="photo" />
</template>
```

Code 5.6: code snippet showing the use of the v-for directive

In the user page this is used with a list of the users photos fetched from the API. The key attribute on the element with the v-for directive is bound to a unique value for each item. Here the photos unique id is used for this purpose.

The PhotoSummary component provides similar information to the photo view, but in a more compressed space. It includes a small thumbnail and some information like the current praise score and the assigned tags. The same Vue component that provides the interaction buttons on the Photo view is used here too, so all these interactions are available here too.

5.2 Modals

A Modal is a popup window embedded in the website. In the new UI, modals are used for various different purposes. Those are for example the login dialog and the file upload dialog. This prevents navigating to a new page for every single action, as was the case with the old UI. The transition to the modal is smoother than the transition to a new page, resulting in a more responsive experience.

To implement the modals it uses the modal component from Bootstrap. There is a generic Modal component that provides a slot and some properties for configuration. This component consists of a button/link that opens up the modal and the modal popup itself. The Modal has a property for the title and an optional confirmation button. The component has a property to configure the buttons CSS classes to style it as needed. The content for the modal is provided through the default slot.

The modal content is moved to a dedicated section in the document body. For this purpose Vue has a special teleport tag. It is used to move certain part of a Vue component to a different part of the website, away from where the component was originally included. The destination is specified in the `to` attribute of the teleport tag.[2] This is to prevent the modal to fall behind its own backdrop, making it impossible to be interacted with. The backdrop is the the transparent gray area that hides the webpage and makes the user unable to interact with it, while a modal is open. Clicking on the backdrop closes the opened modal. Each individual modal is its own component, using that base modal component.

The following code example shows how to use the generic Modal component to implement a specific modal.

```
<template>
  <Modal
    :btn-class="btn btn-primary"
    id="modal-id"
    @confirm="confirm"
    modal-link="Open Modal"
    modal-header="Modal Header"
    modal-button="Confirm"
  >
    Modal body content.
  </Modal>
</template>

<script lang="ts">
import { Options, Vue } from "vue-class-component";
import Modal from "@/components/modals/Modal.vue";

@Options({
  components: { Modal }
})
export default class SpecificModal extends Vue {
  confirm() {
    // this code is executed when the Confirm button is pressed
  }
}
</script>
```

Code 5.7: Example use of the generic Modal component

In the template section the Modal tag is used to specify the content of the modal. The btn-class property is used to specify the CSS class attribute of the button/link for opening the modal. The code above uses some Bootstrap button classes as an example. The text for the opening button is set using the modal-

[2]'Vue Teleport', 2021.

link property. The modal-header and modal-button properties are used to specify the heading and text of the confirm button respectively. The modal-button property can also be set to any false value (ie. an empty string or the boolean false value using v-bind) to remove the confirm button and only have a close button. The modal body content is specified inside of the Modal tag.

This current modal design looks good, and has a nice fade in and fade out transition. It also has a few problems. Opening and closing the modals is handled from within bootstraps own JavaScript code. Including the Bootstrap JavaScript code in TypeScript does not work properly, so to interact with the modal from within the Vue component TypeScript code a workaround is used. Due to how the UI is structured this workaround was not needed in the final version of the code. Because of how the bootstrap modals work and are used in Wahlzeit, they all exist at all times in the background and are shown and hidden as needed.

Some modals exist multiple times in the same view. This is the case with the modals for interacting with photos (Tell, Message, Report) in the photo list on a users page. To avoid conflicts with html element ids, manual precautions must be taken to ensure unique ids.

This is a general issue when using the same Vue component multiple times, for which Vue does not offer a built in solution. There are different plugins out there that solve this in a simple way, but they are not used in Wahlzeit to avoid adding additional dependencies. The id issue appeared mostly in combination with the multiple modals and their form fields. To fix this, the photo related modals use the corresponding photo id to generate a unique id attribute.

Most of the time modals are used for forms in this project. The login modal for example has a form with an input field for each the username and password. The upload modal has a file select input, to select the photo to upload, and a text input for adding tags. Sometimes the modal serves a different purpose. An example is the upload modal, which, after successfully uploading a photo, changes to only show a success message.

To display the labels for a form field, Bootstrap is used to generate floating labels. Floating labels combine the label for an input field with the input area. When the input field is empty, the label is shown as placeholder text inside the input field. Once the user starts typing, the label moves up, above the users input, still showing, now a bit smaller, inside the input field.

5.3 API Interface

The new UI sends requests to the API using JavaScript fetch. The fetch response object provides us with a `.json()` method to decode the data from the API into a JavaScript object. To integrate these into the TypeScript type system an interface is used. This works well as long as there is no mismatch between the interface and the Json data. Production code might want to add some checks here, to ensure the received data matches what is expected. For Wahlzeit this would be overkill and to satisfy the goal of having as simple code as possible the simple interface is enough.

```json
{
  "email": "root@localhost",
  "id": 0,
  "name": "admin"
}
```

Code 5.8: example json response for a user object

```typescript
interface User {
  id: number;
  name: string;
  email: string;
}
```

Code 5.9: TypeScript interface for the user object

Due to the use of only a TypeScript interface, there are no runtime checks whether the object members are as expected or not. This results in undefined member values should the API json data not match the interface definition. This simple approach was chosen for Wahlzeit due to its simplicity and straight forwardness to satisfy the requirement for use in the lecture.

6 Demonstration

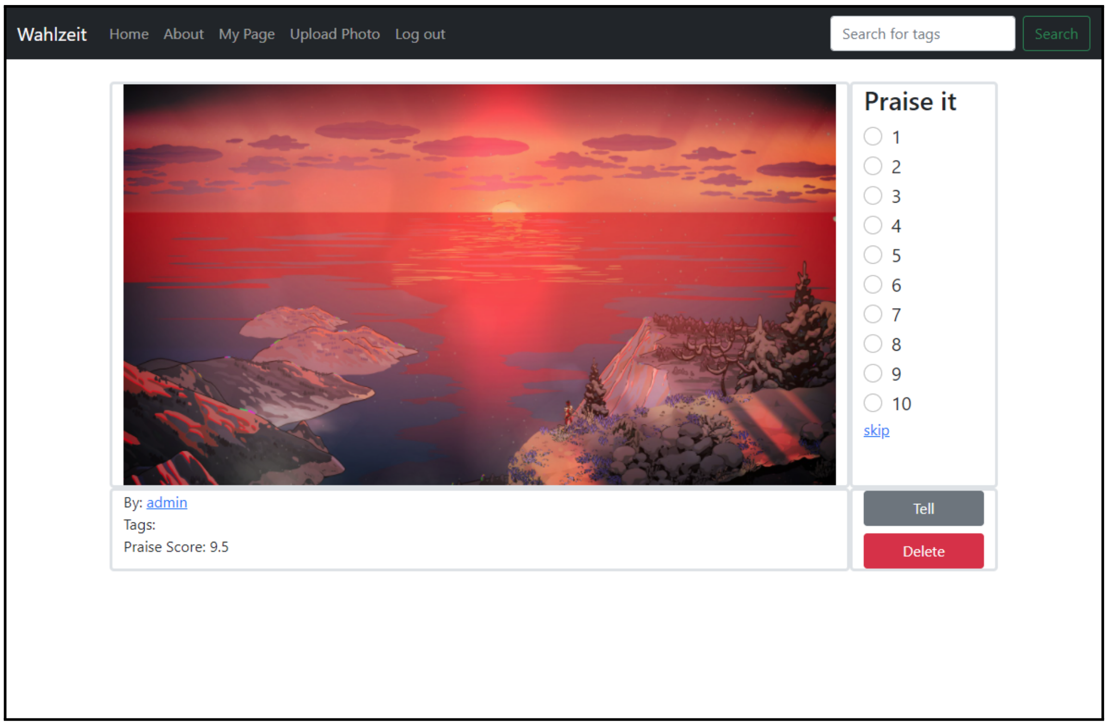
This chapter describes the results of the work done on the UI.

Code example 6.1 shows the new photo page. It consists of four parts. The first is the photo itself, which takes up most of the space. The next part is the praise options, a list of radio buttons allowing the user to rate the photo. The description below the photo provides information about the user who uploaded the photo, the photos assigned tags and the current praise score.

The last part are a set of buttons for interacting with the photo. These include the options to tell a friend about the photo, or message the owner. Should the photo violate the websites rules, it can be reported to the moderator. The photos owner can delete the photo. Once the backend API provides a fitting interface, these actions will include editing the photos metadata like tags or selecting it as a profile picture.

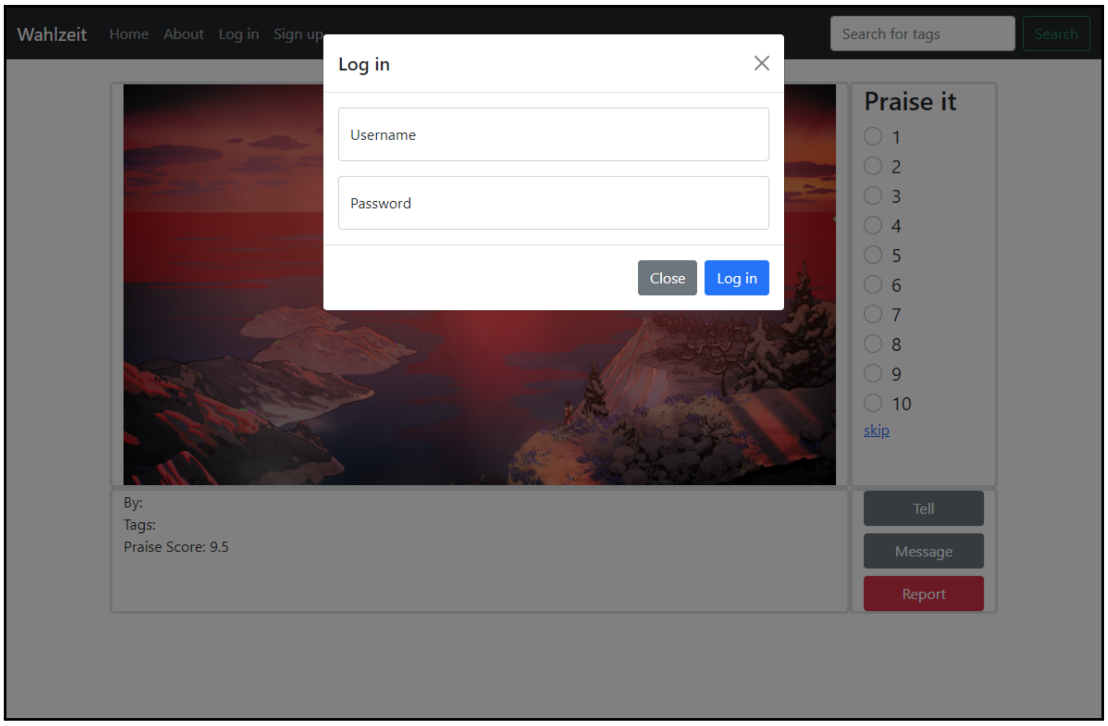
The modals make the new UI feel more smooth, getting to login and similar locations without needing to load a new page. Figure 6.2 shows the login modal opened over the photo page. It can be seen on its own in Figure 6.3a.

Figure 6.4 show how the floating labels work. As long as there is no input in the form field, the placeholder text is shown inside. Once the user starts typing, it moves above the input as a floating label.



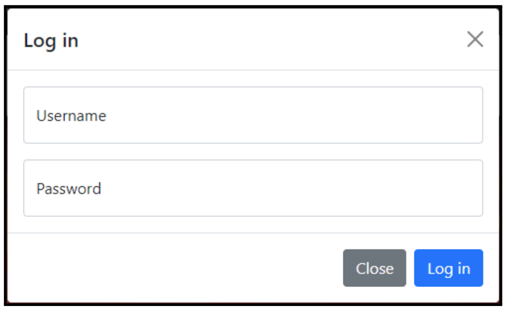
By: admin
Tags:
Praise Score: 9.5

Figure 6.1: New photo page

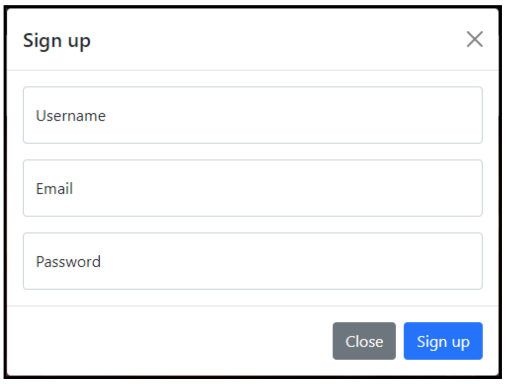


By:
Tags:
Praise Score: 9.5

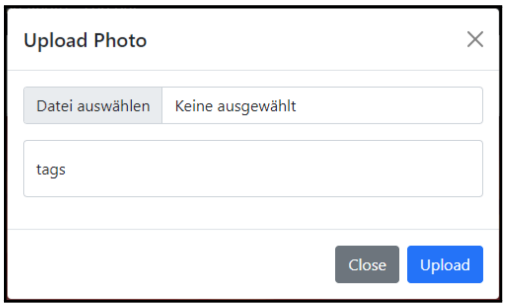
Figure 6.2: New photo page



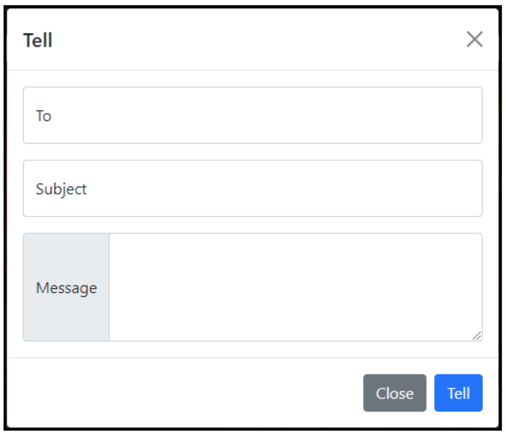
(a) Login modal



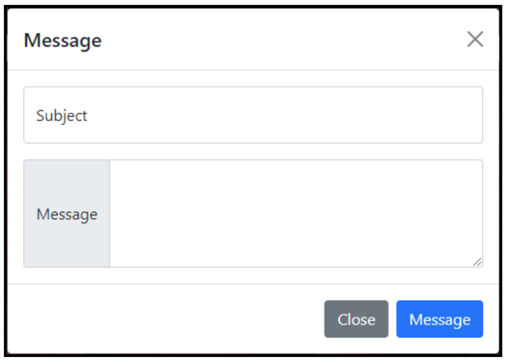
(b) Signup modal



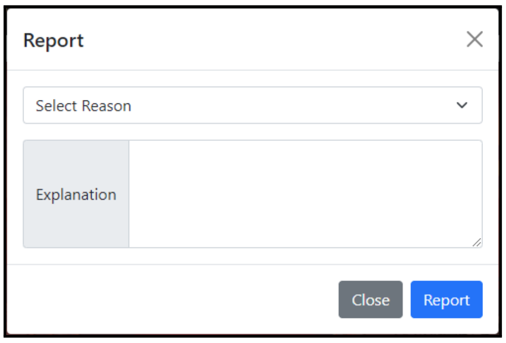
(c) Upload modal



(d) Tell modal

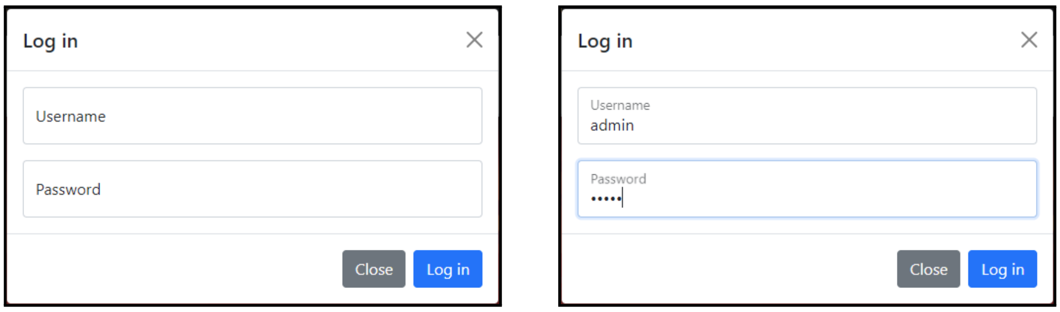


(e) Message modal



(f) Report modal

Figure 6.3: Images of the modals in the new UI



(a) No input - placeholder text **(b)** With input - floating label

Figure 6.4: Demonstration of floating labels on the login modal

7 Evaluation

7.1 Modern look

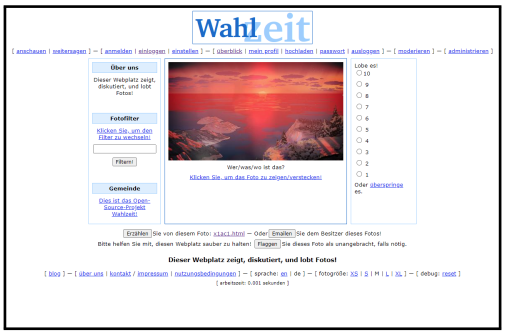
Figure 7.1 shows a comparison between the photo page in the old and new UI.

The new modals improve the interactivity of the website a lot compared to the old website where these forms where on a different page that had to be loaded first. An example can be seen in figure 7.2.

A nice side effect of using Bootstrap for the new UI is that the pages fit better on mobile phones or other devices with small screens. This can be seen in Figure 7.3. This is the case even without putting any focus on making the website suitable for mobile, Bootstrap does all that without much extra work.

7.2 Cleaner implementation

The separation of front and backend already greatly increases the readability of the Wahlzeit code. But also the code changes that only influence the UI have made significant improvements to the code clarity. Comparing the code of the two differetn implementations directly sis difficult, since they use a fundamentally dif-


Wahlzeit
[anschauen | weitersagen] — [anmelden | einloggen | einstellen] — [überblick | mein profil | hochladen | passwort | ausloggen] — [moderieren] — [administrieren]
Über uns
Fotofilter
Filtern!
Gemeinde
Lobe es!
10
9
8
7
6
5
4
3
2
1
Erzählen
Emailen
Flaggen

Dieser Webplatz zeigt, diskutiert, und lobt Fotos!

Klicken Sie, um den Filter zu wechseln!

Dies ist das Open-Source-Projekt Wahlzeit!

Wer/was/wo ist das?

Klicken Sie, um das Foto zu zeigen/verstecken!

Oder überspringe es.

Erzählen Sie von diesem Foto: x1ac1.html — Oder Emailen Sie dem Besitzer dieses Fotos!

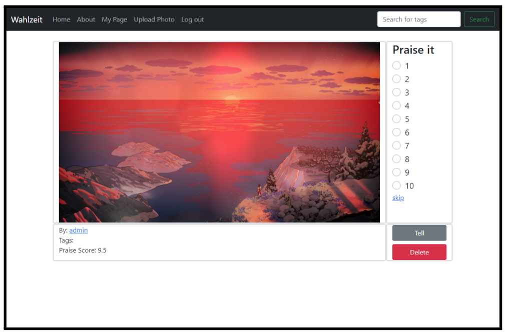
Bitte helfen Sie mit, diesen Webplatz sauber zu halten! Flaggen Sie dieses Foto als unangebracht, falls nötig.

Dieser Webplatz zeigt, diskutiert, und lobt Fotos!

[blog] — [über uns | kontakt / impressum | nutzungsbedingungen] — [sprache: en | de] — [fotogröße: XS | S | M | L | XL] — [debug: reset]

[arbeitszeit: 0.001 sekunden]

(a) old (see figure 2.1)



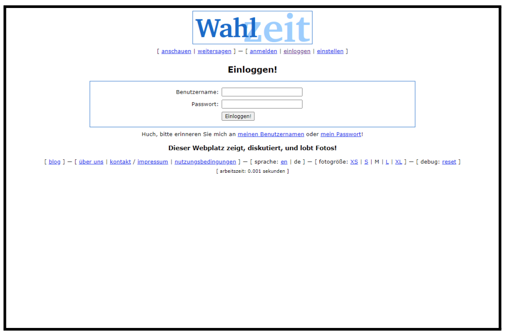
By: admin

Tags:

Praise Score: 9.5

(b) new (see figure 6.1)

Figure 7.1: Comparison of photo pages


Wahlzeit

[anschauen | weitersagen] — [anmelden | einloggen | einstellen]

Einloggen!

Benutzername:

Passwort:

Einloggen!

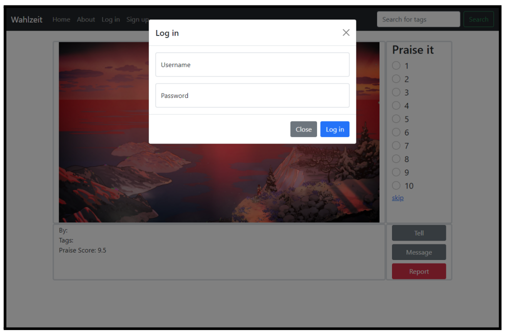
Huch, bitte erinnern Sie mich an meinen Benutzernamen oder mein Passwort!

Dieser Webplatz zeigt, diskutiert, und lobt Fotos!

[blog] — [über uns | kontakt / impressum | nutzungsbedingungen] — [sprache: en | de] — [fotogröße: XS | S | M | L | XL] — [debug: reset]

[arbeitszeit: 0.001 sekunden]

(a) old login page


Wahlzeit Home About Log in Sign up

Search for tags Search

Log in

Username

Password

Close Log in

Praise it

1 2 3 4 5 6 7 8 9 10

skip

By:

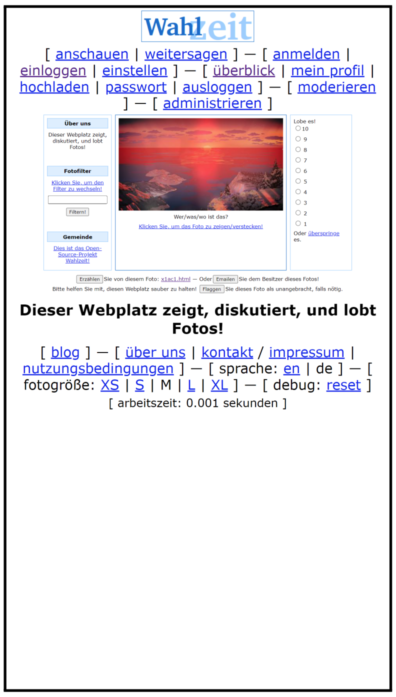
Tags:

Praise Score: 9.5

Tell Message Report

(b) new photo page with opened login modal (see figure 6.2)

Figure 7.2: Comparison of login pages


Wahlzeit

[anschauen | weitersagen] — [anmelden | einloggen | einstellen] — [überblick | mein profil | hochladen | passwort | ausloggen] — [moderieren] — [administrieren]

Über uns

Dieser Webplatz zeigt, diskutiert, und lobt Fotos!

Fotofilter

Klicken Sie, um den Filter zu wechseln!

Filtern!

Gemeinde

Dies ist das Open-Source-Projekt Wahlzeit!

Wer/was/wo ist das?

Klicken Sie, um das Foto zu zeigen/verstecken!

Lobe es!

10 9 8 7 6 5 4 3 2 1

Oder überspringe es.

Erzählen Sie von diesem Foto: x1ac1.html — Oder Emailen Sie dem Besitzer dieses Fotos!

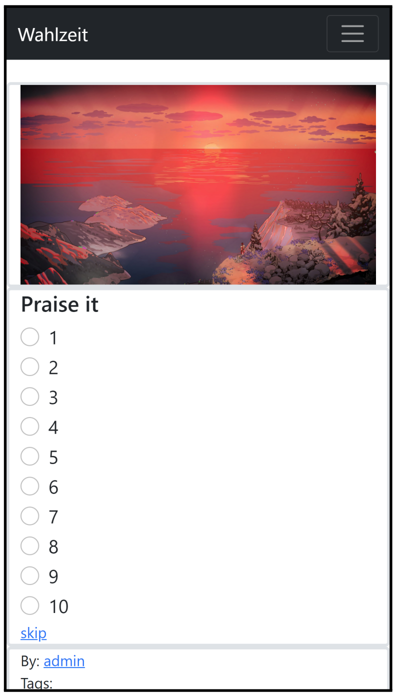
Bitte helfen Sie mit, diesen Webplatz sauber zu halten! Flaggen Sie dieses Foto als unangebracht, falls nötig.

Dieser Webplatz zeigt, diskutiert, und lobt Fotos!

[blog] — [über uns | kontakt / impressum | nutzungsbedingungen] — [sprache: en | de] — [fotogröße: XS | S | M | L | XL] — [debug: reset]

[arbeitszeit: 0.001 sekunden]

(a) old


Wahlzeit

Praise it

1 2 3 4 5 6 7 8 9 10

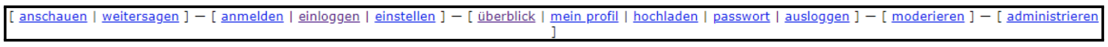
skip

By: admin

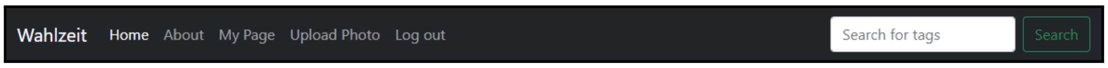
Tags:

(b) new

Figure 7.3: Comparison of Photo pages on mobile


[anschauen | weitersagen] — [anmelden | einloggen | einstellen] — [überblick | mein profil | hochladen | passwort | ausloggen] — [moderieren] — [administrieren]

(a) old


Wahlzeit Home About My Page Upload Photo Log out

Search for tags Search

(b) new)

Figure 7.4: Comparison of old menu links and new navbar

ferent approach. The string concatenation methods from the old implementation, that were heavily critizised in chapter 2, are completely absent in the new implementation. They were substituted be the use of a more powerful template engine, resulting in more readable code, or with the use of Vue constructs that generate the required code for us. An example for such a construct is the router-link tag.

7.3 Evaluation of number and complexity of used dependencies

The new UI implementation only has two dependencies. Those are the Vue.js framework, and the Bootstrap CSS library.

You need some framework for making a JavaScript application like the UI Wahlzeit. Vue components are very easy to read even for people not familiar with the framework, due to their structure. This makes Vue a very good choice for Wahlzeit.

Bootstrap has a good collection of ready to use CSS classes. It is also very well documented. The only alternative to using a library like Bootstrap is writing a lot of custom CSS. Doing so would result in a lot of extra code, making the UI code a lot harder to read. The addition of a simple library like Bootstrap reduces the overall complexity. Bootstrap does also come with a lot of advanced features that are quite complex. The usage of those feature is kept to a minimum in this project.

8 Conclusions

The implementation of the new UI was a success. It does look a lot more appealing than the old UI, and due to its new dynamic features it is also a lot more responsive. Because of that it is clear, that the first goal set out at the beginning was met. The second goal of keeping the implementation simple due to the projects use in a university lecture was also met, due to the simplicity and readability of the code. The fact that the frontend only relies on two dependencies, both absolutely necessary to implement a UI like this with a manageable codebase, also contributes to that. There are a few features from the old website, that couldn't yet be implemented in the new version. But due to the finished groundwork and the simple structures of the code, adding these will be easy one their requirements are fulfilled.

While the new UI is in a very good state, there are still some improvements that could be done in future works.

One such improvement can be made with the Modal component. While the current implementation does work, a different approach might yield some benefits. A different solution would have been to make a custom modal implementation with Vue, using bootstrap for CSS styling only. The initial impression was, that this would require a lot more code to be written, but in hindsight that would probably not even be the case. This solution would completely bypass any complications with external JavaScript code, since there wouldn't be any used anymore. It would also easily allow for the modal body to only be present in the html page when the modal is open. That would also bypass the conflicting id problem, since only up to one modal would be open at a given time, so its content and their ids would only exist once. Any interactions with the modal itself would also be resolved directly by Vue, not relying on id attributes for this and reducing the surface for possible conflicts. Due to the modular nature of the Vue components, replacing the modal implementation would work without complications, only the base component would need to be replaced (and possibly the individual modals should the need for adjusting the interface arise).

A few features couldn't yet be fully implemented, due to the API not supporting the required actions yet. One such feature is the reporting of photos and the handling of these reports by the moderators. Some endpoints do exist in the API for this, but they are not yet functional. The tell and message functionality does not send any emails at the moment. There is an existing email setup in the backend for the purpose of sending notifications to users. Adding email functionality to the frontend would be redundant. JavaScript (and in extension TypeScript) also does not have the capabilities to send emails.[1] Because of this an additional library utilizing a third party backend server would have to be used, which not only goes against the goals of this project, it's also unnecessary when there already is an existing backend with the required capabilities. To finish the tell and message features, fitting endpoints need to be provided in the API.

Wahlzeit is now available with a better look and feel and well structure and readable code.

[1]'How to send an email from JavaScript', 2021.

References

Different history modes. (2021). Retrieved June 18, 2021, from https://next.router.vuejs.org/guide/essentials/history-mode.html

How to send an email from javascript. (2021). Retrieved June 29, 2021, from https://stackoverflow.com/questions/7381150/how-to-send-an-email-from-javascript

Vue teleport. (2021). Retrieved June 29, 2021, from https://v3.vuejs.org/guide/teleport.html

Why does typescript exist? (2021). Retrieved June 18, 2021, from https://www.typescriptlang.org/why-create-typescript