Dokumentation

Feueralarm Berufliche Schulen Obersberg Bad Hersfeld

Lorenz Matthäus, Silas Manns, Manuel Pilz, Jakob Körber, Samuel Schütrumpf, Justin Delle

Inhalt

1. Vorwort

Zur Auswahl der Projektarbeiten gab es 5 verschiedene Projekte. Unsere Gruppe bekam das Projekt der Anwesenheitsdokumentation der Klassen im Falle eines Feueralarmes. Diese Anwesenheitsdokumentation der Klassen wurde bereits im vorherigen Jahr behandelt, allerdings hatte das vorherige Projekt klar erkennbare Schwächen. Unser erstes Ziel war daher eine neue Version zu konzipieren, da die Umstrukturierung des letzten Projektes mit mehr Arbeit verbunden wäre. An dem Projekt arbeiten 3 Anwendungsentwickler und 3 Systemintegratoren. Ziel des Projektes ist, die Aktionen, die während eines Feueralarms zu tätigen sind, vollständig zu digitalisieren und über alarm.bso-hef.de abzurufen. Die nötigen Aktionen während eines Feueralarms sollten dabei so schnell und einfach wie möglich, mit einer geringen Anzahl von Nutzerschritten, verarbeitet und dargestellt werden. Die Herausforderung bei diesem Projekt stellte sich dabei, eine Hosting Möglichkeit zu finden, die für die Schule kostenfrei ist. Hierbei stellen wir der Schule mehrere Möglichkeiten zur Verfügung: Hosting auf einem Virtuellen Server bei der HEFCOM IT GmbH als Sponsor, Implementation auf dem derzeitigen Webspace der Schule oder 500 freie Stunden pro Monat der Nutzung durch den Hoster Heroku.

2. Abkürzungsverzeichnis

API ..... *Application Programming Interface*
App ..... *Applikation*
DB ..... *Datenbank*
JS ..... *JavaScript*
NPM ..... *NodeJS Package Manager*
GUI ..... *Graphical User Interface*
HTML ..... *Hypertext Markup Language*
CSS ..... *Cascading Style Sheets*
PHP ..... *PHP Hypertext Preprocessor*
SQL ..... *Structured Query Language*
QoL ..... *Quality of Life*
AdobeXD ..... *User Experience Design Tool*
REST ..... Representational State Transfer

3. Begriffsverzeichnis

Framework: Programmiergerüst, das die Anwendungsarchitektur vorgibt.

Ionic: Open-Source-Webframework zur Erstellung von Hybrid-Apps und Progressive Web Apps auf Basis von HTML5, CSS, Sass und JavaScript/TypeScript.

NodeJS: Plattformübergreifende Open-Source-JavaScript-Laufzeitumgebung, die JavaScript-Code außerhalb eines Webbrowsers ausführen kann. Für Server geeignet.

Socket.IO: JavaScript-Bibliothek für Echtzeit-Webanwendungen. Ermöglicht bidirektionale Echtzeit-Kommunikation zwischen Webclients und Servern.

Debugging: Prozess der Suche und Behebung von Fehlern in Software.

API: Ermöglicht den Austausch von Informationen zwischen verschiedenen Systemen.

4. Einleitung

4.1 Projektumfeld

Das Projektteam besteht aus Lorenz Matthäus, Silas Manns, Manuel Pilz, Jakob Körber, Samuel Schütrumpf, Justin Delle aus der Klasse 12INFO der Modellschule Obersberg.

Rolle / Rollen	**Name**
Projektleiter/Frontend Designer	Manuel Pilz
Frontend Entwickler	Silas Manns
Organisator/Technischer Planer	Lorenz Matthäus
Organisator/Technischer Planer	Justin Delle
Backend Entwickler	Samuel Schütrumpf
Backend Entwickler	Jakob Körber

Auftraggeber ist der Schulleiter der Beruflichen Schulen Obersberg, Herr Lomb.

4.2 Projekt Ziel

Es soll eine App entwickelt werden, die den Nutzern, in diesem Fall Herr Lomb und andere ausgewählte Lehrer, dabei unterstützt, alle nötigen Schritte der Anwesenheitserfassung während eines Feueralarms möglichst intuitiv abzuwickeln.

4.3 Projektbegründung

Da die Schule mehrere, zum Teil weit auseinander liegende, Sammelpunkte hat, müssen zum Informationsaustausch lange Strecken zurückgelegt werden. Durch die App wird Zeit eingespart, da die Informationen in Echtzeit ausgetauscht werden.

Des Weiteren hatte die App aus dem Projekt des letzten Jahres klare Schwächen, welche in Punkt 6.4 beschrieben werden, die mit dieser App korrigiert werden sollten.

4.4 Projektschnittstellen

Die Anwendung arbeitet über die WebUntis-API, welche vom Hersteller Untis, dem Stundenplan-System-Hersteller der Beruflichen Schulen Bad Hersfeld bereitgestellt wurde, um die Daten des Stundenplans zu erhalten. Des Weiteren befindet sich eine MongoDB Datenbank Anbindung im Backend, um die Daten der Feueralarme extern, sicher und cloud-basiert zu speichern.
Nutzer der Anwendung sind einerseits Lehrer, die mit von der Schule bereitgestellten Tablets die Anwesenheit kontrollieren, als auch der Schulleiter, der mit seinem eigenen Tablet oder anderen Geräten die Anwesenheit auswerten und die Informationen ggf. an die zuständigen Einsatzkräfte weiterleiten kann.
Die Kommunikation zwischen Front- und Backend findet mittels Socket.IO statt.

5. Projektplanung

5.1 Projektphasen

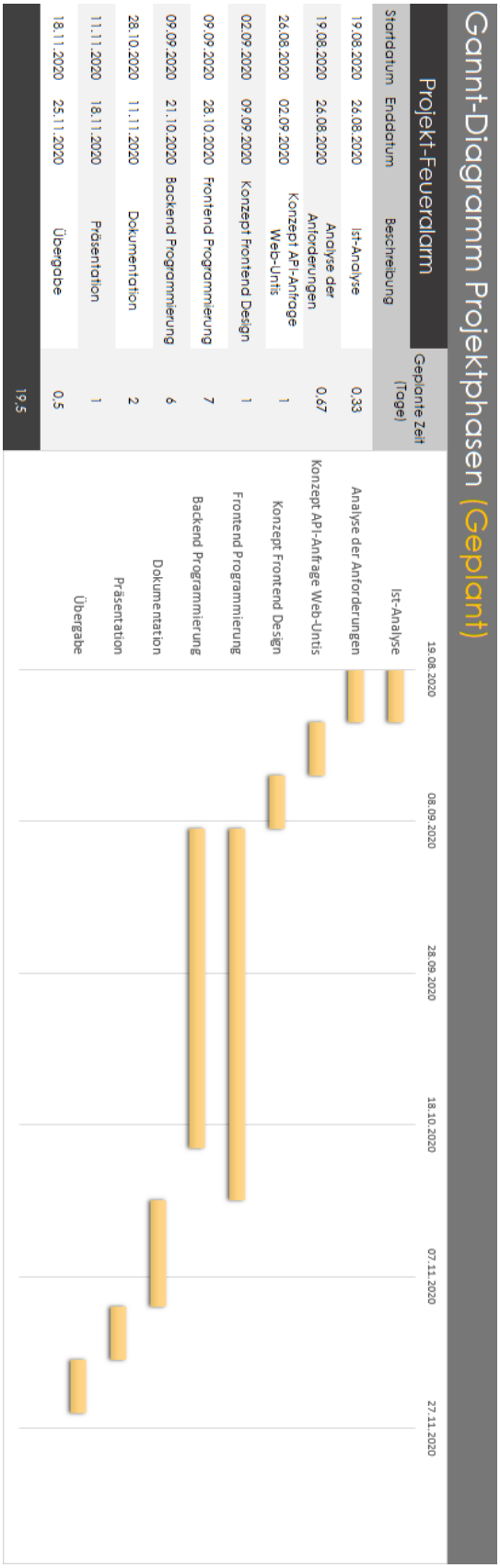
Das Projekt findet immer mittwochs, i.d.R. von der ersten bis zur sechsten Schulstunde, in den Räumlichkeiten der Beruflichen Schulen statt.

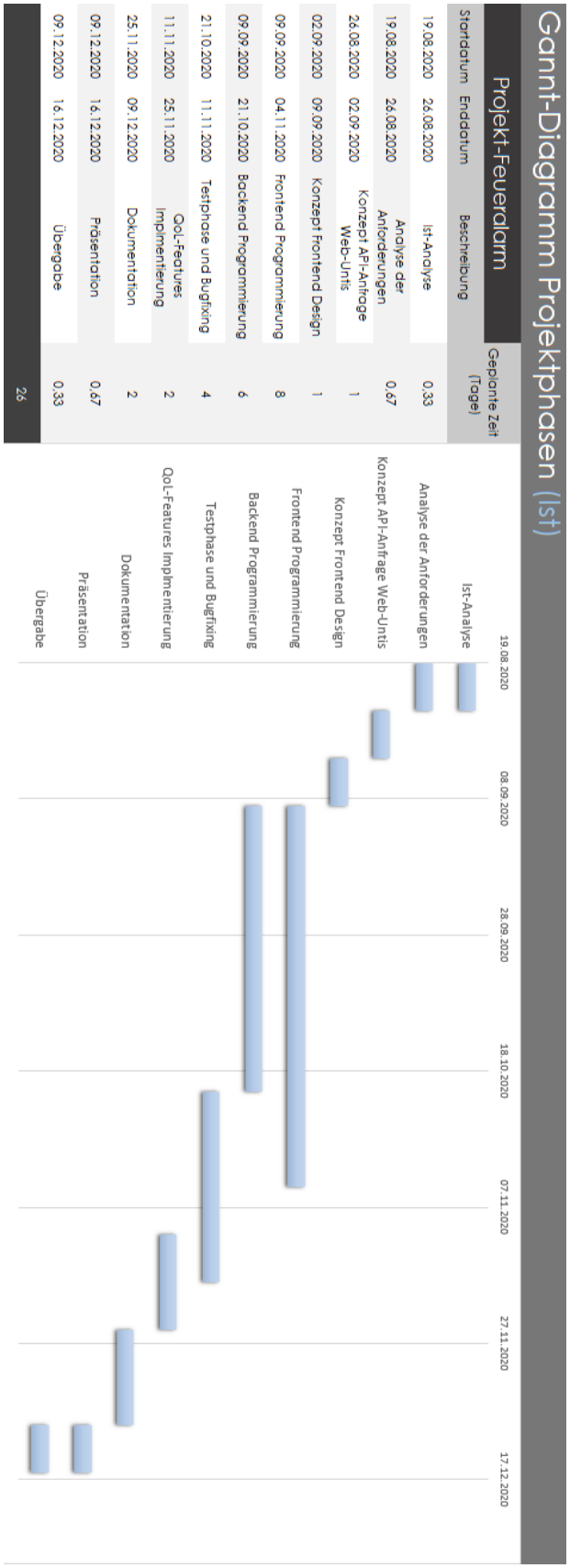
Insgesamt standen 16 Tage zur Verfügung. Die abweichende Anzahl der Tage im Gantt-Diagramm kommt dadurch zustande, da manche Phasen parallel statt kontinuierlich bearbeitet wurden.

Die Abweichungen zwischen dem Ist- und Geplant-Diagramm kommen durch verschiedene Faktoren zustande:

- Testphase wurden bei der Planung nicht miteinberechnet
- Bei der Phasendauer wurde sich verschätzt
- QoL-Features wurden erst während der Implementierung des Frontends bestimmt

Mit QoL-Features sind Funktionen, die dem Nutzer die Bedienung oder das Erlebnis der App erleichtern, wie z.B. Benachrichtigungen, die in 9.5.3 beschrieben werden, gemeint.


Gannt-Diagramm Projektphasen (Geplant)
Projekt-Feueralarm
Startdatum Enddatum Beschreibung Geplante Zeit (Tage)
19.08.2020 26.08.2020 Ist-Analyse 0,33
19.08.2020 26.08.2020 Analyse der Anforderungen 0,67
26.08.2020 02.09.2020 Konzept API-Anfrage Web-Untis 1
02.09.2020 09.09.2020 Konzept Frontend Design 1
09.09.2020 28.10.2020 Frontend Programmierung 7
09.09.2020 21.10.2020 Backend Programmierung 6
28.10.2020 11.11.2020 Dokumentation 2
11.11.2020 18.11.2020 Präsentation 1
18.11.2020 25.11.2020 Übergabe 0,5
19,5
Ist-Analyse
Analyse der Anforderungen
Konzept API-Anfrage Web-Untis
Konzept Frontend Design
Frontend Programmierung
Backend Programmierung
Dokumentation
Präsentation
Übergabe
19.08.2020
08.09.2020
28.09.2020
18.10.2020
07.11.2020
27.11.2020


Gannt-Diagramm Projektphasen (Ist)
Projekt-Feueralarm
Startdatum Enddatum Beschreibung Geplante Zeit (Tage)
19.08.2020 26.08.2020 Ist-Analyse 0,33
19.08.2020 26.08.2020 Analyse der Anforderungen 0,67
26.08.2020 02.09.2020 Konzept API-Anfrage Web-Untis 1
02.09.2020 09.09.2020 Konzept Frontend Design 1
09.09.2020 04.11.2020 Frontend Programmierung 8
09.09.2020 21.10.2020 Backend Programmierung 6
21.10.2020 11.11.2020 Testphase und Bugfixing 4
11.11.2020 25.11.2020 QoL-Features Implmentierung 2
25.11.2020 09.12.2020 Dokumentation 2
09.12.2020 16.12.2020 Präsentation 0,67
09.12.2020 16.12.2020 Übergabe 0,33
26
Ist-Analyse
Analyse der Anforderungen
Konzept API-Anfrage Web-Untis
Konzept Frontend Design
Frontend Programmierung
Backend Programmierung
Testphase und Bugfixing
QoL-Features Implmentierung
Dokumentation
Präsentation
Übergabe
19.08.2020
08.09.2020
28.09.2020
18.10.2020
07.11.2020
27.11.2020
17.12.2020

5.2 Ressourcenplanung

Es werden mehrere Tablets benötigt, die in der Schule gelagert werden, und im Falle eines Feueralarms an die zuständigen Lehrkräfte, die die Kontrolle der Anwesenheit an den verschiedenen Sammelpunkten der Schule durchführen, übergeben werden. Entwickelt wird die App auf den privaten Notebooks der Projektteilnehmer, es wird lediglich das Internet der Schule benutzt. Des Weiteren wird sowohl der Quellcode des Frontend als auch der des Backend im Versionsverwaltungsdienst https://github.com/ stattfinden. Dies ermöglicht eine dynamische, kollaborative und fehlerfreie Entwicklung.

6. Analysephase

6.1 Ist-Analyse

Derzeit gibt es ein Programm zur Anwesenheitskontrolle während eines Feueralarms. Da dieses Programm allerdings vom Auslösen des Feueralarms bis zur Anwesenheitskontrolle viele manuelle Schritte des Nutzers erfordert und die Nutzung bei einem Internetausfall der Schule nicht möglich ist, ist das Programm anfällig für Benutzerfehler und Komplettausfälle. Des Weiteren ist das derzeitige Design der Software nicht gerade ansprechend, übersichtlich, noch modern gehalten. Dies wollen wir durch eine App, die nur noch wenige Schritte benötigt, extern gehostet wird und durch Cloudtechniken ausfallsicher betrieben wird, verbessern.

6.2 Risiko-Analyse

Falls die BSO das Stundenplansystem Untis wechselt, und das neue System keine REST-API besitzt, ist dieses Projekt ohne Wert. Falls das neue System eine API besitzt, muss der Backend-Server auf diese neue API angepasst werden, was mit mittlerem Zeitaufwand verbunden ist.

Das alte Projekt ist auch von Untis abhängig, als auch vom Schulnetz.

6.3 Wirtschaftlichkeitsanalyse

Das Projekt ist für die Schule in der Hinsicht lohnenswert, dass durch die Echzeit-Synchronisation Laufwege und dadurch auch Zeit während eines Feueralarms gespart wird. Außerdem wird eine Ausfallsicherheit erzeugt, wodurch die App nicht mehr abhängig vom internen Schulnetz ist. Ein finanzieller Profit ist nicht vorhanden.

6.4 Projektkosten

Zu den bestehenden Kosten des Webspace-Servers, kommen zusätzliche Anschaffungskosten für die von der Schule bereit zu stellenden Tablets, die nur für den Einsatz dieser Software vorgesehen sind. Andere Kosten entstanden bei diesem Projekt nicht.

Kosten:
3x Terra Tablet à ca. 200€ entspricht 600€

Gesamtkosten des Projekts = 600€

6.5 Nutzwertanalyse

		Altes Projekt		Neues Projekt	
	Gewichtungs-faktoren	Bewertung	Teilnutzwert	Bewertung	Teilnutzwert
Design	0,2	4	0,8	9	1,8
Bedienbarkeit	0,4	3	1,2	9	3,6
Risiko	0,2	5	1	8	1,6
Erweiterbarkeit	0,2	5	1	9	1,8
Summe		17	4	35	8,4

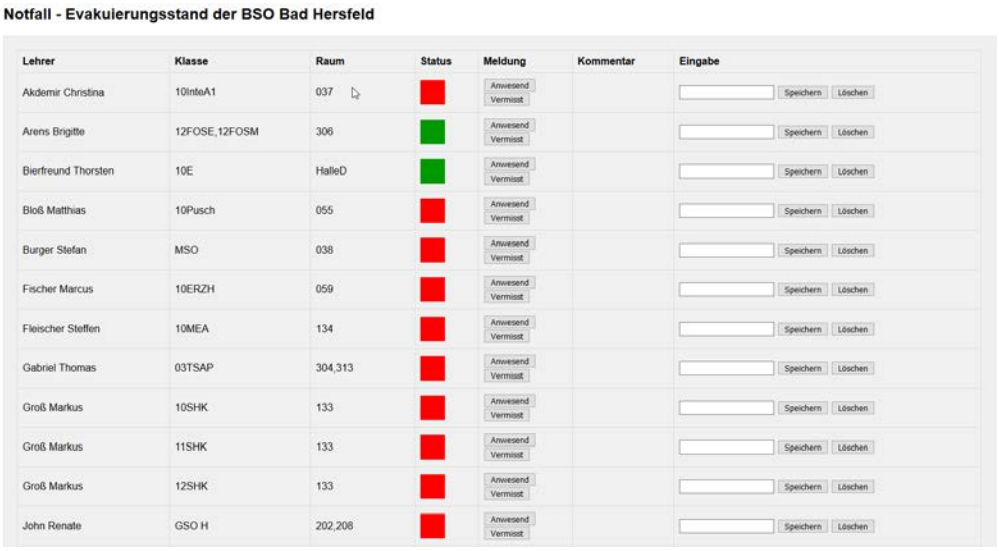
Beim in der Tabelle genannten Risiko bedeutet ein hoher Wert ein niedriges Risiko. Der Punkt Erweiterbarkeit ist auf die Weiterentwicklungsmöglichkeiten des Projekts bezogen. Weiteres zu diesem Punkt ist in 10.0 zu finden.

Ein deutlicher Unterschied ist zwischen den beiden Ergebnissen zu sehen, was klar für das neue Projekt spricht.

Die Punkte, aufgrund dessen diese Entscheidung getroffen wurde, sind noch einmal hier darunter gelistet.

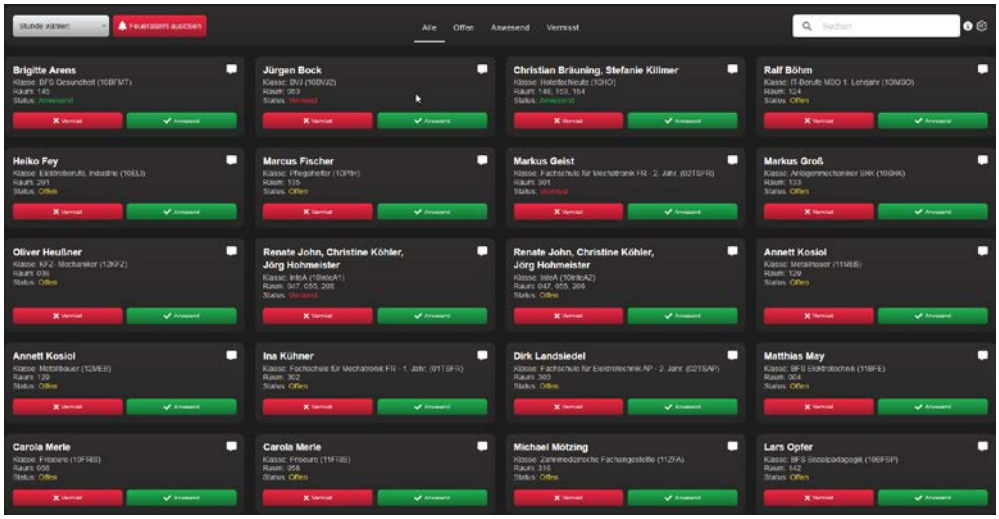
Altes Projekt:

- An das Schulnetz gebunden
- Unübersichtliches und unmodernes Design
- Viele manuelle Eingriffe nötig
- Hat seinen Zweck erfüllt



Neues Projekt:

- Übersichtlich und strukturiert
- Ansprechendes und modernes Design
- Bessere Bedienbarkeit während einer Stresssituation
- Nicht an das Schulnetz gebunden
- Überall vom Internet aus mit Authentifizierung erreichbar



6.6 Qualitätsanforderungen

Die Anwendung sollte möglichst schnelle Ladezeiten haben, intuitiv bedienbar sein und eine gewisse Ausfallsicherheit gewährleisten. Des Weiteren sollte das Design ansprechender als das der letzten Version sein.

7. Entwurfsphase

7.1 Zielplattform

Die App kann über jedem beliebigen Webbrowser, wobei der Internet Explorer bei der Darstellung einiger GUI-Teile Probleme haben könnte, verwendet werden. Google Chrome ist eine Empfehlung. Des Weiteren wurde auf den Tablet-Endgeräten eine native Applikation (.apk-Datei) vorkonfiguriert. Die .apk-Datei kann von jedem beliebigen Android-Gerät unter https://alarm.bso-hef.de/app heruntergeladen werden. Ein Zugriff auf die normale Webseite https://alarm.bso-hef.de ist von mobilen Endgeräten ebenfalls möglich.

7.2 Entwurf der Benutzeroberfläche

Ein erster Entwurf der GUI wurde mit AdobeXD gemacht, einer vektorbasierten Grafiksoftware zum Entwurf von grafischen Benutzeroberflächen für Web-Apps und Mobile Apps.

Aufgrund der Anforderungen an die App, fiel die Entscheidung schnell auf Ionic, ein Framework für Web-Apps, dass des Weiteren auch eine einfache Möglichkeit zur Transferierung in eine .apk-Datei ermöglicht. Ebenfalls sieht Ionic sehr modern aus und ist unter der MIT-Lizenz lizensiert, welche eine (für unsere Zwecke) uneingeschränkte Verwendung ermöglicht.

8. Backend

8.1 Genutzte Module

- Node.JS
- Express
- axsios
- Mongoose
- JsonWebToken
- SocketIO

8.2 Funktion

Das Backend ist zuständig für die Authentifizierung des Frontend Nutzers.

Hat sich ein Admin-Nutzer authentifiziert, so kann dieser einen Feueralarm auslösen. Das hat zur Folge, dass das Backend benötigte Stundenplan-Daten der WebUntis-API bezieht, diese verarbeitet und zwischenspeichert.

Darüber hinaus besitzt das Backend die Funktion, jene Daten an die Clients weiterzugeben und Änderungsanfragen entgegenzunehmen. Änderungen werden an alle anderen Clients in Echtzeit übermittelt und aktualisiert.

8.3 Programmierung

8.3.1 Authentifizierung

Die Authentifizierung verläuft in erster Linie über eine REST-API. Der Client übermittelt per Request seinen Nutzernamen und sein Passwort, welche vom Backend entgegengenommen werden. Ist der Nutzer in der Datenbank vorhanden, dann erhält der Client einen JsonWebToken der für 30 Minuten gültig ist.

Dieser Token ist notwendig um daraufhin eine Socket-Verbindung mit dem Server aufzubauen, über die alle anwesenheitsrelevanten Daten transferiert werden.

Mithilfe des Tokens kann zudem die Berechtigungsstufe des Nutzers ermittelt werden, was für die Auslösung des Alarms nötig ist.

8.3.2 Framework

Die gesamte Backendstruktur basiert auf NodeJS und ist in JavaScript geschrieben. Die Anwendung ist auf Heroku (EC2/AWS) gehostet.

Das Routing der REST-Schnittstelle ist mit ExpressJS umgesetzt. Die Etablierung und Bedienung der Socket-Verbindungen ist mit Socketio implementiert. Die Anbindung an die WebUntis-Schnittstelle ist mit axsios realisiert. Die Verwendung der MongoDB Datenbank erfolgt über Mongoose.

8.3.3 Datenbank

Gehostet wird die Datenbank über MongoDB Atlas in der ‚free tier'-Variante in der Azure-Cloud. Zur Verfügung stehen hierbei 512MB Speicherkapazität.

9. Frontend

9.1 Genutzte Module

- Ionic (Angular)
- Angular-Router
- HTTPClient
- Socket.IO-Client

9.2 Programmierung

Der Quellcode der Hauptseite befindet hauptsächlich in den Dateien home.page.ts, home.page.scss und home.page.html im Verzeichnis „Frontend/src/app/home".

Der Quellcode der Loginseite befindet hauptsächlich in den Dateien login.page.ts, login.page.scss und login.page.html im Verzeichnis „Frontend/src/app/login".

Der Quellcode der Modals, Fenster die sich über dem Inhalt öffnen und erst geschlossen werden müssen, bevor die App weiter fortgesetzt werden kann, befindet sich unter „Frontend/src/app/_modals".

Die Services befinden sich unter „Frontend/src/app/_services“. In diesem Verzeichnis befindet sich auch die Datei socket.service.ts, die die Kommunikation mit dem Backend steuert.

Die Daten können jederzeit aus dem Git-Repository https://ww.github.com/manuelvongivenchy/projekt-feuerlarm,wie In 10.1 beschrieben, abgerufen und weiterentwickelt werden.

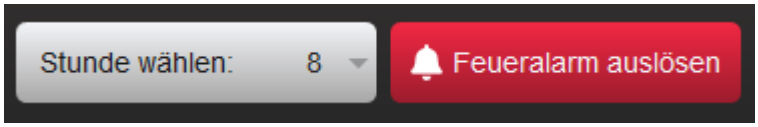
9.3 Authentifizierung

Alle Anfragen an die Seite werden auf /login weitergeleitet. Falls ein Benutzer bereits authentifiziert ist, wird man auf die Hauptseite weitergeleitet. Falls nicht wird man auf die Login Seite geleitet.

Bei dieser Seite müssen bekannte Anmeldedaten angegeben werden, um sich authentifizieren zu können.

Folgende Nutzer sind verfügbar:

- Admin, Benutzer des Schulleiters
- User, Benutzer der gekauften Tablets
- Dev, Entwickler-Benutzer



User sehen diese beiden Buttons nicht, da die Auslösung des Feueralarms nur für den Schulleiter (und für die Entwickler zu debug-Zwecken) vorgesehen ist.

9.4 Funktionen

Das Frontend dient in erster Linie der Darstellung der Anwesenheitskontrolle.

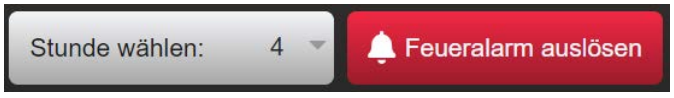
Der Nutzer erhält hier eine Übersicht über alle Berufsschulklassen, die sich während der Auslösung des Alarms im Gebäude befinden. Diese sind jeweils mit einem Anwesenheits-Status und einen Kommentar versehen. Beide sind durch den Benutzer anpassbar.

Es gibt 3 verschiedene Anwesenheits-Status:

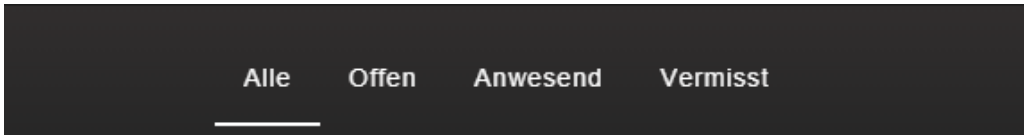
1. Unvollständig
2. Offen
3. Anwesend

Ebenso verfügt das Frontend über eine Maske zur Abfrage von Benutzername und Passwort. Ohne Authentifizierung vom Backend-Server bekommt der Nutzer keinen Zugang zur Anwesenheitskontrolle. Zudem ist nur ein Nutzer mit Admin-Account dazu berechtigt, über die Weboberfläche einen Alarm auszulösen, um Dopplungen und Fehler zu verhindern.

Um die aktuellen Daten von Untis zu bekommen, muss der Nutzer den Feueralarm-Button betätigen. Die Auswahl links neben dem Button wird bei Betätigung mitgeschickt. Falls nichts ausgewählt wurde, weil der Button zum Beispiel außerhalb der Schulzeit gedrückt wurde, kommt eine Fehlermeldung zurück.

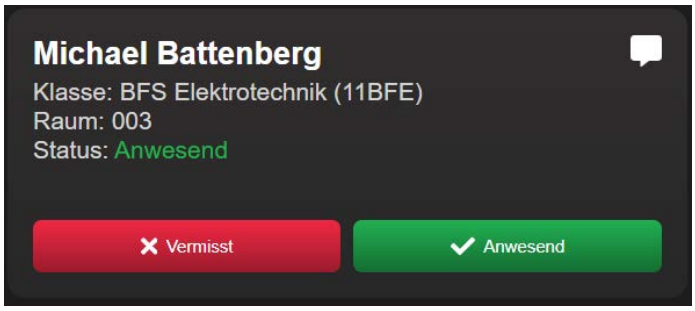


Um die Anwesenheit besser Überblicken zu können, kann man die Klassen nach ihrem Status filtern.

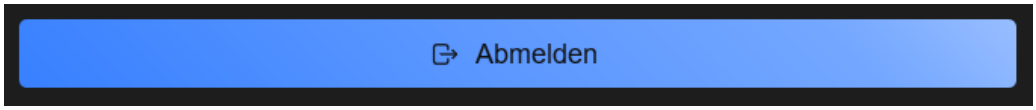


Über die Suche kann man gezielt bestimmte Klassen oder Lehrer finden.

In den einzelnen Klassen-Objekten kann man mit dem roten und grünen Button die Anwesenheit setzen und mit der Sprechblase rechts-oben ein Kommentar setzen, welches als genauere Beschreibung im Falle einer unvollständigen Klasse gedacht ist wie z.B. „Schüler xy fehlt, sonst vollständig“. Dieser kann auch wieder entfernt werden kann.



Wenn man auf das Information-Icon rechts neben der Suchleiste klickt, werden verschiedene Informationen aufgelistet.
Darunter das Datum als auch die Uhrzeit, an dem der aktuelle Feueralarm ausgelöst wurde, die Rolle des Nutzers, mit dem man gerade angemeldet ist und die aktuelle Version der App.
Weiter unten befindet sich auch noch ein Abmelde-Button, mit dem man wieder zur Login Seite zurückgelangt.



9.5 QoL-Features

Über das Einstellungs-Icon rechts neben der Suchleiste kann man auf die verschiedenen Funktionen zugreifen.

9.5.1 Sortierung

Bei der Sortierung gibt es zwei Möglichkeiten; Klassen und Lehrer. Bei beidem wird alphabetisch sortiert. Von dieser Option sind die Klassen- bzw. Lehrerobjekte betroffen, als auch Benachrichtigungen.

Die Sortierung wird nur lokal am Klienten geändert, andere Nutzer werden von einer Änderung dieses Wertes nicht beeinträchtigt.

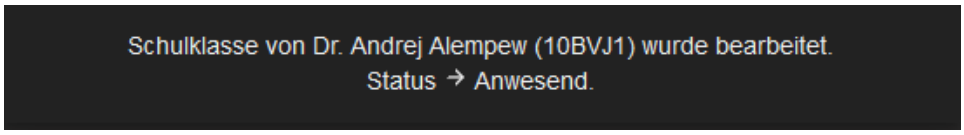
9.5.2 Archivierten Feueralarm anschauen

Alte Feueralarme werden im Backend archiviert sobald ein neuer Feueralarm ausgelöst wurde, trotz dessen können sie, mit Änderung der Einstellung „Archivierten Feueralarm auswählen“, nochmal begutachtet werden.
Die Daten können nichtmehr verändert werden. Bis zu 20 Feueralarme werden gespeichert, danach wird der älteste permanent gelöscht.

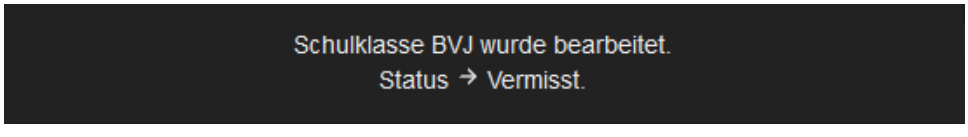
Diese Einstellung wird nur lokal am Klienten geändert, andere Nutzer werden von einer Änderung dieses Wertes nicht beeinträchtigt. Bei Neuladen der App/Seite wird diese Einstellung zurückgesetzt.

9.5.3 Benachrichtigungen

Benachrichtigungen werden an andere Klienten, die die App oder Webseite geöffnet und sich erfolgreich authentifiziert haben, gesendet. Sie sind folgendermaßen aufgebaut:



Sortieren nach Lehrern ausgewählt.



Sortieren nach Klassen ausgewählt.

Der erste Satz beschreibt an welcher Klasse oder Lehrer, je nachdem welche Sortierung ausgewählt ist, eine Änderung vorgenommen wurde.

Der zweite Satz beschreibt was für eine Änderung vorgenommen wurde. Die Optionen hierbei sind:

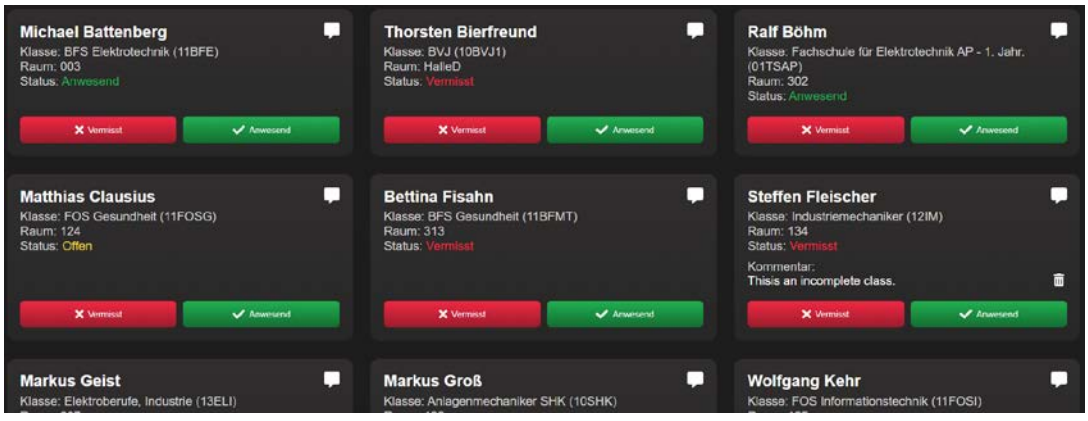
- Kommentar auf „xy“ geändert
- Kommentar gelöscht
- Status → Anwesend
- Status → Unvollständig

9.6 Design

Das Design ist im modernen Dark-Mode-Design gehalten. Ein Light-Mode ist nicht verfügbar.

Der Aufbau ist simpel und effektiv in zwei Abschnitte, den Header und den Body, aufgeteilt.
Im Header sind administrative oder unterstützende Funktionen wie z.B. Filterung und Feueralarm auslösen als auch Informationen und Einstellung wie z.B. Benachrichtigungen aktivieren platziert.

Im Body sind die Daten der Klassen, die von der Web-Untis-API kommen, gelistet.



10.0 API und Weiterentwicklung des Projekts

10.1 Allgemein

Alle Daten des Frontends, Backends und der Doku sind im Git-Repository, https://ww.github.com/manuelvongivenchy/projekt-feuerlarm, verfügbar. Das Repository ist privat, das heißt Zugriff darauf muss erst genehmigt werden, um es vor Veränderungen dritter zu schützen. In diesem Fall bitte eine E-Mail an manuelpilz473@gmail.com mit dem Betreff „Zugriff auf Projekt-Feueralarm Repository“.

10.2 Frontend Weiterentwicklung

Für die Weiterentwicklung des Frontends ist folgender Nutzer vorgesehen:
Benutzer: dev
Passwort: develop123!
Um sich am Backend zu authentifizieren, um mit diesem kommunizieren zu können, muss man als erstes die signin-Funktion in der rest.service.ts im unter 8.3 beschriebenen Verzeichnis der Sevices ausführen, um einen Token zu erhalten. Danach muss ein Socket-Client erstellt werden und ein „authenticate“ mit dem erhaltenen Token an das Socket-Backend gesendet werden, um mit diesem kommunizieren zu können.
Alle nötigen Aktionen für diesen Schritt um sich am Backend zu authentifizieren sind in der socket.service.ts zu finden.

Wenn beide Schritte erledigt sein kann man über verschiedene emit-Funktionen die im Frontend angezeigten Daten für alle Nutzer verändern.

10.3 Backend Weiterentwicklung

Der gesamte Backend-Quelltext ist unter „Projekt-Feueralarm/Backend/nodeJs-Backend“ ersichtlich.
Über server.js wird der Server gestartet. Dabei wird die express-Schnittstelle in app.js hochgezogen, die mittels des Routings in „./routes/users.js“ REST-Authentifizierungsanfragen an den controller in „./controllers/users.js“ weiterleitet und beantwortet.
Ebenso wird hier die Verbindung zum MongoDB-Server aufgebaut.

Zusätzlich beginnt der Server, auf Socketanfragen mittels Socketio zu horchen. Die Logik dazu befindet sich in „./routes/sockets.js“.
Die Logik zur Erstellung und Verwaltung von Feueralarmen steckt in „./controllers/posts.js“ und wird von der Socketebene angesteuert.
Der Postcontroller muss zur Erzeugung eines neuen Feueralarms mit der REST-Schnittstelle von WebUntis kommunizieren. Die Anfragen dazu finden sich unter „./untis/requests.js“.
Diese Anfragen werden über mehrere Threads (sog. Worker threads) parallel abgesendet und verarbeitet. Der dazugehörige Quelltext befindet sich in „./untis/timeTableWorker.js“. Hier müsste man im Zweifelsfall bei Änderungen zuerst ansetzen.
Deployen lässt sich das Backend am besten über die heroku-CLI. Dafür bietet sich die Nutzung von „git bash“ an.
Zunächst meldet man sich mit „heroku login“ an.
Daraufhin wechselt man auf das richtige Server-Verzeichnis mit „heroku git:remote -a feueralarm“.
Anschließend kann man seine Änderungen in den lokalen git-Container durch folgenden Befehl übernehmen: „git add .“.
Über „git commit -m „“ wird ein neuer Versionsstand erzeugt.
Letztendlich deployed wird das Backend dann mit „git push heroku master“.

11. Lizenz

Die Software steht unter der MIT-Lizenz.

“MIT License

Copyright (c) 2020 Manuel Pilz, Silas Manns, Lorenz Matthäus, Jakob Körber, Samuel Schütrumpf, Justin Delle

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.”

Eine deutsche Fassung der Lizenz ist unter https://de.wikipedia.org/wiki/MIT-Lizenz zu finden.

12. Eidesstattliche Erklärung

Wir, Die Autoren, versichere hiermit, dass wir unsere Dokumentation zur schulischen Projektarbeit mit dem Thema
Feueralarm BSO – Anwesenheitskontrolle während eines Feueralarms an den Beruflichen Schulen Obersberg
selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben, wobei wir alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet haben.

Bad Hersfeld, den 17.12.2020

_____
Manuel Pilz (Projektleiter)

_____
Silas Manns

_____
Lorenz Matthäus

_____
Samuel Schütrumpf

_____
Jakob Körber

_____
Justin Delle

12. Quellenverzeichnis

- Versionsverwaltung https://github.com/
- Backend-Hosting https://heroku.com/
- Frontend-Hosting https://ionos.de/
- Datenbank-Hosting https://mongodb.org/
- NodeJS-Framework https://nodejs.org/
- Ionic-Framework https://ionicframework.com/
- Google Documents https://docs.google.com/
- WebUntis https://webuntis.com/
- Wortmann AG (Tablet-Hersteller) https://wortmann.de/
- MIT-Lizenz https://de.wikipedia.org/wiki/MIT-Lizenz

13. Fazit

Abschließend zu sagen ist, dass wir nun eine Produktivumgebung, für die Kontrolleure und den Schulleiter, zur Überprüfung der Anwesenheit, der Klassen und deren Schüler sowie der Lehrer geschaffen haben. Diese Daten dienen dann den zuständigen Einsatzkräften, um vermisste Personen oder ganze Klassen besser auffinden und ggf. Rettungsmaßnahmen einleiten zu können.