Kommunale Micromobility-Dashboards in Eigenentwicklung

Praxisleitfaden zu Datenzugang, Schnittstellen, Geodaten und Betrieb

Holger Friedel, David Rüdiger und Andreas Schmidt
Connected Mobility Düsseldorf GmbH

Gefördert durch:

Bundesministerium für Verkehr

mFUND
Das Startkapital für die Mobilität der Zukunft

28. Juli 2026

Dokumentinformationen

Titel	Kommunale Micromobility-Dashboards in Eigenentwicklung
Version	1.0
Projekt	ScooP2City – Service-Ökosystem für stadtverträgliche geteilte Mobilität
Förderkennzeichen	01F2261D
Datum	28. Juli 2026
Autoren	Holger Friedel, David Rüdiger und Andreas Schmidt Connected Mobility Düsseldorf GmbH
Lizenz	© 2026 Connected Mobility Düsseldorf GmbH. Dieses Werk ist lizenziert unter CC BY 4.0: https://creativecommons.org/licenses/by/4.0/. Ausgenommen sind Logos, Marken und Inhalte Dritter.
Status	Veröffentlicht

Dieses Dokument entstand im Rahmen des Projekts *ScooP2City*, das vom Bundesministerium für und Verkehr (BMV) im Rahmen der Innovationsinitiative mFUND unter dem Förderkennzeichen 01F2261D gefördert wurde. Es dokumentiert einen fortschreibbaren Praxisleitfaden zur kommunalen Eigenentwicklung von Micromobility-Dashboards. Zukünftige Versionen können bestehende Inhalte überarbeiten, erweitern oder präzisieren.

Inhaltsverzeichnis

1 Einleitung

Shared-Mobility-Angebote wie E-Scooter-, Fahrrad- oder Lastenradverleihsysteme sind in vielen Städten Bestandteil des Mobilitätsangebots. Für Kommunen entstehen damit neue Aufgaben: Flotten müssen nachvollzogen, räumliche Entwicklungen bewertet, Abstellregeln überprüft und Auswirkungen auf den öffentlichen Raum eingeordnet werden. Das Projekt *ScooP2City* [1] entwickelt hierzu ein datengestütztes Service-Ökosystem für den Aufbau und Betrieb stadtverträglicher Shared-Mobility-Angebote.

Die im Dokument *ScooP2City Anforderungen* [2] konsolidierten Bedarfe beruhen auf einer schrittweisen Erhebung und Reflexion mit Kommunen und weiteren Beteiligten. Wiederkehrende Themen sind insbesondere die Integration und Qualitätssicherung von Daten, kommunales Monitoring, Analyse und Planung, adressatengerechte Auswertungen, standardisierte Schnittstellen sowie die kontrollierte Bereitstellung von Informationen. Dieser Leitfaden greift diese Ergebnisse in verdichteter Form auf, ohne die zugrunde liegenden User Stories oder eine projektspezifische Plattformspezifikation zu übernehmen.

Ein kommunales Micromobility-Dashboard kann diese Aufgaben unterstützen. Es führt Daten verschiedener Anbieter zusammen, ergänzt sie um kommunale Geo- und Fachdaten und stellt daraus abgeleitete Kennzahlen und Karten bereit. Die Visualisierung ist jedoch nur die sichtbare Spitze. Die eigentliche Qualität entsteht durch verbindliche Datenanforderungen, robuste Schnittstellen, nachvollziehbare Berechnungsregeln und einen dauerhaft organisierten Betrieb.

Dieser Leitfaden soll Kommunen ausdrücklich dazu ermutigen, eine Eigenentwicklung oder eine gezielt beauftragte Entwicklung als realistische Option zu prüfen. Ein erster nutzbarer Stand muss weder alle Anbieter noch alle denkbaren Auswertungen umfassen. Bereits eine schrittweise Lösung mit klar begrenzten Anwendungsfällen, einer belastbaren PostgreSQL/PostGIS-Datenbasis und wenigen nachvollziehbaren Auswertungen kann einen hohen praktischen Nutzen schaffen. Die technische Umsetzung ist anspruchsvoll, aber mit verbreiteten Kompetenzen und einer sauberen Aufgabenverteilung beherrschbar.

Für die technische Umsetzung trifft dieser Leitfaden eine klare Grundannahme: PostgreSQL mit der Erweiterung PostGIS ist für die Verarbeitung kommunaler Micromobility-Daten im Open-Source-Umfeld aus unserer Sicht die praktisch alternativlose Basiskombination. Damit ist nicht gemeint, dass keine anderen Datenbanksysteme existieren. Ausschlaggebend ist vielmehr, dass PostgreSQL eine ausgereifte relationale Datenhaltung mit Transaktionen, Integritätsregeln und der nativen Speicherung und Verarbeitung von JSON und JSONB verbindet [3, 4]. PostGIS ergänzt diese Grundlage um Geometrietypen, räumliche Indizes und Funktionen für Verschneidungen, Entfernungsanalysen und Punkt-in-Polygon-Prüfungen [5].

Damit lassen sich technische Abrufinformationen, unveränderte API-Antworten, normalisierte Fahrzeug-, Fahrt- und Ereignisdaten, Qualitätsmerkmale sowie kommunale Gebietsgrenzen und Zonen in einer gemeinsamen offenen Datenbasis verwalten. Das reduziert zusätzliche Schnittstellen, verbessert die Nachvollziehbarkeit und vermeidet für den Datenkern die Abhängigkeit von proprietären Datenbanklizenzen oder einem einzelnen Produktanbieter. Im Folgenden wird PostgreSQL/PostGIS deshalb als zentrale Datenbasis vorausgesetzt. Werkzeuge für Datenabruf, Visualisierung, Karten, Berichte oder Exporte bleiben dagegen austauschbar.

Viele dieser Aufgaben fallen unabhängig davon an, ob eine Kommune selbst entwickelt, eine Entwicklung beauftragt oder ein fertiges Produkt beschafft. Anwendungsfälle und Kennzahlen müssen fachlich definiert, Datenlieferungen vereinbart, kommunale Geodaten gepflegt, Ergebnisse abgenommen und der Betrieb verantwortlich begleitet werden. Eine Eigenentwicklung erzeugt diese Aufgaben daher nicht erst; sie macht sie lediglich sichtbarer und gibt der Kommune mehr Einfluss auf Datenmodell, Prüfregeln und Weiterentwicklung.

Eine Eigen- oder Auftragsentwicklung ist realistisch, wenn die erforderlichen Kompetenzen im Projektteam oder bei einem Dienstleister abgedeckt sind. Dazu gehören vor allem Skriptprogrammierung für automatisierte Abrufe, der sichere Umgang mit Web-APIs, Authentifizierung und JSON, SQL einschließlich JSON-Funktionen und räumlicher Abfragen in PostGIS sowie die Erstellung nachvollziehbarer Auswertun-

gen mit einem Self-Service-BI-Werkzeug. Nicht jede Kompetenz muss in einer Person oder vollständig innerhalb der Verwaltung liegen. Unverzichtbar bleiben jedoch eine kommunale Produktverantwortung, prüfbare Anforderungen und die Fähigkeit, Ergebnisse fachlich abzunehmen.

Der Leitfaden richtet sich an Fachverwaltungen, Mobilitätsmanagement und kommunale IT. Er soll dazu anregen, eine eigene oder beauftragte Entwicklung als ernsthafte Option zu prüfen, ohne eine bestimmte Software für Datenabruf, Aufbereitung oder Darstellung vorzuschreiben.

Leitgedanke: Die benötigten Daten werden aus kommunalen Anwendungsfällen abgeleitet – nicht umgekehrt. Ein technisch verfügbares Feld ist noch keine fachlich sinnvolle Kennzahl.

2 Kommunales Zielbild und Anwendungsfälle

Die *ScooP2City Anforderungen* verdichten die erhobenen Bedarfe zu wiederkehrenden Nutzungskontexten: kommunales Monitoring und operative Steuerung, Analyse und Planung, Integration und kontrollierte Bereitstellung von Daten, Self-Service-Auswertungen, strukturierte Wissensangebote sowie die Aufnahme externer Hinweise. Für ein kommunales Micromobility-Dashboard stehen vor allem die ersten vier Bereiche im Mittelpunkt; Wissensangebote und Rückmeldungen bestimmen jedoch die organisatorische Einbettung und die Weiterentwicklung des Datenprodukts [2].

2.1 Typische kommunale Fragestellungen

Ein Dashboard kann unter anderem folgende Fragen beantworten:

- Wie viele Fahrzeuge befinden sich aktuell im Stadtgebiet oder in einzelnen Teilräumen?
- Wie entwickeln sich Flottengröße, Fahrten und Auslastung im Zeitverlauf?
- Wo beginnen und enden besonders viele Fahrten?
- Werden ausgewiesene Abstellflächen genutzt und Verbotszonen eingehalten?
- Wie lange stehen Fahrzeuge ungenutzt an einem Ort?
- Werden Flottenobergrenzen, Mindestverteilungen oder Bediengebiete eingehalten?
- Wie zuverlässig und aktuell liefern die Anbieter ihre Daten?
- Welche Auswirkungen haben neue Sharingstationen oder geänderte Regeln?

Nicht jede Frage benötigt dieselben Daten. Für einen aktuellen Überblick können Echtzeitdaten genügen. Für abgeschlossene Fahrten, Statuswechsel oder langfristige Evaluationen werden historische Daten und eindeutige Aufbewahrungsregeln benötigt.

2.2 Vom Anwendungsfall zur Datenanforderung

Für jeden Anwendungsfall sollte die Kommune sieben Punkte dokumentieren:

1. fachliche Frage und erwartete Entscheidung,
2. notwendige Datenfelder und Quellen,
3. zeitliche und räumliche Auflösung,
4. zulässiger Nutzerkreis,
5. Aufbewahrungs- und Löschfrist,
6. Berechnungsregel der Kennzahl,
7. Verfahren zur fachlichen Plausibilisierung.

2.3 Eigenentwicklung realistisch einordnen

Die Entscheidung zwischen Eigenentwicklung, Auftragsentwicklung und Produktbeschaffung betrifft vor allem die Verteilung von Verantwortung, Know-how und Gestaltungsspielraum. Die fachliche Vorarbeit bleibt in allen Varianten weitgehend gleich. Auch bei einer Ausschreibung muss die Kommune beschreiben, welche Entscheidungen unterstützt werden sollen, welche Daten und Qualitätsmerkmale erforderlich sind, wie kommunale Geometrien gepflegt werden und nach welchen Regeln Ergebnisse abgenommen werden.

Tabelle 2.1: Aufgaben und Kompetenzen nach Beschaffungsmodell

Bereich	In jeder Umsetzungsvariante erforderlich	Für Eigen- oder Auftragsentwicklung benötigte Kompetenz
Fachliches Zielbild	Anwendungsfälle, Kennzahlen, Zuständigkeiten und Abnahmekriterien festlegen.	Fachliche Anforderungen in testbare Daten- und Funktionsregeln übersetzen.
Auswertung	Zielgruppen, fachliche Definitionen, Filter, Karten und Berichtsinhalte bestimmen.	Ein Self-Service-BI-Werkzeug konfigurieren und Kennzahlen so umsetzen, dass Herkunft und Berechnung nachvollziehbar bleiben.
Betrieb	Produktverantwortung, Monitoring, Anbieterkommunikation, Änderungsmanagement und Finanzierung sichern.	Skripte, Schnittstellen und Datenbank betreiben, testen, dokumentieren und bei Versionswechseln anpassen.

Eine Kommune muss diese Kompetenzen nicht vollständig selbst vorhalten. Sie können zwischen Fachverwaltung, kommunaler IT und einem Auftragnehmer verteilt werden. Für den Einstieg genügt häufig ein begrenzter Umfang: ein Anbieter, wenige Endpunkte, ein klarer Anwendungsfall und eine kleine Zahl stabil definierter Kennzahlen. Die so aufgebaute PostgreSQL/PostGIS-Datenbasis kann anschließend schrittweise erweitert werden.

Praxisbaustein: Abstellflächen

Die Kommune möchte feststellen, ob Fahrzeuge außerhalb ausgewiesener Abstellflächen abgestellt werden. Benötigt werden mindestens die Position und der Zeitpunkt des Fahrtendes, eine stabile Anbieter- und Fahrzeugzuordnung, versionierte Flächengeometrien sowie eine Regel für GPS-Ungenauigkeiten und Randpunkte. Erst danach lässt sich entscheiden, welcher API-Endpunkt geeignet ist.

3 Organisation, Recht und Datenquellen

3.1 Verantwortlichkeiten früh festlegen

Ein kommunales Dashboard berührt meist mehrere Organisationseinheiten. Dazu gehören Mobilitätsmanagement, Straßenverkehrsbehörde, Ordnungsamt, Datenschutz, Informationssicherheit, Geodatenmanagement und IT. Vor Entwicklungsbeginn sollten mindestens eine fachliche und eine technische Produktverantwortung benannt werden.

Die projektweiten Anforderungen unterscheiden dabei vereinfacht zwischen Kommune, Shared-Mobility-Anbietern, externen Daten- und Systempartnern sowie Plattformbetrieb und Administration. Diese Rollen helfen, Datenlieferung, fachliche Nutzung, technische Verantwortung und kontrollierte Weitergabe nicht miteinander zu vermischen [2].

Die fachliche Verantwortung legt Anwendungsfälle, Kennzahlen und Bewertungsregeln fest. Die technische Verantwortung organisiert Datenabrufe, Zugänge, Datenbank, Monitoring und Wiederanlauf. Die Bewertung eines vermeintlichen Regelverstoßes bleibt eine fachliche Aufgabe und darf nicht allein aus einer technischen Ampel abgeleitet werden.

3.2 Datenanforderungen verbindlich regeln

Eine Eigenentwicklung ist nur tragfähig, wenn der Zugang zu den erforderlichen Daten nicht von informellen Absprachen abhängt. Je nach örtlicher Rechtslage kommen Sondernutzungserlaubnisse, Nebenbestimmungen, Kooperationsvereinbarungen, Ausschreibungen oder separate Data Sharing Agreements in Betracht. Die GBFS-Datenrichtlinie [6] stellt hierfür beispielhafte Formulierungen für Ausschreibungen und Genehmigungsprozesse bereit.

Auch die im Projekt erhobenen Anforderungen betonen standardisierte Schnittstellen, klare Rahmenbedingungen, nachvollziehbare Regeln und eine qualitätsgesicherte, kontrollierte Datenbereitstellung. Für den Leitfaden folgt daraus: Anforderungen sollten nicht nur fachlich beschrieben, sondern als prüfbare Liefer- und Betriebsbedingungen formuliert werden [2].

Tabelle 3.1: Mindestinhalte einer Vereinbarung zur Datenbereitstellung

Regelungsbereich	Praktische Konkretisierung
Zweck	Kommunale Aufgaben und konkrete Anwendungsfälle benennen.
Standard und Version	GBFS, MDS oder anderes Format einschließlich Version und Migrationsverfahren festlegen.
Datenumfang	APIs, Endpunkte, Pflichtfelder, historische Tiefe und gegebenenfalls Telemetrie benennen.
Aktualität	Abruf- oder Lieferintervalle, maximale Verzögerung und Umgang mit Nachlieferungen definieren.
Authentifizierung	Verfahren, Token-Laufzeiten, Secret-Austausch, IP-Freigaben und Testzugang dokumentieren.
Qualität	Validierung, Fehlermeldung, Korrekturfristen und Eskalationswege vereinbaren.
Änderungsmanagement	Vorlaufzeiten für URL-, Schema-, Versions- und Authentifizierungsänderungen vorsehen.
Betriebsende	Datenverfügbarkeit, Dokumentation und Ansprechpartner nach Anbieterwechsel oder Marktaustritt regeln.

3.3 Datenminimierung und Datenschutz

MDS ist modular aufgebaut [7]. Eine Kommune kann APIs und Felder entsprechend ihren Zwecken auswählen, statt pauschal alle verfügbaren Daten zu verlangen. Dies reduziert Datenschutzrisiken, Speicherbedarf und technische Komplexität. Die *ScooP2City*-Anforderungen ergänzen diese Sicht um Rollen- und Rechtekonzepte, zweckgebundene Bereitstellung und Datensouveränität als durchgängige Anforderungen [2]. Mobilitätsdaten können trotz fehlender direkter Personenkennung sensibel oder in Kombination personenbeziehbar sein. Die Open Mobility Foundation stellt hierzu eine eigene Orientierung für die Nutzung von MDS unter der Datenschutz-Grundverordnung bereit [8].

Vor der Verarbeitung sollten daher insbesondere Rechtsgrundlage, Zweckbindung, Zugriffsschutz, Aufbewahrungsfrist, Aggregationsniveau und Löschung geklärt werden. Ein öffentliches Dashboard sollte grundsätzlich mit aggregierten Daten arbeiten und keine Einzelfahrten oder dauerhaft verfolgbaren Fahrzeughistorien darstellen.

Praxisbeispiel: Fahrtdaten auf das Erforderliche begrenzen

Ein Beispiel für Datenminimierung sind detaillierte Fahrtverläufe. In MDS 1.2 [9] enthielt der Endpunkt `/trips` neben Start- und Endzeit sowie weiteren Fahrtangaben eine GeoJSON-Route. Diese sollte grundsätzlich alle vom Anbieter erfassten GPS- oder GNSS-Messpunkte enthalten. Damit konnten wesentlich detailliertere Bewegungsverläufe geliefert werden, als sie für viele kommunale Dashboard-Auswertungen erforderlich sind.

Für zahlreiche Anwendungsfälle reichen Start- und Endpunkt einer Fahrt aus, etwa für Start-Ziel-Auswertungen, die Ermittlung räumlicher Schwerpunkte, die Bewertung von Abstellvorgängen oder aggregierte Analysen zwischen Teilräumen. Detaillierte Zwischenpunkte sollten daher nur angefordert und gespeichert werden, wenn ein klar benannter Zweck dies tatsächlich erfordert und die damit verbundenen Datenschutz- und Sicherheitsfragen geklärt sind.

MDS 2.0 [10] trennt diese Daten entsprechend stärker: Der Endpunkt `/trips` stellt Start- und Endposition bereit, während weitere Messpunkte einer Fahrt dem gesonderten Endpunkt `/telemetry` zugeordnet sind. Kommunen können dadurch den Datenumfang genauer am jeweiligen Anwendungsfall ausrichten und auf detaillierte Telemetrie verzichten, wenn Start- und Endpunkte genügen.

3.4 APIs und Datenquellen

Die Projektanforderungen verbinden standardisierte Schnittstellen mit der Anforderung, Anbieter-, kommunale und externe Daten gemeinsam auswertbar zu machen. Ein Dashboard sollte deshalb nicht als isolierter API-Viewer geplant werden, sondern als fachliche Sicht auf mehrere konsolidierte Datenquellen [2].

3.4.1 GBFS und MDS richtig einordnen

GBFS und MDS werden häufig gemeinsam genannt, erfüllen aber unterschiedliche Aufgaben. Zum Redaktionsstand dieses Leitfadens bezieht sich die offizielle GBFS-Referenz [11] auf Version 3.0. Sie beschreibt einen öffentlich zugänglichen Echtzeitstandard, dessen Schwerpunkt auf dem aktuellen Zustand eines Sharing-Systems und nicht auf historischen Analysen liegt.

Die Mobility Data Specification dient dagegen dem standardisierten Datenaustausch zwischen Mobilitätsanbietern und öffentlichen Stellen. Zum 16. Juli 2026 ist MDS 2.1.0 die aktuelle Veröffentlichung [7]. MDS umfasst unter anderem Provider-, Agency-, Policy-, Geography-, Jurisdiction- und Metrics-APIs und ist ausdrücklich modular nutzbar.

Tabelle 3.2: Rollen von GBFS, MDS und kommunalen Daten

Quelle	Stärke	Typische kommunale Nutzung
GBFS	Aktueller, öffentlicher Systemzustand	Fahrzeuge, Stationen, Fahrzeugtypen, Systeminformationen und Geofencing-Zonen.
MDS	Regulatorische und historische Daten; digitale Regeln	Fahrzeugstatus, Fahrten, Ereignisse, Telemetrie sowie Policy- und Geography-Daten.
Kommunale Geodaten	Verbindlicher räumlicher Kontext	Stadtgrenze, Bezirke, Abstellflächen, Verbotszonen, Haltestellen, Straßen und weitere Fachdaten.

Zu den für Dashboards häufig relevanten GBFS-3.0-Dateien gehören:

- `system_information.json`, `vehicle_types.json` und `vehicle_status.json`,
- `station_information.json` und `station_status.json`,
- `geofencing_zones.json`.

Der frühere Name `free_bike_status.json` wurde durch `vehicle_status.json` ersetzt [11].

Die MDS Provider API [12] unterscheidet unter anderem zwischen `/vehicles` für eher selten geänderte Fahrzeugeigenschaften und `/vehicles/status` für den aktuellen Status. Historische Fahrten werden über `/trips`, Statusereignisse über `/events/recent` und `/events/historical` bereitgestellt.

3.4.2 Endpunkte aus Anwendungsfällen ableiten

Tabelle 3.3: Beispielhafte Endpunktmatrix

Anwendungsfall	Mögliche Datenquelle
Aktuelle Zahl verfügbarer Fahrzeuge	GBFS `vehicle_status` oder MDS `/vehicles/status`
Fahrzeugtypen und Antriebe	GBFS `vehicle_types` oder MDS `/vehicles`
Stationen und aktuelle Auslastung	GBFS `station_information` und `station_status`
Historische Fahrten	MDS `/trips`
Statusänderungen	MDS `/events/recent` und `/events/historical`
Anbieter-Geofences	GBFS `geofencing_zones`
Kommunale Regeln und Gebiete	MDS Policy/Geography oder versioniertes kommunales GeoJSON
Datenverfügbarkeit	Metadaten der Feeds und eigenes technisches Monitoring

Ein Endpunkt allein definiert noch keine Kennzahl. Für eine Flottenzählung muss beispielsweise festgelegt werden, welche Zustände als im öffentlichen Raum befindlich gelten, wie alte Statusmeldungen behandelt werden und welche Gebietsgrenze maßgeblich ist. MDS weist ausdrücklich darauf hin, dass bei Fahrzeugzählungen nicht alle gelieferten Zustände ungefiltert addiert werden sollten [12].

3.4.3 Login und Authentifizierung

In der Praxis unterscheiden sich Anbieter trotz Standards erheblich. Neben öffentlichen GBFS-Feeds ohne Login treten statische Bearer-Tokens, tokenbasierte Client-Credentials-Verfahren, anbieterspezifische Token-Endpunkte und kurzlebige, signierte JWTs auf. Hinzu kommen unterschiedliche Content-Types, Scopes, Schlüsselkennungen, Issuer- und Audience-Angaben, Accept-Header, Versionsangaben, IP-Freischaltungen und Rate Limits.

Die CMD-interne Konfiguration der Anbieterzugänge [13] zeigt, dass die Authentifizierung selbst bei gleicher fachlicher MDS-Nutzung nicht einheitlich umgesetzt ist. In einem Fall wird ein dauerhaft hinterlegter Token verwendet, in anderen Fällen muss ein Zugangstoken über einen separaten Endpunkt angefordert werden; wiederum andere Zugänge erfordern ein sehr kurzlebiges, mit einem privaten Schlüssel signiertes Token. Auch die erwarteten MDS-Versionen und Header unterscheiden sich.

Tabelle 3.4: Typische Authentifizierungsvarianten in der Anbieteranbindung

Variante	Typische Besonderheiten	Kommunale Konsequenz
Statischer Zugangstoken	Einfacher Bearer-Token ohne eigenen Tokenabruf; Austausch und Ablauf sind anbieterspezifisch.	Sichere Ablage, dokumentierte Rotation und eindeutige Zuständigkeit vorsehen.
Token-Endpunkt	Zugangstoken wird mit Client-ID und Secret angefordert; Request-Body, Content-Type, Scope und Antwortformat können abweichen.	Tokenabruf und Erneuerung in einer zentralen Zugriffsschicht kapseln.
Signiertes Kurzzeit-Token	Token wird lokal mit privatem Schlüssel erzeugt; Schlüssel-ID, Issuer, Audience und sehr kurze Laufzeit sind zu beachten.	Schlüsselverwaltung, korrekte Systemzeit und automatisierte Erneuerung sicherstellen.
Zusätzliche Vorgaben	Unterschiedliche Basis-URLs, Accept-Header, Versionen, IP-Freigaben oder Mengenbegrenzungen.	Anbieterbezogene Konfiguration getrennt von Datenmodell, Auswertung und Visualisierung halten.

Die allgemeinen MDS-Hinweise [14] sehen für Provider-, Agency- und Metrics-Schnittstellen eine Authentifizierung vor; Policy-, Geography- und Jurisdiction-Endpunkte sollen dagegen öffentlich und ohne Authentifizierung erreichbar sein. Die konkrete Authentifizierung bleibt dennoch ein wesentlicher Teil des Anbieter-Onboardings. Zugangsdaten und Tokenlogik sollten nicht in Datenbankauswertungen, Kennzahlen oder Dashboards eingebaut werden, sondern in einer zentralen Zugriffsschicht gekapselt sein.

4 Onboarding, Datenqualität, Geodaten und Policies

4.1 Anbieter einheitlich anbinden

Empfehlenswert ist eine anbieterunabhängige Abruf- und Konfigurationsschicht. Pro Anbieter werden Basis-URL, Version, Endpunkte, Authentifizierung, Abrufintervalle, Zeitüberschreitungen, Mengenbegrenzungen und bekannte Abweichungen hinterlegt. Diese Schicht übernimmt Anmeldung, Erneuerung von Zugangstokens, Verarbeitung mehrseitiger Antworten, Wiederholungsversuche, Rohdatenspeicherung und Fehlerprotokollierung. Welche Software dafür eingesetzt wird, bleibt eine kommunale Umsetzungsentscheidung.

Authentifizierung und Datenabruf sollten dabei getrennt behandelt werden: Die Zugriffsschicht stellt einen gültigen, für den jeweiligen Anbieter passenden Zugang her; erst danach beginnt die fachliche Verarbeitung der Antwort. Dadurch lassen sich geänderte Token-Endpunkte, neue Schlüssel, andere Header oder Versionen austauschen, ohne Datenmodell und Auswertungen umzubauen.

Vor der Freischaltung sollte ein technisches Abnahmeprotokoll erstellt werden. Es prüft mindestens Erreichbarkeit, Version, Authentifizierung, mehrseitige Antworten, Zeitstempel, ID-Stabilität, leere Zeiträume, Testdaten, Aktualität und Ansprechpartner. Zusätzlich sollte ein stichprobenartiger Abgleich mit weiteren Oberflächen des Anbieters erfolgen, sofern beispielsweise ein Anbieter-Backend oder eine App-Schnittstelle verfügbar ist.

4.2 Rohdaten als nachvollziehbare Grundlage

API-Daten sollten nicht ausschließlich in bereits aufbereiteter Form gespeichert werden. Die unveränderte Antwort bildet zusammen mit den wichtigsten Abrufinformationen die Rohdatenebene. Sie ist weder ein vollständiges technisches Protokoll aller Einzelschritte noch nur ein kurzfristiger Zwischenspeicher. Ihr Zweck ist, fachliche Abfragen nachvollziehbar abzuschließen, Datenlücken zu erklären und eine spätere Neuverarbeitung zu ermöglichen.

Als einfacher Grundsatz gilt: Eine gespeicherte Zeile beschreibt eine fachliche Abfrage. Bei einem aktuellen Fahrzeugstatus ist dies beispielsweise der Anbieter zusammen mit dem Zeitpunkt der Momentaufnahme. Bei stundenweise abgefragten Fahrten, Telemetrie- oder Ereignisdaten ist es der Anbieter zusammen mit der angefragten UTC-Stunde. Dadurch bleibt erkennbar, auf welchen fachlichen Zeitraum sich eine Antwort bezieht – unabhängig davon, wann der technische Abruf tatsächlich stattgefunden hat.

Zu einer solchen Abfrage sollten mindestens gespeichert werden:

- Anbieter, Endpunkt und verwendete API-Version,
- fachlicher Zeitpunkt oder angefragter Zeitraum in UTC,
- Zeitpunkt des ersten und letzten Abrufversuchs,
- HTTP-Status und Zeitpunkt eines erfolgreichen Abschlusses,
- unveränderte Antwort und ausgewählte Antwort-Header,
- Fehlerzeitpunkt, verständliche Fehlermeldung und Zahl der Versuche.

Für die Interpretation sind drei Zustände besonders hilfreich: Fehlt eine Zeile, wurde die fachliche Abfrage noch nie versucht. Liegt eine Zeile ohne erfolgreichen Abschluss vor, wurde sie versucht, aber noch nicht

verwertbar beantwortet. Eine erfolgreiche Antwort gilt auch dann als abgeschlossen, wenn sie für den angefragten Zeitraum keine fachlichen Datensätze enthält. Eine Stunde ohne Fahrten ist daher nicht automatisch ein Fehler.

Bereits erfolgreich gespeicherte Antworten sollten durch spätere technische Fehler nicht überschrieben werden. Aus den erfolgreichen Rohantworten lassen sich aufbereitete Tabellen bei Bedarf neu erzeugen, ohne die Anbieter-API erneut abzufragen. Dies ist besonders wichtig, wenn Berechnungsregeln, Datenmodelle oder räumliche Zuschnitte später geändert werden. Die Aufbewahrung der Rohdaten ist mit Datenschutz und Löschkonzept abzustimmen.

Diese Trennung unterstützt zugleich mehrere projektweit erhobene Anforderungen: Datenqualität muss prüfbar sein, Datenlücken sollen erklärbar bleiben und aufbereitete Informationen sollen kontrolliert für unterschiedliche Rollen bereitgestellt werden können [2].

4.3 Typische Fehlerklassen

Formale Schema-Konformität ist notwendig, aber nicht hinreichend. Für GBFS steht ein kanonischer Validator [15] zur Verfügung; die offizielle Dokumentation zur Datenqualität [16] empfiehlt zusätzlich laufende Aktualitäts- und Qualitätsprüfungen.

Tabelle 4.1: Typische Fehlerbilder und geeignete Kontrollen

Klasse	Beispiele	Kontrolle
Syntax und Metadaten	Ungültiges JSON, fehlende Pflichtfelder oder Metadaten, falsche Datentypen, HTTP 200 mit Fehlertext	Schema-Validierung, Content-Type-, Status- und Pflichtfeldprüfung
Zeit	Sekunden statt Millisekunden, lokale Zeit als UTC, Zukunftswerte, veraltete Feeds	Altersprüfung, Zeitreihenlücken, Reihenfolge von Ereignissen
Geometrie	Vertauschte Koordinaten, Positionen außerhalb des erwarteten Gebiets, falsches Bezugssystem, ungültige Polygone	Gebietsprüfung, Geometrievalidierung, Ausreißerregeln
Semantik	Widersprüchliche Zustände, unplausible Fahrzeugzahlen, Fahrtende vor Beginn, wechselnde IDs	Fachliche Plausibilitätsregeln und Quervergleich zwischen Endpunkten
Systemabgleich	Unterschiedliche Fahrzeugbestände oder Zustände in Anbieter-Backend, MDS und App-Schnittstelle	Stichproben, Abstimmberichte und definierte fachliche Referenzquelle je Kennzahl
Betrieb	Abgelaufene Tokens, unangekündigte URL-Änderung, Rate Limits, Teilantworten	Monitoring, Alarmierung, Regeln für Wiederholungsversuche und Abnahmeprozess
Pagination	Nicht endende Folgelinks, Duplikate, Lücken an Zeitgrenzen, nachträgliche Ergänzungen	Abbruchgrenzen, wiederholbare Importe, eindeutige Schlüssel und kontrollierte Neuimporte

Praxisbeobachtungen aus der Anbieteranbindung

Die CMD-interne Problemnachverfolgung [17] dokumentiert mehrere Fehler, die bei einer reinen Erreichbarkeitsprüfung unentdeckt geblieben wären: Eine leere Ergebnisseite enthielt weiterhin einen Folgelink und konnte dadurch eine Endlosschleife auslösen; erwartete Metadaten fehlten; Zahlen wurden mit einem anderen Datentyp als in der Spezifikation geliefert; Längen- und Breitengrad waren vertauscht; und Fahrzeugzahlen sowie Fahrzeugzustände widersprachen sich zwischen MDS-Endpunkten.

Darüber hinaus wurden im Projekt Abweichungen zwischen verschiedenen Systemebenen desselben Anbieters beobachtet: Anbieter-Backend, MDS-Schnittstelle und die von der App genutzte Schnittstelle zeigten teilweise unterschiedliche Fahrzeugbestände oder Zustände. Vergleichbare Muster traten bei mehreren Anbietern auf und sollten deshalb nicht als Einzelfall behandelt werden.

Daraus folgt eine wichtige Prüfregel: Datenqualität darf nicht nur innerhalb eines Endpunkts bewertet werden. Für zentrale Kennzahlen sollten auch verwandte Endpunkte und – soweit zugänglich – weitere Anbieteroberflächen stichprobenartig verglichen werden. Treten Unterschiede auf, braucht die Kommune eine dokumentierte Entscheidung, welche Quelle für welchen Anwendungsfall als fachlich maßgeblich gilt.

4.4 Qualitätsmonitoring als eigene Auswertung

Sinnvolle Betriebskennzahlen sind der letzte erfolgreiche Abruf, das Alter des jüngsten Datensatzes, Datensätze pro Anbieter und Stunde, Fehlerrate, fehlende Pflichtfelder, räumliche Ausreißer, nicht zuordenbare Fahrzeugtypen, Laufzeiten und Authentifizierungsfehler. Ergänzend sollten Abweichungen zwischen Endpunkten und Systemebenen sichtbar werden, beispielsweise durch den Vergleich von Fahrzeugzahlen, Zustandsverteilungen und Zeitstempeln. Fachliche Auswertungen sollten kennzeichnen, wenn Daten fehlen, veraltet oder zwischen Quellen widersprüchlich sind.

Grundsatz: Ein erreichbarer Endpunkt ist nicht automatisch eine belastbare Datenquelle. Verfügbarkeit, Aktualität, Vollständigkeit, Konsistenz zwischen Schnittstellen und fachliche Plausibilität müssen getrennt überwacht werden.

4.5 Kommunale Geodaten und digitale Policies

4.5.1 Zusatzdaten als fachlicher Kontext

Mobilitätsdaten erhalten ihren kommunalen Wert häufig erst durch die Verbindung mit weiteren Daten. Typische Zusatzdaten sind Stadt- und Bezirksgrenzen, Abstell- und Verbotszonen, Fußgängerzonen, Grünanlagen, Gewässer, Brücken, Haltestellen, Mobilstationen, Radverkehrsnetze, Baustellen oder Veranstaltungsflächen. Die im Projekt erhobenen Anforderungen nennen darüber hinaus unter anderem Bevölkerungs-, ÖPNV-, Verkehrs-, Umwelt- und weitere Open-Data-Bestände als mögliche Grundlagen für Planung und Bewertung. Entscheidend ist nicht die Vollständigkeit einer langen Quellenliste, sondern der nachweisbare Bezug zum jeweiligen Anwendungsfall [2].

Für jeden Datensatz sollten Quelle, Lizenz, fachliche Verantwortung, Aktualisierungsintervall, Koordinatenreferenzsystem, Genauigkeit, Gültigkeitszeitraum und letzte Aktualisierung dokumentiert werden. Besonders wichtig ist eine eindeutige digitale Stadtgrenze. Die MDS Provider-Spezifikation empfiehlt, diese als online zugängliches Polygon beziehungsweise Multipolygon in GeoJSON oder Shapefile bereitzustellen [12].

4.5.2 GeoJSON und uMap als kommunales Redaktionswerkzeug

uMap kann Fachstellen als niedrigschwelliges Redaktionswerkzeug für die Erstellung und Pflege von Karten und Geometrien dienen. Die technische uMap-Dokumentation [18] beschreibt GeoJSON als zentrales Speicherformat für Layerdaten. Auch ein dauerhaft produktiver Einsatz ist möglich, wenn Zuständigkeiten, Attribute, Veröffentlichung und technische Übernahme verbindlich geregelt sind. Ein Export sollte dabei nicht ungeprüft direkt in Auswertungen einfließen.

Für kommunale und andere nicht-kommerzielle beziehungsweise gemeinwohlorientierte Anwendungen kann die vom FOSSGIS e. V. betriebene Instanz `https://umap.openstreetmap.de` einen kostenfreien Einstieg bieten. Sie ermöglicht die browserbasierte Pflege von Punkten, Linien und Polygonen sowie den Import und Export geostrukturierter Daten [19]. Da der Dienst ohne garantierte Verfügbarkeit betrieben

wird und die Nutzungsbedingungen unter anderem hohe Zugriffszahlen und massenhafte Downloads begrenzen, sollte die dauerhaft maßgebliche Datenhaltung weiterhin in PostgreSQL/PostGIS erfolgen oder regelmäßig dorthin übernommen werden [20].

Ein robuster Ablauf besteht aus:

1. Geometrie in uMap oder einem GIS erstellen oder ändern,
2. GeoJSON exportieren,
3. Geometrien und Attribute automatisiert validieren,
4. eindeutige ID, Version und Gültigkeitszeitraum vergeben,
5. Datei unter einer stabilen URL veröffentlichen,
6. in PostGIS übernehmen und Änderungen protokollieren,
7. relevante Anbieter über neue Versionen informieren.

Wichtige Attribute sind Zonen-ID, Name, Regeltyp, betroffene Fahrzeugarten, Gültig-von und Gültig-bis, Priorität, Park- oder Fahrregel, fachliche Kontaktstelle und Versionsstand. Bei überlappenden Zonen muss die Vorrangregel eindeutig sein. Projektweit wurden zudem die zeitliche Gültigkeit, die nachvollziehbare Historie, optional anbieterbezogene Regeln und die Verarbeitung von Geometrien thematisiert, die kommunale Grenzen schneiden. Für einen allgemeinen Leitfaden lässt sich dies zu einem Grundsatz verdichten: Regeln und Geometrien benötigen eindeutige Identitäten, Versionen, Zeitbezüge und einen dokumentierten Geltungsbereich [2].

4.5.3 MDS Policy und Geography

Für eine standardisierte Veröffentlichung können MDS Policy und Geography [7] eingesetzt werden. Die Policy API dient dazu, lokale Regeln, Ereignisse oder Notlagen digital zu beschreiben; die Geography API liefert die zugehörigen Gebiete. Ein versioniertes GeoJSON kann trotzdem ein sinnvoller Zwischenschritt sein, sofern eine eindeutige Quelle und ein kontrollierter Pflegeprozess bestehen.

Ansatz für Nordrhein-Westfalen: MOBIDROM Zonenmanager

Kommunen in Nordrhein-Westfalen steht mit dem MOBIDROM Zonenmanager ein kostenfreier und neutraler Webdienst zur Festlegung, Abstimmung und Verwaltung von Gebots- und Verbotszonen zur Verfügung. Nach Angaben des Betreibers können unter anderem Bediengebiete, Fahrverbots-, Parkverbots- und Gebotszonen sowie zulässige Fahrzeugzahlen gepflegt, bestehende Datensätze importiert und Vorgaben mit Sharing-Anbietern abgestimmt werden. Die Informationen werden zentral aktualisiert und in maschinenlesbarer Form bereitgestellt [21].

Der Dienst ist damit insbesondere für nordrhein-westfälische Kommunen ein naheliegender Einstieg in eine strukturierte Zonenpflege. Für die Nutzung in einem kommunalen Dashboard sollten dennoch Verantwortlichkeiten, Versionsstände, Export- oder Schnittstellenwege und die Übernahme in PostgreSQL/PostGIS verbindlich festgelegt werden. Der Zonenmanager ersetzt den zentralen Datenkern nicht, kann aber die fachliche Erfassung und Abstimmung von Regeln und Geometrien wesentlich vereinfachen.

Praxisbeispiel: Pflege kommunaler Zonen in Düsseldorf

In Düsseldorf werden Sharingstationen und Parkverbotszonen seit mehreren Jahren in einer uMap gepflegt. Neben den Geometrien werden fachliche Attribute wie Name, Kapazität und weitere für die jeweilige Zone benötigte Merkmale hinterlegt. Die Kartenadresse wird gezielt an die beteiligten Mobilitätsanbieter weitergegeben. Diese können die benötigten Layer dort als GeoJSON abrufen.

Wenn sich Zonen oder Attribute ändern, informiert die Kommune die Anbieter zusätzlich per E-Mail. Dadurch wird nicht allein auf einen automatischen Abruf vertraut, sondern eine fachliche Änderungsmitteilung mit einem klaren Kommunikationsweg verbunden. Die Geodaten sind über eine proprietäre

Schnittstelle mit PostGIS gekoppelt. So können die gepflegten Geometrien in die zentrale Datenhaltung übernommen und für räumliche Auswertungen genutzt werden. Dieses Verfahren hat sich im laufenden Betrieb über mehrere Jahre bewährt.

Das Beispiel zeigt, dass kein komplexes Fachverfahren erforderlich sein muss, wenn Redaktion, Datenformat, Verteilung und technische Übernahme sauber zusammenspielen. Zu beachten ist jedoch, dass eine nur gezielt weitergegebene URL keine technische Zugriffskontrolle ersetzt. Sobald Inhalte schutzbedürftig sind oder differenzierte Berechtigungen benötigt werden, sollte ein geeignetes Zugriffs- und Rollenkonzept ergänzt werden.

5 Datenarchitektur, Visualisierung und Betrieb

5.1 Datenfluss einer kommunalen Eigenentwicklung

Eine robuste Architektur trennt Datenabruf, Rohdaten, fachliche Aufbereitung und Darstellung. Dadurch bleiben Anbieterbesonderheiten nachvollziehbar und Auswertungen greifen nicht direkt auf wechselnde API-Strukturen zu.

Die *ScooP2City*-Anforderungen ordnen die fachlichen Bedarfe in Datenintegration, Analyse und Planung, Visualisierung und Reporting sowie Informations- und Wissensangebote ein. Für diesen Leitfaden wird daraus keine vollständige Plattformarchitektur übernommen. Relevant ist lediglich die Trennung eines gemeinsamen Datenkerns von den darauf aufbauenden fachlichen Sichten und Diensten [2].

PostgreSQL mit PostGIS wird in diesem Leitfaden als feste zentrale Datenbasis vorausgesetzt. Die Auswahl anderer Datenbankplattformen ist nicht Gegenstand des Leitfadens. In derselben Plattform können technische Abrufinformationen, unveränderte Rohantworten, aufbereitete Fachdaten, kommunale Geometrien, Qualitätsmeldungen und vorbereitete Auswertungen nachvollziehbar getrennt werden.

Abbildung 5.1: Vereinfachter Datenfluss einer kommunalen Eigenentwicklung.

Die Rohdatenebene bewahrt Originalantworten und Abrufinformationen. Die Fachebene bildet Anbieterinformationen in ein gemeinsames kommunales Modell ab. Eine Auswertungsebene stellt vorbereitete Sichten oder Tabellen für wiederkehrende Kennzahlen bereit.

5.2 PostgreSQL und PostGIS als zentrale Datenbasis

PostgreSQL verwaltet die relationalen und zeitlichen Daten; PostGIS ergänzt die notwendigen räumlichen Funktionen. Dadurch können Fahrzeugzustände, Fahrten, Ereignisse und kommunale Geometrien in einer gemeinsamen Datenbasis gespeichert, geprüft und ausgewertet werden.

Ein einfaches fachliches Modell umfasst Anbieter, Fahrzeuge, Fahrzeugtypen, Zustände, Fahrten, Ereignisse, Stationen, kommunale Gebiete, Policy-Zonen, Importläufe und Qualitätsmeldungen. Geometrien sollten validiert, in geeigneten Bezugssystemen verarbeitet und mit räumlichen Indizes versehen werden. Solche Indizes sind entscheidend, damit räumliche Abfragen auch bei wachsenden Datenmengen schnell bleiben [22].

Typische Verschneidungen sind die Zuordnung eines Fahrzeugs zu Stadtteil und Zone, die Prüfung eines Fahrtendes gegen Abstellflächen sowie Entfernungsanalysen zu Haltestellen. Randfälle müssen fachlich definiert werden. Dazu gehört zum Beispiel die Frage, wie ein GPS-Punkt unmittelbar auf dem Rand einer Abstellfläche behandelt wird. PostGIS stellt dafür unterschiedliche räumliche Prüfungen bereit [23].

5.3 Visualisierung nach Zielgruppen strukturieren

Für die Darstellung kommen unterschiedliche Dashboard-, Karten- und Berichtswerkzeuge in Betracht. Grafana ist ein mögliches Beispiel für Zeitreihen, Kennzahlen, Tabellen, einfache Karten und Warn-

meldungen, aber keine Voraussetzung für die Architektur. Die Projektanforderungen betonen einen verständlichen, rollenbezogenen und möglichst wiederverwendbaren Zugang zu Karten, Kennzahlen, Tabellen und Berichten, ohne für jede fachliche Frage eine technische Einzellösung aufzubauen [2].

Empfehlenswert sind getrennte Sichten:

- **Betriebssicht:** Abrufe, Token-Probleme, Verzögerungen, Datenmengen und Fehler.
- **Fachsicht:** Flotten, Fahrten, Raumverteilung und Policy-Auswertungen.
- **Managementsicht:** wenige stabile Kennzahlen mit klarer Definition.
- **Öffentliche Darstellung:** aggregierte, datenschutzgeprüfte und erläuterte Ergebnisse.

Die Wahl des Darstellungswerkzeugs sollte sich nach Zielgruppe und Aufgabe richten. Einfache Kennzahlen, räumliche Einzelfallprüfungen, interaktive Karten und veröffentlichungsfähige Berichte können unterschiedliche Oberflächen erfordern, greifen aber auf dieselbe geprüfte Datenbasis zu.

5.4 Betrieb, schrittweises Vorgehen und Fazit

5.4.1 Dauerbetrieb von Beginn an mitdenken

Ein Dashboard ist kein einmaliges Softwareprojekt, sondern ein dauerhaftes kommunales Datenprodukt. Standards und Anbieter-APIs verändern sich, Tokens laufen ab, Geometrien werden aktualisiert und Datenfehler müssen bewertet werden. Die *ScooP2City*-Anforderungen benennen kontinuierlichen Zugang, Rechte- und Rollenverwaltung, Qualitätssicherung, Konfiguration von Datenquellen und kontrollierte Datenweitergabe als betriebliche Daueraufgaben [2].

Für den Betrieb sind mindestens folgende Prozesse erforderlich:

- Monitoring und Alarmierung aller Datenabrufe,
- sichere Verwaltung und Erneuerung von Zugangsdaten,
- regelmäßige Qualitätsprüfung und Eskalation an Anbieter,
- versionierte Pflege kommunaler Geodaten und Policies,
- Backup, Wiederherstellung und dokumentierter Wiederanlauf,
- Rollen- und Berechtigungskonzept,
- Löschung nach festgelegten Fristen,
- Testverfahren für neue Standard- und API-Versionen,
- Vertretungsregelungen und aktuelle Betriebsdokumentation.

5.4.2 Schrittweise statt vollständig starten

1. **Aktuelle Übersicht:** einen Anbieter und GBFS anbinden, Stadtgrenze verschneiden, Monitoring einrichten.
2. **Historie:** Snapshots, MDS-Fahrten oder Ereignisse integrieren und Zeitreihen aufbauen.
3. **Policies:** Abstell- und Verbotszonen versioniert bereitstellen und räumlich prüfen.
4. **Mehrere Anbieter:** Adapter und gemeinsames Datenmodell stabilisieren; Qualitätsunterschiede transparent machen.
5. **Dauerbetrieb:** Verantwortlichkeiten, Datenschutz, Finanzierung und öffentliche Kommunikation verstetigen.

5.4.3 Kompakte kommunale Checkliste

Tabelle 5.1: Kompakte kommunale Checkliste

Prüfbereich	Leitfragen
Zielbild	Sind Anwendungsfälle, Entscheidungen, Zielgruppen und Kennzahlendefinitionen dokumentiert?
Verantwortung	Gibt es fachliche und technische Produktverantwortung sowie Vertretungen?
Datenvereinbarung	Sind Versionen, Endpunkte, Aktualität, Testzugang, Qualität und Änderungsmanagement geregelt?
Datenschutz	Sind Zweck, Rechtsgrundlage, Zugriff, Speicherung, Aggregation und Löschung geprüft?
Technische Anbindung	Werden Rohdaten gespeichert, Pagination und Neuimporte beherrscht und Zugangsdaten sicher verwaltet?
Qualität	Werden Schema, Aktualität, Vollständigkeit, Geometrie und Semantik getrennt geprüft?
Geodaten	Existieren eindeutige, versionierte Grenzen und Policy-Zonen mit Zuständigkeiten?
Darstellung	Sind Betriebs-, Fach-, Management- und öffentliche Sichten getrennt?
Betrieb	Sind Monitoring, Backup, Wiederanlauf, Updates und dauerhafte Ressourcen gesichert?

5.4.4 Fazit

Ein kommunales Micromobility-Dashboard besteht nicht nur aus einer Karte und Diagrammen. Die entscheidenden Grundlagen sind eine verbindliche und zweckgebundene Datenbereitstellung, robuste Anbieteradapter, systematische Qualitätsprüfungen, versionierte kommunale Geodaten und ein organisierter Dauerbetrieb.

Fünf Empfehlungen fassen den Leitfaden zusammen:

1. Anwendungsfälle vor Datenanforderungen definieren.
2. Datenlieferungen technisch präzise und rechtlich belastbar vereinbaren.
3. Standards nutzen, aber Anbieterabweichungen und Versionswechsel einplanen.
4. Rohdaten, Qualität und Geometrien nachvollziehbar versionieren.
5. Betrieb, Datenschutz und Verantwortlichkeiten von Anfang an mitdenken.

Das Ziel ist nicht die größtmögliche Datensammlung, sondern ein wartbares, transparentes und fachlich belastbares Datenprodukt, das kommunale Entscheidungen unterstützt. Wer APIs automatisiert abfragen, Daten in PostgreSQL/PostGIS strukturiert verarbeiten und Ergebnisse mit einem Self-Service-BI-Werkzeug nachvollziehbar aufbereiten kann, verfügt bereits über die wesentlichen technischen Grundlagen. Fehlende Kompetenzen können gezielt beauftragt werden, ohne die fachliche Produktverantwortung aus der Hand zu geben.

Literaturverzeichnis

[1] ScooP2City. Innovatives Service-Ökosystem für stadtverträgliche Shared Mobility, 2026. URL https://scoop2.city/. Zugriff am 10. Juli 2026.

[2] Holger Friedel, David Rüdiger, and Andreas Schmidt. ScooP2City Anforderungen: Grundlage für die Entwicklung einer urbanen Datenplattform. Arbeitsergebnis AP 3 im Projekt ScooP2City Förderkennzeichen 01F2261D, Connected Mobility Düsseldorf GmbH, 2026.

[3] PostgreSQL Global Development Group. About PostgreSQL, 2026. URL https://www.postgresql.org/about/. Zugriff am 16. Juli 2026.

[4] PostgreSQL Global Development Group. JSON Types, 2026. URL https://www.postgresql.org/docs/current/datatype-json.html. PostgreSQL-Dokumentation, Zugriff am 16. Juli 2026.

[5] PostGIS Project. About PostGIS, 2026. URL https://postgis.net/. Zugriff am 16. Juli 2026.

[6] GBFS Community. GBFS Data Policy, 2026. URL https://gbfs.org/documentation/data-policy/. Zugriff am 10. Juli 2026.

[7] Open Mobility Foundation. Mobility Data Specification, 2026. URL https://github.com/openmobilityfoundation/mobility-data-specification. Version 2.1.0 zum Abrufdatum; Zugriff am 10. Juli 2026.

[8] Open Mobility Foundation. Using MDS under GDPR, 2026. URL https://www.openmobilityfoundation.org/using-mds-under-gdpr/. Zugriff am 10. Juli 2026.

[9] Open Mobility Foundation. Mobility Data Specification 1.2.0: Provider API, Abschnitt Trips und Routes, 2021. URL https://github.com/openmobilityfoundation/mobility-data-specification/blob/1.2.0/provider/README.md. Zugriff am 10. Juli 2026.

[10] Open Mobility Foundation. Mobility Data Specification 2.0.0: Release Notes, 2023. URL https://github.com/openmobilityfoundation/mobility-data-specification/releases/tag/2.0.0. Trennung von Fahrtmetadaten und Telemetrie; Zugriff am 10. Juli 2026.

[11] GBFS Community. General Bikeshare Feed Specification: Current Version, 2026. URL https://gbfs.org/documentation/reference/. Referenzversion 3.0; Zugriff am 10. Juli 2026.

[12] Open Mobility Foundation. Mobility Data Specification: Provider API, 2026. URL https://github.com/openmobilityfoundation/mobility-data-specification/blob/main/provider/README.md. Zugriff am 10. Juli 2026.

[13] Connected Mobility Düsseldorf GmbH. CMD-interne Konfiguration zur Authentifizierung von MDS-Anbieterzugängen. Unveröffentlichte technische Arbeitsunterlage, Stand Juli 2026, 2026.

[14] Open Mobility Foundation. Mobility Data Specification: General Information, 2026. URL https://github.com/openmobilityfoundation/mobility-data-specification/blob/main/general-information.md. Zugriff am 10. Juli 2026.

[15] GBFS Community. GBFS Tools and Validators, 2026. URL https://gbfs.org/tools/. Zugriff am 10. Juli 2026.

[16] GBFS Community. GBFS Data Quality, 2026. URL https://gbfs.org/documentation/data-quality/. Zugriff am 10. Juli 2026.

[17] Connected Mobility Düsseldorf GmbH. Nachverfolgung von Problemen: MDS- und Datenabweichungen in der Anbieteranbindung. CMD-interner Issue-Tracker, Stand 7. Juli 2026, unveröffentlicht, 2026.

[18] uMap Project. uMap Documentation, 2026. URL https://docs.umap-project.org/en/stable/. Zugriff am 10. Juli 2026.

[19] FOSSGIS e. V. uMap – Online Map Creator, 2026. URL https://umap.openstreetmap.de/. Zugriff am 16. Juli 2026.

[20] FOSSGIS e. V. Nutzungsbedingungen OSM-Server des FOSSGIS e. V., 2026. URL https://www.fossgis.de/arbeitsgruppen/osm-server/nutzungsbedingungen/. Zugriff am 16. Juli 2026.

[21] NRW.Mobidrom GmbH. MOBIDROM Zonenmanager: Shared Mobility einfacher steuern, 2026. URL https://www.mobidrom.nrw/produkte/mobidrom-zonenmanager. Zugriff am 16. Juli 2026.

[22] PostGIS Project. Spatial Queries and Spatial Indexes, 2026. URL https://postgis.net/docs/manual-dev/using_postgis_query.html. Zugriff am 10. Juli 2026.

[23] PostGIS Project. ST_Within, 2026. URL https://postgis.net/docs/ST_Within.html. Zugriff am 10. Juli 2026.