

Inhaltsverzeichnis

| | |
|--|-----------|
| 1 Einführung in die erwarteten Entwicklungsleistungen | 3 |
| 2 Organisation und Umsetzung der Entwicklung | 4 |
| 2.1 Entwicklungskonzept | 4 |
| 2.2 Entwicklungsprozess und Rollen..... | 5 |
| 2.3 Projektmanagementtools | 7 |
| 2.4 Sourcecode Verwaltung | 7 |
| 2.5 Entwicklerteam | 8 |
| 2.5.1 Zusammensetzung | 8 |
| 2.5.2 Kompetenzcluster | 8 |
| 3 Entwicklungsleistung | 11 |
| 3.1 Funktionale Anforderungen | 12 |
| 3.2 Kernfunktionen des MVP | 13 |
| 3.2.1 Routenplanung..... | 13 |
| 3.2.2 Einfache Kartenfunktion | 13 |
| 3.2.3 Echtzeit & Störungsinformationen | 14 |
| 3.2.4 Personalisierte Routingparameter / Personalisierbare Einstellungen | 14 |
| 3.2.5 Verfügbarkeit | 14 |
| 3.3 Version 1.0 funktionale Anforderungen | 14 |
| 3.3.1 Intermodale Routenplanung | 14 |
| 3.3.2 Kartenfunktion..... | 15 |
| 3.3.3 Echtzeit & Störungsinformationen | 16 |
| 3.4 Version 2.0 funktionale Anforderungen | 16 |
| 3.4.1 Erweiterte intermodale Routenplanung | 16 |
| 3.4.2 Preisinformationen | 16 |

| | |
|--|-----------|
| 3.4.3 Verfügbarkeit | 17 |
| 3.4.4 Personalisierte Routingparamete & Nutzerprofil..... | 17 |
| 3.5 Optionaler Projektbaustein – Native mobile Applikation (iOS / Android) | 17 |
| 3.6 Optionaler Projektbaustein – KI-gestützter Datenmodelle | 18 |
| 4 Technische Anforderungen | 19 |
| 4.1 Solldaten..... | 21 |
| 4.2 Tarifdaten | 21 |
| 4.3 Open-Source-Datenquellen | 22 |
| 4.4 Tech Stack..... | 22 |
| 4.4.1 Routing-Kern: OpenTripPlanner (OTP) | 22 |
| 4.4.2 Frontend: Digitransit UI (React)..... | 23 |
| 4.4.3 API-Schicht: Digitransit GraphQL API | 23 |
| 4.4.4 Open Source..... | 23 |
| 5 Weitere Anforderungen | 23 |
| 5.1 Entwicklungsumgebung | 23 |
| 5.2 Übergabe der Entwicklungsleistung für Los 2 | 24 |
| 6 Nicht funktionale Anforderungen | 25 |
| 6.1 Qualität26 | |
| 6.2 Sicherheit..... | 27 |
| 6.3 Erstellungsanforderungen | 29 |

1 Einführung in die erwarteten Entwicklungsleistungen

Im Rahmen der Entwicklung des SH-Navi wird eine digitale Mobilitätsplattform aufgebaut, die verschiedene Mobilitätsangebote integriert und für Nutzerinnen und Nutzer gebündelt verfügbar macht. Kern der Entwicklungsleistungen ist die Umsetzung einer Plattform mit Funktionen für intermodale Routenplanung, Echtzeitinformationen, Tarifauskünfte sowie eine standortbasierte Übersicht von Mobilitätsangeboten im Umfeld eines Ortes.

Die Plattform umfasst insbesondere folgende funktionale Entwicklungsbereiche:

- Routenplanung über verschiedene Verkehrsmittel hinweg, inklusive optimierter Reiseketten auf Basis individueller Nutzerpräferenzen sowie der Berücksichtigung von Echtzeitdaten, Auslastungsinformationen und lokalen Ereignissen wie Baustellen oder Sperrungen.
- Echtzeitinformationssysteme, die aktuelle Betriebs- und Auslastungsinformationen des öffentlichen Verkehrs sowie weiterer Mobilitätsangebote integrieren und darstellen.
- Tarifinformationsfunktionen, die Preisinformationen für ÖPNV-Verbindungen sowie perspektivisch für intermodale Reiseketten bereitstellen.
- Umkreis- und Standortübersichten, die alle verfügbaren Mobilitätsangebote in der Nähe eines Standortes darstellen und auch für die Darstellung auf digitalen Informationsstellen geeignet sind.
- Schnittstellen und Integrationsmöglichkeiten für Kommunen und weitere Akteure, um lokale Informationen (z. B. Betriebszustände, Parkraumverfügbarkeiten oder Points of Interest) einzubinden sowie Routing- und Erreichbarkeitsinformationen in eigene digitale Angebote zu integrieren.
- Einbettungsmöglichkeit auf anderen Webseiten per iframe (Widget)
- **Optional:** Umsetzung einer nativen App
- **Optional:** Integration von bereits entwickelten KI-gestützten Auslastungs- und Prognosemodellen (bitte als gesonderten Posten in der ANLAGE_3_Preisblatt_Funktionale_Entwicklungsanforderungen aufnehmen)

Die Entwicklung erfolgt in mehreren Versionen und Ausbaustufen. Funktionsumfänge und Prioritäten sind dem jeweils definierten Versionsumfang der einzelnen Entwicklungsstufen zu entnehmen. Detaillierte Entwicklungsleistungen werden in den einzelnen Sprints konkretisiert und sind der agilen Arbeitsweise entsprechend flexibel anzupassen.

2 Organisation und Umsetzung der Entwicklung

Die Bereitstellung der ausgeschriebenen Entwicklungsleistung erfolgt als gesamthafte Entwicklungspaket und wird gemäß den folgenden methodischen Grundsätzen und Anforderungen durchgeführt.

2.1 Entwicklungskonzept

Die vertragsgegenständlichen Entwicklungsleistungen sind auf Grundlage des vom Auftragnehmer im Vergabeverfahren eingereichten und bezuschlagten Entwicklungskonzepts (Anlage 9) zu erbringen, soweit dieses den Anforderungen der Vertragsunterlagen nicht widerspricht. Der Umfang des Entwicklungskonzepts sollte 3 - 5 A4 Seiten umfassen.

Das Entwicklungskonzept beschreibt die methodische, organisatorische und technische Ausgestaltung der Leistungserbringung und muss gewährleisten, dass die Entwicklung unter Anwendung eines strukturierten, belastbaren und den Anforderungen dieses Vertrages entsprechenden agilen bzw. inkrementellen Vorgehensmodells erfolgt.

Das Entwicklungskonzept hat insbesondere nachvollziehbare Regelungen zu enthalten hinsichtlich:

- Entwicklungsprozess und Sprintlogik,
- Planungs-, Refinement-, Review- und Abstimmungsprozessen,
- Rollen, Verantwortlichkeiten und Zusammenarbeit im vorgesehenen Personal,
- Backlog-Management und Anforderungsstrukturierung,
- Versionskontrolle sowie Branching-, Pull-Request- und Merge-Prozessen,
- Code-Review- und Qualitätssicherungsverfahren,
- Teststrategie, Testabdeckung und Testautomatisierung,
- Issue-Tracking und Fehlerdokumentation,
- Dokumentationsstandards sowie
- Build-, Release-, Deployment- und Übergabeprozessen.

Der Auftragnehmer ist verpflichtet, die Entwicklungsleistungen entsprechend den bewertungsrelevanten und vertragskonformen Grundsätzen des eingereichten Entwicklungskonzepts durchzuführen.

Fortschreibungen oder Anpassungen des Entwicklungskonzepts während der Vertragslaufzeit sind zulässig, soweit die vertragsgemäße Leistungserbringung, die Einhaltung der Mindestanforderungen sowie das qualitative Niveau des bezuschlagten Konzepts insgesamt gewahrt bleiben.

2.2 Entwicklungsprozess und Rollen

Sofern im **Kick-Off-Termin** keine abweichende Vereinbarung getroffen wird, erfolgen die Entwicklungsarbeiten in zweiwöchigen Sprints. Dabei werden folgende Events durchgeführt:

| Event | Inhalte und Verantwortlichkeiten |
|--|---|
| Sprint Planning | <ul style="list-style-type: none">• Inhaltliche Projektleitung bringt mit Stakeholdern abgestimmte Anforderungen und Prioritäten ein• Technische Projektleitung übersetzt Anforderungen in User Stories & Backlog-Einträge• Entwicklerteam schätzt Aufwände, gibt technische Einschätzungen• Community-Feedback wird bei Bedarf integriert |
| Refinement der Entwicklungsaufgaben / User Stories | <ul style="list-style-type: none">• In Ergänzung zum Sprint Planning:• Technisch orientiert: Durchsprache und Klärung der erstellten User Stories mit dem Entwicklerteam• Fachlich orientiert: Ausformulierung der User Stories inklusive Akzeptanzkriterien und Definition of Done |
| Sprint Review | <ul style="list-style-type: none">• Entwicklerteam präsentiert Inkremente• Technische Projektleitung steuert Feedback-Runde, dokumentiert Ergebnisse• Stakeholder, Projektpartner & Community geben Rückmeldungen. Ergebnisse fließen zurück ins Backlog |
| Sprint | <ul style="list-style-type: none">• Entwicklerteam implementiert User Stories, sorgt für Qualitätssicherung |

Optional kann eine **Retrospektive** zur Optimierung der Zusammenarbeit im Team durchgeführt werden. Ideale Zeitabstände sind während der Projektlaufzeit und während der Sprintplanung mit dem Auftraggeber abzustimmen. Abbildung 1 veranschaulicht den Prozess.

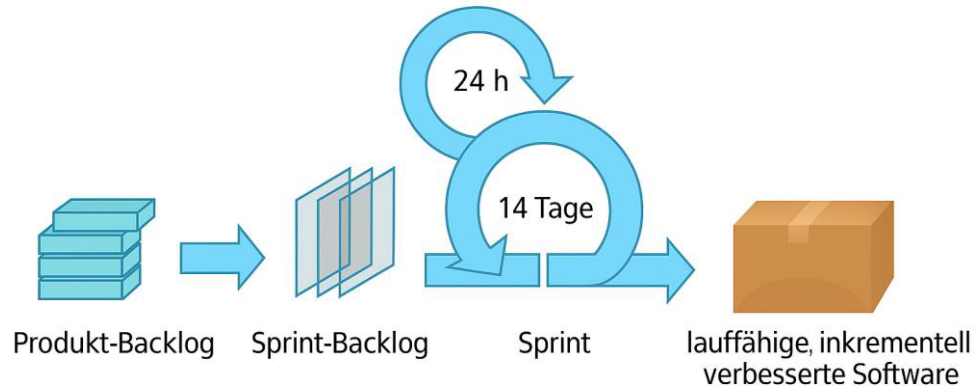


Abbildung 1: Agiles Entwicklungsvorgehen

Folgende Rollen sind entsprechend wahrzunehmen und zu besetzen:

| Rolle | Verantwortlich | Inhalte |
|---------------------------------------|----------------|---|
| Product Owner | Auftraggeber | <ul style="list-style-type: none">• Erstellung des Product Backlogs (gemeinsam mit Entwicklerteam)• Pflege und Priorisierung des Product Backlogs• Definition von Anforderungen• Stakeholder-Abstimmung• Vertretung der Stakeholder-Interessen• fachliche Abnahme (auch in Bezug auf Testing) |
| Technische Projektleitung (Tech Lead) | Auftraggeber | <ul style="list-style-type: none">• Erstellung des Product Backlogs• Pflege und Priorisierung des Product Backlogs aus technischer Sicht• Definition von technischen Anforderungen• Abnahme (Testing)• Enge Abstimmung mit der Entwicklung, technische Bewertung und Priorisierung• Sicherstellung, dass technische Entscheidungen zum Gesamtprodukt passen• Planung von Releases.• Moderation der Scrum-Events• Beseitigung von Hürden im Tech-Bereich• Onboarding des Entwickler-Teams• Förderung kontinuierlicher Verbesserung |

| | | |
|----------------|--------------------------------|--|
| Entwicklerteam | Auftragnehmer | <ul style="list-style-type: none">• Gemeinsame Erstellung des Product Backlogs• Umsetzung der Backlog-Einträge• technische Konzeption und Lösungsansätze• Aufwandsschätzung• Durchführung softwareseitiger Tests• Qualitätssicherung• Selbstorganisation im Team |
| Tester | Auftraggeber und Auftragnehmer | <ul style="list-style-type: none">• Planung und Durchführung von Tests• Testautomatisierung• Dokumentation von Fehlern• Unterstützung bei softwareseitigen Tests sowie Vorbereitung und Begleitung von Abnahmetests |

2.3 Projektmanagementtools

Für die Projektkoordination wird ein für agile Entwicklungsprojekte gängiges Board / Backlog (z.B. Kanban-Board) genutzt. Dieses muss im OpenCode-Projekt aufgesetzt sein. Das Board besitzt mindestens folgende Spalten:

- **Backlog** – nicht priorisierte Sammlung von Anforderungen/User Stories
- **To Do / Review** – umsetzungsbereite Aufgaben
- **In Progress / aktueller Sprint** – aktuell bearbeitete Aufgaben
- **in QA** – Ergebnisse zur fachlichen und technischen Prüfung

2.4 Sourcecode Verwaltung

Für die Verwaltung des Sourcecodes wird ein zentrales, mit OpenCode verknüpftes Repository genutzt. Sofern im Kick-Off-Workshop nicht anders besprochen, übernimmt der Auftragnehmer die Erstellung, Konfiguration und Pflege des Repositories.

Zur Absicherung und Vermeidung von Datenverlusten stellt der Auftragnehmer zusätzlich ein separates Mirror-Repository bereit. Dieses dient der redundanten Sicherung des Quellcodes.

Der gesamte Projektcode muss strukturiert und revisionssicher in diesem Repository verwaltet werden. Änderungen am Quellcode erfolgen nachvollziehbar über Branching- und Merge-Strategien (z. B. Feature-Branches, Merge-Requests) und unterliegen klar definierten Review- und Freigabeprozessen. Jede Änderung ist versioniert, dokumentiert und jederzeit reproduzierbar.

Es müssen die beigefügten Open-Source-Gebote angewandt und beachtet werden.

2.5 Entwicklerteam

2.5.1 Zusammensetzung

Das vom Auftragnehmer eingesetzte **Entwicklerteam** besteht aus mindestens zwei Personen. Wobei einer Person der Lead in dem jeweiligen Kompetenzcluster zugewiesen ist. In diesem Fall muss die Person über mindestens 5 Jahre nachweislich einschlägige praktische Berufserfahrung im jeweiligen Kompetenzcluster verfügen. Einschlägig ist eine Berufserfahrung insbesondere dann, wenn sie in fachlich, technisch oder methodisch vergleichbaren Projekten erworben wurde.

Das Entwicklerteam muss in seiner Gesamtheit über hinreichende deutsche Sprachkenntnisse zur sicheren Durchführung projektbezogener Abstimmungen, Dokumentationen und Kommunikation mit dem Auftraggeber sowie über hinreichende englische Sprachkenntnisse zur Nutzung technischer Dokumentationen, Open-Source-Ressourcen und zur fachlichen Kommunikation in einschlägigen internationalen Entwicklungs- oder Community-Kontexten verfügen.

2.5.2 Kompetenzcluster

Die nachfolgenden Kompetenzcluster beschreiben die fachlichen, technischen, methodischen und projektspezifischen Mindestkompetenzen, die durch das vom Auftragnehmer eingesetzte **Entwicklerteam** in seiner Gesamtheit jederzeit abgedeckt werden.

Die Kompetenzcluster dienen der Sicherstellung, dass das **Entwicklerteam** insgesamt über die zur vertragsgemäßen Umsetzung, Weiterentwicklung, Dokumentation und Übergabefähigkeit des Projekts erforderliche Mindestsubstanz verfügt:

2.5.2.1 Kompetenzcluster 1 – Open-Source-, Lizenz- und Community-Kompetenz

Das Entwicklerteam muss über praktische Erfahrung im rechtssicheren Einsatz, in der Analyse, Integration, Anpassung, Dokumentation und Weiterentwicklung von Open-Source-Software verfügen. Hierzu zählen insbesondere Kenntnisse über unterschiedliche Open-Source-Lizenzmodelle, Lizenzkompatibilität, Copyleft-Anforderungen, Dokumentationspflichten, Contribution-Guidelines sowie der professionelle Umgang mit bestehenden Open-Source-Codebasen.

Das Entwicklerteam muss in der Lage sein,

- Open-Source-Komponenten projektkonform auszuwählen,
- Lizenzpflichten technisch und organisatorisch einzuhalten,

- Software-Komponenten nachvollziehbar zu dokumentieren (einschließlich SBOM-nahem Verständnis),
- bestehende Open-Source-Projekte fachlich zu analysieren und weiterzuentwickeln sowie
- mit Open-Source-Communities über etablierte Prozesse (z. B. Issue-Tracker, Pull Requests, Contribution-Prozesse) professionell zu interagieren.

2.5.2.2 Kompetenzcluster 2 – Mobilitäts-, Verkehrs- und Geodatenkompetenz

Das Entwicklerteam muss über praktische Erfahrung in der Verarbeitung, Integration, Modellierung und Nutzung komplexer Mobilitäts-, Verkehrs- und Geodaten verfügen. Dies umfasst insbesondere statische und dynamische Datenformate im Mobilitätsumfeld, beispielsweise GTFS, GTFS-RT, NeTEx, SIRI, GBFS, OSM oder vergleichbare Datenquellen.

Erforderlich ist insbesondere:

- Import, Validierung und Transformation von Mobilitätsdaten,
- softwareseitige Modellierung und Nutzung solcher Daten,
- Verarbeitung umfangreicher Geodaten- und Fahrplandatenbestände,
- Verständnis von Linien-, Netz-, Fahrplan- und Echtzeitlogiken sowie
- Berücksichtigung intermodaler Mobilitätsformen einschließlich Sharing-, Fuß-, Rad- und Anschlusslogiken.

2.5.2.3 Kompetenzcluster 3 – Backend-, Routing- und Integrationskompetenz

Das Entwicklerteam muss über nachweisliche Erfahrung in der technischen Umsetzung, Konfiguration, Anpassung, Erweiterung oder Integration routingbezogener Backend-Systeme und softwareseitiger Datenverarbeitung verfügen.

Hierzu zählen insbesondere:

- Backend-nahe Softwareentwicklung,
- Routinglogiken und Routingparameter,
- Integrationsfähigkeit routingrelevanter Open-Source-Komponenten (z. B. OpenTripPlanner oder vergleichbare Systeme),
- Datenmodellverständnis im Bereich Routing / OSM / Mobilitätsdaten,
- API-Anbindungen, Datenpipelines und Schnittstellen sowie
- technische Fehleranalyse, Debugging und Build-/Release-Prozesse.

2.5.2.4 Kompetenzcluster 4 – Frontend-, UI-/UX- und Barrierefreiheitskompetenz

Das Entwicklerteam muss praktische Erfahrung in der Entwicklung moderner, webbasierter, benutzerorientierter und möglichst kartenbezogener Benutzeroberflächen besitzen, die komplexe Mobilitäts- und Routinginformationen nachvollziehbar darstellen können.

Dies umfasst insbesondere:

- Frontend-Entwicklung für Routing-, Mobilitäts- oder vergleichbare datenintensive Anwendungen,
- Darstellung und Visualisierung komplexer Routingergebnisse,
- Benutzerführung bei intermodalen Reiseoptionen,
- Darstellung von Umstiegen, Wegeketten und Reisealternativen,
- Berücksichtigung digitaler Barrierefreiheit sowie
- nachvollziehbare, wartbare Frontend-Architektur.

2.5.2.5 Kompetenzcluster 5 – Softwarearchitektur-, API- und Systemdesignkompetenz

Das Entwicklerteam muss über belastbare Erfahrung in der Konzeption, Umsetzung, Weiterentwicklung und Dokumentation modularer, wartbarer und integrationsfähiger Softwarearchitekturen verfügen.

Erforderlich sind insbesondere:

- modulare Softwarearchitektur,
- Integration und Erweiterung bestehender Softwarekomponenten,
- API-Design (z. B. REST, GraphQL oder vergleichbar),
- Schnittstellenlogik und Sicherheitsmechanismen,
- Umgang mit proprietären und offenen Datenquellen sowie
- technische Skalierbarkeit, Wartbarkeit und langfristige Entwicklungsfähigkeit.

2.5.2.6 Kompetenzcluster 6 – Qualitätssicherung, Test- und Dokumentationskompetenz

Das Entwicklerteam muss in der Lage sein, Softwareartefakte nachvollziehbar, reproduzierbar, testbar und wartbar zu entwickeln.

Dies umfasst insbesondere:

- Unit-, Integrations-, Regressions- und End-to-End-Testverständnis,
- automatisierte Qualitätssicherung,
- Definition und Umsetzung technischer Qualitätsstandards,
- strukturierte Fehleranalyse und Fehlerbehebung,

- nachvollziehbare Software- und Architektur-Dokumentation sowie
- Einhaltung projektbezogener Definition-of-Done-Anforderungen.

2.5.2.7 Kompetenzcluster 7 – Agile Delivery-, Backlog- und Umsetzungskompetenz

Das Entwicklerteam muss über praktische Erfahrung in der strukturierten Mitarbeit in agilen oder vergleichbar inkrementellen Entwicklungsmodellen verfügen.

Dies umfasst insbesondere:

- Arbeit mit User Stories,
- Mitwirkung an Product Backlogs,
- Konkretisierung fachlicher Anforderungen,
- Definition und Mitwirkung an Akzeptanzkriterien,
- Aufwandsschätzung in iterativen Entwicklungsmodellen sowie
- aktive, eigenverantwortliche Mitwirkung an Planungs-, Abstimmungs- und Reviewprozessen.

2.5.2.8 Kompetenzcluster 8 – Projekt-, Stakeholder- und Kommunikationskompetenz

Das Entwicklerteam muss in der Lage sein, technische, organisatorische und fachliche Anforderungen in einem strukturierten Projektumfeld nachvollziehbar umzusetzen und projektbezogen transparent zu kommunizieren.

Dies umfasst insbesondere:

- strukturierte Zusammenarbeit mit Auftraggebern und Projektverantwortlichen,
- nachvollziehbare Darstellung technischer Sachverhalte,
- Kommunikation technischer Risiken und Umsetzungsbedarfe,
- Zusammenarbeit in mehrstufigen Stakeholder-Strukturen,
- schriftliche technische Kommunikationsfähigkeit in deutscher Sprache sowie
- grundlegende Fähigkeit zur Nutzung englischsprachiger technischer Dokumentationen und Community-Strukturen.

3 Entwicklungsleistung

Die Entwicklung erfolgt iterativ und inkrementell; MVP, V1 und V2 bilden übergeordnete Release- bzw. Meilensteinstufen. Ziel dieser Struktur ist es, Funktionsumfang und technische Komplexität schrittweise aufzubauen, frühzeitig lauffähige Ergebnisse bereitzustellen und die Weiterentwicklung auf Basis von Feedback und technischen Erkenntnissen zu steuern.

Der **MVP** bildet eine erste lauffähige Ausprägung des SH-Navi mit einem reduzierten Funktionsumfang. Er dient primär der Validierung zentraler Systemkomponenten, der Überprüfung der Datenintegration (z. B. Fahrplan- und Echtzeitdaten) sowie der ersten Bewertung von Nutzbarkeit und Performance.

Version 1 erweitert den MVP um die vollständige intermodale Verbindungsauskunft, zusätzliche Kartenlayer, wie z.B. ein Layer mit Ladepunkten oder Parkraumsensorik, sowie weitere Mobilitätsangebote. Ziel dieser Version ist die Bereitstellung einer funktional vollständigen Auskunftsplattform für intermodale Reiseketten.

Version 2 ergänzt darauf aufbauend Funktionen zur Personalisierung, erweiterten Entscheidungsunterstützung und Integration zusätzlicher Mobilitätsangebote. Dazu zählen u. a. alternative Verbindungen bei Störungen, Preisinformationen, Nutzerprofile sowie erweiterte Verfügbarkeits- und Statusinformationen.

Die in diesem Dokument beschriebenen Inhalte definieren den funktionalen Zielrahmen der jeweiligen Versionen. Auf dieser Grundlage wird der Product Backlog gemeinsam mit dem Entwicklungsteam erarbeitet und priorisiert. Die konkrete Umsetzung erfolgt iterativ im Rahmen der gewählten Entwicklungsmethodik (Scrum), wobei einzelne Anforderungen weiter spezifiziert, technisch bewertet und in umsetzbare Arbeitspakete überführt werden.

3.1 Funktionale Anforderungen

Die fachlichen Anforderungen an das SH-Navi leiten sich größtenteils aus dem zuvor beschriebenen Zielbild ab. Dieses definiert den angestrebten fachlichen Rahmen, die zentralen Handlungsfelder sowie die funktionalen und prozessualen Zielsetzungen der Anwendung.

Die fachlichen Anforderungen umfassen dabei sämtliche Funktionen, fachlichen Regeln, Datenverarbeitungslogiken sowie Interaktionsmöglichkeiten. Sie berücksichtigen sowohl die Perspektive der Nutzer als auch die Anforderungen aus dem Stakeholderkreis und administrativer bzw. politischer Rollen.

Die Konkretisierung der fachlichen Anforderungen erfolgt im Rahmen des beschriebenen agilen Entwicklungsansatzes. Anforderungen werden fortlaufend in Form von User Stories beschrieben und priorisiert. Diese werden in enger Abstimmung zwischen Product Owner, Fachvertretern sowie dem Entwicklungsteam im Rahmen von Refinement-Terminen inhaltlich geschärft, bewertet und hinsichtlich Umsetzbarkeit sowie Aufwand eingeordnet. Auf dieser Basis erfolgten die Sprint-Planung und die iterative Umsetzung.

Die nachfolgend beschriebenen Funktionen konkretisieren das Zielbild und leiten sich unmittelbar aus den dort definierten Leitlinien ab. Zur strukturierten Umsetzung der Anforderungen sind im Folgenden Versionen mit jeweils definiertem Funktionsumfang dargestellt, die den konkreten Entwicklungsleistungen der jeweiligen Ausbaustufen entsprechen.

3.2 Kernfunktionen des MVP

Das MVP dient dazu, erste Einblicke in die geplante Anwendung zu geben und zentrale Funktionen der intermodalen Reiseplanung zu testen. Es soll mindestens die zentralen Funktionen, die im derzeitigen Prototyp realisiert wurden, umfassen. Diese sollen im Rahmen des MVP vollständig nachgebildet und zur produktiven Nutzung vorbereitet werden.

3.2.1 Routenplanung

Nutzer:innen können eine einfache Verbindung von Start zu Ziel berechnen lassen. Folgende Verkehrsoptionen sollen unterstützt werden:

- ÖPNV / SPNV (Soll-Fahrplandaten)
- On-Demand-Verkehr (NAHSHUTTLE (Smile24, Lüttbus, Remo etc.), Alfa)
- Fußweg
- Fahrrad
- Bereits zur Verfügung stehende Sharing-Angebote (Bikesharing SprottenFlotte, E-Scooter-Sharing Bolt und Carsharing Flow)

Funktionen:

- Anzeige der empfohlenen Route
- Darstellung der intermodalen Reisekette
- Anzeige mehrerer alternativer Reiseketten
- Anzeige der Umstiege
- Anzeige von Via-Verbindungen

3.2.2 Einfache Kartenfunktion

Eine interaktive Karte zeigt relevante Mobilitätsinfrastruktur. Folgende Objekte sollen dargestellt werden:

- Haltestellen
- POIs (Basis-Kategorien)
- Fahrradabstellanlagen
- Ladesäulen (wo möglich inkl. Belegung über Open-Source-Datenquellen)

3.2.3 Echtzeit & Störungsinformationen

- Echtzeit-Abfahrten
- Verspätungen
- Störungsmeldungen
- Hinweismeldungen

3.2.4 Personalisierte Routingparameter / Personalisierbare Einstellungen

Nutzer:innen können ihre Routingparameter individuell konfigurieren. Darunter:

- Gehgeschwindigkeit
- Verkehrsmittel ausschließen (z. B. kein PKW, kein Fahrrad)
- Barrierearme Route (Basisprofil in Abhängigkeit der vorhandenen Datenbasis)
- Anzahl der Umstiege
- Alternative Haltestelle für Start und Ziel anzeigen

Diese Einstellungen werden bei der Routenberechnung automatisch berücksichtigt.

3.2.5 Verfügbarkeit

Nutzer:innen haben die Möglichkeit, sowohl die Auslastung an den Sharing-Stationen und Parkplätzen als auch die Verfügbarkeit einzelner Fahrzeuge beim Klick auf das Icon auf der Karte oder in Verbindungsdetails angezeigt zu bekommen. Anzeige im Umkreis:

- Auslastung Parkplätze
- Auslastung Bikeshaaring-Stationen

3.3 Version 1.0 funktionale Anforderungen

Ziel der ersten Version (V1.0) ist die Bereitstellung einer performanten, intermodalen Verbindungsauskunft, die verschiedene Verkehrsmittel kombiniert und Nutzenden eine einfache Planung ihrer Reise ermöglicht.

Die Anwendung stellt Routen, Mobilitätsangebote im Umfeld sowie Echtzeitinformationen übersichtlich dar und ermöglicht eine grundlegende intermodale Reiseplanung.

3.3.1 Intermodale Routenplanung

Die Anwendung ermöglicht die Berechnung von intermodalen Reiseketten unter Einbeziehung verschiedener Verkehrsmittel.

Der “Absprung” aus der Anwendung per Deep Link zur Buchung einzelner Elemente der Reisekette inkl. Parameterübergabe soll möglich sein (bspw. Bike Sharing, On-Demand-Verkehre).

Unterstützte Verkehrsoptionen in Version 1:

- ÖPNV / SPNV
 - Soll-Fahrplandaten
 - Echtzeitdaten (Verspätungen, Abfahrten)
- Mikromobilität (weitere E-Scooter- und Bikesharing)
- Fußweg
- Eigenes Fahrrad als Routingoption mit Möglichkeit der Mitnahme im Zug
 - inkl. Fahrrad-Routing
- Motorisierter Individualverkehr (MIV / PKW) als Routingoption
- Park & Ride als kombinierte Routingoption
- On-Demand-Verkehre
- Anzeige von Baustellen, sofern Daten vorhanden sind

3.3.2 Kartenfunktion

Die Anwendung stellt eine interaktive Karte mit aktivierbaren Layern bereit. Darstellbare Layer in Version 1:

- Haltestellen
- Bikesharing-Stationen (SprossenFlotte)
- Carsharing-Stationen (StattAuto, Car.los!)
- E-Scooter-Fahrzeuge
- Mobilitätsstationen
- E-Ladestationen (Einbindung über Open-Source-Datenquellen)
- POIs (Kategorien werden noch definiert)
- Fahrradabstellanlagen
- Parkhäuser und Parkflächen
- Taxistände
- Öffentliche Toiletten
- Service-Stationen

Die Karte dient sowohl zur Visualisierung der Umgebung als auch zur Auswahl von Start- und Zielpunkten.

3.3.3 Echtzeit & Störungsinformationen

Die Anwendung integriert verfügbare Echtzeitdaten und zeigt relevante Störungen an.

- Anzeige von Echtzeit-Abfahrten
- Anzeige von Verspätungen
- Anzeige von Ausfällen (Fahrtausfällen, Haltausfällen, Zusatzhalten, Zusatzfahrten)

Zusätzlich erfolgt eine:

- Integration von mobil.live zur Informationsverknüpfung und Anzeige relevanter Hinweise auf digitalen Stelen

3.4 Version 2.0 funktionale Anforderungen

Ziel der zweiten Version (V2.0) ist die Verbesserung der Routenqualität, Personalisierung und Entscheidungsunterstützung für Nutzer:innen.

Die Anwendung erweitert die bestehende intermodale Auskunft um alternative Verbindungen, personalisierte Routingparameter, erweiterte Mobilitätsangebote sowie Preisinformationen.

Darüber hinaus werden Verfügbarkeits- und Auslastungsinformationen integriert und eine Routenüberwachung mit Live-Navigation eingeführt.

3.4.1 Erweiterte intermodale Routenplanung

- Berücksichtigung unterschiedlicher Optimierungskriterien (z. B. schnellste Route, wenig Umstiege)
- stärkere Berücksichtigung von Echtzeitdaten (Verspätungen)
- dynamische Anpassung der Verbindungsauskunft
- Integration von Störungsmeldungen, sofern verfügbar
- Vorschläge für alternative Verbindungen bei Störungen
- Routenüberwachung mit Live-Navigation
- Einfache Rerouting-Logik bei Störungen

3.4.2 Preisinformationen

Integration von Preisdaten für den öffentlichen Verkehr.

- Vollständige Integration von GTFS-Fares
- Preisauskunft für ÖPNV-Verbindungen

- Intermodale Preisauskunft (z. B. Kombination ÖV + Sharing-Angebote, sofern Daten verfügbar)

3.4.3 Verfügbarkeit

Nutzer:innen haben die Möglichkeit, sowohl die Auslastung an den Sharing-Stationen und Parkplätzen als auch die Verfügbarkeit einzelner Fahrzeuge beim Klick auf das Icon auf der Karte oder in Verbindungsdetails angezeigt zu bekommen. Anzeige im Umkreis:

- Verfügbarkeit Carsharing
- Status On Demand (verfügbar/nicht verfügbar - auch ohne Buchung)

3.4.4 Personalisierte Routingparamete & Nutzerprofil

- Personalisierte Routen als Profil speicherbar
- Vordefinierte Profile zum optimierten Routing hinterlegt (allein unterwegs, mit Kind, auf dem Weg zur Arbeit etc.)

3.5 Optionaler Projektbaustein – Native mobile Applikation (iOS / Android)

Soweit der optionale Projektbaustein „Native App“ beauftragt wird, hat der Auftragnehmer ergänzend zur Webanwendung eine plattformübergreifende mobile Applikation für iOS und Android bereitzustellen, die sich konzeptionell, funktional und technisch möglichst eng an den Grundsätzen einer offenen, Open-Source-basierten Mobilitätsplattform orientiert.

Ziel ist die Entwicklung einer weitestgehend auf Open-Source-Komponenten basierenden nativen oder plattformübergreifenden mobilen Anwendung, die sich – soweit projektbezogen sachgerecht und technisch umsetzbar – an etablierten Open-Source-Referenzprojekten (z. B. dem StadtNavi-/Trufi-Ansatz) orientiert, ohne dass hierdurch eine zwingende Bindung an ein konkretes Drittprodukt oder Repository entsteht.

Die mobile Applikation soll funktional grundsätzlich auf eine möglichst weitgehende Spiegelung der jeweils umgesetzten Webanwendungsfunktionalitäten ausgerichtet sein, wobei die Webanwendung das vorrangige Hauptprodukt bleibt und die Priorisierung der App-Umsetzung nachgelagert erfolgt.

Der konkrete Funktionsumfang der nativen App richtet sich nach den jeweils maßgeblichen Entwicklungsstufen (insbesondere MVP, Version 1.0 und Version 2.0) sowie den dort priorisierten Leistungsbestandteilen. Die Umsetzung kann parallel erfolgen, hat sich jedoch an der funktionalen Reife und Priorisierung der Webanwendung zu orientieren.

Zu den grundsätzlich berücksichtigungsfähigen Funktionsbereichen zählen insbesondere:

- intermodales und multimodales Routing einschließlich ÖPNV und regionaler Echtzeitdaten,
- Verknüpfung und Integration verfügbarer Mobilitätsangebote,
- On-Demand-Routing (z. B. AST, ALFA, NAH.SHUTTLE), soweit geeignete Schnittstellen, Datenqualitäten und Nutzungsrechte vorliegen,
- Tarifinformationen,
- Bike-, Car- und weitere Sharing-Angebote,
- Park & Ride / Bike & Ride,
- E-Ladesäulen-, Parkplatz- und Verfügbarkeitsinformationen,
- Einbindung privater Mitfahr- oder Fahrgemeinschaftsangebote,
- Deeplinks oder vergleichbare Weiterleitungen zu externen Buchungs- oder Anbieterportalen,
- Integration relevanter Kartenlayer und Zusatzinformationen.

Die mobile Applikation ist einschließlich der technischen, organisatorischen und rechtlichen Voraussetzungen für die Veröffentlichung in den jeweils relevanten App-Stores (insbesondere Apple App Store / Google Play Store) bereitzustellen, soweit dies beauftragt ist.

Der Auftragnehmer hat bei Konzeption und Umsetzung in besonderem Maße Anforderungen an digitale Barrierefreiheit zu berücksichtigen. Hierzu zählen mindestens barrierearme Bedienkonzepte sowie – soweit technisch sachgerecht – Funktionen der Spracheingabe und Sprachausgabe für zentrale Routingfunktionen.

Der Auftragnehmer darf zur Umsetzung der nativen App geeignete Unterauftragnehmer einsetzen. Die Gesamtverantwortung für vertragsgemäße Planung, Koordination, Qualitätssicherung, Open-Source-Konformität, Übergabefähigkeit und vertragsgerechte Leistungserbringung verbleibt ausschließlich beim Auftragnehmer.

3.6 Optionaler Projektbaustein – KI-gestützter Datenmodelle

Soweit im Projektverlauf durch Dritte – insbesondere wissenschaftliche Einrichtungen, Forschungspartner oder datenhaltende Institutionen – KI-gestützte Prognose-, Verfügbarkeits- oder Auslastungsmodelle (z. B. für Parkraumauslastung, Bike-Sharing-Verfügbarkeit, Ladeinfrastruktur oder vergleichbare Mobilitätsdaten) bereitgestellt werden, hat der Auftragnehmer die technische Integrationsfähigkeit des SH-Navi für die Übernahme, Verarbeitung und nutzungsbezogene Berücksichtigung solcher Modelle im Rahmen der vertraglich vorgesehenen Systemarchitektur sicherzustellen.

(Die fachliche Entwicklung, Modellierung, wissenschaftliche Validierung, Prognosegüte sowie inhaltliche Qualitätssicherung solcher externen KI-Modelle sind nicht Leistungsgegenstand des Auftragnehmers, soweit nicht ausdrücklich abweichend vereinbart.

Leistungsgegenstand des Auftragnehmers ist vielmehr die technisch-strukturelle Befähigung des Systems, externe KI-basierte Prognoseergebnisse oder Auslastungsdaten – insbesondere über geeignete Schnittstellen, Datenplattformen oder Datenquellen – systemkompatibel zu übernehmen, softwareseitig zu verarbeiten und im Rahmen der verfügbaren Routing-, Informations- oder Entscheidungslogik nutzbar zu machen.

Hierzu zählen insbesondere:

- Schnittstellenfähigkeit zur Anbindung externer Daten- oder Prognosequellen,
- Integration prognostischer Daten in Datenmodelle, Routing- oder Informationslogiken,
- Berücksichtigung externer Prognosedaten im Rahmen vorausschauender Mobilitäts- und Routingfunktionen,
- nachvollziehbare Dokumentation der technischen Integration.

Die konkrete Integration steht unter dem Vorbehalt, dass die hierfür erforderlichen Datenmodelle, Schnittstellen, Nutzungsrechte, Datenqualitäten und technischen Voraussetzungen durch die jeweils verantwortlichen Drittanbieter oder Forschungspartner bereitgestellt werden.

Der Auftragnehmer schuldet keine eigenständige Verantwortung für die inhaltliche Richtigkeit, Prognosegüte oder wissenschaftliche Qualität extern entwickelter KI-Modelle, sondern ausschließlich deren vertragsgemäß vereinbarte technische Einbindung, soweit die hierfür erforderlichen Voraussetzungen vorliegen.

Soweit Prognosemodelle über externe urbane Datenplattformen, Datenräume oder vergleichbare Infrastrukturen bereitgestellt werden, ist die Systemarchitektur des SH-Navi möglichst offen, interoperabel und integrationsfähig auszugestalten, um eine spätere Einbindung entsprechender Datenquellen sachgerecht zu ermöglichen.

4 Technische Anforderungen

Die Systemarchitektur des SH-Navi basiert im Wesentlichen auf der Digitransit UI Plattform, als Frontend-Komponente und dem OpenTripPlanner, als Backend zur intermodalen Routenplanung, die öffentliche Verkehrsdaten (z.B. Verkehrsmittel, Fahrradstrecken, Fußwege und Autofahrten) in einer gemeinsamen Wegberechnung kombiniert.

Die anzubindenden Informationen der Verkehrsdaten stammen aus verschiedenen Datenquellen, die über unterschiedliche Datenplattformen integriert werden. Im Projektverlauf sind teilweise noch unklare Datenintegrationsschnittstellen zu konzipieren und auszuarbeiten. Abbildung 2 veranschaulicht den aktuellen Arbeitsstand.

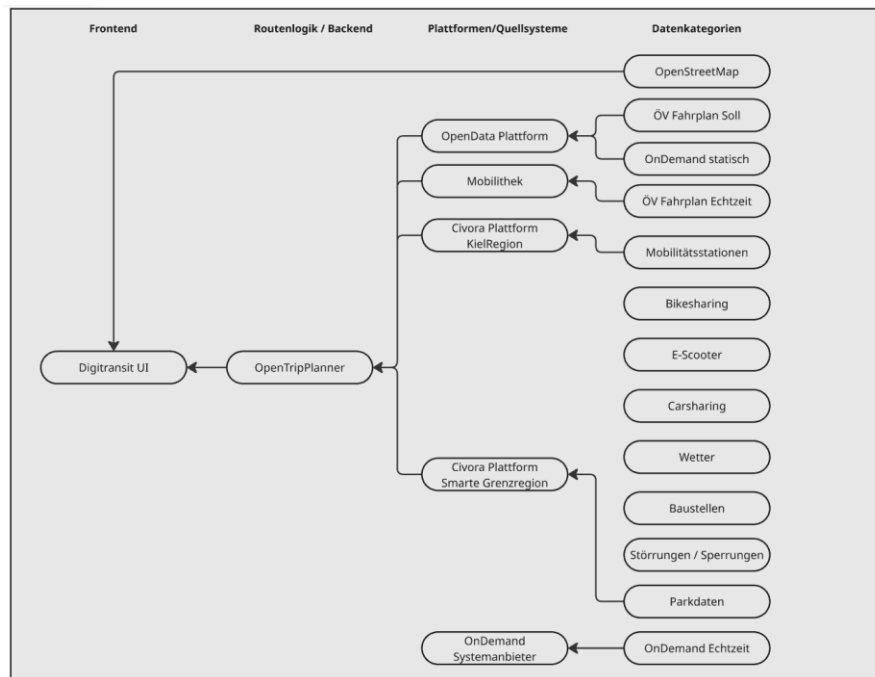


Abbildung 2: Rudimentäre Systemarchitektur

Der Auftragnehmer muss, tarifliche Informationen auf Basis von GTFS-Fares bzw. GTFS-Fares v2 und/oder NeTeX analysieren, modellieren und in ein für OpenTripPlanner nutzbares Format überführen. Dazu soll ein Umsetzungskonzept entwickelt werden, dass die Möglichkeiten und Grenzen von GTFS-Fares bewertet, die tariflichen Modellierungsoptionen von NeTeX analysiert und Unterschiede hinsichtlich Flexibilität und Anbindungsfähigkeit dokumentiert. Ziel ist es, ein technisches Modell für das Projekt zu entwerfen, inwiefern tarifliche Daten aufbereitet, exportiert und durch den OpenTripPlanner / das SH.Navi genutzt werden können. Dazu zählen vor allem:

1. die vorhandenen Tarifinformationen wo möglich in GTFS-Fares zu überführen
2. aufzeigen, welche Tarifkomponenten aktuell nicht über Fares abbildbar sind und aufzeigen, welche Erweiterungen nötig wären
3. eventuelle Limitierungen des OTP bez. NAH-SH-Tarifs in GTFS-Fares aufzeigen und Lösungen vorschlagen/umsetzen

4.1 Solldaten

Im Projektkontext ist sicherzustellen, dass Solldaten aus der Mobilithek im Format NeTEx (Network Timetable Exchange) als zentrale Datengrundlage für Routing und Auskunft in OpenTripPlanner (OTP) nutzbar gemacht werden.

Ziel ist die nachhaltige und interoperable Integration der Solldaten in die Gesamtarchitektur des SH-Navi.

Ein grundlegendes fachliches und technisches Verständnis von NeTEx sowie der OTP-Datenanforderungen wird vorausgesetzt, um eine zuverlässige Integration der Solldaten zu gewährleisten.

4.2 Tarifdaten

Im Projekt ist der bestehende SH-Tarif grundsätzlich in einem für den OpenTripPlanner nutzbaren Format abzubilden, vorzugsweise auf Basis von GTFS-Fares (inkl. GTFS-Fares v2). Dabei ist zu berücksichtigen, dass sich der Tarif aktuell in einem Transformationsprozess befindet.

Mit der geplanten Tarifreform in Schleswig-Holstein wird eine Vereinfachung der Tarifstruktur angestrebt. Die konkrete Ausgestaltung dieser Reform wird jedoch erst im Verlauf des Projekts veröffentlicht. Daraus ergibt sich die Anforderung, ein flexibles und anpassungsfähiges Daten- und Umsetzungsmodell zu entwickeln, das sowohl den bestehenden Tarif als auch zukünftige Änderungen berücksichtigen kann.

Aktuell basiert der Tarif auf einem relationsbezogenen Modell, bei dem sich die Preisbildung an den jeweiligen Start-Ziel-Relationen orientiert. Auch im Rahmen der Weiterentwicklung bleibt dieses Grundprinzip erhalten, wird jedoch stärker auf den regionalen Bereich fokussiert. Ergänzend dazu werden ein lokaler Bereich sowie ein netzweiter Gültigkeitsbereich eingeführt.

Die angestrebte Vereinfachung der Tarifstruktur erfolgt insbesondere durch eine Reduzierung der Preisstufen sowie eine Verschlinkung des Fahrkartensortiments.

Als fachliche Grundlage dienen die aktuell vorliegenden Spezifikationen des Schleswig-Holstein-Tarifs (Schnittstellenspezifikation Schleswig-Holstein-Tarif 2.0, Version V06, Stand 21.11.2025), die als Anlage Bestandteil der Ausschreibung sind (Anlage 8 – Schnittstellenspezifikation_SH-Tarif 2.0_V06 – Los 1). Diese sind hinsichtlich ihrer Abbildbarkeit in GTFS-Fares zu analysieren. Im Rahmen der Umsetzung sind insbesondere folgende Aspekte zu berücksichtigen:

- Überführung der bestehenden Tariflogik in GTFS-Fares, soweit technisch möglich,
- Identifikation von Tarifbestandteilen, die mit GTFS-Fares nicht oder nur eingeschränkt modellierbar sind,
- Entwicklung von Vorschlägen für notwendige Erweiterungen oder alternative Modellierungsansätze (z. B. unter Einbeziehung von NeTEx),
- sowie die Sicherstellung, dass zukünftige Tarifänderungen (insbesondere durch die Tarifreform) ohne grundlegende Systemanpassungen integrierbar sind.

Darüber hinaus sind bestehende Limitierungen des OpenTripPlanner im Kontext komplexer Tarifsysteme zu analysieren und geeignete Lösungsansätze zu entwickeln, um eine möglichst vollständige und korrekte Tarifabbildung im SH-Navi zu gewährleisten.

4.3 Open-Source-Datenquellen

Im Rahmen der Entwicklung sollen auch Open-Source-Datenquellen berücksichtigt und integriert werden. Die Auswahl und Integration der Datenquellen erfolgt unter Berücksichtigung von Datenqualität, Aktualität, Lizenzbedingungen sowie technischer Integrationsfähigkeit. Es ist sicherzustellen, dass alle verwendeten Open-Source-Daten rechtskonform genutzt und entsprechend gekennzeichnet werden.

Beispiele:

- Open Street Map (OSM)
- Open Charge Map (OCM) (<https://www.openchargemap.org/develop/api>)
- Open Charging Point Database (OCPDB) (<https://github.com/binary-butterfly/ocpdb>)
- Deutscher Wetterdienst (DWD) (<https://opendata.dwd.de/>)
- Autobahn GmbH - Autobahn API (<https://autobahn.api.bund.dev/>)

4.4 Tech Stack

4.4.1 Routing-Kern: OpenTripPlanner (OTP)

Als Routing-Kern kommt OpenTripPlanner (OTP) zum Einsatz. Dabei handelt es sich um eine etablierte Open-Source-Engine für multimodales Routing, die Tür-zu-Tür-Verbindungen unter Einbeziehung von ÖPNV, Fuß- und Radverkehr sowie weiteren Mobilitätsangeboten berechnet.

Die Software ist international produktiv im Einsatz und zeichnet sich durch hohe Flexibilität, Erweiterbarkeit und eine aktive Entwickler-Community aus. Dank der offenen Architektur

können spezifische Anforderungen – beispielsweise hinsichtlich Tariflogiken, besonderer Verkehrsangebote oder zusätzlicher Datenquellen – effizient umgesetzt werden.

4.4.2 Frontend: Digitransit UI (React)

Die Benutzeroberfläche basiert auf der Digitransit UI, einer modernen Webanwendung auf Basis von React. Sie ermöglicht eine intuitive Verbindungssuche und bietet ein responsives Design für Desktop- und Mobilgeräte.

Durch die komponentenbasierte Architektur ist eine strukturierte Weiterentwicklung sowie die Integration zusätzlicher Funktionen (z. B. Kartenmodule, Favoritenfunktionen oder Sharing-Optionen) langfristig sichergestellt.

4.4.3 API-Schicht: Digitransit GraphQL API

Die Systemkommunikation erfolgt über die Digitransit GraphQL API. Diese stellt eine flexible und performante Schnittstelle zwischen Frontend und Routing-Kern bereit. Die GraphQL-basierte Architektur ermöglicht bedarfsgerechte und effiziente Datenabfragen, transparente Datenmodelle sowie eine hohe Integrationsfähigkeit in bestehende IT-Strukturen. Damit wird eine skalierbare und zukunftssichere Systemintegration unterstützt.

4.4.4 Open Source

Die eingesetzten Kernkomponenten sind Open Source und in unterschiedlichen Kontexten produktiv erprobt. Diese basieren auf der bestehenden Anwendung des „stadtnavi“. Referenzimplementierungen sind unter anderem in:

- Helsinki (landesweite Digitransit-Plattform in Finnland)
- Herrenberg ([kommunale Mobilitätsplattform](#))
- KL Navi (<https://navi.kaiserslautern.digital/>)
- Ludwigsburg (<https://stadtnavi.swlb.de/>)

5 Weitere Anforderungen

5.1 Entwicklungsumgebung

Der Auftragnehmer stellt für die Dauer der Leistungserbringung eine geeignete Entwicklungsumgebung (Development) bereit, die die Umsetzung, Weiterentwicklung, Fehlerbehebung sowie technische Anpassung der Anwendung ermöglicht und einen effizienten, strukturierten und nachvollziehbaren Entwicklungsprozess sicherstellt.

Die Entwicklungsumgebung umfasst insbesondere die für die Leistungserbringung erforderlichen technischen Ressourcen, Werkzeuge und Systeme, insbesondere:

- Entwicklungsserver, Container- oder vergleichbare Entwicklungsumgebungen,
- Versionsverwaltung / Sourcecode-Management,
- Build-, Test- und Integrationswerkzeuge,
- Entwicklungs- und Testdatenbanken,
- erforderliche Schnittstellen- und Integrationsumgebungen.

In dieser Entwicklungsumgebung werden insbesondere folgende Tätigkeiten durchgeführt:

- Entwicklung neuer Funktionen,
- Anpassung und Weiterentwicklung bestehender Softwarekomponenten,
- Implementierung von Fehlerbehebungen,
- Entwicklung, Anbindung und Erprobung von Schnittstellen,
- technische Experimente, Prototypen und Tests.

Der Auftragnehmer ist für Bereitstellung, Betrieb, Wartung, Pflege, Absicherung und Funktionsfähigkeit der Entwicklungsumgebung verantwortlich.

Die Entwicklungsumgebung dient ausschließlich internen Entwicklungs-, Test- und Qualitätssicherungszwecken und ist nicht für fachliche Abnahmen, produktive Nutzung oder den Regelbetrieb vorgesehen, sofern nicht ausdrücklich abweichend vereinbart.

5.2 Übergabe der Entwicklungsleistung für Los 2

Zur Sicherstellung eines reibungslosen Übergangs zwischen den Entwicklungsleistungen dieses Loses 1 und dem DevOps / Betrieb des Loses 2 sind die nachfolgenden Aufgaben, Verantwortlichkeiten und Mindestanforderungen zu beachten. Zur Information über die Leistungen des Loses 2 wird die diesbezügliche Leistungsbeschreibung als **Anlage 2b** beigelegt.

Das im Rahmen von Los 1 beauftragte Entwicklerteam stellt lauffähige, deploybare Softwareartefakte bereit. Die Übergabe an Los 2 umfasst insbesondere:

- Bereitstellung einer funktionsfähigen und getesteten Version der Anwendung
- Dokumentation der Deploybarkeit, einschließlich Installations- und Konfigurationsanleitungen
- Schnittstellenbeschreibungen (z. B. APIs, Abhängigkeiten, externe Systeme)
- Definition und Dokumentation erforderlicher Betriebsparameter (z. B. Umgebungsvariablen, Skalierungsanforderungen, Ressourcenbedarf)
- Bereitstellung geeigneter Build- und Deployment-Artefakte (z. B. Container, Pakete)

Die bereitgestellten Inhalte müssen so beschaffen sein, dass eine eigenständige Übernahme und Inbetriebnahme durch Los 2 grundsätzlich möglich sind.

Die Auftragnehmer verpflichtet sich eng mit dem Auftraggeber und vom ihm benannten Dritten eng zusammenzuarbeiten der Lose 1 und 2 wirken eng zusammen und stellen sicher, dass:

- Anforderungen an Betrieb, Deployment und Infrastruktur frühzeitig zwischen den Auftragnehmern abgestimmt werden
- Rückfragen, Anpassungsbedarfe und Optimierungen zeitnah und transparent kommuniziert werden
- die Artefaktübergabe iterativ, im Rahmen des definierten Prozesses, erfolgen kann

Die detaillierte Ausgestaltung der Prozesse, Schnittstellen, Verantwortlichkeiten und Übergabeformate ist nach Zuschlagserteilung zwischen den Auftragnehmern beider Lose, gemeinsam mit dem Auftraggeber und der technischen Projektleitung verbindlich abzustimmen und zu dokumentieren.

6 Nicht funktionale Anforderungen

Die nichtfunktionalen Anforderungen gliedern sich nach Qualität, Sicherheit, sowie Anforderungen an die Erstellung) Diese sind in drei Kategorien zusammengefasst:

- **Kategorie 1 – Verbindliche Mindestanforderungen („Muss“):** Diese Anforderungen sind zwingend umzusetzen und zwingender Bestandteil der geschuldeten Leistung.
- **Kategorie 2 – Zielanforderungen / Soll-Anforderungen („Soll“):** Diese Anforderungen sind grundsätzlich umzusetzen, soweit technisch sinnvoll Priorisierung im Projekt möglich.

- **Kategorie 3 – Optionale / Weiterentwicklungsanforderungen („Kann“):** Diese Anforderungen sind nur bei Abruf / Sprint / Zusatzbeauftragung umzusetzen.

6.1 Qualität

| Lfd. Nr | Anforderung | Beschreibung | Kategorie |
|---------|--------------------------------------|--|--------------------|
| 6.1.1 | Aktualität | Die angebunden Daten / Datenbanken / Datenquellen unterliegen einer Aktualisierung (u.U. in Abhängigkeit Dritter, gerade im Bezug auf Open Source Daten über OSM) Es ist ein Konzept zur Datenaktualisierung zu erarbeiten und umzusetzen. | Kategorie 2 - SOLL |
| 6.1.2 | Mehrsprachigkeit | Eine Umstellung der angezeigten Sprache aller Inhalte soll leicht möglich sein. Deutsch als "App-Sprache" ist standardmäßig vorzusehen. Darüber hinaus mindestens noch eine englische Sprachvariante. | Kategorie 3 - KANN |
| 6.1.3 | Pop-Up zur Hilfe und Bedienbarkeit | Bei Benutzereingaben soll es dem User mit Hilfe von Pop-Ups oder ergänzenden, hinweisenden Texten erleichtert werden, die Eingaben zu verstehen und durchzuführen. Die Pop-Ups sollen beim "Überfahren" der Maus über das jeweilige Fenster oder beim Klicken eines Hilfesymbols erscheinen. | Kategorie 3 - KANN |
| 6.1.4 | Benutzer-Feedback | Der Benutzer soll immer in der Lage sein, ein Feedback an die Entwickler und die Betreiber der Software zu versenden - auch ohne eine vorher benötigte Registrierung. Hierbei muss ein Spam-Filter (Captcha) eingebaut werden. | Kategorie 3 - KANN |
| 6.1.5 | Benutzeroberfläche und Bedienbarkeit | Die Benutzeroberfläche der Anwendung muss intuitiv bedienbar sein und gängigen Richtlinien für benutzerfreundliches Design (z. B. Konsistenz, klare Navigation und verständliche Beschriftungen) entsprechen. | Kategorie 1 - MUSS |
| 6.1.6 | Responsive Design | Die Anwendung muss bezüglich ihrer Erscheinung und Schriftgröße für eine möglichst große Anzahl an Endgerät angepasst sein, dazu gehören Smartphones, Tablets, Laptops, Desktop-Computer, stationäre Infoscreens und Stelen. | Kategorie 1 - MUSS |

| | | | |
|--------|-------------------------------------|--|--------------------|
| 6.1.7 | Testfälle und Testfalldokumentation | Es müssen Tests durchgeführt werden. Auftretende Fehler müssen nachweislich durch neue Tests beseitigt werden. | Kategorie 1 - MUSS |
| 6.1.8 | Anwenderhandbuch | Es ist ein Anwenderhandbuch zu liefern. | Kategorie 1 - MUSS |
| 6.1.9 | Administratorenhandbuch | Es ist ein Administratorenhandbuch zu liefern. | Kategorie 1 - MUSS |
| 6.1.10 | Dokumentation | <p>Der Auftragnehmer ist verpflichtet, sämtliche im Rahmen der Entwicklungsleistung erstellten Softwareartefakte nachvollziehbar, vollständig und wartbar zu dokumentieren. Die Dokumentation ist Bestandteil der geschuldeten Leistung und umfasst insbesondere die Codebasis, die Softwarearchitektur, Schnittstellen, Daten- und Informationsflüsse, Build- und Deployment-Prozesse sowie für den Betrieb erforderliche Konfigurations- und Betriebsparameter.</p> <p>Die Dokumentation ist fortlaufend projektbegleitend zu erstellen und aktuell zu halten. Sie muss so beschaffen sein, dass der Auftraggeber, ein Dritter sowie der Auftragnehmer von Los 2 in die Lage versetzt werden, die Anwendung nachzuvollziehen, weiterzuentwickeln, zu warten und zu betreiben.</p> <p>Die Vorgaben des Open-Source-Gebots im Projekt Smarter-Leben gemäß Anlage 4 sind zu beachten.</p> <p>Eine User Story bzw. ein Sprint-Inkrement gilt nur dann als abgeschlossen, wenn die zugehörige Dokumentation vollständig und aktuell vorliegt.</p> | Kategorie 1 - MUSS |

6.2 Sicherheit

| Lfd. Nr | Anforderung | Beschreibung | Kategorie |
|---------|-----------------------------|--|--------------------|
| 6.2.1 | Datenschutz und Transparenz | Die gesetzlichen Datenschutz-Vorgaben der DSGVO sind einzuhalten. Des Weiteren ist der Anwender über eventuell gespeicherte Daten zu informieren. Jeder Benutzer kann seine gespeicherten Daten einsehen und die Löschung im Rahmen geltender Fristen verlangen. | Kategorie 1 - MUSS |

Auftraggeber: Digitalagentur Smarte Grenzregion GmbH

Projekt: SH-Navi

zentrale Plattform für intermodale Mobilitätsauskunft und -planung für Schleswig-Holstein

Los 1 Entwicklungsleistungen

Anlage 2 – Leistungsbeschreibung

Stand: 29.05.2026

| | | | |
|-------|--|--|--------------------|
| 6.2.2 | Verschlüsselte Datenübertragung | Jegliche Datenübertragung im Netzwerk muss - unabhängig vom Endgerät - verschlüsselt erfolgen. Ist eine verschlüsselte Übertragung aus technischen Gründen nicht möglich, muss der Benutzer darüber aufgeklärt werden. | Kategorie 1 - MUSS |
| 6.2.3 | Benutzerlogin | Die Bereiche, die nur für autorisierte Benutzer öffentlich sind, sind mit einem verschlüsselten (SSL) Login zu schützen. Auch nach erfolgreichem Login müssen alle Aktionen verschlüsselt ablaufen. Wenn ein Benutzer sein Passwort vergessen hat, wird ihm eine automatisch generierte E-Mail an seine Adresse gesendet. Diese enthält einen Link, der die Zurücksetzung seines Passworts ermöglicht. Die Funktion soll während der MVP Phase bereits konzeptionell erarbeitet werden. Die Umsetzung erfolgt erst in späteren Ausbaustufen. (Betrifft auch 2.04. und 2.05) | Kategorie 1 - MUSS |
| 6.2.4 | Passwortsicherheit | Ein Passwort muss aus mindestens 8 Zeichen bestehen und jeweils mindestens einen Großbuchstaben, ein Sonderzeichen und eine Ziffer enthalten. Das Passwort darf zu keiner Zeit im Klartext angezeigt werden. | Kategorie 1 - MUSS |
| 6.2.5 | Fehlgeschlagene Login-Versuche und Botabwehr | Wenn Benutzername und Passwort drei mal fehlerhaft eingegeben wurden, gibt es eine temporäre Sperre vor dem nächstmöglichen Versuch. Die Dauer dieser Sperre erhöht sich um einen festgelegten Wert (z.B. 1,3 , 5 Minuten). Maschinelle Versuche (durch Bots), Benutzername und Passwort zu knacken, sollen unmöglich gemacht werden. Falls dies nicht möglich ist, müssen die Bots erkannt werden und deren IP gespeichert werden. | Kategorie 1 - MUSS |
| 6.2.6 | Anwendungssicherheit | Die Anwendung muss so konzipiert und implementiert sein, dass sie wirksam gegen unbefugten Zugriff, Manipulation von Daten sowie die Ausnutzung bekannter und unbekannter Sicherheitslücken geschützt ist (z.B. Injektionsangriffe, Cross-Site-Scripting, Cross-Site-Request-Forgery, unsichere Authentifizierung sowie unzureichende Zugriffskontrollen) | Kategorie 1 - MUSS |

6.3 Erstellungsanforderungen

| Lfd. Nr | Anforderung | Beschreibung | Kategorie |
|---------|---|--|--------------------|
| 6.3.1 | Verwendung von OpenSource Komponenten | Digitransit UI, OpenTripPlanner und OpenStreetMap sind als Softwarekomponenten einzusetzen. | Kategorie 1 - MUSS |
| 6.3.2 | Code Repository | Im Rahmen der "Open Code" Initiative des Projekts, soll der Code der Anwendung in einem öffentlich zugänglichen (Git) Repository abgelegt werden. Es ist OpenCode zu nutzen und die Richtlinien / Gebote Open Source in SmarterLeben anzuwenden (siehe Anlage 7) | Kategorie 1 - MUSS |
| 6.3.3 | Einbindung und Verwendbarkeit von Altsystemen | Systeme, die bisher schon im Einsatz sind, sollen im Rahmen der skizzierten Systemübersicht angebunden werden. Dazu sind die jeweiligen Anbindungen (z.B. standardisierte Schnittstellen) wiederverwendet werden. | Kategorie 1 - MUSS |
| 6.3.4 | Verwendbarkeit externer Datenquellen | Inhalte externer Datenquellen sollen entsprechend der skizzierten Systemübersicht angebunden werden. | Kategorie 3 - KANN |
| 6.3.5 | Barrierefreiheit | Die Anwendung muss Kriterien der Barrierefreiheit entsprechen. Diese werden im Laufe des Projekts mit dem Auftragnehmer abgestimmt. | Kategorie 1 - MUSS |
| 6.3.6 | Logging | Sowohl wichtige Systemereignisse als auch Fehlermeldungen sollen protokolliert werden. | Kategorie 1 - MUSS |
| 6.3.7 | Kompatibilität | In Ergänzung zu "Responsive Design": Die Anwendung muss auf unterschiedlichen Endgeräten (z. B. Desktop, Tablet, Smartphone, Info-Stelen und Monitore) sowie in verbreiteten Webbrowsern zuverlässig nutzbar sein. | Kategorie 1 - MUSS |