ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

STL SYSTEMTECHNIKLABOR

Entwicklung eines Backends zur digitalen Schulwegplanung – Analyse, Konzeption und Implementierung

Daniel Grätz

Technical Report – STL-TR-2025-01 – ISSN 2364-7167

stl.htwsaar.de

Technische Berichte des Systemtechniklabors (STL) der htw saar
Technical Reports of the System Technology Lab (STL) at htw saar
ISSN 2364-7167

Daniel Grätz: Entwicklung eines Backends zur digitalen Schulwegplanung – Analyse, Konzeption und Implementierung
Technical report id: STL-TR-2025-01

First published: September 2025
Last revision: September 2025
Internal review: Markus Esch, Maximilian Altmeyer

For the most recent version of this report see: https://stl.htwsaar.de/

Title image source: Daniel Grätz

htw saar – Hochschule für Technik und Wirtschaft des Saarlandes (University of Applied Sciences)
Fakultät für Ingenieurwissenschaften (School of Engineering)
STL – Systemtechniklabor (System Technology Lab)
Prof. Dr.-Ing. André Miede (andre.miede@htwsaar.de)
Goebenstraße 40
66117 Saarbrücken, Germany
https://stl.htwsaar.de

ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

Master-Thesis

zur Erlangung des akademischen Grades
Master of Science (M. Sc.)
an der Hochschule für Technik und Wirtschaft des Saarlandes
im Studiengang Praktische Informatik
der Fakultät für Ingenieurwissenschaften

Entwicklung eines Backends zur digitalen Schulwegplanung – Analyse, Konzeption und Implementierung

vorgelegt von
Daniel Grätz

betreut und begutachtet von
Prof. Dr. Markus Esch
Prof. Dr. Maximilian Altmeyer

Saarbrücken, 30. September 2025

Selbständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit (bei einer Gruppenarbeit: den entsprechend gekennzeichneten Anteil der Arbeit) selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich erkläre hiermit weiterhin, dass die vorgelegte Arbeit zuvor weder von mir noch von einer anderen Person an dieser oder einer anderen Hochschule eingereicht wurde.

Darüber hinaus ist mir bekannt, dass die Unrichtigkeit dieser Erklärung eine Benotung der Arbeit mit der Note „nicht ausreichend“ zur Folge hat und einen Ausschluss von der Erbringung weiterer Prüfungsleistungen zur Folge haben kann.

Saarbrücken, 30. September 2025

Daniel Grätz

Zusammenfassung

Im Straßenverkehr sind Kinder in erhöhtem Maße gefährdet, vor allem auf ihrem Schulweg. In der Vergangenheit wurde mit verschiedenen Maßnahmen wie Verkehrserziehung, Kampagnen oder bauliche Anpassungen an Gefahrenstellen versucht, die Zahl der verunglückten Kinder zu reduzieren; die erhoffte ist Trendwende bislang ausgeblieben. Eine weitere geeignete Maßnahme ist die Erstellung sogenannter Schulwegpläne, die Handlungsempfehlungen für einen sicheren Weg zur Schule geben sollen. Allerdings ist die Erstellung dieser Pläne häufig ineffizient und aufwendig, da sie papierbasiert abläuft und mehrere Beteiligte miteinbeziehen muss.

Ziel dieser Arbeit ist die Analyse, Konzeption und Umsetzung eines Backends zur Unterstützung der Schulwegplanung, während in einer Parallelarbeit die Entwicklung eines Frontends gezeigt wird. Es ermöglicht die papierlose Erfassung, Verarbeitung und Nutzung von Daten wie Umfragen, Gefahrenstellen und Schulwegen für den Planungsprozess. Dabei werden gezielt unterschiedliche Benutzergruppen untersucht und systematisch die Anforderungen an das System herausgearbeitet.

Die Arbeit beschreibt die Vorgehensweise, die Anforderungen der Nutzer, das Datenmodell sowie die Systemarchitektur und zeigt, wie das entwickelte System die Erstellung von Schulwegplänen effizienter gestalten kann. Abschließend werden die Implementierungsergebnisse anhand von Tests evaluiert und von bestehenden Lösungen abgegrenzt.

Inhaltsverzeichnis

1 Einleitung

Die vorliegende Arbeit untersucht die Schulwegplanung mit dem Ziel, den Prozess der Datenerhebung und -aufbereitung effizienter zu gestalten. Sie betrachtet insbesondere, wie relevante Informationen strukturiert erfasst und systematisch genutzt werden können, um die Erstellung von Schulwegplänen zu unterstützen.

1.1 Motivation

Im Jahr 2024 wurden nach Angaben des Statistischen Bundesamtes 27260 Kinder unter 15 Jahren im Straßenverkehr verletzt oder getötet; die Zahl der tödlich verunglückten Kinder stieg dabei im Vergleich zum Vorjahr um neun auf 53 Fälle. Ein erheblicher Teil dieser Unfälle ereignete sich in den Zeiträumen zwischen 7 und 8 Uhr am Morgen sowie zwischen 15 und 17 Uhr am Nachmittag, also zu den typischen Zeiten des Schulweges [13]. Damit ist der Schulweg nach wie vor eine besonders kritische Phase im Schulalltag vieler Kinder.

Um die Verkehrssicherheit in diesem Bereich zu verbessern, setzen Kommunen, Schulträger und Länder seit Jahrzehnten auf verschiedene Maßnahmen. Neben Verkehrserziehungsprogrammen, baulichen Anpassungen im öffentlichen Raum oder begleitenden Kampagnen gehören sogenannte Schulwegpläne zu den zentralen Instrumenten. Diese Pläne werden von den zuständigen Stellen erstellt und veröffentlicht, um Eltern und Kindern Handlungsempfehlungen für einen sicheren Schulweg zu geben. Sie enthalten in der Regel empfohlene Routen aus verschiedenen Teilen des Schulbezirkes, Hinweise auf sichere Überquerungsmöglichkeiten, Bushaltestellen sowie markante Orientierungspunkte wie Spielplätze oder Denkmäler. Einige dieser Pläne können in Anhang A eingesehen werden.

Die Erstellung solcher Pläne ist jedoch mit verschiedenen Herausforderungen verbunden. Zwar wird die Sichtweise der Kinder bereits berücksichtigt, etwa durch Befragungen oder durch die Einbindung von Rückmeldungen seitens Eltern und Schulen. Der damit verbundene Prozess ist jedoch häufig aufwendig, da die Datenerhebung und -auswertung größtenteils in Papierform erfolgt und manuelle Arbeitsschritte erfordert. Fragebögen werden verteilt, handschriftlich ausgefüllt, gesammelt und anschließend händisch ausgewertet. Dieser hohe organisatorische Aufwand erschwert es, die Pläne in akzeptablen Zeiträumen zu erstellen und zu veröffentlichen.

Hinzu kommt, dass die fertigen Pläne in der Regel statisch bleiben. Sie werden als Druckversion in Elternbriefen oder als Ausdruck verteilt, teilweise auch als PDF veröffentlicht, jedoch nicht kontinuierlich gepflegt, auch wegen des anstehenden erneuten Verwaltungsaufwandes. Damit fehlt die Möglichkeit, auf veränderte Rahmenbedingungen wie neue Baustellen, geänderte Verkehrsführungen oder zusätzliche Gefahrenstellen einzugehen.

Darüber hinaus variiert das Vorgehen zur Erstellung von Schulwegplänen stark zwischen den Bundesländern und oft auch zwischen einzelnen Kommunen. Diese Unterschiede sind durch den Bildungsföderalismus bedingt, führt aber dazu, dass keine einheitlichen Standards bestehen. Manche Kommunen verfügen über etablierte Verfahren und aktuelle Pläne, während andere Schulen oder Schulträger nur eingeschränkt Pläne

erstellen, wenn überhaupt.

Gleichzeitig eröffnen technologische Entwicklungen neue Möglichkeiten: Die weite Verbreitung mobiler Endgeräte und digitaler Anwendungen schafft Potenzial, den Prozess der Datenerhebung zu vereinfachen, die Aktualisierung von Informationen zu beschleunigen und die Bereitstellung der Ergebnisse in nutzerfreundlicher Form zu verbessern. Insbesondere die Einbindung der Erfahrungen von Kindern in digitaler Form bietet die Möglichkeit, Planungsprozesse effizienter zu gestalten.

Die genannten Aspekte zeigen, dass die bestehenden Verfahren zur Erstellung und Bereitstellung von Schulwegplänen einer Weiterentwicklung bedürfen. Insbesondere im Hinblick auf Aktualisierbarkeit, Effizienz und digitale Verfügbarkeit besteht ein erkennbarer Handlungsbedarf.

1.2 Zielsetzung

Ziel dieser Arbeit ist die Konzeption und Umsetzung eines Backends zur Unterstützung der Schulwegplanung. Dabei liegt der Schwerpunkt auf der Entwicklung eines Systems, das den bislang papierbasierten Prozess der Datenerhebung, -aufbereitung und -verwaltung effizienter gestaltet. Das Backend soll die Erfassung und Strukturierung relevanter Informationen ermöglichen, um die Erstellung von Schulwegplänen zu unterstützen. Die Entwicklung eines Frontends wird vorliegend nicht behandelt sondern ist Thema einer Parallelarbeit.

Die vorliegende Arbeit untersucht, welche Anforderungen die verschiedenen Nutzergruppen (Planer, Schüler, Eltern, Schulen) an ein solches System stellen. Dazu gehören insbesondere die Erfassung von Umfragen, Gefahrenstellen und Schulwegen. Das System soll diese Daten in geeigneter Form bereitstellen, sodass sie für die Planung genutzt werden können, ohne das bereits genutzte Verfahren der papierbasierten Bereitstellung zu verändern. Laufende Aktualisierungen der Pläne sind möglich; die digitale Unterstützung dient aber primär der Effizienzsteigerung im erstmaligen Erstellungsprozess.

Methodisch soll die Arbeit aufzeigen, wie ein Backend aufgebaut werden kann, um die Daten zu erfassen, zu verarbeiten und für die Nutzer in aufbereiteter und visueller Form zugänglich zu machen. Dabei wird untersucht, welche Datenmodelle, Schnittstellen und Technologien notwendig sind, um die Anforderungen umzusetzen.

Die Zielsetzung lässt sich anhand der folgenden zentralen Fragestellungen formulieren, die im Verlauf der Arbeit beantwortet werden:

- Welche funktionalen und nicht-funktionalen Anforderungen bestehen an ein Backend für die digitale Schulwegplanung?
- Welche Nutzerrollen sind relevant, und wie lassen sich deren Anforderungen an das System berücksichtigen?
- Inwiefern kann ein digitales System den Erstellungsprozess von Schulwegplänen effizienter gestalten?
- Wie können erhobene Daten wie Schulwege, Gefahrenstellen und Umfragen zuverlässig erfasst, verarbeitet und aufbereitet werden?
- Wie kann das Backend effizient implementiert werden, um die definierten Anforderungen erfüllen?
- Wie kann die Sicherheit der bereitgestellten Daten gewährleistet werden, insbesondere im Hinblick auf den Datenschutz?

Die abschließende Beantwortung dieser Fragen erfolgt im Fazit, das die erzielten Ergebnisse zusammenführt, reflektiert und aufzeigt, in welchem Umfang die entwickelte Lösung den Erstellungsprozess von Schulwegplänen effizienter gestalten kann.

Wesentlich ist noch, dass das vorliegende Gesamtprojekt lediglich einen Prototypen darstellt, das im Rahmen einer Zusammenarbeit mit dem Ministerium für Wirtschaft und Verkehr des Saarlandes entstanden ist. Dieser Prototyp soll verdeutlichen, welchen Mehrwert eine digitale Lösung für das Voranbringen der Schulwegplanung im Saarland und ggf. darüber hinaus hat.

1.3 Aufbau der Arbeit

Nach der vorliegenden Einleitung, in der Motivation und Zielsetzung dargelegt wurden, folgt im zweiten Kapitel die Darstellung der theoretischen und technischen Grundlagen. Zunächst wird die Schulwegplanung im bildungsföderalen Kontext eingeordnet; anschließend wird die menschenzentrierte Gestaltung nach ISO 9241-210 erläutert, bevor die für die Umsetzung benötigten technischen Grundlagen vorgestellt werden.

Das dritte Kapitel beschäftigt sich mit der menschenzentrierten Anforderungsanalyse gemäß der bereits erwähnten Norm. Neben der Beschreibung des Vorgehens zur Erhebung von Anforderungen werden verschiedene Personas entwickelt, die unterschiedliche Nutzergruppen repräsentieren. Aus deren Bedürfnissen werden die funktionalen und nicht-funktionalen Anforderungen abgeleitet, deren Implikationen für die Entwicklung eines Backends dargestellt werden. Außerdem werden bestehende Lösungen für vergleichbare Anwendungsfälle untersucht und Lücken aufgezeigt, die das zu entwickelnde Backend zu füllen versucht.

Darauf aufbauend wird im vierten Kapitel das Konzept der Lösung vorgestellt. Es beschreibt die Architektur des Gesamtsystems mit Frontend, Backend und Datenbank auf der einen Seite, und die Systemarchitektur speziell des Backends. Das Datenmodell wird erläutert, der Datenfluss wird dargestellt und es wird auf die Gestaltung der REST-API sowie der externen Schnittstellen eingegangen. Sicherheits- und Datenschutzaspekte werden hier ebenfalls diskutiert.

Das fünfte Kapitel widmet sich der Implementierung. Nach einer Begründung der ausgewählten Technologien wird das Datenbankdesign vorgestellt. Anschließend wird anhand von ausgewählten Implementierungsbeispielen gezeigt, wie einzelne Anforderungen umgesetzt werden und welche Herausforderungen bewältigt werden müssen. Abgeschlossen wird das Kapitel mit einer Vorstellung der Teststrategie.

Im Anschluss daran erfolgt die Evaluation im sechsten Kapitel. Dort werden die erzielten Ergebnisse anhand von Qualitätskriterien beurteilt, Möglichkeiten für Feldtests und qualitative Analysen dargestellt und schließlich die Umsetzung der zuvor definierten Anforderungen überprüft.

Bevor die Arbeit mit einer Zusammenfassung endet, wird im siebten Kapitel das entwickelte Backend vom Stand der Technik abgegrenzt.

Das abschließende achte Kapitel fasst die Arbeit noch einmal zusammen, reflektiert die Zielerreichung und zeigt mögliche Weiterentwicklungen sowie Verbesserungspotenziale auf. Ein Gesamtfazit schließt die Arbeit ab.

2 Grundlagen

Im Folgenden wird zunächst der Stand der Schulwegplanung in Deutschland dargestellt, anschließend die menschenzentrierte Gestaltung nach ISO 9241-210 erläutert und schließlich auf technische Begriffe eingegangen, die für das Verständnis der vorliegenden Arbeit wichtig sind.

2.1 Schulwegplanung im (bildungsföderalen) Überblick

Aufgrund des Bildungsföderalismus in Deutschland und der daraus folgenden Kulturhoheit der Länder liegt die alleinige Zuständigkeit für Bildungspolitik bei jedem einzelnen Bundesland. Dementsprechend kann es keine bundesweite Regelung zur Schulwegplanung geben, die über nicht bindende Beschlüsse der Kultusministerkonferenz hinausgeht. In diesem Abschnitt soll der aktuelle Stand der Bemühungen rund um das Thema Schulwegsicherheit in den Bundesländern kurz dargelegt werden.

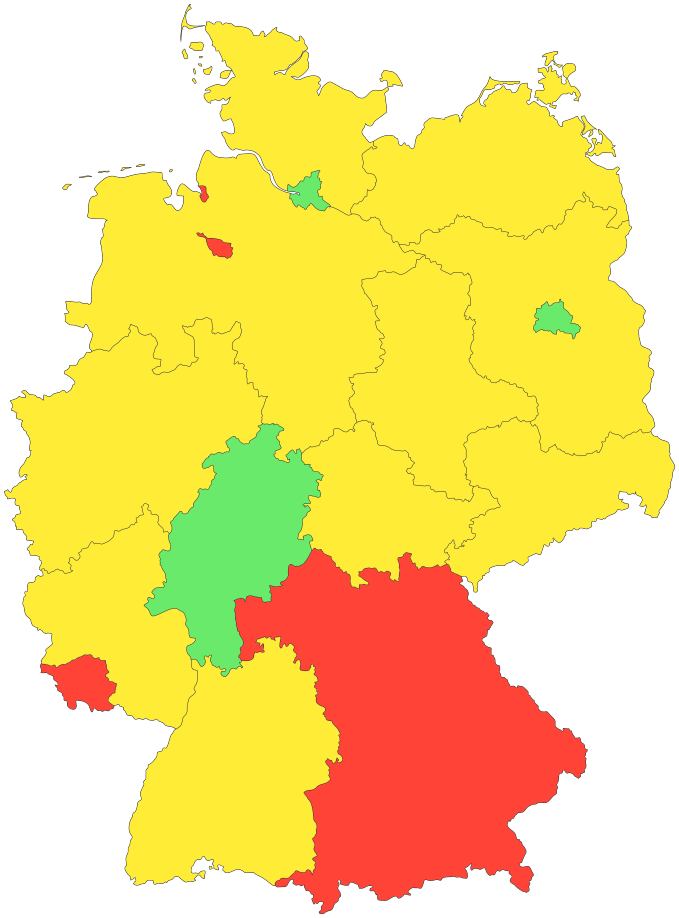

Abbildung 2.1: Deutschlandkarte mit dem aktuellen Stand der Schulwegplanung; siehe Text für Legende. (Karte: modifiziert nach `https://d-maps.com`)

Wie es um die Schulwegplanung in Deutschland bestellt ist, zeigt Abbildung 2.1. Die

Farben stehen für das Vorhandensein von Regelungen zur Schulwegplanung: Rote Länder haben weder Regelungen noch Empfehlungen, gelbe Länder nur Empfehlungen, grüne Länder Regelungen. Zu erkennen ist der häufig so bezeichnete „Flickenteppich“, wenn es um Angelegenheiten der Gesetzgebungskompetenz der Länder geht. Der FUSS e.V. hat die folgenden Informationen auf seiner Webseite gesammelt [59]:

- In **Berlin**, **Hamburg** und **Hessen** gibt es landesrechtliche Regelungen, die die Erstellung und den Einsatz von Schulwegplänen vorschreiben. Allerdings unterscheidet sich die Zuständigkeiten:

  Berlin: Bis 2022 war die Landesverkehrswacht zuständig; seitdem wird das Projekt von einer Designerin fortgeführt [45].

  Hamburg: In Hamburg war die Polizei zuständig, die Schulen standen beratend zur Seite. Nach deren Angaben verfügt jede Grundschule in Hamburg über Schulwegpläne. Allerdings ist mit Stand 30.11.2024 niemand mehr für die Planung zuständig.

  Hessen: Zuständig sind die Schulen für die Klassenstufen 1 bis 7, in Zusammenarbeit mit der Polizei und anderen Behörden.

- In **Baden-Württemberg**, **Brandenburg**, **Mecklenburg-Vorpommern**, **Niedersachen**, **Nordrhein-Westfalen**, **Rheinland-Pfalz**, **Sachsen**, **Sachsen-Anhalt**, **Schleswig-Holstein** und **Thüringen** gibt es keine rechtlichen Rahmenbedingungen, sondern nur Empfehlungen, die sich wie folgt unterscheiden:

  Baden-Württemberg: Hier wird ein Schulwegplaner zur Verfügung gestellt, der die Kinder in den Planungsprozess durch Umfragen einbezieht. Sie werden von den jeweiligen Kommunen ausgewertet, ist aber keine Pflicht für die Schulen [31].

  Brandenburg: Es wird lediglich auf die Broschüre der Bundesanstalt für Straßen- und Verkehrswesen verwiesen [55].

  Mecklenburg-Vorpommern: Nur eine Empfehlung zur Erstellung; Zuständigkeit bei den Schulverwaltungsämtern.

  Niedersachsen: Verweis auf den Landesverkehrswacht Niedersachen e.V.; ansonsten nur Empfehlungen.

  Nordrhein-Westfalen: Empfehlungen für den Unterricht und die Schulen; ansonsten Ausloten von Möglichkeiten der Digitalisierung, aber zum Stand dieser Arbeit noch ohne konkrete Ergebnisse [37].

  Rheinland-Pfalz: Schulen müssen Verantwortlichen für Verkehrs- und Mobilitätserziehung benennen; Schulwegpläne sind Teil der Lehrerausbildung.

  Sachsen: Empfehlungen an die Straßenverkehrsbehörden.

  Sachsen-Anhalt: Empfehlungen an die Straßenverkehrsbehörden.

  Schleswig-Holstein: Empfehlungen an alle Beteiligten. Verpflichtung war kurzzeitig in Planung, wurde aber wieder verworfen.

  Thüringen: Nur Empfehlungen.

- Die drei verbleibenden Bundesländer, **Bayern**, **Bremen** und das **Saarland**, geben keine (landesweiten) Empfehlungen ab.

Zusammenfassend lässt sich feststellen, dass die Schulwegplanung in Deutschland stark von der föderalen Struktur geprägt ist. Während einzelne Bundesländer verbindliche Regelungen haben, bestehen in vielen anderen lediglich unverbindliche Empfehlungen oder gar keine Vorgaben. Insgesamt ergibt sich damit ein sehr unterschiedliches Bild, das den eingangs erwähnten Begriff des „Flickenteppichs“ nochmal unterstreicht.

2.2 Menschenzentrierte Gestaltung nach ISO 9241-210

Im Folgenden wird die Norm ISO 9241-210 vorgestellt, anhand derer im weiteren Verlauf die Anforderungsanalyse durchgeführt wird.

2.2.1 Einführung

Die Norm DIN EN ISO 9241-210 ist der internationale Standard zu menschenzentrierten Gestaltung interaktiver Systeme. Das Ziel ist es, gebrauchstaugliche und effiziente Systeme zu erstellen, die die Bedürfnisse aller Benutzergruppen bestmöglich befriedigen und menschliche Faktoren und die Ergonomie berücksichtigen. Die Norm definiert den Begriff „Ergonomie“ anhand der Norm ISO 6385, die den Begriff wiederum von der *International Ergonomics & Human Factors Association* (*IEA*) übernommen hat. Demnach ist Ergonomie

> „[...] die wissenschaftliche Disziplin, die sich mit dem Verständnis der Wechselwirkungen zwischen Menschen und anderen Elementen eines Systems befasst, sowie die Profession, die Theorie, Prinzipien, Daten und Methoden anwendet, um Systeme so zu gestalten, dass das Wohlbefinden des Menschen und die Gesamtleistung des Systems optimiert werden.“ [6].

Daraus folgt zudem die Bedeutung der Berücksichtigung von Eigenschaften, Fähigkeiten und Grenzen der (menschlichen) Benutzer [25, 26].

Bei dem vorliegenden Projekt ist die frühe Einbindung der Nutzer wichtig. Nur dann, wenn die Bedürfnisse aller Beteiligten frühzeitig erfasst, analysiert und verstanden werden, kann die entwickelte Lösung einen echten Mehrwert bieten und einen Beitrag zur Sicherheit von Schulwegen leisten.

2.2.2 Grundprinzipien

Kern der ISO 9241-210 sind eine Reihe von Grundprinzipien, die das Fundament der menschenzentrierten Gestaltung bilden. Sie bilden dern Kern der Norm und geben vor, wie die Anforderungen erhoben und in den Entwicklungsprozess einbezogen werden. Diese sechs Grundprinzipien sollen nun im Folgenden dargestellt werden.

Nutzungskontext

Die Anforderungsanalyse erfordert es, sich mit dem späteren Nutzungskontext auseinanderzusetzen. Nutzungskontext meint dabei bestimmte Merkmale von Benutzern, ihre Aufgaben und ihre Umgebungen. Dabei müssen alle relevanten Benutzergruppen identifiziert und im Planungsprozess berücksichtigt werden. Der Norm nach ist das unvollständige oder falsche Verständnis der Benutzerbedürfnisse der Hauptgrund für den Misserfolg. Außerdem macht die Norm deutlich, dass die Tauglichkeit eines Produktes immer vom Benutzer abhängt; so kann man die Benutzererfahrung und -bedürfnisse eines Produktes zur Verwaltung von Fitnessdaten beispielsweise nicht mit denen einer Unternehmenssoftware gleichsetzen [25].

Bezogen auf das vorliegende Projekt kann es beispielsweise bedeuten, dass auf eine kindgerechte Oberfläche geachtet werden muss, da die Nutzergruppe der Grundschüler ggf. nicht oder nur unzureichend lesen kann.

Einbeziehung der Benutzer

Die wichtigste Wissensquelle ist der Benutzer. Er liefert den Nutzungskontext, kann seine Aufgaben beschreiben und wie er das zu entwickelnde System benutzen wird (oder benutzen möchte). Die Formen der Beteiligung sind vielfältig: Im Vorfeld der Entwicklung bei der Anforderungsanalyse, während der Entwicklung zur Fortschrittskontrolle und nach der Entwicklung zur Evaluation des Gesamtsystems. Je mehr Kontakt und Austausch es zwischen Entwicklern und Benutzern gibt, desto wertvoller können die Beiträge sein [25].

Ein wichtiger Teil der Einbeziehung ist die benutzerzentrierte Evaluierung. Die Rückmeldungen der Benutzer sind ein wesentliches Mittel, ein System während des Entwicklungsprozesses fortlaufend zu prüfen und zu verbessern. Durch kontinuierliche Evaluierung wird das Risiko verringert, dass das System nach Abschluss nicht den Anforderungen entspricht. Zudem liefern Rückmeldungen während des praktischen Einsatzes wertvolle Hinweise für langfristige Verbesserungen [25].

Praktisch kann dies durch Interviews, Demos, Diskussionen in Fokusgruppen oder Begleitung im Arbeitsalltag geschehen.

Iterationen

Viele Anforderungen stellen sich erst im Laufe der Entwicklung heraus und sind nicht von Beginn an fassbar. Daher ist eine gewisse Iteration in der Entwicklung benutzerzentrierter Systeme nötig, durch die Modelle, Szenarien und Prototypen kontinuierlich überarbeitet werden und ständiger Neubewertung unterworfen sind. Grundlage für diese Iterationen sind die bereits erwähnte Einbeziehung der Benutzer während des gesamten Entwicklungsprozesses. Auch technische oder organisatorische Rahmenbedingungen können sich ändert und können eine Iteration erforderlich machen [25].

Im Kontext der Schulwegplanung bedeutet dies beispielsweise, dass Formulare oder Anzeigeelemente kontinuierlicher Veränderung unterliegen, bis sie bei Projektabschluss von den Benutzergruppen angenommen werden.

Berücksichtigung der User Experience

Die User Experience umfasst nicht nur die Bedienbarkeit, sondern auch die Wahrnehmung, Gefühle, Zufriedenheit und das langfristige Wohlbefinden bei der Arbeit mit dem System. Deshalb ist es wichtig, dass auch organisatorische Aspekte und begleitende Ressourcen berücksichtigt werden. Die Aufgaben zwischen Benutzern und System müssen sorgfältig aufgeteilt werden; dafür können repräsentative Benutzer(-gruppen) erstellt und untersucht werden [25].

Interdisziplinäre Zusammenarbeit

Ein Entwicklungsteam, das ein System menschenzentriert gestaltet, sollte unterschiedliche Kompetenzen verschiedener Fachrichtungen vereinen. Neben den klassischen Informatikrichtungen helfen die Beiträge von Experten für Ergonomie, HCI oder anderen Fachrichtungen enorm beim Verständnis und der Begegnung von Benutzerbedürfnissen. Eine Vielfalt von Perspektiven unterstützt die Kreativität aller und fördert das Verständnis

für die Einschränkungen anderer Disziplinen. Dadurch steigt die Qualität und Akzeptanz der entwickelten Systeme [25].

2.2.3 Nutzen und Herausforderungen

Die menschenzentrierte Gestaltung nach ISO 9241-210 bietet einige Vorteile. Durch die frühe Einbeziehung der späteren Benutzer können Anforderungen sehr genau erfasst werden; die weitere Einbeziehung ermöglicht es, Fehlentwicklungen vorzubeugen. Dies führt langfristig zu besseren Systemen, die auf die Bedürfnisse der Benutzer ausgerichtet sind und dadurch eine höhere Akzeptanz und Zufriedenheit unter diesen erreichen. Die iterative Vorgehensweise entwickelt das Produkt, ausgehend von einem Prototypen, kontinuierlich weiter; dies reduziert das Risiko später Änderungswünsche von Kunden während der Implementierungsphase oder danach.

Allerdings bestehen auch Herausforderungen. Die Identifikation und Einbindung aller relevanten Benutzergruppen kann sehr zeitaufwendig sein, insbesondere, wenn es gegenläufige Interessen und Anforderungen gibt. Der iterative Entwicklungsprozess erfordert organisatorischen Aufwand und eine gewisse Disziplin innerhalb des Entwicklerteams. Auch kann es manchmal schwierig sein, die Balance zwischen Benutzerwünschen und den technischen und wirtschaftlichen Rahmenbedingungen zu finden. Letztlich kann auch die korrekte Interpretation von Benutzerwünschen am mangelnden Verständnis für HCI oder Ergonomie scheitern.

2.2.4 Zusammenfassung

Die Norm ISO 9241-210 liefert einen anerkannten Rahmen für die menschenzentrierte Gestaltung interaktiver Systeme. Wesentliche Punkte sind die frühzeitige Einbindung der Benutzer, die Berücksichtigung der Benutzerumgebung, der iterative Entwicklungsprozess, die Rücksicht auf eine gute Benutzungserfahrung und die interdisziplinäre Zusammenarbeit. Diese Prinzipien der ISO 9241-210 ermöglichen es, Systeme zu entwickeln, die den Anforderungen der Zielgruppen entsprechen. Auf der anderen Seite müssen der zeitliche Aufwand, die Organisation auf Seite der Entwickler und die fachliche Kompetenz des Teams auf dem Bereich der HCI und Ergonomie beachtet werden.

2.3 Technische Grundlagen

Für das Verständnis der vorliegenden Arbeit, speziell der Implementierung, sind einige technische Grundlagen nötig, die in diesem Abschnitt erläutert werden.

2.3.1 Client-Server-Architektur

Einer der grundlegendsten Architekturen von verteilten Systemen ist die Client-Server-Architektur. Sie bildet die Basis vieler moderner Softwaresysteme und beschreibt im Kern die Aufteilung von Funktionalität in mindestens zwei getrennte Komponenten: den Client und den Server [11].

Der Client ist in aller Regel für die Interaktion mit dem Benutzer zuständig. Er präsentiert Daten in geeigneter Form, etwa in einer grafischen Benutzeroberfläche, und nimmt Eingaben entgegen. Typischerweise läuft der Client auf einem Endgerät wie einem Computer, einem Mobiltelefon oder einem eingebetteten System. Seine Hauptaufgabe besteht darin, die Kommunikation mit dem Server zu führen und die Antworten in eine für den Benutzer verständliche Form zu bringen [53]. Ein Server kann auch gleichzeitig Client sein [11].

Im Gegensatz dazu übernimmt der Server die Rolle eines Dienstleisters: Er verarbeitet eingehende Anfragen, greift auf Datenbanken oder andere Ressourcen zu und antwortet entsprechend. Er enthält häufig die gesamte Geschäftslogik des Gesamtsystems, also die zentralen Prozesse, nach denen Daten verarbeitet und Entscheidungen getroffen werden. Der Server läuft in der Praxis fast immer auf einem vom Client verschiedenen Gerät, meist auf speziell dafür vorgesehenen Systemen im Internet oder in Rechenzentren, die hohe Verfügbarkeit und Rechenkapazität gewährleisten, je nach Anwendungsfall [53].

Client und Server kommunizieren über eine festgelegte Schnittstelle. Auf der Anwendungsschicht (Schicht 7 des OSI-Modells) erfolgt diese Kommunikation häufig über das HTTP(S)-Protokoll, in manchen Anwendungsfällen auch über andere Protokolle wie FTP, SMTP oder WebSocket [7, 52]. Wesentlich ist dabei, dass die Schnittstellen definiert und unabhängig von der konkreten Implementierung auf Client- oder Serverseite sind. Damit wird sichergestellt, dass unterschiedliche Systeme miteinander interagieren können, solange sie das vereinbarte Protokoll unterstützen.

Zusammenfassend lassen sich für die Client-Server-Architektur also einige Eigenschaften formulieren [7]:

Trennung: Eine klare Trennung zwischen Präsentationsschicht (Client) und Geschäftslogik (Server) sorgt für fest definierte Zuständigkeiten. Der Client muss nichts über die Funktionsweise des Servers wissen und umgekehrt, solange die Schnittstellen bekannt sind. Diese Entkopplung ermöglicht sowohl Flexibilität als auch Austauschbarkeit einzelner Komponenten.

Mehrbenutzerfähigkeit: Ein Server kann mehrere Clients gleichzeitig bedienen. Die konkrete Ausgestaltung dieser Mehrbenutzerfähigkeit ist implementierungsabhängig: Synchrone Verarbeitung bearbeitet die Anfragen in der Reihenfolge ihres Eingangs, während asynchrone Verarbeitung mit (scheinbarer) Parallelität arbeitet. Darüber hinaus können mehrere Server hinter einem Lastverteiler zusammengefasst werden, sodass Anfragen parallel und gleichmäßig verteilt abgearbeitet werden. Auf diese Weise können Systeme auch bei erhöhten Zugriffszahlen arbeiten.

Skalierbarkeit: Durch die Entkopplung beider Komponenten können diese unabhängig voneinander erweitert oder dupliziert werden. Ein Server lässt sich durch den Einsatz leistungsfähigerer Hardware (vertikale Skalierung) oder durch das Hinzufügen zusätzlicher Instanzen (horizontale Skalierung) an wachsende Anforderungen anpassen. Ebenso können Clients in großer Zahl hinzugefügt werden, ohne dass die Architektur angepasst werden muss.

Wartbarkeit: Die Geschäftslogik des Servers kann angepasst werden, ohne dass Clients verändert werden müssen. Solange die öffentliche Schnittstelle unverändert bleibt, erhalten sämtliche Clients sofort Korrekturen, Optimierungen oder neue Funktionen. Dies erleichtert die Pflege und Weiterentwicklung erheblich, da Änderungen nur an einem Ort und nicht auf allen Endgeräten durchgeführt werden müssen.

Aus diesen Eigenschaften lassen sich Vor- und Nachteile ableiten, die in Tabelle 2.1 näher dargestellt werden [7].

Vorteile	Nachteile
Unabhängigkeit von der konkreten Client-Implementierung: Je nach Endgerät können verschiedene Clients entwickelt werden, die alle denselben Server nutzen.	Der Server ist ein Single Point of Failure, sofern keine Redundanzen vorhanden sind. Ein Ausfall kann dazu führen, dass alle Clients gleichzeitig funktionsunfähig werden.
Pflege und Aktualisierung der Geschäftslogik nur auf dem Server nötig, wodurch sich neue Funktionalitäten oder Fehlerbehebungen unmittelbar auf alle Clients auswirken.	Netzwerklatenzen können zu verzögerter Bereitstellung von Daten oder Diensten führen. Insbesondere bei komplexen Anfragen oder schlechter Verbindung kann die Benutzerfreundlichkeit beeinträchtigt sein.
Da die eigentliche Verarbeitung beim Server erfolgt, benötigen Clients im Vergleich weniger Rechenleistung und Speicher. Dies ermöglicht den Einsatz auch auf leistungsschwächeren Geräten.	

Tabelle 2.1: Vor- und Nachteile der Client-Server-Architektur

Insgesamt stellt die Client-Server-Architektur eine verbreitete und flexible Architektur dar, die sich durch klare Aufgabenteilung, zentrale Datenhaltung und einfache Erweiterbarkeit auszeichnet, jedoch stets mit Blick auf Ausfallsicherheit, Skalierbarkeit und Sicherheit umgesetzt werden muss.

2.3.2 REST

Eine REST-API ist ein Architekturstil für die Kommunikation in verteilten Systemen. REST steht dabei für „**Re**presentational **S**tate **T**ransfer“ und soll verdeutlichen, dass ein Zustand der Anwendung in einen anderen Zustand der Anwendung übergeht; ein Zustand bezeichnet dabei die Gesamtheit aller Komponenten einer Anwendung, inklusive der gesteuerten Abhängigkeiten (Datenbank, Dateisysteme, ...) [18].

REST folgt dabei definierten Prinzipien: Ressourcen werden beispielsweise über eindeutige URI identifiziert (`/schools`, `/planning`, `/survey`, ...) und über die bekannten HTTP-Methoden `GET`, `POST`, `DELETE`, `PUT`, `PATCH` angesprochen [19]:

GET: Die `GET`-Methode dient dazu, Ressourcen von einem Server abzufragen. Sie verändert den Zustand der API nicht, sondern liefert ausschließlich die angeforderten Informationen zurück. Anwendungsfälle sind oft das Abrufen einer einzelnen Ressource, wie bspw. einer bestimmten Schule, oder einer Liste von Ressourcen, etwa aller Fragebögen.

POST: Mit `POST`-Anfragen wird eine neue Ressource erstellt. Der Client sendet dabei Daten an den Server, der diese in einer neuen Ressource speichert. Beispielsweise kann durch eine `POST`-Anfrage ein neuer Fragebogen erzeugt werden. Da hierbei der Zustand der API verändert wird, gilt `POST` nicht als sicher in dem Sinne, dass wiederholte Anfragen zu Duplikaten führen können.

PUT: Die Methode `PUT` wird verwendet, um eine bestehende Ressource vollständig zu ersetzen. Dabei sendet der Client die aktualisierte Ressource an den Server; eine

Identifizierung über einen Schlüssel ist dabei unentbehrlich. Falls die Ressource noch nicht existiert, wird sie angelegt, ansonsten wird sie ersetzt.

PATCH: `PATCH` dient dazu, eine Ressource zu ändern; allerdings müssen nicht, wie bei `PUT`, alle Felder angegeben werden, sondern nur die zu ändernde. Denkbare Anwendungsfälle sind vorliegend beispielsweise die Änderung einer eingezeichneten Route. `PATCH` ist nicht Teil der ursprünglichen HTTP/1.1-Methoden und wird deshalb besonders in älteren System nicht immer unterstützt [14].

DELETE: Mit `DELETE` wird eine bestehende Ressource dauerhaft gelöscht.

Die Methoden `GET`, `PUT` und `DELETE` gelten als *idempotent*, das heißt, dass wiederholte Anfragen den Zustand der API nicht verändern [19]. Bei `PATCH` hängt dies von der Implementierung ab [14].

Der wohl wichtigste Aspekt ist aber die Zustandslosigkeit einer REST-API: Eine Anfrage enthält alle nötigen Informationen, um sie auszuführen, beispielsweise im HTTP-Body der Anfrage oder in einer parametrisierten URI. Auf diese Weise bleibt die REST-API stets unabhängig vom Client, wobei es keine Rolle spielt, ob es sich um eine Webanwendung, ein Mobilgerät oder eine andere API handelt [18].

REST ist nicht auf ein spezielles Datenformat beschränkt. Häufig benutzt werden JSON oder XML, wobei JSON mittlerweile am verbreitetsten ist, wohl aufgrund der nativen Unterstützung moderner Webframeworks wie NodeJS/TypeScript. Gerade diese Unabhängigkeit vom Datenformat führt dazu, dass die Programmiersprache der Komponenten keine Rolle spielt und auch verschieden sein kann, solange eine HTTP-Verbindung aufgebaut werden kann und das gewählte Datenformat verstanden wird.

2.3.3 ORM und Hibernate

ORM steht für *Object Relational Mapping* und ist eine Technologie, bei der Tabellen einer Datenbank als Klasse und Einträge der Tabelle als Objekte dieser Klasse betrachtet werden. Auf diese Weise können Datenbankinhalte direkt in der Programmiersprache verarbeitet werden, ohne dass dafür SQL-Abfragen von Hand formuliert werden müssen [3].

Ein einfaches Beispiel für die Funktionsweise eines ORM zeigt Abbildung 2.2. Links ist eine Java-Klasse `User` dargestellt, die Attribute für eine Benutzer-ID, einen Benutzernamen und ein Passwort enthält. Rechts ist die zugehörige Datenbanktabelle abgebildet, die mit passenden Spalten diese Attribute speichert. Ein ORM-Framework übernimmt die Abbildung zwischen beiden Repräsentationen, sodass Änderungen an der Klasse unmittelbar in den gespeicherten Datensätzen durchschlagen und umgekehrt.

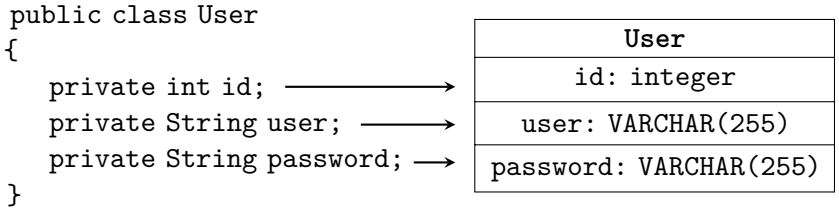

Abbildung 2.2: Zusammenhang zwischen Klasse und Tabelle.

Hintergrund ist ein Problem, das in der Literatur als *impedance mismatch* bezeichnet wird. Damit ist gemeint, dass sich die Prinzipien der Datenhaltung von objektorientierter Programmierung, also die Kapselung in Klassen, nicht ohne weiteres auf relationale Datenbanken übertragen werden können. Der Grund dafür liegt in den beiden Technologien

zu Grunde liegenden Prinzipien: Objektorientierung basiert auf (von Softwareentwicklern) ausgedachten Prinzipien, während relationale Datenbanken auf dem mathematischen Fundament der relationalen Algebra stehen. Ein ORM versucht diese Differenz zu überbrücken [4, 22].

Verbreitete ORM (wie Hibernate in Java oder Prisma in NodeJS) kümmern sich aber nicht nur um die Tabellen, sondern auch um die Beziehungen (*One-To-One*, *One-To-Many*, *Many-To-Many*) der Tabellen untereinander. Transaktionen, Lazy Loading oder das Erstellen/Ändern von Tabellen werden ebenfalls oft unterstützt. Auf diese Weise wird die Logik der Datenhaltung (bzw. der Zugriff darauf) von der Geschäftslogik der Anwendung getrennt, was insgesamt die Struktur vereinfacht und zu einer besseren Wartbarkeit führt [8, 22].

Die Verwendung eines ORM bringt verschiedene Vorteile mit sich. Projekte können schneller realisiert werden, da die ggf. mühsame Erstellung von SQL-Anweisungen entfällt. Durch die Abstrahierung der Datenhaltung ist es auch nicht relevant, welches konkrete relationale Datenbanksystem benutzt wird; das ORM kann sich (je nach Verfügbarkeit) an viele verschiedene relationale Datenbanksysteme anpassen, ohne dass sich die Anwendungslogik ändert. Die bereits erwähnte Trennung von Datenhaltung und Geschäftslogik ist ebenfalls ein Vorteil [8].

Als Nachteil kann gesehen werden, dass sehr komplexe Operationen durch ein ORM oft in ineffiziente SQL-Anweisungen übersetzt und ausgeführt werden. Daher wird hier häufig ein gemischter Ansatz gewählt, bei dem komplexe Operationen direkt in SQL geschrieben werden und das ORM nur zur Weiterleitung zu benutzen [8].

Ein verbreitetes ORM für Java ist die Hibernate-Bibliothek. Es unterstützt alle oben genannten Funktionen, ist sehr gut dokumentiert und leicht anzuwenden. Die oben gezeigte `User`-Klasse würde in Hibernate wie in Listing 2.1 gezeigt aussehen.

```
@Entity
public class User
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(unique = true)
    private String user;

    private String password;
}
```

Listing 2.1: Eine Entitätsklasse in Hibernate.

Hibernate funktioniert grundsätzlich über Annotationen. Die `@Entity`-Annotierung markiert die Klasse zunächst als Entität, für die eine Tabelle in der Datenbank vorgesehen ist. Die `@Id`-Annotation benennt das Attribut `id` (bzw. die dazu passende Spalte in der Tabelle) als Primärschlüssel; zusammen mit der darauf folgenden Annotation `@GeneratedValue` wird eine Spalte mit automatischer Zählung (Auto-Increment) erstellt. Schließlich markiert die `@Column`-Annotation zusammen mit der gegebenen Option die Spalte für den Benutzernamen als einzigartig; auch solche Einschränkungen (engl. *constraint*) werden von Hibernate beachtet und verwaltet.

2.3.4 GIS

Geoinformationssysteme (GIS) sind computergestützte Systeme zur Erfassung, Verwaltung, Analyse und Visualisierung von Daten, die einen Bezug zu einem bestimmten Ort

oder Raum haben. Neben diesen Ortsinformationen (bspw. Koordinaten, Adressen oder Flächen) werden auch Informationen der Objekte verwaltet, etwa die Art einer Straße, die Bevölkerungszahl eines Stadtteils oder die Geschwindigkeit auf einem Straßenabschnitt. Diese Kombination aus Ortsangabe und zusätzlichen Attributen ermöglicht es, komplexe Fragestellungen zu bearbeiten, bei denen sowohl die Position von Objekten als auch deren Eigenschaften eine Rolle spielen [33, 57].

Ein GIS beruht auf zwei verschiedenen Arten von Daten; dabei wird zwischen Vektordaten und Rasterdaten unterschieden [32]:

Vektordaten: Vektordaten beschreiben geografische Objekte durch einfache geometrische Formen: Punkte stehen zum Beispiel für einzelne Standorte wie Schulen, Linien für Verbindungen wie Straßen oder Radwege; Flächen für abgegrenzte Gebiete wie Wälder, Verkehrsinseln oder Bahnanlagen.

Rasterdaten: Rasterdaten bestehen aus einem gleichmäßigen Gitter von Zellen (Pixeln). Jede Zelle enthält einen Wert, der eine bestimmte Eigenschaft beschreibt, etwa die Höhe über dem Meeresspiegel, die durchschnittliche Verkehrsdichte oder die Temperatur an diesem Ort.

Ein wichtiger Punkt ist die Projektion. Da die Erde (annähernd) eine Kugel ist, müssen diese Kugelkoordinaten auf eine zweidimensionale Karte übertragen werden. Dazu gibt es verschiedene mathematische Abbildungen, wobei die verbreitetste die Mercator-Projektion ist; auf ihr basieren heute gängige, rechteckige Weltkarten [5]. Je nach Anwendungsfall kann auch eine andere Art der Projektion genutzt werden; für die genaue Berechnung von Abständen und Flächen ist eine projektionstreue Abbildung nötig.

Die Anwendungsmöglichkeiten von GIS sind vielfältig. Beispielsweise können Overlay-Analysen erstellt werden, bei der mehrere Kartenebenen übereinander gelegt werden, um Informationen zu verknüpfen. In der Verkehrsplanung können Netzwerkanalysen, wie die genaue Abmessungen von Straßen oder Routen, durchgeführt werden.

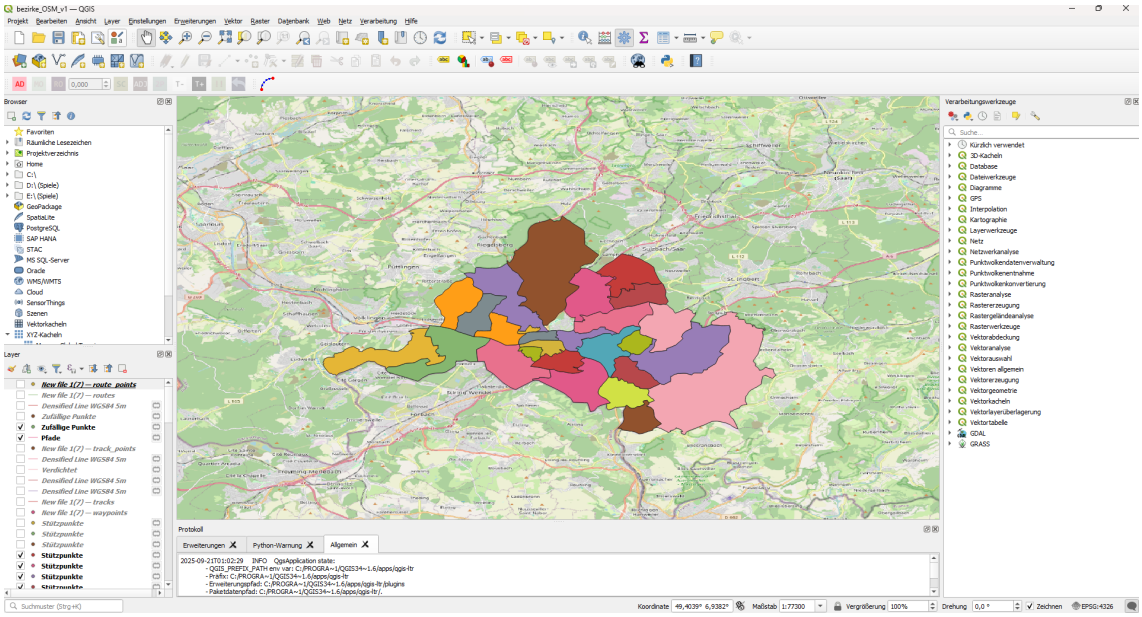

Abbildung 2.3: QGIS mit manuell eingezeichneten Schulbezirken.

Neben proprietären Lösungen wie ArcGIS existieren zahlreiche Open-Source-Systeme, die sich in den letzten Jahren etabliert haben. PostGIS ist eine Erweiterung des relationalen

Datenbanksystems PostgreSQL, die die Speicherung, Abfrage und Analyse räumlicher Daten in einer Datenbank ermöglicht. Mit QGIS (vgl. Abbildung 2.3) existiert eine freie Anwendung für den Desktop, die tiefgehende Analysen und Werkzeuge bereitstellt [21, 48].

2.3.5 OpenStreetMap

OpenStreetMap (OSM) ist ein freies, kollaboratives Projekt zur Erstellung einer frei verfügbaren, weltweiten Datenbank für Geoinformationen. Das Projekt wurde 2004 in Großbritannien ins Leben gerufen mit dem Ziel, eine Alternative zu den bis dahin bestehenden proprietären GIS-Datenbanken zu schaffen. Der wesentliche Unterschied zu bestehenden GIS-Anbietern bestehen darin, dass jedermann mitarbeiten und Daten einpflegen kann.

Durch die Vielfältigkeit der OSM-Community gehen verschiedenste Arten von Datensätzen in das Projekt ein. GPS-Aufzeichnungen, Luftbilder, (gemeinfreies) amtliches Kartenmaterial oder einfach die Ortskenntnis der Benutzer. So entsteht ein sehr detailliertes Bild, insbesondere in den Städten werden Neuerungen und Änderungen sehr schnell übernommen, teilweise schneller als bei kommerziellen Anbietern. Hier liegt aber gleichzeitig auch eine Schwäche von OSM: In weniger besiedelten Gebieten oder kleineren Städten kann das verfügbare Datenmaterial dünner sein als bei anderen GIS-Systemen.

Technisch basiert OSM auf einem Datenmodell, wie es im vorhergehenden Abschnitt vorgestellt wurde. Dabei werden Daten in drei Typen unterteilt:

Nodes: Georeferenzierte Punkte, die einzelne Objekte wie Bäume, Gebäude, Haltestellen oder Straßen repräsentieren.

Ways: Nodes können zu Ways verbunden werden. Solche Linien können mit genügend Punkten Straßen, Flüsse oder Flächen darstellen.

Relation: Verbindet Nodes und Ways mittels logischer Zusammenhänge, beispielsweise den Geschwindigkeitslimits in einem Stadtteil.

Durch ein System von Tags können diesen Objekten Informationen hinzugefügt werden, in Form von Schlüssel-Wert-Paaren. So beschreibt `kerb=lowered` bei einem Straßenübergang beispielsweise, dass der Bordstein abgesenkt ist, oder `maxspeed=50`, dass die Höchstgeschwindigkeit 50 Kilometer pro Stunde beträgt, wie auf Abbildung 2.4 zu erkennen. Die Art der Informationen hängt also immer von dem abgefragten Objekt ab und ist keinesfalls konsistent. Auch kann von der Existenz eines Wertes nicht auf die Existenz des selben Wertes bei einem gleichartigen Objekt geschlossen werden. Daher gibt es in der OSM-Community aktive Diskussionsprozesse, die in einem Wiki als sog. Map Features niedergeschrieben werden. Auf diese Weise wird versucht, eine Konsistenz zwischen den Einträgen sicherzustellen [35].

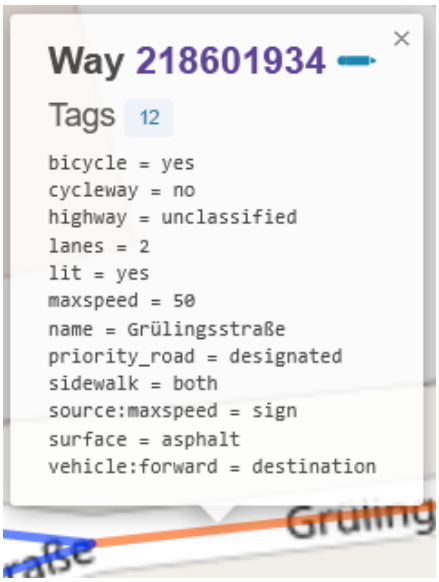

Abbildung 2.4: Tags eines Straßensegments.

3 Menschenzentrierte Anforderungsanalyse

Die menschenzentrierte Anforderungsanalyse dient der systematischen Erhebung und Berücksichtigung der Bedürfnisse und Anforderungen aller relevanten Nutzergruppen im Kontext der Entwicklung eines digitalen Schulwegplaners.

3.1 Erhebung

Das vorliegende Projekt entstand in Zusammenarbeit mit dem Ministerium für Umwelt, Klima, Mobilität, Agrar und Verbraucherschutz des Saarlandes als federführende Institution und der Landeshauptstadt Saarbrücken in beratender Funktion. Zusammen mit den Projektpartnern wurden in mehreren Treffen Anforderungen ausgelotet, Wünsche aufgenommen, die bestehenden Arbeitsprozesse erörtert und nach Verbesserungspotenzial durchleuchtet.

- Bei einem ersten Treffen im Januar im Rathaus der Landeshauptstadt Saarbrücken wurden zusammen mit Mitarbeitern der Stadt, Repräsentanten des Ministeriums und politischen Entscheidungsträgern der Kommune das aktuelle Problem der Schulwegplanung erörtert. Dabei wurde insbesondere der immense Aufwand der papierbasierten Erhebung und Verarbeitung hervorgehoben.
- Zudem wurde an einer Begehung mit dem VCD Saar e.V. für die Grundschule Hilschbach-Walpershofen in Riegelsberg teilgenommen. Dabei wurde demonstriert, wie eine Begehung generell abläuft und die Abläufe mit Mitarbeitern der Schule besprochen.
- Bei einem anschließenden runden Tisch an der HTW des Saarlandes Ende April wurden den mit der Schulwegplanung betrauten Mitarbeitern der Stadt der aktuelle Stand besprochen und erste Entwürfe gezeigt. Dabei wurden auch neue Anforderungen erhoben und der zu dem Zeitpunkt bestehende Funktionsumfang demonstriert.
- Das letzte Treffen fand mit Mitarbeitern des Ministerium für Umwelt, Klima, Mobilität, Agrar und Verbraucherschutz statt, ebenfalls an der HTW des Saarlandes. Dabei wurde ebenfalls eine Demonstration gezeigt und die weitere Projektplanung besprochen, allerdings keine weiteren Anforderungen erhoben.

Die nun folgende Anforderungsanalyse ist auf Basis der bei diesen Treffen gesammelten Informationen entstanden.

3.2 Personas

Im Rahmen einer menschenzentrierten Anforderungsanalyse hat sich der Einsatz sogenannter *Personas* etabliert. Bei Personas handelt es sich um fiktive Repräsentationen tatsächlicher Nutzergruppen, die alle ihre eigenen Bedürfnisse, Ziele und Ziel der Nutzung eines Produktes mit sich bringen; zusätzlich kann, falls relevant, auch die Gefühlswelt der

Benutzer in die Personas integriert werden. Sinn und Zweck dieser Personas ist es, verschiedene, reale Benutzer in einer zusammenfassenden Form abzubilden und auf ihnen die Kernbedürfnisse zu extrahieren, um ein benutzerfreundliches System zu gestalten, das den realen Benutzern bei der Problemlösung hilft. Zudem sind Personas besonders dazu geeignet, in interdisziplinären Teams eine gemeinsame Kommunikation zu schaffen, die unabhängig von fachspezifischen Modellierungssprachen oder -modellen verwendet werden kann [10, 24, 41].

Wichtig für das Verständnis von Personas ist die Abgrenzung zu sogenannten Marktsegmenten. Marktsegmente sind abgegrenzte Gruppen potenzieller Kunden, die ähnliche Bedürfnisse, Merkmale und Verhaltensweisen aufweisen und daher auf vergleichbare Weise auf Marketingmaßnahmen oder Produktanpassungen reagieren. Häufig werden diese Gruppen auf Basis demographischer Daten besetzt, also dem Alter, der Kaufkraft, dem Bildungsstand und weiterer, häufig benutzter Metriken. Personas hingegen spiegeln vorrangig Verhalten, Denkweisen, Fähigkeiten und konkrete Nutzungsszenarien. Etwas abstrakter formuliert: Marktsegmente sind auf der Makroebene angesiedelt, während Personas eher auf der Mikroebene ansetzen und damit feiner und detaillierter gehalten sind. Zwar können Merkmale beider Konzepte auch im jeweils anderen vorkommen, es lässt sich jedoch immer der Unterschied zwischen „grober“ und „feiner“ Klassifizierung ziehen [24, 30].

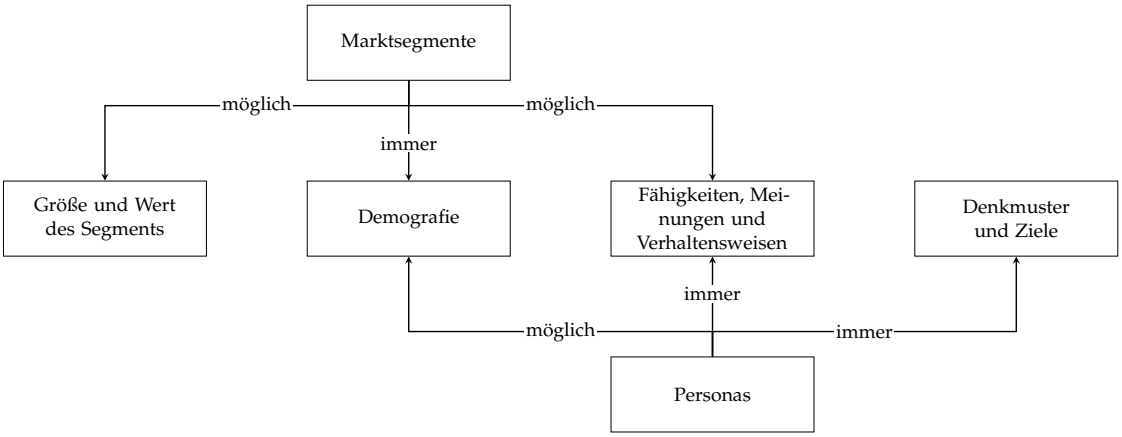

Abbildung 3.1: Überschneidungen von Marktsegmenten und Personas nach Goodwin, S. 237.

Abbildung 3.1 zeigt diese Überschneidungen; es ist zu erkennen, dass „die Überschneidung variieren kann und es möglich ist, beide Werkzeuge in die Richtung des jeweils anderen zu bewegen, ohne dass sie jemals austauschbar werden“ [24, S. 237].

Im Rahmen einer Schulwegplanung gibt es verschiedene Akteure mit eigenen Aufgaben, Anforderungen und Bedürfnissen, beispielsweise Kinder und Eltern, Planer oder das Schulpersonal, das mit der Schulwegplanung betraut ist. Durch die Personas wird es ermöglicht, die verschiedenen Bedürfnisse zu erfassen und im Rahmen des weiteren Entwicklungsprozesses zu verwenden.

Die Auswahl der Personas erfolgte so, dass wesentliche Rollen innerhalb des Prozesses abgebildet werden:

- Der Schulwegplaner, der für die Analyse der Daten und die Erstellung der Schulwegpläne zuständig ist.
- Ein Schüler, der täglich den Schulweg nutzt und somit unmittelbar von der Sicherheit dessen betroffen ist.

- Ein Elternteil, der den Schulweg seines Kindes als nicht sicher erachtet und das Kind deswegen immer öfter zur Schule fährt.
- Ein Mitarbeiter der Schule, der als Bindeglied zwischen der Planungsstelle und den Schülern steht.

Diese vier Personas dienen im weiteren Verlauf der Anforderungsanalyse als Bezugsrahmen für die Ableitung und Priorisierung funktionaler und nicht-funktionaler Anforderungen.

3.2.1 Persona A: Schulwegplaner

- **Name:** Andreas Beyrich
- **Alter:** 48
- **Beruf:** Verwaltungsfachangesteller in der Schulbehörde eines Landkreises

Andreas Beyrich ist Verwaltungsfachangesteller in der Schulbehörde und, neben seinen eigentlichen Aufgaben, zusätzlich mit der Entwicklung von Schulwegplänen betraut. Aufgrund seiner langjährigen Berufserfahrung ist er vertraut mit der Bedienung gängiger Software auf seinem Desktop-PC, die in der Verwaltung genutzt wird. Allerdings bereiten ihm neue Technologien häufig Einstiegsprobleme, besonders die Gewöhnung an nicht vertraute grafische Oberflächen und die Einarbeitung in komplizierte Prozesse. Seine Vision einer guten Software ist, dass er durch wenige Klicks sofort ein Ergebnis bekommt, das er einfach interpretieren kann. Von Geoinformationssystemen (GIS) versteht Andreas nur sehr wenig und findet diese Programme viel zu aufgebläht und komplex. Obwohl ihn die zeitintensive Arbeit an den Schulwegplänen von seinen eigentlichen Aufgaben in der Behörde ablenkt, ist er motiviert, den Schulkindern in seinem Zuständigkeitsbereich einen sicheren Schulwegplan bereitzustellen, insbesondere da er selbst Vater zweier Töchter im Grundschulalter ist.

Seine Aufgaben als Schulwegplaner umfassen zunächst, zusammen mit der Schule den Bedarf eines Schulwegplanes auszuloten. Dabei stützt er sich auf mündliche oder schriftliche Aussagen der Schulangestellten und die Meinung herbeigerufener Experten für Verkehrssicherheit, mit denen er Begehungen durchführt, um mögliche Gefahrenstellen im Schulbezirk zu identifizieren. Zusätzlich muss er von Hand die auf Papier durchgeführten Schülerumfragen auswerten; dies umfasst nicht nur die Fragen, sondern gerade auch eine Karte des individuellen Schulweges, der erfasst werden muss, um Hotspots in den Schulwegen der Kinder zu identifizieren. Außerdem berücksichtigt er bei der Planung Gegebenheiten vor Ort wie Querungshilfen (Ampeln, Verkehrsinseln), Beleuchtung oder Blindenleitlinien.

Andreas wünscht sich in erster Linie, dass die Arbeit mit Papier weniger wird oder sogar ganz entfällt, da der größte Zeitfaktor die Auswertung der Umfragen ist. Auch ist es ihm wichtig, alles in einem System gebündelt zu haben, damit er nicht mehr mit verschiedenen Tools die Auswertung der Umfragen und die Gestaltung der Pläne durchführen muss. Er möchte sich auch nicht mit den komplexen Funktionen eines GIS beschäftigen müssen, sondern seine Arbeit schnell und effizient erledigen. Idealerweise können sowohl Geh- als auch Radwege visualisiert werden. Andreas wünscht sich außerdem eine übersichtliche Darstellung der gelaufenen Schulwege der Kinder.

Andreas steht für eine Nutzergruppe mit geringer GIS-Erfahrung, die eine einfache, integrierte Lösung benötigt, um Schulwegpläne effizient zu erstellen. Seine Anforderungen betreffen insbesondere die Reduzierung manueller Datenerfassung und die Vereinfachung der Auswertung.

3.2.2 Persona B: Schülerin

- **Name:** Amélie Schönthal
- **Alter:** 10
- **Beruf:** Grundschülerin (4. Klasse)

Amélie geht seit Beginn ihrer Grundschulzeit zu Fuß oder fährt mit dem Fahrrad zur Schule. Ihr Schulweg ist etwa zwei Kilometer lang und führt über eine stark befahrene innerörtliche Bundesstraße, die sie überqueren muss. Eine Fußgängerampel an einer wichtigen Querungsstelle ist aufgrund einer langfristigen Baustelle immer wieder außer Betrieb; außerdem weichen aufgrund der Baustelle und damit verbundener Staus immer mehr Autofahrer auf Nebenstraßen aus. Auf dem Fahrrad nutzt Amélie den Gehweg, obwohl ein benutzungspflichtiger Radweg auf der Straße vorgesehen ist, da sie sich auf diesen unsicher fühlt.

Amélie nimmt den Schulweg als alltägliche, aber teils riskante Aufgabe wahr. Situationen im Straßenverkehr wirken für sie oft unübersichtlich, insbesondere im Bereich der Baustelle und bei vielen Autos vor der Schule. Die immer stärker befahrenen Nebenstraßen vereinfachen die Situation auch nicht. Sie kann selbst am besten schildern, wo sie sich unsicher fühlt, und trägt damit wertvolle Informationen zur Gefährdungsanalyse bei.

Sie wünscht sich, dass sie einfach mitteilen kann, welche Probleme ihr jeden Tag auf dem Schulweg begegnen. Als Besitzerin eines einfachen Smartphones fände sie es gut, wenn sie damit ihren Schulweg beschreiben könnte und auch Bescheid geben kann, wenn ihr etwas gefährlich erscheint. Allerdings ist ihr eine einfache Bedienung ebenfalls wichtig, da sie in der Vergangenheit schon schlechte Erfahrungen mit übermäßig vollen Apps gemacht hat. Da ihr Smartphone auch bereits ein älteres Modell ist, sollte die App einfach gehalten sein, damit sie schnell und zuverlässig läuft.

Diese Persona repräsentiert die direkte Nutzergruppe der Kinder, die auf dem Schulweg tagtäglich Risiken ausgesetzt sind. Ihre Perspektive ist entscheidend, um Gefahrenstellen identifizieren zu können, die Erwachsenen möglicherweise entgehen.

3.2.3 Persona C: Elternteil

- **Name:** Jasmin Schönthal
- **Alter:** 39
- **Beruf:** Industriekauffrau

Jasmin ist Mutter von Amélie und begleitet die Entwicklung ihrer Selbstständigkeit im Straßenverkehr. Aufgrund zunehmender Zwischenfälle und Unfälle mit Kindern im Straßenverkehr hat sie entschieden, ihre Tochter häufig mit dem Auto direkt zur Schule zu bringen. Sie nimmt dabei in Kauf, dass vor dem Schulgebäude morgens chaotische Situationen durch viele sog. Elterntaxis entstehen. Immer öfter stellt sie dabei fest, dass ihr zwar Gefahrenstellen auffallen, aber sie nicht weiß, an wen sie sich wenden kann.

Ihr Wunsch ist es, konkrete und belastbare Daten zu Gefahrenstellen auf dem Schulweg erfassen und weitergeben zu können – nicht nur, um Amélies Sicherheit zu erhöhen, sondern auch, um einen Beitrag für alle Kinder im Schulbezirk zu leisten. Zwar führt die Schule ihrer Tochter Umfragen auf Papier durch, was jedoch niemals so genau sein kann wie vor Ort erhobene Daten. Sie wünscht sich eine Software, mit der Gefahrenstellen unkompliziert gemeldet werden können. Dabei legt sie großen Wert auf einfache Bedienung der Software, die sich ohne zusätzlichen Stress in den Alltag integrieren lässt.

Diese Persona steht für die Perspektive der Eltern, die sich sowohl um die Sicherheit ihrer Kinder im Straßenverkehr sorgen als auch aktiv an Verbesserungen für Kinder mitwirken wollen.

3.2.4 Persona D: Schulmitarbeiter

- **Name:** Claudia Rupp
- **Alter:** 36
- **Beruf:** Lehrerin und Sicherheitsbeauftragte der Schule

Claudia Rupp arbeitet seit zehn Jahren an einer Grundschule und hat vor drei Jahren zusätzlich die Aufgabe der Sicherheitsbeauftragten übernommen. Neben dem Unterrichten engagiert sie sich für Unfallprävention und eine sichere Gestaltung des Schulumfelds. Sie ist die Ansprechpartnerin für Themen wie Verkehrsunterricht, Gefahrenstellen rund um die Schule und die Zusammenarbeit mit der Stadtverwaltung.

Besonders frustriert ist Claudia über die hohe Zahl sogenannter Elterntaxis, die morgens und mittags vor der Schule halten. Die enge Straße vor dem Schulgebäude ist für den Autoverkehr eigentlich ungeeignet, was regelmäßig zu unübersichtlichen Situationen und riskanten Manövern führt. Sie sieht hier eine konkrete Gefährdung für die Schüler, die zu Fuß oder mit dem Fahrrad kommen. Ein weiteres Problem ist die Durchführung der jährlich durchgeführten Schülerumfragen zum Schulweg. Diese werden noch in Papierform durchgeführt, was für Claudia einen erheblichen zeitlichen Mehraufwand bedeutet. Das mühsame Sortieren, Auszählen und Zusammenfassen der Antworten bindet Ressourcen, die sie lieber in konkrete Sicherheitsprojekte investieren würde.

Claudia wünscht sich ein Werkzeug, das Gefahrenstellen strukturiert und mit präzisen Standortangaben erfasst, um Gespräche mit der Stadt führen zu können. Auch die digitale Erfassung von Umfrageergebnissen oder Schulwegmeldungen wäre für sie ein enormer Fortschritt. Besonders hilfreich fände sie es, wenn sie auf einen Blick sehen könnte, wo sich die meisten Gefahrenpunkte häufen, um Präventionsmaßnahmen gezielter vorschlagen zu können.

Diese Persona steht für die Sichtweise der Schule auf die Schulwegsicherheit. Claudia ist die Schnittstelle zwischen Schülern, Eltern und Behörden und verfügt über einen direkten Einblick in die alltäglichen Herausforderungen. Für die Anforderungsanalyse ist ihre Perspektive relevant, da sie sowohl den Aufwand als auch die praktischen Probleme im Schulumfeld kennt.

3.3 User Needs

Die zuvor behandelten Personas stehen exemplarisch für typische Nutzer (und Nutznießer) einer Schulwegplanung. Darauf aufbauend lassen sich konkrete User Needs ableiten; diese umfassen die Bedürfnisse, Erwartungen und Probleme, die in einer menschenzentrierten Anforderungsanalyse sowie in der späteren Konzeption und Implementierung berücksichtigt werden müssen. User Needs sind also eine verdichtete Darstellung der Analyseergebnisse, die anschließend in funktionale und nicht-funktionale Anforderungen übertragen werden können. Somit stellen sie das Bindeglied zwischen Personas und einer formalen Spezifikation eines Systems, indem sie beschreiben, *was* ein Benutzer erreichen möchte und *welche Bedingungen* dabei erreicht werden müssen [39, 44].

User Needs bestehen, neben der Beschreibung, aus einer Priorität und den Akzeptanzkriterien. Die Priorität kann, muss aber nicht, durch die „MoSCoW“-Methode dargestellt

werden. Dabei werden den einzelnen Punkten „**M**ust have“, „**S**hould have“, „**C**ould have“ oder „**W**on't have“ zugeordnet (wobei letzteres in dieser Arbeit nicht benutzt wird). Hintergrund ist, dass Zahlensysteme verwirrend sein können und manchmal dazu tendieren, alles der höchsten Priorität (in der Regel 1) zuzuordnen, wodurch der Zweck einer Priorisierung ad absurdum geführt wird [36].

Bei Akzeptanzkriterien handelt es sich um Bedingungen, die erfüllt sein müssen, damit ein Produkt (oder ein Teil davon) vom Benutzer akzeptiert und benutzt wird. Demnach ist ein User Need erst dann befriedigt, wenn alle Akzeptanzkriterien erfüllt sind. Sie müssen präzise und ohne Mehrdeutigkeiten formuliert sein [2].

Schulwegplaner

Der Schulwegplaner möchte vor allem den manuellen Planungsaufwand reduzieren. Dafür ist es nötig, alle verfügbaren Daten (Stammdaten, Verkehrsdaten, Umfragedaten) auf einen Blick einsehen und nutzen zu können.

Nr.	User Need	Priorität	Akzeptanzkriterien
UN01	Reduktion manueller Datenerfassung	Must have	Umfragen und Routenaufzeichnungen können ohne Papier ausgewertet werden.
UN02	Arbeiten mit Karten ohne GIS-Kenntnisse	Must have	Auf einer Karte können Schulwege eingezeichnet und POI markiert werden.
UN03	Intuitives Benutzerinterface	Must have	Zentrale Funktionen benötigen nicht mehr als drei Klicks; selbsterklärende Symbole.
UN04	Einheitliches Interface für alle Arbeitsschritte	Must have	Erfassung, Auswertung und Visualisierung auf der selben Seite.
UN05	Anzeige von Verkehrsdaten	Must have	Es müssen Verkehrsdaten ohne externe Karten angezeigt werden.
UN06	Schulwege der Kinder	Must have	Die Schulwege der Kinder müssen übersichtlich dargestellt werden können.
UN07	Radwege visualisieren	Should have	Die Karte bietet eine Visualisierung von Radwegen an.
UN08	Export fertiger Pläne	Should have	Fertige Pläne lassen sich direkt als PDF exportieren bzw. drucken.
UN09	Kollaboratives Arbeiten	Could have	Mehrere Mitarbeiter können gleichzeitig an einem Plan arbeiten.

Tabelle 3.1: User Needs des Schulwegplaners.

Die abgeleiteten User Needs zeigen, dass für einen Schulwegplaner eine Vereinfachung der Prozesse im Vordergrund steht. Dies soll vor allem durch eine Reduzierung von manuellen Tätigkeiten erreicht werden. Unterstützend muss die Oberfläche benutzerfreundlich gestaltet und ohne Kenntnisse gängiger GIS-Systeme bedienbar sein. Alles sollte an einer Stelle übersichtlich zusammenlaufen, sodass ein Plan in nur wenigen Klicks erstellt wer-

den kann. Ergänzend kommen noch die Visualisierung von Radwegen und der Export fertiger Pläne hinzu.

Schüler

Schüler nehmen den täglichen Schulweg unmittelbar wahr und können konkrete Unsicherheiten oder Gefahrenstellen benennen. Ihre Perspektive ist wichtig, um Probleme zu erkennen, die Erwachsenen häufig entgehen.

Nr.	User Need	Priorität	Akzeptanzkriterien
UN10	Unsichere Stellen auf dem Schulweg melden	Must have	Kinder können Gefahrenstellen dokumentieren, ohne technisches Vorwissen zu benötigen.
UN11	Aufzeichnung der Route	Must have	Kinder können ihren Schulweg aufzeichnen.
UN12	Verständliche und altersgerechte Bedienung	Must have	Die Benutzeroberfläche ist intuitiv und selbsterklärend für Grundschulkinder.
UN13	Performance	Should have	Das System sollte auch auf älteren Systemen flüssig laufen

Tabelle 3.2: User Needs der Schülerin.

Die User Needs der Kinder verdeutlichen, dass ihre Rolle vor allem in der direkten Meldung von Gefahrenstellen liegt. Damit ihre Angaben zuverlässig in die Analyse einfließen können, ist eine einfache, altersgerechte Bedienbarkeit entscheidend. So wird sichergestellt, dass die Kinder ohne Hürden ihre Erfahrungen einbringen können.

Eltern

Eltern tragen Verantwortung für die Sicherheit ihrer Kinder und möchten aktiv zu Verbesserungen beitragen. Sie benötigen einfache Möglichkeiten, Gefahrenstellen zu melden und verlässliche Informationen zu erhalten, um die Sicherheit auf dem Schulweg nachhaltig zu erhöhen.

Nr.	User Need	Priorität	Akzeptanzkriterien
UN14	Gefahrenstellen melden	Must have	Eltern können Beobachtungen zum Schulweg erfassen.
UN15	Einfache Integration in den Alltag	Must have	Die Nutzung der Software ist ohne hohen Zeitaufwand oder komplizierte Abläufe möglich.
UN16	Beitrag zur Verbesserung der Schulwegsicherheit	Could have	Eltern sehen, dass ihre Angaben nicht nur dem eigenen Kind, sondern allen Kindern zugutekommen.

Tabelle 3.3: User Needs des Elternteils.

Bei den Eltern steht im Vordergrund, Risiken melden zu können und einen Beitrag zur allgemeinen Sicherheit zu leisten. Dafür müssen Gefahrenstellen unkompliziert dokumentiert werden können, sodass die Nutzung in den Alltag integrierbar ist. Zudem besteht der Wunsch, dass eigene Angaben allen betroffenen Kindern im Schulbezirk zugute kommen.

Schulmitarbeiter

Schulmitarbeiter benötigen eine kompakte, leicht zugängliche Übersicht über die von Eltern und Schülern gemeldeten Gefahrenstellen und die daraus erstellten Auswertungen. Diese Informationen unterstützen sie dabei, präventive Maßnahmen zu planen und die Kommunikation mit Eltern, Schulleitung und Verwaltung zu erleichtern, ohne zusätzlichen manuellen Arbeitsaufwand.

Nr.	User Need	Priorität	Akzeptanzkriterien
UN17	Einsicht in fertige Auswertungen	Must have	Schulmitarbeiter können die von der Schulverwaltung erstellten Pläne abrufen.
UN18	Niedriger Nutzungsaufwand	Must have	Der Zugriff auf Informationen unkompliziert möglich; wenige Klicks genügen.
UN19	Überblick über Umfrageergebnisse	Should have	Die Schulmitarbeiter haben Zugriff auf die Ergebnisse von Umfragen, Begehungen und Gefahrenstellen.
UN20	Verteilung der Pläne	Should have	Die Pläne sollen einfach exportierbar sein.

Tabelle 3.4: User Needs einer Schulmitarbeiterin.

Damit wird die Rolle der Schulmitarbeiter vor allem in der Nutzung und Weitergabe von Ergebnissen sichtbar: Sie profitieren von den Informationen, ohne selbst aktiv Daten erfassen zu müssen.

Querschnittliche bzw. nicht-funktionale Bedürfnisse

Neben den personaspezifischen Anforderungen lassen sich mehrere übergreifende Bedürfnisse identifizieren, die für den Erfolg der Anwendung grundlegend sind.

Nr.	User Need	Priorität	Akzeptanzkriterien
UN21	Anonymität und Datenschutz	Must have	Die erhobenen Daten sind anonymisiert; es dürfen keine Rückschlüsse auf einzelne Personen möglich sein.
UN22	Sicherer Zugriff für Planer	Must have	Nur autorisierte Mitarbeiter können Pläne bearbeiten oder Auswertungen einsehen; die mobile Datenerfassung bleibt für alle offen zugänglich.
UN23	Intuitive Bedienung	Must have	Neue Nutzer verstehen die Kernfunktionen innerhalb von zehn Minuten ohne Schulung; Menüführung und Icons sind selbsterklärend.
UN24	Schnelle Reaktionszeiten	Should have	Alle Nutzeraktionen reagieren ohne spürbare Verzögerung
UN25	Präzise Kartenanzeige	Should have	Aufgezeichnete und eingezeichnete Schulwege sind auf der Karte sehr genau darstellbar, damit Routen eindeutig zuzuordnen sind.
UN26	Einfache Bereitstellung	Should have	Das Gesamtsystem kann als Paket installiert werden; Installation ist in wenigen Minuten möglich.
UN27	Einfacher Datenimport / Wartbarkeit	Could have	Neue Schulen oder geänderte Schulbezirke können unkompliziert eingepflegt werden, ohne dass hoher technischer Aufwand nötig ist.

Tabelle 3.5: Querschnittliche bzw. nicht funktionale User Needs.

3.4 Funktionale Anforderungen

Während die User Needs Teil der menschenzentrierten Anforderungsanalyse sind, müssen nun funktionale Anforderungen aus technischer Sicht aus ihnen formuliert werden.

Funktionale Anforderungen beschreiben, welche Leistungen oder Funktionen ein System erbringen muss, um die zuvor identifizierten User Needs zu erfüllen. Sie legen fest, was das System tun soll, ohne jedoch zwingend zu definieren, wie die Umsetzung erfolgt. Sie können definieren, bei welchen Eingaben das System welche Ausgaben liefern soll und wie es sich verhalten soll. In der Norm ISO/IEC/IEEE 29148:2018 werden funktionale Anforderungen als diejenigen verstanden, die den Betrieb des Systems betreffen und dessen Funktionen festlegen [28, 43, 47].

Zur besseren Übersicht werden die nun folgenden funktionalen Anforderungen nach den verschiedenen Nutzungskontexten aufgeteilt: den der kommunalen Mitarbeiter und den der Eltern und Kinder.

3.4.1 Nutzungskontext: Datenerhebung

FR01 - Teilnahme an Umfragen: Über die mobile Ansicht sollen Schüler und Eltern an bereitgestellten Umfragen teilnehmen können. Diese können verschiedene Fragetypen umfassen (z. B. Multiple Choice, Freitext, Bewertungsskalen). Alle Antworten werden erfasst und gespeichert.

Akzeptanzkriterien:

- Nutzer können eine Umfrage öffnen.
- Verschiedene Fragetypen (Multiple Choice, Freitext, Dropdown, etc.) sind beantwortbar.
- Abgeschlossene Umfragen werden gespeichert.

Löst: UN01
Priorität: Must have

FR02 - Erfassung von Gefahrenstellen: Nutzer sollen in der Lage sein, Gefahrenstellen im Straßenverkehr zu melden. Jede Meldung besteht aus einer Kartenposition, einer kurzen Beschreibung und optional einem Foto.

Akzeptanzkriterien:

- Nutzer können einen Punkt auf der Karte setzen.
- Eine Beschreibung ist anzugeben.
- Die Gefahrenstelle wird gespeichert und erscheint in der Übersicht für Planer.

Löst: UN10, UN14
Priorität: Must have

FR03 - Aufzeichnung des Schulwegs: Es soll möglich sein, den täglichen Schulweg zu erfassen. Dies kann entweder automatisch per GPS-Aufzeichnung oder manuell durch Einzeichnen auf einer Karte geschehen.

Akzeptanzkriterien:

- Nutzer können eine GPS-Aufzeichnung starten und beenden.
- Der aufgezeichnete Weg wird auf einer Karte angezeigt.

Löst: UN11
Priorität: Must have

FR04 - Benutzerfreundlichkeit auf mobilen Geräten: Die mobile Ansicht ist so zu gestalten, dass sie auf Smartphones und Tablets lauffähig ist und die Eingabe mit Touchbedienung intuitiv erfolgen kann.

Akzeptanzkriterien:

- Die Anwendung ist auf mobilen Endgeräten vollständig bedienbar.
- Inhalte sind auch auf kleineren Displays übersichtlich darstellbar.

Löst: UN12, UN13, UN15, UN24
Priorität: Should have

3.4.2 Nutzungskontext: Planung

Das Planungssystem ist die zentrale Ansicht des zu entwickelnden Gesamtsystems und bietet alle Funktionen, die für Planer wichtig sind.

FR05 - Anlage von Plänen: Das System muss es erlauben, verschiedene Pläne anzulegen und zu bearbeiten. So soll erlaubt werden, an mehreren Plänen gleichzeitig zu arbeiten. Die Pläne sollen sich jeweils auf einen Schulbezirk bzw. eine Schule beziehen. Zur weiteren Identifizierung von Plänen soll es möglich sein, den Plänen eigene Namen und ggf. ein Bild zuzuweisen.

Akzeptanzkriterien:

- Ein Benutzer kann mindestens einen neuen Plan anlegen.
- Mehrere Pläne können parallel gespeichert und bearbeitet werden.
- Jeder Plan muss einen Namen bekommen.
- Optional kann ein Bild hinzugefügt werden.
- Gespeicherte Pläne erscheinen in der Übersicht.

Löst: UN01, UN09
Priorität: Must have

FR06 - Übersicht: Alle aktuell angelegten Pläne können auf einer Hauptseite dargestellt werden. Den Gewohnheiten eines Benutzers folgend soll der zuletzt bearbeitete Plan als erstes angezeigt werden. Alle weiteren Pläne sollen übersichtlich mit allen sie beschreibenden Informationen dargestellt werden.

Akzeptanzkriterien:

- Auf der Hauptseite werden alle Pläne angezeigt.
- Der zuletzt bearbeitete Plan steht oben in der Liste.
- Jeder Plan zeigt Name, Bild (falls vorhanden) und Bezirksinformationen.
- Durch einen Klick öffnet sich die jeweilige Planungsansicht.

Löst: UN01, UN03, UN04
Priorität: Must have

FR07 - Planungsansicht: Die Planungsseite muss die Möglichkeit bieten, mit einer zum Schulbezirk passenden Landkarte zu arbeiten.

Akzeptanzkriterien:

- Die Planungsseite zeigt eine Landkarte, die den gewählten Schulbezirk abbildet.
- Die Karte erlaubt grundlegende Interaktion (Zoom, Verschieben).

Löst: UN02
Priorität: Must have

FR08 - Radwege: Auf der Planungsseite soll eine Möglichkeit geboten werden, Radwege zu visualisieren und mit ihnen zu arbeiten.

Akzeptanzkriterien:

- Radwege sind in der Karte erkennbar.
- Nutzer können Radwege ein- oder ausblenden.

- Radwege können für die Planung als Referenz genutzt werden.

Löst: UN07
Priorität: Should have

FR09 - Umfrageergebnisse: Die Ergebnisse der Schülerumfragen müssen dargestellt werden. Je nach Art der Umfrage können dabei Balkendiagramme, Kreisdiagramme oder einfache Textübersichten zur Anwendung kommen.

Akzeptanzkriterien:

- Umfrageergebnisse sind pro Frage darstellbar.
- Ergebnisse erscheinen in einem passenden Visualisierungstyp (Balken, Kreis, Text).
- Diagramme zeigen die korrekten Werte.

Löst: UN01, UN17, UN19
Priorität: Must have

FR10 - Routenaufzeichnungen: Die aufgezeichneten Routen der Kinder müssen visualisiert werden. Dabei sollen Häufungen von Routen ebenfalls visualisiert werden.

Akzeptanzkriterien:

- Aufgezeichnete Routen werde auf der Planungsansicht visualisiert.
- Häufungen von Routen, also wo viele Kinder herlaufen, können erkannt werden.
- Die aufgezeichneten Routen sind korrekt und weisen keine GPS-Fehler auf.

Löst: UN06, UN25
Priorität: Must have

FR11 - Verkehrsdaten: Auf einer Karte müssen alle relevanten Verkehrsdaten angezeigt werden. Dies umfasst mindestens Straßenübergänge, Bushaltestellen und Höchstgeschwindigkeiten.

Akzeptanzkriterien:

- Straßenübergänge, Bushaltestellen und Tempolimits sind sichtbar.
- Verkehrsdaten sind korrekt im Kartenausschnitt positioniert.
- Nutzer können Kartenebenen ein- oder ausblenden.

Löst: UN05
Priorität: Must have

FR12 - Einzeichnen von Routen: Es muss möglich sein, auf einer Karte Routen einzuzeichnen und zu speichern.

Akzeptanzkriterien:

- Nutzer können eine Route durch Punkte auf der Karte erstellen.
- Routen werden als Linie angezeigt.
- Eine erstellte Route kann gespeichert und wieder aufgerufen werden.

Löst: UN02
Priorität: Must have

FR13 - Export: Das System sollte in der Lage sein, fertige Pläne als PDF zu exportieren oder zum Druck bereitstellen.

Akzeptanzkriterien:

- Ein Plan kann als PDF-Datei exportiert werden.
- Exportierte Dateien enthalten alle relevanten Informationen (Name, Karte, Objekte).

Löst: UN08, UN20
Priorität: Should have

FR14 - POI: Es muss eine Möglichkeit geben, POI (Points of Interest) anzulegen. Eine Anlage besteht aus einem Namen und einer Bilddatei.

Akzeptanzkriterien:

- Ein Nutzer kann einen neuen POI anlegen.
- Jeder POI enthält einen Namen und ein Bild.
- Angelegte POIs sind in der Liste sichtbar.

Löst: UN02
Priorität: Must have

FR15 - Platzierung von POI: POI müssen auf einer Karte platziert werden können.

Akzeptanzkriterien:

- Nutzer können einen POI auf die Karte ziehen und durch Klick platzieren.
- Platzierte POIs sind an der richtigen Position sichtbar.
- POIs bleiben auch nach dem Speichern erhalten.

Löst: UN02
Priorität: Must have

3.5 Nicht-funktionale Anforderungen

Nachdem die funktionalen Anforderungen formuliert wurden, müssen nun die nicht-funktionalen Anforderungen abgeleitet werden.

Nicht-funktionale Anforderungen beziehen sich weniger auf die Funktionalität eines Systems, sondern mehr auf das Verhalten. Es setzt dem System Grenzen, innerhalb dessen es funktionieren muss. Dies können Anforderungen an Laufzeiten, Antwortzeiten, Leistungsfähigkeit, Präzision der Ergebnisse oder auch die Mehrsprachigkeit oder die Sicherheit eines Systems sein. Auch die Konformität gemäß einer Norm oder einer Rechtsnorm (bspw. der DSGVO) zählt zu den nicht-funktionalen Anforderungen [28, 47].

NR01 - Datenschutz: Die erforderlichen Daten werden von Kindern erhoben. Je nach Umfrage könnte das Alter abgefragt werden, durch die Aufzeichnung ist auch der Schulweg nachzuvollziehen und wird mit dem Umfrageergebnis verknüpft. Gerade wegen dieser besonders schützenswerten Benutzergruppe ist konsequenter Datenschutz von immenser Bedeutung. Es darf nicht möglich sein, von den Daten auf ein einzelnes Kind Rückschlüsse zu ziehen.

Akzeptanzkriterien:

- Alle erhobenen Daten sind anonymisiert gespeichert.

- Eine Identifizierung einzelner Kinder ist nicht möglich.
- Verknüpfungen (bspw. zwischen Schulweg und Umfrageergebnissen) sind möglich, lassen aber keinen Rückschluss auf die Kinder zu.

Löst: UN21
Priorität: Must have

NR02 - Sicherheit: Nur befugte Personen dürfen die erhobenen Daten einsehen, Statistiken sehen und die Pläne bearbeiten. Gleichzeitig muss sichergestellt werden, dass die Aufzeichnung der Wege und die Umfragen keinerlei Zugangsbeschränkung unterworfen ist, um den Prozess nicht unnötig zu verkomplizieren und damit von einer Nutzung abzuschrecken. Ein Login für Planer sollte maximal 24 Stunden gültig sein; eine Session für Umfrageteilnehmer maximal eine Woche.

Akzeptanzkriterien:

- Zugriff auf Pläne und Statistiken ist nur mit gültigem Login möglich.
- Sessions von Planern verfallen nach spätestens 24 Stunden.
- Sessions von Umfrageteilnehmern verfallen nach spätestens einer Woche.
- Umfragen, Gefahrenmeldung und Wegaufzeichnungen sind ohne Login möglich.

Löst: UN22
Priorität: Must have

NR03 - Benutzbarkeit: Neue Benutzer sollen die Kernfunktionalität innerhalb von zehn Minuten verstehen. Dies setzt ein konsistentes Design, selbsterklärende Icons und eine einfache Menüführung voraus. Es soll keine Schulung oder Einweisung nötig sein.

Akzeptanzkriterien:

- Nutzer können die Hauptfunktionen ohne Anleitung bedienen.
- Icons und Menüs sind selbsterklärend.
- Erste Bedienung benötigt weniger als zehn Minuten Einarbeitung.

Löst: UN18, UN23
Priorität: Must have

NR04 - Antwortzeit: Anfragen sollten ohne sichtbare Verzögerung beantwortet werden. Eine Ausnahme können komplexe Anfragen sein, die etwas mehr Rechenaufwand erfordern; hier sollte ein Limit von einer halben Sekunde gesetzt werden.

Akzeptanzkriterien:

- Standardanfragen werden ohne wahrnehmbare Verzögerung beantwortet.
- Komplexe Anfragen haben eine Antwortzeit von maximal 0,5 Sekunden.

Löst: UN24
Priorität: Should have

NR05 - Auflösung von Kartendetails: Eingezeichnete oder aufgezeichnete Routen sollten mit einer Auflösung von fünf Metern genau nachvollziehbar sein. So soll verhindert werden, dass Straßen als Gehwege interpretiert werden oder umgekehrt.

Akzeptanzkriterien:

- Alle Routen werden mit einer Genauigkeit von $\pm$ 5 Metern dargestellt.
- Kartendetails erlauben eine klare Unterscheidung zwischen Straßen und Gehwegen.

Löst: UN25
Priorität: Should have

NR06 - Deployment: Das Gesamtprojekt sollte als ein gesamtes Paket bereitgestellt werden können. Dies kann durch ein lauffähiges Docker-Image passieren oder mindestens durch die gemeinsame Bereitstellung von Front- und Backend. Eine einfache Textanleitung soll nur Installation genügen.

Akzeptanzkriterien:

- Das System ist als ein Paket verfügbar (bspw. Docker-Image).
- Installation erfordert nur wenige manuelle Schritte.
- Eine Textanleitung reicht aus, um das System in Betrieb zu nehmen.

Löst: UN26
Priorität: Should have

NR07 - Wartbarkeit: Neue Schulen oder veränderte Schulbezirke sollen ohne größeren technischen Aufwand eingepflegt werden können. Dieser Schritt eher für technisch versierte Nutzer oder Administratoren nach einer Einweisung gedacht.

Akzeptanzkriterien:

- Änderungen erfordern keinen Eingriff in den Quellcode.
- Ein geschulter Administrator kann die Pflege eigenständig durchführen.

Löst: UN27
Priorität: Could have

UN16 konnte nicht in eine Anforderung überführt werden. Der Beitrag zur Verbesserung der Schulwegsicherheit ergibt sich aus der Funktionalität des Gesamtsystems und hängt nicht von einer speziellen Funktionalität ab.

3.6 Stand der Technik

Zum Startzeitpunkt des Projektes im Oktober 2024 wurden bestehende Lösungen im Rahmen des Moduls PIM-PA analysiert. Bis zum September 2025 wurden diese Lösungen aber erweitert und überarbeitet, sodass es nun ähnliche Produkte gibt, die einige der herausgearbeiteten Anforderungen erfüllen.

schulwegcheck.de bietet ein Tool an, mit dem Wege aufgezeichnet werden können. Dabei ist dieses Projekt nicht nur auf die Schulwegplanung beschränkt, sondern richtet sich allgemein an Schüler als Unterstützung für diverse Projekte zum Thema Mobilität. Außerdem erlaubt es die Webseite, Pläne anzulegen, die gesammelten Daten anzusehen und auszuwerten und die Pläne zur Verfügung zu stellen. Über die App werden Routen aufgezeichnet und Gefahrenstellen gemeldet. Das Tool hat seit der Erstrecherche im Vorfeld zu der vorliegenden Arbeit einige Updates bekommen; eine Einsicht in konkrete Funktionen ist hier ohne Anmeldung nicht möglich [20].

Bei **gefahrenstellen.de** handelt es sich um einen Routenplaner der *Initiative für sichere Straßen GmbH*. Dort können Routen berechnet werden, bei denen Stellen mit einer einstellbaren Gefahrenbewertung vermieden werden können. Die Datenbasis dafür sind von

Benutzern eingetragene Gefahrenstellen. Die Webseite geht auf die *Initiative für sichere Straßen* zurück, eine Forschungsgruppe der RWTH Aachen, an der auch die Berechnung der Gefahrenbewertung entwickelt wurde [49, 50].

Eine Weiterentwicklung des vorgenannten Projektes ist **schulwege.de**. Auch dieses Projekt war zum Startzeitpunkt des vorliegenden Projektes und der anfänglichen Recherche noch nicht so ausgebaut wie es zum jetzigen Zeitpunkt. Das Projekt umfasst eine kostenpflichtige, digitale Planungsplattform. Es basiert auf der selben Datenbasis wie gefahrenstellen.de, erlaubt aber zusätzliches Einzeichnen von Schulwegen inklusive Bereitstellung für die Schüler und Eltern. Es erlaubt den Kindern und Eltern außerdem, Gefahrenstellen zu melden oder Umfragen auszufüllen. Eine Einsicht in das System ist hier ohne Zugang nicht möglich [49, 51].

Hamburg stellt auf dem Geoportal Hamburg mit dem **Schulinfosystem** einen Routenplaner zur Verfügung, mit dem Kinder ihren Schulweg zu einer auswählbaren Schule im Stadtgebiet Hamburgs berechnen können. Bei Auswahl einer Adresse wird die zuständige Grundschule des Bezirks als Zielort vorgeschlagen. Die berechnete Route ist die schnellste Route, Gefahrenstellen werden scheinbar nicht berücksichtigt [56].

Der **Schulwegplaner Baden-Württemberg** bietet eine Plattform, die vom Desktop aus für alle an der Planung beteiligten Benutzer zugänglich sind. Eine direkte Einsicht in das System ist nicht möglich, da dafür ein Schulzugang nötig ist. Allerdings gibt es Dokumentationen, die näheren Aufschluss über die Funktionalitäten geben. Zunächst tragen die Schüler ihren Schulweg am Desktop ein. Auf einer Karte wird der Schulweg eingezeichnet, mit zu einer Route verbundenen Punkten; dabei wird differenziert zwischen Gehwegen und Radwegen. Anschließend können noch Problemstellen erfasst werden. Auf der Ansicht für Planer werden die erfassten Routen und Problemstellen angezeigt, dort können dann die Schulwege eingezeichnet werden [31].

Die App **Travorus** bietet die Möglichkeit, den Schulweg von Kindern aufzuzeichnen und Gefahrenstellen zu sammeln. Anschließend nutzen Kommunen die Daten zur Planung, allerdings ohne diese direkt mit Travorus durchführen zu können [23].

	Umfragen	Gefahrenerfassung	Aufzeichnung	Mobile Geräte	Planerstellung	Radwege	Umfrageergebnisse	Heatmap	Verkehrsdaten	Einzeichnen	Export
schulwegcheck.de	✗	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓
gefahrenstellen.de	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗
schulwege.de	✓	✓	✗	✗	✓	✗	✓	✗	✓	✓	✓
Schulinfosystem	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Planer Baden-Württemberg	✗	✓	✗	✗	✓	✗	✗	✗	✗	✓	✓
Travorus	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗

Tabelle 3.6: Funktionsvergleich bestehender Lösungen.

Tabelle 3.6 zeigt die Unterschiede in der Funktionalität übersichtlich dargestellt. Dabei

ist zu erkennen, dass keine der vorhandenen Lösungen die erhobenen Anforderungen zu 100% abdeckt. Wichtig ist auch zu erwähnen, dass nicht zu allen Lösungen detaillierte Funktionsbeschreibungen gefunden werden konnten, da viele Systeme hinter einer (kostenpflichtigen) Registrierung verborgen sind.

Es fällt auf, dass es vor allem im Bereich der digitalen Umfragen, der Auswertung, der Verkehrsdaten und der Einbeziehung von Radwegen mangelnde Unterstützung gibt. Bei der Visualisierung von aufgezeichneten Routen scheint keine der Lösungen eine Heatmap zu besitzen, zumindest auf Basis der verfügbaren Informationen.

Somit wurde festgestellt, dass es keine bestehende Lösung gibt, die alle Anforderungen umsetzt und der zu entwickelnde Prototyp seine Daseinsberechtigung erhält.

3.7 Implikationen für das Backend

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung eines Backends. Da sich die vorhergehenden funktionalen und nicht-funktionalen Anforderungen auf das Gesamtsystem beziehen, muss an dieser Stelle also herausgearbeitet werden, welche Folgen sich für die Entwicklung des Backends ergeben.

3.7.1 Datenhaltung

Aus den Anforderungen FR01, FR02, FR03, FR05, FR12, FR14 ergibt sich, dass das Backend über eine Form der Persistierung von Daten verfügen muss. Dies umfasst insbesondere die Speicherung von Plänen, Routen, POI, und Umfrageergebnissen. Die Daten müssen in einem konsistenten, relationalen oder objektorientierten Modell abgebildet werden, sodass Beziehungen eindeutig erkennbar sind.

Die Datenhaltung muss so gestaltet sein, dass neue Schulen, Schulbezirke ohne größeren Entwicklungsaufwand integrieren zu können. Gleichzeitig müssen Datenschutz und Sicherheit gewährleistet sein, sodass insbesondere personenbezogene Daten anonymisiert gespeichert werden und nur autorisierte Nutzer Zugriff auf sensible Informationen haben.

3.7.2 Sicherheit und Datenschutz

Da das System Daten von Kindern erhebt, ist ein besonders hohes Maß an Sicherheit und Datenschutz notwendig; dies ergibt sich aus FR01, FR02, FR03, NR01 und NR02. Erhobene Informationen wie Alter, Schulweg oder Angaben zu Gefahrenstellen dürfen nicht auf einzelne Kinder zurückgeführt werden können. Anonymisierung aller Daten ist damit ein oberstes Gebot, wie schon im vorherigen Abschnitt angesprochen.

Darüber hinaus muss der Zugriff auf die Planungsoberfläche strikt kontrolliert werden. Nur autorisierte Mitarbeiter der Kommunen dürfen auf die vollständigen Datensätze zugreifen und Pläne bearbeiten. Um dies zu gewährleisten, ist ein zeitlich begrenzter Login vorgesehen: Sitzungen für Planer sind auf 24 Stunden begrenzt, während Umfrageteilnehmer ohne Registrierung teilnehmen können, um den Aufwand gering zu halten. Allerdings muss gewährleistet werden, dass jeder Teilnehmer nur an der Umfrage teilnimmt, die für ihn vorgesehen ist.

Dadurch soll einerseits die Datensicherheit gewährleistet und andererseits eine niedrigschwellige Teilnahme an den Umfragen ermöglicht werden.

3.7.3 Performance

Gemäß NR04 muss das Backend schnell arbeiten, um Wartezeiten für die Benutzer zu verhindern oder zumindest so kurz wie möglich zu halten. Insbesondere Umfragen und Kartendarstellungen sollen ohne wahrnehmbare Verzögerungen geladen werden. Für komplexere Auswertungen oder umfangreiche Datenabfragen wird ein Limit von maximal einer halben Sekunde als akzeptabel angesehen.

So soll sichergestellt werden, dass das System auch bei steigender Anzahl von Benutzern und damit auch steigender Datenmenge effizient nutzbar bleibt. Ein stabiles und schnell reagierendes Backend trägt somit auch implizit zur Benutzerfreundlichkeit bei.

3.7.4 Deployment und Wartbarkeit

Für den praktischen Einsatz in Kommunen ist gemäß NR06 eine einfache Installation und Wartung des Systems notwendig. Das Deployment soll daher als Gesamtpaket erfolgen, beispielsweise in Form eines Docker-Images oder einer gemeinsamen Bereitstellung von Frontend und Backend. Eine kurze Textanleitung soll genügen, um das System in wenigen Minuten einsatzbereit zu machen.

Langfristig soll auch die Wartbarkeit des Systems gewährleistet sein, wie NR07 besagt: Neue Schulen oder veränderte Schulbezirke müssen ohne größeren technischen Aufwand in das System integriert werden können. Dafür kann ein gewisses technischen Verständnis erforderlich sein, allerdings ist das vertretbar in Ansicht der Tatsache, dass dies sehr selten erforderlich ist (alle paar Jahre). Wichtig ist auch, dass dafür keine Änderungen am Quellcode notwendig sind.

3.7.5 Export

FR13 verlangt, dass fertige Pläne exportiert werden können. Dafür müssen sie im Backend generiert werden können und in ein gängiges, druckfähiges Format wie PDF oder ein Bildformat gebracht werden. Die Pläne enthalten alle Informationen, die während des Planungsprozesses eingetragen wurden, wie Schulwege und POI.

3.8 Zusammenfassung

Nachdem die Anforderungsanalyse an diesem Punkt abgeschlossen ist, soll an dieser Stelle eine kurze Zusammenfassung die gefundenen Ergebnisse nochmal resümieren.

Die Entwicklung und Vorstellung der Personas hat ergeben, dass es vier verschiedene Nutzergruppen gibt. Allen voran liegt der Schulwegplaner, der mit der Auswertung und Gestaltung der Schulwegpläne betraut ist. Er möchte vor allem den manuellen, papierbasierten Arbeitsprozess vereinfachen, damit mehr Schulwegpläne in kürzerer Zeit erstellt werden können; dies betrifft die Umfragen und die Analyse der Schulwege der Kinder. Die anderen beiden wichtigen Nutzergruppen sind die Kinder und ihre Eltern. Sie wünschen sich beide eine niedrigschwellige Möglichkeit, Gefahrenstellen zu melden. Für die Kinder ist es wichtig, dass das System benutzerfreundlich ist und sich gut auf mobilen Geräten bedienen lässt. Zuletzt bleiben noch ein Mitarbeiter der Schule, die sich Zugriff auf die Pläne wünschen und eine Möglichkeit, die papierbasierte Umfrageerhebung zu digitalisieren.

Aus den Personas wurden zunächst User Needs abgeleitet, die die Grundlage für die funktionalen und die nicht-funktionalen Anforderungen bilden. Funktional steht die digitale Erhebung der Umfragen und der Schulwegrouten im Vordergrund; außerdem sollen eine Karte zum Einzeichnen von Wegen, eine Visualisierung der aufgezeichneten Routen

und eine Exportmöglichkeit bestehen. Auf nicht-funktionaler Seite ist besonders der Datenschutz wichtig, da persönliche Daten einer sehr sensiblen Benutzergruppe gespeichert werden.

Für das Backend ergibt sich dabei vor allem, dass eine Datenhaltung nötig ist, um Pläne, Routen, POI, Umfragedaten, Gefahrenstellen und natürlich wichtige Stammdaten zu speichern. Der bereits erwähnte Datenschutz muss durch eine Anonymisierung der aufgezeichneten Daten sichergestellt werden, damit kein Rückschluss auf ein einzelnes Kind möglich ist. Als Teil des Sicherheitskonzepts wird außerdem eine Zugriffskontrolle durch einen Login benötigt. Außerdem muss das System performant genug sein, um Anfragen in ausreichender Geschwindigkeit, aber ohne merkliche Verzögerung beantworten zu können. Abschließend muss das Deployment so gestaltet sein, dass das Gesamtsystem in den Kommunen problemlos installiert werden kann.

4 Konzept

Ausgehend von der zuvor durchgeführten menschenzentrierten Anforderungsanalyse wird in diesem Kapitel ein Konzept zur Implementierung und Umsetzung der erhobenen Anforderungen vorgestellt.

4.1 Systemarchitektur

Das System soll aus einem Frontend, einem Backend und einer Datenbank bestehen. Damit folgt es im Wesentlichen einer Schichtenarchitektur. Auf diese Weise werden die beteiligten Rollen klar getrennt: Das Frontend ist für die Präsentation zuständig, das Backend für die Geschäftslogik und die Datenbank für die Persistenz. Die Kommunikation erfolgt ausschließlich über definierte Schnittstellen, die in Abschnitt 4.2 genauer erläutert werden. Auf die Schichtenarchitektur wird noch in Abschnitt 4.5 (S. 46) weiter eingegangen.

Abbildung 4.1: Vereinfachte Systemarchitektur.

Ziel dieses Aufbaus ist eine lose Kopplung, eine einfache Erweiterbarkeit und eine klare Trennung von Verantwortlichkeiten. Jede Schicht soll unabhängig von den anderen weiterentwickelt oder ausgetauscht werden können, was langfristig die Wartung und Erweiterung des Systems erleichtert. Darüber hinaus eröffnet eine klare Schichtenarchitektur die Möglichkeit, die einzelnen Komponenten bei Bedarf getrennt zu betreiben und so die Skalierbarkeit zu erhöhen. Abbildung 4.1 zeigt die drei Komponenten in ihrem Zusammenspiel.

4.1.1 Frontend

Wie bereits angesprochen, wird das Frontend als Webprojekt realisiert und dient den Planern als zentrale Benutzeroberfläche. Es ermöglicht die Ansicht und Bearbeitung von Schulwegdaten sowie die Anzeige von Umfrageergebnissen und relevanten Informationen zur Schulwegplanung. Die Kommunikation mit dem Backend erfolgt über eine fest definierte Schnittstelle; externe Schnittstellen werden nicht angesteuert.

Da die Entwicklung eines Frontends nicht Thema der vorliegenden Arbeit ist, wird auf eine weitere Konzeptualisierung an dieser Stelle verzichtet.

4.1.2 Backend

Das Backend übernimmt jegliche Verarbeitung von Daten und Informationen. Dabei werden Stammdaten hinterlegt (Schulen, Bezirke), Nutzerdaten (Pläne, gesetzte Markierungen oder Routen) und externe Daten (Antworten der Schülerumfragen) gespeichert. Diese Informationen werden aufbereitet und, im Falle der Umfragen, ausgewertet. Die Daten werden für das Frontend aufbereitet und über eine Schnittstelle zum Abruf bereitgestellt.

Das Backend nimmt somit eine Vermittlerrolle zwischen dem Frontend und der Datenbank bzw. externen Schnittstellen ein. Es stellt sicher, dass Daten aus unterschiedlichen Quellen zusammengeführt, konsistent verwaltet und in einer für das Frontend geeigneten Form bereitgestellt werden. Wesentliche Aufgaben sind dabei die Validierung eingehender Daten, die Zusammenführung und Vereinheitlichung von Nutzereingaben sowie die Zusammenfassung und Aufbereitung von Ergebnissen. Als zentrales Drehkreuz für Daten jedweder Art ist die Stabilität und Performance des Backends von entscheidender Bedeutung. Ebenso wichtig ist eine konsistente Schnittstellengestaltung, damit Änderungen im Backend nicht zu unmittelbaren Anpassungen im Frontend führen müssen. Dazu gehört auch eine sorgfältige Dokumentation der Schnittstellen.

Wichtig ist außerdem, dass das Backend nicht für die Darstellung zuständig ist, also beispielsweise HTML rendert und über die Schnittstelle zurückgibt. Dies ist ausschließlich Aufgabe des Frontends. Damit bleibt die Aufgabenteilung zwischen den Schichten eindeutig.

4.1.3 Datenbank

Die Datenbank dient der Datenhaltung des Gesamtsystems. Alle Eingaben der Benutzer werden dort gespeichert: Umfragedaten, aufgezeichnete Routen, Pläne und Stammdaten. Da viele der gespeicherten Informationen untereinander verknüpft sind, bietet sich ein relationales Datenbanksystem an. Neben klassischen Entitäten wie Schulen und Bezirken müssen auch Geodaten effektiv gespeichert und verarbeitet werden können, beispielsweise zur Abfrage von Routenverläufen oder Gefahrenstellen. Hierfür ist eine Unterstützung von speziellen Datentypen und räumlichen Indizes erforderlich.

In der Datenbank wird zwischen verschiedenen Arten von Daten unterschieden. Stammdaten sind langfristig stabil, während Nutzereingaben wie Umfragedaten oder Routen dynamisch hinzukommen und regelmäßig aktualisiert werden. Entsprechend muss die Datenbank sowohl Integrität und Konsistenz gewährleisten als auch hohe Zugriffszahlen bei Schreib- und Leseoperationen unterstützen. Zudem ist zu berücksichtigen, dass personenbezogene Daten unter besonderen Datenschutzanforderungen stehen. Dies betrifft insbesondere die Speicherung und Anonymisierung von Antworten aus den Umfragen.

4.2 Schnittstellenarchitektur

Schnittstellen sind die Bindeglieder zwischen den verschiedenen Komponenten des Gesamtsystems. Es muss unterschieden werden zwischen der internen Schnittstelle, also der Verbindung zwischen Front- und Backend, und den Schnittstellen nach außen zu Drittanbietern (oder einfach entfernten Systemen).

4.2.1 Interne Schnittstellen

Eine Schnittstelle zwischen dem Front- und Backend muss existieren. Wichtig ist hier eine klare Definition, da sie die Kopplung der Komponenten, die Erweiterbarkeit des Systems und die langfristige Wartbarkeit bestimmt.

Im Folgenden werden unterschiedliche Ansätze für die Schnittstellengestaltung vorgestellt und hinsichtlich ihrer Eignung für das hier entwickelte Backend diskutiert. Grundsätzlich stehen mehrere Alternativen für die Realisierung der internen Schnittstellen zur Verfügung:

Direkte Datenbankzugriffe: Eine Möglichkeit wäre es, das Frontend direkt auf die Datenbank zugreifen zu lassen. Dies hätte zwar den Vorteil einer einfachen Umsetzung,

jedoch überwiegen die Nachteile deutlich: Eine so starke Kopplung zwischen Frontend und Datenbank würde bei jeder Änderung im Datenbankschema eine Kaskade von Änderungen im Frontend auslösen. Außerdem müssten Sicherheitsaspekte, Datenvalidierung und Zugriffssteuerung im Frontend durchgeführt werden.

SOAP/WSDL: SOAP ist ein Protokoll, das den Austausch von XML-Nachrichten zwischen Systemen ermöglicht und dabei (im WS-*-Standard) Mechanismen wie Sicherheit und Transaktionen unterstützt. Damit Clients nicht nur die Nachrichten selbst, sondern auch die angebotenen Funktionen eines Dienstes verstehen können, wird in der Regel zusätzlich die WSDL (Web Services Description Language) verwendet. WSDL beschreibt, welche Operationen verfügbar sind und wie Eingabe bzw. Ausgabe aussehen. Während SOAP somit für den eigentlichen Nachrichtentransport verantwortlich ist, dient WSDL als formaler Vertrag zwischen Server und Client. Allerdings bringt dies oft einen erheblichen Overhead mit sich, sodass SOAP-basierte Dienste bei der Performance häufig das Nachsehen haben, gerade auf mobilen Clients [12].

REST-API: REST (Representational State Transfer) benutzt das HTTP-Protokoll und verwendet dessen Methoden (`GET`, `POST`, `PUT`, `DELETE`, `PATCH`). Es ist häufig weniger komplex zu nutzen als das vorher erwähnte SOAP/WSDL und erlaubt eine klare Trennung von Ressourcen. REST ist weit verbreitet und wird von gängigen Frameworks (Spring, NodeJS/Express, Drogon) nativ unterstützt und benötigt keine besondere Unterstützung durch Browser. Änderungen lassen sich in aller Regel einfach durchführen [18].

Event-basierte Kommunikation: Verschiedene Technologien wie MQTT und AMQP lassen sich unter dem Oberbegriff der event-basierten Kommunikation zusammenfassen. Ihnen allen ist gemein, dass sie asynchron über einen dazwischengeschalteten Broker kommunizieren, wodurch Sender und Empfänger entkoppelt werden. AMQP basiert dabei auf einem Message-Queueing-Modell; MQTT hingegen verwendet eine Publish/Subscribe-Architektur. Ohne auf die konkreten Funktionsweisen im Rahmen dieser Arbeit einzugehen, bringen sie zusätzliche Komplexität in das System, die für einfache Anfrage-Antwort-Szenarien oft nicht gerechtfertigt ist [34, 40].

Vor dem Hintergrund der Anforderungen erweist sich REST als die am besten geeignete Lösung. REST verbindet eine klare Trennung von Frontend und Backend mit einer einfach nutzbaren und implementierbaren Kommunikationsform, die ohne zusätzliche Bibliotheken auskommt. Damit wird eine lose Kopplung der Komponenten erreicht, die langfristig die Wartbarkeit und Erweiterbarkeit gewährleistet.

Zudem erlaubt REST die Einhaltung wichtiger Qualitätsmerkmale: Durch definierte Endpunkte kann Konsistenz sichergestellt werden, während gleichzeitig Flexibilität für Erweiterungen bleibt. Sicherheit kann über Mechanismen wie Authentifizierungstoken oder HTTPS gewährleistet werden. Schließlich entspricht REST dem etablierten Stand der Technik für Webanwendungen. Für weitere Details wird auf das Grundlagenkapitel, Abschnitt 2.3.2, verwiesen.

Aus diesen Gründen wird die Schnittstelle auf einer REST-API basieren, die im weiteren Verlauf spezifiziert wird. Die API soll sich dabei an den Ressourcen des Gesamtsystems orientieren und stellt Endpunkte für den lesenden und schreiben Zugriff bereit, je nach Art der Ressource; so ergeben Endpunkte zum Ändern der Stammdaten (Schulen, Bezirke) wenig Sinn, sehr wohl aber die Änderung von bestehenden Plänen oder Fragebögen.

Die Datenübermittlung soll grundsätzlich mittels JSON-konformer Datenstrukturen erfolgen; dabei soll auch darauf geachtet werden, dass alle JSON-Nachrichten dieselbe

Struktur haben, damit sich das Frontend immer auf dieselben Rahmenbedingungen von Antworten verlassen kann.

Ressource	URI	Beschreibung
Authentifizierung	/api/auth	Stellt Endpunkte zur Benutzerauthentifizierung bereit (Login/Logout)
Daten	/api/data	Liefert Stamm- und Benutzerdaten zurück, insbesondere POI- und Schuldaten
Kartenebenen	/api/layers	Bereitet Kartenebenen vor; das sind insbesondere die Geschwindigkeits-, Ampel- und Übergangsanzeige. Auch die Bezirksdaten kommen von diesem Endpunkt
Pläne	/api/planning	Enthält alle Endpunkte, die mit den Plänen zusammenhängen, ebenso wie die Fragebögen
Umfragen	/api/survey	Stellt Umfragen bereit, nimmt neue Antworten entgegen und wertet sie aus

Tabelle 4.1: Übersicht aller Ressourcen der REST-API.

Eine Zugriffssteuerung mittels Rollen ist nicht nötig. Allerdings muss der Großteil der Endpunkte hinter einem Login verborgen werden, damit nicht jedermann potentiell persönliche Daten, wenn auch anonymisiert, abrufen kann. Eine Versionierung ist ebenfalls nicht nötig. Außerdem liefern alle Endpunkte entsprechende HTTP-Statuscodes zurück, um Erfolg oder Misserfolg einer Anfrage zu signalisieren.

Tabelle 4.1 zeigt eine Übersicht über die Ressourcen, zusammen mit ihrer eindeutigen URI. Sie unterstützen, je nach Anwendungsfall, verschiedene HTTP-Methoden und Unterpfade.

4.2.2 Externe Schnittstellen

Bestimmte Anforderungen benötigen Informationen von externen Diensten. Dies sind vor allem die Verkehrsdaten für die Schulbezirke, also Straßenübergänge, Bushaltestellen und Geschwindigkeitsinformationen. All diese Informationen können von der Overpass-API zur Verfügung gestellt werden.

Die Overpass-API stellt ein eigenes Abfrageformat zur Verfügung, mit dem Geodaten in frei definierbaren Kartenausschnitten abgefragt werden können. So ist es beispielsweise möglich, in einem bestimmten Radius um eine Schule nach Querungshilfen, Fußgängerampeln oder Haltestellen zu suchen. Das Backend vermittelt an dieser Stelle: Es formuliert die Abfrage, wertet die Ergebnisse aus und stellt die Daten in aufbereiteter Form dem Frontend zur Verfügung; ein Beispiel dieses Datenflusses wurde bereits in Abbildung 4.4 (S. 44) gezeigt.

Für das Gesamtsystem ergeben sich daraus zwei wesentliche Folgen:

- Abhängigkeit von externer Verfügbarkeit: Die Overpass-API ist ein öffentlicher Dienst, sodass Ausfälle oder längere Antwortzeiten das System in wesentlichem Umfang beeinträchtigen können.
- Datenmenge: Da die API sehr umfangreiche Rohdaten liefert, muss das Backend die Ergebnisse formatieren und auf das für das Frontend Wesentliche beschränken.

Beiden Punkten kann begegnet werden; die Overpass-API könnte lokal betrieben werden, beispielsweise zusammen mit dem Backend auf demselben Server. Damit würden Antwortzeiten drastisch reduziert.

Der Datenmenge kann durch geeignete Caching-Mechanismen begegnet werden. Dies löst auch eventuelle Verfügbarkeitsprobleme, falls die Overpass-API nicht zur Verfügung steht.

4.3 Datenmodell und -fluss

Die Anforderungsanalyse hat aufgezeigt, welche Daten gespeichert werden müssen; die Gesamtheit der Datenhaltung soll in einem ersten Datenmodell skizziert werden. Anschließend wird der Datenfluss innerhalb des Gesamtsystem diskutiert.

4.3.1 Datenmodell

Am Ende der Anforderungsanalyse wurden die Implikationen für das Backend besprochen. Dabei wurde in Abschnitt 3.7.1 herausgearbeitet, welche Daten gespeichert werden müssen. Dabei stehen folgende Entitäten im Vordergrund:

Benutzer: Diese Entität speichert berechtigte Benutzer für den Login im Planungssystem. Dazu gehört ein Nutzername und ein Passwort.

Session: Die Sessions der Benutzer müssen in der Datenbank gespeichert werden.

Schule: Repräsentiert eine Schule mit Stammdaten. Zu den Stammdaten gehören der Name und die Anschrift. Da die Planung pro Schule durchgeführt wird und immer für den gesamten Schulbezirk gilt, muss auch die Zugehörigkeit zu einem bestimmten Schulbezirk gespeichert werden. Außerdem soll auch noch die Position der Schule in Form von Geokoordinaten gespeichert werden, um sie bei der Planung auf der Karte korrekt darstellen zu können.

Schulbezirk: Der Schulbezirk enthält Informationen zu dessen Bezeichnung (in Form einer Bezirksnummer) und seine Grenzen als Folge von Geokoordinaten. Letztere sind nötig, um die Grenzen des Bezirks auf der Karte darstellen zu können.

Plan: Diese Entität repräsentiert einen Plan mit allen dazugehörigen Basisinformationen. Das sind der Name des Planes, das Datum der Erstellung, das Datum der letzten Änderung und die Schule, für die dieser Plan erstellt wurde. Zusätzlich sind Verknüpfungen auf Fragebogen und eingezeichnete Routen erforderlich.

Fragebogen: Die Fragebögen müssen ebenfalls in der Datenbank gespeichert werden. Dies zieht noch weitere Abhängigkeiten wie die Fragen und die eventuellen Antwortoptionen mit sich. Ansonsten enthält die Entität keine weiteren Informationen.

Fragen: Die Fragen müssen einem Fragebogen zugeordnet werden und mit ihrem Typ gespeichert werden.

Frageoptionen: Je nach Fragetyp kann es verschiedene Optionen geben (Multiple Choice, Dropdown). Diese müssen hier mit Verweis auf die Frage gespeichert werden.

Antwort: Die Antworten der Schüler müssen gespeichert werden. Dabei gibt es, in Abhängigkeit des Fragetyps, verschiedene Möglichkeiten: Optionen, Freitextfelder oder Zahlenwerte. Die Zugehörigkeit zum Fragebogen muss ebenfalls sichergestellt werden.

Aufgezeichnete Route: Ebenso wie die Antworten müssen auch die aufgezeichneten Routen der Kinder gespeichert werden. Dazu gehört ebenfalls eine Zuordnung zum Fragebogen, sowie eine Reihe von Geokoordinaten.

Gefahrenstelle: Die Gefahrenstellen müssen, analog zu den aufgezeichneten Routen, ebenfalls repräsentiert werden. Neben einer Beschreibung und den Geokoordinaten muss auch hier eine Zuordnung zum Fragebogen existieren.

POI: POI müssen gespeichert werden. Das betrifft sowohl eine Beschreibung und das Bild selbst; da die Bilder nicht viel Speicher benötigen, können diese auch in der Datenbank als Base64 kodiert gespeichert werden.

Einzeichnungen: Die von den Planern eingezeichneten Routen und die gesetzten POI müssen gespeichert werden; die Speicherung erfolgt in eigenen Entitäten. Eine Zuordnung zum Plan ist ebenfalls nötig.

Eingezeichnete Routen: Routen, die ein Planer auf den Plan einzeichnet. Enthält eine Sammlung von Koordinaten.

Eingezeichnete POI: POI, die ein Planer einzeichnet. Verweis auf einen POI.

In einem ER-Diagramm dargestellt ergibt sich das Bild aus Abbildung 4.2. Dabei sind auch die Kardinalitäten erkennbar; auf die Attribute wurde im Diagramm bewusst verzichtet, da es hier nur um die Fremdschlüsselbeziehungen zwischen den Entitäten gehen soll.

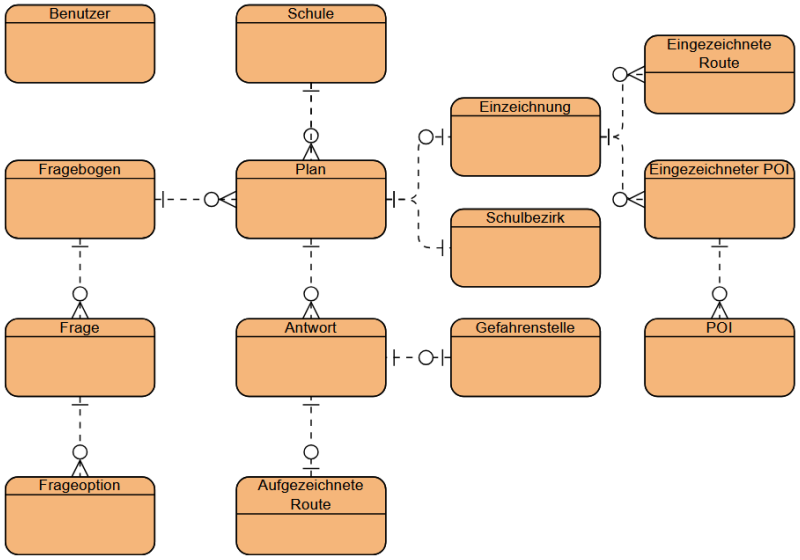

Abbildung 4.2: ER-Diagramm für das Datenmodell.

Hier ist wichtig zu erwähnen, dass dies nur ein Konzept des Datenmodells darstellt. Im Laufe der Implementierung und der Normalisierung der Tabellen werden möglicherweise noch weitere Tabellen dazukommen. Hervorzuheben ist aber schon jetzt die hierarchische Anordnung der Tabellen. Am Beispiel eines Fragebogens mit der von ihm abhängenden Frage-Entität und der wiederum davon abhängenden Frageoption-Entität ist dieses Vorgehen zu erkennen. Auf diese Weise kann später durch Kaskadierung sichergestellt werden, dass es zu keinen verwaisten Datensätzen kommt.

4.3.2 Datenfluss

Der Datenfluss beschreibt, welche Daten von wo durch das Backend an welche Stelle gelangen. Dabei ist es zunächst wichtig, die Datenquellen zu identifizieren. Im vorliegenden Projekt gibt es drei wesentliche Datenquellen:

- Die von den Kindern erhobenen Daten zu den Umfragen, der Gefahrenstellen und den aufgezeichneten Routen.
- Daten, die ein Planer eingibt. Dazu zählt die Neuanlage von Daten und das Bearbeiten bestehender Daten, wie beispielsweise das Einzeichnen von Schulwegen auf eine Karte.
- Externe Daten, wie Kartendienste oder andere Dienste.

Daten werden im Frontend eingegeben und gelangen über die Endpunkte der REST-API zur weiteren Verarbeitung in das Backend. Dort werden sie in die Datenbank geschrieben und stehen den Planern zur Verfügung, wie Abbildung 4.3 verdeutlicht. Sie zeigt den Vorgang des Ausfüllens einer Umfrage. Zunächst benutzt der Schüler die Anzeigeelemente des Frontends, um Daten einzugeben. Anschließend generiert das Frontend eine Anfrage an die REST-API mithilfe der `POST`-Methode. Die API empfängt die Daten und leitet sie weiter in das Backend, wo sie gespeichert werden. Die Kaskade von Erfolgsmeldungen lässt die API letztlich eine Erfolgsmeldung in Form des Statuscodes `200 - OK` an das Frontend senden, was es zum Anlass nimmt, dem Benutzer eine Erfolgsmeldung anzuzeigen.

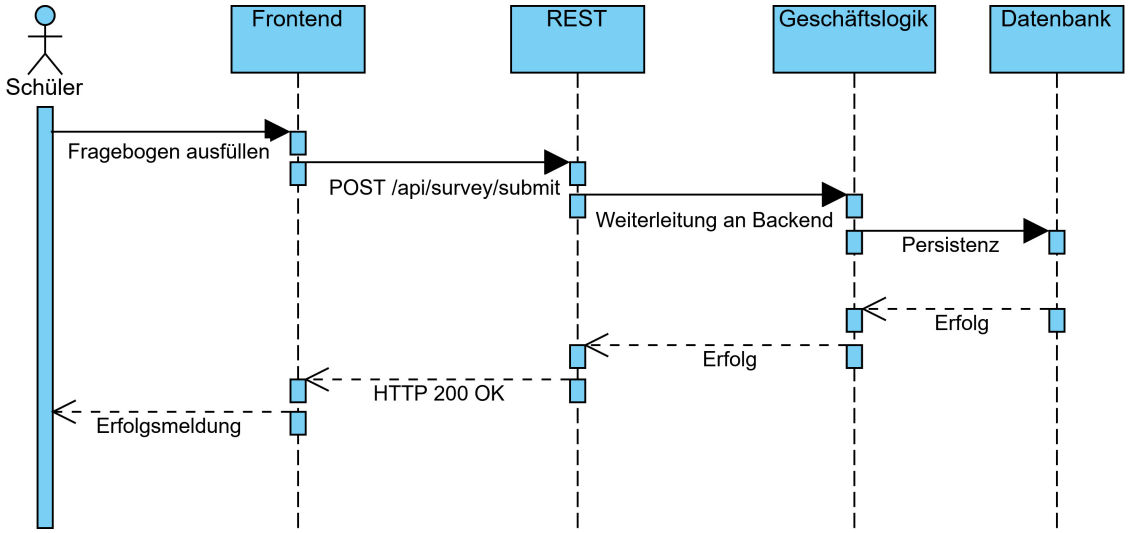

Abbildung 4.3: Sequenzdiagramm für den Vorgang „Umfrageantwort speichern“.

Abbildung 4.3 zeigt einen geradlinigen Ablauf, der häufig vorkommt; die Daten werden nur weitergereicht und gespeichert. Es sind aber auch Datenflüsse möglich, die nicht so geradlinig sind, sondern innerhalb des Backends noch weitere Aktionen auslösen, wie Abbildung 4.4 zeigt.

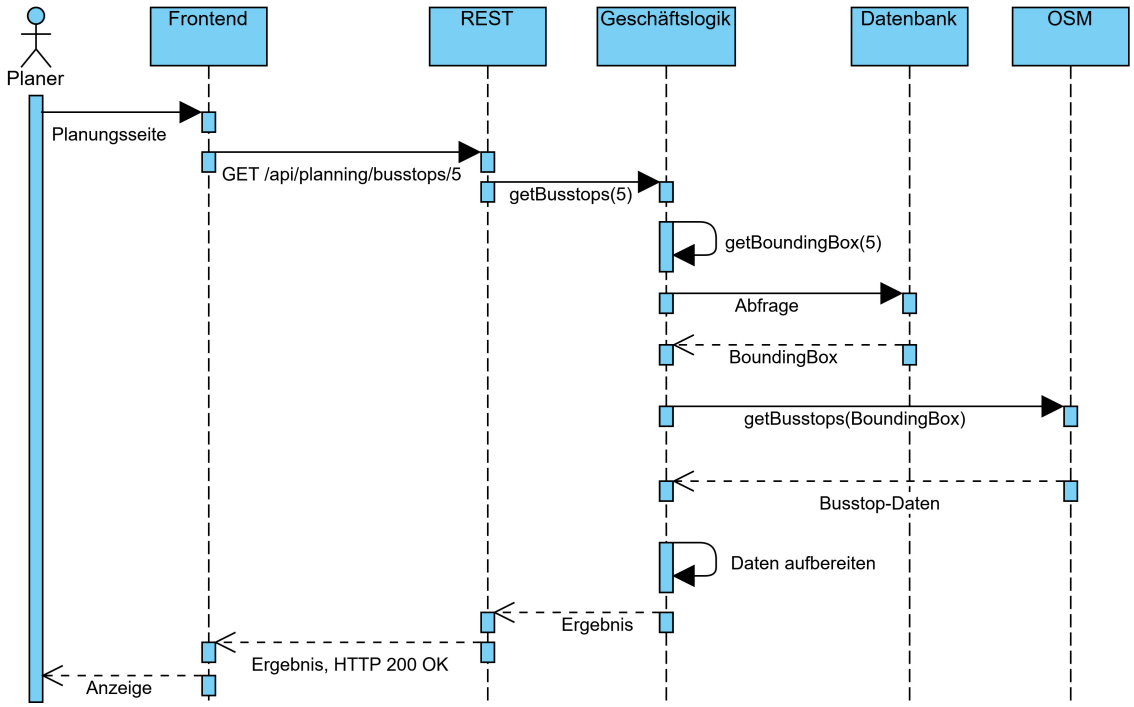

Abbildung 4.4: Sequenzdiagramm für den Vorgang „Bushaltestellen abfragen“.

Hier ist zu sehen, wie eine Anfrage weitere Aktionen im Backend auslöst. Es sollen alle Bushaltestellen in einem Bezirk (hier Bezirk 5) abgefragt werden. Das Frontend macht eine Anfrage an die REST-API, die den Aufruf wieder an das Backend weiterleitet. Bevor die Bushaltestellen von OSM abgefragt werden können, muss allerdings noch die *bounding box* (deutsch nicht wirklich einheitlich, oft als *Begrenzungsrahmen* bezeichnet) abgefragt werden, deren Daten in den Stammdaten gespeichert sein könnten. Die Antwort der Datenbank geht zunächst wieder an das Backend, wo dann der eigentliche Aufruf an OSM (bzw. eine OSM-API) durchgeführt wird. Die Antwort gelangt dann in das Backend, wo die Daten aufbereitet werden müssen, d.h., überflüssige Informationen werden entfernt und alles in ein Format gebracht, das die Verarbeitung im Frontend vereinfacht. Das aufbereitete Ergebnis gelangt dann zum Frontend und wird schließlich angezeigt.

Abschließend lässt sich sagen, dass der Datenfluss gewissen Mustern folgt. Entweder werden Daten geradlinig abgefragt oder gespeichert, oder es werden noch weitere Komponenten der Geschäftslogik in Gang gesetzt. Das könnten Zwischenschritte oder Aufbereitung sein (wie gezeigt) oder auch Validierung von Daten.

4.4 Sicherheit und Datenschutz

Zentrale, nicht-funktionale Anforderungen sind die Sicherheit und der Datenschutz. Bei dem vorliegenden Projekt sind beide Punkte umso wichtiger, da es sich um sensible persönliche Daten von Kindern im Grundschulalter handelt. In der Datenbank werden Antworten der Fragebögen gespeichert und persönliche Bewegungsprofile (Schulwegaufzeichnungen) der Kinder. Aus diesem Grund muss dieser Aspekt bereits hier diskutiert werden und nicht nur als Randnotiz in der folgenden Implementierung.

4.4.1 Grundprinzipien

Sicherheit umfasst den Schutz vor unerlaubtem Zugriff, Änderung oder Löschung von Daten. Ein verbreitetes Schutzkonzept bietet das CIA-Modell [15, 58]:

Confidentiality (Vertraulichkeit): Daten dürfen nur von berechtigten Personen eingesehen werden. Dies betrifft insbesondere die erwähnten Standortdaten und Fragebögen, wobei erstere den persönlichen Lebensbereich eines Kindes betreffen .

Integrity (Integrität): Daten müssen korrekt und vollständig gespeichert und übertragen werden. Änderungen sind auf dem Übertragungsweg ausgeschlossen.

Availability (Verfügbarkeit): Daten müssen für (berechtigte) Benutzer immer zu erreichen sein.

Gelegentlich wird noch die Accountability (Nachvollziehbarkeit) in das CIA-Konzept aufgenommen; dabei handelt es sich um das Prinzip, dass jede Art von Datenverarbeitung oder -änderung protokolliert und somit nachvollzogen werden kann [38].

4.4.2 Datenschutzrechtliche Bestimmungen

Den maßgeblichen Rechtsrahmen für das europäische Datenschutzrecht gibt die Datenschutz-Grundverordnung (DSGVO) vor. In ihr wird festgeschrieben, dass personenbezogene Daten nur für definierte Zwecke verarbeitet werden dürfen. Die wichtigsten, für das Konzept relevanten Punkte der DSGVO sind:

Datenminimierung: Es dürfen nur die Daten erhoben werden, die für den jeweiligen Zweck absolut notwendig sind. Im Kontext der Schulwegplanung bedeutet das beispielsweise, dass nicht mehr Standortdaten gespeichert werden dürfen, als für die Planung nötig sind (Art. 5 Abs. 1 Buchst. c DSGVO).

Zweckbindung: Erhobene Daten dürfen nicht für andere Zwecke als für die Schulwegplanung benutzt werden (Art. 5 Abs. 1 Buchst. b DSGVO).

Rechte: Schüler, in der Regel vertreten durch ihre Eltern, müssen über die Datenerhebung informiert werden und haben ein Auskunfts-, Korrektur- und Löschungsrecht (Art. 15, Art. 16, Art. 17 DSGVO).

Speicherdauer: Daten dürfen nur so lange wie nötig und so kurz wie möglich gespeichert werden; mindestens aber so lange, wie es für die Analyse, Planung und Aktualisierung von Plänen notwendig ist (Art. 5 Abs. 1 Buchst. e DSGVO).

Die DSGVO nennt explizit die beiden ersten CIA-Prinzipien, Vertraulichkeit und Integrität, im Bezug auf persönliche Daten, so dass die Wichtigkeit dieser Prinzipien nochmal hervorgehoben wird (Art. 5 Abs. 1 Buchst. f DSGVO); eine Nennung der Verfügbarkeit als drittes CIA-Prinzip ergibt im Kontext der DSGVO keinen Sinn, da es sich vor allem auf den berechtigten Personenkreis der datenerhebenen Stelle bezieht [17].

4.4.3 Sicherheit auf Systemebene

Das Sicherheitskonzept des Systems basiert auf mehreren Säulen, die zusammen die Vertraulichkeit, Integrität und Verfügbarkeit sensibler Daten gewährleisten.

Zunächst sorgt eine Zugriffskontrolle dafür, dass jede Benutzerrolle nur die Berechtigungen erhält, die für ihre Aufgaben notwendig sind. Schüler erhalten somit beispielsweise keinen Zugriff auf die Listen aller Schulen oder Fragebögen, und Planer können keine persönlichen Informationen einzelner Umfrageteilnehmer einsehen.

Darauf aufbauend wird eine Authentifizierung eingeführt: Vor dem Zugriff auf Funktionen müssen sich Benutzer eindeutig identifizieren. Bei Planern erfolgt dies in Form eines klassischen Logins, bei Schülern können alternative Methoden zur Authentifizierung genutzt werden, um die Teilnahme möglichst niedrigschwellig zu gestalten; denkbar ist hier ein Zugang mittels QR-Code und einer einfachen, cookiebasierten Session.

Ein selbstverständlicher Bestandteil des Sicherheitskonzepts ist die verschlüsselte Datenübertragung über HTTPS, sodass sensible Informationen wie Standortdaten während der Übertragung nicht abgezweigt oder verändert werden können. Auch die verschlüsselte Speicherung besonders sensibler Daten auf dem Server ist möglich, ergänzt durch ein geeignetes Backup-Konzept, um Datenverlust zu vermeiden.

Schließlich werden sämtliche Zugriffe und Änderungen in einem Log festgehalten. Dieser Log enthält nicht nur die Zugriffe, sondern kann auch interne Fehlermeldungen und Warnungen aufnehmen. Durch die Auswertung dieser Logs lassen sich Schwachstellen erkennen und potenzielle Angriffspunkte frühzeitig identifizieren.

4.5 Softwarearchitektur und Qualitätsanforderungen

Das Gesamtsystem wird durch eine Schichtenarchitektur beschrieben. Die Schichtenarchitektur ist eine Form der monolithischen, hierarchischen Architektur, bei der jede Schicht eine spezifische Aufgabe hat und von den anderen Schichten isoliert ist [42].

Hier wird unterschieden zwischen einer offenen und einer geschlossenen Schichtenarchitektur. Bei einer offenen Schichtenarchitektur ist es möglich, dass eine Schicht eine andere „überspringt“, also auf eine tiefere Schicht direkt zugreift. Bei einer geschlossenen Schichtenarchitektur hingegen ist dies nicht erlaubt; eine Schicht kann jeweils nur auf die Schnittstelle der unter ihr liegenden Schicht zugreifen. Dies begünstigt die Wartbarkeit, da jede Schicht unabhängig von den anderen entwickelt und verändert werden kann, solange die Schnittstelle gleich bleibt [42].

Die häufigste Unterteilung geht von vier Schichten aus: Präsentationsschicht, Geschäftslogik, Persistenz und Datenbank, wobei letztere auch zu dann insgesamt drei Schichten zusammengefasst werden können [42]. Die erste ist für die Darstellung nach außen zuständig, beispielsweise die grafische Oberfläche; eine bloße textbasierte Repräsentation reicht aber auch aus.

Im vorliegenden Projekt gibt es zwei Varianten, eine Schichtenarchitektur zu beschreiben: Betrachtet man das Gesamtprojekt, wäre das Frontend die Präsentationsschicht, das Backend (REST-API und Geschäftslogik) die Geschäftslogikschicht und das ORM zusammen mit der Datenbank die Persistenzschicht. Wird aber nur das Backend betrachtet, entspräche die REST-API der Präsentationsschicht. Aber letztlich ändert die Betrachtungsweise nichts an der Klassifizierung des Backends als Schichtenarchitektur.

Im Sinne der Softwarequalität soll die Implementierung als geschlossene Schichtenarchitektur erfolgen. Dies verbessert, wie erwähnt, die Wartbarkeit, da alle Schichten unabhängig voneinander bearbeitet werden können.

Die Wartbarkeit ist nur der Oberbegriff für mehrere Kategorien der Softwarequalität. Sie teilt sich in folgende Punkte auf, die in das Konzept übernommen werden und somit am Ende einer Evaluation standhalten müssen [27]:

Modularität: Die Fähigkeit des Systems, Änderungen an einer Komponente vorzuneh-

men, ohne dass andere Komponenten davon betroffen sind (lose Kopplung der Komponenten).

Änderbarkeit: Änderungen müssen einfach umgesetzt werden können, ohne großen Aufwand oder das Risiko, Fehler in anderen Komponenten auszulösen.

Testbarkeit: Das System muss (automatischen) Tests zugänglich sein, also so aufgebaut sein, dass die Komponenten getrennt getestet werden können.

Zuletzt muss noch der Begriff der Skalierbarkeit betrachtet werden. Skalierbarkeit beschreibt die Fähigkeit eines Systems, mit stärker oder schwächer werdender Arbeitslast umzugehen. Dabei ist mit der vertikalen Skalierung gemeint, dass die für das System verfügbaren Ressourcen erhöht werden, während bei der horizontalen Skalierung mehrere Instanzen eines Systems gestartet werden, um mit der Arbeitslast umzugehen [7].

4.6 Betrieb und Deployment

Im vorliegenden Projekt muss sowohl das Backend als auch das Frontend so bereitgestellt werden, dass eine zuverlässige Nutzung zu jeder Zeit gewährleistet werden kann. Gleichzeitig sind Punkte wie Skalierbarkeit, Aktualisierbarkeit und einfache Wartung zu berücksichtigen.

Die Anwendung besteht aus drei wesentlichen Komponenten: dem Backend auf Basis von Spring, einer relationalen Datenbank sowie dem Frontend für Desktops und mobile Endgeräte. Abbildung 4.5 zeigt den grundsätzlichen Aufbau und die Verbindungen zwischen den Komponenten.

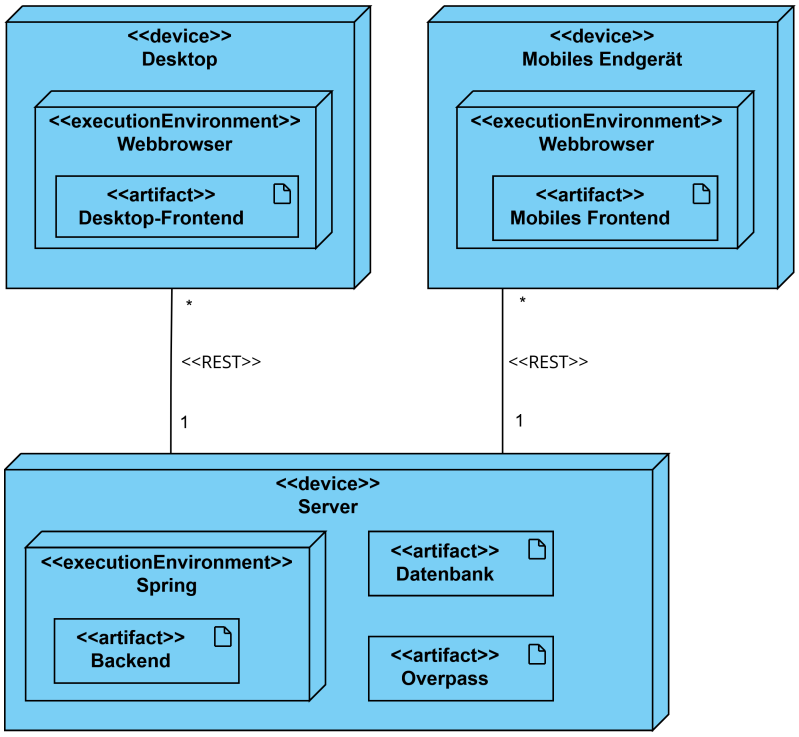

Abbildung 4.5: Verteilungsdiagramm des Gesamtsystems.

Wie Abbildung 4.5 zeigt, greifen sowohl Desktops als auch mobile Endgeräte über REST auf das Backend zu. Es handelt sich um ein klassisches Client-Server-Modell, bei dem

mehrere Clients gleichzeitig auf einen Server zugreifen können, wie es in Abschnitt 2.3.1 erläutert wird. Das Backend ist verantwortlich für die Verarbeitung der Anfragen, die Geschäftslogik, die Kommunikation mit der Datenbank sowie den Zugriff auf die Overpass-API, die auf demselben (physischen) Server läuft.

4.6.1 Varianten des Deployments

Es kommen zwei Varianten des Deployments in Betracht. Die erste Variante besteht darin, das Backend getrennt vom Frontend bereitzustellen. Dabei müssen beide Komponenten nicht unbedingt auf zwei physisch getrennten Servern liegen, sie können auch auf einem Server in getrennten Prozessen koexistieren (logische Trennung). Einige Vorteile dieser Vorgehensweise sind:

- Das Frontend und das Backend können getrennt voneinander (weiter)entwickelt werden.
- Die Skalierung des Backends wird vereinfacht; beispielsweise kann ein Backend bei Bedarf horizontal skaliert oder redundant ausgebaut werden. Da dann mehrere Instanzen des Backends existieren, wären mehrere mit ihnen verbundene Instanzen des Frontends sinnlos.
- Je nach Backendtechnolgie ist das Deployment beider Teile einfacher; zwar kann bei Spring das Frontend statisch bereitgestellt werden, d.h. die kompilierten Frontend-Dateien werden über den eingebauten Webserver ausgeliefert. Das erschwert das Deployment allerdings, wenn beide Komponenten unabhängig voneinander entwickelt werden.

Dem gegenüber steht ein komplexeres Deployment und eine Konkurrenz um die Serverressourcen. Der erste Punkt mag auf den ersten Blick widersprüchlich zum letztgenannten Vorteil klingen; gemeint ist hier allerdings der erhöhte Konfigurationsaufwand zweier bereitzustellender Anwendungen.

Die zweite Variante ist nun also das gemeinsame Deployment beider Komponenten. Wie bereits in den Vorteilen der getrennten Bereitstellung angeschnitten, kann Spring (oder auch andere Backendtechnologien) die Dateien des Frontends zur Verfügung stellen, d.h. über das Internet abrufbar machen.

- Das Gesamtsystem kann als eine große Anwendung bereitgestellt werden. Dies erleichtert auch eventuelle Containerisierung durch Docker oder vergleichbare Technologien.
- Da kein eigener Webserver für das Frontend benötigt wird, ist weniger Infrastruktur nötig.

Die Nachteile dieser Art der Bereitstellung entsprechen den Vorteilen der getrennten Bereitstellung.

Es ist zu erkennen, dass beide Varianten ihre Vor- und Nachteile mit sich bringen. Für das vorliegende Projekt mit seinem Status als Prototyp ergibt sich ein Mittelweg mit folgendem Vorgehen:

- Das Projekt wird zunächst als zwei getrennte Komponenten entwickelt und zum Testen bereitgestellt. Dies vereinfacht die unabhängige Arbeit an Front- und Backend.

- Nach Projektabschluss wird ein gemeinsames Deployment durchgeführt, um eine Containerisierung zu vereinfachen und eine schnelle Installation aller benötigten Komponenten zu gewährleisten.

4.6.2 Betrieb

Neben der Frage der Bereitstellung müssen für den laufenden Betrieb noch weitere Punkte berücksichtigt werden.

Das Backend muss horizontal skalierbar sein, falls es mit der Last von Anfragen nicht zurechtkommt. Dieses Szenario ist zum Zeitpunkt der Entwicklung nicht zu erwarten, sollte aber dennoch beachtet werden. Denkbar ist dieses Szenario für mögliche Weiterentwicklungen, wenn das Gesamtsystem auch Informationen für Außenstehende bereitstellen soll, was sich aber aus den aktuellen Anforderungen nicht ergibt.

Wie bereits im Kapitel 4.4 angesprochen, muss eine Backup-Strategie dafür sorgen, dass es zu keinerlei gravierenden Datenbankausfällen kommt bzw. die Folgen eines Ausfalls durch Datenrettungsmaßnahmen abfangbar sind.

Für weitere Aspekte des Betriebes wird erneut auf Kapitel 4.4 verwiesen, bzgl. der Nutzung von HTTPS und konsequentem Logging.

4.7 Zusammenfassung

Die Grundlage für das Backend bildet eine mehrschichtige Architektur, in der das Backend als die hauptverantwortliche Instanz für jede Datenverarbeitung mit der Datenbank und externen Diensten kommuniziert, um Daten aufbereitet an das Frontend übergeben zu können. Es übernimmt die Verwaltung von Stamm- und Nutzerdaten, Umfragen, Routen, Gefahrenstellen und stellt diese zum Abruf durch das Frontend mittels einer REST-API bereit.

Das Datenmodell umfasst die relevanten Entitäten wie Schulen, Bezirke, Pläne, Fragebögen, Antworten, Routen, Gefahrenstellen und POI. Diese Entitäten stehen untereinander durch Fremdschlüssel in Beziehung, damit eine konsistente Datenhaltung möglich wird. Die Analyse des Datenfluss hat gezeigt, wie Eingaben validiert, gespeichert und, je nach Anwendungsfall, mit zusätzlichen Schritten aufbereitet wird.

Die Schnittstellenarchitektur basiert auf einer REST-API, die Ressourcen eindeutig adressiert und standardisierte Methoden zur Verfügung stellt. Externe Datenquellen wie Verkehrsinformationen werden über die Overpass-API eingebunden und serverseitig verarbeitet, wobei Caching vorgesehen ist, um die Antwortzeiten gering zu halten.

Besonderes Augenmerk lag auf Sicherheit und Datenschutz. Neben technischen Maßnahmen wie Zugriffskontrolle, Authentifizierung, verschlüsselter Datenübertragung mittels HTTPS und Logging wurden die Vorgaben der DSGVO berücksichtigt, insbesondere Datenminimierung, Zweckbindung sowie Auskunfts-, Löschungs- und Änderungsrechte.

Ergänzend wurde die Softwarearchitektur als geschlossene Schichtenarchitektur ausgearbeitet, die Modularität, Änderbarkeit, Testbarkeit und Skalierbarkeit unterstützt. Für den Betrieb wurden Varianten des Deployments vorgestellt, wobei ein zunächst getrenntes und später gemeinsames Deployment vorgesehen ist.

5 Implementierung

Nach der Konzeptualisierung der Anforderungen soll in diesem Kapitel eine Implementierung vorgestellt werden. Dabei wird zunächst eine Übersicht über die ausgewählten Technologien geschaffen, anschließend werden die Projektstruktur, die Struktur der Datenbank und einige Implementierungsdetails vorgestellt.

5.1 Auswahl der Technologien

Für verschiedene Bereiche der Implementierung standen verschiedene Technologien zur Verfügung, deren Auswahl im Folgenden erläutert werden soll.

5.1.1 Spring

Spring ist ein weit verbreitetes Open-Source-Framework für die Entwicklung von Java-Anwendungen. Es zielt darauf ab, die Komplexität bei der Erstellung moderner Software (insbesondere im Enterprise-Bereich) zu verringern und gleichzeitig dem Entwickler einen umfassenden Werkzeugkasten zur Verfügung zu stellen. Der Kern von Spring ist ein Container zur Verwaltung von Objekten und deren Lebenszyklen, um so eine möglichst lose Kopplung der Komponenten untereinander zu erreichen. Durch die sog. Dependency Injection (DI) werden Abhängigkeiten zentral verwaltet und das Framework kümmert sich um die Initialisierung etwaiger Abhängigkeiten, was die Wartbarkeit und Testbarkeit deutlich verbessert. Neben dieser Kernfunktionalität stellt Spring eine ganze Palette von Erweiterungen bereit, um verschiedene Anforderungen umsetzen zu können; so gibt es beispielsweise Erweiterungen für die Webentwicklung, den Zugriff auf Datenbanken oder zur Umsetzung von Sicherheitsanforderungen. Davon kann ein Entwickler jeweils nur das nutzen, was er tatsächlich benötigt, ohne überflüssigen Ballast in das Projekt einbinden zu müssen oder viel Boilerplate-Code generieren zu müssen.

Spring ist in der Lage, schnell bei der Projektentwicklung zu unterstützen, indem der Großteil aller benötigen Komponenten bereits vorkonfiguriert ist (*convention over configuration*). Durch sogenannte Starter-Pakete lassen sich häufig zu entwickelnde Projekte schnell realisieren; das vorliegende Projekt wurde beispielsweise mithilfe eines solchen Starter-Paketes realisiert, das alles enthält, was für eine REST-API nötig ist, inklusive eingebetteter Serverumgebung (Tomcat), sodass die REST-API sofort ohne großen Konfigurationsaufwand lauffähig ist. Gängige Logging-Frameworks sind ebenfalls bereits integriert und arbeiten nahtlos mit den Komponenten zusammen. Eine Übersicht aller implementierten REST-Endpunkte findet sich in Anhang B.

5.1.2 RDBMS

Das Backend verfügt über eine Datenbank, um die eingegebenen Informationen zu speichern und strukturiert abfragen zu können. Dabei wurde sich für eine SQL-Datenbank entschieden; gegen NoSQL (oder andere) spricht hier vor allem, dass es viele Beziehungen zwischen den Entitäten untereinander gibt. Gerade für solche Datenbeziehungen ist eine strukturierte Abfragesprache nötig, um effizient Daten verarbeiten zu können.

Im SQL-Bereich gibt es zwei, eventuell auch drei Platzhirsche: MySQL (MariaDB), Microsoft SQL Server und PostgreSQL. Während der Microsoft SQL Server bereits am proprietären Modell und der (einfachen) Installation auf einem Unix-Server scheitert, musste hier eine Abwägung zwischen MariaDB und PostgreSQL getroffen werden.

In Anbetracht der Anforderungen und der Anforderung, Geodaten effizient zu speichern und zu verarbeiten, kommt PostgreSQL vorliegend zum Zug. Der Grund dafür ist die Erweiterung *PostGIS*, die auf die Verarbeitung eben solcher Daten spezialisiert ist. Sie erlaubt durch spezielle Anfragen, Daten direkt zu aggregieren und auszuwerten.

5.2 Stammdaten

Die Stammdaten enthalten alle Informationen, die wichtig für die Schulwegplanung sind, aber nicht dauernden Änderungen unterworfen sind. Das sind vor allem die Informationen über die Schulen, ihre Adressen, Standort und Zugehörigkeit zu einem Schulbezirk. Der Schulbezirk muss ebenfalls gespeichert werden, allerdings mit den Koordinaten seines Umrisses. Da die Bezirke sehr groß sein können und die Schulbezirke in der Regel (aber nicht immer) den Stadtbezirken folgen, sind folglich eine große Zahl Koordinaten nötig.

Im Paket `de.htwsaar.zielsicher.init` gibt es eine Komponente `Basedata`, die das Spring-Interface `CommandLineRunner` implementiert. Die Implementierung dieses Interfaces und die Annotation als Komponente sorgt dafür, dass die `run()`-Methode beim Systemstart ausgeführt wird. Dieser Mechanismus wird genutzt, um die Stammdaten einzulesen, sofern sie nicht vorhanden sind.

Die Daten selbst stehen in einer JSON-Datei. Sie enthält alle Schulen zusammen mit den Koordinaten der Bezirksgrenzen, insgesamt 5631 Stück. Mangels offizieller Daten mussten diese Punkte manuell mit QGIS eingezeichnet und ausgewertet werden; tatsächlich bestehen die Polygone, die auf Abbildung 2.3 in Abschnitt 2.3.4 zu erkennen sind, aus diesen manuell eingezeichneten Punkten.

Das JSON-Format selbst ist wie folgt aufgebaut:

```
[
    {
        "name": "Grundschule Saarbrücken Klarenthal",
        "street": "Hauptstraße",
        "houseNo": "53",
        "postalCode": "66127",
        "city": "Saarbrücken",
        "lat": 49.2292474,
        "lon": 6.899320765349371,
        "district": {
            "number": 1,
            "coordinates": [
                { "lat": 49.2451062367, "lon": 6.9001223137 },
                { "lat": 49.2440566236, "lon": 6.8993324679 },
                ...
                { "lat": 49.2452056819, "lon": 6.8995167634 },
                { "lat": 49.2451062367, "lon": 6.9001223137 }
            ]
        }
    },
    ...
]
```

Listing 5.1: Stammdateneintrag der Grundschule Saarbrücken Klarenthal.

Es fällt auf, dass die erste und die letzte Koordinate gleich sind. Das ist nötig, damit eine durchgehende Linie vom Start bis zum Ende gezogen werden kann, um das Polygon zu schließen.

Die Initialisierungsklasse liest all diese Daten ein und legt sie in der Datenbank ab. Zusätzlich werden die *bounding boxes* berechnet und gespeichert, um die Overpass-Abfragen später zu beschleunigen.

5.3 Feature-orientierte Umsetzung der Funktionalität

Hier sollen die Implementierungen einiger Schlüsselfeatures des Systems gezeigt werden.

5.3.1 Overpass

OSM ist, abstrakt betrachtet, zunächst eine Datensammlung. Es erfordert größeren Rechenaufwand, alle Daten manuell zu verarbeiten, was weder sinnvoll noch Aufgabe des Anwendungsentwicklers sein kann.

Die Overpass-API dient dazu, gezielt Daten aus den OSM-Daten zu extrahieren und ggf. aufzubereiten. Dabei kann speziell nach (Daten-)Typen von OSM-Einträgen (`node`, `way`, `relation`) gefiltert, im Falle der letzten beiden Typen können auch Transformationen durchgeführt werden. Overpass erwartet Anfragen dabei in einem eigenen Format, das vom Backend zusammengebaut werden muss:

```
[out:json];
(
    way["highway"]["maxspeed"](49.2427741752,6.9788527993,49.2589316416,7.0005731429);
);
(._;>;);
out body;
```

Listing 5.2: Overpass-Anfrage für Höchstgeschwindigkeiten in einem Bereich.

Zunächst wird das Ausgabe auf JSON festgelegt. Dann werden alle OSM-*Ways* gesucht, die den Tag `highway` und den Tag `maxspeed` besitzen. Als Suchgebiet wird ein Begrenzungsrahmen angegeben. Die nächste Zeile, `(._;>;);`, beschreibt eine Transformation. In dem Kontext bedeutet `._`, dass die zuvor gefundenen Ergebnisse herangezogen werden. `>;` extrahiert alle *Nodes* der gefundenen *Ways*. Die umschließenden Klammern kombinieren das Ergebnis, sodass sowohl die Straßenzüge als auch die einzelnen Knotenpunkte im Ergebnis ausgegeben werden. Die letzte Angabe bedeutet, dass nur Tags und grundlegende Metadaten extrahiert werden, aber keine überflüssigen Informationen wie Zeitstempel oder Benutzerinformationen.

Die Methoden der Geschäftslogik übergeben die gesuchten Informationen und die gewünschte Transformation an eine Klasse `OverpassHelper`, die daraus die passende Anfrage generiert und ausführt. Die Geschäftslogik nimmt dann weitere Transformationen vor, um die Daten für das Frontend aufzubereiten.

5.3.2 Auswertung von Umfragen

Die Umfragen erlauben verschiedene Arten von Fragetypen: Radiobuttons, Dropdown-Menüs, Checkboxen, Nummernfelder und Texteingaben (einzeilig oder mehrzeilig). Texteingaben können nicht automatisch ausgewertet werden.

Zunächst muss das Datenmodell der Fragen beschrieben werden. Wie in Abschnitt 4.3.1 bereits gezeigt, können Fragen mehrere Frageoptionen enthalten, die als exklusiv zu

verstehen sind. Der genaue Aufbau des Komplexes „Fragen und Antworten“ ist Abbildung 5.1 zu entnehmen. Unterschieden werden muss zwischen `response` und `answer`; ersteres beschreibt die Gesamtheit aller Rückmeldungen eines Umfrageteilnehmers, während letzteres eine konkrete Antwort auf eine Frage bezeichnet.

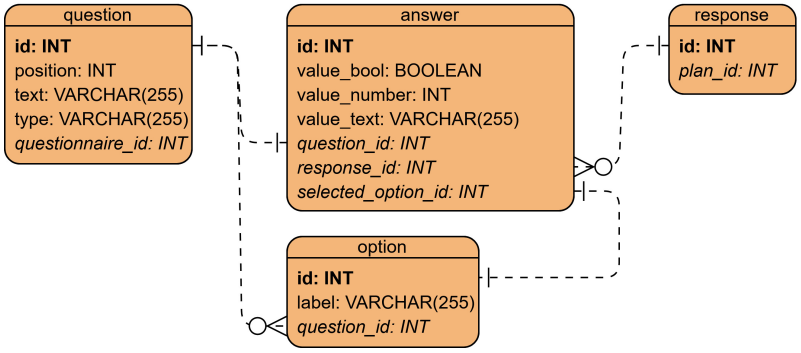

Abbildung 5.1: ER-Diagramm für den Kontext der Fragen und Antworten.

Bei der Auswertung werden also zunächst alle `response` eines Plans abgefragt und danach über alle zugeordneten `answer` iteriert. Dort findet eine Fallunterscheidung nach dem Typ der Frage statt. Radiobuttons, Dropdown-Menüs und Checkboxen können gemeinsam ausgewertet werden, da sie strenggenommen die gleiche Semantik haben.

```
Map<Integer, Long> counts = answers.stream()
    .filter(a -> a.getSelectedOption() != null)
    .collect(Collectors.groupingBy(a -> a.getSelectedOption().getId(),
    ↪  Collectors.counting()
));

List<OptionCount> options = question.getOptions().stream()
    .map(opt -> new EvaluationResponseEntry.OptionCount(
        opt.getLabel(), counts.getOrDefault(opt.getId(), 0L).intValue()
)).toList();
```

Listing 5.3: Auswertung von Radiobuttons, Dropdown-Menüs und Checkboxen.

Listing 5.3 zeigt die Vorgehensweise. Durch die Gruppierung anhand des Primärschlüssels der ausgewählten Fragenoption wird deren Vorkommen gezählt und in einen Teil des Ergebnis-JSON geschrieben, das hier aber aus Platzgründen nicht gezeigt wird.

Bei den Zahlenfeldern ist die Vorgehensweise etwas anders; hier müssen die Vorkommen verschiedener Zahlenwerte gezählt werden, wie Listing 5.4 zeigt.

```
Map<String, Integer> counts = new HashMap<>();
for (Answer answer : answers) {
    if (answer.getValueNumber() != null) {
        String key = String.valueOf(answer.getValueNumber());
        counts.merge(key, 1, Integer::sum);
    }
}
```

Listing 5.4: Auswertung von Zahlenfeldern.

In der Hashmap werden die bisher gefundenen Antworten zusammen mit ihrer Häufigkeit gespeichert. Wird ein weiterer Wert gefunden, wird die Anzahl in der Hashmap um 1 erhöht, ansonsten mit einem Vorkommen neu angelegt.

5.3.3 Heatmaps

Der Code in Listing 5.5 zeigt, wie die Daten für die Heatmaps mithilfe von PostGIS abgefragt werden. Relevant ist hier vor allem die innere `SELECT`-Abfrage, die spezielle Funktionen von PostGIS aufruft.

```
@Query(value = """
    WITH grid_cells (cell_center, point_count) AS (
        SELECT
            ST_Transform(
                ST_SnapToGrid(ST_Transform(geom, 3857), 5),
                4326
            ),
            COUNT(*)
        FROM movementdata
        WHERE response_id IN (:responseIds)
        GROUP BY ST_SnapToGrid(ST_Transform(geom, 3857), 5)
    )
    SELECT
        ROUND(ST_Y(cell_center)::NUMERIC, 6) AS lat,
        ROUND(ST_X(cell_center)::NUMERIC, 6) AS lon,
        point_count
    FROM grid_cells
    ORDER BY point_count DESC
""", nativeQuery = true)
List<Object[]> findHeatmapDataByResponseIds(@Param("responseIds") List<Integer>
↪  responseIds);
```

Listing 5.5: Eine PostGIS-Abfrage für Heatmaps.

Abbildung 5.2: Heatmap für eine Schulumgebung.

`ST_Transform(geom, 3857)` überträgt die Koordinaten aus dem System mit Längen- und Breitengraden in ein anderes Koordinatensystem, die Mercator-Projektion, in der alle Einheiten in Metern gerechnet werden. Das eingeschlossene `ST_SnapToGrid(..., 4326)` nimmt die vorher berechneten Koordinaten und „tackert“ sie in ein Raster mit Zellen, die jeweils $25m^2$ (fünf Meter mal fünf Meter) groß sind. Dadurch erhalten viele Punkte denselben Koordinatenwert, nach denen sich anschließend gruppieren lässt. `ST_Transform(..., 4326)` projiziert die Koordinaten aus dem metrischen Raster wieder zurück in das System mit Längen- und Breitengrad. `COUNT(*)` und `GROUP BY ST_SnapToGrid(...)` zählt abschließend, wie viele Punkte auf jeden Punkt im Raster fallen. Die äußere Abfrage fragt die Informationen nur noch ab und gibt sie zurück. Das Frontend kann diese Punkte dann als Heatmap anzeigen, wie in Abbildung 5.2 zu sehen ist.

5.3.4 Einheitliches JSON-Format

Für das Frontend ist es wichtig, dass die Antworten der REST-Endpunkte immer gleich aufgebaut sind. Wenn es sich darauf verlassen kann, dass alle Antworten gleich sind, wird Codeduplikation vermieden, die Fehlerbehandlung (vgl. Abschnitt 5.3.5, S. 56) vereinfacht und die Erweiterbarkeit sichergestellt. Zu diesem Zweck wurde ein Antwort-Format verwendet, das alle wichtigen Informationen bündelt und dem Frontend flexibel ermöglicht, eventuelle Fehlermeldungen abzufangen.

```java
@JsonInclude(JsonInclude.Include.NON_NULL)
public record Response<T>(
    Meta meta,
    T data
) implements Serializable {}

@JsonInclude(JsonInclude.Include.NON_NULL)
public record Meta(
    boolean success,
    String lastUpdate,
    String error
) implements Serializable {}
```

Listing 5.6: Die generelle Antwort mit dem Format der Metadaten

Listing 5.6 zeigt die verwendete `Response<T>`. Sie bündelt ein Objekt vom Typ `Meta` mit den tatsächlichen Nutzdaten vom Typ `T`. Der Client kann sich darauf verlassen, dass jede Antwort vom Server so aufgebaut ist; selbst die Fehlermeldungen, bei denen der Erfolg der Anfrage und die eventuelle Fehlermeldung in den Metadaten zu finden ist.

Darüber hinaus erleichtert die klare Trennung von Metadaten und Nutzdaten die automatische Verarbeitung: Das Frontend kann stets an der gleichen Stelle prüfen, ob eine Anfrage erfolgreich war, und muss dafür nicht die eigentlichen Nutzdaten interpretieren. Durch die Erweiterbarkeit des `Meta`-Objekts lassen sich zukünftig weitere Informationen wie Paginierung, Warnungen oder Fehlercodes ergänzen, ohne das bestehende Antwortformat zu brechen. Damit bleibt die Schnittstelle stabil und gleichzeitig flexibel für zukünftige Anforderungen.

5.3.5 HTTP-gerechte Exceptions

Kommt es Fehlern jeglicher Art, werden von Spring in aller Regel Exceptions geworfen. Die Gründe können verschieden sein: Ein Namensattribut könnte eine Unique-Constraint haben, das Frontend übergibt bei Pflichtfeldern `null` oder es wird eine Ressource angefordert, die nicht existiert, um nur einige zu nennen. Unbehandelt führen diese Exceptions zu einem `500 - Internal Server Error`, der den aktuellen Stacktrace und die (technische) Formulierung des Fehlers beinhaltet.

Gerade der Fehlercode birgt das Risiko, die wahre Natur des Fehlers zu verschleiern, da der Statuscode 500 oft als „Auffang-Fehlercode“ benutzt wird. Aus diesem Grund sollten Fehler, die an das Frontend zurückgemeldet werden, den korrekten HTTP-Statuscode und eine menschenlesbare Fehlermeldung beinhalten. Zudem sollten auch Fehlermeldungen das Antwortformat beibehalten, das im vorherigen Abschnitt beschrieben wurde, um dem Frontend die Verarbeitung zu vereinfachen.

```java
@ExceptionHandler(EntityNotFoundException.class)
public ResponseEntity<Response<Object>> handleNotFound(EntityNotFoundException ex)
{
    Response<Object> response = new Response<>(this.createMeta(ex.getMessage()));
    return ResponseEntity.status(HttpStatus.NOT_FOUND).body(response);
}
```

Listing 5.7: Exceptionhandler für fehlende Datensätze

Aus diesem Grund wurde die Klasse `GlobalExceptionHandler` eingeführt. Diese Klasse enthält mehrere Exceptionhandler, die jeweils auf eine Familie von Exceptions warten (im Sinne einer Vererbungshierarchie).

Die `@ExceptionHandler`-Annotation von Spring erlaubt es, an beliebiger Stelle in von dem Framework verwalteten Code Exceptions zu werfen, die an zentraler Stelle aufgefangen werden. So würde der Handler in Listing 5.7 alle Exceptions auffangen, die von der Klasse `EntityNotFoundException` erben. Die Unterklassen können vielfältig sein: Konkret implementiert wurden Exceptions für fehlende Bezirke, Fragebögen, Umfragen und Schulen. Durch diese Zentralisierung kann nun immer der korrekte Statuscode (404) zurückgegeben werden. Zusätzlich baut der Handler eine JSON-Antwort im Stil der normalen Antworten, bei der die Fehlermeldung aus den Metainformationen ausgelesen werden kann.

5.3.6 Latenzen

Die Verwendung einer REST-API mithilfe von Spring bringt, neben den in Kapitel 5.1.1 genannten Vorteilen, auch gewisse Nachteile mit sich. Im Vergleich zu einer vorherigen Implementierung (in TypeScript mit NodeJS und Express) waren Abfragen im Mittel um den Faktor 10 schneller, je nach Berechnungsaufwand. Dieser Unterschied lässt sich durch mehrere Ursachen erklären.

Ein grundlegender Unterschied liegt in den beiden verschiedenen Implementierungsansätzen beider Frameworks. NodeJS setzt dabei auf eine Event-getriebene (engl. *event-driven*) Architektur, die Anfragen asynchron in einer Event-Loop verarbeitet; Spring hingegen arbeitet mit blockierendem Multithreading, beispielsweise mit Tomcat als Servlet-Container. Das Erstellen dieser Threads ist aufwendig, sowohl was die Erstellung als auch die Verwaltung angeht, was zu einem Overhead führt [29, 46].

NodeJS mit Express ist zudem leichtgewichtiger und sehr nah mit HTTP und JSON verflochten, während das ganze Ökosystem von Spring Boot auf Reflection, umfangreiche Auto-Konfigurationen, Dependency Injection und Abstraktion bei Datenbankzugriffen durch Hibernate ausgerichtet ist. All dies kostet Rechenzeit, die für jede Anfrage anfällt. Der generelle Overhead durch die JVM kommt an dieser Stelle ebenso dazu [1, 9].

Über die Gründe der Wahl von Spring statt wurde bereits in Abschnitt 5.1.1 geschrieben, aber im konkreten Vergleich überwiegt der größere Funktionsumfang des Spring-Ökosystems (Transaktionen, Sicherheit, umfangreiche Bibliotheken) den Performance-Nachteil deutlich. Trotzdem macht dieser Performance-Nachteil eine Lösung erforderlich, um schnelle Antwortzeiten zu ermöglichen.

Die implementierte Lösung besteht in der Verwendung eines Caches für alle Anfragen, die Datensätze zurückliefern. Dabei stehen insbesondere jene im Fokus, die alle Einträge einer Tabelle zurückliefern (bspw. alle Pläne) oder solche, die statische Daten abfragen (bspw. die Koordinaten eines Polygons, das einen Bezirk umschließt).

```
@Cacheable(value = "districtCoords", key = "#schoolId")
public DistrictDto getDistrictCoordinates(Integer schoolId)
{
    // Koordinaten aus der Datenbank holen und in ein DistrictDto packen
    // ...
    return districtDto;
}
```

Listing 5.8: Beispiel für Caching in einfacher Abfrage statischer Daten

Listing 5.8 zeigt den generellen Aufbau einer gecachten Anfrage aus einem Service (in diesem Fall der `LayersService`). Durch die `@Cacheable`-Annotation wird diese Methode so konfiguriert, dass bei einem Aufruf im Cache mit dem Namen `districtCoords` nach einem Schlüssel `schoolId` gesucht wird. Existiert der Eintrag (cache hit), wird der

Wert aus dem Cache zurückgegeben. Andernfalls (cache miss) wird der Methodenrumpf ausgeführt, der Wert zurückgegeben und in den Cache geschrieben. Bei einem weiteren Aufruf mit demselben Argument wird dann der Cache genutzt.

Werden Daten bearbeitet, muss auch der Cache aktualisiert werden. Dies kann entweder durch die `@CachePut`-Annotation erfolgen oder durch ein `@CacheEvict`, wobei letzteres auch beim Löschen von Daten benutzt wird. Welche Version benutzt wird, hängt im Wesentlichen davon ab, ob es für einen Datensatz zwei verschiedene Caches gibt: einen für alle Daten (bspw. alle Pläne) und einen für die einzelnen Datensätze (bspw. ein Plan, identifiziert über seinen Primärschlüssel). Aufgrund von Limitierungen der Caching-Engine von Spring Boot können in einem Cache, der alle Daten enthält, nicht einzelne Datensätze extrahiert werden; es müssen also zwei Caches verwaltet werden.

Als Caching-Engine wird EhCache verwendet. Die Bibliothek integriert sich nahtlos in Spring und erlaubt über die bereits erwähnten Annotationen eine klare Trennung zwischen Anwendungslogik und Caching-Mechanismen. Neben klassischen Funktionen wie TTL/TTI bietet EhCache auch die Möglichkeit zur Persistenz über Neustarts hinweg, sodass gespeicherte Einträge unmittelbar wieder verfügbar sind und eine Phase zahlreicher *cache misses* vermieden wird. Damit eignet sich EhCache besonders für den hier behandelten Anwendungsfall, bei dem häufig auf unveränderte Stammdaten und statische Geodaten zugegriffen wird [16].

Eine Besonderheit sind die umfangreichen Daten der Overpass-Abfragen. Je nachdem, ob sich die OSM-Daten ändern, erhält man unter Umständen veränderte Overpass-Daten von der API zurück. Würde man dem bekannten Cache-Muster (TTL) folgen, würden somit möglicherweise veraltete Daten im Cache gehalten werden.

```
@Cacheable(value = "speedsCache", key = "@versionFetcher.version + '_' + #districtId")
public Response<List<SpeedsResponse>> overpassSpeeds(Integer districtId)
{
    ...
}
```

Listing 5.9: Caching von Overpass-Daten

Abhilfe wird hier durch einen Mechanismus in der Klasse `OverpassVersionHelper` geschaffen. Dabei handelt es sich um eine geplante, sich endlos wiederholende Aufgabe (engl. *scheduled task*), die einmal pro Stunde die Version der Overpass-Daten von der API abfragt. Diese Version (ein Zeitstempel) wird dann als Schlüssel für das Caching der Overpass-Anfragen verwendet, wie in Listing 5.9 zu erkennen. Damit sind Daten maximal um eine Stunde veraltet; zur zusätzlichen Kontrolle wird im Frontend dieser Zeitstempel auch angezeigt.

5.3.7 Weitere Funktionalitäten

Es gibt weitere Funktionalitäten, deren Implementierung hier aber nicht gezeigt wird. Der Grund dafür ist, dass es sich um simple Speicher- und Abfrage-Vorgänge handelt. Das betrifft folgende Funktionalität:

- Erfassung von Gefahrenstellen und Schulwegen
- Einzeichnen von Routen und Setzen von POI
- Anlage Fragebögen

Diese Funktionalitäten nehmen lediglich das JSON des Frontends an und legen es in der Datenbank ab. Natürlich gibt es einige Transformationen, nicht aber auf einem Komplexitätslevel, das hier einer weiteren Erläuterung bedarf.

Außerdem wurde die Lombok-Bibliothek benutzt, um Boilerplate-Code zu reduzieren.

5.4 Teststrategien

Folgend wird beschrieben, welche Teststrategien umgesetzt wurden. Das Ziel ist die systematische Testung der Funktionalität und Zuverlässigkeit, aber auch die Einhaltung von Architekturanforderungen.

5.4.1 Unit-Tests

Unit-Tests prüfen kleinste Funktionseinheiten wie Methoden oder Klassen isoliert von dem größeren Zusammenhang dieser Elemente. Auf diese Weise können Fehler sehr schnell gefunden werden, bevor sie sich auf das Gesamtsystem auswirken und möglicherweise schwerer zu entdecken sind. Dadurch, dass mit solchen Tests (bestenfalls) alle Methoden getestet werden, kann eine hohe Codeabdeckung erreicht werden. Da Unit-Tests klein, zahlreich und sehr spezifisch sind, wird hier auf ein Beispiel verzichtet; verwendet wird das verbreitete JUnit-Framework.

5.4.2 Integrationstest

Im Gegensatz zu Unit-Tests verfolgen Integrationstests das Ziel, das Zusammenspiel mehrerer Komponenten zu überprüfen. Sie setzen dort an, wo einzelne Module oder Klassen miteinander interagieren und Datenflüsse oder Kontrollstrukturen über Modulgrenzen hinweg stattfinden. Dies mag auf den ersten Blick redundant wirken, da die Methoden implizit mitgetestet werden. Allerdings sind Integrationstests nicht spezifisch genug, um Fehler in einzelnen Methoden oder Ähnlichem zu finden.

In Integrationstests können beispielsweise die Kommunikation zwischen Services und Repositories oder die Funktionsfähigkeit einer REST-API in Bezug auf die darunter liegende Datenbank geprüft werden. Dabei können durchaus reale Systeme benutzt werden, um ein realistisches Testszenario zu schaffen. Somit lassen sich Integrationstests in drei Punkten von den Unit-Tests abgrenzen.

Ziel: Integrationstests testen größere Teilsysteme, Unit-Tests die Korrektheit kleiner Codeteile.

Testobjekte: Unit-Tests prüfen einzelne Methoden oder Klassen, Integrationstests deren Zusammenarbeit.

Abhängigkeiten: Unit-Tests können externe Komponenten simulieren („mocken“), während Integrationstests reale Abhängigkeiten oder Module nutzen.

Integrationstests sind aufwendiger als Unit-Tests, haben aber eine höhere Aussagekraft über die Funktionsfähigkeit des Gesamtsystems.

5.4.3 ArchUnit

Die beiden zuvor erläuterten Teststrategien hatten die Logik des Systems zum Gegenstand, also ob alle Komponenten die von ihnen erwarteten Anforderungen erfüllen. Allerdings ist die Funktionalität nicht die alleinige Anforderung an das System: Nichtfunktionale Anforderungen wie Erweiterbarkeit und Wartbarkeit gehören ebenfalls dazu.

Diese Aspekte sind nicht unmittelbar über klassische Unit- oder Integrationstests messbar, da sie nicht auf das Verhalten einzelner Methoden oder Komponenten abzielen. Vielmehr resultieren sie aus strukturellen Eigenschaften der gewählten Softwarearchitektur. Eine konsistente Einhaltung von Schichtgrenzen, klar definierte Abhängigkeiten sowie das Vermeiden zyklischer Strukturen sind Kriterien, die maßgeblich die Wartbarkeit und Erweiterbarkeit beeinflussen. Ohne eine technische Unterstützung besteht jedoch die Gefahr, dass solche Vorgaben im Laufe der Weiterentwicklung verletzt werden, beispielsweise durch unbewusste Einführung unerwünschter Abhängigkeiten zwischen Komponenten.

Zur Überprüfung solcher Architekturvorgaben bietet sich der Einsatz der Bibliothek ArchUnit an. ArchUnit ist ein in Java geschriebenes Framework, das es erlaubt, Architekturregeln in Form von Testfällen auszudrücken. Diese Testfälle werden wie reguläre Unit-Tests ausgeführt und können so kontinuierlich sicherstellen, dass die Architekturregeln eingehalten werden.

```
@Test
void controllerServiceRule()
{
    ArchRule rule = ArchRuleDefinition.classes()
        .that()
        .resideInAPackage("..controller..")
        .should()
        .onlyAccessClassesThat()
        .resideOutsideOfPackages(
            "..components..", "..repository..", "..config..", "..exception..", "..init..",
            ↪  "..model..", "..repository..", "de.htwsaar.zielsicher.util..");

    rule.check(projectClasses);
}
```

Listing 5.10: Ein ArchUnit-Test.

Beispielsweise kann festgelegt werden, dass Klassen aus dem Paket `controller` nur auf Klassen des Paketes `service` und denen von Java und Spring selbst zugreifen dürfen. Ein entsprechender ArchUnit-Test ist in Listing 5.10 zu sehen.

5.5 Zusammenfassung

Es wurde gezeigt, dass es durch verschiedene Technologien möglich war, die Kernfunktionalitäten des zu entwickelnden Backends zu implementieren. Dabei nimmt das Spring-Framework eine wichtige Rolle ein, um Abläufe zu vereinfachen, Abhängigkeiten zu reduzieren und die Wartbarkeit des Systems zu erhöhen. Außerdem wurde PostgreSQL ausgewählt, um die Geodatenerweiterung PostGIS nutzen zu können.

Die wichtigsten Kernfunktionalitäten sind die Abfragen von Verkehrsdaten über Overpass, wofür ein Mechanismus gezeigt wurde, der die nötige Abfragesprache generiert. Die automatische Auswertung von Umfragen macht sich Streams und ein hierarchisches Datenmodell zunutze, um effizient Antworten zu zählen. Die Heatmaps werden mit der Unterstützung von PostGIS-internen Funktionen erzeugt, ohne dabei auf eine langsamere Java-Implementierung zurückgreifen zu müssen.

Zudem sorgt ein einheitliches JSON-Format unter Benutzung der leichtgewichtigen Records für Konsistenz bei den Antworten der REST-API, was die Erweiterung des Frontends vereinfacht; zusammen mit den HTTP-gerechten Exceptions wird zudem dafür gesorgt, dass es mit jeder Fehlermeldung des Backends arbeiten kann.

Der Tatsache, dass ein so umfangreiches Framework wie Spring im Geschwindigkeitsvergleich mit leichteren Frameworks im Nachteil ist, wird durch den effizienten Einsatz von Caches begegnet, die die Ergebnisse komplexer Operationen zwischenspeichern.

Zuletzt wurden noch die Teststrategien vorgestellt, die aus einer Kombination aus Unit-Tests, Integrationstests und ArchUnit-Tests bestehen.

6 Evaluation

In der Evaluation soll untersucht werden, ob das entwickelte Backend den Anforderungen an die Softwarequalität genügt und inwieweit die erarbeiteten Anforderungen umgesetzt wurden.

6.1 Softwarequalität

Einen möglichen Rahmen für die Evaluation der Softwarequalität bietet die Norm ISO 25010. Sie listet viele Qualitätsmerkmale auf, die für eine Evaluation eines Systems von Bedeutung sind:

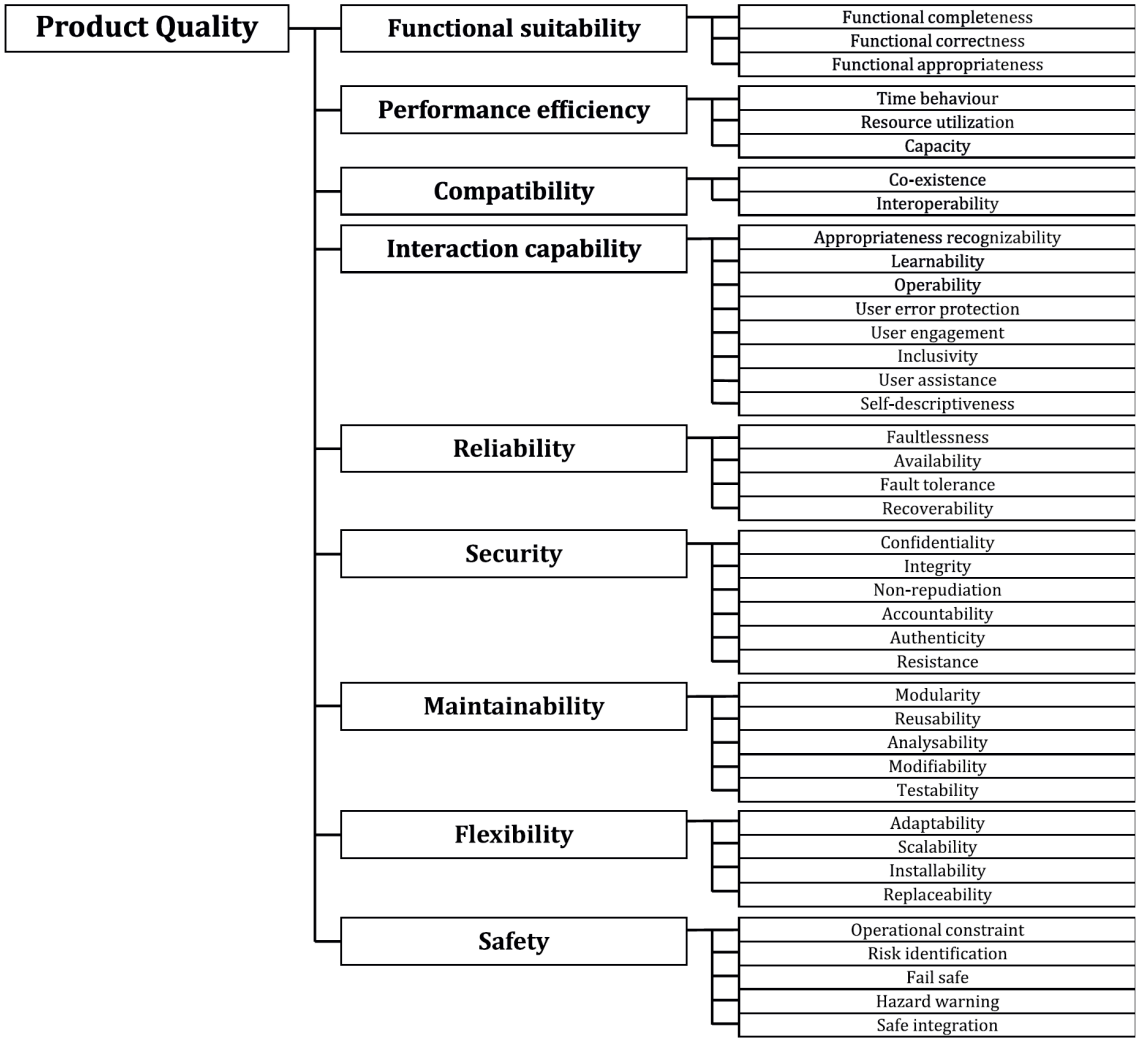

Abbildung 6.1: Qualitätskriterien nach ISO 25010 [27, S. 10]

Eine genaue Erläuterung aller Punkte und Unterpunkte würde den Rahmen der Arbeit an dieser Stelle deutlich sprengen. Alle relevanten Punkte werden in der nun folgen-

den Bewertung der Softwarequalität erklärt. Allerdings sollen in diesem Abschnitt nur die Punkte besprochen werden, die sich aus den nicht-funktionalen Anforderungen in Kapitel 3 ergeben; die Erfüllung der funktionalen Anforderungen wird weiter unten in Abschnitt 6.3 betrachtet. Die Interaktionsfähigkeit (vgl. Abbildung 6.1) wird in der vorliegenden Arbeit nicht besprochen, da sie sich auf eine Benutzeroberfläche und damit das Frontend bezieht. Demnach bleiben für diesen Teil der Evaluation fünf Kategorien aus der obenstehenden Abbildung 6.1: Performance, Zuverlässigkeit, Sicherheit, Wartbarkeit und Flexibilität.

6.1.1 Performance

Das wichtigste Kriterium für die Performance war, dass Anfragen ohne spürbare Verzögerung beantwortet werden sollen, wobei eine halbe Sekunde für komplexe Anfragen als Höchstgrenze anzusehen ist.

In Abschnitt 5.3.6 wurde ein Caching-System vorgestellt, das es erlaubt, Ergebnisse von komplexen Anfragen oder größere Ergebnisse zwischenzuspeichern. Die dort besprochenen Geschwindigkeitsnachteile von Spring gegenüber anderen Web-Frameworks konnten damit erfolgreich verhindert werden. Durch den Cache konnten die Antwortzeiten um bis zu 95% reduziert werden und erfüllen die Anforderungen an die Antwortzeit. Allerdings dauert die erste Anfrage, also der *cache miss*, deutlich länger; dieses Problem könnte man noch beheben, indem während der Initialisierung des Servers alle Daten einmal abgefragt und in den Cache gelegt werden.

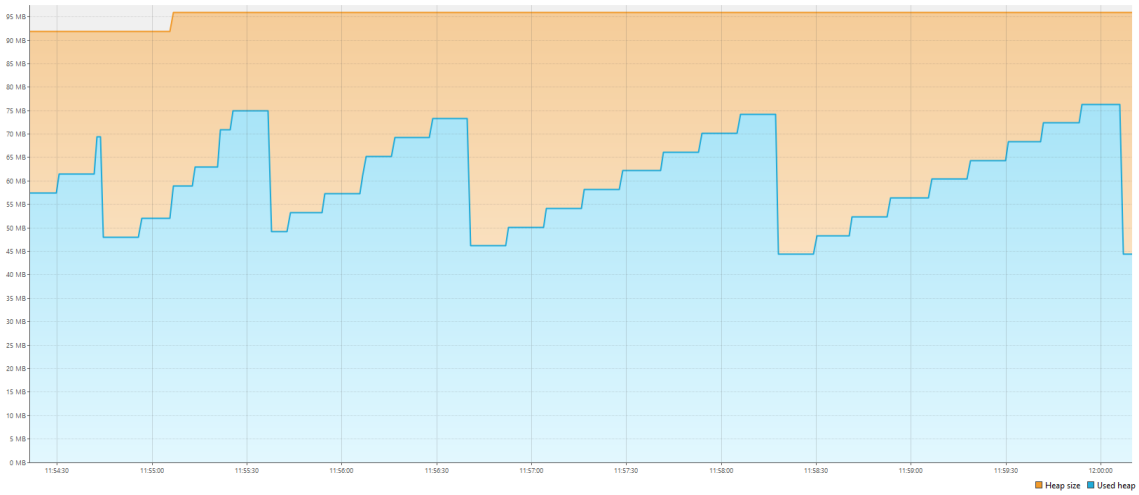

Abbildung 6.2: Speicherverbrauch in einem typischen Anwendungsszenario (Planabfrage, Umfrageergebnisse, Heatmap); ohne Caching.

Im Bereich der Ressourcennutzung hat das System den für ein Spring-System typischen Speicherverbrauch, nimmt aber nicht übermäßig viel ein. Lediglich das ressourcenintensive Rendern einer Karte lässt den Speicherverbrauch kurzzeitig größer werden, bevor die Speicherbereinigung eingreift. Im Allgemeinen sieht der Speicherverbrauch aber so aus wie in Abbildung 6.2 dargestellt.

6.1.2 Zuverlässigkeit

Das Backend läuft auf einem Server in einem Rechenzentrum, das 99% Erreichbarkeit verspricht. Im Laufe der Entwicklung konnte nicht festgestellt werden, dass der Server

einmal nicht erreichbar war; allerdings hängt dies natürlich immer von dem konkreten Server bzw. dessen Anbieter ab.

Im Fehlerfall werden Exceptions vom globalen Exceptionhandler abgefangen, der in Abschnitt 5.3.5 vorgestellt wurde, und mit einer entsprechenden Fehlermeldung quittiert. Diese Fehler führen aber nicht dazu, dass das System abstürzt, sondern eine Fehlermeldung in den Log geschrieben wird und es weiterläuft. Vollständige Abstürze sind während der Entwicklung aufgetreten, waren aber auf Fehler zurückzuführen, die selbst für Spring nicht mehr zu handhaben waren.

6.1.3 Sicherheit

Als fundamentaler Punkt wurde in der Anforderungsanalyse die Datensicherheit gemäß der CIA-Prinzipien und der DSGVO herausgearbeitet.

Die Daten der Kinder werden anonymisiert gespeichert. Von der Client-Seite aus besteht durch die Session-ID keine Möglichkeit, Antworten anderen Kindern zuzuordnen; auf der Serverseite ist zwar eine Zuordnung der Session-ID zu den Antworten der Kinder möglich, aber ein Rückschluss auf einzelne Kinder nicht, da hier das Prinzip der Datenminimierung verfolgt wird. Somit ist der Datenschutz gewährleistet.

Die Vertraulichkeit der Daten wird durch einen Login gewährleistet. Die Integrität der Daten wird alleine schon durch die Übertragung mit HTTPS (und den darunterliegenden OSI-Schichten) gewährleistet; die Verfügbarkeit wurde bereits im Rahmen der Zuverlässigkeit besprochen.

Neue Datensätze oder Änderungen an bestehenden Datensätzen werden im Log protokolliert; somit ist auch das Prinzip der Nachvollziehbarkeit gewahrt.

6.1.4 Wartbarkeit

Die Wartbarkeit unterteilt sich in mehrere Bereiche, von denen hier die Modularität, die Veränderbarkeit (Anpassbarkeit, Erweiterbarkeit) und die Testbarkeit besprochen werden sollen.

Das Projekt ist modular aufgebaut; die Controller-Klassen sind unabhängig von der konkreten Implementierung der Service-Klassen, abgesehen von der Schnittstelle zwischen beiden. Die Datenbank ist durch Hibernate von der Geschäftslogik in den Services abstrahiert und kann somit jederzeit durch eine andere ersetzt werden.

Durch die Trennung innerhalb der Projektstruktur können Änderungen vorgenommen werden, ohne die gesamte Architektur umschreiben zu müssen. Auch neue Funktionen lassen sich ohne weitere Änderungen hinzufügen, beispielsweise neue Controller oder neue Endpunkte.

Die Testbarkeit wird durch einen entsprechenden Aufbau der Methoden gewährleistet. Jede Methode hat nur eine Zuständigkeit und ist somit durch die in Abschnitt 5.4 besprochenen Tests testbar.

6.1.5 Flexibilität

Bei der Flexibilität sind vor allem die Skalierbarkeit und die „Bereitstellbarkeit“ (engl. *installability*) relevant.

Die vertikale Skalierbarkeit, also die Erhöhung der Ressourcen, ist naturgemäß gegeben. Eine Auswirkung ist aufgrund des niedrigen Speicherverbrauchs allerdings fraglich, da der Speicherverbrauch auch bei der Benutzung außerhalb einer Testumgebung keine Ausmaße erreichen sollte, die eine solche Skalierung nötig machen würden.

Das System ist mit einigen Anpassungen horizontal skalierbar. Es lassen sich ohne Weiteres neue Instanzen des Backends starten; allerdings muss in einem solchen Fall ein Lastverteiler vorgeschaltet und eine gemeinsame Datenhaltung geschaffen werden. Eine Notwendigkeit für eine solche Skalierung hat sich bislang noch nicht ergeben, allerdings fehlen Informationen über den Betrieb im Einsatz (siehe folgendes Kapitel).

Die Bereitstellbarkeit wurde getestet und lässt sich einfach und ohne tiefere technische Kenntnisse durchführen, einen Server vorausgesetzt. Das gemeinsame Deployment in dem Sinne, dass das Backend das Frontend bereitstellt, wurde zum Zeitpunkt dieser Arbeit noch nicht umgesetzt, ist aber geplant und durchführbar.

6.1.6 Zusammenfassung

Die Evaluation der Softwarequalität hat gezeigt, dass die definierten nicht-funktionalen Anforderungen weitgehend erfüllt werden. Im Bereich der Performance konnten durch den Einsatz eines Caches die Antwortzeiten reduziert werden, sodass die Vorgabe von maximal einer halben Sekunde eingehalten wird, mit Ausnahme des ersten Zugriffs.

Die Zuverlässigkeit des Systems hängt von der Verfügbarkeit des Serverbetreibers ab; während der Entwicklung traten keine nennenswerten Ausfälle auf. Fehler werden durch einen globalen Exceptionhandler abgefangen und führen nicht zu einem Absturz des Systems.

Hinsichtlich der Sicherheit sind die Anforderungen der DSGVO erfüllt. Daten werden anonymisiert gespeichert, vertraulich übertragen und Änderungen protokolliert. Somit ist auch die Nachvollziehbarkeit gewährleistet.

Die Wartbarkeit des Systems ist durch eine Trennung der Komponenten und den modularen Aufbau sichergestellt. Anpassungen und Erweiterungen können ohne tiefgreifende Eingriffe in die bestehende Architektur umgesetzt werden. Zusätzlich wurde durch die Struktur der Methoden eine gute Testbarkeit erreicht.

Bei der Flexibilität konnte gezeigt werden, dass das System sowohl vertikal als auch horizontal skalierbar ist, auch wenn ein Bedarf dafür bislang nicht vorliegt. Die Bereitstellung gestaltet sich unkompliziert und kann ohne tiefere technische Kenntnisse durchgeführt werden.

6.2 Feldtests und qualitative Analyse

Aus zeitlichen Gründen sind ausgedehnte Feldtests zum Zeitpunkt des Abschlusses der vorliegenden Arbeit nicht mehr möglich; alle Funktionalitäten, die GPS erfordern, wurden simuliert.

Feldtests sind aber in jedem Fall eine sinnvolle Möglichkeit der Evaluation. Dabei ist denkbar, von jeder Nutzergruppe einen kleinen Teil auszuwählen und das System unter kontrollierten Bedingungen testen zu lassen. Die Vorgehensweise wäre dabei wie folgt:

1. Gezielte Anwerbung von Testpersonen (Schüler und Eltern) innerhalb eines Schulbezirkes. Dabei sollten die Testpersonen so ausgewählt werden, dass ein möglichst großer Bereich des Schulbezirkes erfasst, aber gleichzeitig auch ein gewisses Maß an Überlappung erreicht wird. Auf diese Weise soll die Visualisierung der aufgezeichneten Routen getestet werden.

2. Die Testpersonen führen die Befragung, Meldung von Gefahrenstellen und die Routenaufzeichnung durch. Die Testpersonen sollten dabei auch dazu angehalten

werden, Fehleingaben zu machen, um die Fähigkeit des Systems zu testen, mit ungültigen Eingaben umzugehen. Auch die Unterbrechung bzw. der Abbruch einer Befragung oder Aufzeichnung sollte gezielt getestet werden.

3. Der Planer wertet die eingehenden Informationen aus. Dabei testet er die Planungsfeatures mehrmals, jeweils zu verschiedenen Zeitpunkten der Befragung; so soll der Umgang mit unvollständigen oder wenigen Datensätzen getestet werden. Der Planer soll auch hier gezielt versuchen, dem System Fehleingaben aufzuzwingen, um auch hier den Umgang damit zu testen. Ein Export eines fertigen Planes beendet den Test.

Diese Vorgehensweise sollte mit mehreren Schulbezirken durchgeführt werden, da sie sich in Verkehrsführung und Topografie voneinander unterscheiden. So kann das Verhalten der Benutzer unter verschiedenen Bedingungen beobachtet werden.

Bei einer anschließenden Befragung der Benutzergruppen steht die Frage im Vordergrund, wie das System wahrgenommen wird. Diese Fragestellung besteht aus folgende Aspekten, die abgefragt werden sollen:

- Benutzerfreundlichkeit
- Verständlichkeit
- Verhalten bei Fehleingaben
- Performance
- Güte des Ergebnisses (der fertige Plan)
- Vergleich zum papierbasierten Prozess

Als wichtigste Methoden der Befragung sind hier direkte Interviews, Fragebögen oder offene Fragen, bei denen die Benutzer ohne Rahmenbedingungen ihre Erlebnisse schildern können, möglich.

Als erster Schritt ist geplant, das Gesamtsystem den für die Planung zuständigen Mitarbeitern der Stadt Saarbrücken vorzuführen und deren Feedback entgegenzunehmen. Während eines Treffens im Juli wurde schon eine erste Vorführung des Gesamtsystems gezeigt, allerdings auf einem Entwicklungsstand. Dabei wurde zwar naturgemäß nur das Frontend besprochen, aber ein funktionierendes Frontend kommt nicht ohne funktionierendes Backend aus. Die Rückmeldungen waren ausgesprochen positiv, kamen aber nicht von unmittelbar an der Planung beteiligten Personen.

6.3 Umsetzung der Anforderungen

An dieser Stelle soll nun geprüft werden, in wie weit die in Kapitel 3 erhobenen Anforderungen umgesetzt werden.

6.3.1 Funktionale Anforderungen

Wie in Tabelle 6.1 zu erkennen ist, wurde der weitaus größte Teil der funktionalen Anforderungen erfüllt. Anforderungen, bei denen in der rechten Spalte „(Frontend)“ angemerkt wurde, benutzen zwar Technologie des Backends, beziehen sich aber mehr auf Anzeigeelemente und Bedienbarkeit, die in der vorliegenden Arbeit keine Rolle spielen. Da über deren Status im Bezug auf die Akzeptanzkriterien hier keine Aussage getroffen werden

kann, wird an dieser Stelle nicht weiter darauf eingegangen. Die zugrunde liegende Funktionalität des Backends wird aber noch von anderen Komponenten benötigt, sodass diese Punkte mittelbar durch deren anderen funktionalen Anforderungen abgedeckt sind.

Als einzige funktionale Anforderung wurde FR13 (Export) nicht vollständig umgesetzt. Statt eines PDF liefert das Backend ein Bild im PNG-Format, hochskaliert auf DINA4. Hintergrund war an dieser Stelle der Gedanke, dass der fertige Plan nicht nur die Karte enthält, sondern ggf. noch das Corporate Design der zuständigen Planungsbehörde oder der Stadt. Denkbar wäre auch die Angabe einer Legende oder weiteren Hinweisen. Deshalb wird aus Gründen der Flexibilität nur ein Bild zurückgegeben, das die Planer dann manuell in ein Dokument einfügen können.

FR-Nr.	**Beschreibung**	**Akzeptanzkriterien erfüllt**
FR01	Teilnahme an Umfragen	ja
FR02	Erfassung von Gefahrenstellen	ja
FR03	Aufzeichnung des Schulwegs	ja
FR04	Benutzerfreundlichkeit auf mobilen Geräten	(Frontend)
FR05	Anlage von Plänen	ja
FR06	Übersicht	(Frontend)
FR07	Planungsansicht	(Frontend)
FR08	Radwege	(Frontend)
FR09	Umfrageergebnisse	ja
FR10	Routenaufzeichnungen	ja
FR11	Verkehrsdaten	ja
FR12	Einzeichnen von Routen	ja
FR13	Export	teilweise
FR14	POI	ja
FR15	Platzierung von POI	ja

Tabelle 6.1: Die funktionalen Anforderungen und ihr Status.

6.3.2 Nicht-funktionale Anforderungen

Bei den nicht-funktionalen Anforderungen zeigt sich in Tabelle 6.2 ein ähnliches Bild wie schon zuvor bei den funktionalen Anforderungen: Der größte Teil wurde erfüllt. NR03 (Benutzbarkeit) hat wieder einen Bezug zum Frontend und wird deswegen hier nicht weiter diskutiert. NR04 (Antwortzeit) wurde teilweise erfüllt, wie bereits dargelegt wurde (vgl. Abschnitt 6.1.1, S. 64); da aber bei der ersten Anfrage das Limit nicht eingehalten wird, wurde die Anforderung streng genommen nicht ganz erfüllt; aus diesem Grund wird sie in der Tabelle nicht mit „ja“ geführt.

NR-Nr.	Beschreibung	Akzeptanzkriterien erfüllt
NR01	Datenschutz	ja
NR02	Sicherheit	ja
NR03	Benutzbarkeit	(Frontend)
NR04	Antwortzeit	teilweise
NR05	Auflösung von Kartendetails	ja
NR06	Deployment	ja
NR07	Wartbarkeit	ja

Tabelle 6.2: Die nicht-funktionalen Anforderungen und ihr Status.

Insgesamt wurden damit die wesentlichen funktionalen Anforderungen umgesetzt; lediglich beim Export bestehen begründete Einschränkungen. Die nicht-funktionalen Anforderungen wurden größtenteils erfüllt, wobei die Antwortzeit die definierten Kriterien nur teilweise erreicht.

6.4 Zusammenfassung

Die vorhergegangene Evaluation hat gezeigt, dass das entwickelte Backend die gesetzten Ziele weitgehend erreicht. Bei der Softwarequalität wurde gezeigt, dass die gängigsten Kriterien der ISO 25010 erfüllt wurden. Das System läuft zuverlässig, die Sicherheit wird gewährleistet. Zudem ist es durch den modularen Aufbau des Projektes und die lose Kopplung der Komponenten einfach zu erweitern und abzuändern. Das Deployment ist als Gesamtpaket möglich, wurde aber noch nicht umgesetzt. Lediglich bei der horizontalen Skalierungen sind Anpassungen nötig, allerdings außerhalb des Systems. Bei der Performance werden die gewünschten Antwortzeiten erreicht, solange der Cache gefüllt ist: die erste Anfrage wird also zwangsläufig den Cache nicht benutzen.

Aus zeitlichen Gründen konnten keine Feldtests durchgeführt werden. Stattdessen wurde GPS-Daten simuliert und das System darauf basierend auf Funktionsfähigkeit geprüft. Allerdings wurde ein tragfähiges Konzept für künftige Tests erarbeitet, das eine Erhebung einer qualitativen Analyse ermöglicht und beim weiteren Ausbau des Systems über den Prototypen hinaus helfen kann.

Bezüglich der Anforderungen wurden nahezu alle funktionalen und nicht-funktionalen Anforderungen erfüllt. Ausnahme bildet hier der Export, der aus Gründen der Flexibilität für Planer eine Bilddatei anstelle einer PDF generiert. Außerdem wurde die nicht-funktionale Anforderung der Antwortzeit nicht in allen Szenarien erfüllt.

Insgesamt wurde ein stabiles und erweiterbares System geschaffen, das die zentralen Anforderungen erfüllt und als Grundlage für weitere Entwicklung eine gute Basis darstellt.

7 Abgrenzung von verwandten Arbeiten

In Abschnitt 3.6 wurden andere Lösungen für vergleichbare Anwendungsfälle vorgestellt. Anhand der dortigen Tabelle 3.6 ist erkennbar, dass keine der vorgestellten Lösungen alle Anforderungen erfüllt, auch wenn es einige Überschneidungen gibt.

Der Beobachtung, dass wesentliche Anforderungen nicht von den bestehenden Lösungen abgedeckt sind, wurde bei der Entwicklung des Backends Rechnung getragen. Es versucht also, in die Lücken zu stoßen, die andere Arbeiten offen gelassen haben.

Die Möglichkeit, eigene Icon-Sammlungen anzulegen, wird von anderen Anbietern ebenfalls angeboten, grenzt sich aber von den dortigen proprietären Lösungen ab. Darüber hinaus verfügt das Backend über die Möglichkeit, Planungsübersichten anzuzeigen und nach bestimmten Kriterien zu filtern oder zu sortieren; eine Funktionalität, die in den untersuchten Lösungen nicht in dieser Form vorhanden ist.

Die digitalen Umfragen wurden, auf Basis der verfügbaren Informationen, nur von einer anderen Lösung unterstützt. Die Kombination aus digitalen Umfragen und deren automatischer Auswertung ist ein wesentliches Merkmal des entwickelten Backends und eine der wichtigsten funktionalen Anforderungen, da der Verwaltungsaufwand für die papierbasierte Auswertung bislang einen erheblichen Teil der Planungszeit beansprucht hat.

Auch im Bereich der Verkehrsdaten hebt sich die Lösung ab. Zwar verfügen mehrere Systeme über entsprechende Funktionen, allerdings in unterschiedlicher Ausprägung. Die hier umgesetzte Kombination aus Geschwindigkeitsbegrenzungen, Straßenübergängen und Bushaltestellen stellt im Vergleich ein Alleinstellungsmerkmal dar.

Der wohl wichtigste Faktor zur Abgrenzung sind jedoch die entwickelten Heatmaps. Andere Lösungen zeigen die aufgezeichneten oder eingezeichneten Schulwegrouten lediglich in Form sich überlagernder Linien. Eine präzise Auswertung der Verdichtung von Wegen, wie sie anhand der Heatmaps ermöglicht wird, findet sich zum Zeitpunkt dieser Arbeit in keiner anderen Lösung.

Neben den funktionalen Abgrenzungen weist das entwickelte Backend auch auf der Ebene der nicht-funktionalen Anforderungen Stärken auf. Es ist modular aufgebaut, erweiterbar und erfüllt die Anforderungen an Datenschutz und Datensicherheit gemäß DSGVO. Eine Abgrenzung ist hier nicht möglich, da die Backend-Implementierungen der verwandten Arbeiten nicht einsehbar sind.

Somit ist das entwickelte Backend genau auf die erhobenen Anforderungen zugeschnitten und schließt Lücken, die verwandte Arbeiten offenlassen. Die Ziele aller untersuchten Benutzergruppen werden erfüllt und schafft es dabei, sich funktional von den anderen Lösungen abzugrenzen.

8 Zusammenfassung und Ausblick

Im abschließenden Kapitel werden die Ergebnisse der Arbeit zusammengefasst, Potenziale für Verbesserungen und Erweiterungen beleuchtet und ein Gesamtfazit gezogen.

8.1 Rückblick auf Zielerreichung

Das Hauptziel der vorliegenden Arbeit war die Konzeption und Umsetzung eines Backends zur Unterstützung der digitalen Schulwegplanung. Es sollte vor allem dazu beitragen, die bisher papierbasierte Erhebung und Auswertung durch ein einfach zu bedienendes und zuverlässiges System zu ersetzen. Das Backend soll in der Lage sein, Daten zu erfassen, zu strukturieren und so aufzubereiten, dass ein geeignetes Frontend den Planer bei seiner Arbeit unterstützen kann. Um dieses Gesamtziel zu erreichen, wurden einige Fragestellungen formuliert, deren Beantwortung nun vorgenommen werden soll.

Die Anforderungen (funktional und nicht-funktional) wurden systematisch durch eine menschenzentrierte Anforderungsanalyse nach ISO 9241-210 ermittelt. Als relevante Nutzerrollen wurden dabei Schulwegplaner, Schüler/Eltern und Mitarbeiter der Schule identifiziert. Anhand von Personas wurden User Needs extrahiert, die von den erhobenen Anforderungen aufgegriffen und zusammengefasst wurden, um möglichst viele Bedürfnisse einzubeziehen. Dabei wurde ermittelt, dass die Reduktion der papierbasierten Verarbeitung von Fragebögen im Vordergrund steht. Weitere Aspekte wurden ebenfalls herausgearbeitet, wie beispielsweise eine benutzerfreundliche Oberfläche, hohe Anforderungen an den Datenschutz und die Möglichkeit, Pläne direkt in einem Gesamtsystem zu erstellen, ohne die Notwendigkeit weiterer Software.

Aus den ermittelten Anforderungen ergibt sich auch die Antwort auf eine weitere Fragestellung. Durch die Bereitstellung eines Systems, das Umfragen anbieten und auswerten kann, Verkehrsdaten automatisch abfragt und dem Planer die aufgezeichneten Routen der Kinder in übersichtlicher Form darstellt, entfällt ein immenser Arbeitsaufwand und macht die Schulwegplanung insgesamt effizienter.

Erhobene Daten werden mittels eines durchdachtes Datenmodells sinnvoll gespeichert. Abhängigkeiten zwischen den Daten sorgen dafür, dass es keine verwaisten Datensätze geben kann; außerdem werden für den Benutzer relevante Informationen so aufbereitet, dass alle nötigen Informationen zurückgegeben werden, ohne ihn mit zu vielen Daten zu überfordern. Schulwege werden als Heatmap dargestellt, damit einfach zu erkennen ist, wo wie viele Schüler entlang laufen. Um diesen Prozess zuverlässig zu gestalten, wurde mit PostGIS eine Datenbank benutzt, die auf die effiziente Verarbeitung von Geoinformationen ausgelegt ist.

Die effiziente Implementierung eines Backends ist durch die Nutzung des Spring-Frameworks gelungen. Es bietet durch seine umfassenden Möglichkeiten zur Datenhaltung, Abstraktion und loser Kopplung einen starken Rahmen für die Umsetzung einer REST-API. Es erlaubt eine Strukturierung der Endpunkte und Trennung von der Geschäftslogik und Datenhaltung, wodurch langfristig die Wartbarkeit und Erweiterbarkeit gewährleistet wird.

Dem bereits angesprochenen Datenschutz wird durch eine konsequente Nutzung von HTTPS und der Einhaltung des CIA-Modells Rechnung getragen. Umfragedaten und

Routenaufzeichnungen werden so gespeichert, dass eine nachträgliche Identifizierung der Kinder nicht möglich ist. Eine Zugriffskontrolle soll sicherstellen, dass nur befugte Benutzer Einsicht in die Daten haben. Auch ist es ohne direkten Datenbankzugriff nicht möglich, Einsicht in einzelne Datensätze zu erhalten, da alle Daten nur in aggregierter Form zur Darstellung bereitgestellt werden.

Insgesamt lässt sich also festhalten, dass die Arbeit alle Fragestellungen beantworten konnte. Es wurde ein effizientes, stabiles, wartbares System geschaffen, das die Schulwegplanung in das digitale Zeitalter hebt.

8.2 Verbesserungspotenzial

Auch wenn die wesentlichen Anforderungen umgesetzt wurden, gibt es verschiedene Punkte, an denen das Backend in zukünftigen Arbeiten weiterentwickelt werden kann. Im Kontext des Verbesserungspotenzials ist hierbei eine Änderung *bestehender* Funktionalität gemeint. Das Hinzufügen *neuer* Funktionalität wird im folgenden Kapitel im Rahmen der Erweiterungsmöglichkeiten diskutiert.

Ein erster Bereich betrifft den Export: Aktuell wird lediglich ein Bild im PNG-Format zurückgegeben. Eine direkte PDF-Ausgabe mit zusätzlichen Gestaltungsmöglichkeiten wie Legenden, standardisierten Layouts oder einem Corporate Design könnte den Exportprozess vereinfachen und die Nachbearbeitung reduzieren. Hier wären Vorlagen denkbar, auf die die Karte gedruckt wird. Auf diese Weise ließe sich die erzeugte Ausgabe stärker an die Anforderungen von Schulen oder Kommunen anpassen.

Ein weiteres Verbesserungspotenzial liegt in der Performance. Zwar konnten durch den Einsatz von Caches deutliche Verbesserungen erzielt werden, die Antwortzeit der ersten Anfrage überschreitet jedoch weiterhin das definierte Limit. Hier wäre ein Vorladen bestimmter Daten beim Serverstart denkbar; damit ließe sich die Antwortzeit auch beim ersten Zugriff zuverlässig reduzieren.

Darüber hinaus stellt sich die Frage nach der Qualität und Aktualität der zugrunde liegenden OSM-Daten. Da diese je nach Region unterschiedlich gepflegt sind, können Abweichungen in Genauigkeit und Vollständigkeit auftreten. Künftige Verbesserungen sollten daher vorsehen, regelmäßig auf Aktualität des Kartenmaterials zu prüfen und Mechanismen zur Qualitätssicherung der Daten zu entwickeln, um eine korrekte Planungsgrundlage sicherzustellen.

Schließlich konnte eine umfassende Evaluation im praktischen Einsatz im Rahmen dieser Arbeit nicht durchgeführt werden. Feldtests mit den relevanten Nutzergruppen bieten daher großes Verbesserungspotenzial für das Gesamtsystem, da sie nicht nur technische Aspekte, sondern auch Fragen der Benutzerfreundlichkeit beantworten würden. Gerade hier sind Rückmeldungen aus der alltäglichen Benutzung wichtig und bergen möglicherweise viele Vorschläge und Maßnahmen zur Verbesserung.

8.3 Erweiterungsmöglichkeiten

Wie bereits am Anfang der vorliegenden Arbeit erwähnt, handelt es sich bei dem entwickelten Gesamtsystem um einen Prototyp. Demnach liegt es in der Natur der Sache, dass sich gewisse Erweiterungsmöglichkeiten ergeben.

Auch wenn der manuelle Auswertungsaufwand durch das entwickelte System stark reduziert wurde, werden die fertigen Pläne noch immer auf Papier ausgegeben. Hier könnte angesetzt werden, indem die fertigen Pläne auch für die Kinder und ihre Eltern über das System abrufbar sind - entweder als simpler Download einer statischen Karte oder als

interaktive Karte in einer App. Dies würde den Prozess besonders für den Endbenutzer benutzerfreundlicher machen und entspricht auch den Erwartungen an eine moderne Anwendung.

Ein weiterer Punkt ist die Einbeziehung weiterer externer Daten. So würde die Planung erheblich bereichert werden, wenn Daten von Polizeibehörden über die Unfallhäufigkeit an bestimmten Stellen im Straßenverkehr den Planern zur Verfügung gestellt werden. Auch der ÖPNV kann durch Daten von Verkehrsbetrieben mit in die Planung eingebunden werden. Geländekarten können für die Planung im Winter hilfreich sein, wenn es um die Gefährdung des Schulweges durch Witterungseinflüsse geht; diese Daten gibt es bereits innerhalb des OSM-Projektes. Auch gibt es Daten von kommerziellen Anbietern, die beispielsweise Verkehrsdaten in Echtzeit bereitstellen können. Dies würde noch einmal ganz andere Möglichkeiten eröffnen: Statt statischer Karten könnte das System zu einer Art Routenplaner umgebaut werden, wo zu jeder Zeit der aktuell sicherste Schulweg generiert und angezeigt werden kann.

Während der ersten Anforderungserhebung und der anfänglichen Planung ist ein Vorschlag aufgekommen: die Generierung von Routenvorschlägen für den Planungsprozess. Dafür gibt es die Valhalla-API, die Routen mit beliebigen Zwischenzielen berechnen kann. Besonders ist bei dieser API, dass einem Aufruf eine Liste von „Vermeidungspolygonen“ mitgegeben werden kann, die von der zu berechnenden Route keinesfalls berührt werden dürfen. Das System könnte dann auf Basis der eingegangenen Daten erste Vorschläge liefern, wie Routen aussehen könnten; besonders mit Blick auf die im vorherigen Absatz genannten externen Daten könnte ein solches Feature sehr gewinnbringend sein [54].

Ein weiterer Schritt zur erleichterten Planung kann durch den Einsatz von künstlicher Intelligenz erfolgen. So können Zusammenhänge in den verfügbaren Materialien erkannt werden, die einem menschlichen Planer vielleicht nicht auf den ersten Blick auffallen oder bei komplexen Sachlagen zu viel Zeit für die Analyse beanspruchen. Auch eine KI-gestützte Auswertung der Textfelder in den Umfragen ist denkbar.

8.4 Gesamtfazit

Die vorliegende Arbeit hat aufgezeigt, dass die Konzeption und Umsetzung eines Backends für die digitale Schulwegplanung erfolgreich realisiert werden konnte. Durch die Kombination geeigneter Technologien wie dem Spring-Framework für die Implementierung einer REST-API, PostgreSQL mit der PostGIS-Erweiterung für eine effiziente Verarbeitung von Geodaten und einem durchdachten Datenmodell zur Speicherung und Auswertung von Umfragen und aufgezeichneten Routen wurde ein stabiles und wartbares System geschaffen, das die Anforderung erfüllt, die papierbasierte Bearbeitung zu ersetzen.

Die formulierten Fragestellungen aus der Zielsetzung konnten beantwortet werden. Dazu wurden die Anforderungen aller relevanten Benutzergruppen analysiert und ihre Bedürfnisse bei der Umsetzung des Systems berücksichtigt. Die Erhebung, Verarbeitung und Anzeige der erhobenen Daten erfolgt effizient und bietet dem Planer die Möglichkeit, erhobene Daten aggregiert und ohne weiteren Auswertungsaufwand in die Planung einzubeziehen. Dabei wird auf Sicherheits- und Datenschutzprinzipien Rücksicht genommen, um sensible Schülerdaten nicht zu gefährden.

Es wurde gezeigt, wie der papierbasierte Planungsprozess digitalisiert und damit deutlich effizienter gestaltet werden kann. Die Architektur ist dabei flexibel und ermöglicht es, künftige Erweiterungen einfach vorzunehmen. Das Backend benutzt ein einheitliches Datenformat und eine einheitliche Schnittstelle, wodurch die Erweiterbarkeit des Systems unterstützt wird.

Insgesamt zeigt der entwickelte Prototyp den Mehrwert digitaler Schulwegplanung und bildet eine gute Grundlage für eine Weiterentwicklung und den Einsatz im Alltag der Planer.

Literatur

[1] Walter Akolo. *Node.js vs. Spring Boot: Which is superior*. URL: https://hostadvice.com/blog/web-hosting/node-js/node-js-vs-spring-boot (besucht am 05. 09. 2025).

[2] Scrum Alliance. *Everything You Need to Know About Acceptance Criteria*. URL: https://resources.scrumalliance.org/Article/need-know-acceptance-criteria (besucht am 30. 08. 2025).

[3] Scott W. Ambler. *Mapping Objects to Relational Databases: O/R Mapping In Detail*. URL: https://agiledata.org/essays/mappingObjects.html (besucht am 17. 09. 2025).

[4] Scott W. Ambler. *Overcoming The Object-Relational Impedance Mismatch*. URL: https://agiledata.org/essays/impedanceMismatch.html (besucht am 17. 09. 2025).

[5] ArcGIS. *Mercator-Projektion*. URL: https://desktop.arcgis.com/de/arcmap/latest/map/projections/mercator.htm (besucht am 19. 09. 2025).

[6] International Ergonomics & Human Factors Association. *What Is Ergonomics (HFE)?* URL: https://iea.cc/about/what-is-ergonomics/ (besucht am 10. 09. 2025).

[7] Len Bass, Paul Clements und Rick Kazman. *Software Architecture in Practice*. 4. Aufl. Pearson, 2022.

[8] Christian Bauer und Gavin King. *Java Persistence with Hibernate*. 2. Aufl. 2007.

[9] *Benchmarking Api Response Times Across Different Web Frameworks*. URL: https://peerdh.com/blogs/programming-insights/benchmarking-api-response-times-across-different-web-frameworks (besucht am 05. 09. 2025).

[10] Alan Cooper, Robert Reimann, David Cronin und Christopher Noessel. *About Face: The Essentials of Interaction Design*. 4. Aufl. Wiley, 2014.

[11] George Coulouris, Jean Dollimore, Tim Kindberg und Gordon Blair. *Distributed Systems - Concepts and Design*. 5. Aufl. Addison-Wesley, 2012.

[12] Francisco Curbera, Matthew Duftler, Rania Khalaf, William Nagy, Nirmal Mukhi und Sanjiva Weerawarana. „Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI“. In: *IEEE Internet Computing* 6.2 (2002), S. 86–93.

[13] DeStatis. *27260 Kinder im Jahr 2024 bei Verkehrsunfällen verunglückt*. Statistisches Bundesamt. URL: https://www.destatis.de/DE/Presse/Pressemitteilungen/2025/08/PD25_N043_46241.html (besucht am 14. 08. 2025).

[14] Lisa M. Dusseault und James M. Snell. *PATCH Method for HTTP*. RFC 5789. März 2010. URL: https://www.rfc-editor.org/info/rfc5789.

[15] Claudia Eckert. *IT-Sicherheit: Konzepte - Verfahren - Protokolle*. 11. Aufl. De Gruyter Oldenbourg, 2023.

[16] *EhCache 3.11 Documentation*. URL: https://www.ehcache.org/documentation/3.11/ (besucht am 06. 09. 2025).

[17] Europäisches Parlament und Rat der Europäischen Union. *Verordnung (EU) 2016/679 des Europäischen Parlaments und des Rates vom 27. April 2016 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/46/EG (Datenschutz-Grundverordnung)*. `http://data.europa.eu/eli/reg/2016/679/oj`. Zugriff am 22.09.2025. 2016.

[18] Roy T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. 2000.

[19] Roy T. Fielding und Julian Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. RFC 7231. Juni 2014. URL: `https://www.rfc-editor.org/info/rfc7231`.

[20] Entwicklung und Evaluation Büro für Forschung. *https://www.schulwegcheck.de/*. URL: `https://www.schulwegcheck.de/` (besucht am 19. 09. 2025).

[21] Open Source Geospatial Foundation. *PostGIS*. URL: `https://postgis.net/` (besucht am 21. 09. 2025).

[22] Martin Fowler. *Patterns of Enterprise Application Architecture*. Pearson, 2002.

[23] fast2work GmbH. *Travorus*. URL: `https://www.travorus.de/de/` (besucht am 16. 09. 2025).

[24] Kim Goodwin. *Designing for the Digital Age*. Wiley, 2009.

[25] „ISO International Standard - Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems“. In: *ISO 9241-210:2019(E)* (2019).

[26] „ISO International Standard - Ergonomics principles in the design ofwork systems“. In: *ISO 6385:2016(E)* (2016).

[27] „ISO/IEC International Standard - Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model“. In: *ISO/IEC 25010:2023(E)* (2023).

[28] „ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering“. In: *ISO/IEC/IEEE 29148:2018(E)* (2018).

[29] Ganesh Iyer. *Node.js: Event-driven Concurrency for Web Applications*. Techn. Ber. Jan. 2013.

[30] Philip Kotler, Kevin Lane Keller, Alexander Chernev und Marc Oliver Opresnik. *Marketing-Management*. 16. Aufl. Pearson, 2023.

[31] Landesamt für Geoinformation und Landesentwicklung. *Schulwegplaner Baden-Württemberg*. URL: `https://schulwegplaner-bw.de/` (besucht am 21. 09. 2025).

[32] Norbert de Lange. *Geoinformatik in Theorie und Praxis*. 4. Aufl. Springer, 2020.

[33] Paul A. Longley, Michael F. Goodchild, David J. Maguire und David W. Rhind. *Geographic Information Systems and Science*. 4. Aufl. Wiley, 2015.

[34] *MQTT Version 5.0*. OASIS Standard mqtt-v5.0-os. OASIS, März 2019. URL: `https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.pdf`.

[35] *Map features*. URL: `https://wiki.openstreetmap.org/wiki/Map_features` (besucht am 20. 09. 2025).

[36] *MoSCoW Prioritisation*. Coley Consulting. URL: `https://www.coleyconsulting.co.uk/moscow.htm` (besucht am 16. 08. 2025).

[37] Zukunftsnetz Mobilität NRW. *Landesweiter Austausch zur digitalen Schulwegplanung*. URL: https://www.zukunftsnetz-mobilitaet.nrw.de/aktuelles/news/digitale-schulwegplanung (besucht am 21. 09. 2025).

[38] Michael Nieles, Kelley Dempsey und Victoria Yan Pillitteri. *An Introduction to Information Security*. NIST Special Publication 800-12 Revision 1. National Institute of Standards und Technology, Mai 2017.

[39] Don Norman. *The Design of Everyday Things*. 2. Aufl. Basic Books, 2013.

[40] *OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0*. OASIS Standard amqp-core-complete-v1.0. OASIS, Okt. 2012. URL: http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf.

[41] John S. Pruitt und Tamara Adlin. *The Persona Lifecycle - Keeping People in Mind Throughout Product Design*. Morgan Kaufmann, 2006.

[42] Mark Richards und Neal Ford. *Fundamentals of Software Architecture*. 2. Aufl. O'Reilly, 2025.

[43] James Robertson, Suzanne Robertson und Adrian Reed. *Mastering the Requirements Process*. 4. Aufl. Pearson, 2025.

[44] Yvonne Rogers, Helen Sharp und Jennifer Preece. *Interaction Design: Beyond Human-Computer Interaction*. 6. Aufl. Wiley, 2023.

[45] Heike Rusch. *SCHULWEGPLAN - Schulwegpläne für Berlin und Brandenburg*. URL: https://www.schulwegplan-berlin.de/ (besucht am 21. 09. 2025).

[46] Vikas Singh. *Node.js vs Spring Boot: Backend Decision Guide for Developers*. URL: https://www.brilworks.com/blog/node-js-vs-spring-boot (besucht am 05. 09. 2025).

[47] Ian Sommerville. *Software Engineering*. 10. Aufl. Pearson, 2016.

[48] *Spatial without Compromise - QGIS*. URL: https://qgis.org/ (besucht am 21. 09. 2025).

[49] Initiative für sichere Straßen GmbH. *Initiative für sichere Straßen*. URL: https://www.sichere-strassen.org/ (besucht am 18. 09. 2025).

[50] Initiative für sichere Straßen GmbH. *gefahrenstellen.de*. URL: https://www.gefahrenstellen.de/ (besucht am 18. 09. 2025).

[51] Initiative für sichere Straßen GmbH. *schulwege.de*. URL: https://www.schulwege.de/ (besucht am 18. 09. 2025).

[52] Andrew S Tanenbaum, Nick Feamster und David J. Wetherall. *Computernetzwerke*. 6. Aufl. Pearson, 2024.

[53] Andrew S. Tanenbaum und Maarten van Steen. *Distributed Systems - Principes and Paradigms*. 2. Aufl. Pearson, 2007.

[54] *Valhalla*. URL: https://valhalla.github.io/valhalla/ (besucht am 22. 09. 2025).

[55] Bundesanstalt für Straßen-und Verkehrswesen. *Schulwegpläne leicht gemacht - Der Leitfaden*. URL: https://www.bast.de/DE/Publikationen/Medien/Verhalten-und-Sicherheit/Schulwegplaene.html (besucht am 21. 09. 2025).

[56] Landesbetrieb Geoinformation und Vermessung. *Schulinfosystem*. URL: https://geoportal-hamburg.de/schulinfosystem/ (besucht am 18. 09. 2025).

[57] Cláudia M. Viana, Patrícia Abrantes und Jorge Rocha. „Introductory Chapter: Geographic Information Systems and Science". In: *Geographic Information Systems and Science*. Hrsg. von Jorge Rocha und Patrícia Abrantes. IntechOpen, 2019. Kap. 1, S. 3–19.

[58] Christoph Wegener, Thomas Milde und Wilhelm Dolle. *Informationssicherheits-Management*. 2016.

[59] FUSS e.V. *Schulwegplanung in den Ländern*. URL: `https://www.schulwegplaene.de/schulwegplanung.html` (besucht am 21. 09. 2025).

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Anhang

A Schulwegpläne

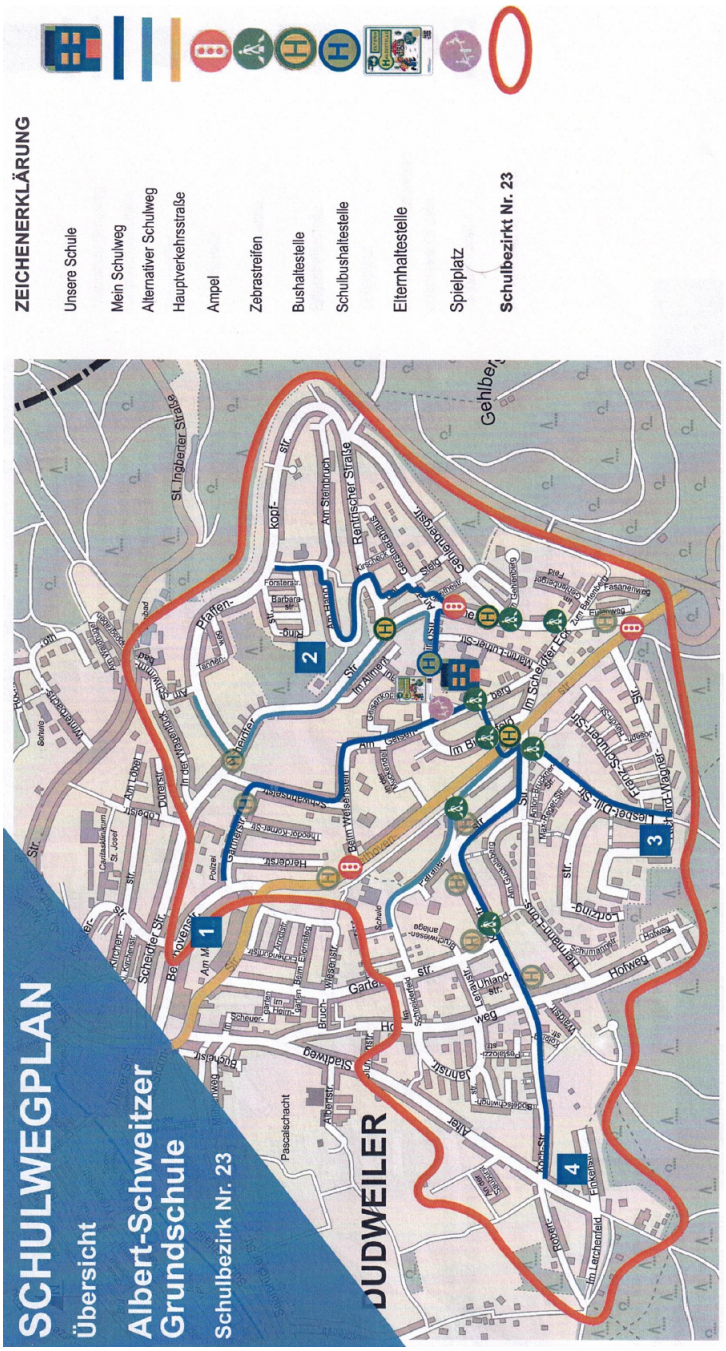

Abbildung A.1: Schulwegplan für die Albert-Schweitzer-Grundschule in Dudweiler.

Abbildung A.2: Schulwegplan für die Grundschule Am Ordensgut in Saarbrücken.

B API-Dokumentation

Allen Endpunkten wird noch ein `/api` vorangestellt, das hier aber aus Platzgründen weggelassen wurde.

Pfad	Methode	Bedeutung
`/auth/login`	`POST`	Loggt einen Benutzer ein.
`/auth/logout`	`POST`	Loggt einen Benutzer aus.
`/data/version`	`GET`	Gibt die aktuelle Version der Overpass-API zurück.
`/data/schools`	`GET`	Gibt eine Liste aller Schulen zurück.
`/data/poi`	`GET`	Gibt eine Liste aller POI zurück.
`/data/poi/<id>`	`GET`	Gibt den POI mit der `<id>` zurück.
`/data/poi/<id>/delete`	`DELETE`	Löscht den POI mit der `<id>`.
`/data/poi/create`	`POST`	Erstellt einen neuen POI.
`/layers/busstops/<districtId>`	`GET`	Gibt alle Bushaltestellen innerhalb des Schulbezirks `<districtId>` zurück.
`/layers/speeds/<districtId>`	`GET`	Gibt alle Höchstgeschwindigkeiten innerhalb des Schulbezirks `<districtId>` zurück.
`/layers/crossings/<districtId>`	`GET`	Gibt alle Überquerungsmöglichkeiten innerhalb des Schulbezirks `<districtId>` zurück.
`/layers/districts/<schoolId>`	`GET`	Gibt die Grenzen des Bezirks zurück, zu dem die Schule mit der `<schoolId>` gehört.
`/planning/questionnaire`	`POST`	Erstellt einen neuen Fragebogen.
`/planning/questionnaire/<id>`	`GET`	Gibt einen Fragebogen anhand seiner ID `<id>` zurück.
`/planning/questionnaires`	`GET`	Gibt alle Fragebögen zurück.
`/planning/<planId>/previewImage`	`GET`	Gibt ein Vorschaubild für einen Plan `<planId>` zurück.
`/planning/<planId>/delete`	`DELETE`	Löscht einen Plan anhand seiner ID `<planId>`.
`/planning/all`	`GET`	Gibt alle Pläne zurück.
`/planning/<planId>`	`GET`	Gibt einen Plan anhand seiner ID `<planId>` zurück.
`/planning/create`	`POST`	Erstellt einen neuen Plan.
`/planning/<planId>/map`	`GET`	Gibt die generierte Karte eines Plans `<planId>` als PNG zurück.

Pfad	Methode	Bedeutung
/planning/<planId>/addQuestionnaire	PATCH	Fügt einem Plan <planId> einen Fragebogen hinzu.
/planning/<planId>/drawing	GET	Gibt die Zeichnung eines Plans <planId> zurück.
/planning/<planId>/drawing	POST	Speichert eine Zeichnung für einen Plan <planId>.
/planning/<planId>/drawing/delete	DELETE	Löscht die Zeichnung eines Plans <planId>.
/survey/getQuestionnaire/<surveyString>	GET	Liefert den Fragebogen für die angegebene Umfrage mit dem Code <surveyString>.
/survey/submitQuestionnaire/<surveyString>	POST	Speichert Eingaben einer Umfrage mit dem Code <surveyString>.
/survey/submitRecording/<surveyString>	POST	Speichert aufgezeichnete Bewegungsdaten und Gefahrenstellen für eine Umfrage <surveyString>.
/survey/progress	GET	Fragt den Fortschritt der Umfrage für die aktuelle Session ab.
/survey/evaluate/<surveyString>	GET	Wertet einen Fragebogen aus und liefert die Ergebnisse.
/survey/heatmap/<surveyString>	GET	Gibt die Heatmap-Daten für die Umfrage <surveyString> zurück.
/survey/dangerspots/<surveyString>	GET	Gibt die Gefahrenstellen für die Umfrage <surveyString> zurück.
/survey/session	POST	Erstellt eine neue Umfrage-Session für den Benutzer.

Tabelle B.1: API-Endpunkte.

Kolophon

Dieses Dokument wurde mit der LaTeX-Vorlage für Abschlussarbeiten an der htw saar im Bereich Informatik/Mechatronik-Sensortechnik erstellt (Version 2.25, August 2024). Die Vorlage wurde von Yves Hary und André Miede entwickelt (mit freundlicher Unterstützung von Thomas Kretschmer, Helmut G. Folz und Martina Lehser). Daten: (F)10.95 – (B)426.79135pt – (H)688.5567pt