Staatliche Berufsschule Lauingen

Friedrich-Ebert-Straße 14, 89415 Lauingen a. d. Donau

**BSL-Maps**

von

Marc Rettinger, David Kramer, Paul Bischoff

EIT12A

28. Februar 2025

Inhaltsverzeichnis:

**David = Farbe**

**Marc = Farbe**

**Paul = Farbe**

1. Projektbeschreibung

1.1. Ist-Zustand:

An die Berufsschule Lauingen kommen neben Schüler und Studenten, auch Besucher von außerhalb, unter anderem auch aus Schulen aus anderen Ländern, mit denen die Berufsschule kooperiert. Es ist dabei auffällig, dass diese Gruppe Probleme damit hat, sich im Gebäude zurechtzufinden. Entsprechend kommt es dabei dazu, dass die Personen sich im Schulhaus schnell verlaufen können und auch das Risiko besteht, dass sie eventuell bei Terminen erst verspätet eintreffen, da sie viel Zeit für die Orientierung innerhalb der Einrichtung benötigen.

1.2. Soll-Zustand:

Damit die Orientierungsschwierigkeiten bei Besuchern minimiert wird, soll eine über das Internet aufrufbare Webanwendung entwickelt werden. Zum einen soll es möglich sein, eine Indoor-Navigation vom Schulgebäude aufrufen zu können. Mit dieser kann die Karte der Berufsschule angezeigt werden, sowie auch eine Route zu einem gewünschten Raum ermittelt werden. Dadurch werden dann Besucher in der Lage sein, schneller und zielsicherer zu Ihrem gewünschten Ziel sich zu begeben.

Es soll zudem noch ein Administratives Panel für die Pflege der angezeigten Inhalte zu den jeweiligen Räumen geben. Dadurch wird das Administrieren der angezeigten Daten komfortabel und schnell erfolgen, da eine Datenänderung mittels einer Oberfläche und nicht über die Datenquelle direkt erfolgt.

2. Funktionsumfang:

2.1 Muss-Kriterien:

Adminstratorfunktionen:

Es soll ein Adminpanel entwickelt werden, indem der Administrator die Informationen für die Indoor-Navigation verwalten kann. Das Adminpanel soll als Webseite aufrufbar sein. Dieses soll jedoch nicht von Webcrawlern und Suchmaschinen indexiert werden. Es gibt nur einen Admin. Es soll keine Möglichkeit geben sich als Nutzer dort zu registrieren. Der Administrator-Nutzer soll bereits angelegt worden sein, sodass sich dieser nur einloggen kann. Bevor dies möglich ist müssen die Cookies akzeptiert werden, da diese unter anderem für die Nutzerauthentifizierung benötigt werden. Anschließend kann sich der Admin auf der Seite einloggen. Sobald der Admin eingeloggt ist, kann sich dieser über das Logout-Icon in der Navbar wieder ausloggen. Es wird eine Stockwerkwechsel Funktion geben. Außerdem soll es über der Kartenansicht eine Suchleiste geben, in der bestimmten Klassenräumen gesucht werden kann. Es soll möglich sein auf der Karte des Schulgebäudes einen Raum anzuklicken oder nach einem Raum zu suchen. Damit werden die Daten zu dem jeweiligen Raum darunter in Eingabefeldern und in Tabellen angezeigt. Zu diesen Daten gehören der Raumname, die Beschreibung und der Raumtyp. Diese Daten sollen sich ändern lassen. Wenn für diesen Raum Klassen oder Lehrer zugewiesen sind, werden diese in einer Tabelle angezeigt. Wenn der Raumtyp z. B. von einem Klassenraum zu einem Büro geändert wird soll eine Warnung erscheinen, dass die Zuweisungen gelöscht werden. Dies gilt auch wenn ein Büro zu einem Klassenraum geändert wird. Auf der Klassen Seite gibt es eine Übersicht aller angelegten Klassen. Wird eine Klasse neu angelegt oder wenn auf eine bestehende geklickt wird können die Daten wie Klassenname, Klassenleitung oder Raum gepflegt werden. Außerdem gibt es eine Lehrer Übersichtsseite auf der eine Tabelle mit allen angelegten Lehrern angezeigt wird. Diese können ebenfalls bearbeitet werden und neue Lehrer können auch angelegt werden. Daten die dort gepflegt werden können sind der Lehrername und das Büro dem der Lehrer zugewiesen ist.

Benutzerfunktionen:

Als Inspiration für die Indoor-Navigation dienten gängige Anwendungen wie Google Maps, Apple Karten und Baidu Maps. Insbesondere die Indoor-Navigation dieser, da unter anderem in Flughäfen auch eine Navigation dieser Art möglich ist. Dem Endnutzer soll es möglich sein in einer Webanwendung eine Übersicht des Schulgebäudes zu erhalten. Es soll die selbe Ansicht der Karte wie im Adminpanel sein. Ebenfalls soll die Suchleiste des Adminpanels genutzt werden, mit eventuell leicht angepasstem Design, um eine einfachere Ansicht für fremdsprachige Nutzer zu gewährleisten. Wenn in der Suchleiste nach einem bestimmten Raum gesucht wird, so wird dieser anschließend auf der Karte markiert. Standardmäßig kann der Nutzer ähnlich wie im Adminpanel auf Raumbezeichnungen klicken, sodass dieser dann ausgewählt wird. Der Nutzer hat des Weiteren die Möglichkeit Routen zu planen. Die Routenplanfunktion soll unterhalb der Karte erscheinen. Diese besteht aus zwei Eingabefeldern die wie eine Suchleiste Suchvorschläge zeigt, wenn etwas eingeben wird. Standardmäßig soll außerdem unterhalb der Karte ein Schnellauswahlmenü erscheinen. Dort werden wichtige Räume wie der Ausgang, Toiletten, etc. angegeben. Wenn auf einen der Räume in diesem Schnellauswahlmenü geklickt wird, so wird auf der Karte dieser Raum ausgewählt.

Qualitätsanforderungen:

Besonders hohen Stellenwert hat für uns hohe Ladegeschwindigkeiten, responsives Design und Barrierefreiheit, sodass für jeden, egal ob Endnutzer oder Administrator die Anwendung angenehm zu benutzen ist. Das heißt es soll eine gute Struktur in der PC-Ansicht geben, sowie in der mobilen Ansicht. Außerdem sollen sich die Ladezeiten so gering wie möglich gehalten werden.

2.2 Wunsch-Kriterien:

Ein Wunsch-Kriterium ist eine AR-Navigation wie es von Google Maps bekannt ist. Dies soll in einer Handy App eingebaut sein. Wenn die Navigation gestartet wird. Es soll mit Hilfe der Kamera die Umgebung sichtbar sein und mit AR soll die Route angezeigt werden. Des Weiteren ist noch eine Homepage Wünschenswert, in der zur Navigationsseite weitergeleitet werden kann und auf der noch zusätzliche Infos enthalten sind.

2.3 Abgrenzungskriterien:

Aufgrund von zeitlichen Einschränkungen konnten wir nicht die Wunschkriterien außer der Homepage umsetzen. Da die Umsetzung dieser Funktionen die zeitlichen Vorgaben bei weitem übertroffen hätten. Dies liegt nicht nur an der Umsetzung selbst, sondern auch an den technischen Fähigkeiten die wir uns in

diesem kurzen Zeitraum beibringen hätten müssen. Dies ist uns nach einer kurzen Testphase klar geworden, in der wir getestet haben wie schwer die Wunschfunktionen umzusetzen sind.

4. Produkteinsatz:

Anwendungsbereich:

Die Homepage soll von überall aufrufbar sein und auch in Suchergebnissen erscheinen. Jedoch ist der primäre Anwendungsbereich im Schulgebäude der Berufsschule Lauingen. Die Administrationsseite ist zwar auch von überall aus aufrufbar, jedoch erscheint diese nicht in den Suchergebnissen von Suchmaschinen.

Zielgruppe:

Die Zielgruppe für die Anwendung ist ziemlich unterschiedlich. Dies können unter anderem Mitarbeiter der Schule wie z. B. Lehrer sein, wenn diese für eine Vertretungsstunde ein Klassenzimmer suchen müssen. Aber auch Schüler, wenn diese in einem neuen Schuljahr in einem anderen Klassenzimmer befinden, welches ihnen noch nicht bekannt ist. Aber auch wenn Dritte wie Besucher anderer Bildungseinrichtungen einen Termin in der Schule haben ist die Anwendung ebenfalls nützlich, somit Verirrungen im Schulgebäude vermieden werden können. Insgesamt kann unsere Zielgruppe somit Verspätungen vermeiden.

3. Planung des Projekts:

In dieser Phase des Projekts wurden die entsprechend Ressourcen ermittelt, welche für das Projekt benötigt werden, sowie der zeitliche Ablauf des Projektes geregelt.

3.1. Ressourcen

Zur Entwicklung des Projektes sowie zum lokalen Testen, werden die Laptops der jeweiligen Entwickler hergenommen. Entsprechend läuft es auf drei Rechner hinaus, wobei zwei auf dem Betriebssystem MacOS und eins auf Windows 11 läuft. Die Endanwendungen wird über den Webhosting-Dienst Ionos gehostet worauf zum einen die Webanwendungen als auch eine Microsoft-SQL (MSSQL) Datenbank zur Abspeicherung von Daten läuft. Zum Testen der Mobilfähigkeit der Webanwendungen werden auch zudem Smartphones benötigt.

Bei der Auswahl der Software mussten die Betriebssysteme der jeweiligen Entwicklungsrechner beachtet werden, da einige der ausgewählten Softwares zwar für Windows aber nicht für MacOS verfügbar waren. Grundsätzlich wird auf allen Entwicklungsgeräten eine Entwicklungsumgebung für das Schreiben von Quellcode, sowie eine Datenbank und ein Datenbankmanagementsystem zum Testen der Datenbankanbindung genutzt. Die Unterschiede der verwendeten Softwares werden durch folgende Tabelle dargestellt:

|  |  |  |
| --- | --- | --- |
|  | Windows | Mac-OS |
| Entwicklungsumgebung | Visual Studio 2022 | Visual Studio Code |
| Datenbank | Microsoft SQL Developer Edition | SQLite |
| Datenbankmanagement-system | Microsoft SQL Server Management Studio | DBeaver |

Des Weiteren wird für die Gestaltung der Karte, welche der Endbenutzer sieht, Photoshop verwendet. Für das Design der Weboberfläche wird Adobe XD verwendet. Für die Erstellung von Anwendungsfalldiagrammen wird Umletino zur Erstellung von UML konformen Diagrammen verwendet. Zur Definierung der Datenbankstruktur wird Lucidchart eingesetzt.

Zur Organisation des zeitlichen Ablaufes des Projektes, sowie die Einteilung der verschiedenen Aufgaben des Projekts, wird das Tool Redmine verwendet. Dieses wird von der Berufsschule Lauingen bereitgestellt. Für die Versionskontrolle des Quellcodes wird GitHub eingesetzt.

3.2. Verwendete Technologien

Beide Webanwendungen sollen als Blazor Server App entwickelt werden. Hierbei handelt es sich um eine Abweichung von der originellen Planung im Projektantrag, die Nutzeranwendung mit React Native und die Administrations Oberfläche mit React zu erstellen. Gründe hierfür waren unter anderem die fehlende Zeit, welche zum Erlernen beider Frameworks benötigt wäre. Zudem würde die zuvor angestrebte Entwicklung einer Handyanwendung zu hohen Mehrkosten führen, weswegen diese Idee verworfen wurde und anstelle dessen eine Webanwendung entwickelt wird. Das Backend hingegen soll als C# Klassenbibliothek entwickelt werden und für die beiden Webanwendungen bereitgestellt werden. Alle drei Komponenten basieren auf dem .Net 8 Framework von Microsoft.

3.2. Zeitplanung

Die Aufteilung der jeweiligen Prozessphasen sowie der Zeiteneinteilung kann folgender Tabelle entnommen werden. Die Zeiteinplanung ist dabei für jeden der drei Entwickler gedacht:

|  |  |  |
| --- | --- | --- |
| Projektphase | Anfallende Aufgaben | Schätz-ungen |
| AnAlyse | Machbarkeitsanalyse  Einarbeitung in die Doku des Ionos Servers | 7 |
| Definition | Definierung des Aufbaus der Datenbank  Erstellung von Mockups der Webseite | 7 |
| Umsetzung | Entwicklung der Backend Klassenbibliothek  Entwicklung der Administrationsoberfläche  Entwicklung der Benutzeroberfläche  Implementierung der entwickelten Webanwendungen in den Ionos Server | 30 |
| Testen | UNIT-Tests  Schnittstellentests  Integrationstests | 5 |
| Dokumentation |  | 5 |

Der Zeitliche Ablauf kann wie folgt als Gantt Diagramm per Feature (Nummer im Anhang) und per User Story (Nummer im Anhang) dargestellt werden.

**2. Kostenplanung**

**2.1 Entwicklungskosten**

Da der Stundenlohn sich auf 6€. Das Team besteht aus drei Entwicklern und jeder hat circa 60 Stunden daran gearbeitet. Deshalb belaufen sich die insgesamten Kosten auf 360€.

**2.2 Softwarekosten**

Um eine bestmögliche Nutzererfahrung zu bieten, lagen wir großen Wert auf ansprechendes Design. Dadurch waren zwei Arten von Software für das gestalten der Anwendung erforderlich. Zum einen Adobe XD um Mockups für die Anwendung zu designen. Zum anderen mussten Karten mit gutem Design vorhanden sein. Dafür nutzten wir Photoshop.

- **Design-Software**: 20 € pro Monat (Kosten für Photoshop und Adobe XD)

**2.3 Serverkosten**

Die Webseiten unserer Indoor-Navigation benötigt einen Server für den Betrieb und die Bereitstellung der Anwendungen. Hier fallen folgende Kosten an:

- **Hosting & Server**: 15 € pro Monat (inklusive Domain von Ionos)

**2.4 Gesamtkosten**

* **Einmalige Personalkosten:** 600 €
* **Monatliche Server- und Softwarekosten**: 35 €
* **Jährliche Kosten**: 420 €

**3. Nutzen der Anwendung:**

**3.1 Zeitersparnis**  
Die Anwendung ermöglicht es Schülern, Lehrern und Besuchern, schneller den gewünschten Raum zu finden. Dadurch werden Verspätungen reduziert und der Unterricht bzw. die Termine, aber auch Veranstaltungen können pünktlicher beginnen. Besonders neue Schüler oder Vertretungslehrer profitieren davon, aber auch externe Gäste von beispielsweise anderen Bildungseinrichtungen.

**3.2 Verbesserte Orientierung**  
Die App zeigt den schnellsten Weg zum Ziel und verringert dadurch lange Umwege.

**3.3 Förderung der Schulkultur**  
Weniger Störungen durch verspätetes Eintreffen im Unterricht tragen zu einem besseren Schulklima bei. Die Anwendung hilft, eine organisierte und pünktliche Schulgemeinschaft zu fördern.

**3.4 Verbesserung des Schulimages**  
Eine moderne digitale Lösung steigert die Attraktivität der Schule für neue Schüler und Lehrkräfte, aber auch das ansehen anderer Schulen sodass sich diese an der Berufsschule Lauingen orientieren. Auch Besucher erhalten einen positiven Eindruck von der digitalen Ausstattung und Innovationsfreude der Schule.

4. Entwurf:

4.1. Datenbankstruktur

Für die Datenstruktur wurde das Relationale Datenbanksystem Microsoft-SQL ausgewählt. Beide Webanwendungen sollen später auf dieselbe Datenbank zugreifen. Damit später festgestellt werden kann, auf welchem Stand sich die Datenbank befindet, damit entsprechende neue Versionen der Datenbank installiert werden können, wurde entsprechend hierfür eine Tabelle für die Versionierung implementiert. Es resultiert daraus folgendes logisches Datenbankmodell (Anhangsnummer mit Link).

4.2. Designentwürfe

Um einheitliches Design zu ermöglichen wurden Mockups in Adobe XD erstellt. Dies hat uns ermöglicht über die ganze Anwendung hinweg einheitliches und optisch ansprechendes, aber auch möglichst barrierefreies Design zu gestalten. Die Mockups dienten als Grundlage für die spätere Entwicklung.

5.1. Das Backend

5.1.1. Implementierung der Datenbankverbindung

Damit Daten in eine Datenbank ein- und ausgelesen werden können, ist es notwendig in der Anwendung eine Verbindung zur Datenbank herzustellen. Um dies zu ermöglich wird das Object Relations Mapping (ORM) NuGet Paket Dapper verwendet. Damit können Werte aus einer Datenbank direkt in ein C# Objekt anhand der Namen von Eigenschaften überführt werden. Der String für das Öffnen einer Verbindung wird dabei in einer separaten Klasse gebaut. Ein Beispiel dafür ist zum Beispiel die Methode zum Abrufen einer Liste von Elementen anhand einer Select Abfrage (Beispielnummer im Anhang)

5.1.2. Installer und Installationsskripte

Um eine Installation der Datenbank zu ermöglichen, wurden entsprechende SQL-Skripte angelegt und im Backend abgespeichert. Um Updates zu ermöglichen, werden die Skripte nach einem bestimmten Format abgespeichert: BSL-MapsDB\_v{Große Version}.{Kleine Version}.sql. Ein Installationsskript kann dann zum Beispiel heißen „BSL-MapsDB\_v1.0.sql“. Zudem wird in allen Skripten die Datenbankversion entsprechend auf die Version des SQL-Skripts aktualisiert. Dadurch können auch neue Skripte hinzugefügt werden, wenn es Änderungen an der Datenbank geben soll.

Um eine sichere und schnelle Installation der Datenbank für Testzwecke zu ermöglichen, wurde hierfür ein Installer erstellt. Dieser kann über eine öffentliche Schnittstelle aufgerufen werden. Beim Ausführen des Installers wird immer sichergestellt, dass auch nur die neueren SQL-Installationsskripte ausgeführt werden, indem die derzeitige Version aus der Datenbank abgerufen und geprüft wird, ob es sich dabei um die aktuelle Version handelt oder es hierfür schon neuere Skripte zum Ausführen gibt. Sollte die Datenbank nicht vorhanden sein, so wird immer eine vollständige Installation bis zur aktuellen Version ausgeführt.

5.1.3. Repositories und Dto’s

Für die jeweiligen Tabellen im Backend wurden Repositories entwickelt, welche anhand von Data Transfer Objects (Dto’s), Daten entweder aus der jeweiligen Tabelle abrufen, aktualisieren, löschen oder in sie hineinschreiben. Aufgabe eines Repositories ist immer zudem, dass dieses in der Lage ist, aus einem Model ein Dto zu bilden und es dann an die Datenbankverbindung weiterzugeben beziehungsweise das Ergebnis einer Abfrage, welches immer ein Dto ist, in ein Modell zu konvertieren.

5.1.4. Services und Models

Um Inhalte aus der Datenbank im Frontend anzuzeigen, wurden Models angelegt. Der Inhalt der jeweiligen Models basiert dabei auf die Tabellen in der Datenbank.

Über Services wird es den Webanwendungen ermöglicht mit der darunterliegenden Datenbank zu interagieren, um zum Beispiel Zuweisungen zu den Räumen abspeichern zu können.

6. Adminpanel:

6.1 Login:

Diese Blazor-Seite implementiert eine Login-Oberfläche für das Adminpanel und verwendet JWT-Authentifizierung. Die Seite ist von der Authentifizierung ausgenommen, da sie mit [AllowAnonymous] versehen ist. Benutzer können sich mit einem Nutzernamen und Passwort anmelden. Bei erfolgreicher Validierung wird ein JWT-Token generiert und als Cookie gespeichert. Um das Login-Formular anzuzeigen müssen die Cookies akzeptiert werden. Die Methode OnLoginSubmit überprüft die Anmeldedaten mit Hilfe des UserServices des Backends und setzt das Token im Browser. Wenn falsche Nutzerdaten angegeben wurden, so wird dementsprechend eine Fehlermeldung angezeigt. Zudem ist eine Logout-Funktion entwickelt worden, die auf jeder Seite eingesetzt werden kann.

6.2 Karte:

Um die Kartenimplementation zu vereinfachen wurde die JavaScript-Bibliothek leaflet.js eingesetzt. Diese Bibliothek stellt eine Möglichkeit bereit Bilder als Karten zu nutzen, ermöglicht eine Zoom-Funktionalität und das setzen von Markern. Die Entwicklung dessen wurde in der JavaScript-Funktion initializeMap gemacht. Hier wurde eine interaktive Karte mit mehreren Etagen und Raum-Markern, basierend auf Leaflet.js genutzt. Die Karte verwendet einfache Koordinaten (L.CRS.Simple), was zur Folge hat, da keine Geo-Koordinaten genutzt werden müssen sondern die Pixelgröße der Bilder. Zunächst werden die Kartenbilder für drei Stockwerke. Nutzer können zwischen den Etagen wechseln, wobei sich die sichtbaren Räume und Marker entsprechend anpassen. Die Marker stellen Räume dar, deren Positionen aus dem JSON-Objekt roomsJSON geladen werden. Beim Klicken auf einen Marker wird dieser hervorgehoben, und eine Methode wird aufgerufen, um Daten in das Blazor-Frontend zu übergeben. Dort kann anhand der Id des aktuell ausgewählten Raums nun Input-Feldern mit den dazugehörigen Informationen gefüllt werden. Zusätzlich gibt es eine Funktion highlightMarker, die einen bestimmten Raum anhand seiner ID hervorhebt und das Stockwerk bei Bedarf automatisch wechselt. Damit kann nach einer Suche in der Suchleiste der gefundene Raum direkt auf der Karte fokussiert werden.

6.3 Startseite:

In dieser Blazor Seite wird die interaktive Raumkarte mit Hilfe eines Aufrufs der „initializeMap“ Methode gerendert. Hier wurde eine Suchleiste implementiert, in der nach Räumen gesucht werden kann. Die dazu benötigten Daten werden durch Methodenaufrufen des Backends geladen. Hier wird auch die JavaScript-Funktion „highlightMarker“ aufgerufen, wenn in der Suchleiste nach einem Raum gesucht wird. Zudem können Rauminformationen bearbeitet und gespeichert werden. Änderungen am Raumtyp kann zum Löschen aller dazugehörigen Zuweisungen führen. Dies muss jedoch durch den Nutzer bestätigt werden.

6.4 Klassen Seite:

Auf dieser Seite ist eine Verwaltungsoberfläche für Klassen implementiert. Sie ermöglicht die Anzeige, Bearbeitung und Erstellung von Klassen mit zugehörigen Lehrern und Räumen. Die Daten werden beim Initialisieren aus den entsprechenden Services geladen. Nutzer können eine Klasse auswählen, um deren Details in Input-Feldern zu bearbeiten. Änderungen werden durch eine Aktualisierungsfunktion gespeichert, während eine Klasse auch gelöscht oder eine neue angelegt werden kann. Die Benutzeroberfläche aktualisiert sich automatisch nach Datenänderungen.

6.5 Lehrer Seite:

Die Lehrer Seite ist technisch ähnlich aufgebaut wie die Klassen Seite. Sie bietet eine Tabelle zur Anzeige aller Lehrer mit ihren zugewiesenen Büros sowie Input-Felder zur Bearbeitung und Erstellung neuer Einträge. Beim Auswählen eines Lehrers werden dessen Details in die Eingabefelder geladen. Zudem können Lehrer bearbeitet, gelöscht oder neue hinzugefügt werden. Die Daten werden beim Initialisieren aus den zugehörigen Services geladen und bei Änderungen aktualisiert.

**Navigationsseite:**

Die Navigationsseite bildet das Herzstück der Anwendung und stellt den Nutzern eine interaktive Übersicht des Schulgebäudes zur Verfügung. Im Zuge der Umsetzung wurde zunächst die grundlegende Kartenansicht realisiert, die als Basis für sämtliche weitere Funktionalitäten dient. Dabei wurde auf eine responsive Gestaltung geachtet, sodass die Seite sowohl auf Desktop-Computern als auch auf mobilen Endgeräten optimal dargestellt wird.

In einem ersten Schritt wurde die interaktive Karte implementiert. Hierbei kamen moderne Webtechnologien zum Einsatz, um eine flüssige Darstellung und Navigation innerhalb des Gebäudes zu gewährleisten. Über die Kartenansicht können Nutzer einzelne Räume auswählen, wodurch detaillierte Informationen wie Raumname, Raumtyp und eine kurze Beschreibung angezeigt werden. Die visuelle Markierung des ausgewählten Raumes erfolgt durch eine farbliche Hervorhebung, sodass der Nutzer jederzeit den aktuellen Fokus erkennen kann.

Ein zentrales Element der Navigationsseite ist die integrierte Suchfunktion. Über eine Suchleiste können spezifische Räume oder Einrichtungen abgefragt werden. Bei Eingabe von Suchbegriffen werden in Echtzeit passende Ergebnisse vorgeschlagen, die den Nutzer direkt zur entsprechenden Position auf der Karte führen. Diese dynamische Suche wurde so konzipiert, dass sie auch bei großen Datenmengen zuverlässig und schnell reagiert.

Des Weiteren wurde eine Routenplanungsfunktion integriert, die es dem Benutzer ermöglicht, von einem Startpunkt zu einem gewünschten Ziel innerhalb des Schulgebäudes zu navigieren. Zwei Eingabefelder erlauben es, den Ausgangspunkt sowie das Ziel einzugeben, woraufhin die Anwendung den optimalen Weg berechnet und diesen direkt auf der Karte visualisiert. Dies trägt maßgeblich dazu bei, dass sich auch Besucher und neue Schüler schnell und unkompliziert im Gebäude zurechtfinden.

Abschließend wurde die Seite einem umfassenden Test unterzogen, bei dem nicht nur die Funktionalität der interaktiven Elemente, sondern auch die Performance und Barrierefreiheit im Fokus standen. Durch gezielte Optimierungen und die Implementierung von Best Practices wurde sichergestellt, dass die Navigationsseite nicht nur technisch einwandfrei funktioniert, sondern auch höchsten Ansprüchen an Benutzerfreundlichkeit und Zugänglichkeit genügt.

**Homepage:**

Die Startseite von BSL-Maps fungiert als zentrale Anlaufstelle für Nutzer und ist strategisch SEO-optimiert, um eine hohe Sichtbarkeit in Suchmaschinen zu gewährleisten. Sie bietet eine klare und ansprechende Übersicht über die Anwendung und deren Funktionen, während gleichzeitig rechtliche Anforderungen erfüllt werden.

Ein wichtiger Bestandteil der Startseite sind die rechtlichen Seiten, die den Nutzern Informationen zu Cookies, Datenschutz und Impressum bereitstellen. Diese Seiten sind einfach zugänglich und gewährleisten die Einhaltung der geltenden Vorschriften, um das Vertrauen der Nutzer zu stärken.

Die Startseite enthält zudem prägnante Verlinkungen zur Navigationsseite und zu einer „Über uns“-Seite, auf der die Philosophie und das Team hinter BSL-Maps vorgestellt werden. Diese Verlinkungen erleichtern den Nutzern die Navigation innerhalb der Anwendung und bieten einen klaren Überblick über die verschiedenen Bereiche.

Ein weiteres zentrales Element sind die integrierten Links zu GitHub und einer E-Mail-Adresse. Dies ermöglicht den Nutzern, direkt mit dem Entwicklungsteam in Kontakt zu treten, Fragen zu stellen oder Feedback zu geben. Die Verlinkungen fördern die Interaktion und schaffen eine offene Kommunikationslinie zwischen den Nutzern und den Entwicklern.

Insgesamt wurde die Startseite so gestaltet, dass sie sowohl informativ als auch benutzerfreundlich ist, um den Besuchern ein angenehmes Erlebnis zu bieten und sie dazu zu ermutigen, die Möglichkeiten von BSL-Maps voll auszuschöpfen.

### **Deployment auf Ionos**

Für das Deployment der BSL-Maps-Anwendung auf Ionos wurde zunächst eine klare Struktur der Anwendung in Form von Ordnern erstellt. Früher erfolgte der Upload der Anwendung manuell, indem die Ordnerstruktur direkt auf den Server gezogen wurde. Mittlerweile wird dieser Prozess dank eines FTP-Publish-Profils in Visual Studio automatisiert. Die Anwendung ist so organisiert, dass statische Ressourcen wie CSS, JavaScript und Bilder in einem eigenen Ordner liegen, während Datenmodelle und Razor-Views in separaten, thematisch organisierten Ordnern abgelegt werden. Diese strukturierte Organisation erleichtert die Wartung und den reibungslosen Betrieb der Anwendung.

Nachdem die lokale Version der Anwendung erfolgreich getestet wurde, wurde das Deployment über das FTP-Publish-Profil durchgeführt. In Visual Studio wurde ein Publish-Profil konfiguriert, das alle notwendigen FTP-Zugangsdaten und Zielinformationen für den Ionos-Server enthält. Dabei wurde als Zielverzeichnis auf dem Server beispielsweise „public\_html/bsl-maps“ festgelegt. Zusätzlich mussten einige Anpassungen vorgenommen werden: Die automatische HTTPS-Weiterleitung, die standardmäßig von Blazor implementiert wird, wurde entfernt, da Ionos diese Funktion bereits automatisch übernimmt. Andernfalls hätte dies zu einer Endlosschleife geführt. Weiterhin wurde der Process Mode auf „OutOfProcess“ umgestellt, um die gleichzeitige Ausführung mehrerer Websites auf dem gleichen Server zu ermöglichen.

Der eigentliche Veröffentlichungsvorgang wurde durch einen Klick auf „Publish“ in Visual Studio gestartet, wobei alle erforderlichen Dateien und Ressourcen automatisch auf den Ionos-Server übertragen werden.

Testen:

{Nummer}.{Nummer} Unit Tests in der Backend Klassenbibliothek

Zum unabhängigen Testen der Backend Klassenbibliothek wurde ein TestClient entwickelt. Dabei handelt es sich um eine .Net 8 in der Programmiersprache C# geschriebene Konsolenanwendung. In dieser können unterschiedliche Testfälle ausgewählt werden, abhängig davon welcher Service am Ende getestet werden soll. Dadurch konnten die Funktionalitäten der Bibliothek auf Fehlerfreiheit geprüft werden, ohne dass dabei eine der beiden Webanwendung zum Ausführen der Tests benötigt werden. Gleichzeitig konnten auch durch die automatisch ausführbaren Testfälle abgesichert werden, dass selbst bei Anpassungen an den bestehenden Methoden, diese sich genauso verhalten wie erwartet.

### **Integrationstest**

Um die einwandfreie Zusammenarbeit der verschiedenen Komponenten der Webanwendung sicherzustellen, wurden **Integrationstests** mit **Cypress** durchgeführt. Cypress ist ein leistungsstarkes End-to-End-Test-Framework, das speziell für moderne Webanwendungen entwickelt wurde und sich ideal für das Testen von **Blazor Server** eignet.

Die Tests konzentrierten sich auf zentrale Funktionen der Anwendung, insbesondere auf die **Navigation**, die **Suche nach Räumen** sowie die **Interaktion mit der Karte und den Administrationsfunktionen**. Dabei wurden automatisierte Tests erstellt, die sicherstellten, dass Benutzer Eingaben korrekt vornehmen können und die erwarteten Ergebnisse erhalten.

Ein wichtiger Testfall war beispielsweise die **Routenberechnung**: Hierbei wurde überprüft, ob die Anwendung nach Eingabe eines Start- und Zielraums die richtige Route berechnet und visuell darstellt. Ebenso wurde getestet, ob das **Adminpanel** ordnungsgemäß funktioniert, indem Änderungen an Räumen und Klassen vorgenommen und korrekt in der Datenbank gespeichert wurden.

Zusätzlich wurden **Fehlerszenarien simuliert**, um zu überprüfen, wie die Anwendung auf ungültige Eingaben oder unerwartete Serverantworten reagiert. So wurde sichergestellt, dass die Benutzerführung auch in problematischen Fällen intuitiv bleibt und keine unvorhergesehenen Fehler auftreten.

Durch den Einsatz von Cypress konnten potenzielle Fehler frühzeitig erkannt und behoben werden, sodass eine stabile und zuverlässige Webanwendung gewährleistet werden konnte.

### **Performance- und Barrierefreiheitstest**

Um sicherzustellen, dass die Webanwendung sowohl schnell als auch für alle Nutzer gut zugänglich ist, wurden umfassende **Performance- und Barrierefreiheitstests** durchgeführt. Hierfür kam das Tool **Google PageSpeed Insights** zum Einsatz, das verschiedene Kennzahlen zur Ladegeschwindigkeit und Benutzerfreundlichkeit einer Webseite analysiert.

Bei der Performance-Analyse wurde insbesondere die **Ladezeit der Kartenansicht** überprüft, da diese einen zentralen Bestandteil der Anwendung darstellt. Durch die Optimierung von **Bildern, Skripten und Caching-Mechanismen** konnte die Ladegeschwindigkeit verbessert werden. Zudem wurden Maßnahmen zur **Minimierung von HTTP-Anfragen** ergriffen, indem CSS- und JavaScript-Dateien zusammengefasst und komprimiert wurden.

Die Barrierefreiheitstests konzentrierten sich auf die **Nutzbarkeit für verschiedene Zielgruppen**, einschließlich Personen mit eingeschränktem Sehvermögen oder motorischen Einschränkungen. Dazu gehörten die Überprüfung von **Farbkontrasten, Tastatur-Navigation und Screenreader-Kompatibilität**. Anpassungen wie die **Verwendung klarer Schriftarten, ausreichend großer Klickbereiche und alternativer Texte für Bilder** verbesserten die Zugänglichkeit der Anwendung.

Die durch **PageSpeed Insights** gewonnenen Erkenntnisse führten dazu, dass sowohl die Performance als auch die Benutzerfreundlichkeit der Webanwendung optimiert werden konnte. Dadurch wird sichergestellt, dass die Anwendung auch bei schlechter Internetverbindung schnell geladen wird und von möglichst vielen Menschen problemlos genutzt werden kann.

Fazit:

Anhänge:

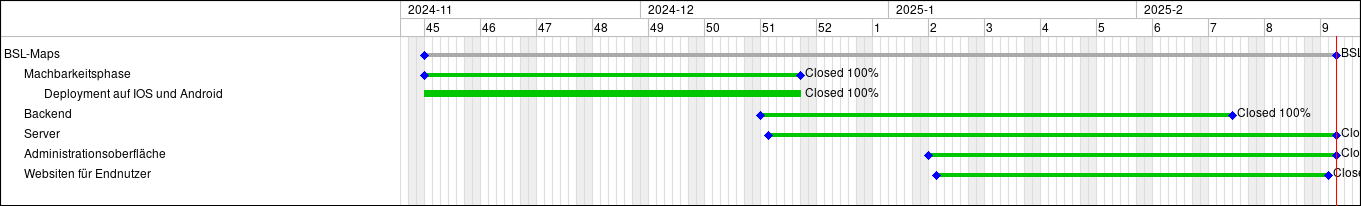
Quellen:

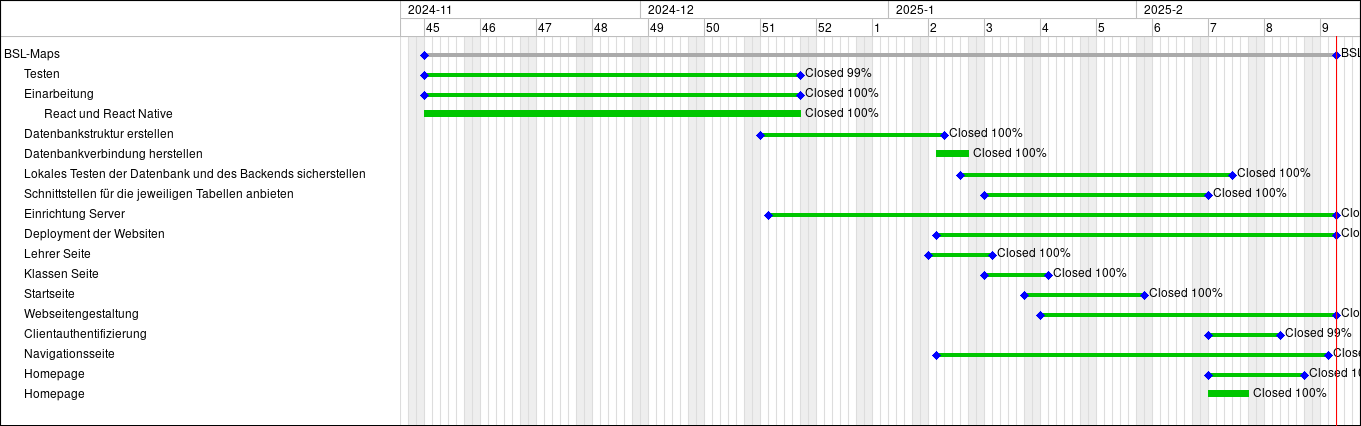
ZZZ Projects: Learn Dapper. URL: <https://www.learndapper.com/>

[Zugriff am: 18.12.2024]

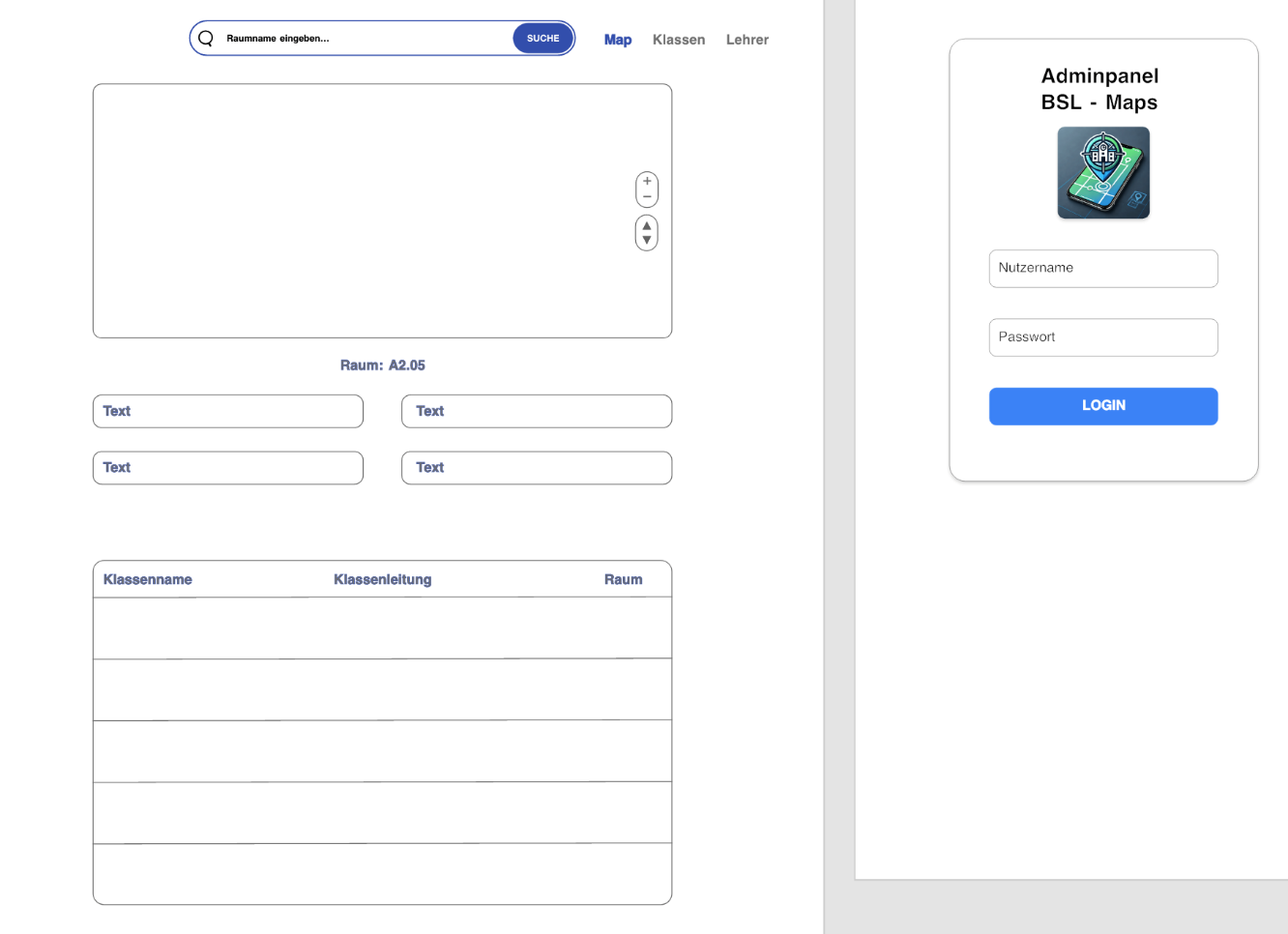
Microsoft: Hosten und Bereitstellen von serverseitigen Blazor-Apps. URL: <https://learn.microsoft.com/de-de/aspnet/core/blazor/host-and-deploy/server?view=aspnetcore-8.0> [Zugriff am: 19.12.2024]

Gantt Diagramm (Features):

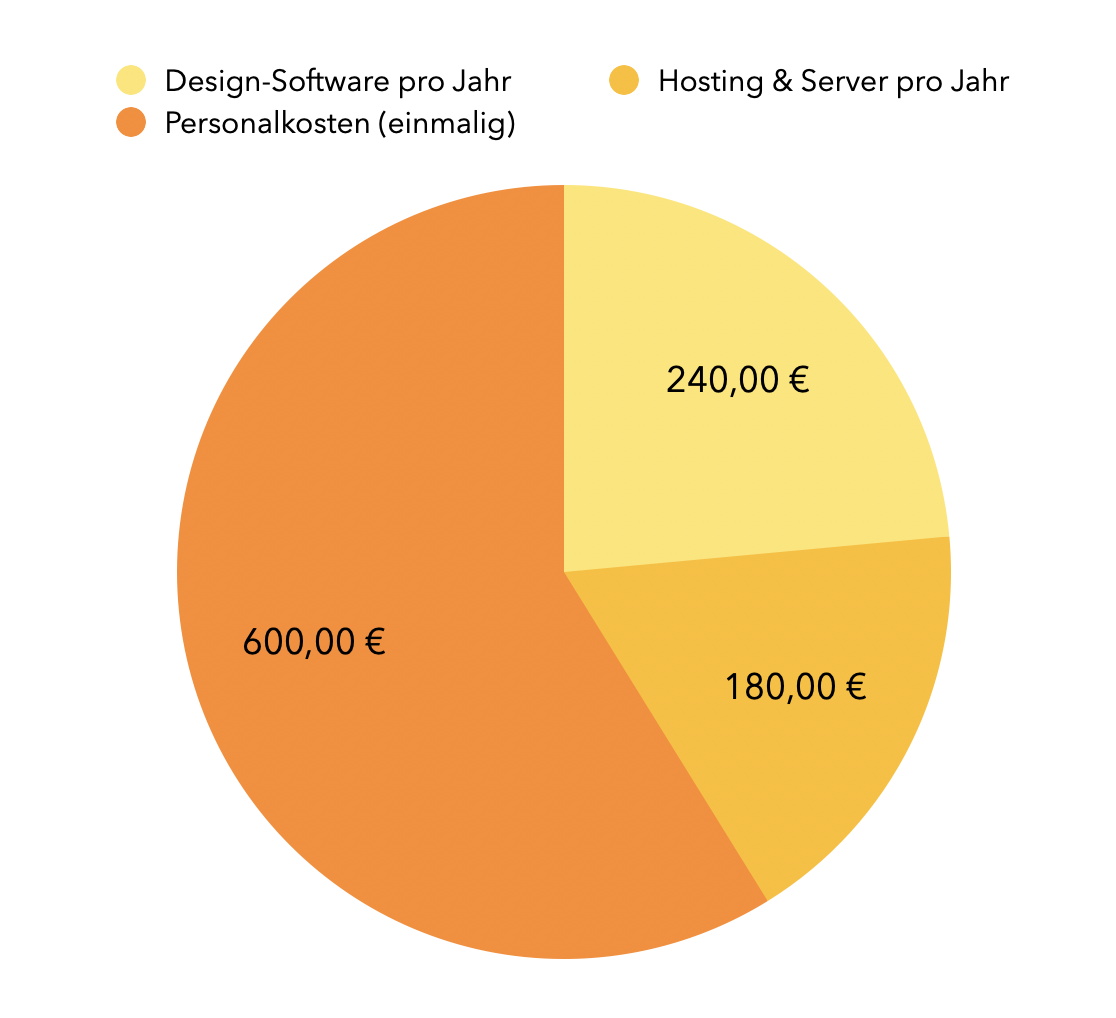


Gantt Diagramm (Stories): 

Mockup Adminpanel:



Kostenplanung:



Datenbankmodell:

Ein Bild, das Text, Schrift, Screenshot, Diagramm enthält.

KI-generierte Inhalte können fehlerhaft sein.

Use-Case Administrations- und Benutzeroberfläche:

Ein Bild, das Text, Diagramm, Reihe, parallel enthält.

KI-generierte Inhalte können fehlerhaft sein.

Backendmethode für das Abrufen von Objekten anhand einer Select Abfrage:

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.