# Technische Machbarkeit von ländlichem On-Demand-Verkehr: Anforderungen an eine Open-Source-Software-Architektur

Tingting Wang*, André Nitze*, Petra Hofstedt†, Sven Löffler†, Silvia Hennig#, Alexander Klinge#
*Technische Hochschule Brandenburg, †Brandenburgische Technische Universität Cottbus-Senftenberg, #neuland21 e.V.

**Abstract**

*On-Demand-Verkehr bezeichnet Mobilitätsangebote, die auf Bestellung verfügbar sind. Ziel ist es, durch flexible Mobilitätsangebote Lücken im ländlichen öffentlichen Personennahverkehr (ÖPNV) zu schließen, insbesondere in dünn besiedelten Räumen und zu Randzeiten des Tages. Die Einführung solcher Dienste gestaltet sich für viele Kommunen jedoch aufgrund der hohen Lizenzgebühren für proprietäre Software als äußerst schwierig, insbesondere in strukturschwachen Regionen, die solche Dienste besonders benötigen. Im Rahmen des Projekts "Open-Source-Software für ländlichen On-Demand-Verkehr" (OSLO) wurde die Umsetzbarkeit einer niedrigschwelligen Open-Source-Lösung geprüft. Diese Lösung soll langfristig in die Open-Source-Mobilitätsplattform bbnavi integriert werden, um die Nutzung vorhandener Mobilitätsdaten zu erleichtern, Interoperabilität zu gewährleisten und die Anpassung an andere Regionen zu vereinfachen. Das Konzept beinhaltet eine Software-Architektur, einen speziell für den ländlichen Nahverkehr entwickelten intermodalen Routing-Algorithmus und ein Betriebs- und Organisationsmodell ([1]). In diesem Beitrag wird beschrieben, wie eine effiziente Software-Architektur für das OSLO-Projekt entworfen wurde. Die zentralen Themen beinhalten die Strukturierung der Komponenten, die Anwendung von GTFS-Flex-V2, die Integration des angepassten Routing-Algorithmus sowie die Einbindung externer Schnittstellen.*

## 1. Einleitung

In ländlichen Regionen erleben Bedarfsverkehre einen Aufschwung. Laut der Branchenumfrage 2022 des Verbandes Deutscher Verkehrsunternehmen (VDV) sind deutschlandweit mehr als 400 Fahrzeuge im Linienbedarfsverkehr im Einsatz, und bis Ende 2022 wurden mehr als 80 Projekte erwartet, die sich auf die Integration von ÖPNV-basiertem Bedarfsverkehr konzentrieren (vgl. Abbildung 1).

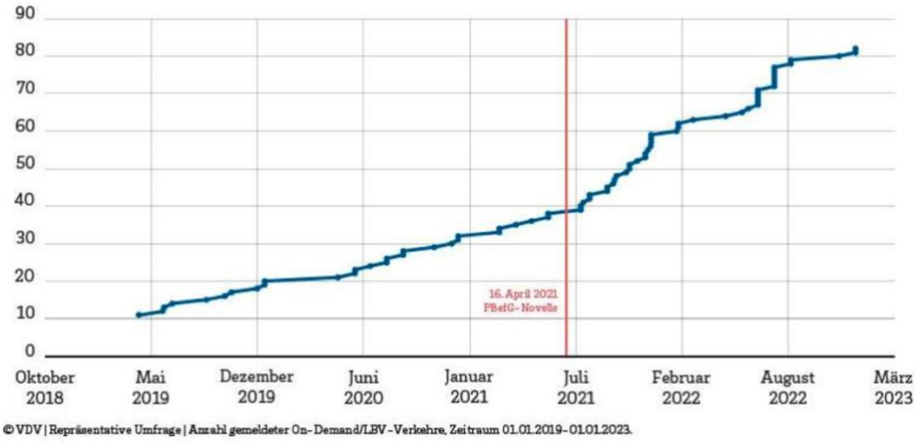


*Abbildung 1: Hochlauf von ÖV-integrierten digitale On-Demand-Verkehrsprojekte. Quelle: "Hochlauf der On-Demand-Verkehre im ÖPNV | VDV - Die Verkehrsunternehmen"*

Im Rahmen des OSLO-Projekts (01.11.2022 - 31.10.2023, Bundesministerium für Digitales und Verkehr) wurden Ergebnisse einer Machbarkeitsstudie zur Nutzung von Open-Source-Software für die Integration eines neuen On-Demand-Verkehrssystems (ODV) in Spremberg erarbeitet. Dies umfasste eine Bedarfsanalyse basierend auf vorhandenen Daten und in Zusammenarbeit mit der Bevölkerung. Zudem wurden der Entwicklungsstand von Open-Source-Software im Bereich Mobilitätsplattformen, Routing und On-Demand-Konzessionen zusammengefasst sowie die Softwarearchitektur der Open-

Source-Mobilitätsplattform "bbnavi" aufbereitet. Des Weiteren wurden verfügbare Routing-Algorithmus und -logiken analysiert, um Möglichkeiten zur Vermeidung von Parallelverkehr und zur Sicherung von Anschlüssen zu diskutieren ([1]). An anderer Stelle wurde bereits auf die Routing- Algorithmus ([6]) eingegangen. In diesem Beitrag wird die Entwicklung der Software-Architektur näher beschrieben.

## 2. Methode

Dieses Projekt basiert auf der Open-Source-Mobilitätsplattform bbnavi und verfolgt das Ziel, eine kontinuierliche Analyse und Erweiterung durchzuführen. bbnavi ist eine multi- und intermodale Mobilitätsplattform mit Fokus auf Kommunen. Die Mobilitätsplattform ermöglicht die kombinierte Fahrplanauskunft für insbesondere klimafreundliche Mobilitätsangebote (unter anderem ÖPNV, Fahrrad, Sharing, Mitfahren) und die Darstellung von Daten auf Karten – auch von Live-Daten (zum Beispiel Fahrradverfügbarkeit bei Ausleihstationen). Insbesondere die Daten von lokalen Mobilitätsangeboten im Land Brandenburg werden eingebunden ([3]). Im Fokus steht die Weiterentwicklung der Softwareinfrastruktur, um den spezifischen Anforderungen des On-Demand-Verkehrs gerecht zu werden. Unsere Methode ([5]) umfasst mehrere Schritte, die im Folgenden näher erläutert werden.

- **Quellcode-Analyse:** Im Rahmen der Codeanalyse werden nicht nur die strukturellen Aspekte des Codes betrachtet, sondern auch eine eingehende Untersuchung der verwendeten Open-Source-Software durchgeführt. Hierbei liegt der Fokus darauf, die Funktionsweise dieser Softwarekomponenten im Kontext des Gesamtprojekts zu verstehen. Neben der Codeebene werden die Datenquellen und Datenübertragungsformate analysiert, um sicherzustellen, dass eine reibungslose Interaktion zwischen verschiedenen Systemkomponenten gewährleistet ist.

- **Lizenzprüfung:** Es erfolgt eine Überprüfung der Lizenzen, um sicherzustellen, dass das Softwareprojekt den geltenden rechtlichen Anforderungen entspricht und sich vor allem für den Open-Source-Einsatz durch Kommunen eignet. Alle lizenzbezogenen Informationen werden erfasst und daraufhin geprüft, ob sie mit den Projektzielen übereinstimmen.

- **Schnittstellenprüfung:** Die Verbindungen und Interaktionen zwischen verschiedenen Systemkomponenten oder Schnittstellen innerhalb der bbnavi-Plattform werden überprüft. Dies beinhaltet die Untersuchung, ob die Datenübertragung reibungslos erfolgt, die Schnittstellen korrekt implementiert sind und die Kommunikation zwischen den Komponenten ohne Probleme erfolgt.

- **Vergleich von Versionen und Standards:** Dieser Schritt beinhaltet die Überprüfung und Gegenüberstellung unterschiedlicher Softwareversionen sowie der angewendeten Standards innerhalb der Plattform. Dies dient dazu, festzustellen, ob die verwendeten Versionen auf dem neuesten Stand sind und den geltenden Standards entsprechen. Dabei können auch mögliche Verbesserungen, Aktualisierungen oder Anpassungen identifiziert werden, um Verbesserungsmöglichkeiten zu identifizieren und die Integration aktueller bewährter Verfahren zu bewerten.

- **Ableitung funktionaler und nicht-funktionaler Anforderungen:** Funktionale Anforderungen beschreiben die spezifischen Aufgaben und Leistungen, die das System erbringen muss, wie zum Beispiel die Möglichkeit, Fahrten online zu buchen oder eine Schnittstelle zur Integration von Echtzeitdaten bereitzustellen. Nicht-funktionale Anforderungen hingegen betreffen Qualitätsmerkmale des Systems, die nicht direkt mit den Funktionen zusammenhängen. Hierzu

gehören Aspekte wie Leistungsfähigkeit, Sicherheit, Benutzerfreundlichkeit und Skalierbarkeit. Beispiele hierfür sind eine angemessene Latenzzeit für die Routenplanung und die Implementierung von Datenschutzmechanismen.

## 3. Ergebnis

In diesem Kapitel wird in Abschnitt 3.1 die Software-Architektur von bbnavi vorgestellt, Abschnitt 3.2 beschäftigt sich mit den Erweiterungen zur Umsetzung von ODV auf der Basis von bbnavi.

### 3.1 Open-Source-Mobilitätsplattform bbnavi

Die Mobilitätsplattform bbnavi wurde auf Open-Source-Software-Projekten OpenTripPlanner (OTP)[1] und Digitransit UI[2] entwickelt. bbnavi bietet die Möglichkeit die Daten strukturiert und standardisiert zu erfassen, um diese über die Auskunftsplattform bbnavi zu nutzen und als Open Data zu veröffentlichen ([2]). OTP greift auf verschiedene Datenquellen zu, darunter General Transit Feed Specification (GTFS) Daten, die den Fahrplan- und Routeninformationen der Verkehrsbetriebe in Berlin und Brandenburg entsprechen. Weitere Quellen sind OpenStreetMap (Berlin-Brandenburg) für Straßennetze sowie Echtzeitdaten über die Standorte von mehreren hundert Fahrradvermietungen in Berlin und Brandenburg (GBFS). Zusätzlich existieren GTFS-Flex-Daten, die den Routeninformationen der Rufbus-Linien entsprechen (vgl. Abbildung 2).

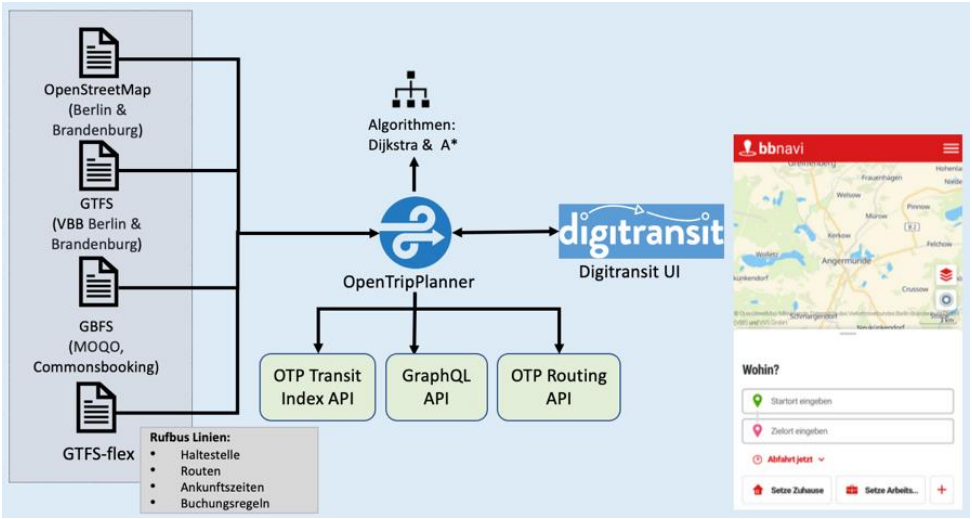


*Abbildung 2: Darstellung der Software-Architektur von bbnavi*

OTP bietet verschiedene APIs und Endpunkte, die es Entwicklern ermöglichen auf verschiedene Funktionalitäten im Zusammenhang mit dem öffentlichen Nahverkehr und der multimodalen Reiseplanung zuzugreifen (vgl. Abbildung 2). Die OTP-Transit-Index-API liefert Informationen, die aus den GTFS-Feeds abgeleitet wurden. OTP erstellt einen transitiven Graphen aus den Datenquellen, um die öffentlichen Verkehrsrouten berechnen zu können. Dieser Graph enthält Informationen über Haltestellen, Routen, Fahrpläne, Umsteigeverbindungen usw. Die GraphQL API ermöglicht den Zugriff auf den zugrundeliegenden Verkehrsnetzgraph, den OTP für die Routenberechnung verwendet. Die Berechnung des Graphen ermöglicht effiziente Routenplanungen. OTP verwendet Algorithmen wie Dijkstra oder A* für die Berechnung der besten Routen unter Berücksichtigung verschiedener Faktoren wie Entfernung, Reisezeit, Umstiege, Fußwege usw. Die Routing-API bietet eine Möglichkeit zur Planung von Routen und zur Abfrage von Informationen über Haltestellen und Fahrpläne im öffentlichen

[1] https://www.opentripplanner.org/
[2] https://digitransit.fi/en/developers/services/5-digitransit-ui/

Nahverkehr mit GraphQL. Über diese Komponente können also auch Zusatzinformationen eingespeist werden, die für das Routing relevant sind ([8]).

Digitransit UI ermöglicht den Nutzer:innen die Eingabe von Start- und Zielorten sowie Reisepräferenzen ([4]). Es verarbeitet Eingaben und sendet entsprechende Anfragen an OTP, um die Routenplanung anzustoßen und mögliche Verbindungen als Reiseplanungsdaten abzurufen (vgl. Abbildung 2).

### 3.2 Erweiterungen zur Umsetzung von ODV

Die Analyse ergab vier wesentliche, notwendige Erweiterungen zur Umsetzung von ODV auf der Basis von bbnavi (vgl. Abbildung 3):

1) die Nutzung von GTFS-Flex-V2 für die Nahe-Echtzeit-Generierung dynamischer Fahrpläne,

2) die Integration des angepassten Routing-Algorithmus in die OTP-Architektur,

3) die Entwicklung einer Schnittstelle für die Anbindung von Buchungssystemen und

4) die Anbindung von bbnavi über eine Schnittstelle mit der Fahr-App.

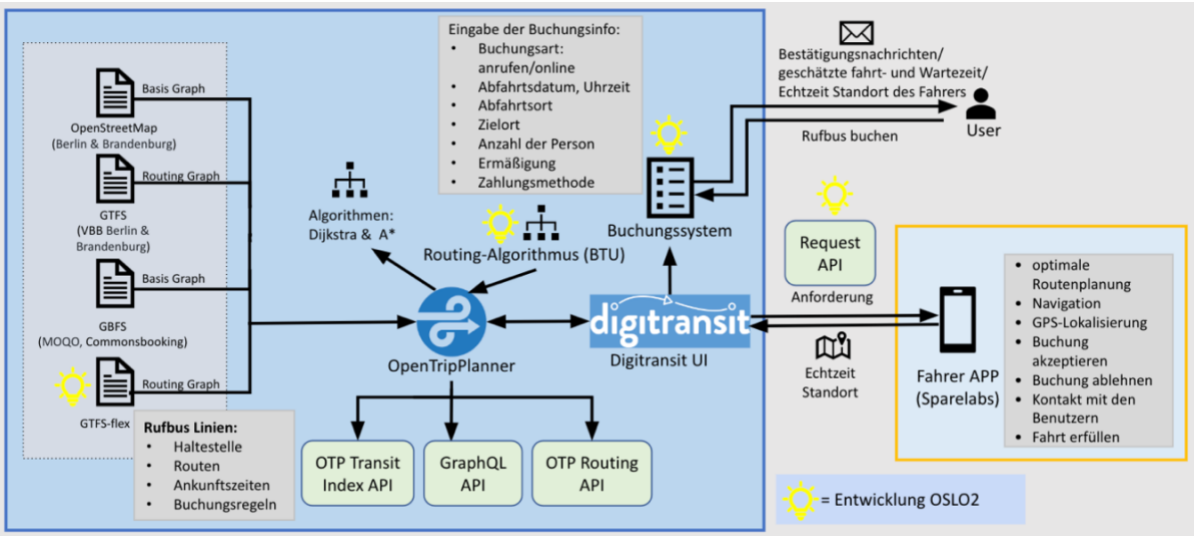


*Abbildung 3: Darstellung der Software-Architektur mit Erweiterungen für OSLO auf der Basis von bbnavi. Quelle: "Wissenschaftlicher Schlussbericht: Open-Source-Software für ländlichen On-Demand-Verkehr OSLO Machbarkeitsstudie"*

**1) Die Nutzung von GTFS-Flex-V2 für die Nahe-Echtzeit-Generierung dynamischer Fahrpläne.** Da GTFS-Flex eine Erweiterung des GTFS-Formats darstellt, enthalten GTFS-Flex-Daten im Allgemeinen die gleichen Grundelemente wie GTFS-Daten (Agentur, Haltestellen, Routen usw.), jedoch auch zusätzliche Informationen, die speziell für die Darstellung von flexiblen Verkehrsdiensten relevant sind. Die neueste Version, GTFS-Flex v2 (vgl. Abbildung 4), besteht aus zwei Erweiterungen, die darauf abzielen, die Vielfalt der nachfrageabhängigen Dienste zu modellieren, die nicht immer denselben festen Haltestellen folgen. Eine davon ist GTFS-FlexibleTrip, was flexible Dienste bezeichnet, die zwar einem bestimmten Fahrplan folgen, jedoch auf die individuelle Nachfrage einzelner Fahrgäste reagieren. Die andere ist GTFS-BookingRules, welche Buchungsinformationen für von Fahrgästen angeforderte Dienste bereitstellt und dabei GTFS-FlexibleTrips verwendet ([7]). Das beinhaltet beispielsweise Angaben dazu, wie weit im Voraus die Buchung erfolgen soll oder welche Telefonnummer für Buchungen angerufen werden kann. Mit der Generierung von Dateien innerhalb dieser Spezifikation ist es möglich, einen mit der bestehenden Architektur kompatiblen Transit Feed für On-Demand-Verkehre zu erzeugen.

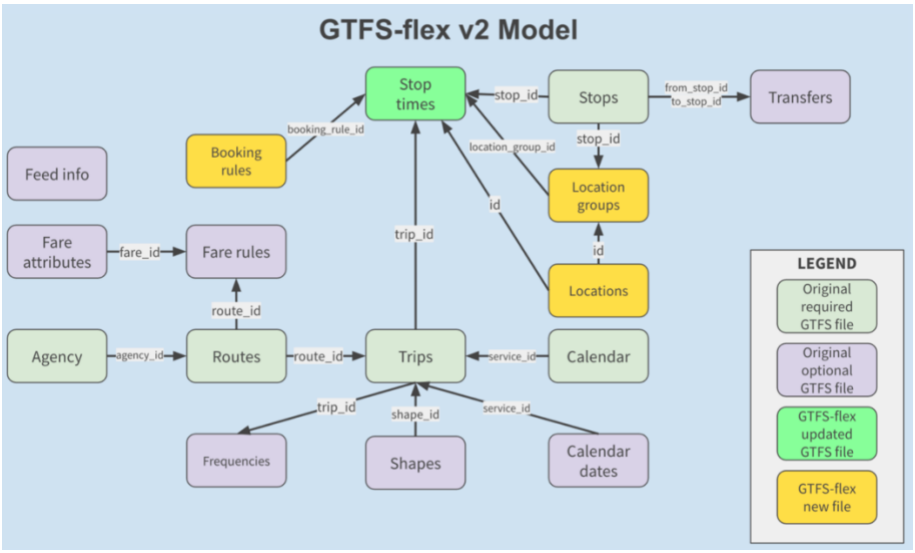


*Abbildung 4: Darstellung des GTFS-Flex v2 Models. Quelle: https://github.com/MobilityData/gtfs-flex/blob/130eb67c7dfac846b74625747e2623429d9f8f64/spec/GTFS_GTFS-flex_v2_Schema_Diagram.png*

**2) Die Integration des angepassten Routing-Algorithmus in die OTP-Architektur.** Um optimale Routen für kommunale On-Demand-Verkehre (ODV) zu finden, wird ein speziell für den ländlichen Nahverkehr entwickelten intermodalen Routing-Algorithmus entwickelt und in die Routing-Komponente (OTP-Routing-API) integriert ([6]).

**3) Die Entwicklung einer Schnittstelle für die Anbindung von Buchungssystemen**. Für den reibungslosen Betrieb des On-Demand-Verkehrs ist die Entwicklung eines Buchungssystems notwendig, das den Fahrtwunsch entgegennimmt. Nutzerinnen und Nutzer haben die Möglichkeit, den Rufbus entweder telefonisch oder online zu buchen und dabei relevante Buchungsinformationen wie Zeit, Abfahrtsort, Zielort usw. anzugeben (vgl. Abbildung 3). Eine unterstützende Vielfalt von Zahlungsmöglichkeiten ist ebenfalls erforderlich. Nach Auswahl einer Zahlungsmethode werden die Nutzerinnen und Nutzer auf die entsprechende Zahlungsseite weitergeleitet. Zusätzlich ist eine zeitnahe Bearbeitung der Buchungsinformationen und deren Übermittlung an die Fahr-App durch die Request-API von entscheidender Bedeutung.

**4) Die Anbindung von bbnavi über eine Schnittstelle mit der Fahr-App.** Für den Betrieb von On-Demand-Verkehr ist es notwendig, dass Fahrzeuge zu jeder Zeit neue Routen- Informationen erhalten und die Route bedarfsgerecht verändert werden kann. Daneben ist auch für die Routenplanung ein genauer Standort der Fahrzeuge unerlässlich. Die Standorte im Netzgraph sollten annähernd in Echtzeit und möglichst feingranular, also auf Knotenebene des Straßennetzes, zur Verfügung stehen, um eine praxistaugliche Routenplanung in angemessener Zeit zu ermöglichen. Dabei wird die übliche GPS-Lokalisierung mobiler Endgeräte genutzt. Bei der Realisierung über die Fahr-App müssen diese Koordinaten jedoch für die Routenplanung explizit an bbnavi zurückgegeben werden. Darüber hinaus benötigen die Fahrer eine Funktion zur Bearbeitung von Mobilitätsanfragen, um eine direkte, digitale Rückmeldung (Bestätigung der Anforderungen, voraussichtliche Wartezeiten, Änderungen, usw.) an den Nutzern zu ermöglichen (vgl. Abbildung 3). Die Analyse der Anforderungen hat ergeben, dass von der Realisierung einer eigenen mobilen Fahr-App abzusehen ist. Die Erwartungen von Nutzern in Bezug auf Komfort (Geschwindigkeit, Sprachausgabe), Visualisierung (2D/3D, Karten, Verkehrsdichte), Funktionsumfang (Offline-Modus, Stauumfahrung) sind zu hoch für eine eigene Entwicklung im Rahmen eines Umsetzungsprojekts. Die abschließende Erweiterung ist daher die Integration über eine

Schnittstelle mit einer Fahr-App von Drittanbietern, wie zum Beispiel Spare Labs [3], der auf die Entwicklung von Tools für On-Demand- und geteilte Transportdienste spezialisiert ist.

## 4. Fazit

Die technische Machbarkeit einer Lösung für On-Demand-Verkehr auf der Basis der Open-Source-Mobilitätsplattform bbnavi konnte bestätigt werden. Eine Umsetzung ist unter den Rahmenbedingungen – darunter die technische Kompatibilität mit existierenden Verkehrssystemen und Infrastrukturen sowie die Bereitschaft von Drittanbietern zur Kooperation bei der Integration von Buchungssystemen und Fahr-Apps – für die Zielgruppe möglich. Mit dem entwickelten Konzept ist eine Umsetzung der genannten Schritte in einem Folgeprojekt möglich. Diese Umsetzung unterstützt die ländlichen Gemeinden maßgeblich beim Aufbau und Betrieb wirtschaftlich tragfähiger ODVe. Trotz dieser positiven Entwicklung sind jedoch einige Risiken und Herausforderungen zu berücksichtigen. Die Integration verschiedener Systeme und Schnittstellen könnte technisch anspruchsvoll sein und zusätzliche Komplexität mit sich bringen. Die Handhabung personenbezogener Daten bei der Buchung von Fahrten erfordert strikte Datenschutzmaßnahmen, um gesetzlichen Anforderungen zu genügen. Die Zusammenarbeit mit externen Partnern, insbesondere bei der Integration von Buchungssystemen und Fahr-Apps, erfordert eine effektive Kommunikation und Koordination. Die langfristige Finanzierung des Projekts und die Sicherstellung der Ressourcen für den Betrieb sind entscheidende Herausforderungen.

## 5. Danksagung

An dieser Stelle möchten wir uns herzlich beim Bundesministerium für Digitales und Verkehr (BMDV) für die Förderung unseres Projekts „OSLO" (Förderkennzeichen: 19FS1005B) bedanken.

## 6. Literaturverzeichnis

[1] Alexander Klinge, André Nitze, Petra Hofstedt, Sven Löffler, Silvia Hennig, Tingting Wang (2023): Wissenschaftlicher Schlussbericht: Open-Source-Software für ländlichen On-Demand-Verkehr OSLO Machbarkeitsstudie, [online] https://neuland21.de/wp-content/uploads/2023/10/231026-anlage2-oslo-wiss-schlussbericht-final.pdf

[2] bbnavi (2024): bbnavi: ein Pilotprojekt der DigitalAgentur Brandenburg, [online] https://github.com/bbnavi [26.02.2024]

[3] bbnavi (2024): Mobilitätsplattform für Kommunen in Brandenburg, [online] https://bbnavi.de/ [26.02.2024]

[4] Digitransit-UI (2024): For developers, [online] https://digitransit.fi/en/developers/ [26.02.2024]

[5] L. Dobrica and E. Niemela (2002), "A survey on software architecture analysis methods," in IEEE Transactions on Software Engineering, vol. 28, no. 7, pp. 638-653, doi: 10.1109/TSE.2002.1019479.

[6] Löffler, Sven; Becker, Ilja; Hofstedt, Petra; Nitze, André; Hennig, Silvia; Klinge, Alexander (2023): Planung des Ländlichen On-Demand-Verkehr - Probleme, Analyse und Algorithmen. INFORMATIK 2023 - Designing Futures: Zukünfte gestalten. DOI: 10.18420/inf2023_177. Bonn: Gesellschaft für Informatik e.V.. PISSN: 1617-5468. ISBN: 978-3-88579-731-9. pp. 1739-1750.

[7] MobilityData (2024): MobilityData/gtfs-flex: GTFS-Flex v2, [online] https://github.com/MobilityData/gtfs-flex/blob/master/spec/reference.md [26.02.2024]

[8] OpenTripPlanner (2024): OpenTripPlanner Multimodal Trip Planning, [online] https://www.opentripplanner.org/ [23.02.2024]

[3] https://sparelabs.com/en