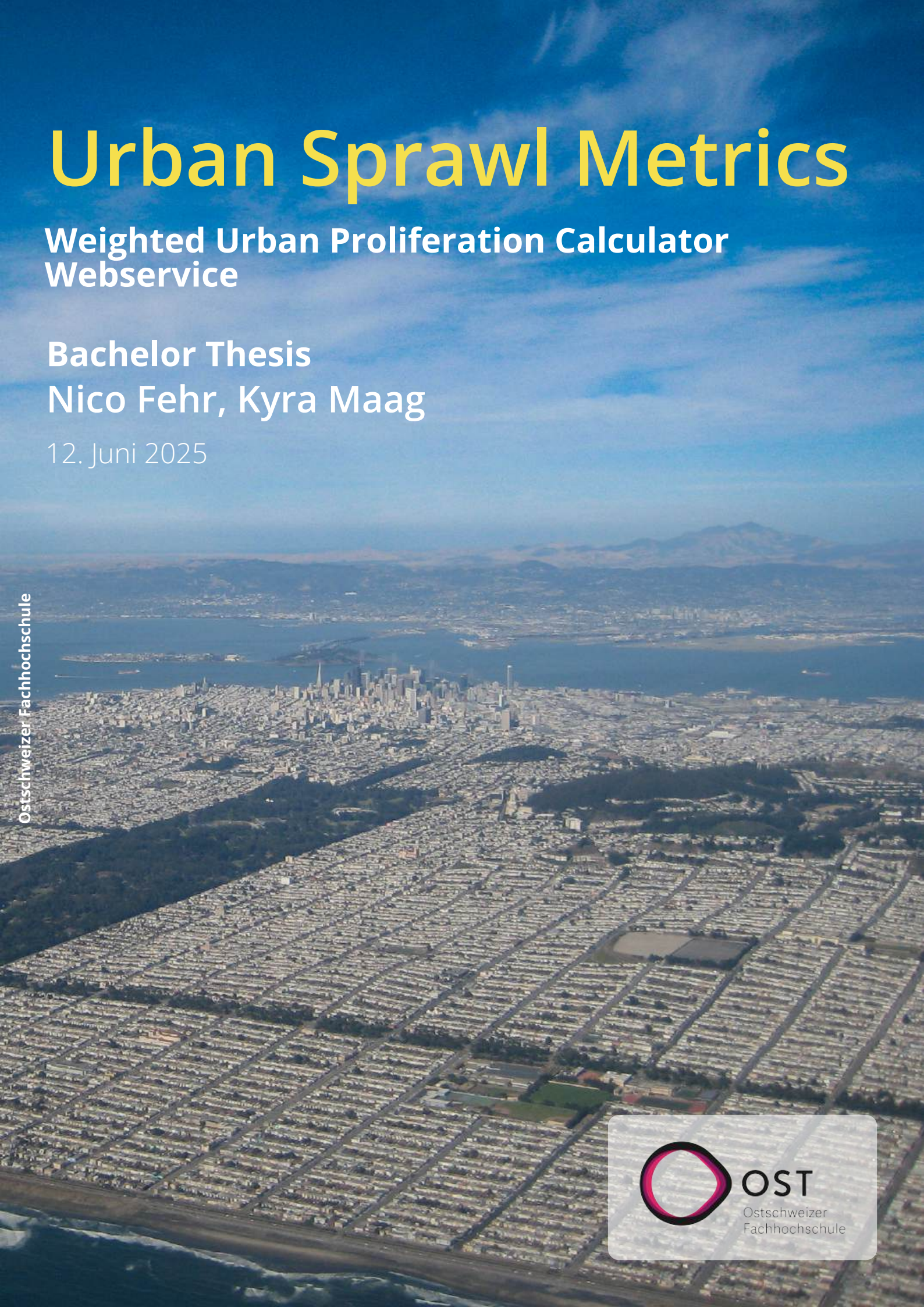
Urban Sprawl Metrics

Weighted Urban Proliferation Calculator Webservice

Bachelor Thesis

Nico Fehr, Kyra Maag

12. Juni 2025



Ostschweizer Fachhochschule

OST
Ostschweizer
Fachhochschule

Urban Sprawl Metrics

Weighted Urban Proliferation Calculator Webservice

von

Nico Fehr, Kyra Maag

Name	E-Mail
Nico Fehr	nico.fehr@ost.ch
Kyra Maag	kyra.maag@ost.ch

Betreuer:	Prof. Stefan Keller, Joël Schwab, Institut für Software (IFS) OST
Externer Partner:	Yves Maurer Weisbrod, Bundesamt für Raumentwicklung (ARE), Bern
Projektdauer:	Februar, 2025 - Juni, 2025
Fakultät:	Departement Informatik, Ostschweizer Fachhochschule OST

Eine elektronische Version dieser Arbeit ist verfügbar unter `https://eprints.ost.ch/`.

OST
Ostschweizer
Fachhochschule

Abstract

Die Zersiedelung beschreibt die unkontrollierte Ausbreitung der Siedlungsfläche auf Kosten von Natur- und Landwirtschaftsflächen[1]. Das Bundesamt für Raumentwicklung (ARE)[2] hat den Auftrag des Bundes, die Zersiedelung in der Schweiz zu beobachten und diese einzugrenzen. Politisch bezeugte besonders die gescheiterte "Zersiedelungsinitiative"[3] vom 10. Februar 2019 die Dringlichkeit der Thematik. Die Zersiedelung ist demnach nicht nur ein geographisches, sondern auch ein gesellschaftliches Thema. Mit der von Schwick et al. im Buch "Zersiedelung messen und begrenzen"[4] begründeten Methoden zur Bestimmung der Zersiedelung wurde ein Fundament zur Erfassung dieses Problems geschaffen.

Das Projekt zielt darauf ab, den "USM Calculator" Webservice[5], eine Webapplikation, basierend auf den erwähnten Methoden, um zusätzliche Funktionen zu erweitern. Die angestrebten Ziele umfassen die Ermöglichung weiterer Schritte in Richtung Unabhängigkeit von dritter Software, wie beispielsweise QGIS[6]. Ausserdem wird die Machbarkeit einer parallelisierten Berechnung mehrerer Regionen untersucht, um grössere Datenmengen effizienter zu berechnen.

Im Rahmen dieser Arbeit wurde der Webservice um zusätzliche Funktionen ergänzt, die es ermöglichen, die statistischen Inputdaten für die Bestimmung der Zersiedelung zu bearbeiten. Ausserdem wurde eine Lösung konzipiert, um die Rasterdaten der bebauten Fläche anzupassen und deren Einfluss auf die Zersiedelung unmittelbar zu untersuchen.

Die Inputdaten können direkt im Frontend (umgesetzt mit Vue.js) der Webapplikation angepasst werden. Dabei kann die bebaute Fläche, die mittels GeoTIFF-Raster auf einer Leaflet-Karte[7] dargestellt wird, pixelweise verändert werden. Anschliessend kann das Raster als Input für die Verarbeitung im Python-Backend verwendet werden. Zudem können statistische Inputdaten (GeoPackage, GeoJSON), wie z.B. die Einwohneranzahl oder Arbeitsplätze einer Region bearbeitet werden.

Die parallelisierte Berechnung mehrerer Untersuchungsgebiete wurde evaluiert und führt zu einer reduzierten Rechenzeit.

Die Erweiterungen tragen massgeblich dazu bei, Veränderungen der Siedlungsstruktur besser verfolgen zu können. Für Raumplanende fällt die Anpassung der Inputdaten und anschliessende Kalkulation einfacher aus. Die Förderung des Verständnisses von Zersiedelung sowie die Entwicklung von Massnahmen zu deren Eindämmung kann durch den Einsatz der Software in der Lehre und Forschung weiter vorangetrieben werden.

[1] https://www.wsl.ch/de/landschaft/siedlung-und-raum/zersiedelung/
[2] https://www.are.admin.ch/are/de/home.html
[3] https://www.admin.ch/zersiedelungsinitiative
[4] https://www.haupt.ch/buecher/natur-garten/zersiedelung-messen-und-begrenzen.html
[5] https://eprints.ost.ch/id/eprint/1268/
[6] https://www.qgis.org/
[7] https://leafletjs.com/

Management Summary

Ausgangslage

Die Zersiedelung der Landschaft (englisch: ”Urban Sprawl”) ist das Abbild der rapide wachsenden Ausbreitung der Siedlungsfläche und nimmt in der Schweiz zu, was klar gegen das Ziel einer nachhaltigen Raumentwicklung verstösst. Einfamilienhäuser, Gewerbe- und Industriezonen verdrängen dabei Landwirtschafts- und Naturräume. Besonders fernab der Ortskerne zeigt sich das Bild der dünn besiedelten und locker bebauten Gebiete einer zersiedelten Landschaft.

Eine in der Schweiz entwickelte Methode zur Messung der Zersiedelung mit der Messgrösse Z (= Zersiedelung, englisch WUP für ”Weighted Urban Proliferation”) wurde von Schwick et al. (2018) in der Literatur ”Zersiedelung messen und begrenzen” definiert. Als Ausgangslage für die Herleitung von Z dienen die Lage der Gebäude als Fläche der bebauten Gebiete, die Grenzen ebendieser Raumgliederungen (z.B. Gemeinde, Kanton) sowie die Population und die Anzahl der Beschäftigten des Untersuchungsgebiets. Optional kann der Anteil der Siedlungsfläche des Untersuchungsgebiet in die Berechnung miteinbezogen werden. In einer zuvor durchgeführten Studienarbeit wurde ein Webservice auf Grundlage eines quelloffenen Plugins für die Geoinformationssoftware QGIS entwickelt. Der Webservice ermöglicht durch den Einsatz von Parallelisierungsmethoden eine schnellere Berechnung verschiedener Zersiedelungsmetriken, die auf der Methode von Schwick et al. basieren und bietet eine Weboberfläche, um die Ergebnisse der Berechnung zu visualisieren und zu exportieren. Die Berechnungsdauer konnte signifikant reduziert und die Abhängigkeit zu Geoinformationssystemen verringert werden. Mit dieser Arbeit wird der Open Source ”Proof of Concept” des ”USM Calculators” der Studienarbeit weiterentwickelt und um neue Funktionalitäten erweitert.

Ein Ziel des Bundesamtes für Raumentwicklung (ARE) ist es Prozesse zu entwickeln, um Zersiedelungsindikatoren in der kommunalen Planung einzubinden. Die zusätzlichen Funktionalitäten sollen es Raumplanern, Architekten und politischen Entscheidungsträgern ermöglichen, Auswirkungen von geplanten Massnahmen auf die Struktur der Siedlungsfläche besser zu analysieren, zu visualisieren und zu bewerten.

Ziele, Vorgehen, Technologien

Im Wesentlichen sind die Ziele der Arbeit eine weitere Optimierung der Software durch eine parallele Berechnung der Untersuchungsgebiete (Parallelisierung innerhalb der parallelisierten Berechnung) und die Implementierung von neuen Funktionalitäten, um die Anwendungsfreundlichkeit zu verbessern. Relevant wird dies vor allem, um grössere Datenmengen effizienter zu verarbeiten und mehrere Gebiete gleichzeitig zu untersuchen. Nebenbei soll die Benutzeroberfläche weiter ausgebaut werden, um Hilfsfunktionen zu bieten, die es den Benutzern ermöglichen, Anpassungen an den Eingabedaten vorzunehmen, ohne dass dafür zusätzliche Software benötigt wird. Dabei liegt der Fokus auf der Anpassung der Geodaten, wie z.B. das Entfernen und Hinzufügen von Siedlungsflächen oder das Bearbeiten von statistischen Daten, die beide in die Berechnung der Zersiedelung einfliessen. Wenn diese Anpassungen einfach und schnell vorgenommen werden können, sind Benutzer in der Lage, Auswirkungen von geplanten Siedlungsentwicklungen bequemer zu untersuchen und zu visualisieren. Ausserdem soll es möglich sein, eine Selektion von zu berechnenden Gebieten zu treffen und Gebiete sollen verschmolzen und zusammengeführt werden können. Sämtliche Resultate der ausgewählten Untersuchungsgebiete (z.B. die visualisierte Dispersion in Abb. 1) sollen anschliessend mithilfe von Leaflet und Kartenmaterial von OpenStreetMap (OSM) im Frontend dargestellt werden. Für diese Art von Anpassungen war es bisher notwendig, die Geodaten in einem Geoinformationssystem (GIS) zu bearbeiten, was für vor allem für weniger erfahrene Benutzer, für Demonstrationszwecke in der Lehre, oder für rasche Analysen in der Forschung nicht immer praktikabel ist.

Um diese Ziele zu erreichen, wird der in der Studienarbeit entwickelte Webservice (Python-Backend mit Starlette) weiterentwickelt und um die in der Aufgabenstellung definierten Funktionalitäten erweitert. Die Funktionalitäten werden der Weboberfläche (Frontend mit Vue.js und Ant Design UI Framework) mit neuen Komponenten hinzugefügt und anschliessend mittels User Experience (UX) Tests auf Benut-

zerfreundlichkeit geprüft. Für die Rasterbearbeitung muss eine Lösung entwickelt werden, die sich in die bestehende Leaflet-Komponente einfügen lässt und eine responsive Bearbeitung der Rasterdaten ermöglicht.

Verschiedene Parallelisierungsansätze sollen in einer Analyse empirisch untersucht und miteinander verglichen werden, um die effizienteste und wartungsfreundlichste Lösung zu finden, die sich in die bestehende Softwarearchitektur einfügen lässt. Konkret werden dabei verschiedene Python-Bibliotheken wie Dask, Joblib oder die Standardbibliothek multiprocessing evaluiert, um die Berechnungen zu parallelisieren.

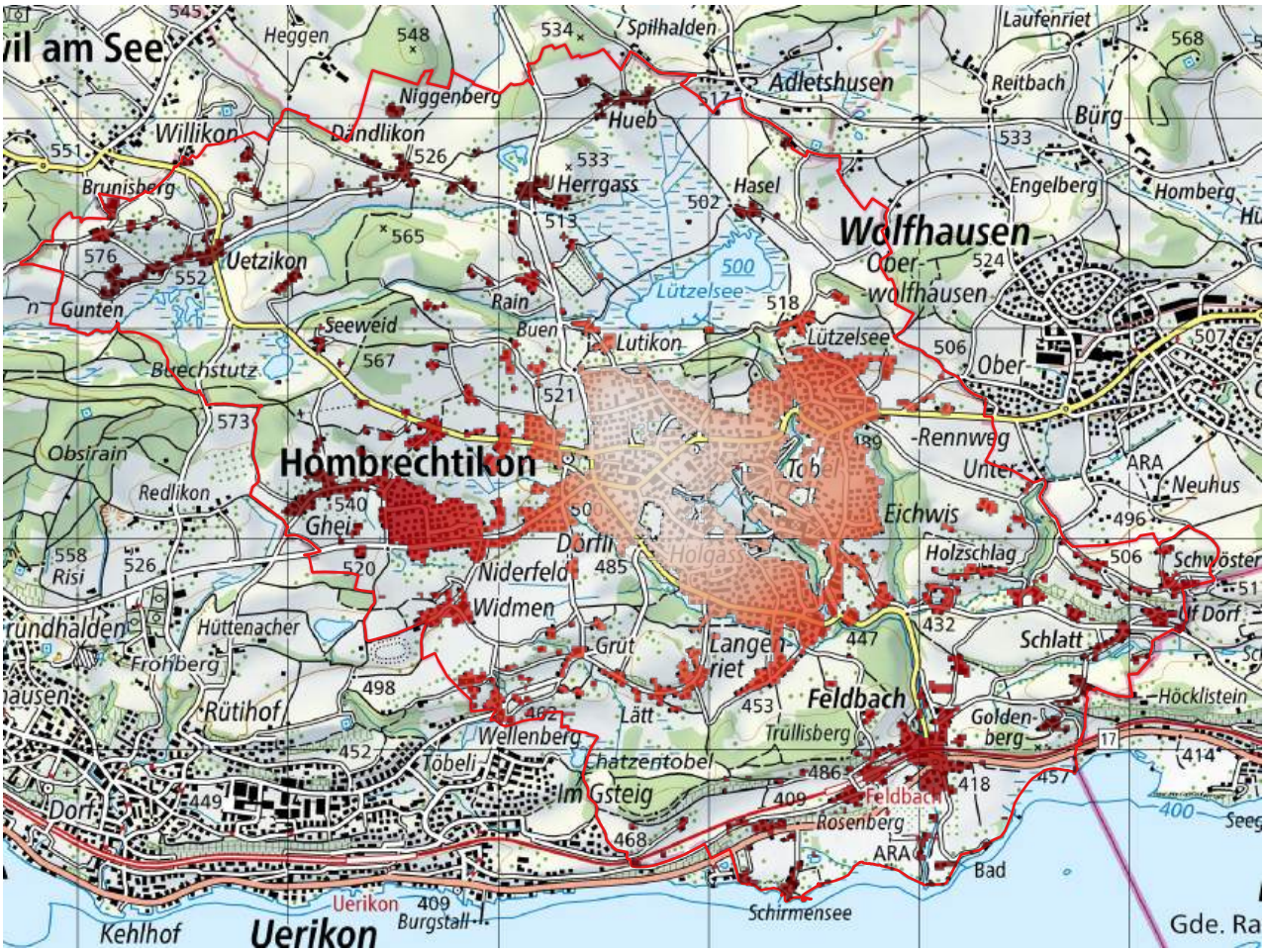


Abbildung 1: Kartographische Darstellung der Dispersion: Dunkelrote Gebiete zeigen eine grosse Zerstreuung der bebauten Flächen

Ergebnisse

Das Resultat der Arbeit ist eine erfolgreiche Erweiterung des ”USM Calculators” um die in der Aufgabenstellung definierten Funktionalitäten. Die Benutzeroberfläche (vgl. Abb. 2) wurde um neue Komponenten erweitert, die es den Benutzern ermöglichen, Eingabedaten zu bearbeiten, Untersuchungsgebiete auszuwählen und zu verschmelzen. Für die Integration in andere Anwendungen wurde die bestehende standardisierte und dokumentierte REST-API (nach OpenAPI Spezifikation) um eine ausführlichere Dokumentation und neue Endpunkte ergänzt.

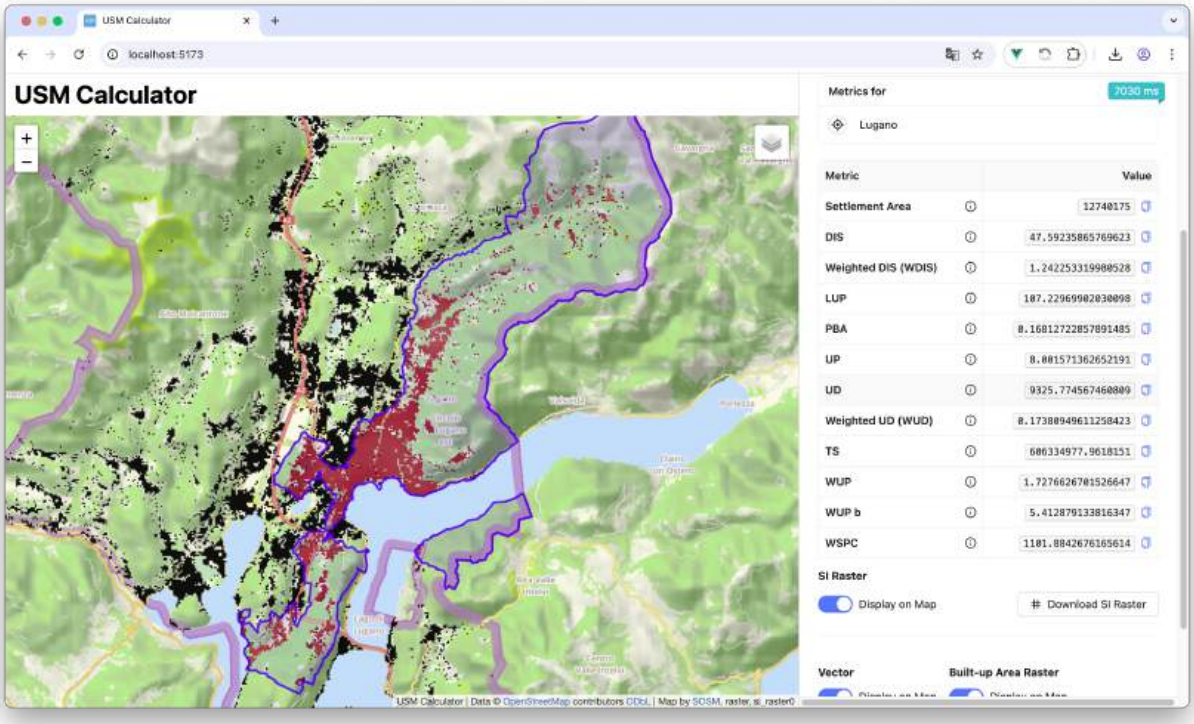


Abbildung 2: Benutzeroberfläche des ”USM Calculators” mit neuen Funktionalitäten. Eine intuitive Oberfläche vereinfacht das Arbeiten mit Geodaten und Zersiedelungsindikatoren.

Grössere Datenmengen können nun dank einer parallelen Berechnung der Untersuchungsgebiete effizienter verarbeitet werden. Der Einsatz des Frameworks Joblib ermöglicht eine wartungsfreundliche und simple Implementierung der Parallelisierung, die sich in die bestehende Softwarearchitektur einfügt. Es zeigt sich, dass die Parallelisierung der Berechnung im Optimalfall zu einer Reduktion der Berechnungsdauer von bis zu 50% führt, insbesondere bei mehreren Untersuchungsgebieten (vgl. Abb. 3).

Das Produkt dieser Arbeit ist eine vollständige Open Source Software, die auf einem eigenen Server betrieben werden kann und in der Lehre und Forschung zur Förderung der Zersiedelungsforschung eingesetzt werden kann. Diese Dokumentation hält eine ausführliche Beschreibung der Erkenntnisse und Erfahrungen, die aus der Entwicklung dieser Erweiterungen gewonnen wurden, für zukünftige Weiterentwicklungen fest.

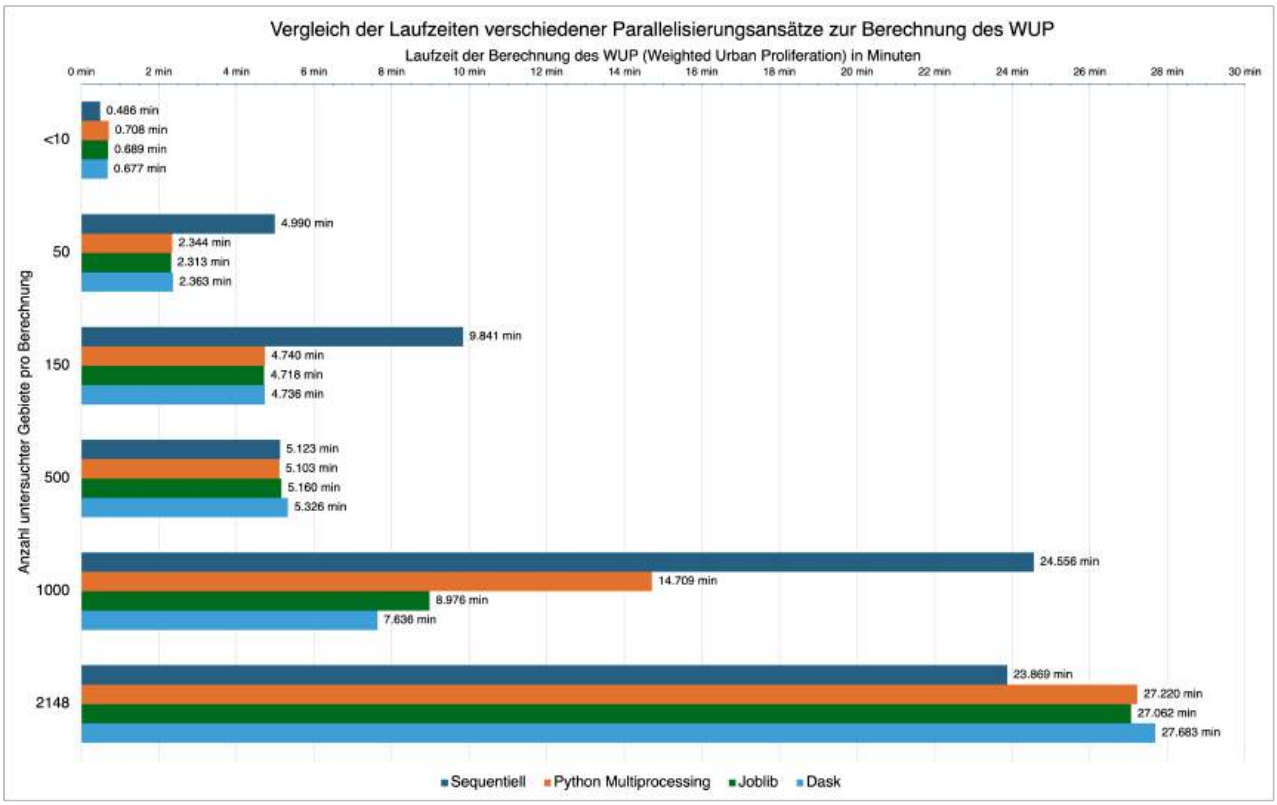


Abbildung 3: Benchmark der Berechnungsdauer: Vergleich verschiedener Parallelisierungsansätze. Die Berechnungsdauer wird in Minuten angegeben

Ausblick

Diese Arbeit zeigt, dass die Umsetzung der neuen Anforderungen erfolgreich war und die Benutzerfreundlichkeit des ”USM Calculators” verbessert wurde, wobei die Hauptziele erreicht wurden.

Zukünftige Arbeiten könnten sich auf die weitere Optimierung der Software konzentrieren, um die Berechnungsdauer und Parallelisierung weiter zu verbessern. Vor allem Konfigurationsparameter für die Parallelisierung könnten in der Benutzeroberfläche von den Benutzern angepasst werden, um die Berechnungen genauer zu gestalten. Weitere Einstellungen könnten es den Benutzern erlauben, Aspekte wie z.B. der Radius der Nachbarschaftsbetrachtung bei der Bestimmung des SI (Sprawl at Index) abzuändern, die Grenzen des Untersuchungsgebiets anzupassen oder eine Auswahl der zu berechnenden Zersiedelungsmetriken zu treffen.

Als wichtigster Punkt zur Erweiterung wird die Implementierung einer langfristigen Speicherung der Berechnungsergebnisse angesehen, um Projekte über längere Zeiträume hinweg zu verfolgen. Die Ergebnisse der Berechnung könnten in einer Datenbank gespeichert werden, um bestimmten Anwendern, wie z.B. Raumplanern oder Architekten eine einfache Abfrage und Analyse der Zersiedelungsmetriken zu ermöglichen und diese in ihre Planungsprozesse zu integrieren.

Inhaltsverzeichnis

Aufgabenstellung

Hintergrund und Problemstellung

Die Herausforderung: Die Zersiedelung (”urban sprawl”) nimmt zu, aber niemand weiss wie stark... Die Lösung: Eine Messmethode - eine Metrik namens ”Urban Sprawl Metrics” (USM). Diese Metrik ist bereits in einem QGIS Plugin implementiert. QGIS ist ein freies und quelloffenes geographisches Informationssystem (GIS) zur Erfassung, Analyse und Darstellung räumlicher Daten. Die Abhängigkeit vom QGIS-Ökosystem bringt jedoch auch Nachteile mit sich, wie z.B. eine geringere Geschwindigkeit.

Aufgaben

Die in der Studienarbeit implementierte Anwendung soll nun um folgende Funktionalitäten erweitert werden:

- Der Benutzer kann ein oder mehrere Gebiete auswählen, für die die Berechnung der Zersiedelung durchgeführt werden soll.
  - Die Berechnung ist für jedes Gebiet einzeln durchzuführen.
  - Die Berechnung der einzelnen Bereiche sollte durch Parallelisierung (Parallelisierung in der Parallelisierung) optimiert werden.
- Der Benutzer kann die statistischen Daten (z.B. Einwohnerzahl) für jedes Gebiet leicht ändern.
- Der Benutzer kann das geladene Raster ändern.
  - Der Benutzer kann die bebaute Fläche vergössern.
- (Evtl.) Der Benutzer kann Gebiete für die Berechnung zusammenfassen.
  - Gebietsgrenzen, statistische Daten usw. werden zusammengefasst und als ein Gebiet behandelt.

Technologien

- Frontend: Modernes Javascript Framework
- Backend: Python-Framework (Django/Flask, Starlette), OpenAPI/Swagger, Database PostgreSQL, GeoJSON (evtl. OGC API Processes).
- Browser/Runtime/OS/Hardware: Win/Unixoids/macOS sowie die gängigsten Browser (Chrome, Firefox, Edge, Safari).

Daten

- Schweizer Gemeindegrenzen inkl. Statistikdaten der Gemeinden
- Siedelungsraster der Schweiz

Teil I

Technischer Bericht

1 Einführung

1.1. Problemstellung

Die ungehemmte Ausdehnung der Siedlungsfläche zeigt sich in der Zunahme von Einfamilienhäusern, Industrie- und Gewerbegebieten und Infrastruktur ausserhalb der bestehenden Siedlungszentren. Dabei geschieht dies auf Kosten der Landwirtschaftsfläche und der Natur [1].

Das Problem ist nicht nur ein rein geospezifisches oder raumplanerisches, sondern auch ein gesellschaftliches. In der Schweiz hat das Bundesamt für Raumentwicklung (ARE) [2] den Auftrag des Bundes, die Zersiedelung in einem nachhaltigen Rahmen zu messen und zu begrenzen. Verschiedene Metriken und deren Formeln zur Bemessung der Zersiedelung wurden von Schwick et al. (2018) [3] in der Literatur "Zersiedelung messen und begrenzen" begründet. Als Grundlage für die Berechnung der Zersiedelung dienen ein Raster der Siedlungsfläche (vgl. [4]), die Grenzen der Raumgliederungen (z.B. Gemeinde, Kanton) sowie die Bevölkerungs- und Beschäftigtenzahlen innerhalb des Untersuchungsgebiets. Ein solcher Algorithmus zur Berechnung wurde bereits in der Bachelorarbeit von Horiguchi und Schwab (2024) [5] als QGIS [6] Plugin [7] entwickelt. Das quelloffene Plugin wendet die von Schwick et al. definierten Metriken an und wurde später in der Studienarbeit von Fehr und Maag (2024) [8] als Open Source Webservice umgesetzt, mit dem Ziel die Abhängigkeit zu Geoinformationssystemen zu verringern und die Berechnungsdauer zu reduzieren. Die Zielsetzung im Rahmen dieser Arbeit ist es, den "Urban Sprawl Calculator" als Werkzeug funktional zu erweitern, um die Abhängigkeit zu weiteren Softwarelösungen wie QGIS für ad hoc Zersiedelungsanalysen zu verringern.

Mit dem Webservice des Urban Sprawl Calculators sollen Planungsbehörden, Raumplaner und Architekten ein weiteres Werkzeug erhalten, um die Auswirkungen von geplanten Massnahmen auf die Siedlungsentwicklung zu visualisieren und zu analysieren. Zusammen mit politischen Entscheidungsträgern stellt dieses Publikum die Zielgruppe des Webservices dar.

1.2. Vision

Der bestehende Webservice soll um zusätzliche Funktionalitäten erweitert werden, um den Calculator als Werkzeug für die Raumplanung weiter zu etablieren. Als Webtool soll der Urban Sprawl Calculator einer breiteren Zielgruppe zugänglich gemacht werden, um allenfalls in der Lehre und Forschung eingesetzt zu werden. Die Evaluation der Zersiedelung soll für Anwender möglichst einfach gestaltet werden, so dass auch weniger technisch versierte Benutzer in der Lage sind, die Auswirkungen von geplanten Siedlungsentwicklungen zu untersuchen und zu visualisieren. Durch schnelle Resultate und eine einfache Visualisierung der Metriken kann der Urban Sprawl Calculator als Werkzeug für die Raumplanung etabliert werden.

1.3. Ziele

Das Hauptziel liegt in der Umsetzung weiterer Funktionalitäten, die es ermöglichen, die statistischen Inputdaten für die Berechnung der Zersiedelungsmetriken zu bearbeiten. Ohne Abhängigkeiten zu weiteren Softwarelösungen wie QGIS soll die Lösung Vergleiche zwischen verschiedenen raumplanerischen Ausgangslagen rascher und einfacher ermöglichen. Um grössere Datenmengen effizienter zu berechnen, soll die Machbarkeit einer Parallelisierung von Berechnungen mehrerer Regionen untersucht werden. Weil die Berechnung der Zersiedelung bereits auf parallelisierte Algorithmen setzt, soll die Umsetzung einer parallisierten Berechnung in einer anderen Parallelisierung erprobt werden.

Diese Ziele sollen erreicht werden, indem die bestehende Softwarearchitektur erweitert wird.

1.3.1. Usability

Mittels User Experience (UX) Tests soll die Benutzerfreundlichkeit der neuen Funktionalitäten überprüft und optimal umgesetzt werden. Als Werkzeug für die Usability-Tests kann das Prototyping-Tool Balsamiq Mockups [9] verwendet werden, um die Benutzeroberfläche zu entwerfen und zu testen. Dabei soll ein Prototyp des User Interfaces (UI) erstellt werden, der die neuen Funktionalitäten und die Benutzerführung simuliert.

1.3.2. Performance

Ein weiterer Fokus liegt aufgrund der Umsetzung als Webservice auf der Performance der Berechnungen. Die Berechnung der Zersiedelung soll so optimiert werden, dass grössere Datenmengen effizienter verarbeitet werden können. Dafür sollen zunächst Ansätze zur Parallelisierung der Berechnungen in Python [10] implementiert werden. Mit diesen Ansätzen soll die Berechnung von mehreren verschiedenen Konstellationen von Untersuchungsgebieten empirisch untersucht und verglichen werden. Die Wartbarkeit der Ansätze soll dabei ebenfalls in die Bewertung einfliessen, um eine langfristige Nutzung der Software zu gewährleisten. Die passendste Lösung soll anschliessend in das bestehende auf dem Starlette-Webframework [11] basierte Backend integriert werden.

Die Umsetzung soll mit Python erfolgen, da bestehender Code auf dieser Sprache basiert und die bestehende Softwarearchitektur auf Python-Paketen aufbaut.

1.4. Erweiterung der bestehenden Lösung

Die bestehende Dokumentation des Application Programming Interface (API) mit Swagger [12] wird mit detaillierteren Information zu den Endpunkten erweitert. Die Parallelisierung der Berechnung von den Regionen wird mit verschiedenen Ansätzen geprüft und anschliessend eine passende Lösung implementiert. Verschiedene Python-Bibliotheken wie Dask [13], Joblib [14] oder die Standardbibliothek multiprocessing [15] werden evaluiert, um die Berechnungen zu parallelisieren. Neue Vue.js-Komponenten [16] werden in das Frontend integriert, um die Statistikdaten der Untersuchungsgebiete zu bearbeiten oder zusammenzuführen und diese sofort für die Berechnung der Zersiedelung zu verwenden. Die Rasterdaten sollen ebenfalls in der Komponente der Leaflet-Karte [17] bearbeitet werden können, um die Anpassung der Siedlungsfläche zu ermöglichen und deren direkten Auswirkungen auf die Zersiedelungsmetriken zu analysieren. Dafür werden verschiedene Ansätze evaluiert, wobei eine Integration in die bestehende Leaflet-Komponente über einen Leaflet ImageOverlay [18] angestrebt wird.

1.5. Rahmenbedingungen

Mit dem Start der Arbeit wurden von den Autoren die Rahmenbedingungen des Projekts wie folgt festgelegt:

Bedingung	Beschreibung
Dokumentationssprache	Deutsch
Softwaresprache	Englisch
Dokumentationstool	LaTeX
Versionsverwaltung	Git (OST-intern Gitlab)
Projektmanagement	Microsoft Teams (Kanban-Board von Microsoft Planner)
Technologien	Python 3, GDAL, Starlette, Vue.js, Docker

Tabelle 1.1: Rahmenbedingungen des Projekts.

In einer späteren Phase des Projekts wurden die nicht-funktionalen Anforderungen definiert und im Abschnitt 6.2 auf Seite 20 festgehalten.

1.6. Vorgehen

Die Durchführung erfolgt nach dem Rational Unified Process (RUP) [19]. Im Kapitel 11 auf Seite 71 wird das Vorgehen genauer beschrieben. Die Versionsverwaltung erfolgt mit Git, wobei die Softwareentwicklung nach dem Git Feature Branch Workflow [20] [21] organisiert wird. Das Bundesamt für Raumentwicklung ARE fungiert als Auftraggeber.

2

Stand der Technik

2.1. Metriken der Zersiedelung nach Schwick et al.

Die Metriken der Zersiedelung wurden von Schwick et al. (2018) [3] in der Literatur "Zersiedelung messen und begrenzen" definiert und die wichtigste Metrik ist weithin als "Weighted Urban Proliferation" (WUP) oder "Zersiedelung" (Z) bekannt. Diese quantitative Methode zur Messung der Zersiedelung basiert auf der räumlichen Verteilung von Siedlungsflächen und der Bevölkerungs- und Beschäftigtenzahlen in einem definierten Untersuchungsgebiet.

Die Metriken bilden auch in dieser Arbeit die Grundlage für die Berechnung der Zersiedelung.

2.2. USM Toolset für QGIS

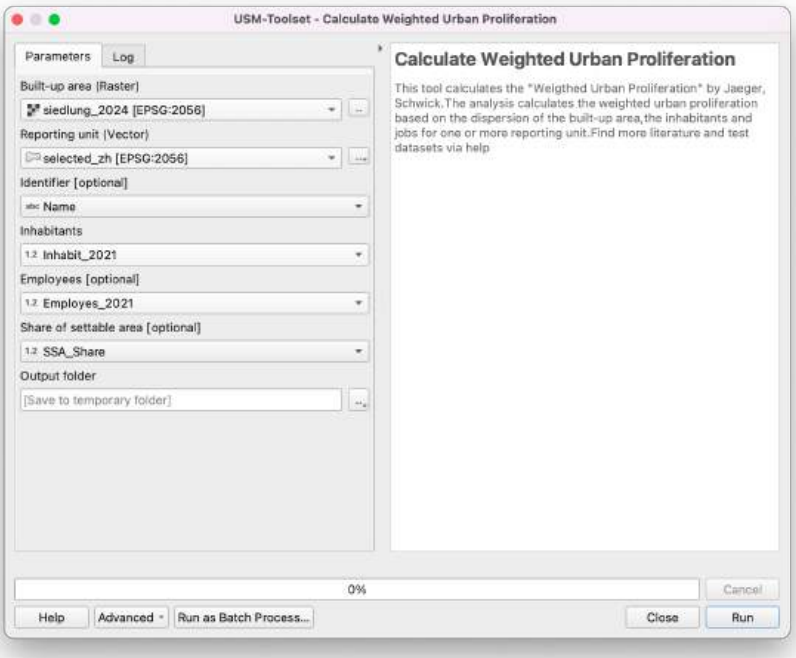


Abbildung 2.1: USM Toolset Plugin in QGIS [7].

Das USM Toolset (vgl. Abbildung 2.1) ist ein quelloffenes Plugin für die Geoinformationssoftware QGIS, das von Horiguchi und Schwab (2020) [5] entworfen wurde. Weiterentwickelt wurde das Plugin am IFS der Ostschweizer Fachhochschule (OST), unter der GNU General Public License (GPL) v3.0 [22] veröffentlicht und ist auf der QGIS Plugin Seite [7] verfügbar. Der Code nutzt die Python-Bibliothek PyQGIS [23] und ist in Python mit einer Qt-basierten Benutzeroberfläche implementiert. Die Berechnung der Zersiedelung erfolgt sequenziell für alle Regionen, die im hochgeladenen Geopackage enthalten sind.

2.3. Urban Sprawl Metrics Calculator Webservice

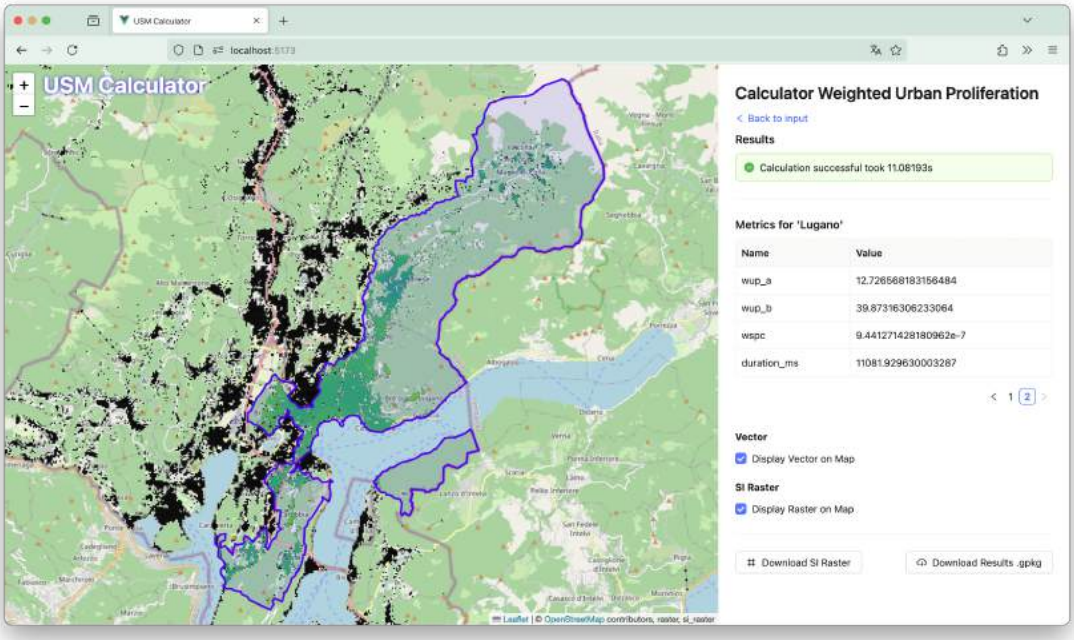


Abbildung 2.2: Urban Sprawl Metrics Calculator Webservice [8].

Der Urban Sprawl Metrics Calculator Webservice (siehe Abbildung 2.2) ist eine Weiterentwicklung des USM Toolset (von Horiguchi und Schwab (2020) [5]). Dieser Webservice wurde von den Autoren Fehr und Maag (2024) [8] als Proof of Concept umgesetzt. Der Webservice ermöglicht die Berechnung der Zersiedelung anhand von Statistik- und Vektordaten im Geopackage-Format, und Rasterdaten im GeoTIFF-Format, welche die bebaute Fläche darstellen (z.B. Siedlungsraster). Aufgeteilt ist der Webservice in ein Frontend, das mit Vue.js und dem Ant Design UI [24] Framework umgesetzt wurde, und ein Backend, das auf Python mit Starlette basiert. Für die Geodatenverarbeitung werden die umfrangreiche Bibliothek GDAL [25] und die Python-Bibliothek numpy [26] verwendet.

Der Fokus der Lösung liegt auf einer schnellen Berechnung der Zersiedelung, die auf den Metriken von Schwick et al. basiert. Dabei wurde die Berechnung des SI-Rasters (Sprawl at Index Raster) unter Anwendung der Python-Bibliothek numba [27] [28] optimiert, um die Berechnungsdauer zu reduzieren. Der Webservice ist als Open Source Software verfügbar und kann auf einem eigenen Server betrieben werden.

Die Berechnung der Zersiedelung erfolgt sequenziell für alle Regionen, die im hochgeladenen Geopackage enthalten sind. Die Regionen werden dabei anhand der Grenzen der Raumgliederungen (z.B. Gemeinde, Kanton) definiert. Diese Regionen enthalten die Statistikdaten, die für die Berechnung der Zersiedelung benötigt werden, wie z.B. die Bevölkerungs- und Beschäftigtenzahlen. Im Frontend können die Platzhalter für die Statistikdaten ausgewählt werden, die für die Berechnung der Zersiedelung verwendet werden sollen. Das SI-Raster wird aus den Vektordaten der bebaute Fläche und den Statistikdaten berechnet und kann nur für die erste Region angezeigt werden.

2.4. Zusammenführen von Regionen

Das Zusammenführen von Regionen ist eine Funktion, die in der bestehenden Lösung nicht implementiert ist. Eine Möglichkeit, Regionen zusammenzuführen, besteht momentan darin, die Grenzen der Raumgliederungen manuell zu bearbeiten und die Regionen in einem Geoinformationssystem (GIS) wie QGIS zu verschmelzen.

Bei QGIS kann dies durch die Verwendung des Werkzeugs ”Gebiete verschmelzen” [29] erfolgen, das es ermöglicht, mehrere Regionen zu einer einzigen Region zusammenzuführen. Die verschmolzene Region kann dann als neues Shapefile oder GeoPackage exportiert werden, das im Urban Sprawl Metrics Calculator Webservice verwendet werden kann.

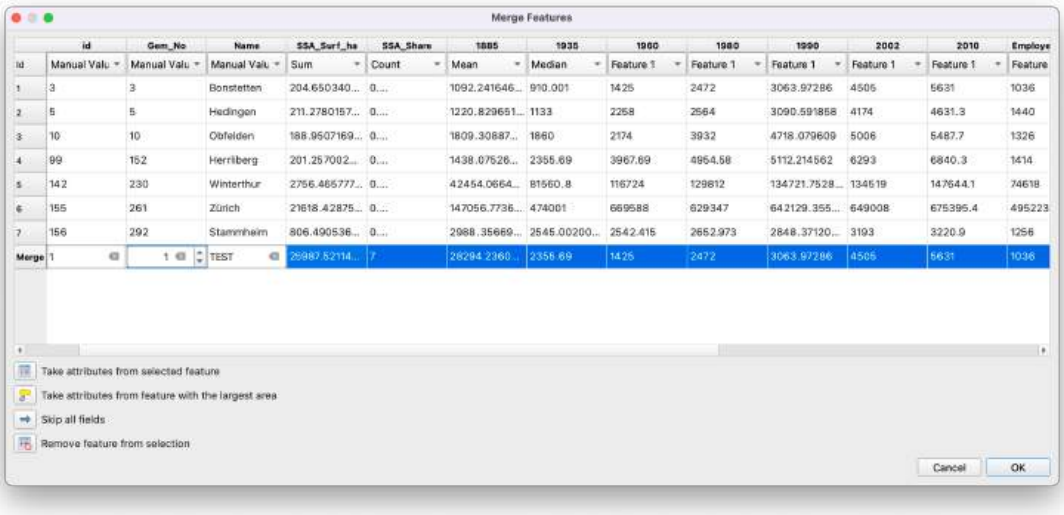


Abbildung 2.3: Zusammenführen von Regionen in QGIS [29].

Diese Funktionalität (vgl. Abbildung 2.3) soll ähnlich in der neuen Lösung implementiert werden, um die Benutzerfreundlichkeit zu verbessern und die Abhängigkeit von Geoinformationssystemen zu verringern.

2.5. Defizite der bestehenden Lösungsansätze

Verbesserungspotential gibt es bei der Abfolge von Berechnungen von mehreren Regionen. Bisher müssen die Regionen über ein zusätzliches Tool (QGIS) ausgewählt werden, um die Zersiedelungsberechnung auf auserwählte Regionen einzuschränken. Eine Möglichkeit zur Auswahl der Regionen würde in diesem Fall die Arbeitsschritte vereinfachen.

Des Weiteren ist die Anpassung der statistischen Daten über den Webservice für die Berechnung der Zersiedelung nicht möglich. Nebenbei kann das Siedlungsraster, das die bebaute Fläche darstellt, nicht über die Lösung angepasst werden. Für Anpassungen dieser Arten muss auf zusätzliche Software zurückgegriffen werden.

Die Anpassung der statistischen Daten und des Siedlungsrasters muss in einem Geoinformationssystem (GIS) wie QGIS erfolgen. Sollte die Auswahl der Regionen eingeschränkt werden, müssen die Regionen manuell ausgewählt und in ein neues Shapefile oder GeoPackage exportiert werden. Rasteranpassungen müssen mit einem Rasterbearbeitungstool innerhalb von QGIS vorgenommen werden, wobei die Rasterdaten in der Regel nicht pixelweise angepasst werden können.

Diese Defizite machen die Arbeitsschritte zur Analyse verschiedener Szenarien innert kurzer Zeit umständlich und zeitaufwändig.

3
Bewertung

Die Bewertung möglicher Lösungsansätze wird in diesem Kapitel beschrieben. Eine geeignete Lösung für die parallele Berechnung der Zersiedelung und der Rasterbearbeitung soll daher gemäss verschiedenen Kriterien bewertet werden.

Designentscheidungen der Softwarearchitektur und Vorgehensweisen werden mit dem ”Y-Statements” [30] festgehalten. Die Vorgehensweise soll dabei die Entscheidung für eine bestimmte Option und die Vernachlässigung anderer Optionen begründen und dokumentieren. Bewertet werden in diesem Sinne die Qualität der Lösung und die Akzeptanz von Nachteilen, um ein bestimmtes Ziel zu erreichen. In der Abbildung 3.1 ist die Vorgehensweise visuell in Form eines Diagramms dargestellt.

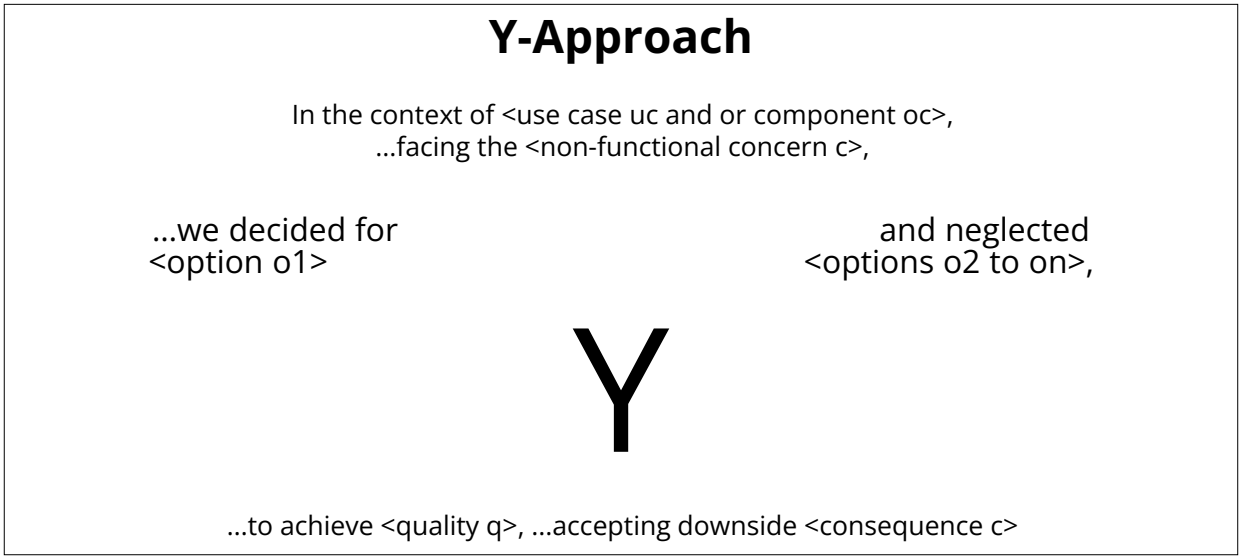


Abbildung 3.1: Y-Approach mit Erläuterungen in englischer Sprache. Eigene Darstellung.

3.1. Kriterien

Die parallelisierte Berechnung der Zersiedelung soll vor allem hinsichtlich der Performance bewertet werden. Dies geschieht, indem die totale Berechnungsdauer der Untersuchungsgebiete in verschiedenen Konfigurationen (z.B. Verschiedene Memoryeinstellungen) und mit verschiedenen Anzahl an parallelen Threads gemessen wird. Mindestens drei Berechnungen werden durchgeführt und anschliessend wird der gemittelte Wert zur Beurteilung der Performance verwendet.

Die Gebiete des Vektors können unabhängig voneinander parallel berechnet werden, da die Berechnung der Zersiedelung für jedes Gebiet separat erfolgt. Es handelt sich somit um ein ”Embarassingly Parallel Problem”, das eine einfache Art der Parallelisierung ohne Synchronisation von Threads ermöglicht. Einzig die Rasterdaten müssen für jedes Gebiet separat geladen werden, was die Performance der Berechnung und den Arbeitsspeicherbedarf beeinflussen kann, was untersucht wird.

Folgende Kriterien werden für die Bewertung der Zielerreichung im Rahmen der parallelisierten Berechnung der Untersuchungsgebiete herangezogen:

Kriterium	Art	Beschreibung
Performance	Quantitativ	Die Performance der Lösung wird anhand der totalen Berechnungsdauer der Untersuchungsgebiete gemessen.
Wartbarkeit	Qualitativ	Die Wartbarkeit der Lösung wird mittels der Lesbarkeit des Codes und der Verfügbarkeit der Dokumentation gemessen.
Korrektheit	Qualitativ	Zur Bestimmung der Korrektheit werden die Resultate mit denen des bestehenden QGIS-Plugins manuell verglichen.

Tabelle 3.1: Kriterien für die Bewertung der Performance der Lösung.

3.1.1. Kriterien zur Bewertung der Rasterbearbeitung

Zur Bewertung der Rasterbearbeitung werden die Kriterien der Wartbarkeit und Korrektheit herangezogen. Die Performance wird nur im Zusammenhang mit einer responsiven Benutzeroberfläche bewertet, da die Messung verschiedener Rasteroperationen nicht sinnvoll ist. Folglich sollen Lösungen anhand ihrer Wartbarkeit und bestmöglichen Korrektheit bewertet werden.

3.2. Schlussfolgerung

Die Messung der Performance der parallelisierten Berechnung erfolgt mit dessen Durchführungen in verschiedenen Konfigurationen und mit verschiedenen Anzahlen an parallelen Threads. Die Durchführungen und die Parameter der Berechnungen sind in Kapitel 10 beschrieben. Nebenbei werden die Ergebnisse der Berechnung mit denen des bestehenden QGIS-Plugins und des Webservices verglichen, um die Korrektheit der Resultate sicherzustellen.

4

Umsetzungskonzept

4.1. Beschreibung

Der Prototyp des Webservice des ”Urban Sprawl Metrics Calculators” (auch ”USM Calculator”) soll um zusätzliche Funktionalitäten erweitert werden, um die Unabhängigkeit zu anderen GIS Tools zu erhöhen und die Benutzerfreundlichkeit zu verbessern. Es soll validiert werden, ob die Zersiedelungsberechnung für mehrere Regionen parallel durchgeführt werden kann. Weiter werden Möglichkeiten der Raster- und Vektordaten-Bearbeitung untersucht und geeignete Lösungen implementiert, um die Eingabedaten im Webtool direkt anpassen zu können.

Die Studienarbeit [8] und der dabei entwickelte Prototyp des ”USM Calculators” dienen als Grundlage für die Umsetzung. Die Weiterentwicklung des Prototyps soll auf den Ergebnissen der Studienarbeit aufbauen und diese massgeblich erweitern. Die Framework- und Software-Architektur des Prototyps soll dabei beibehalten werden, um die Weiterentwicklung zu vereinfachen. Dabei wird Python als Programmiersprache für die Backend-Entwicklung verwendet, während Vue.js für die Umsetzung der Benutzeroberfläche zum Einsatz kommt. Nebenbei sollen bestehende und neue API-Endpunkte besser dokumentiert werden, um Benutzern und Entwicklern eine detaillierte Übersicht über die Funktionalitäten des Webservice zu bieten. Die Software soll schlussendlich in Docker-Containern (jeweils einen Container pro Architekturschicht, siehe Abschnitt 9.3.1) ausgeliefert werden, um eine einfache Installation und Ausführung zu ermöglichen.

Zur Umsetzung der Parallelisierung der Berechnung von mehreren Regionen wird eine geeignete Lösung evaluiert und im Anschluss implementiert. Dabei kommen verschiedene Ansätze in Betracht, wie z.B. die Verwendung des Python-Moduls multiprocessing [15], das die parallele Ausführung von Prozessen ermöglicht, oder die Nutzung von asynchronen Funktionen mit dem Modul asyncio [31]. Weitere Möglichkeiten sind die Verwendung von Task-Queues wie Dask [13] oder Parallel Computing Tools wie Joblib [14].

Die Entscheidung fiel auf die Verwendung von Joblib in Kombination in einem ”Background Task” mit der Python-Bibliothek Starlette [11], die als Web-Framework für den Webservice dient. Bei Starlette handelt es sich um ein leichtgewichtiges, asynchrones ASGI (Asynchronous Server Gateway Interface) Web-Framework, das sich optimal eignet, um Webanwendungen zu erstellen, die eine hohe Leistung und Skalierbarkeit erfordern. Der ”Background Task” ermöglicht es, langwierige Berechnungen im Hintergrund auszuführen, ohne die Benutzeroberfläche des Webservice zu blockieren. Nebenbei bietet Joblib eine einfach implementierbare Möglichkeit, die Berechnungen der Regionen zu parallelisieren.

Im Kapitel 6 befindet sich die Softwaredokumentation zur Implementierung. Dieser können genauere Entscheidungen, deren Begründungen und die Umsetzung der Lösung entnommen werden.

5
Resultate

5.1. Zielerreichung

Die Webapplikation "USM Calculator" wurde erfolgreich erweitert und bietet nun weitere Möglichkeiten zur Erforschung der Zersiedelung. Anwender haben dank der Integration von Vektor- und Rasterdaten-Bearbeitung die Möglichkeit, Eingabedaten direkt im Webservice anzupassen. Die Effizienz der Berechnung der Zersiedelung von mehreren Gebietseinheiten (wie z.B. Gemeinden) konnte gesteigert werden. Das Open Source Projekt kann nun von Entwicklern offen weiterentwickelt werden, weil der Quellcode frei verfügbar ist.

Die API-Schnittstelle des Webservice ist dokumentiert und ermöglicht es Entwicklern, vor allem Berechnungsmethoden in alternativen Anwendungen zu nutzen. Zusätzlich wurde für das Zusammenführen von Gebieten eine Schnittstelle implementiert, die es ermöglicht, Gebiete des Vektors zu vereinigen. Die API könnte in Zukunft ebenfalls von Plugins für GIS-Tools genutzt werden, um die Zersiedelungsberechnung auf einen Server auszulagern, was die Performance der Berechnung verbessern könnte.

Codeänderungen wurden im Verlauf des Projekts in einem Git-Repository dokumentiert. Diese Änderungen (Commit) werden zusammengefasst und mit einem einzigen Commit dargestellt, der die neuen Funktionen und Verbesserungen beinhaltet. Das Vorgehen wurde gewählt, um eine klare Übersicht über die Änderungen zu behalten. Kleinere Commits, die während der Entwicklung gemacht wurden, sollten nicht öffentlich sein, da sie teils nur während der Entwicklung relevant waren.

5.2. Ausblick: Weiterentwicklung

In diesem Abschnitt werden weitere Möglichkeiten der Weiterentwicklung der Software skizziert, die über die im Rahmen dieser Arbeit umgesetzten Funktionen hinausgehen. Die Architektur der Software ist so gestaltet, dass weitere Funktionen möglichst effizient implementiert werden können.

5.2.1. Ad-hoc-Vergleich von Veränderungen in der Siedlungsentwicklung

In der aktuellen Version der Software ist es möglich, die Siedlungsentwicklung in mehreren Schritten zu vergleichen. Dazu kann das Siedlungsraster nach jeder Berechnung einer Region angepasst und gespeichert werden. Nach jeder Berechnung kann das SI-Raster gespeichert und mit dem vorherigen verglichen werden. Dazu ist momentan eine weitere GIS-Software notwending, die den Vergleich der Rasterdaten des SI-Rasters ermöglicht.

In einer zukünftigen Version könnte ein direkter Vergleich der SI-Raster und der berechneten Zersiedelungsmetriken von beliebigen Berechnungen innerhalb des Webservice implementiert werden. Dies würde beispielsweise eine Speicherung der SI-Raster und der Resultate in einer Datenbank erfordern. Die Implementierung eines derartigen Vergleichs würde es ermöglichen, die Siedlungsentwicklung in einem Gebiet zu verschiedenen Zeitpunkten zu vergleichen und die Auswirkungen von Änderungen in der Siedlungsstruktur direkt zu visualisieren.

5.2.2. Langfristige Speicherung der Resultate in einer Datenbank

Bereits in der Studienarbeit der Autoren wurde die Möglichkeit der langfristigen Speicherung im Kapitel "Resultate" skizziert [8]. Dabei geht es darum, die Resultate und Eingabeparameter der Berechnungen langfristig in einer Datenbank zu speichern, um diese später wieder abrufen zu können. Neben einer Anbindung an eine Datenbank würden Überlegungen zur Sicherheit und zum Datenschutz der gespeicherten Daten notwendig sein. Von entsprechenden Massnahmen, wie z.B. Verschlüsselung und den Einsatz von Zugriffsrechten, darf nicht abgesehen werden.

Vergleiche von verschiedenen Siedlungsentwicklungen, wie bereits im vorherigen Abschnitt beschrieben, liessen sich so effizienter umsetzen.

5.2.3. Mehrbenutzersystem: Zugangsberechtigungen und Benutzerverwaltung

Aus Gründen der Sicherheit und des Datenschutzes wäre es sinnvoll, eine Benutzerverwaltung zu implementieren, vorausgesetzt, dass die Software Resultate langfristig speichern sollte. Mittels Benutzerkonten könnten Zugriffsrechte auf gespeicherte Daten und Funktionen der Software vergeben werden. Ein Benutzer könnte beispielsweise Berechnungen starten und die Resultate zu einem späteren Zeitpunkt abrufen. Resultate könnten mit anderen Benutzern mit verschiedenen Rollen geteilt werden, was die Zusammenarbeit an Projekten erleichtern würde. Solche Rollen könnten beispielsweise Administratoren, Projektleiter und Gäste umfasse, wobei jede Rolle unterschiedliche Berechtigungen hat. Mit der Implementierung eines Benutzerprofils könnte die Software auch personalisierte Einstellungen und Präferenzen, wie z.B. die Kartenansicht oder die Farbpalette der SI-Raster-Darstellung für jeden Benutzer speichern. Nebenbei könnten laufende Bearbeitungen an den Statistik- und Rasterdaten zwischengespeichert werden und bei Bedarf wiederhergestellt werden, was die Arbeit an komplexen Projekten erleichtern würde.

Für Zugriffe der API-Schnittstellen können Zugansgsberechtigungen über eine Authentifizierung mittels API-Keys oder Tokens implementiert werden.

Bei einem Mehrbenutzersystem müsste zusätzlich sichergestellt werden, dass die Software in der Lage ist, mehrere Berechnungen gleichzeitig durchzuführen, ohne dass es zu Konflikten oder Datenverlusten kommt.

Teil II

SW-Projektdokumentation

6

Anforderungsspezifikation

6.1. Funktionale Anforderungen (Use Cases)

Die funktionalen Anforderungen an die Erweiterung des Webservices sind in Form von Use Cases beschrieben. Zur detaillierteren Beschreibung der Use Cases werden die User Stories mit Akzeptanzkriterien verwendet. Nebenbei werden Szenarien und deren Akteure beschrieben, welche die Funktionalität der Applikation verdeutlichen.

6.1.1. Use Case Diagramm

Das Use Case Diagramm in Abbildung 6.1 zeigt die neuen funktionalen Anforderungen an die Applikation im Rahmen dieser Arbeit. Die Use Cases korrespondieren mit den User Stories, die im nächsten Abschnitt beschrieben werden. Farblich hervorgehoben sind die unterschiedlichen Epics (Funktionsbereiche) des Projekts in der Tabelle 6.1. Epics umfassen mehrere Use Cases, die zusammen eine grössere Funktionalität der Applikation darstellen.

Farbe	Epic
	Zersiedelung für mehrere Gebiete (Gemeinden) berechnen
	Inputdaten editierbar - Statistische Daten und Raster können angepasst werden
	Verschmelzen von Gebieten (Gemeinden) und deren Statistikdaten für die Berechnung

Tabelle 6.1: Epics der neuen funktionalen Anforderungen

Abbildung 6.1: Use Case Diagramm der neuen funktionalen Anforderungen

6.1.2. User Stories

In diesem Abschnitt sind die User Stories der neuen Funktionalitäten beschrieben. Die User Stories sind in einer Tabelle mit Titel, Story und Akzeptanzkriterien aufgelistet. Bewusst wurde auf eine Priorisierung der User Stories verzichtet, da alle Stories für die Umsetzung des Projekts relevant sind. Beschrieben sind die Akzeptanzkriterien auf Deutsch als Szenarien im Gherkin-Format (Gegeben, Wenn, Dann) [32].

- **UC-01: Gebiete als Input für Zersiedelungsberechnung auswählen**

User Story	Als Besucher der Webseite, möchte ich ein oder mehrere Gebiet(e) als Input für die Zersiedelungsberechnung auswählen, damit ich meine Berechnung auf die für mich interessanten Gebiete einschränken kann.
Akzeptanzkriterien	*Gegeben* ist, dass Website Besucher Geodaten der Gebiete hochgeladen haben, und die Gebiete einzeln auf der Landkarte auf der Startseite sieht, *wenn* die einzelnen Gebiete für die Berechnung der Zersiedelung ausgewählt werden, *dann* werden nur die ausgewählten Gebiete mit deren Statistikdaten als Input für die Berechnung gewählt. *Gegeben* ist, dass Website Besucher Geodaten der Gebiete hochgeladen haben, und die einzelnen Gebiete für die Berechnung der Zersiedelung ausgewählt haben, *wenn* die Berechnung gestartet wird, *dann* wird die Berechnung der Zersiedelungsmetriken für jedes Gebiet einzeln durchgeführt und die Resultate auf der Landkarte angezeigt.

- **UC-02: Statistikdaten pro Gebiet anzeigen**

User Story	Als Besucher der Webseite, möchte ich Statistikdaten für jedes Gebiet, dass ich hochgeladen habe, anschauen können, damit ich die Inputs der Zersiedelungsberechnung verifizieren kann, bevor ich diese starte.
Akzeptanzkriterien	*Gegeben* ist, dass Website Besucher die Vektordaten der Gebiete hochgeladen haben, *wenn* eine Region aus den Vektordaten auf der Karte angeklickt wird, *dann* werden die Statistikdaten für die angeklickte Region in einem kleinen Fenster (”Popup”) angezeigt.

- **UC-03: Statistikdaten pro Gebiet anpassen**

User Story	Als Besucher der Webseite, möchte ich Statistikdaten für jedes Gebiet, dass ich hochgeladen habe, bearbeiten können, damit ich die Inputs der Zersiedelungsberechnung ad hoc vor der Berechnung anpassen kann, um allfällige Auswirkungen auf die Metriken zu erkennen.
Akzeptanzkriterien	*Gegeben* ist, dass Website Besucher die Vektordaten der Gebiete hochgeladen haben, und die Gebiete auf der Karte angezeigt weden, *wenn* eine Region aus den Vektordaten auf der Karte angeklickt wird, *dann* können die Statistikdaten der Region über einen Button im ”Popup” bearbeitet und gespeichert werden.

- **UC-04: Bebaute Fläche (Raster) anpassen**

User Story	Als Benutzer der Webseite, möchte ich die bebaute Fläche (Raster) anpassen, um Auswirkungen von Siedlungsentwicklungen nach erneuter Berechnung untersuchen zu können.
Akzeptanzkriterien	*Gegeben* ist, dass Website Besucher das Raster (der bebauten Gebiete) hochgeladen haben, und das Raster auf der Karte angezeigt wird, *wenn* die Benutzer die Bearbeitung des Rasters über einen Button starten, und neue Pixel (Rasterpunkte) auf der Karte hinzufügen oder entfernen, *dann* wird das Raster nach dem Speichern der Änderungen aktualisiert und die Änderungen werden im Frontend zwischengespeichert und die Karte wird aktualisiert, um die Änderungen anzuzeigen.

- **UC-05: Gebietsdaten als Input für Zersiedelungsberechnung zusammenfassen**

User Story	Als Benutzer der Webseite, möche ich die Gebietsdaten als Input für die Zersiedelungsberechnung zusammenfassen, damit ich in einer Berechnung mehrere Gebiete als ein Ganzes berechnen kann, um Metriken für eine grössere Region berechnen zu können.
Akzeptanzkriterien	*Gegeben* ist, dass Website Besucher die Vektordaten der Gebiete hochgeladen haben, und mehr als ein Gebiet in den Vektordaten vorhanden ist, und die Gebiete auf der Karte angezeigt werden, und ein Button zum Zusammenfassen der Gebiete angezeigt wird, *wenn* die Strategien für die Zusammenfassung der Statistikdaten der Gebiete auswählen, und die Benutzer die Gebiete über den Button zusammenfassen, *dann* werden die Gebiete zu einem ganzen Gebiet zusammengefasst, und die Statistikdaten der Gebiete werden entsprechend der gewählten Strategie zusammengefasst, und die Karte wird aktualisiert, um das neue Gebiet anzuzeigen.

6.1.3. Personas

	Raumplaner - Markus Meierhans	**Geographin - Prof. Lisa Sommer**
Beruf	Raumplaner bei einem kommunalen Planungsamt	Geographin, Dozentin und Forscherin an einer Universität.
Technisches Wissen	Keine tiefgehenden IT-Kenntnisse. Er ist vertraut mit Geoinformationssystemen (GIS) wie QGIS, ArcGIS.	Routiniert im Umgang mit GIS-Software, räumlicher Statistik und Datenmodellierung. Sie ist es gewohnt mit komplexen Webanwendungen zu arbeiten.
Ziel	Markus möchte gezielt einzelne Gebiete auswählen können, um spezifische Planungsfragen zu analysieren. Vor der Berechnung möchte er die Statistikdaten jedes Gebiets prüfen und gegebenfalls anpassen. Er benötigt eine Funktion, die mehrere Gebiete zu einem ganzen zusammenfasst, um übergeordnete Planungsentscheidungen zu treffen. Für bestimmte geplante Bauprojekte will er deren Auswirkungen auf die Siedlungsentwicklung verstehen und simulieren.	Für wissenschaftliche Studien zur Zersiedelung und deren ökologischen Auswirkungen möchte Lisa verschiedene Gebiete analysieren. Sie möchte flexibel Gebiete auswählen können und braucht eine detaillierte Übersicht deren Statistikdaten. Lisa muss statistische Parameter bearbeiten und die bebaute Fläche anpassen, um Bauten grossflächiger Infrastrukturprojekte zu simulieren.
Erwartungen	Markus erwartet eine intuitive Oberfläche, die mit klaren Visualisierungen und Kartenansichten arbeitet. Statistikdaten sollen für jedes Gebiet einfach einsehbar und anpassbar sein. Die Berechnungen der Gebiete soll effizient erfolgen und die Resultate sollen übersichtlich dargestellt werden und einfach exportierbar (GeoJSON, Geopackage) sein.	Lisa erwartet detaillierte und präzise Metriken zur Zersiedelung und deren Export in gängigen Dateiformaten (GeoJSON, Geopackage, GeoTIFF), die sie in ihren Arbeiten verwenden kann. Sie möchte Anpassungen an der bebauten Fläche unkompliziert vornehmen können, um verschiedene Szenarien zu simulieren.

Tabelle 6.2: Personas für die funktionalen Anforderungen.

6.1.4. Szenarien

Szenario 1 - Markus Meierhans, Raumplaner
Anwendung der Webseite zur Bewertung eines neuen Wohnbauprojekts
Markus Meierhans arbeitet für das Planungsamt der Gemeinde Köniz im Kanton Bern. Die Stadtverwaltung plant ein neues Wohngebiet am Stadtrand und Markus soll untersuchen, wie sich das Bauprojekt auf die Zersiedelung der Gemeinde auswirkt. Die Daten lädt Markus in die Webapplikation hoch und wählt die relevanten Gebiete aus, die für die Berechnung der Zersiedelung berücksichtigt werden sollen. Bei der Überprüfung der hinterlegten Statistikdaten wie Bevölkerungsdichte, Flächenverbrauch und bestehende Infrastruktur stellt er fest, dass einige Werte nicht aktuell sind. Er passt die Statistikdaten manuell mit den aktuellen Zahlen der Gemeinde an und verändert die bebaute Fläche im Raster, um die Auswirkungen des geplanten Wohngebiets zu simulieren. Danach startet er die Berechnung und analysiert anschliessend die Resultate. Bei problematischen Ergebnissen, kann er alternative Entwicklungsstrategien vorschlagen, z.B. eine Verdichtung bestehender Siedlungen anstelle eines neuen Wohngebiets und diese erneut überprüfen. Die Ergebnisse der Berechnung exportiert Markus in verschiedenen Formaten (GeoJSON, GeoPackage, GeoTIFF), um sie in der Präsentation für die Gemeindeverwaltung zu verwenden.

Szenario 2 - Prof. Lisa Sommer, Geographin
Einsatz des Webtools in der Forschung und Lehre
Die Professorin untersucht die langfristigen Auswirkungen der Zersiedelung auf Ökosysteme und Mobilitätsstrukturen in den Kantonen Bern und Luzern. Für eine neue Studie möchte sie die Werte aus verschiedenen Regionen aus den genannten Kantonen vergleichen und herausfinden, welche Faktoren die Zersiedelung beeinflussen. Sie arbeitet mit Geodaten von mehreren Untersuchungsgebieten, einer ländlichen Region, einer schnell wachsenden Vorstadt und einer dichten Kernstadt. Lisa überprüft die Zersiedelungsmetriken für jedes Gebiet und passt die Statistikdaten an, um verschiedene Szenarien zu simulieren. Teilweise verändert sie die bebaute Fläche, um die Auswirkungen von geplanten Infrastrukturprojekten zu untersuchen. Die Ergebnisse der Berechnungen exportiert sie in verschiedenen Formaten (GeoPackage für Statistikdaten, GeoTIFF für das SI-Raster), um sie in ihrer Publikation über nachhaltige Siedlungsentwicklung zu verwenden. Als Dozentin möchte sie das Webtool auch in ihren Lehrveranstaltungen einsetzen, um Studierenden die Zersiedelungsthematik näherzubringen.

6.2. Nicht-Funktionale Anforderungen

In den folgenden Tabellen sind die für das Projekt relevanten, nicht-funktionalen Anforderungen (NFR, non-functional requirements) aufgelistet. Die Anforderungen sind nach dem ISO-Standard ISO / IEC 25010:2024 [33] definiert.

Mit der höchsten Priorität werden die Anforderungen NFR-01, NFR-02 und NFR-05 umgesetzt. Diese dienen den Grundsätzen der Freien Software, des Datenschutzes und der Korrektheit der Berechnung.

6.2.1. NFR-01: Open Source

ID	**NFR-01**
Thema	Der Quellcode steht offen zur Verfügung. Die Applikation kann von der Open Source Community weiterentwickelt, abgewandelt oder optimiert werden.
Anforderung	Wartbarkeit - Open Source, Erweiterbarkeit
Kriterien	• Der Quellcode der Applikation ist auf einem öffentlichen Git-Repository verfügbar. • Eine saubere und verständliche Dokumentation zur Bearbeitung des Projekts soll über das öffentliche Internet in Form einer Readme-Datei aufrufbar sein. • Die Applikation ist unter einer Open Source Lizenz veröffentlicht. • Mittels einer verständlichen Clean-Code Architektur wird die Wartbarkeit und Erweiterbarkeit des Codes sichergestellt. • Die Trennung der Schichten (Frontend und Backend) wird durch eine klare Strukturierung des Codes und der Verwendung von Schnittstellen (APIs) gewährleistet.
Verifikation	Das Projekt wurde unter der GNU General Public License v3.0 [34] veröffentlicht und ist auf dem GitLab der OST unter `https://gitlab.ost.ch/sa-urban-sprawl-metrics/usm-calculator` verfügbar. Das Repository ist öffentlich und enthält eine Readme-Datei, welche die Installation und Verwendung der Applikation beschreibt. Die Trennung der Schichten (Frontend und Backend) ist gewährleistet und im Kapitel 9.1 beschrieben.

6.2.2. NFR-02: Sicherheit und Datenschutz

ID	**NFR-02**
Thema	Daten werden verschlüsselt übertragen und nach ihrer Verarbeitung auf dem Server wieder gelöscht. Nach Abschluss der Session des Benutzers werden die hochgeladenen und zwischengespeicherten Daten wieder entfernt.
Anforderung	Funktional
Kriterien	• Temporär angelegte Dateien (z.B. Rasterdaten, Vektordaten, ”Shapefiles” für Rasterausschnitte) werden nach Abschluss der Berechnung innerhalb von 4 Stunden gelöscht. • Die Applikation verwendet HTTPS für die sichere Übertragung der Daten zwischen Client und Server.
Verifikation	Alle von der Applikation hochgeladenen Daten werden innerhalb von 4 Stunden nach Start der Berechnung gelöscht. Kapitel 8 erwähnt die Datenschutzmassnahmen, die in der Applikation implementiert sind. Die Applikation sieht den Einsatz von HTTPS vor, um die Datenübertragung zwischen Client und Server zu sichern.

6.2.3. NFR-03: Wartbarkeit

ID	NFR-03
Thema	Die Wartbarkeit wird gewährleistet durch eine sinnvolle Trennung von Front- und Backend. Ausserdem sind die einzelnen Codekomponenten in Klassen und Pakete sinnvoll unterteilt. Die Skalierbarkeit und die Erweiterbarkeit der Applikation wird durch die Modularität sichergestellt.
Anforderung	Wartbarkeit - Modularität
Kriterien	• Frontend und Backend werden getrennt entwickelt und kommunizieren über eine REST API. • Die beiden Schichten werden in einem Monorepository verwaltet, um die Wartbarkeit zu erhöhen. • Die Codekomponenten sind in Klassen und Pakete sinnvoll gruppiert, um die Lesbarkeit und Wartbarkeit zu erhöhen. • Komponenten und Module in Typescript und Python folgen dem Single Responsibility Principle (SRP), um die Funktionalitäten zu isolieren. • Mit regelmässigen Code-Reviews wird die Modularität mit Ziel des Decouplings sichergestellt.
Verifikation	Die Architektur der Applikation ist im Abschnitt 9.1 beschrieben. Eine Trennung von Frontend und Backend wurde anhand der Ausgangslage der Applikation in einem Monorepository umgesetzt. Die Codekomponenten sind sinnvoll in Klassen und Pakete gruppiert, um die Lesbarkeit und Wartbarkeit zu erhöhen.

6.2.4. NFR-04: Usability

ID	NFR-04
Thema	Eine intuitive Benutzeroberfläche unterstützt die Benutzer der Applikation bei der korrekten Eingabe von Inputdaten für die Berechnung. User Experience Best Practices werden bestmöglich umgesetzt. Eingabefelder werden erklärt und fehlerhafte Eingaben vermieden. Die API-Schnittstelle der Applikation soll nach OpenAPI Spezifikationen ausführlich dokumentiert werden.
Anforderung	Usability
Kriterien	• Die neuen Funktionalitäten sind in der Benutzeroberfläche klar strukturiert und intuitiv bedienbar. • Die Oberfläche ist responsiv und passt sich an verschiedene Desktop-Bidschirmgrössen an. • Die API-Schnittstelle ist gemäss OpenAPI Spezifikationen dokumentiert und ist durch Beispielanfragen und -antworten verständlich. • Eine einheitliche UI-Komponentenbibliothek (Ant Design Vue) wird verwendet, um eine konsistente Benutzererfahrung zu gewährleisten.
Verifikation	Diese nicht-funktionale Anforderung wird durch eine Implementierung der Benutzeroberfläche mit einer modernen UI-Komponentenbibliothek (Ant Design Vue) umgesetzt. Die Benutzeroberfläche ist responsiv und passt sich an verschiedene Bildschirmgrössen an. Die API-Schnittstelle ist gemäss OpenAPI Spezifikationen dokumentiert und enthält Beispielanfragen und -antworten. Ein Usability-Test wurde durchgeführt (vg. Abschnitt 10.7) und die Erkenntnisse daraus wurden in das UI-Design übernommen.

6.2.5. NFR-05: Exaktheit des Rasters

ID	NFR-05
Thema	Das Raster wird unter der Anwendung von Leaflet-Layern möglichst exakt dargestellt. Nach dem Bearbeiten des Rasters auf der Karte, fliessen die Änderungen in die Berechnung mit ein. Benutzer der Applikation können sich auf die Korrektheit der Darstellung verlassen.
Anforderung	Funktional
Kriterien	• Mit den Angaben des Rasters (Pixelgrösse von 15m x 15m) wird die Berechnung der Zersiedelung durchgeführt. • Um die Korrektheit der Berechnung zu gewährleisten müssen die Pixel des Rasters, in die abhängig des Breitengrades korrekten UTM-Koordinaten (EPSG:32600 - nördliche Hemisphäre, EPSG:32700 - südliche Hemisphäre), umgewandelt werden. • So soll sichergestellt werden, dass mit jeglichen Koordinatensystemen gearbeitet werden kann und die Berechnung der Fläche (in m^2) korrekt durchgeführt wird.
Verifikation	Das Raster wird in der Applikation mit Leaflet-Layern dargestellt und die Pixelgrösse von 15m x 15m wird eingehalten. Die Koordinaten des Rasters werden in die entsprechenden UTM-Koordinaten umgewandelt, um die Korrektheit der Berechnung zu gewährleisten (vgl. Abschnitt 10.5.1).

7
Analyse

7.1. Berechnungsschritte

Die einzelnen Schritte des Algorithmus zur Berechnung der Zersiedelung sind in Abbildung 7.1 dargestellt. Schwick et al. haben einzelne Metriken und deren Abhängigkeiten im Buch ”Zersiedelung messen und verstehen” detailliert beschrieben [3]. Die Berechnungsschritte sind ebenfalls im Benutzerhandbuch des ”USM Toolsets”, ein Plugin für QGIS, beschrieben [22].

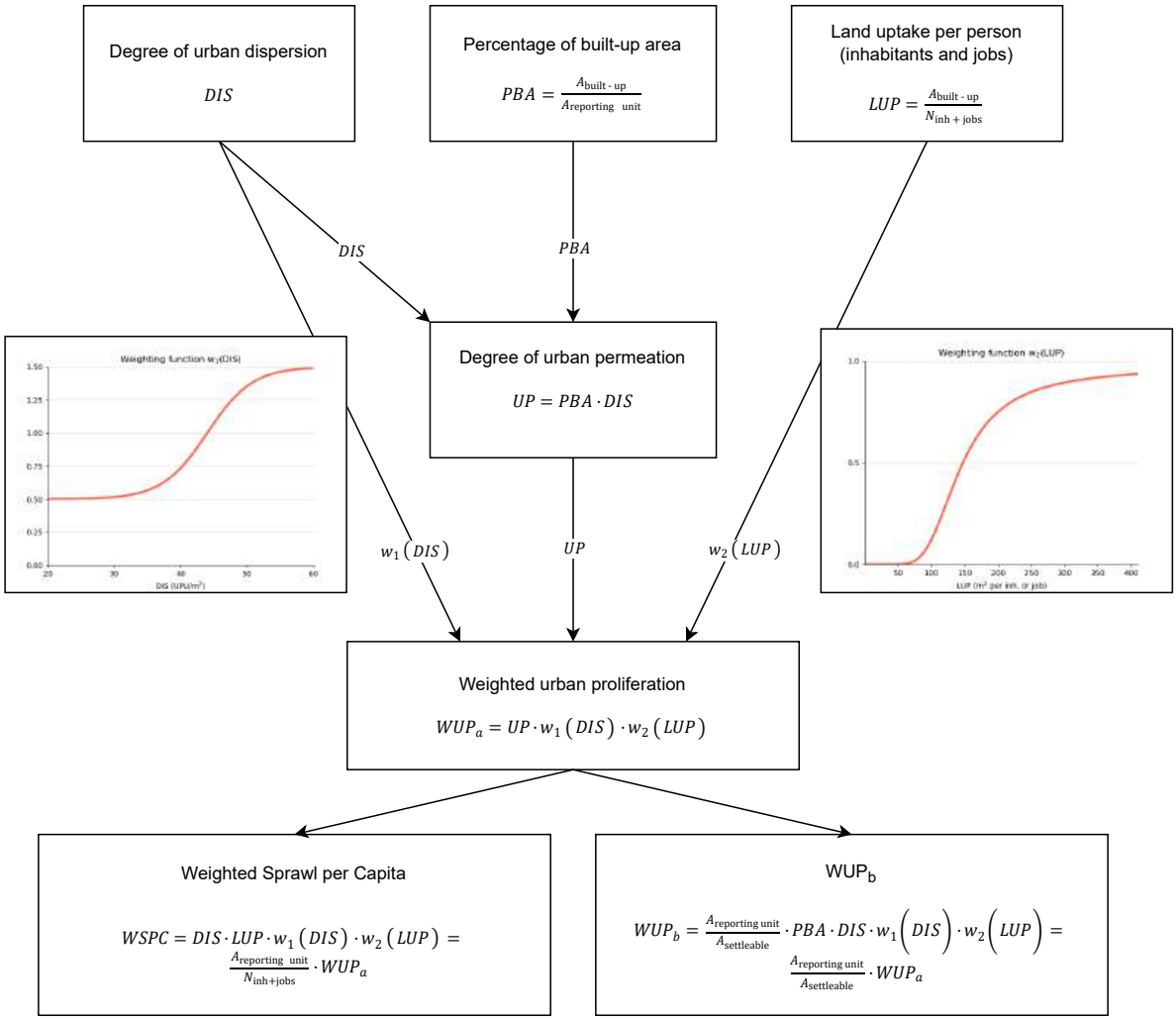


Abbildung 7.1: Berechnungsschritte und Abhängigkeiten der Metriken der Zersiedelung nach Schwick et al. [3] und USM Toolset Manual [22]. Darstellung aus der Studienarbeit [8].

Die Berechnung der Zersiedelung erfolgt anhand von Metriken, die in Abbildung 7.1 dargestellt sind und kann wie folgt zusammengefasst werden:

Als Grundlage dienen die Metriken DIS (Degree of urban dispersion), PBA (Percentage of built-up area) und LUP (Land uptake per person), aus Abbildung 7.2. Die Metrik DIS beschreibt die Streuung von bebauten Gebieten anhand der Distanzen zwischen zwei Punkten innerhalb des bebauten und zu untersuchenden Gebiets. DIS wird in der Einheit UPU (Urban Permeation Units) pro Quadratmeter der bebauten Fläche angegeben. PBA definiert den Anteil der bebauten Fläche in Prozent und LUP die Flächeninanspruchnahme pro Person als Fläche in Quadratmetern pro Person.

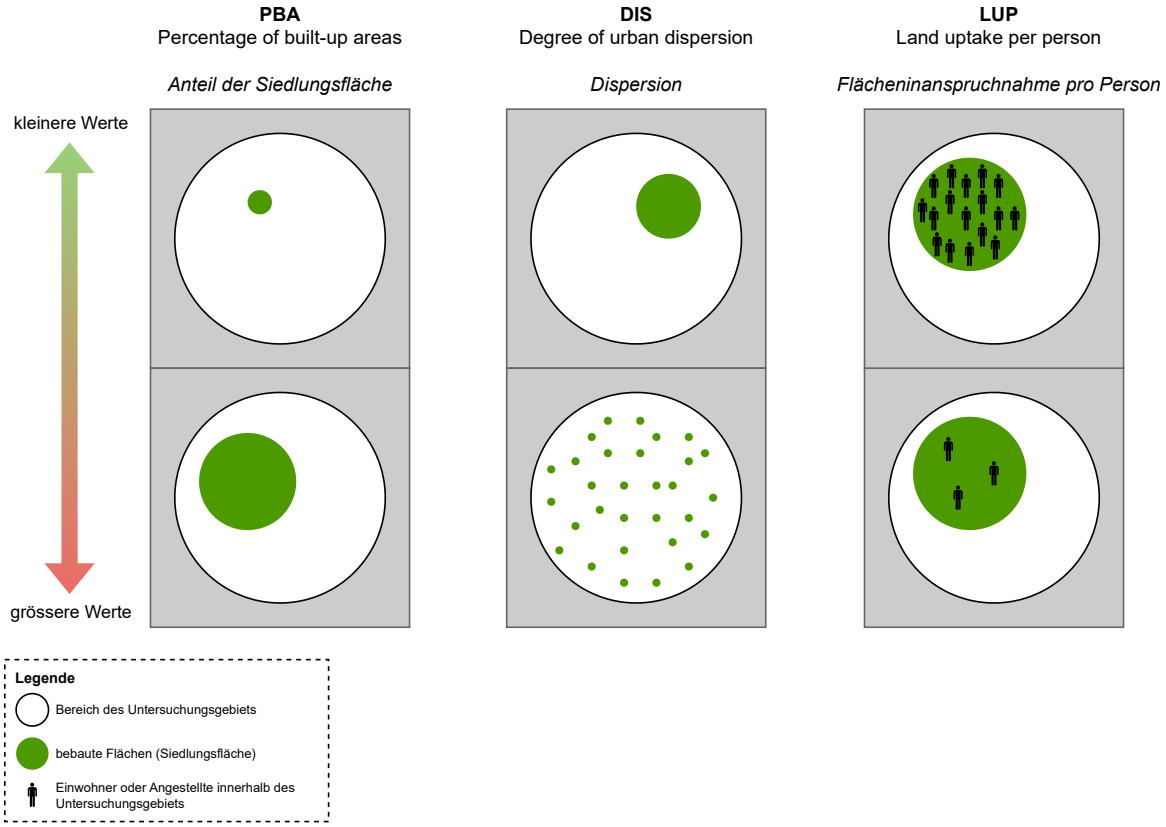


Abbildung 7.2: Wie sich die Bestandteile PBA, DIS und LUP von Z in der Realität widerspiegeln.

Die Metrik UP (Urban Permeation) lässt sich anschliessend aus dem Produkt von DIS und PBA ableiten. UP bezeichnet die Durchdringung einer Landschaft durch Siedlungsflächen bzw. bebaute Flächen und wird in urbanen Durchdringungseinheiten pro Quadratmeter der Landschaftsfläche (UPU/m^2) angegeben.

Der Wert der Zersiedelung (Z = Weighted Urban Proliferation WUP) wird aus dem Produkt der Metriken UP, dem gewichteten DIS-Wert (WDIS) und der gewichteten Metrik LUP berechnet. Angegeben wird der Wert in urbanen Durchdringungseinheiten pro Quadratmeter der Landschaft (UPU/m^2).

Zwei weitere Metriken, die abgeleitet werden können, sind die Metrik WSPC (Weighted Sprawl Per Capita) und WUP_b (Weighted Urban Proliferation of the settleable part of the study area). WSPC misst den Beitrag jeder Person und Arbeitsstelle zur Zersiedelung in dem untersuchten Gebiet. WUP_b verfeinert den Wert der Zersiedelung, indem nur die Siedlungsfläche berücksichtigt wird, die als bebaubar gilt.

Die Funktionen der Gewichtungen der Metriken DIS und LUP sollen die Zersiedelung in Gebieten mit unterschiedlichen Dichtungen und Nutzungen besser abbilden.

7.1.1. Metriken der Zersiedelung

Die Metriken der Zersiedelung aus der Abbildung 7.1 sind in den nachfolgenden Tabellen genauer beschrieben. Bereits in der Studienarbeit [8] wurden die Metriken detailliert beschrieben und werden ergänzend in dieser Arbeit erneut aufgeführt. Sämtliche aufgeführten Metriken lassen sich mit dem "USM Calculator" Webservice und dessen API berechnen und können Aufschluss über die Zersiedelung in einem Gebiet geben.

DIS

Name	Degree of Urban Dispersion *Dispersion*
Beschreibung	Die Metrik quanitfiziert die Streuung von bebauten Gebieten anhand der Distanzen zwischen zwei beliebigen Punkten, die innerhalb des bebauten Gebietes liegen. Die Grundlage des DIS wird in dieser Arbeit als SI-Raster (SI = Sprawl at Index) bezeichnet. Dieses Raster beinhaltet für jeden Pixel (der bebauten Fläche) die gemittelte Summe der Distanzen (Euklidischer Abstand [35]) zu den nächsten bebauten Flächen innerhalb eines Radius von 2 km. DIS wird in "Urban Permeation Units" (UPU) pro Quadratmeter der bebauten Fläche (UPU/m^2) ausgedrückt. Je verstreuter die bebauten Gebiete sind, desto grösser ist der DIS-Wert. Kompakter bebaute Gebiete haben folglich niedrigere DIS-Werte als stärker zerstreut bebaute Gebiete.
Formel [3]	$DIS(b) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{n_i}\left(\sum_{j=1}^{n_i}\left(\sqrt{\frac{2\cdot d_{ij}}{1m}+1}-1\right)\frac{DSE}{m^2}+ZIB(b)\right)$ $ZIB(b)$ bezeichnet dabei den Zellinnenbeitrag als Funktion der Zellengrösse b. n ist die Anzahl aller Siedlungszellen (der Grösse b). n_i ist die Anzahl der Siedlungszellen innerhalb des Radius (auch Beobachtungshorizont) um Siedlungszelle i. d_{ij} ist der euklidische Abstand zwischen den Siedlungszellen i und j. DSE sind Durchsiedelungseinheiten = UPU. Wenn $0 < b < 1000m$ kann $ZIB(b)$ durch die folgende Formel angenähert berechnet werden: $ZIB(b) = \left(\sqrt{0.97428\cdot\frac{b}{1m}+1.046}-0.996249\right)\frac{DSE}{m^2}$
Einheit	UPU/m^2 der bebauten Fläche

PBA

Name	Proportion of Built-up Areas *Anteil der Siedlungsfläche (ASF)*
Beschreibung	Diese Metrik ist das Verhältnis zwischen der Grösse der bebauten Flächen und der Grösse des Untersuchungsgebiets.
Formel	$PBA = A_{built-up}/A_{reporting-unit}$
Einheit	%

LUP

Name	Land Uptake per person *Flächeninanspruchnahme pro Person (Fl)*
Beschreibung	Die Fläche, welche pro Einwohner oder Arbeitsplatz innerhalb der bebauten Gebiete genutzt und in m^2 pro Einwohner oder Arbeitsplatz ausgedrückt wird. Hohe LUP-Werte deuten darauf hin, dass pro Einwohner oder Arbeitsplatz mehr Fläche genutzt wird als in Gebieten mit niedrigeren Werten. LUP ist der Kehrwert von UD: $LUP = 1/UD$
Formel	$LUP = A_{built-up}/N_{inh+job}$
Einheit	m^2 pro Einwohner oder Arbeitsplatz

UD

Name	Utilization Density *Ausnützungsdichte*
Beschreibung	Die Anzahl Einwohner und Arbeitsplätze pro km^2 der bebauten Fläche. Je mehr Einwohner und Arbeitsplätze pro Fläche, desto höher ist die Ausnützungsdichte.
Formel	$UD = N_{inh+job}/A_{built-up}$
Einheit	Einwohner oder Arbeitsplatz pro km^2

UP

Name	Urban Permeation *Urbane Durchdringung*
Beschreibung	Das Mass für die Durchdringung einer Landschaft durch bebaute Gebiete. Es berücksichtigt DIS und PBA und wird in UPU pro m^2 des Untersuchungsgebiet angegeben.
Formel	$UP = PBA \cdot DIS$
Einheit	UPU/m^2 des Untersuchungsgebiets

WUP

Name	Weighted Urban Proliferation *Zersiedelung (Z), Gewichtete Zersiedelung, WUP_a*
Beschreibung	Die wichtigste Kennzahl zur Quantifizierung der Zersiedelung. Sie ist das Produkt aus der Urban Permeation (UP), des gewichteten DIS $w_1(DIS)$ und des gewichteten LUP $w_2(LUP)$. $w_1(DIS)$ ist eine Gewichtungsfunktion für DIS, die Funktionswerte zwischen 0,5 und 1,5 ausgibt, um den stärker dispersen Gebieten ein höheres Gewicht und den weniger dispersen ein geringeres Gewicht zu geben. $w_2(LUP)$ ist eine Gewichtungsfunktion für LUP, die Funktionswerte zwischen 0 und 1 ausgibt, um intensiver genutzte städtische Gebiete, d. h. solche mit mehr Einwohnern und Arbeitsplätzen, geringer zu gewichten.
Formel	$WUP = UP \cdot w_1(DIS) \cdot w_2(LUP)$ Wobei für die Gewichtungsfunktionen gilt: $w_1(DIS) = 0.5 + \frac{e^{0.294432 \cdot DIS - 12.955}}{1+e^{0.294432 \cdot DIS - 12.955}}$ und $w_2(LUP) = \frac{e^{4.159 - 613.125/LUP}}{1+e^{4.159 - 613.125/LUP}}$
Einheit	UPU/m^2 des Untersuchungsgebiets

WUP_b

Name	Weighted Urban Proliferation of the settleable part of the study area *Gewichtete Zersiedelung mit Berücksichtigung der besiedelbaren Flächen eines Referenzgebiets*
Beschreibung	Die Zersiedelung kann mit und ohne Einbeziehung der für den Bau von Gebäuden ungeeigneten Gebiete des Untersuchungsgebiets gemessen werden. Beispiele für solche Gebiete, die sich nicht für den Bau von Gebäuden eignen, sind Gletscher und ewiger Schnee, Wasserläufe, Seen und andere Gewässer, Küstenlagunen, Flussmündungen, Binnensümpfe und Torfmoore. Gebiete, in denen der Bau von Gebäuden nicht erlaubt ist, könnten ebenfalls ausgeschlossen werden, z.B. Naturschutzgebiete in der Schweiz. Schliesst man diese Gebiete aus, führt dies zu grösseren WUP-Werten [22].
Formel	$WUP_b = \frac{A_{reporting-unit}}{A_{settleable}} \cdot WUP$ Wobei $A_{settleable}$ die Fläche des besiedelbaren Teils des Untersuchungsgebiets ist.
Einheit	UPU/m^2 des Untersuchungsgebiets

WSPC

Name	Weighted Sprawl per Capita *Gewichtete Zersiedelung pro Kopf*
Beschreibung	Diese Zahl misst den Beitrag jedes Einwohners oder Arbeitsplatzes zur Zersiedelung im Untersuchungsgebiet.
Formel	$WSPC = \frac{A_{reporting-unit}}{N_{inh+jobs}} \cdot WUP$
Einheit	UPU pro Einwohner oder Arbeitsplatz

TS

Name	Total Sprawl *Gesamtdurchsiedelung*
Beschreibung	Die gemittelte Summe der gewichteten Distanzen zwischen den bebauten Flächen innerhalb des Untersuchungsgebiets. Der Wert lässt sich aus dem Produkt des DIS und der bebauten Fläche berechnen.
Formel	$TS = DIS \cdot A_{built-up}$
Einheit	UPU

7.1.2. Definition der Siedlungsfläche

Die Definition der Siedlungsfläche ist für die Berechnung der Zersiedelung von zentraler Bedeutung. Im Artikel "Ein neuer Geodatensatz für das Montioring der Siedlungsentwicklung" des Fachmagazins "Geomatik Schweiz" wird die Definition genauer erläutert und es können grundsätzlich alle Orte, an denen Menschen sich längerfristig aufhalten, als Siedlung angesehen werden [4]. Eine genauere Bestimmung hängt jedoch vom jeweiligen Kontext ab, wie [4] argumentiert.

In dieser Arbeit wird mit einem entsprechenden Datensatz gearbeitet, der vom Bundesamt für Raumentwicklung (ARE) öffentlich zur Verfügung gestellt wird [36]. Dieser Datensatz wird seit 2019 bis 2024 jährlich aktualisiert und beinhaltet die Siedlungsfläche der Schweiz als Vektor-Polygon. Das Koordinatenreferenzsystem (CRS) des Datensatzes ist CH1903+ / LV95 (EPSG:2056) [37]. Mittels QGIS lässt sich daraus z.B. die bebaute Fläche als Raster ableiten, indem das Werkzeug "Rasterisieren" verwendet wird [38] (vgl. Abbildungen 7.3 und 7.4).

Es empfiehlt sich für die Berechnung ein Raster mit einer quadratischen Pixelgrösse von 15 Metern zu verwenden, weil der Algorithmus für diese Pixelgrösse optimiert ist.

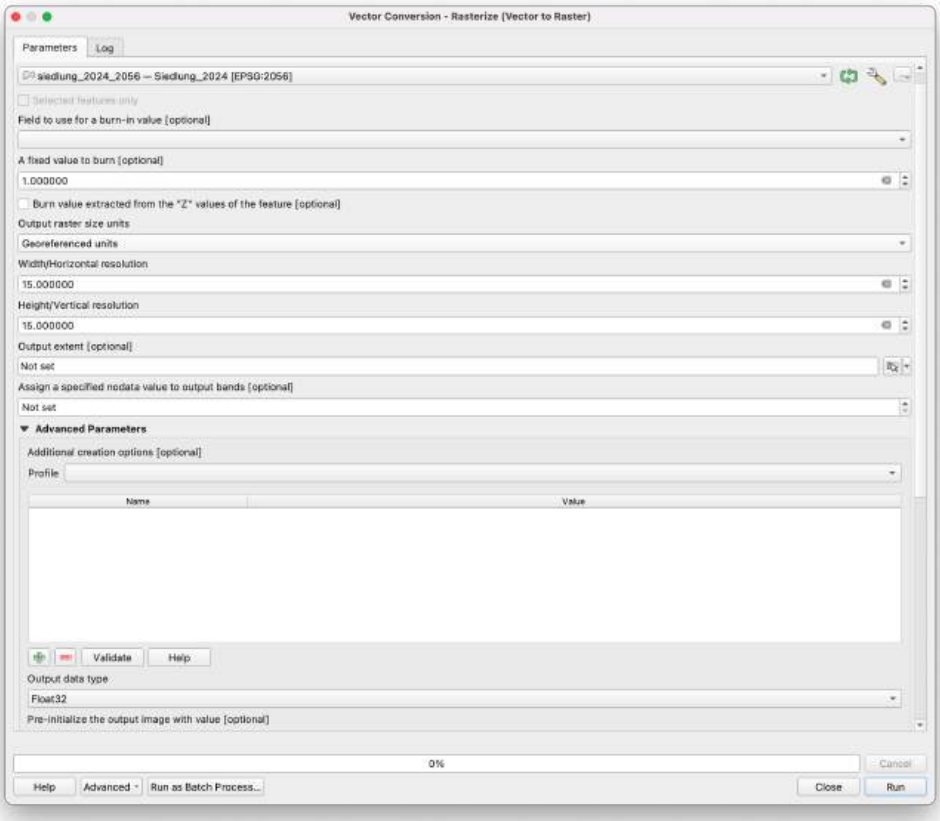
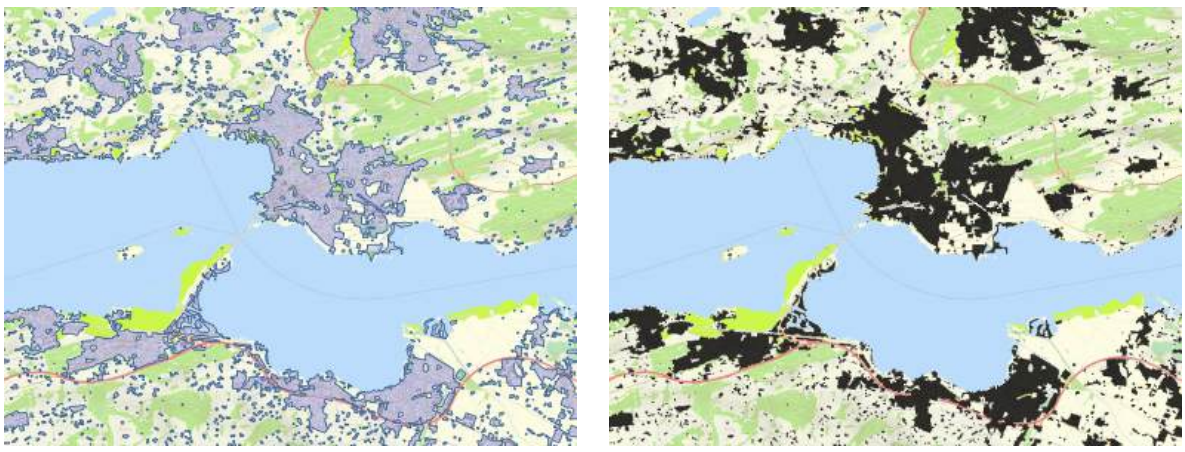


Abbildung 7.3: Rasterisierung der Siedlungsfläche in QGIS.



(a) Die Siedlungsfläche als Vektor-Polygon.

(b) Nach der Rasterisierung mit QGIS.

Abbildung 7.4: Die Siedlungsfläche der Schweiz als Vektor-Polygon und Raster (Ausschnitt Region Oberer Zürichsee).

8

Sicherheit

8.1. Allgemein

Die Gewährleistung der Datensicherheit ist von zentraler Bedeutung für die Erweiterung. Statistikdaten der GeoJSON- und GeoPackage-Dateien, die zur Zersiedelungsberechnung verwendet werden, könnten heikle Informationen enthalten, die nicht ohne Erlaubnis des Benutzers weitergegeben werden dürfen. Einige Statistikdaten könnten, wenngleich nicht direkt personenbezogen, Rückschlüsse auf die Identität von Personen zulassen.

Auch wenn nicht alle Attribute der hochgeladenen Geodaten für die Berechnung der Zersiedelung benötigt werden, wird aus praktischen Gründen der gesamte Inhalt hochgeladen und verarbeitet.

8.2. Persistenz im Webserver

Der USM Calculator speichert während der Laufzeit der Berechnung Daten in einem temporären Verzeichnis des Webservers. Um die Begrenzung eines Rasters zu ermöglichen, wird vom hochgeladenen Vektorpolygon eine Kopie im Shapefile-Format (`.shp`) erstellt. Aus technischen Gründen wird dieses Polygon (Shapefile) verwendet, um das Raster entlang der Shape-Grenzen zuzuschneiden, worauf dieser Ausschnitt des Rasters ebenfalls für die Dauer der Berechnung abgelegt wird (`.tif`). Speziell diese beiden Dateien werden unmittelbar nach der Berechnung gelöscht.

Für jede Berechnung wird ein eindeutiger Bezeichner (UUID) generiert, der als Schlüssel für das Auslesen der Resultate dient. Dabei werden als Teil des Resultates alle erstellten SI-Raster (`.tif`) in einem Verzeichnis abgelegt, wobei die Dateinamen eigene UUIDs beinhalten. Die Dateipfade der SI-Raster und die Resultate der Berechnung werden mit einem EXPIRY-Zeitstempel versehen und in einem Valkey-Store für vier Stunden gespeichert. Der eindeutige Bezeichner (UUID) wird dem Benutzer als Teil der Antwort auf die Anfrage zurückgegeben, damit dieser die Resultate später abrufen kann.

Es wird mit einem Cron-Job sichergestellt, dass jede Stunde alle temporären Dateien (SI-Raster), die älter als vier Stunden sind, gelöscht werden. Es kann garantiert werden, dass die Dateien in keinem Fall auf dem Server verbleiben.

Von einer langfristigen Speicherung der Daten, z.B. in einer Datenbank, wurde abgesehen, um unter anderem den Datenschutz besser zu gewährleisten und um den Umfang der Applikation im Rahmen des Projekts zu begrenzen.

8.3. Verantwortlichkeit

Es ist wichtig zu betonen, dass die Verantwortung für die Daten, die in der Applikation verwendet werden, bei den Benutzern liegt, weil diese die Eingabedaten bereitstellen.

8.4. Missbrauch der API

Es sollte beachtet werden, dass für einen öffentlichen Einsatz des Webservices eine Authentifizierung über einen API-Token empfohlen wird, um Missbrauch zu verhindern. Mögliche DDoS-Angriffe könnten so verhindert werden, indem die Anzahl der Anfragen pro Token begrenzt werden könnten. Ausserdem können Anfragen grosse Datenmengen beinhalten, die den Server überlasten könnten. Die Implementierung möglicher Sicherheitsmassnahmen, ist jedoch nicht Teil dieser Arbeit, wird aber ergänzend erwähnt und soll in einem praktischen Einsatz berücksichtigt werden.

9
Design

9.1. Architektur

Die Softwarearchitektur wird nach Robert C. Martins ”Clean Architecture” [39] umgesetzt. Das Architekturmuster zielt darauf ab, Abhängigkeiten in einem Schichtmodell aufzubauen, wobei die äusseren Schichten von den inneren abhängig sind. Die Geschäftslogik (auch Business Logic) soll von der Infrastruktur getrennt werden (vgl. Abbildung 9.1). In der Clean Architektur wird eine wichtige Abgrenzung (auch ”Boundary”) definiert: Die `api` Boundary. Diese soll das Vue.js Frontend vom Python Backend konzeptionell trennen. Um zwischen den Schichten zu kommunizieren, wird eine REST-API verwenden, die im Backend implementiert ist. Aufgrund dieser Abgrenzung können die Schichten unabhängig voneinander entwickelt werden.

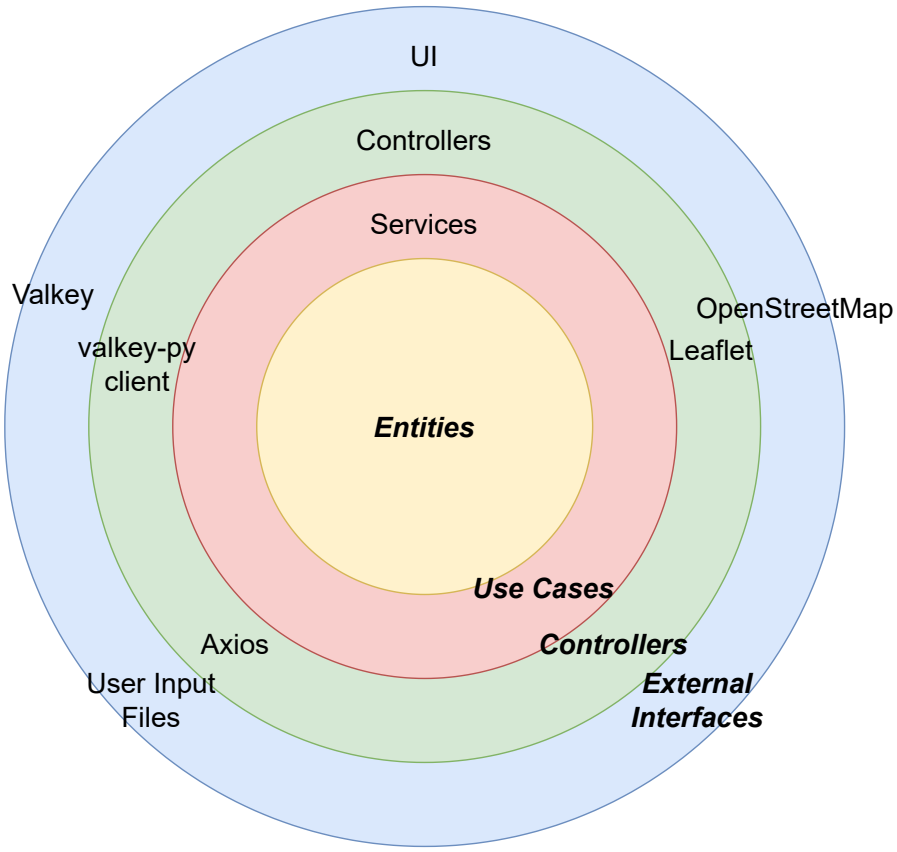


Abbildung 9.1: Clean Architecture nach [39] in eigener Darstellung.

9.2. Technologien und Designentscheidungen

Das Backend baut auf einer bestehenden REST-API auf, die mit dem Python Web-Framework Starlette umgesetzt wurde. Für die Wahl von Python als Programmiersprache sprechen die bestehenden Abhängigkeiten und die vorhandene Codebasis. Der Umfang an verfügbaren Bibliotheken für die Geodatenverarbeitung war ein weiterer Grund, dass Python als Programmiersprache in der Studienarbeit gewählt wurde.

Im Frontend wird mit Typescript und dem Vue.js Framework gearbeitet. Vue.js eignet sich optimal für die Entwicklung von Single-Page-Applications (SPA) und bietet eine moderne und flexible Architektur. In der Studienarbeit wurde die neue Art der Komponentenentwicklung mit Vue 3 und dem Composition API [40] verwendet, die eine flexible und modulare Entwicklung ermöglicht. Ausserdem wurde der Ansatz der Single-File-Components (SFC) verfolgt, um die HTML, CSS und Typescript Logik in einer Datei zu kapseln [41]. An diesen Entscheidungen aus der Studienarbeit wird nach einer erneuten Evaluation festgehalten, da sich diese als geeignet für die Entwicklung der neuen Funktionalitäten durch die gute Modularität und Flexibilität erwiesen haben.

Im Rahmen dieser Arbeit wurde die Berechnung der Zersiedelung im Backend um die Möglichkeit erweitert, mehrere Untersuchungsgebiete parallel zu berechnen. Dies geschieht mithilfe der Python-Bibliothek Joblib [14], die sich in einer empirischen Analyse als die am besten geeignete Bibliothek für die Parallelisierung der Berechnungen herausgestellt hat. Joblib konnte die Berechnungsdauer im Optimalfall um das Doppelte reduzieren, was die Performance, vor allem für grössere Untersuchungsgebiete, signifikant verbessert.

Die Parallelisierung der Berechung wird in einem separaten Background-Task ausgeführt, um den Status der Berechnung zu verfolgen und die Benutzeroberfläche nicht zu blockieren. Während der Berechnung wird laufend ein Zwischenstand in einem Valkey [42] in-memory Key-Value-Store gespeichert. Gleichzeitig wird über den Pub-Sub-Mechanismus von Valkey [43] in Kombination mit dem Websocket Protokol [44] der Status der Berechnung an das Frontend übermittelt. Die Berechnungsergebnisse werden im Valkey gespeichert, um die Ergebnisse der Berechnung bei deren Abschluss im Frontend anzeigen zu können. Nach vier Stunden werden die Informationen im Valkey-Store gelöscht.

9.2.1. Tools

Die in der Studienarbeit verwendeten Tools werden für die Erweiterungen übernommen. Nachfolgend sind die wichtigsten Frameworks und Tools aufgelistet, die für die Umsetzung der gesamten Applikation verwendet werden. Im Rahmen der Erweiterung wurden die Abhängikeiten aktualisiert und auf die neuesten Versionen upgedated.

Bestehende Tools

- **Python** [10]: Eine interpretierte, objektorientierte Programmiersprache.
- **GDAL** [25]: Eine Open-Source-Bibliothek für Geodatenverarbeitung, bietet viele Funktionen für die Verarbeitung von Raster- und Vektordaten an.
- **Starlette** [11]: Ein leichtgewichtiges Web-Framework für Python, das auf ASGI basiert.
- **Numba** [27]: Eine Bibliothek zur Beschleunigung von Python-Code durch Just-in-Time-Kompilierung und Funktionen zur Parallelisierung.
- **Joblib** [14]: Eine Bibliothek zur Parallelisierung von Code in Python.
- **Numpy** [26]: Eine Bibliothek für numerische Berechnungen mit grossen, mehrdimensionalen Arrays.
- **Uvicorn** [45]: Ein ASGI-Server.
- **Pytest** [46]: Ein Framework für das Testen von Python-Code.
- **Poetry** [47]: Ein Tool zur Verwaltung von Python-Abhängigkeiten und virtuellen Umgebungen.
- **Typescript** [48]: Typensichere Programmiersprache, die zu Javascript kompiliert wird.
- **Vite** [49]: Ein Build-Tool und Development-Webserver für moderne Webanwendungen, das auf Rollup, ein Javascript Modul Bundler, basiert.
- **Vue.js** [16]: Ein populäres JavaScript-Framework zur Entwicklung von reaktiven Frontends.

- **Leaflet** [17]: Eine Open-Source-JavaScript-Bibliothek zur Darstellung interaktiver Karten. Konkret wird die Erweiterung "Vue Leaflet" [50] verwendet, die mit Vue.js verwendet werden kann.
- **Ant Design** [24]: Eine React-Komponentenbibliothek, für die es auch eine Vue-Version gibt.
- **Axios** [51]: Ein Promise-basierter HTTP-Client für den Browser und Node.js. Axios ist als Ersatz für Fetch-API in modernen Browsern gedacht und wird ausdrücklich für Vue.js empfohlen.
- **Turf.js** [52]: Eine JavaScript-Bibliothek für Geospatial-Analysen, die viele Funktionen für die Verarbeitung von Geodaten bietet. Turf.js wurde im Frontend verwendet, um die Mittelpunkte der Polygone der Vektordatei zu berechnen.
- **d3.js** [53]: Eine JavaScript-Bibliothek für Datenvisualisierung, die viele Funktionen für die Erstellung von interaktiven Diagrammen und Grafiken bietet. Die Bibliothek wurde eingesetzt um das SI-Raster farblich darzustellen.
- **georaster** [54]: Eine JavaScript-Bibliothek, die das Lesen von Rasterdaten ermöglicht und diese Daten in der Leaflet Karte visuell darstellen kann.
- **Flake8** [55]: Ein Tool zur statischen Codeanalyse in Python, das die Einhaltung des PEP8-Standards [56], ein Style Guide für Python, überprüft.
- **Pylint** [57]: Ein weiteres Tool zur statischen Codeanalyse in Python, das die Codequalität überprüft.
- **Mypy** [58]: Ein statischer Typenchecker für Python.
- **Eslint** [59]: Ein Linter für Typescript, um die Codequalität und Konsistenz zu gewährleisten.
- **QGIS** [6]: Ein Open-Source Geoinformationssystem, das für die Verarbeitung von Geodaten verwendet wird. Unterschiedliche Plugins erweitern die Funktionalität von QGIS. QGIS wurde in diesem Projekt verwendet, um Testdaten zu generieren und die Resultate der Berechnungen zu prüfen.

Ergänzte Tools
Ergänzt wird die Liste um die folgenden Tools, die für die Umsetzung der neuen Anforderungen benötigt werden:

- **Valkey** [42]: Ein in-memory Key-Value-Store, der als Open-Source-Alternative zu Redis entwickelt wurde. Valkey wurde für die Speicherung der Berechnungsergebnisse während der Berechnung verwendet.
- **valkey-py** [60]: Ein Python-Client für Valkey, der die Kommunikation mit dem Valkey-Server ermöglicht.
- **cProfile** [61]: Ein integriertes Profiling-Tool in Python, das die Performance von Python-Code analysiert und die Ausführungszeit von Funktionen misst.
- **gprof2dot** [62]: Ein Tool zur Visualisierung von Profiling-Dateien, das die Ausgabe von Profiling-Tools wie cProfile in ein Graphviz-Diagramm umwandelt.
- **graphviz** [63]: Ein Tool zur Visualisierung von Graphen, das für die Darstellung von Profiling-Daten verwendet wird.
- **memory-profiler** [64]: Ein Tool zur Analyse des Speicherverbrauchs von Python-Code, das für die Optimierung der Performance der Berechnungen verwendet wird.
- **geoblaze** [65]: Eine JavaScript-Bibliothek, die es ermöglicht, Geodaten in Webanwendungen zu verarbeiten und zu analysieren. Diese Bibliothek wurde verwendet, um die Rasterdaten im Frontend zu bearbeiten.
- **geotiff.js** [66]: Eine JavaScript-Bibliothek, die es ermöglicht, GeoTIFF-Rasterdaten im Browser zu lesen und zu verarbeiten. Sie wurde verwendet um die GeoTIFF-Rasterdaten nach der Bearbeitung im Frontend zu speichern.

9.2.2. Designentscheidungen

In diesem Abschnitt werden die für das Projekt getroffenen Designentscheidungen sowie die Begründungen für die jeweils verwendeten Technologien und Ansätze dokumentiert. Entscheidungen werden in Form von Y-Statements [30] dokumentiert, um die Entscheidungsfindung transparent zu machen.

Berechnung als Background-Task

Die Berechnung der Zersiedelung kann bei grossen Datenmengen und komplexen Berechnungen mehrere Minuten in Anspruch nehmen. Die Benutzeroberfläche soll jedoch während der Berechnung weiterhin verfügbar sein und Updates zum Fortschritt der Berechnungen anzeigen. In Abbildung 9.2 wird das Y-Statement zum Einsatz eines Background-Tasks dargestellt, um die Berechnung der Zersiedelung im Hintergrund des Webservice auszuführen.

Y-Approach

Im Kontext der Berechnung der Zersiedelung sollen grosse Datenmengen verarbeitet werden, ohne die Benutzeroberfläche zu blockieren.

Starlette Background-Task

Resultate direkt zurückgeben

Y

Um die Benutzeroberfläche nicht zu blockieren und längere Batch-Berechnungen grosser Datenmengen zu ermöglichen, wird die Berechnung als Background-Task ausgeführt, wobei akzeptiert wird, dass die Resultate der Berechnung zu einem späteren Zeitpunkt verfügbar sein müssen.

Abbildung 9.2: Y-Statement zum Einsatz eines Background-Tasks.

Einsatz von Key-Value-Store
Wenn die Berechnung der Zersiedelung im Hintergrund ausgeführt wird, entsteht eine weitere Herausforderung: Der Stand der Berechnung und die daraus resultierenden Daten müssen kurzfristig gespeichert werden, um den Fortschritt der Berechnung zu verfolgen. Dabei gibt es verschiedene Möglichkeiten, die Daten zu speichern, wie z.B. in einer relationalen Datenbank oder in einem Key-Value-Store. Im Hinblick auf weitere Anforderungen, wie das mehrere Benutzer gleichzeitig auf Berechnungsergebnisse zugreifen können und die Daten trotzdem kurzfristig gespeichert werden sollen, wurde ein Key-Value-Store gewählt. Die Wahl fiel auf Valkey [42], da dieser eine einfache Integration in Python ermöglicht, quelloffen zur Verfügung steht und eine hohe Performance bietet. Die Entscheidung zum Einsatz eines Key-Value-Stores wird in Abbildung 9.3 dokumentiert.

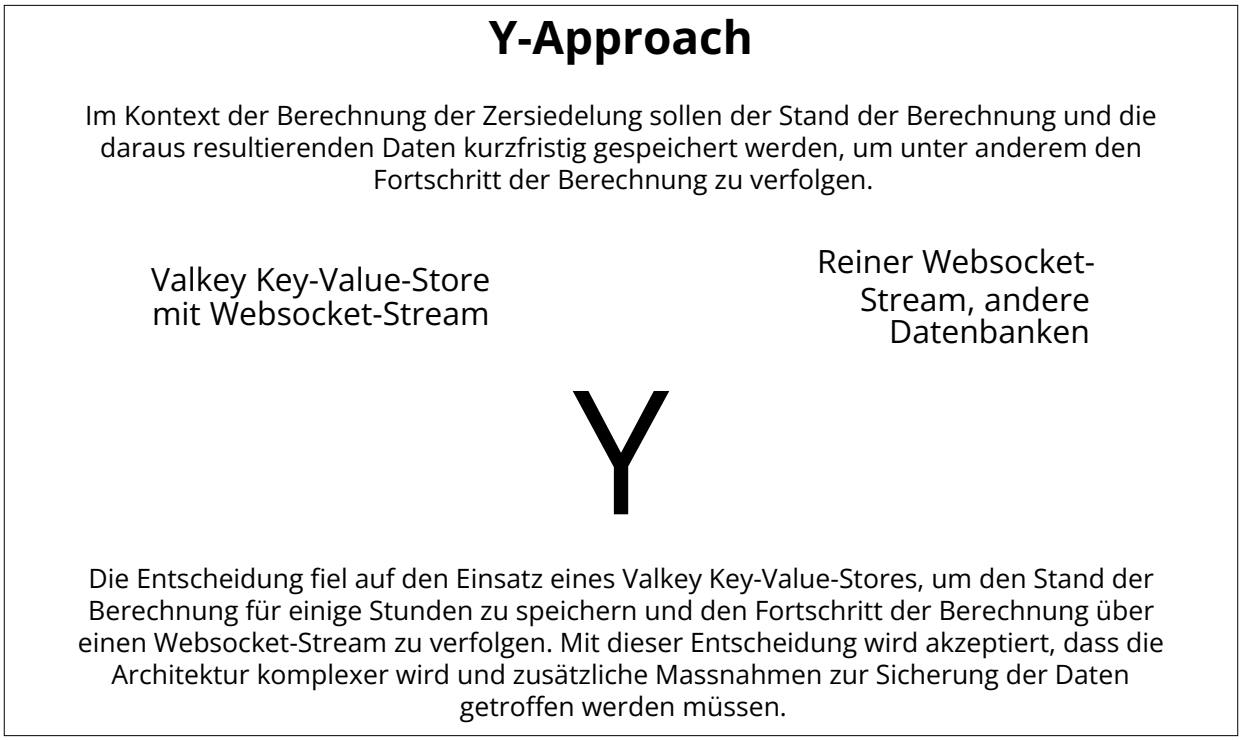


Abbildung 9.3: Y-Statement zum Einsatz eines Key-Value-Stores.

Einsatz von Joblib für die Parallelisierung
Die Berechnung der Zersiedelung kann für grössere Untersuchungsgebiete mehrere Minuten in Anspruch nehmen. Um die Performance zu verbessern, soll die Berechnung der Zersiedelung für mehrere Untersuchungsgebiete parallel durchgeführt werden. Nach einer empirischen Analyse der Performance verschiedener Bibliotheken zur Parallelisierung in Python, wurde Joblib [14] als die am besten geeignete Bibliothek ausgewählt. Joblib bietet einfache Lösungen zur Parallelisierung von Funktionen und ist einfach zu implementieren. Die Entscheidung zum Einsatz von Joblib wird in Abbildung 9.4 dokumentiert.

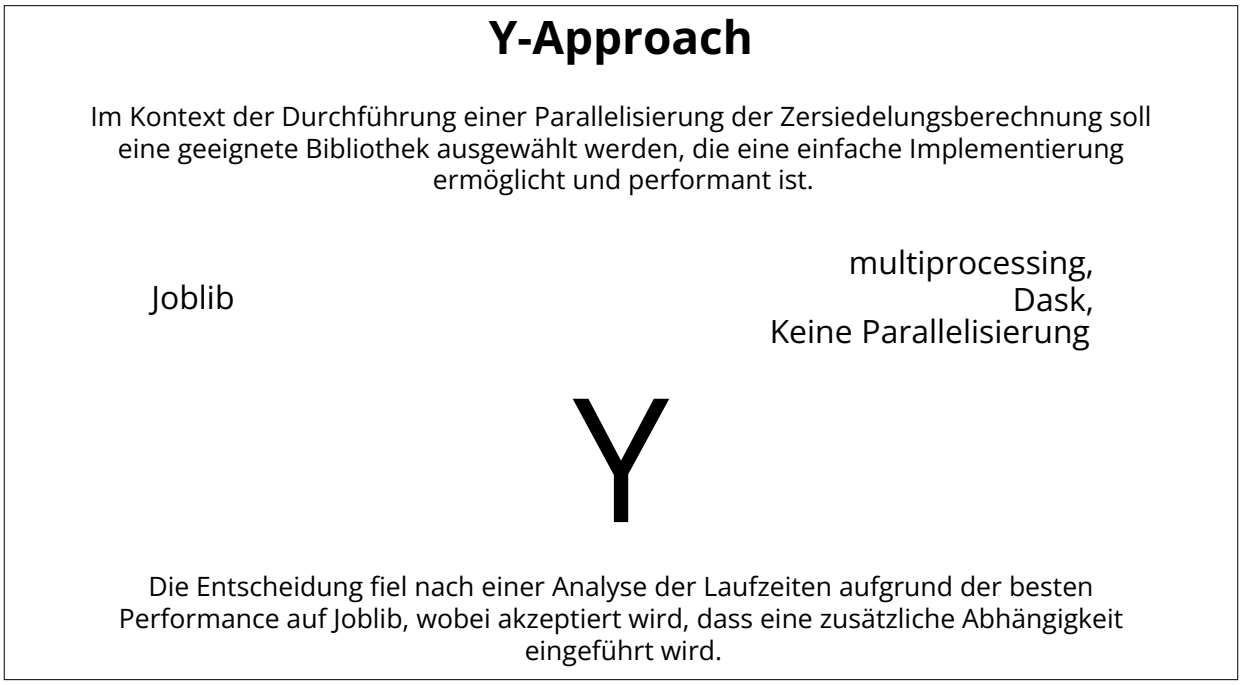


Abbildung 9.4: Y-Statement zum Einsatz von Joblib für die Parallelisierung der Berechnung.

Zusätzliche Erläuterungen zur verschachtelten Parallelisierung der Berechnung von Z sind im Abschnitt 10.4 zu finden.

9.3. Deployment

Das Deployment des USM-Calculators wurde bewusst in dieser Arbeit nicht thematisiert und aus dem Scope genommen, um den Fokus auf die Entwicklung der neuen Funktionalitäten zu legen. Dies impliziert jedoch nicht, dass das Deployment für die Software komplexer wäre.

Am einfachsten lässt sich der USM Calculator mit Docker [67] Compose deployen. Compose ist ein Tool, das es ermöglicht, mehrere Container in einer Gruppe zu verwalten und zu orchestrieren. Dabei kann unter anderem von einem Zusammenschluss der Container in abgekapselten, virtuellen Netzwerken profitiert werden. Um die Containerumgebungen für das Frontend und Backend zu erstellen, wurde jeweils ein `Dockerfile` im Wurzelverzeichnis erstellt. Das Dockerfile definiert die Laufzeitumgebung für die entsprechende Schicht und generiert ein Image zur einfachen Replikation der Umgebung.

Zusätzliche Docker Container werden für den Key-Value-Store Valkey [42] und den Reverse Proxy Traefik [68] verwendet. Der Valkey-Container wird für die kurzfristige Speicherung der Resultate während der Berechnung verwendet. Nebenbei dient er als Pub-Sub-System, um die Berechnungsergebnisse asynchron an die Benutzeroberfläche zu übermitteln. Der Traefik-Container wird als Reverse Proxy verwendet, um die Anfragen des Benutzers an die entsprechenden Container (Frontend oder Backend) weiterzuleiten.

9.3.1. Deployment Diagramm

Ein mögliches Deployment ist in Abbildung 9.5 dargestellt. Der Aufbau des Docker Compose Stacks ist in der Abbildung ersichtlich und zeigt wie die einzelnen Container miteinander kommunizieren. In oranger Farbe ist ein mögliches Konzept für das Deployment mittels Gitlab CI/CD auf einen Server dargestellt. Im Konzept wird der Code in einem Gitlab-Repository verwaltet und gemäss CI/CD-Pipeline-Konfiguration automatisch kompiliert und als Docker-Images in das Gitlab-Container-Registry hochgeladen. Ein Job der Pipeline kann anschliessend die Images auf den Server herunterladen und die Container starten.

Folgende Schritte sind im Deployment-Diagramm dargestellt (vgl. Abbildung 9.5):

1. Gitlab Runner scannt das Repository auf Änderungen und startet die CI/CD-Pipeline.
2. Für jedes Deployment wird eine Gitlab Runner Instanz gestartet und startet den Job des Deployments.
3. Mittels SSH wird eine Verbindung zum Server hergestellt, auf dem ein `docker-compose.yml` File abgelegt wird. (vgl. Docker Compose, Abschnitt 9.3).
4. Docker Compose lädt die Images aus der Gitlab-Container-Registry und von Docker Hub (z.B. traefik) herunter und startet die Container.

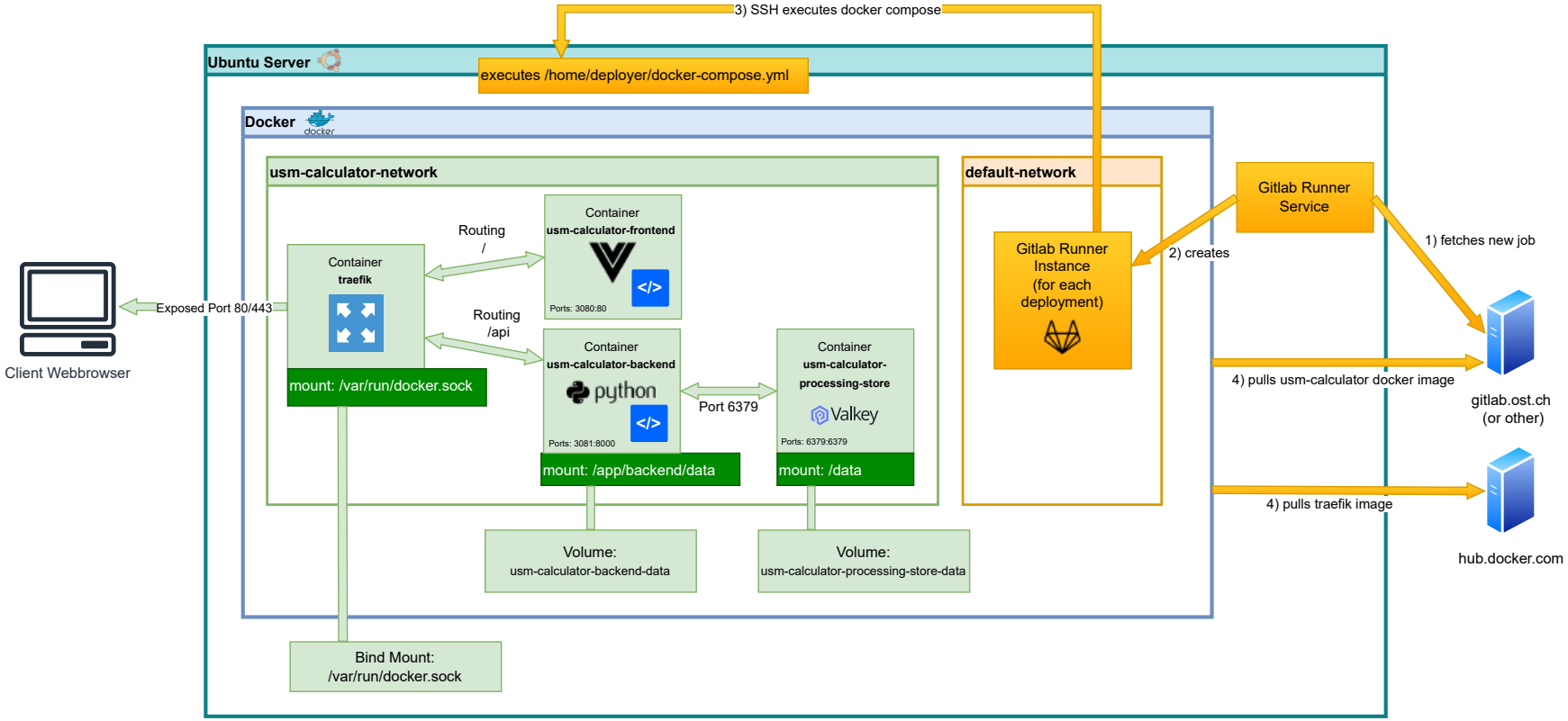


Abbildung 9.5: Diagramm eines möglichen Deployment-Konzepts für den USM-Calculator.

9.4. Paket- und Modulstrukturen

Die Codestrukturen des USM-Calculators sind in den folgenden Abschnitten beschrieben. Die Diagramme geben einen groben Überblick über den Aufbau des Codes und sind für Front- und Backend im selben Stil gehalten.

9.4.1. Frontend (Typescript-Module)

Ein Diagramm der Komponenten-, Script- und Modulstruktur des Frontends ist in Abbildung 9.6 dargestellt. Das Diagramm gibt Einsicht in das Unterverzeichnis `src` des Frontends, das den Quellcode des Frontends enthält. Die Unterschiede zu bestehenden Paketen sind im Diagramm farblich hervorgehoben.

Grundsätzlich wird das Frontend in die drei Hauptbereiche `components`, `services` und `model` unterteilt. Die Vue-Komponenten befinden sich im `components`-Verzeichnis. Diese sind für die Darstellung der Benutzeroberfläche verantwortlich und enthalten reaktive Logik, um die Benutzerinteraktion zu ermöglichen. Die Kommunikation mit dem Backend erfolgt über die `services`-Verzeichnisse, die REST-API-Aufrufe kapseln. Die Datenmodelle, die die Struktur der Daten definieren, werden im `model`-Verzeichnis abgelegt. Diese Modelle dienen als Typdefinitionen und helfen, die Datenintegrität zu gewährleisten.

Abhängigkeiten zu externen Bibliotheken und Paketen werden über die Paketverwaltung mit `yarn` verwaltet. Die Abhängigkeiten sind im `package.json`-Datei definiert, die sich im Wurzelverzeichnis des Frontends befindet.

9.4.2. Backend (Python-Pakete)

Die Unterschiede zu bestehenden Paketen sind im Diagramm der Abbildung 9.7 farblich hervorgehoben. Grün eingefärbte Pakete sind neu, orange eingefärbte Pakete sind bestehende Pakete, die um neue Funktionalitäten erweitert wurden. Bei gelb eingefärbten Kästchen handelt es sich um bestehende Pakete, bei denen keine neuen Funktionalitäten implementiert, sondern lediglich einige fehleranfällige Codeabschnitte verbessert wurden.

Wie bereits in der grundlegenden Studienarbeit beschrieben, wird eine "Relaxed Layer Architecture" [69] nach Robert C. Martin angestrebt. In diesem Prinzip werden die Abhängigkeiten von aussen nach innen aufgebaut, wobei die inneren Schichten keine Abhängigkeiten zu den äusseren Schichten haben. Abhängigkeiten müssen nicht direkt auf das nächstinnere Paket zeigen, sondern können auch über mehrere Pakete hinweg bestehen.

Grüne Linien und offene Pfeile kennzeichnen die Richtung der Abhängigkeiten (wird verwendet von) zwischen den Elementen. Vererbungsbeziehungen sind mit blauen Linien und leeren Pfeilen dargestellt (erbt von).

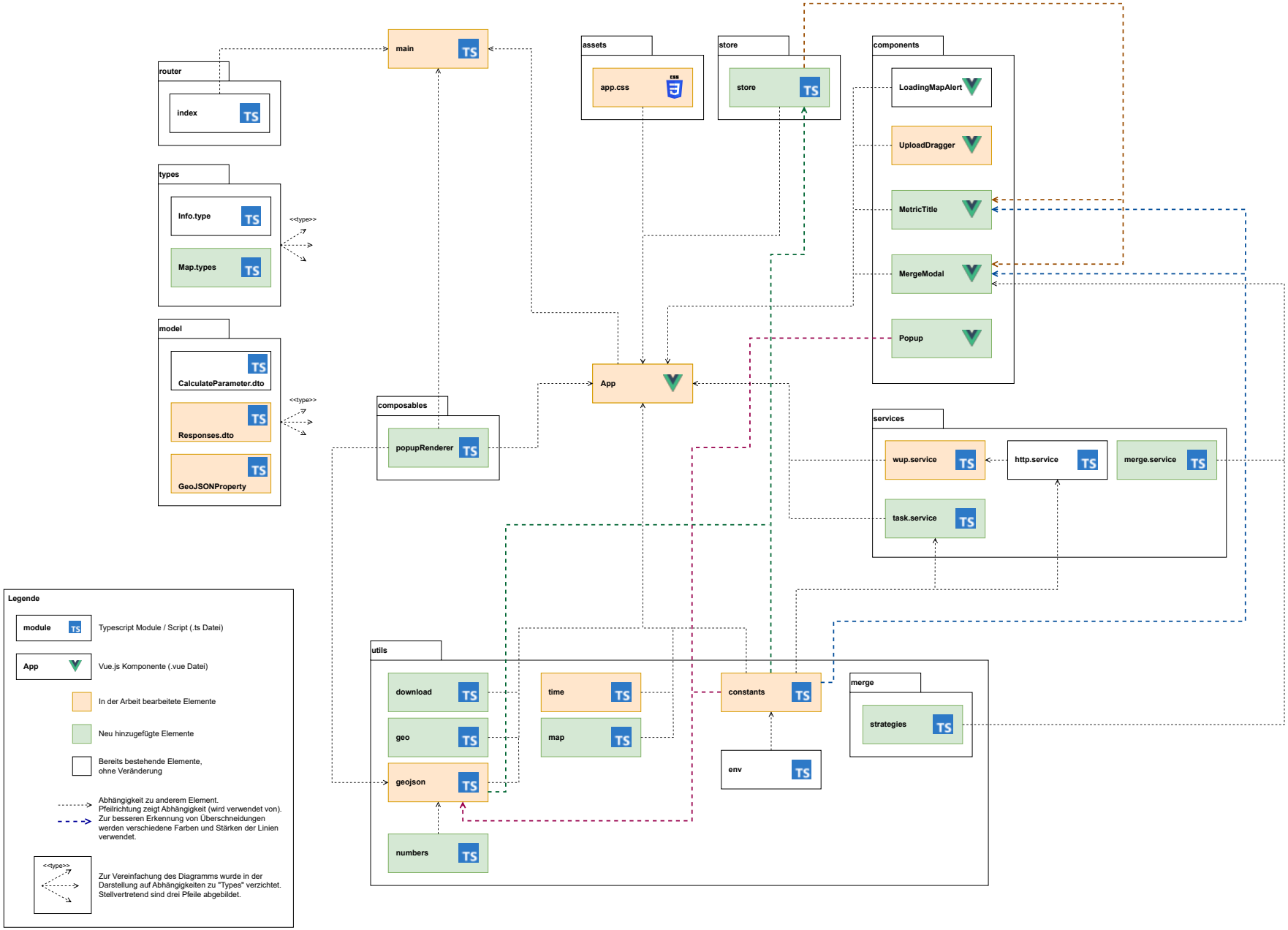


Abbildung 9.6: Typescript-Module des Frontends.

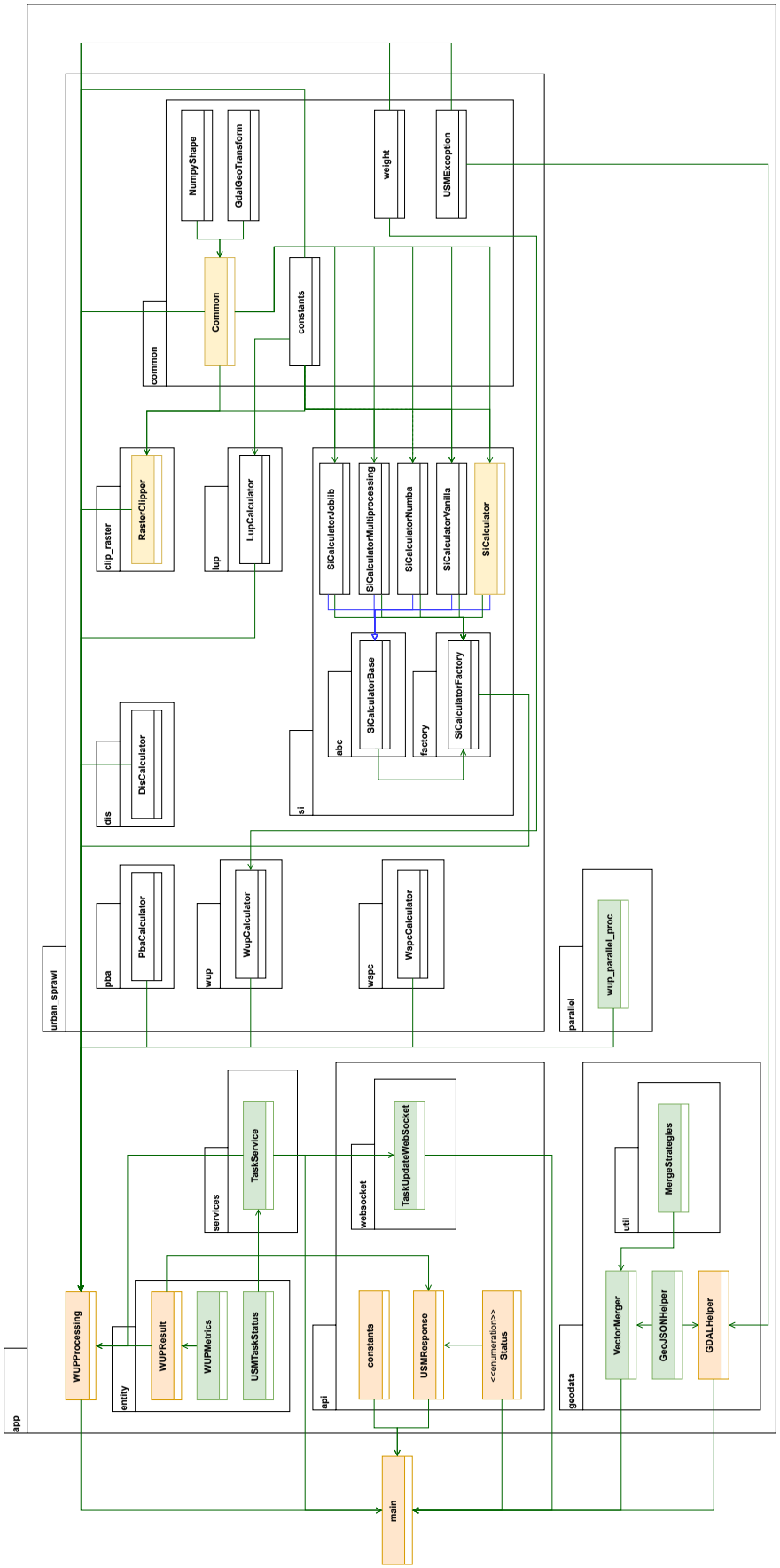


Abbildung 9.7: Python-Pakete des Backends.

9.5. Sequenzdiagramme

In dem folgenden Sequenzdiagramm in Abbildung 9.8 ist die vollständige Berechnung der Zersiedelung der einzelnen Features aus dem Vektor-Layer der Untersuchungsfläche dargestellt. Die Abbildung zeigt die Interaktion ab dem Zeitpunkt, an dem die REST-API den Request erhält. Die parallele Berechnung der Zersiedelung mit dem `SiCalculator` wird vereinfacht dargestellt. Die Berechnung der Zersiedelung erfolgt in einem Starlette Background-Task [70], wobei Starlette die Berechnung in einem separaten Thread über `asyncio` [31] ausführt.

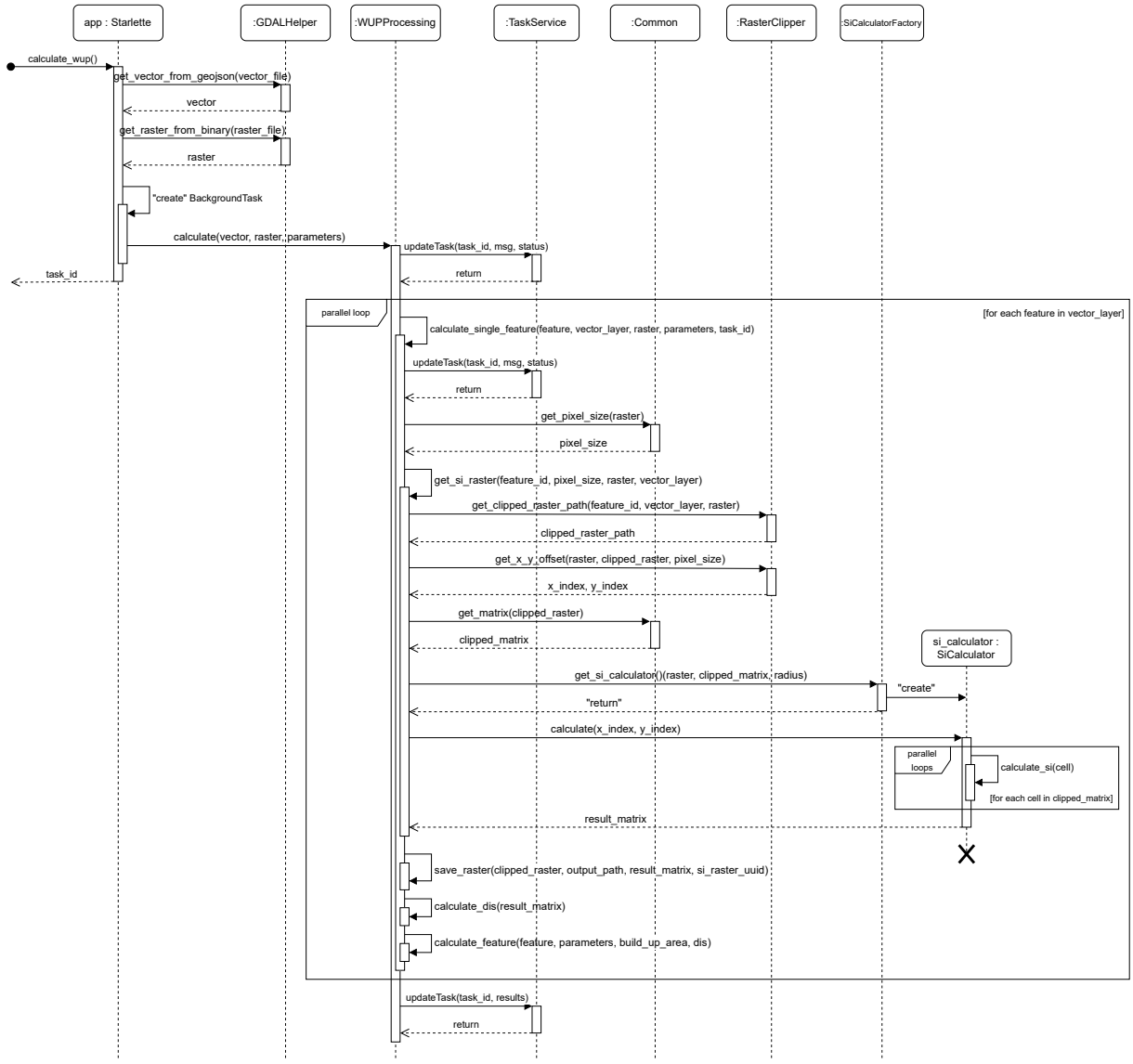


Abbildung 9.8: UML Sequenzdiagramm der vollständigen Berechnung der Zersiedelung von mehreren Features (Regionen) des Vektor-Layers.

In Abbildung 9.9 ist die Abfrage des Status der Berechnung über die API dargestellt. Der `TaskService` baut eine Verbindung zum Valkey-Server auf (Container) und fragt den Status der Berechnung über einen eindeutigen Task-Identifikator (UUID) ab.

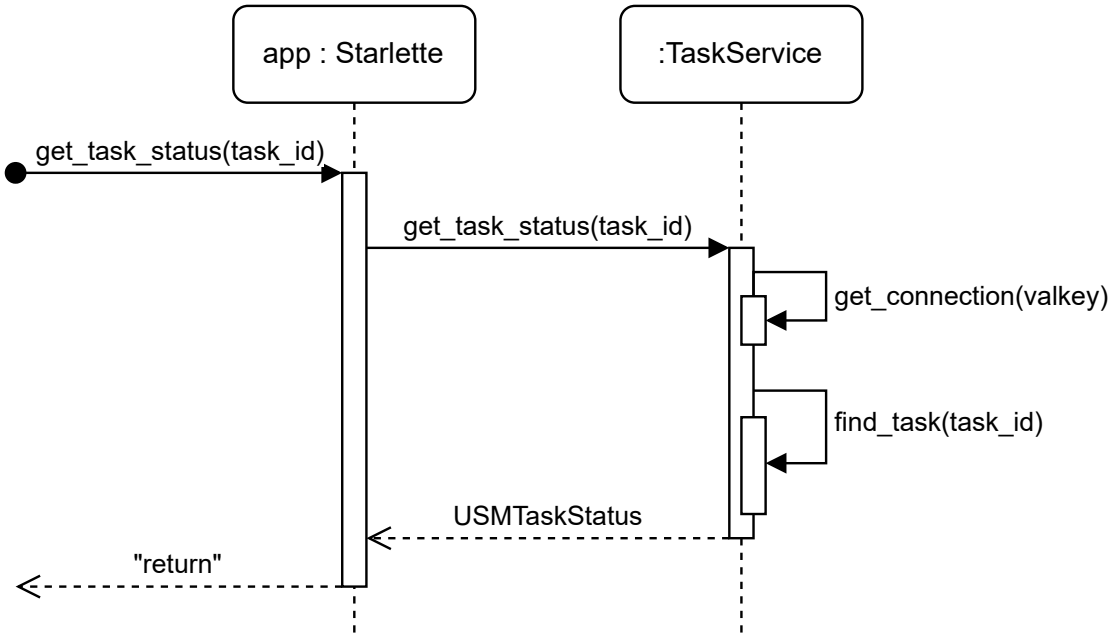


Abbildung 9.9: UML Sequenzdiagramm der Abfrage des Berechnungsstatus über die API.

Um einen vollständigen Ablauf einer Interaktion der Benutzer mit der API über das User Interface aufzuzeigen, wird in Abbildung 9.10 die Funktionalität des Zusammenführens (Merge) von mehreren Features (Regionen) des Vektor-Layers dargestellt. Im Diagramm stellen die Funktionen Pseudo-Code dar, die nicht dem tatsächlichen Code entsprechen, sondern die Logik verdeutlichen.

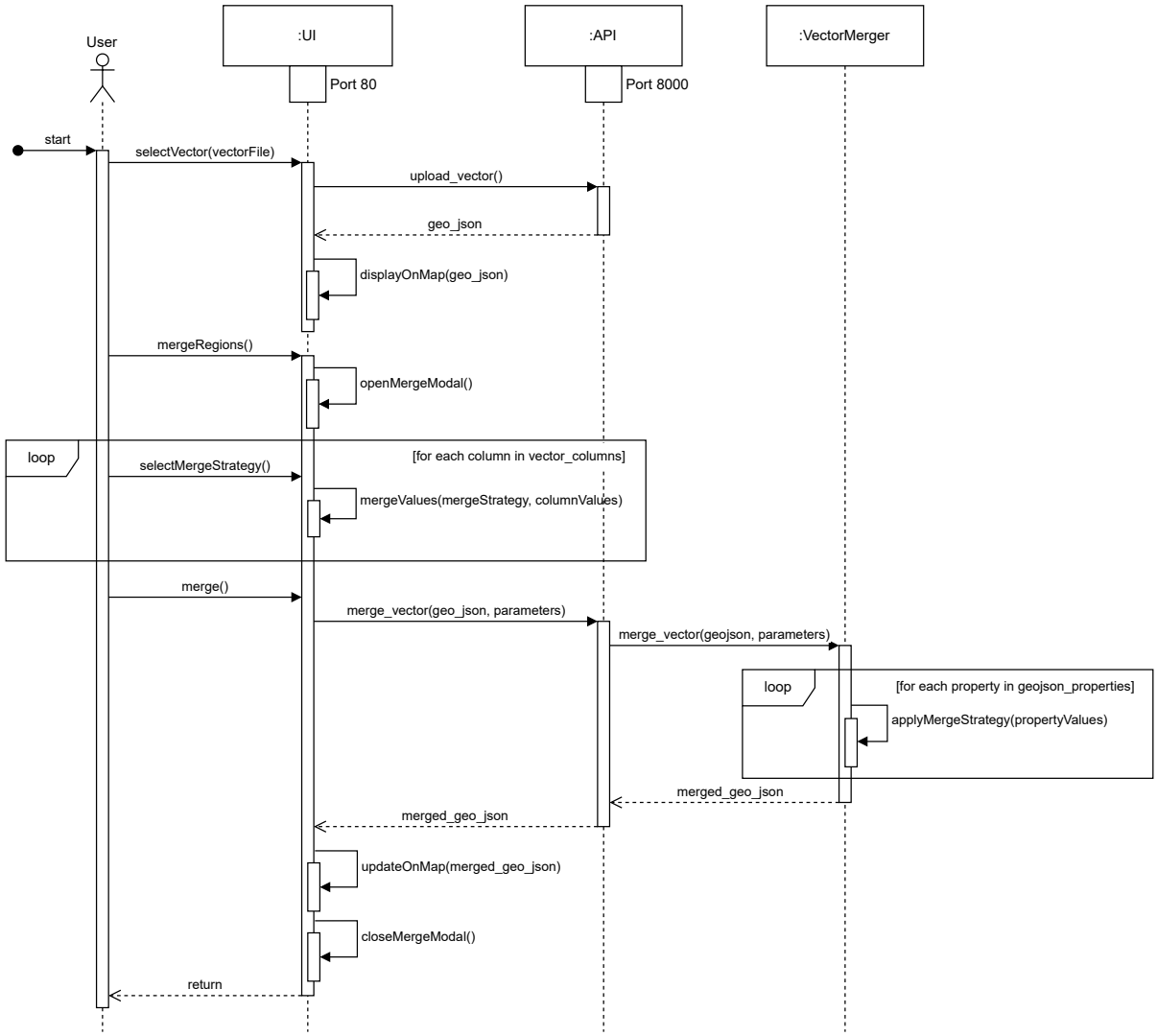


Abbildung 9.10: UML Sequenzdiagramm der Interaktion des Benutzers mit der API über UI bei Zusammenführen (Merge) von Untersuchungsgebieten (Vector).

9.6. UI Design

Ein umfassendes UI Design wurde für die Erweiterungen erstellt, um die Umsetzung graphisch zu planen. Dieses Design dient nebenbei als Grundlage für die Usability-Tests, die im Abschnitt 10.7 im Kapitel 10 beschrieben werden.

Für die Umsetzung des Designs wurde eine Testversion der Software Balsamiq Mockups [9] verwendet. Vordefinierte Komponenten der Software unterstützen die unkomplizierte Erstellung von Mockups.

Das Karten-Framework Leaflet [17] wird für die Darstellung der Karte verwendet. Als Standard-Tile-Layer wurde bisher OpenStreetMap [71] verwendet, wobei die Verwendung fortgeführt wird. Eine dynamische Tile-Layer-Auswahl ist in der Benutzeroberfläche vorgesehen, um den Benutzern die Möglichkeit zu geben, zwischen verschiedenen Stilen zu wechseln. Zusätzlich kann neben der OSM Swiss Style Karte [72] auch die Standardkarte von OpenStreetMap verwendet werden.

9.6.1. Skizzen der Benutzeroberfläche

Die Benutzeroberfläche orientiert sich an der bestehenden Lösung und wird möglichst intuitiv gestaltet. Es wird versucht, das User Interface so zu gestalten, dass möglichst wenige Klicks nötig sind, um die gewünschten Funktionen zu erreichen. Trotzdem soll die Oberfläche übersichtlich bleiben. Erste Entwürfe der zusätzlichen Funktionalitäten sind in den Abbildungen 9.11 und 9.12 dargestellt.

Erste Entwürfe der Benutzeroberfläche
In den ersten Entwürfen der Benutzeroberfläche wurde die Umgebung der Rasterbearbeitung und das ”Popup” auf der Karte für die Bearbeitung der Statistikdaten skizziert.

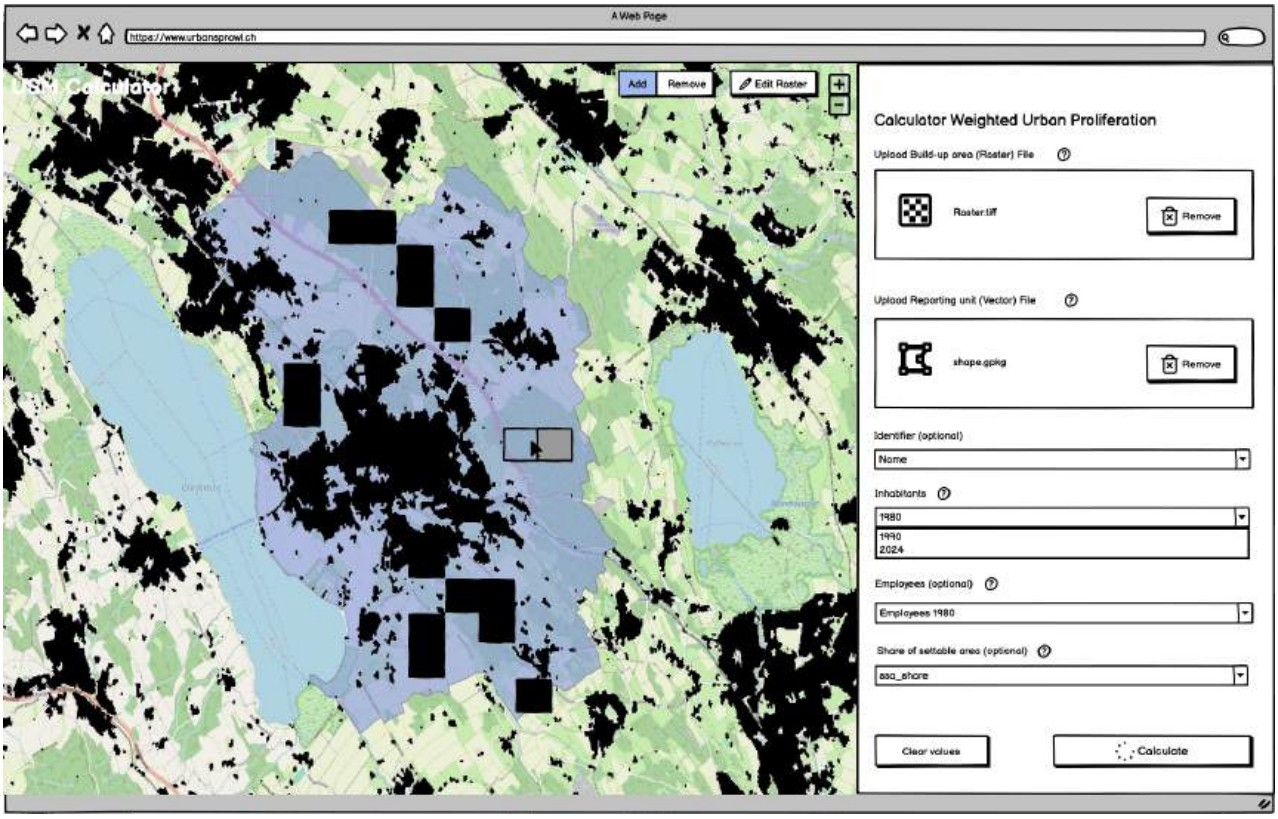


Abbildung 9.11: Erster Entwurf zur Rasterbearbeitung in der Benutzeroberfläche des USM Calculators.

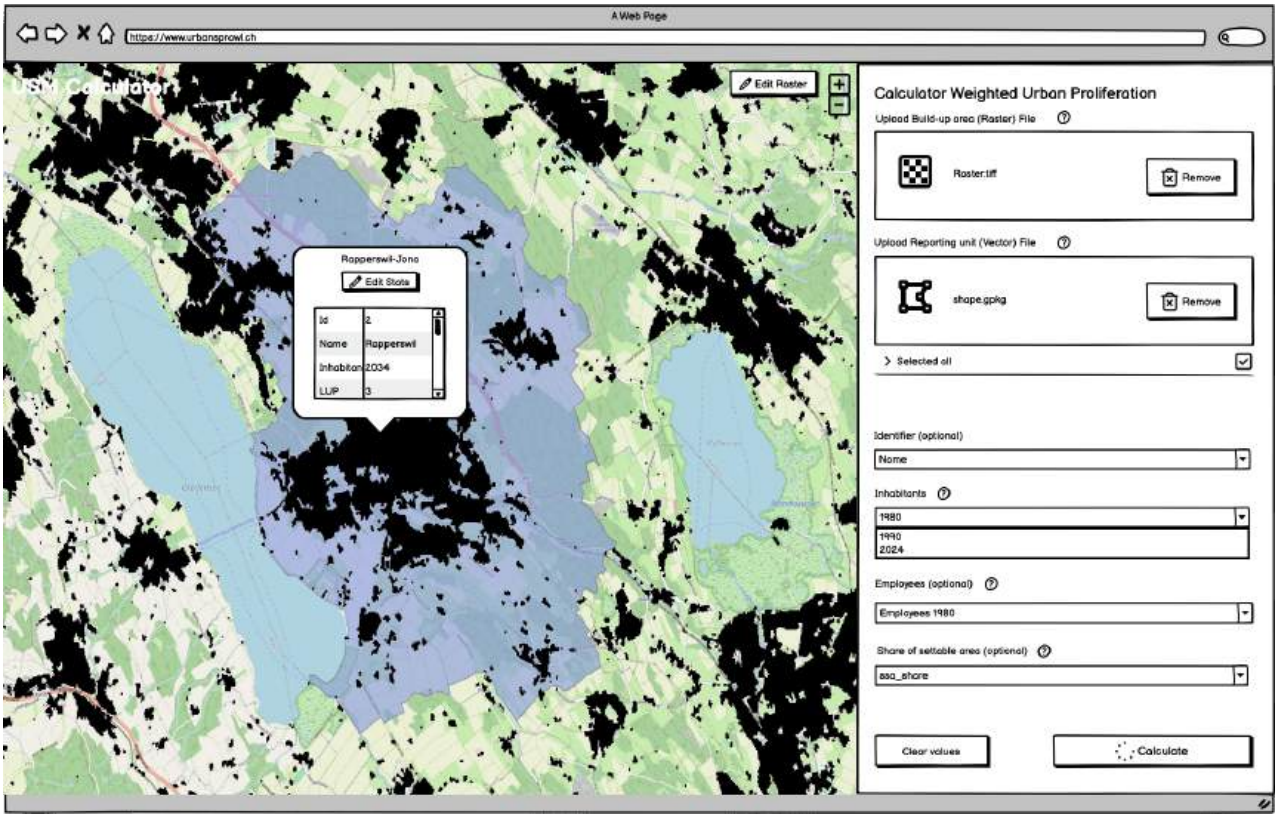


Abbildung 9.12: Erster Entwurf der Darstellung von Statistikdaten in der Benutzeroberfläche des USM Calculators.

9.6.2. Weitere Entwürfe der zusätzlichen Funktionalitäten

Die Benutzeroberfläche wurde weiter verfeinert, um die Bedienung zu verbessern und die neuen Funktionalitäten zu integrieren, was in den Abbildungen 9.13, 9.14 und 9.15 dargestellt ist.

Raster bearbeiten

Im nachstehenden Mockup in Abbildung 9.13 sieht man, dass nun über der Karte ein Balken mit einem Button eingefügt wurde. Mit diesem Button kann man die Bearbeitung des Rasters aktivieren. Während man sich im Bearbeitungsmodus befindet, wird die ganze Karte rot umrahmt, damit dies auch erkennbar wird. Zusätzlich gibt es im Balken über der Karte nun andere Buttons. Mit dem einen Button kann man im Raster etwas hinzufügen, mit dem anderen falsche Ergänzungen wieder entfernen. Wenn man mit der Bearbeitung fertig ist, können die Änderungen durch den ”Save”-Button gespeichert werden. Die erfolgreiche Speicherung wird durch einen Hinweis bestätigt und die Ansicht ändert sich wieder zurück zum Standard.

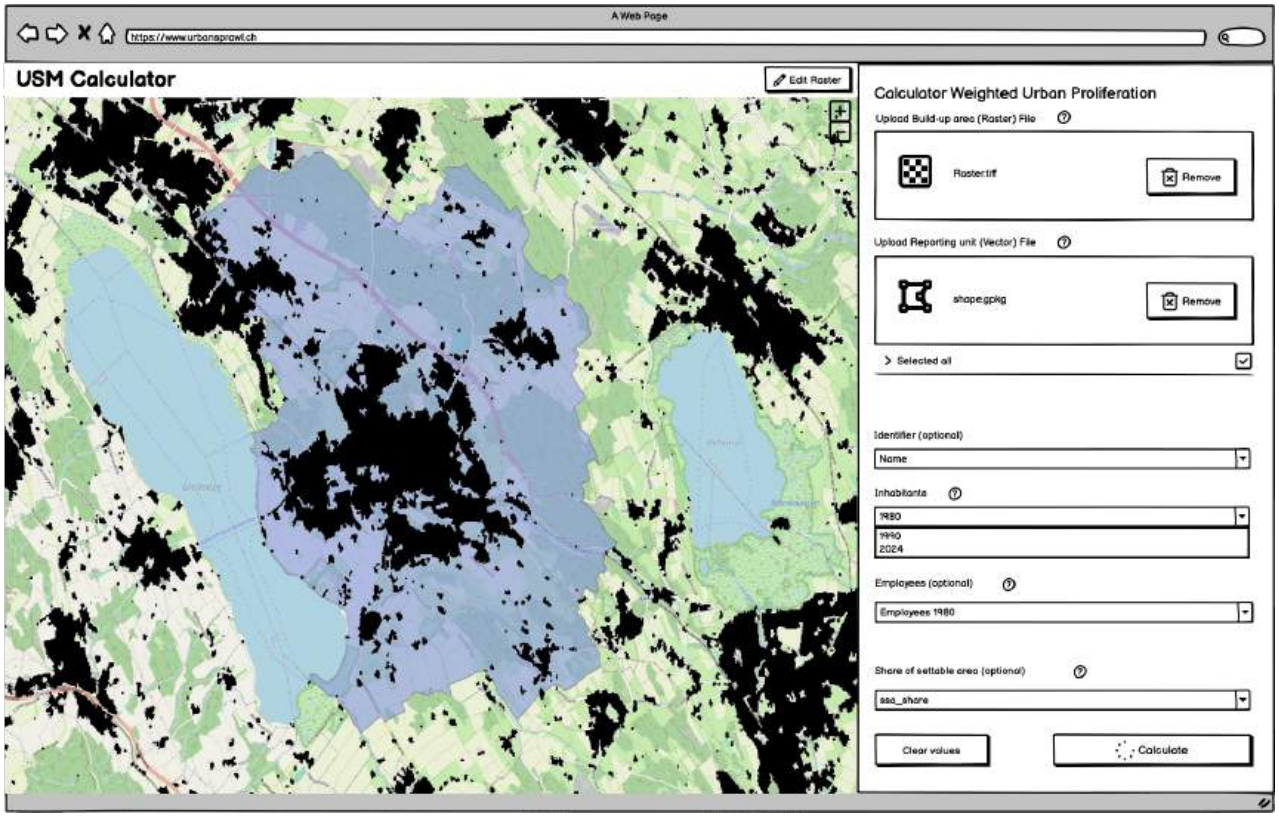


Abbildung 9.13: Mockup der Rasterbearbeitung in der Benutzeroberfläche des USM Calculators.

Statistikdaten bearbeiten

Im Mockup der Abbildung 9.14 werden die Statistikdaten vom Vector-File thematisiert. Diese werden auf der Karte in einem Popup angezeigt. Um diese Daten bearbeiten zu können, kann man auf den ”Edit”-Button klicken. Zusätzlich kann die Vektordatei mehr als ein Gebiet beinhalten. Dies ist an dem Dropdown unterhalb der hochgeladenen Datei zu erkennen. Standardmässig sind alle angewählt.

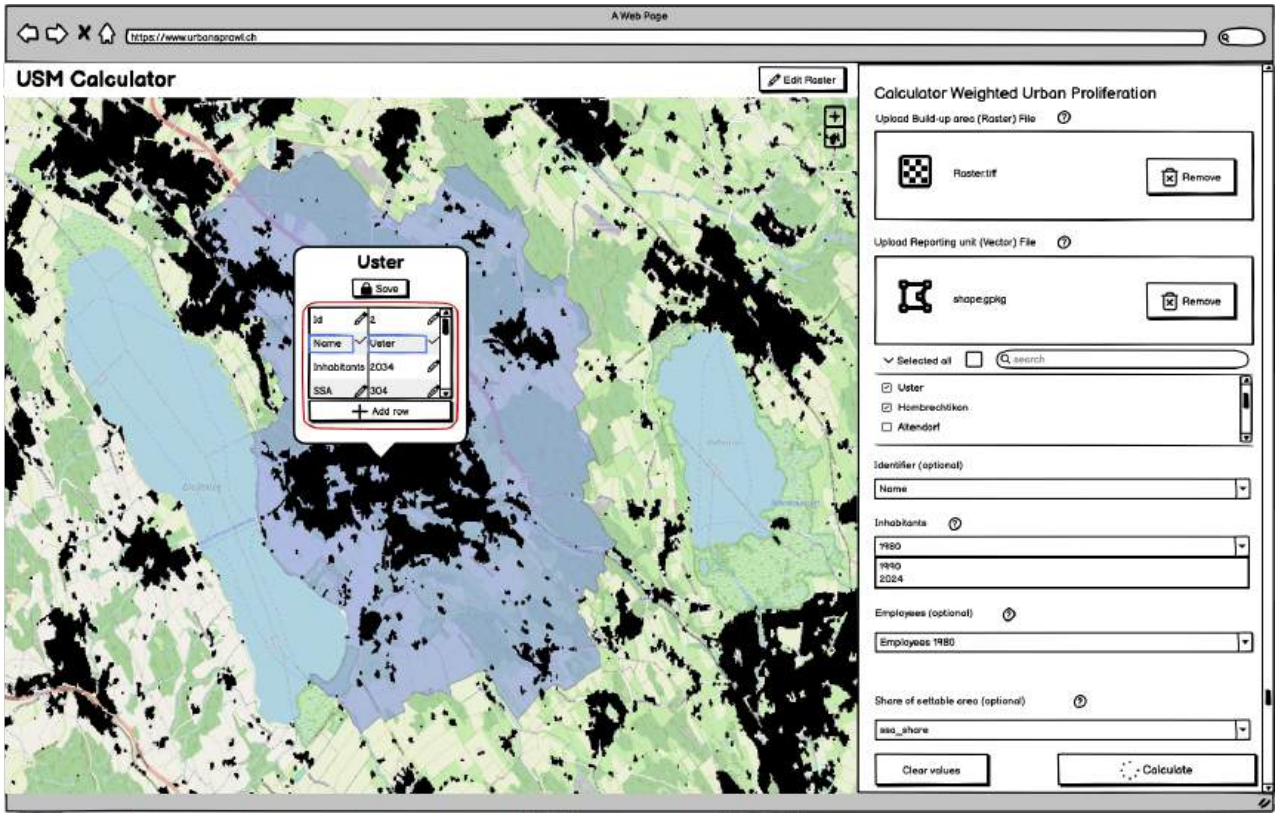


Abbildung 9.14: Statistikdaten bearbeiten in der Benutzeroberfläche des USM Calculators.

Zusammenführen von Gebieten

In Abbildung 9.15 wird das Zusammenführen von Gebieten dargestellt. Über einen Button in der Benutzeroberfläche kann der Benutzer alle Gebiete des Vektor-Layers zu einem einzigen Gebiet zusammenführen. Mit einem ”Modal”-Fenster werden dem Benutzer die Informationen zu den Gebieten angezeigt, die zusammengeführt werden sollen. Die Methode des Zusammenführens kann über einen Dropdown ausgewählt werden.

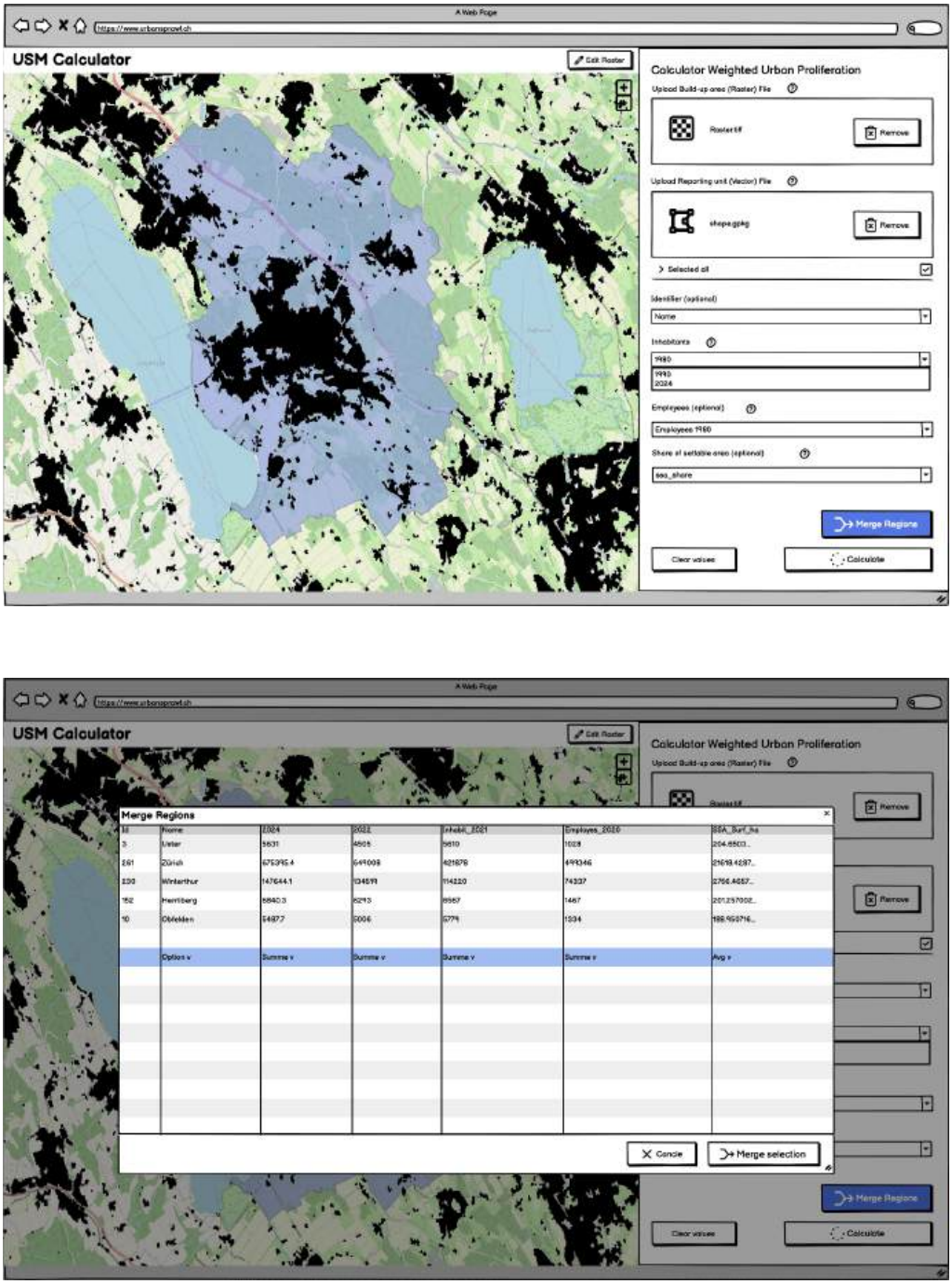


Abbildung 9.15: Mockup des Zusammenführens von Gebieten in der Benutzeroberfläche des USM Calculators.

9.6.3. Endgültiges Design

In den Abbildungen 9.16 und 9.17 sind jeweils zwei Screenshots der Benutzeroberfläche im Vergleich zur Ausgangslage der Studienarbeit dargestellt. Die Umsetzung des Designs wurde mit dem Vue.js-Framework, sowie der Ant Design-Komponentenbibliothek realisiert. Die Erweiterung brachte einen neuen Header-Bereich mit einem Titel. Der neue Bereich schafft Platz für weitere Steuerungselemente.

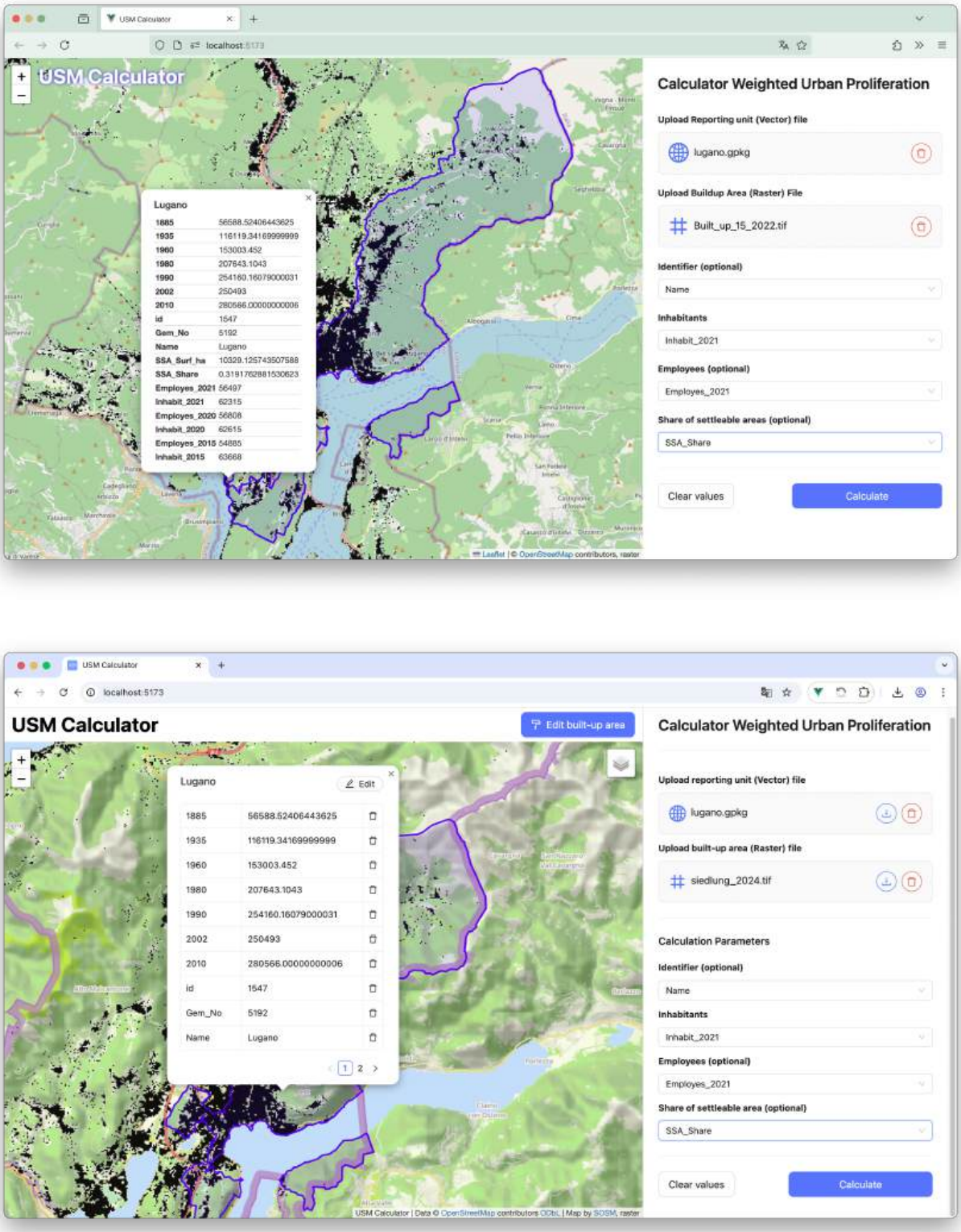


Abbildung 9.16: Vergleich der Startseite des UI vor und nach den Erweiterungen dieser Arbeit.

Weitere Beispiele sind im Kapitel 13 im Abschnitt 13.2 der Bedienungsanleitung zu finden.

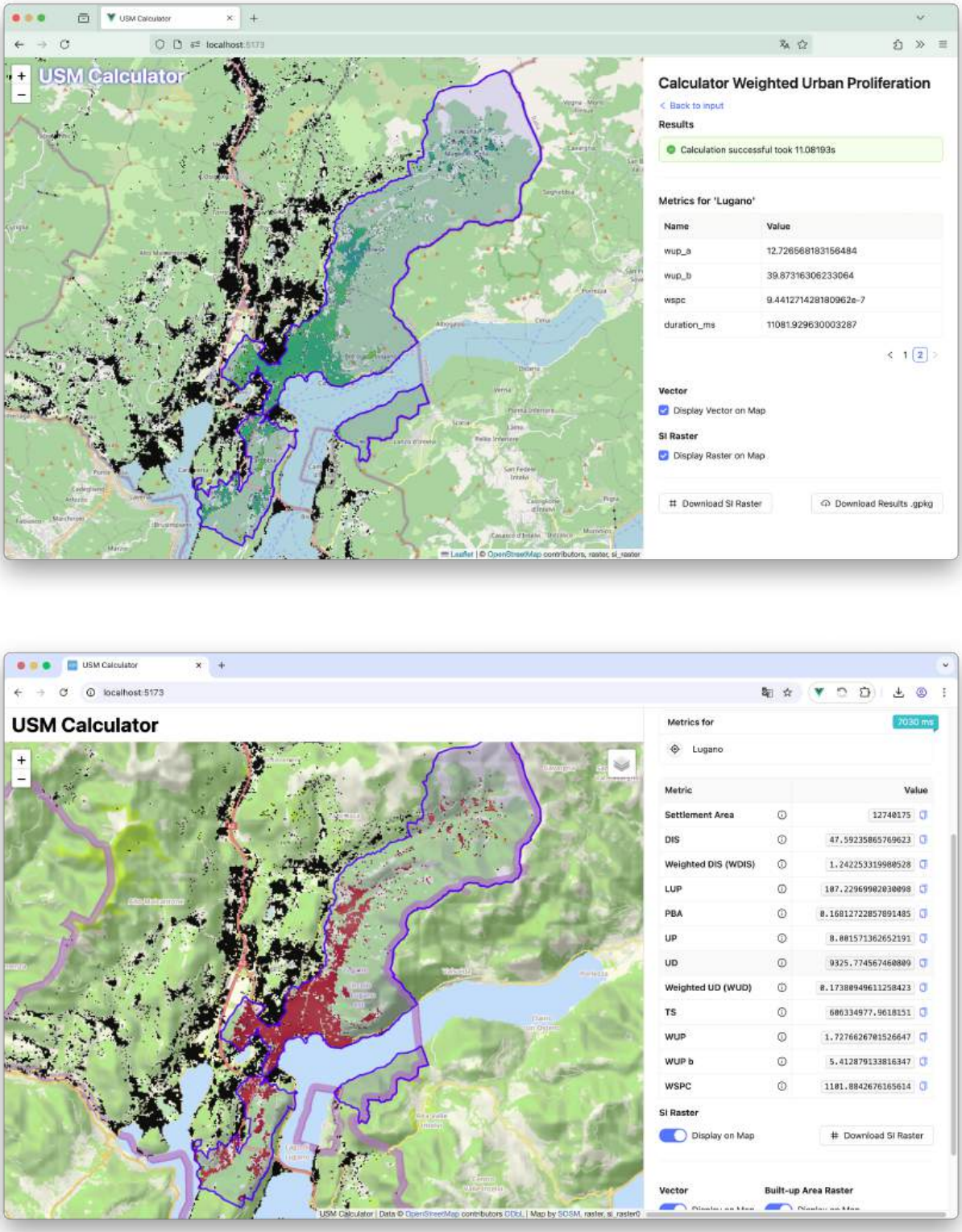


Abbildung 9.17: Vergleich der Resultatsansicht des UI vor und nach den Erweiterungen dieser Arbeit.

10
Implementierung und Test

10.1. Erweiterung

In folgendem Kapitel werden die Details der Implementierung der Erweiterungen, die in den funktionalen Anforderungen (vgl. Abschnitt 6.1) definiert wurden, beschrieben. Weitere technische Details können dem Quellcode entnommen werden.

10.1.1. Webservice

Der bereits in der Studienarbeit entwickelte Webservice wurde als Grundlage genommen und erweitert. Dabei wurden zusätzliche Endpoints für die REST-API implementiert, um die neuen Funktionalitäten zu ermöglichen. Der Webservice ist auf dem Starlette-Webframework [11], das auf ASGI (Asynchronous Server Gateway Interface) basiert, implementiert.

Erweiterung der RESTful API
Grundsätzlich existiert für jeden Rechenschritt des Vorgehens zur Berechnung von Z ein Endpunkt.

In der Tabelle 10.1 sind die Endpunkte der RESTful API aufgelistet, die zur Verfügung stehen. Neue im Rahmen dieser Arbeit implementierte Endpunkte sind in der ersten Spalte mit einem farbigen Punkt markiert. Die OpenAPI Spezifikation der API ist im Quellcode enthalten und wird mittels Swagger UI [12] im Frontend angezeigt. Swagger UI liest dazu ein Schema, das von Starlette generiert wird und stellt die Endpunkte in einer benutzerfreundlichen Oberfläche dar.

Die Erklärung der einzelnen Metriken und deren Schritte zur Berechnung sind detailliert im Abschnitt 7.1.1 beschrieben. Eine vollständige Dokumentation der API kann dem Open-Source-Repository unter `https://gitlab.ost.ch/sa-urban-sprawl-metrics/usm-calculator` entnommen werden.

Neu	Methode (HTTP)	Endpoint	kurze Beschreibung
•	POST	`/calculate`	Berechnet alle folgenden Metriken und gibt neu standardmässig eine `task_id` (UUID) zurück, um die Berechnung zu verfolgen.
	POST	`/calculate/dis`	Degree of Urban Dispersion (DIS), auch Streuung (Dispersion).
	POST	`/calculate/lup`	Land Uptake per Person (LUP).
	POST	`/calculate/pba`	Proportion of Built-up Area (PBA).
	POST	`/calculate/si`	Sprawl at Index (SI). Die Resultate werden als `tif`-Datei (GeoTIFF) zurückgegeben.
	POST	`/calculate/ts`	Total Sprawl (TS).
	POST	`/calculate/ud`	Utilization Density (UD).
	POST	`/calculate/up`	Urban Permeation (UP).
	POST	`/calculate/wdis`	Weighted Degree of Urban Dispersion (WDIS).
	POST	`/calculate/wspc`	Anteil der Zersiedelung pro Person im Untersuchungsgebiet (Weighted Sprawl per Capita, WSPC).
	POST	`/calculate/wud`	gewichtete Utilization Density (WUD). (gewichteter Wert der Anzahl Einwohner und Arbeitsplätze pro Quadratkilometer bebauter Fläche)
	POST	`/calculate/wup-a`	Weighted Urban Proliferation (WUP, deutsch: Zersiedelung Z) für das gesamte Untersuchungsgebiet.
	POST	`/calculate/wup-b`	Weighted Urban Proliferation (WUP) für den besiedelbaren Teil des Untersuchungsgebiets.
•	POST	`/si-raster/zip`	Erstellt ein ZIP-Archiv mit dem SI-Raster (Sprawl at Index Raster) und den zugehörigen Metadaten.
•	GET	`/si-raster/{file_id}`	Gibt das SI-Raster (Sprawl at Index Raster) als GeoTIFF-Datei zurück.
•	GET	`/task/{task_id}`	Gibt den Status der Berechnung zurück. Die Berechnung wird im Hintergrund ausgeführt und der Status kann über eine UUID (task_id) abgefragt werden.
	POST	`/vector`	Ermöglicht das Hochladen einer Geopackage-Datei. Gibt die Datei als GeoJSON, reprojeziert im geodätischen Referenzsystem EPSG:4326 [73], zurück.
	POST	`/vector/download`	Konvertiert das hochgeladene GeoJSON in ein Geopackage und streamt die Datei zurück.
•	POST	`/vector/merge`	Ermöglicht das Zusammenführen von Vektordaten. Die Geometrien der Features werden dabei zusammengeführt und die Attribute der Features werden gemerged.
•	*WebSocket*	`/ws/task-status/{task_id}`	Ermöglicht die Echtzeit-Überwachung des Status der Berechnung. Der WebSocket wird vom Frontend genutzt, um den Status der Berechnung zu aktualisieren.

Tabelle 10.1: API Endpunkte.

10.2. Berechnung

In diesem Abschnitt wird die Umsetzung der Berechnung von Z beschrieben.

10.2.1. Grundlagen

Wie bereits im Abschnitt 7.1 beschrieben, erfolgt die Berechnung der Zersiedelung Z in mehreren Schritten.

Input-Parameter
Die Berechnung der Zersiedelung Z erfolgt auf Basis von Geodaten, die in Form von Raster- und Vektordaten vorliegen. Konkret arbeitet die Applikation mit den folgenden Input-Parametern:

- Ein Vektor-Polygon (im GeoPackage- oder GeoJSON-Format) mit Gebietsgrenzen und Informationen (als Attribute).
- Ein Raster (im GeoTIFF-Format) mit Informationen über die Siedlungsfläche (Optimal ist ein Raster mit einer Auflösung von 15m mit quadratischen Pixeln).
- Anzahl der Einwohner (als Attribut im Vektor-Polygon).
- Anzahl der Arbeitsplätze (als Attribut im Vektor-Polygon), optional.
- Wert für den SSA, den Anteil der besiedelbaren Fläche des Untersuchungsgebiets (als Attribut im Vektor-Polygon), optional.
- Ein Attribut des Vektors, um eine ID des Untersuchungsgebiets zu speichern, damit die Ergebnisse der Berechnung zugeordnet werden können, optional (der Index wird verwendet, wenn nicht angegeben).

Die Berechnung ermittelt Z in einem Radius von 2'000 Metern um die Grenzen des Untersuchungsgebiets. Dies wurde in der Methode zur Bestimmung von Z in der Literatur [3] definiert und wird in der Applikation umgesetzt, weil sonst die Gewichtungsfunktionen (vgl. Abschnitt 7.1) angepasst werden müssten.

Zuschnitt des Rasters an die Gebietsgrenzen
Das Raster wird mithilfe der GDAL-Funktion `Warp` an den Grenzen entlang der einzelnen Untersuchungsgebiete des Vektors zugeschnitten. Dafür wird jedes Gebiet (GeoJSON-Feature) in ein Shapefile umgewandelt, um das Raster zuzuschneiden. Anschliessend erfolgt die Umwandlung des zugeschnittenen Rasters in eine Numpy-Matrix zur weiteren parallelisierten Verarbeitung der SI-Raster-Berechnung, die in der Studienarbeit mit der Numba-Bibliothek [27] umgesetzt wurde.

Berechnung des SI-Rasters
Für jedes bebaute Pixel im Untersuchungsgebiet wird die mittlere Distanz zu anderen bebauten Pixeln innerhalb des definierten Radius (2'000 m) berechnet. Diese Werte werden in einem SI-Raster (Sprawl at Index Raster) gespeichert, die die räumliche Verteilung der Bebauung abbildet (Streuung). Daraus lässt sich der Dispersion-Wert (DIS) ableiten (vgl. Abschnitt 7.1).

10.2.2. Sicherstellung der Datenintegrität

Zur Überprüfung der Datenqualität der Resultate kann der unten beschriebene Ablauf verwendet werden. Als "Quelle der Wahrheit" werden die Resultate des QGIS-Plugins [7] verwendet, die mit den gleichen Geodaten und den gleichen Berechnungsschritten erstellt wurden.

1. Die Berechnung von Z wird für eine beliebige Region mit dem QGIS-Plugin "USM-Toolset" durchgeführt.
2. Dieselben Geodaten (Raster und Vektor) werden in den "USM Calculator" hochgeladen und die Berechnung ebenfalls durchgeführt.
3. Die Resultate der verschiedenen Berechnungen werden verglichen.
4. Die Resultate sollten übereinstimmen, wenn die Geodaten und die Berechnungsschritte identisch sind.

Damit dieses Vorgehen für andere Entwickler replizierbar vollzogen werden kann, wurde ein automatisierter Test mit implementiert (vgl. Abschnitt 10.6).

10.3. Bearbeiten von Geodaten

Die Bearbeitung der Geodaten umfasst zwei Teile: Die Bearbeitung von Rasterdaten und die Bearbeitung von Vektordaten.

10.3.1. Rasterdaten bearbeiten

Vorgehen und Herausforderungen
Die Rasterbeabeitung soll es ermöglichen, die Siedlungsfläche des Rasters zu bearbeiten, um die Auswirkungen von geplanten Siedlungsentwicklungen zu untersuchen. Eine Herausforderung besteht in der effizienten Bearbeitung von Rasterdaten, die in der Regel grosse Datenmengen repräsentieren. Dies erfordert einige Überlegungen zur Performance und Speicherverwaltung, insbesondere weil die Rasterdaten in der Browserumgebung verarbeitet werden sollen.

Bei den Rasterdaten handelt es sich um eine grosse Matrix, die bebaute und unbebaute Pixel (repräsentieren 15m x 15m) enthält. Bebaute Pixel sind dabei mit dem Wert "1" und unbebaute Pixel mit dem Wert "0" kodiert. Die Bearbeitung soll es ermöglichen, einzelne Pixel des Rasters zu bearbeiten. Ein genaueres Abbild der Realität ist nicht notwendig, da die Siedlungsfläche in der Regel in grösseren Regionen betrachtet wird.

Es wurde in Betracht gezogen, die Änderungen im Backend zu verarbeiten, was jedoch zu einer hohen Datenübertragung führen würde, da das gesamte Raster nach jeder Bearbeitung an das Backend gesendet werden müsste. Diese Lösung wurde verworfen, da sie zu ineffizient wäre und weitere Herausforderungen bei Datenschutz und Performance mit sich bringen würde.

Die Entscheidung wird in Abbildung 10.1 in Form eines Y-Statements festgehalten.

Y-Approach

Im Kontext der Bearbeitung von Rasterdaten, soll diese effizient und unkompliziert umgesetzt werden, um das neue Raster unmittelbar nach der Bearbeitung zur Berechnung der Zersiedelung zu verwenden.

Frontend Backend

Y

Die Entscheidung fiel auf die Bearbeitung im Frontend, wobei die Änderungen im Frontend gehalten und erst bei der Berechnung an das Backend gesendet werden, um schwerfällige Datenübertragungen zu vermeiden. Dabei wird in Kauf genommen, dass grössere Datenmengen im Frontend verarbeitet werden müssen.

Abbildung 10.1: Y-Statement zum Vorgehen bei der Bearbeitung von Rasterdaten.

In einer ersten Iteration wurde die Bearbeitung der Pixel direkt im Raster-Layer der Leaflet-Karte vorgenommen. Dabei wird eine "ImageOverlay"-Komponente von Leaflet verwendet, die es ermöglicht, ein Rasterbild auf der Karte darzustellen. Die Bibliothek georaster-layer-for-leaflet [74] wird verwendet, um das Rasterbild in ein Leaflet-Layer zu konvertieren. Die Bearbeitung der Pixel erfolgt durch das Klicken auf die Karte, wobei die Koordinaten des Klicks in Pixelkoordinaten umgerechnet werden. Die Pixelkoordinaten werden dann verwendet, um den Wert des Pixels im Raster zu ändern. Diese Lösung wurde jedoch verworfen, da sie nicht den Anforderungen an die Performance und Benutzerfreundlichkeit entsprach. Ein "flashing" der Karte, also das Neuladen des Rasters nach jeder Bearbeitung, war notwendig, um die Änderungen anzuzeigen. Dies sorgte dafür, dass ständig die gesamte Karte neu geladen werden musste, was zu einer schlechten Benutzererfahrung führte.

In einer zweiten Iteration wurde eine Lösung für die Bearbeitung gefunden und ist nachfolgend beschrieben.

Lösung
Die Idee besteht darin, die Rasterdaten nicht direkt nach jeder Bearbeitung zu speichern, sondern die Änderungen in einen separaten ArrayBuffer zu speichern. Dabei werden nach der Speicherung die Änderungen in das bestehende Raster "eingebrannt", sodass diese effizient und ohne unnötige Datenübertragung durchgeführt werden können. In der Lösung wird weiterhin die Leaflet-Karte in Kombination mit der JavaScript-Bibliothek georaster-layer-for-leaflet [74] verwendet, die Darstellungen von GeoTIFF-Rastern in Leaflet ermöglicht.

Nach dem Einbrennen des Musters in das Raster muss die ursprüngliche Rasterdatei ersetzt werden. Dabei wird ein neuer ArrayBuffer (des neuen Rasters) erstellt und daraus ein neues File-Objekt generiert. Dieses neue File-Objekt wird bei der nächsten Berechnung an das Backend gesendet.

In mehreren Versuchen mit verschiedenen Raster-Dateien zeigte sich, dass die Performance ausreichend ist, um die Änderungen in Echtzeit anzuzeigen.

10.3.2. Vektordaten bearbeiten

Vorgehen
Die Bearbeitung soll es ermöglichen, einzelne Gebiete des Vektors, sogenannte "Features", effizient und benutzerfreundlich zu bearbeiten. Es wurde entschieden, dass die Bearbeitung in einem separaten Fenster ("Popup") erfolgen soll, um die Anzeige der Karte nicht zu stören und den Benutzer visuell zu unterstützen.

Nach der Speicherung der Änderungen soll das Popup geschlossen, die Karte aktualisiert, das Feature orange eingefärbt und die Änderungen im Frontend zwischengespeichert werden. Eine Veränderung des Vektors geschieht vollständig im Frontend. Die aktualisierten Daten werden erst bei der Ausführung einer Berechnung an das Backend gesendet. Ausserdem wird so vorerst die Sicherheit der Daten des Benutzers gewährleistet.

Lösung
Im Frontend der Applikation wurde eine Vue-Komponente für das "Popup" implementiert, die es ermöglicht, die Attribute eines Features zu bearbeiten. Dabei wird beim hochladen der Vektordaten ein "GeoJSON"-Objekt erstellt, das die Geometrie und alle Attribute der Features in der `FeatureCollection` enthält. Um lange Ladezeiten zu vermeiden, wird das Rendern der "Popups" dynamisch gesteuert, so dass erst bei einem Klick auf ein Feature die Attribute des Features geladen, das "Popup" gerendert und dieses dann angzeigt wird. Durch die sehr kurze Renderzeit der "Popups" merkt der Benutzer keinen Unterschied und das Frontend bleibt responsiv, weil die "Popups" nicht alle gleichzeitig im Hintergrund gerendert und in das Document Object Model (DOM) eingefügt werden. Verlässt der Benutzer das "Popup", entweder durch Schliessen des Fensters oder durch Speichern der Änderungen, wird das "Popup" aus dem DOM entfernt und die Attribute des Features werden anschliessend aktualisiert.

Der Code des Composables [75] `renderPopupComponent` (siehe 10.1) ist so gestaltet, dass er die Attribute des Features dynamisch lädt und das "Popup" rendert, wenn der Benutzer auf ein Feature klickt. Der `AppContext` beinhaltet Informationen über die Anwendung, sowie zur Ergänzung der Ant Design UI-Komponenten, die ebenfalls im "Popup" verwendet werden.

```
export const renderPopupComponent = (component: Component, props: Record<string, any>) => {
    const container = document.createElement('div')

    const vnode: VNode = createVNode(component, props)

    if (appContext) {
        vnode.appContext = appContext
    }

    render(vnode, container)

    return {
        container,
        unmount: () => {
            render(null, container)
        }
    }
}
```

Listing 10.1: Code des Composables `renderPopupComponent`

10.4. Berechnung mehrerer Gebiete parallelisieren

Um die Berechnung grösserer Datenmengen zu beschleunigen, ist es ein Ziel, die Berechnung von Z für mehrere Gebiete zu parallelisieren. In einer empirischen Untersuchung wurde die Machbarkeit untersucht und es werden verschiedene Ansätze umgesetzt und evaluiert.

Vorgehen und Herausforderungen
Ein erster Ansatz bestand darin, eine Processing Queue zu nutzen, wobei die Gebiete nacheinander parallel berechnet werden. Die Sortierung sollte dabei so erfolgen, dass stärker bebaute Gebiete, also solche mit einer grösseren Proportion an Gebäuden zur Gesamtfläche, zuerst abgearbeitet werden. Die Idee dahinter ist, dass die Berechnung von Gebieten mit mehr Gebäuden in der Regel länger dauert, da mehr Daten verarbeitet werden müssen. Dadurch soll die Gesamtberechnungszeit reduziert werden, da die Gebiete mit weniger Gebäuden schneller abgearbeitet werden können, während die komplexeren Gebiete parallel bearbeitet werden. Der Ansatz der Queue wurde jedoch verworfen, da die Sortierung der Gebiete einen zusätzlichen Aufwand generierte, weil die Rasterdaten schon zuvor sequenziell analysiert werden mussten, um den Anteil der Siedlungsfläche an der Gesamtfläche zu bestimmen. Die Lösung könnte jedoch als möglicher Ansatz dienen, wenn vom Benutzer ein Attribut dieses Anteils der Siedlungsfläche angegeben wird, wobei der zusätzliche Aufwand zur Bestimmung dieses Anteils wegfällt.

Der Ansatz mit der multiprocessing-Bibliothek von Python [15] wurde nach den Erläuterungen von [76] umgesetzt.

Eine weitere Herausforderung war es den Einsatz von GDAL [25] zu berücksichtigen. GDAL ist normalerweise nicht für die Verarbeitung von parallelen Threads ausgelegt, weil die Bibliothek intern auf einen globalen Zustand zugreift, der nicht thread-sicher ist. Für die Verarbeitung der Vektordatei ist das jedoch kein Problem, da jeder Thread ein Feature des Vektors verarbeitet und diesen nur ausliest. Das Öffnen der Rasterdatei in jedem Thread kann zu hohen Arbeitsspeicherauslastungen führen [77]. In der empirischen Untersuchung zeigte sich, dass es zu Problemen kommt, wenn sehr grosse Rasterdateien in mehreren Threads gleichzeitig geöffnet werden. Entweder wird mehr Arbeitsspeicher zugewiesen, oder die Anzahl der Threads wird begrenzt, um die Auslastung etwas zu reduzieren.

Evaluation der Parallelisierung
Die empirische Untersuchung wurde mit verschiedenen Ansätzen durchgeführt, um die Berechnungszeit zu messen und zu vergleichen. Die Ansätze wurden mit der multiprocessing-Bibliothek von Python [15], der Joblib-Bibliothek [14] und der Dask-Bibliothek [13] umgesetzt. Verglichen wurden die Laufzeiten der parallelisierten Berechnungen mit der sequenziellen Berechnung, um die Effizienz der Parallelisierung zu bewerten. Die Tests wurden mit Gebieten verschiedener Grösse und Anzahl der Features durchgeführt, um die Performance zu untersuchen.

Die Berechnungen wurden lokal auf einem Apple MacBook Pro mit einem M3 Pro Prozessor durchgeführt (vgl. Tabelle 10.2). Die Applikation lief während der Tests in einem Docker-Container, wobei der Shared Memory zunächst auf 4 und später auf 8 GB erhöht wurde, um die Performance in verschiedenen Konfigurationen zu testen. Auch verschiedene Konfigurationen der Anzahl der Threads wurden getestet, um die optimale Anzahl an Threads für die Berechnung in Kombination mit den Testdaten zu ermitteln.

	Macbook Pro [78]
Veröffentlicht	2023
Prozessor	Apple M3 Pro
Architektur	ARM64
Arbeitsspeicher	18 GB
OS (Betriebssystem)	macOS Sonoma 14.7.6

Tabelle 10.2: Hardwarekonfiguration der Rechner, auf denen der Benchmark durchgeführt wurde.

Folgende Vektordaten, wie in Tabelle 10.3 aufgeführt, wurden für den Benchmark ausgewählt. Die Auswahl wurde in den meisten Fällen auf Basis der Einwohnerzahl getroffen.

Anzahl Gebiete	Auswahlkriterien
< 10	Ausgewählte Gemeinden des Kantons Zürich, darunter Zürich und Winterthur, die bevölkerungsreichsten Gemeinden des Kantons.
50	Die 50 bevölkerungsreichsten Gemeinden der Schweiz, die in der Regel auch grösser sind und mehr Siedlungsfläche beinhalten.
150	Die 150 bevölkerungsreichsten Gemeinden der Schweiz.
500	Alle Gemeinden des Kantons Bern und des Kantons Zürich.
1000	Zufällig ausgewählte Gemeinden der Schweiz.
2148	Alle Schweizer Gemeinden (Stand 2022) [79].

Tabelle 10.3: Auswahl der Verschiedenen Vektorgrössen für die Benchmarks.

Resultate der Benchmarks
In der nachfolgenden Tabelle 10.4 sind die Resultate der Benchmarks für die verschiedenen Ansätze zusammengefasst.

Methode	Menge	Dauer (ms)		System	Anmerkungen
Sequenziell	< 10	19 986	(≈ 19 s)	M3 Pro	
Python multiprocessing		34 013	(≈ 34 s)		N_JOBS = 4
Joblib		33 566	(≈ 33 s)		N_JOBS = 4
Dask		32 918	(≈ 33 s)		N_JOBS = 4
Sequenziell	50	250 162	(≈ 4 min 10 s)		
Python multiprocessing		119 640	(≈ 1 min 59 s)		N_JOBS = 4
Joblib		118 889	(≈ 1 min 58 s)		N_JOBS = 4
Dask		125 443	(≈ 2 min 5 s)		N_JOBS = 4
Sequenziell	150	590 460	(≈ 9 min 50 s)		
Python multiprocessing		284 385	(≈ 4 min 44 s)		N_JOBS = 3
Joblib		283 085	(≈ 4 min 43 s)		N_JOBS = 3
Dask		284 175	(≈ 4 min 44 s)		N_JOBS = 3
Sequenziell	500	307 350	(≈ 5 min 7 s)		
Python multiprocessing		306 170	(≈ 5 min 6 s)		N_JOBS = 3
Joblib		309 612	(≈ 5 min 9 s)		N_JOBS = 3
Dask		319 558	(≈ 5 min 19 s)		N_JOBS = 3
Sequenziell	1000	1 473 360	(≈ 24 min 33 s)		
Python multiprocessing		882 511	(≈ 14 min 42 s)		N_JOBS = 2
Joblib		538 577	(≈ 8 min 58 s)		N_JOBS = 2
Dask		458 156	(≈ 7 min 38 s)		N_JOBS = 2
Sequenziell	2148	1 432 168	(≈ 23 min 52 s)		
Python multiprocessing		1 633 186	(≈ 27 min 13 s)		N_JOBS = 2
Joblib		1 623 724	(≈ 27 min 3 s)		N_JOBS = 2
Dask		1 660 999	(≈ 27 min 40 s)		N_JOBS = 2

Tabelle 10.4: Resultate des Benchmarks für die Parallelisierungsansätze mit einer verschiedenen Anzahl an Gebieten des Vektors.

Resultat

Die Entscheidung fiel auf den Einsatz von Joblib [14], einer Python-Bibliothek, die eine einfache Parallelisierung von Berechnungen ermöglicht. Anhand der Ergebnisse der empirischen Untersuchung kann gezeigt werden, dass die Berechnungszeit unter Einsatz von Joblib am besten ausfällt. Zur Umsetzung der Parallelisierung wurde die Funktion `Parallel` von Joblib verwendet, die es ermöglicht, mehrere Berechnungen gleichzeitig auszuführen. Die Funktion `Parallel` in Kombination mit der Funktion `delayed` von Joblib ermöglicht die Umsetzung von "Embarassingly Parallel"-Berechnungen, bei denen die einzelnen Berechnungen unabhängig voneinander durchgeführt werden können. Die Parallelisierung wird Threading-basiert umgesetzt. Wie `Parallel` konfiguriert werden kann, ist in der Dokumentation von Joblib [80] beschrieben.

Die Berechnungszeit für mehrere Gebiete konnte um bis zu 50% (vgl. Abbildung 10.2) reduziert werden, was eine signifikante Verbesserung darstellt. Der Einsatz wurde mit der Siedlungsfläche der gesamten Schweiz des Jahres 2022 getestet. Werden alle Schweizer Gemeinden (2148, Stand 2022) zur Berechnung herangezogen, so wurde in den Tests auf einem Gerät eine optimale Auslastung bei zwei Threads mit 8 GB Arbeitsspeicher erreicht.

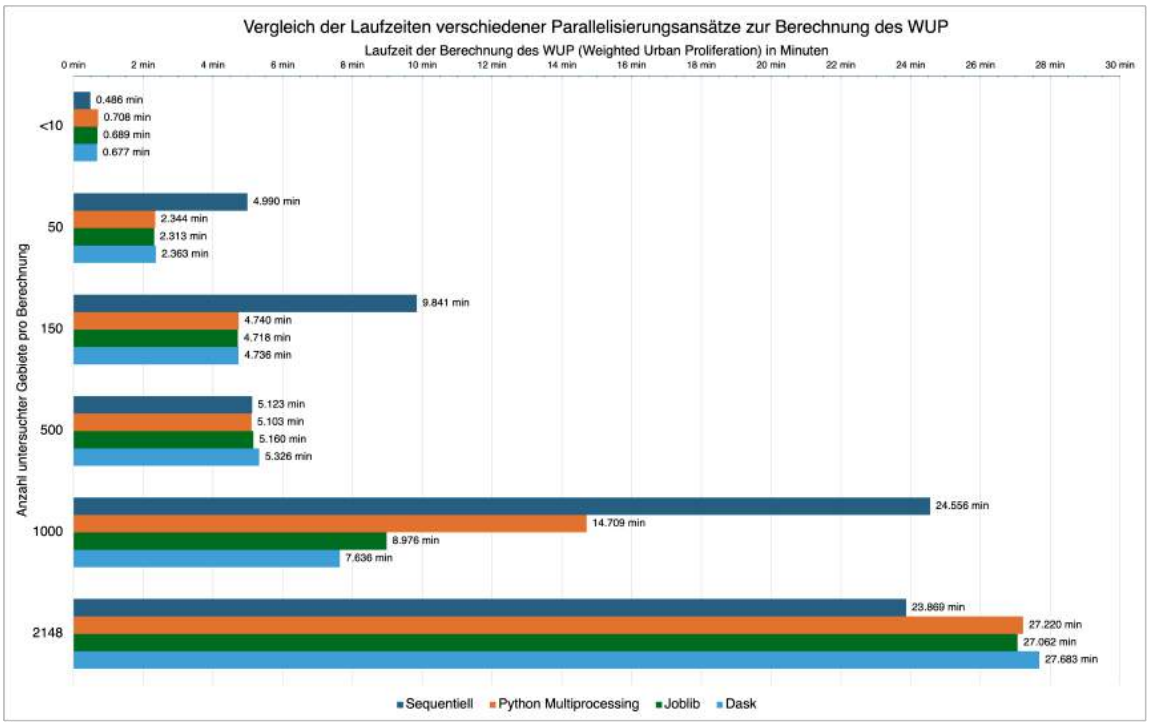


Abbildung 10.2: Resultate der Benchmarks für die Parallelisierung der Berechnung von Z. Die Berechnungszeiten sind in Minuten angegeben.

Die Leistung hängt auch stark von der Grösse des Rasters ab, da grössere Raster mehr Daten beinhalten die den Arbeitsspeicher belasten. Das zeigt sich auch in den Benchmark-Resultaten, wo die Berechnungszeiten bei kleinen Datensätzen (z.B. 10 Gebiete) kürzer in der sequenziellen Berechnung sind, da die Overhead-Kosten der Parallelisierung höher sind als die Einsparungen durch die Parallelisierung. Werden jedoch grössere Datensätze (z.B. 100 Gebiete) verwendet, so ist die Parallelisierung deutlich schneller, da die Overhead-Kosten der Parallelisierung im Vergleich zu den Einsparungen durch die Parallelisierung geringer sind. Der Einsatz muss daher immer im Kontext der Grösse des Datensatzes betrachtet werden, um die optimale Performance zu erzielen.

10.4.1. Gebiete des Vektors vereinigen

Eine Vereinigung von Gebieten des Vektors ist ein notwendiges Feature, um Z für zusammenhängende Gebiete als Ganzes zu berechnen. Die Umsetzung der Funktionalität soll sich an eine QGIS-ähnliche Bedienung anlehnen, um die Benutzerfreundlichkeit zu gewährleisten. In Abbildung 10.3 ist die Funktion "Gebiete verschmelzen" [29] aus QGIS dargestellt, die es ermöglicht, mehrere Gebiete des Vektors zu einem einzigen Gebiet zu vereinigen. Dabei können Strategien für die Zusammenführung der Attribute ausgewählt werden, wie z.B. die Summe der Werte oder die Auswahl des grössten Wertes.

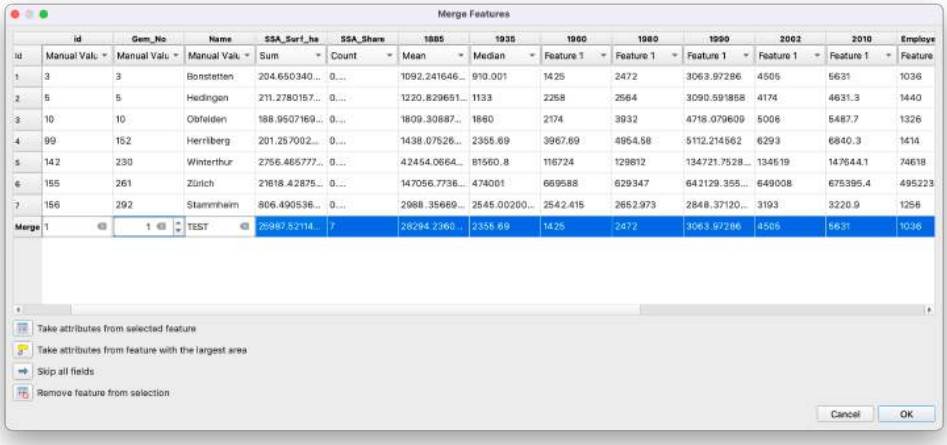


Abbildung 10.3: Die Funktion "Gebiete verschmelzen" in QGIS [29].

Das Fenster dient als Vorlage für die Umsetzung der Funktionalität im "USM Calculator".

Herausforderungen

Grundsätzlich besteht die Herausforderung darin, die Geometrien der Gebiete zusammenzuführen und die Attribute der Gebiete zu mergen.

Weil es sich bei den GeoJSON-Daten um ein standardisiertes JSON-Objekt handelt, könnte die Implementierung im Frontend mittels Typescript erfolgen. Dies würde auch für grosse GeoJSON-Daten funktionieren, weil die Datenmenge zur Darstellung der Geometrien und Attribute in der Regel nicht zu gross ist. Die Geometrien der Gebiete können nicht ohne den Einsatz einer weiteren Frontend-Bibliothek, die ein Mergen von Geometrien ermöglicht, zusammengeführt werden.

Wird der Merge im Backend durchgeführt, so müssen die Geodaten an das Backend übertragen werden, was bei grossen GeoJSON-Daten zu spürbaren Latenzzeiten führen kann. Jedoch erlaubt der bereits bestehende Einsatz von GDAL [25] im Backend, die Geometrien der Gebiete mit einer `Union`-Operation zusammenzuführen, was eine effiziente und performante Lösung darstellt. Des Weiteren ermöglicht eine Umsetzung im Backend die Anbindung an die REST-API, sodass auch andere Anwendungen die Funktionalität nutzen können. Die Entscheidung, ob die Funktionalität im Frontend oder im Backend umgesetzt werden soll, wird in Abbildung 10.4 in Form eines Y-Statements festgehalten.

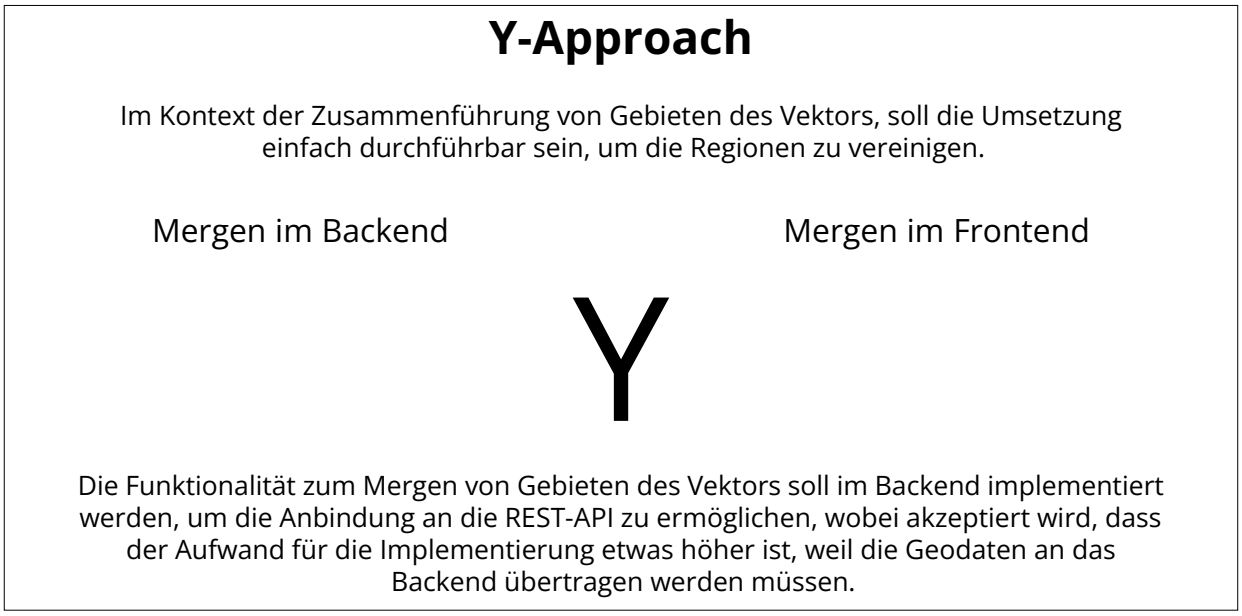


Abbildung 10.4: Y-Statement zur Umsetzung der Funktionalität zum Mergen von Gebieten des Vektors.

Lösung
Die Lösung wurde im Backend implementiert, um eine Anbindung der Funktionalität an die REST-API zu ermöglichen. Die Geometrien der Gebiete werden dabei mit der GDAL-Funktion `Union` zusammengeführt, die eine effiziente und performante Lösung darstellt. Die Attribute der Gebiete werden anschliessend gemäss der gewählten Strategie vereint. Der Rückgabewert besteht aus einem GeoJSON-Objekt mit nur einem einzigen Feature in der `FeatureCollection`, das die zusammengeführten Geometrien und Attribute enthält.

10.5. Weitere Anpassungen

10.5.1. Flächenberechnung des Untersuchungsgebiets

Die Berechnung des PBA (Anteil der Siedlungsfläche am Untersuchungsgebiet) wurde angepasst, um die Fläche in korrekten Einheiten (m^2) zu berechnen. Bisher wurde die Fläche aus den Geometriedaten der hochgeladenen Vektordaten berechnet, was zu fehlerhaften Ergebnissen führte, da die Geometriedaten in der Regel in einem CRS vorliegen, das nicht auf Metern basiert. Die Schweizerische Landeskoordinate (CH1903) ist ein Beispiel für ein solches CRS, das auf Metern basiert. Weil zur allgemeinen Kompatibilität, z.B. mit Leaflet im Frontend, die Geodaten beim Hochladen vor der Berechnung in das CRS EPSG:4326 (WGS84) transformiert werden, wurde die Fläche des Untersuchungsgebiets in diesem CRS berechnet. Folglich wäre die Fläche in Quadratgrad berechnet worden, was zu fehlerhaften Ergebnissen führt. Eine Lösung würde darin bestehen, die Fläche des Untersuchungsgebietes in dem CRS zu berechnen, in dem die Geodaten vorliegen, was aber die Kompatibiltät im Frontend mit Leaflet beeinträchtigen und den Umgang bei grösseren Datensätzen erschweren würde. Die Verantwortung läge beim Benutzer der Anwendung, die Geodaten in einem CRS zu speichern, das auf Metern basiert, um die Fläche korrekt berechnen zu können.

Die Entscheidung fiel auf die Möglichkeit die Grade in Metern mittels der UTM-Projektion [81] zu berechnen, um die Fläche in Quadratmertern zu erhalten. Das UTM-Koordinatensystem (Universal Transverse Mercator) ist ein globales Koordinatensystem, das die Erde in 60 Zonen von Pol zu Pol mit einer Breite von 6 Längegraden unterteilt, wobei jede Zone eine eigene Projektion verwendet. Obwohl die UTM-Projektion nicht für alle Gebiete der Welt geeignet ist und auch das Bundesamt für Landestopographie (Swisstopo) die Anwendung nicht empfiehlt [82], bietet sie eine gute verallgemeinerte Lösung, weshalb sie in dieser Arbeit verwendet wird.

Die Zone des Untersuchungsgebietes wird dabei nach dessen geometrischem Schwerpunkt bestimmt, der aus den Geometriedaten der hochgeladenen Vektordaten berechnet wird. Die Fläche des Untersuchungsgebiets wird anschliessend in Quadratmetern berechnet, indem die Geometriedaten in das UTM-Koordinatensystem transformiert und die Fläche der Geometrie berechnet wird.

10.5.2. Kompression der GeoTIFF-Dateien

Die von der Applikation generierten GeoTIFF-Dateien, die das SI-Raster enthalten, wurden im Rahmen dieser Erweiterung bei der Erstellung komprimiert. In der Ausgangslage wurden die GeoTIFF-Dateien unkomprimiert erstellt, was zu grossen Dateien führen kann und die Übertragung der Dateien im Frontend verlangsamt. Dieses Problem wurde erfasst und behoben, indem die GeoTIFF-Dateien mit dem DEFLATE-Kompressionsalgorithmus komprimiert werden. Die DEFLATE-Kompression ist ein verlustfreier Kompressionsalgorithmus, der eine gute Balance zwischen Kompressionsrate und Geschwindigkeit bietet. Die Kompression wird beim Erstellen des GeoTIFFs im Backend durchgeführt, indem die GDAL-Bibliothek [25], wie im unteren Codebeispiel gezeigt, verwendet wird.)

```python
from osgeo import gdal
import numpy as np

#  Der Fokus liegt speziell auf dem options-Parameter, der die Kompressionseinstellungen definiert.
driver = gdal.GetDriverByName('GTiff')
si_raster = driver.Create(
    path_output,
    bands=1,
    xsize=shape.columns,
    ysize=shape.rows,
    eType=gdal.GDT_Float32,
    options=[
        'COMPRESS=DEFLATE',
        'PREDICTOR=2',
        'ZLEVEL=9',
    ]
)
si_raster.GetRasterBand(1).WriteArray(np.asarray(result_matrix))
si_raster.SetGeoTransform(clipped_raster.GetGeoTransform())
si_raster.SetProjection(clipped_raster.GetProjection())
si_raster.FlushCache()
```

Listing 10.2: Auszug der `RasterClipper`-Klasse, zeigt die Erstellung des komprimierten GeoTIFFs

10.6. Automatisierte Testverfahren

Bereits im Rahmen der Studienarbeit wurden automatisierte Tests im Backend implementiert. Insbesondere für die Berechnung von Z wurden Tests implementiert, die sicherstellen, dass die Berechnung korrekt funktioniert und die Ergebnisse mit den erwarteten Werten übereinstimmen. Die Tests können mit dem Befehl `pytest` [46] ausgeführt werden und sind im Quellcode enthalten. Für die neue Funktionalität des Zusammenführens von Gebieten des Vektors wurden Tests hinzugefügt, die eine korrekte Funktionalität sicherstellen.

Die damals zur Verfügung gestellten Testdaten wurden durch neue Testdaten des ARE erweitert, die aktuellere und realistischere Daten enthalten. Das ARE stellt für den Zeitraum von 2019 bis 2024 Daten zur Siedlung der Schweiz zur Verfügung [36], woraus ein GeoTIFF erstellt wurde mit Rastergrösse von 15 Metern. Die Gemeindeflächen von Swisstopo [83] im Geopackage-Format mit den enthaltenen Attributen zur Einwohnerzahl wurden beibehalten.

Dank dieser Testdaten können zukünftige Entwickler die Tests selbstständig ausführen und die Resultate unabhängig überprüfen.

10.7. Usability-Tests

10.7.1. Zielsetzung

Mittels Usability-Tests soll sichergestellt werden, dass die Anwendung die Bedürfnisse der Zielgruppe erfüllt. Ausserdem sollen die Tests dazu dienen, die Abläufe von der Eingabe der Daten, ihrer Anpassung, bis zur Visualisierung der Ergebnisse zu überprüfen und zu optimieren.

10.7.2. Ablauf

In diesem Abschnitt wird der Ablauf der Usability-Tests festgelegt und die Vorgehensweise beschrieben. Das Testing besteht aus drei Phasen, die nacheinander durchgeführt werden.

Vorbereitungsphase
Dabei werden zuerst Szenarien definiert, die ein Benutzer durchlaufen soll. Ohne Szenarien können keine Schlussfolgerungen aus den Tests gezogen werden, da jede Testperson ohne ein konkretes Ziel im Testablauf agieren würde. Ein Szenario sollte dabei so gestaltet sein, dass eine Testperson eine als Ganzes durchführbare, konkrete Aufgabe zu lösen hat, die sie in der Anwendung durchführen soll.

In einem zweiten Schritt wird der Protoyp der Anwendung erstellt, auf dem die Tests durchgeführt werden und die Testpersonen die Szenarien durchlaufen werden. Der Prototyp kann dabei in Form eines Mockups, also kein fertiges Produkt, sondern eine visuelle Darstellung der Anwendung, erstellt werden. Dieser Prototyp kann digital (z.B. mit Figma [84]) oder physisch (z.B. auf Papier) erstellt werden. Natürlich ist eine digitale Version für die Testpersonen einfacher zu bedienen, da sie direkt mit der Anwendung interagieren können. Jedoch übersteigt der Aufwand für die Erstellung eines digitalen Prototypen oft den Nutzen innerhalb einer vorgegeben Zeitspanne. In dieser Arbeit wurde ein analoger Prototyp in Form von mehreren Papierbögen erstellt, welche die einzelnen Seiten und Elemente der Anwendung darstellen. Während des Tests ”klicken” die Testpersonen mit einem Stift auf die Elemente des Prototypen (z.B. Schaltflächen, Eingabefelder), um die Interaktion mit der Anwendung zu simulieren. Anschliessend wird ein neuer Papierbogen mit dem nächsten Schritt vorgelegt, der die nächste Seite oder den nächsten Schritt des Szenarios darstellt.

Im dritten Schritt der Vorbereitungsphase wird eine Testtabelle erstellt. Diese Tabelle enthält Fragen und eine Auswahl an Antworten, wobei die passende Antwort vom Beobachter des Tests angekreuzt wird. Wichtig ist, dass die gewünschten Erkenntnisse mit Fragen abgedeckt werden und dass diese schnell beantwortet werden können. Der Beobachter des Tests sollte jeden Schritt und jede Überlegung der Testperson aufzeichnen, um den Ablauf und die Erkenntnisse des Tests möglichst lückenlos zu dokumentieren.

Durchführung und Szenarien
Die UX-Tests für den ”USM Calculator” wurden mit einem Prototypen aus Papier durchgeführt, der die Benutzeroberfläche der geplanten Anwendung darstellt.

Zu Beginn des Tests wird der Testperson das Vorgehen des Tests erklärt. Die Hauptaufgabe der Testperson besteht darin, ihre Gedanken laut auszusprechen, während sie die Szenarien durchläuft. Durch das laute Aussprechen der Gedanken kann der Beobachter des Tests die Überlegungen und Entscheidungen der Testperson nachvollziehen und besser verstehen. Missverständnisse der Szenarien können so direkt erkannt und korrigiert werden.

Die Anforderungen der Erweiterung des”USM Calculators” wurden in vier Szenarien aufgeteilt, welche die Testperson nacheinander durchlaufen soll. Die Abbildungen (vlg. 10.5, 10.6, 10.7 und 10.8) zeigen den Prototypen mit den ausgedruckten Szenarien.

Szenario 1:

> Für den Anfang willst du herausfinden, wie die aktuellen Zersiedelungswerte der Region des letzten Jahres lauten. Dafür füllst du nur die zwingenden Felder aus. Die berechneten Daten sollen bei mir gespeichert werden.

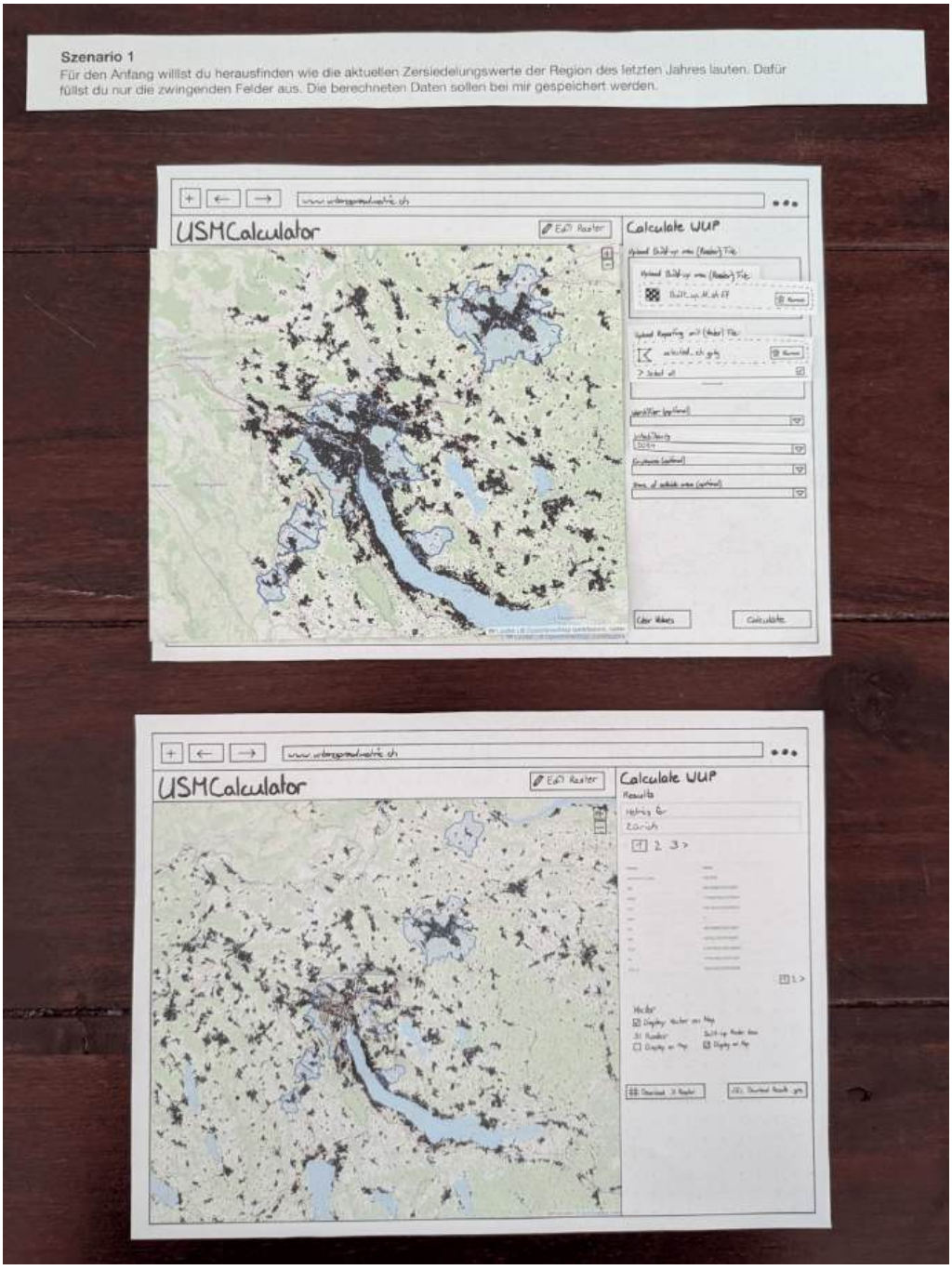


Abbildung 10.5: Szenario 1 testet die Ansicht der Statistikdaten.

Szenario 2:

> Im nächsten Schritt willst du herausfinden, sie sich der Zersiedelungswert der Region des letzten Jahres ändert, wenn die geplante Siedlung im bebauten Gebiet eingezeichnet wird. Gleichzeitig wurde eine grosse Überbauung in der Region abgerissen, was du ebenfalls in der Planung berücksichtigen willst.

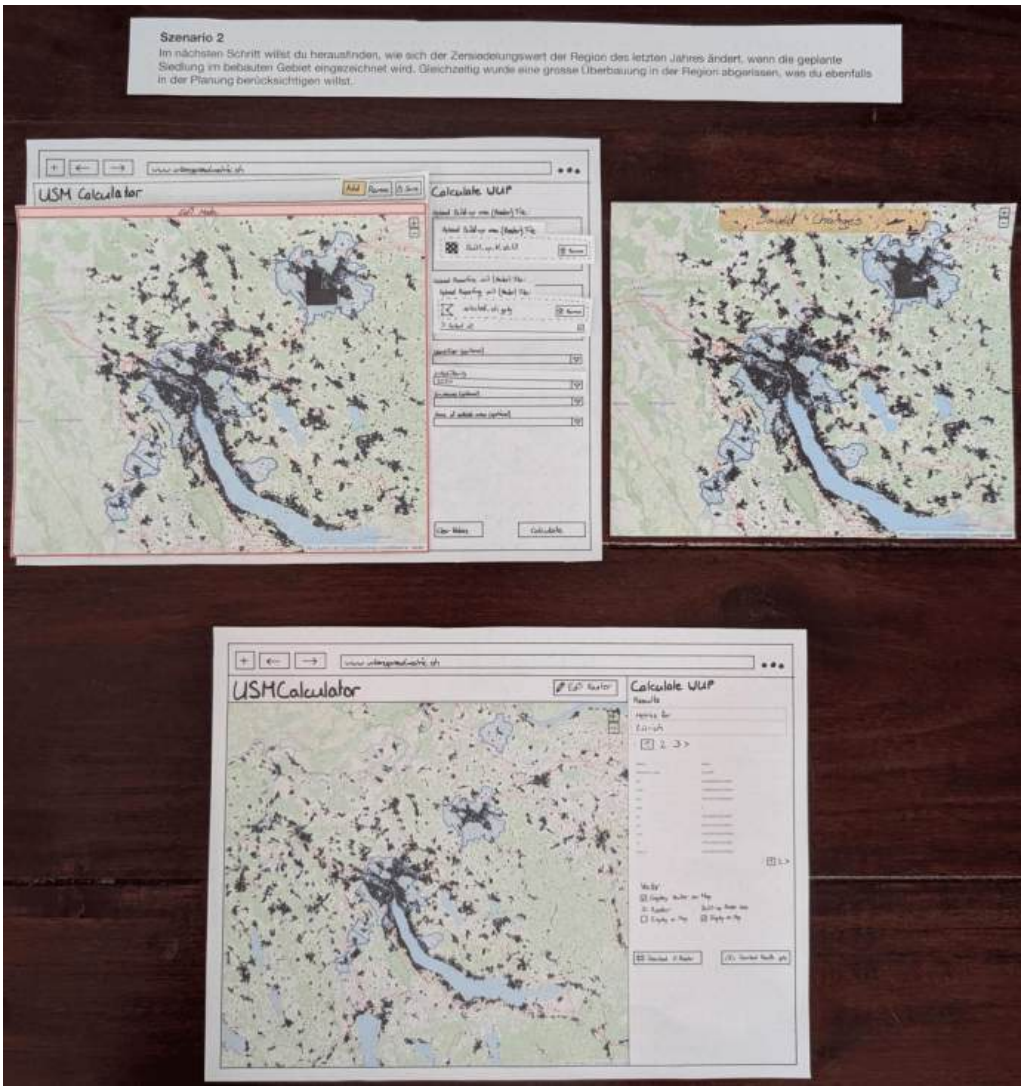


Abbildung 10.6: Szenario 2 testet die Bearbeitung der Rasterdaten.

Szenario 3:

> Um ebenfalls die Auswirkung der erwarteten Bevölkerungszuwachs (ca. 2'500 Personen) in diesem Jahr aufzuzeigen, willst du für Zürich die Statistikdaten erweitern. Auch für dieses Beispiel willst du die Zersiedelungswerte präsentieren können.

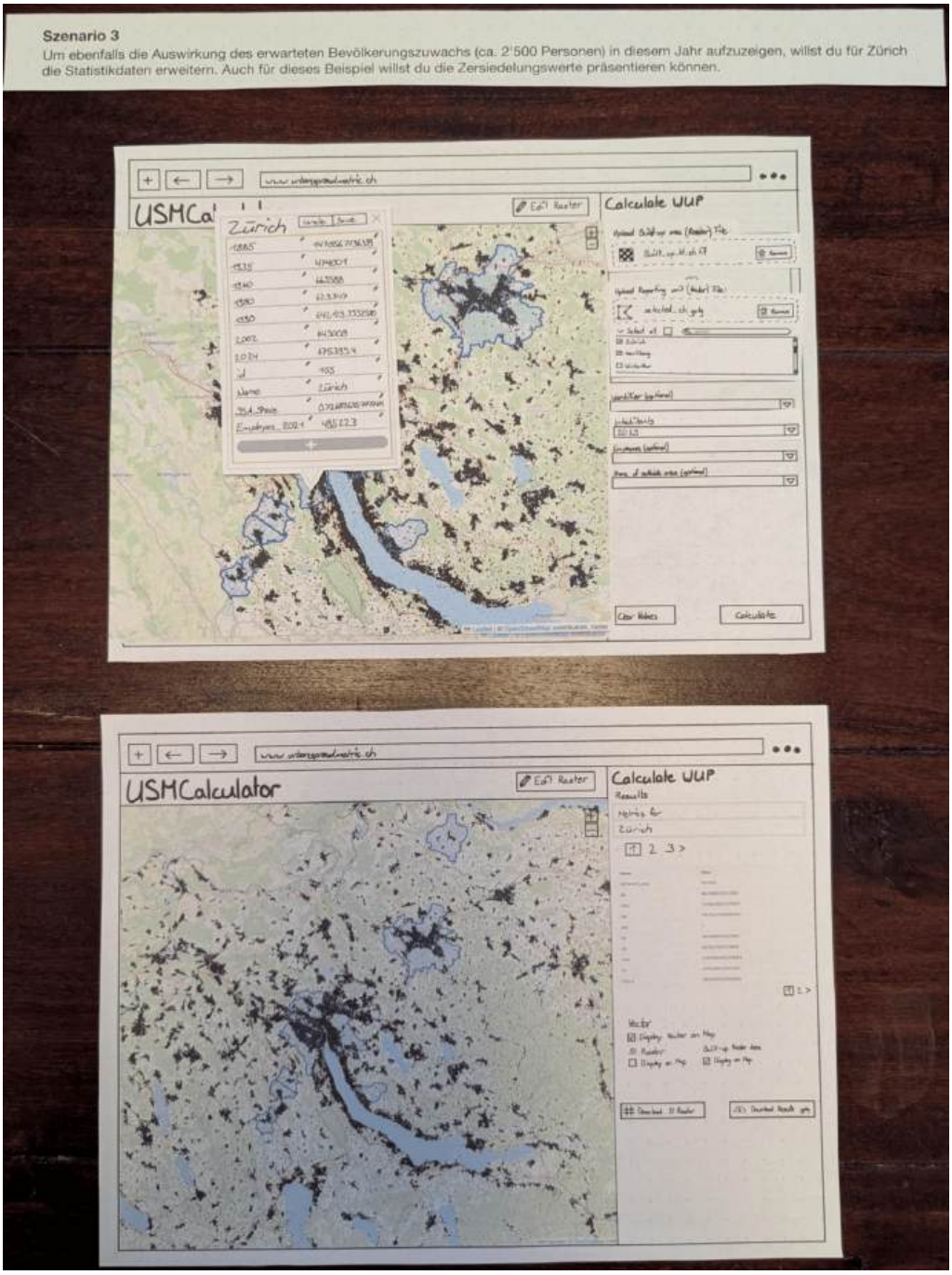


Abbildung 10.7: Szenario 3 testet die Bearbeitung der Statistikdaten.

Szenario 4:

> Um einen Zersiedelungswert für die gesamte Region in einem Wert darzustellen, sollen die Statistikdaten der Gemeinden der Region zusammengeführt werden. Zum Beispiel werden die Einwohnerzahlen der Gemeinden summiert.

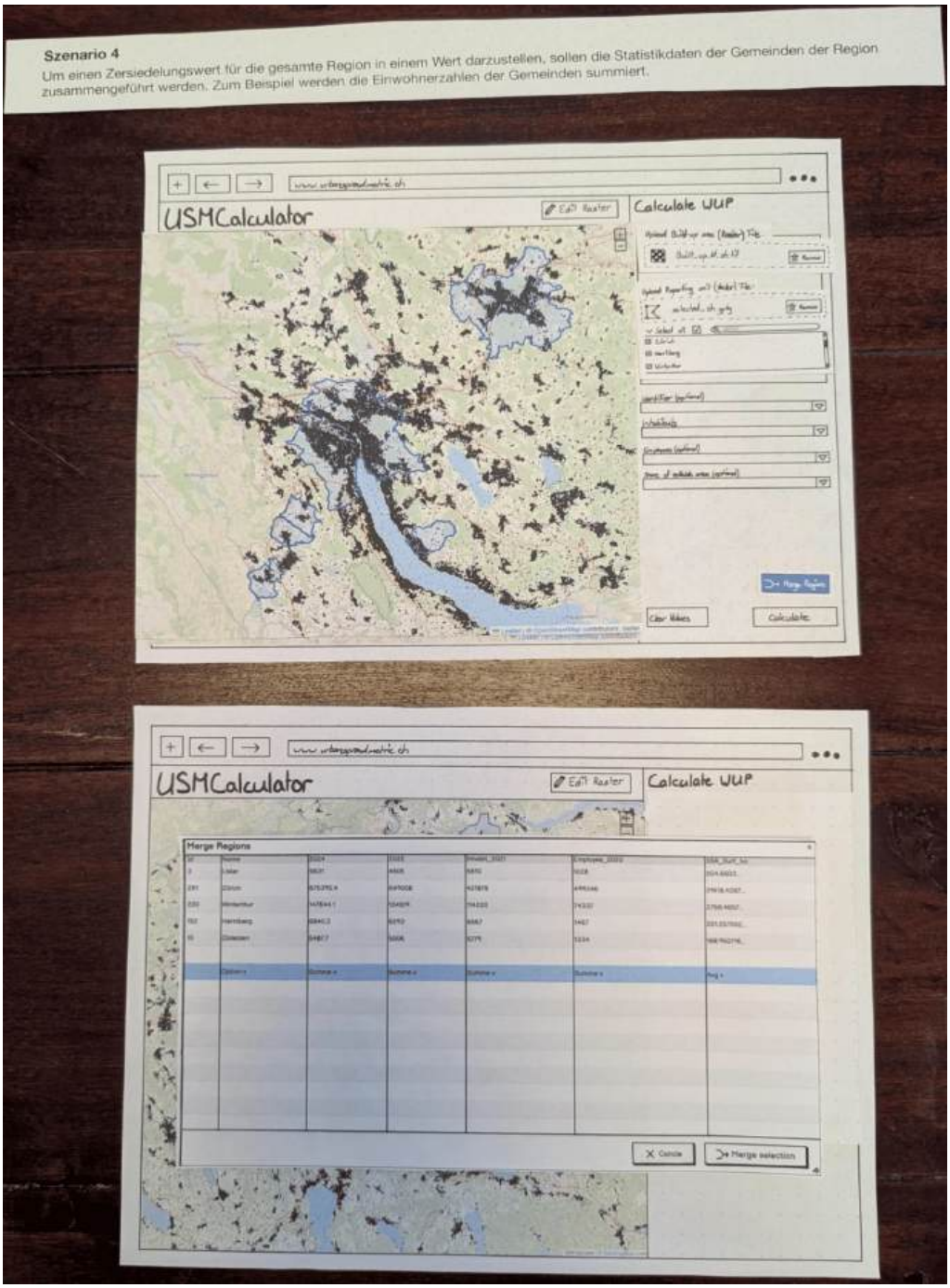


Abbildung 10.8: Im Szenario 4 wird die Zusammenführung der Regionen zu einer Gesamteinheit getestet.

Die Testtabelle, die während des Tests ausgefüllt wird, enthält einige Fragen, die der Beobachter des Tests beantworten soll, während die Testperson laut denkend die Szenarien durchläuft. Eine Auswahl an Fragen, die in der Testtabelle enthalten sind, ist wie folgt:

- Kann die Testperson eine Berechnung durchführen?
- Erkennt die Testpersonen Buttons?
- Kann die Testperson die Eingabefelder ausfüllen?
- Weiss die Testperson, auf welcher Seite sie sich befindet?
- Erkennt die Testperson, dass sie mehrere Gebiete auswählen kann?
- Erkennt die Testperson, dass sie Gebiete zusammenführen kann?
- Kann die Testperson die Rasterdaten bearbeiten?
- Kann die Testperson die Statistikdaten bearbeiten?
- Kann die Testperson die Dateien herunterladen?

Auswertung
Es wurden insgesamt drei Testpersonen ausgewählt, welche die Anwendung mit den definierten Szenarien durchlaufen haben. Vertreten sind Personen im Alter von 20 bis 60 Jahren. Die Kenntnisse beliefen sich von ”Grundkenntnisse für den Alltag” bis ”vertiefte Kenntnisse der Arbeit”.

10.7.3. Ergebnisse

Die konkreten Ergebnisse der Tests werden nicht im Detail beschrieben. Stattdessen werden die wichtigsten Erkenntnisse zusammengefasst, die aus den Tests gewonnen wurden und die in der Umsetzung der Anwendung berücksichtigt wurden.

Diese Erkenntnisse sind wie folgt:

- Texte der Buttons und Beschriftungen müssen mit beschreibenden Namen versehen werden.
- Einheitliche Icons sollten für die Buttons und Beschriftungen verwendet werden.
- Einige Buttons, wie z.B. der ”Merge regions”-Button, wurden in die Nähe der Vektordaten-Auswahl verschoben, um deren Zusammengehörigkeit zu verdeutlichen.
- Im Resultatsscreen wird ein zusätzlicher Button hinzugefügt, um zur Berechnungsseite mit den vorher eingegebenen Daten zurückzukehren. Zuvor mussten Benutzer alle Inputdaten erneut eingeben, um eine neue Berechnung durchzuführen.
- Das Dropdown-Menü für die Gebiets-Auswahl wurde prägnanter platziert und ist standardmässig für alle Gebiete ausgewählt.

11

Projektmanagement

11.1. Vorgehen

Das Projekt wird nach dem Rational Unified Process (RUP) durchgeführt. Das Modell ist iterativ und besteht aus vier Phasen, die im untenstehenden Abschnitt beschrieben sind. Der Plan nach RUP ist in Abbildung 11.1 dargestellt. In den grünen Boxen sind jeweils die Nummern der Meilensteine aus Abschnitt 11.3 eingetragen.

Abbildung 11.2 zeigt einen genaueren Zeitplan für die einzelnen Phasen des Projekts. Sämtliche Tätigkeiten sind in einer Matrix dargestellt, um aufzuzeigen, wie viel Zeit pro Tätigkeit aufgewendet wird.

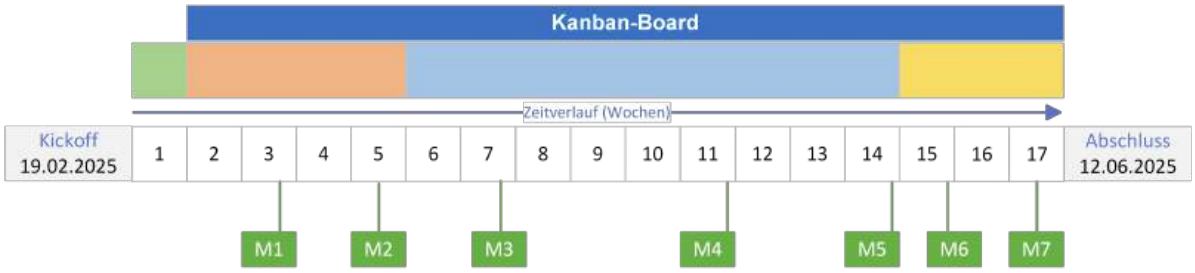


Abbildung 11.1: Plan nach RUP.

Eine detaillierte Planung wurde erstellt, indem die Tätigkeiten in kleinere Aufgaben mit einer Deadline und verantwortlichen Person aufgeteilt wurden. Diese Aufgaben wurden mittels Microsoft Teams in einem Kanban-Board [85] organisiert und überwacht.

Tätigkeit	Anz. Wochen	Inception	Elaboration				Construction									Transition		
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Projektmanagement	14 - 16																	
Projektplanung	2 - 3																	
Tooling / Infrastruktur	1 - 2																	
Requirements Engineering	4 - 5																	
Software Architektur	4 - 5																	
Testing und Bugfixing	2 - 4																	
Admin (Dokumentation, Meetings, Reviews, Refinement)	14 - 17																	
Zersiedelung für mehrere Gebiete (Gemeinden) berechnen	4 - 5																	
Inputdaten editierbar - Statistische Daten und Raster können angepasst werden	7 - 9																	
Verschmelzen von Gebieten (Gemeinden) und deren Statistikdaten für die Berechnung	4 - 5																	

Abbildung 11.2: Detaillierte Planung des Projekts.

11.2. Phasen

Die Phasen des Projekts sind in der untenstehenden Tabelle 11.1 aufgeführt.

Phase	Beschreibung
Inception	Projektinitialisierung. Die Vision wird erstellt und die Machbarkeit wird geprüft.
Elaboration	Detaillierte Planung des Projekts. Anforderungen werden spezifiziert und die Architektur wird definiert.
Construction	Umsetzung. Es wird entwickelt und getestet.
Transition	Abschluss. Das Projekt wird ausgeliefert und in Betrieb genommen.

Tabelle 11.1: Phasen des Projekts nach RUP.

11.3. Meilensteine

ID	Meilenstein	Datum	Beschreibung
M1	Projektplanung bereit	07.03.2025	Der Projektplan ist bereit für die Umsetzung. Der RUP-Plan und die Use Cases sind definiert. Der Server und weitere Softwareentwicklungs-Tools sind eingerichtet.
M2	Evaluation der Libraries für Rasterbearbeitung und Parallelisierung	19.03.2025	Der Vergleich mehrerer unterschiedlicher Libraries ist abgeschlossen. Es ist klar welche Softwarebibliotheken eingesetzt werden.
M3	Zersiedelung für mehrere Gebiete berechnen	03.04.2025	Im Frontend können mehrere Gebiete hochgeladen, ausgewählt und parallel berechnet werden.
M4	Statistische Daten und Raster können angepasst werden	02.05.2025	Die Bearbeitung der statistischen Daten ist im Frontend möglich. Diese überarbeiteten Daten können für die Berechnung verwendet werden.
M5	Verschmelzen von Gebieten um diese zu berechnen	23.05.2025	Die Statistikdaten und Grenzen von mehreren Gebieten werden zusammengefasst (verschmelzt) und als ein gesamtes Gebiet berechnet.
M6	Code Freeze	30.05.2025	Alle Bugs sind behoben. Der Code enthält alle Features aus dem definierten Umfang und wird nicht weiter angepasst.
M7	Finale Abgabe	11.06.2025	Die Dokumentation und das Produkt sind gemäss des definierten Umfangs fertiggestellt und bereit zur Abgabe.

Tabelle 11.2: Meilensteine des Projekts.

11.4. Rollen und Verantwortlichkeiten

Folgende Rollen sind im Rahmen dieser Arbeit definiert:

Rolle	Beschreibung	Interesse
Betreuer	Prof. Stefan Keller, Joël Schwab	Erfolgreiche Durchführung des Projekts, erfolgreiche Arbeit.
Industriepartner	Yves Maurer Weisbrod, Bundesamt für Raumentwicklung (ARE)	Ein in der Praxis einsetzbares Produkt.
Entwicklungsteam	Nico Fehr, Kyra Maag	Ein für die Raumplanung brauchbares Produkt zu erstellen, erfolgreiche Arbeit.
Product Owner (PO)	Kyra Maag	
Architekt	Nico Fehr	

Tabelle 11.3: Rollen im Projekt und deren Interesse.

Die Rolle des ”Product Owner” (PO) ist verantwortlich für die Einhaltung der Planung und die Übersicht des Projekts. Alle Teammitglieder sind für die Entwicklung und Umsetzung des Projekts zuständig. Der Architekt ist verantwortlich für die technische Umsetzung und die Architektur des Projekts.

11.5. Risikomanagement

11.5.1. Risikomatrix

Die Risikomatrix, wie in Abbildung 11.3 dargestellt, dient als Hilfestellung zur Beurteilung der Risiken. Sie zeigt die Eintrittswahrscheinlichkeit und die Auswirkung der Risiken auf das Projekt.

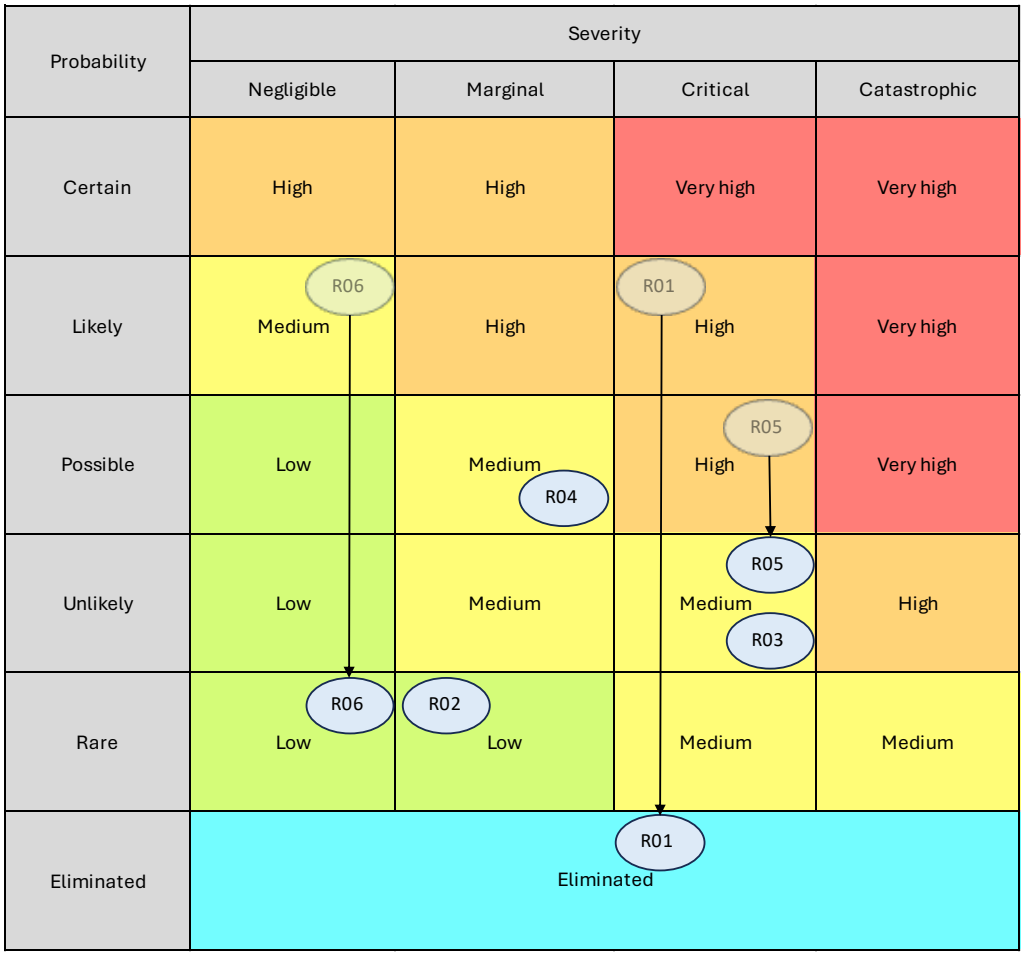


Abbildung 11.3: Aufstellung der Risiken in einer Matrix.

11.5.2. Risikoliste

Zur vollständigen Erfassung der Risiken des Projekts wurde eine Liste in Excel erstellt. Eine schlankere Version dieser Liste ist in der Tabelle 11.4 dargestellt. Jede Phase des Projekts beinhaltet einen entsprechenden Puffer, um die Risiken abzufedern.

ID	Risiko	Wahrscheinlichkeit	Schwere	Risikolevel	Aufwand	Strategie
R-01	Rasterbearbeitung über den Browser ist technisch zu anspruchsvoll oder nicht umsetzbar.	Eliminated	-	Eliminated	50h	Eine frühzeitige Evaluation möglicher Lösungen ist wichtig. Sind alle Lösungen zu anspruchsvoll werden die Resultate dokumentiert und die Anforderung wird verworfen.
R-02	Parallelisierung der Berechnung einzelner Gebiete ist technisch nicht oder nur schwer umsetzbar.	Rare	Marginal	Low	8h	Mehrere Technologien werden evaluiert um die Berechnung zu parallelisieren. Ist die Verwendung von Drittanbieter-Libraries zu komplex, kann auf die Python-Standardbibliothek zurückgegriffen werden.
R-03	Datenschutz kann nicht gewährleistet werden.	Unlikely	Critical	Medium	40h	Die Beschaffenheit und Art der schützenswerten Daten müssen untersucht werden. Allfällige Datenbanken und Schnittstellen müssen nach geltenden Sicherheitsstandards geschützt werden.
R-04	Teammitglieder mit weniger Erfahrung brauchen länger für gewisse Aufgaben.	Possible	Marginal	Medium	30h	Anwendung sinnvoller Software Engineering Praktiken wie Extreme Programming und Pairprogramming. Der Lerneffekt wird verstärkt (Learning by Doing). Erfahrene Entwickler können durch die direkte Weitergabe ihres Wissens dazulernen.
R-05	Performance der Applikation leidet zu stark bei der Rasterbearbeitung.	Unlikely	Critical	Medium	24h	Es wird nach einer performanten Lösung gesucht, die keine Auswirkungen auf die Bedienbarkeit hat. Tests mit Benutzern helfen die Responsivität zu messen. Alternativ müssen weitere performanceoptimierende Technologien in Betracht gezogen werden um die Anzeige und Bearbeitung des Rasters flüssig zu ermöglichen.
R-06	Statistikdaten können nicht sinnvoll zusammengefasst werden.	Rare	Negligible	Low	20h	Die Verantwortlichkeit zur korrekten Zusammenfassung von Statistikdaten kann an Benutzer des Tools delegiert werden. Mithilfe der Webseite kann ein Benutzer bei der Zusammenfassung der Statistikdaten unterstützt werden.

Tabelle 11.4: Risiken des Projekts und deren Bewertung.

11.5.3. Organisation

Die Risiken werden in der Risikoliste in der Tabelle 11.4 erfasst und bewertet. Mit Microsoft Excel wird die Risikomatrix in Abbildung 11.3 erstellt.

11.5.4. Änderungsprotokoll

In der nachfolgenden Tabelle 11.5 werden Änderungen an der Risikoeinschätzung dokumentiert.

Datum	Risiko ID	Änderung	Grund	Neues Risikolevel
16.04.2025	R01	Risiko konnte vollständig eliminiert werden	Die Rasterbearbeitung im Browser konnte mittels eigens umgesetzter Logik und der Verwendung von GeoTIFF-Libraries erfolgreich umgesetzt werden. Die Bearbeitung geschieht flüssig und funktioniert auch mit grösseren Dateien (Fläche der Schweiz).	Eliminated
16.04.2025	R05	Reduzierte Wahrscheinlichkeit von ”Possible” zu ”Unlikely”	Die Performance der Applikation wird durch die Rasterbearbeitung auch bei grösseren Dateien nicht negativ beeinflusst. In einigen Fällen (z.B. ein Raster von Italien) kann das Raster nicht mehr im Browser bearbeitet werden, da es zu viele Pixel sind. Eine entsprechende Einschränkung im Frontend verhindert den Upload solcher Raster.	Medium
28.05.2025	R06	Reduzierte Wahrscheinlichkeit von ”Likely” zu ”Rare”	Die Statistikdaten können mithilfe vordefinierter Strategien (z.B. ”Mittelwert”) sinnvoll zusammengefasst werden. Im Frontend wird der Benutzer bei der Zusammenfassung unterstützt, eine Vorschau der zusammengefassten Statistikdaten wird angezeigt. Es kann auch eine manuelle Zusammenfassung der Statistikdaten vorgenommen werden. Das Risiko, dass die Statistikdaten nicht sinnvoll zusammengefasst werden können, ist somit stark reduziert.	Low

Tabelle 11.5: Änderungsprotokoll für Risiken während der Projektlaufzeit.

11.6. Qualitätssicherung

Die Qualitätssicherung des Projekts stellt sicher, dass die Software den Anforderungen entspricht und die Funktionalität wie erwartet funktioniert. Der Ablauf der Berechnungen und die Werte der Zersiedelung müssen valide sein. Weil die Software als Open-Source-Projekt entwickelt wird, sollte die Qualität des Codes hoch sein und die Entwickler sollten sich an definierte Richtlinien halten.

11.6.1. Code-Richtlinien

Bereits in der Studienarbeit [8] wurden Code-Richtlinien definiert, die für die Weiterentwicklung des Prototyps übernommen werden und zur Darstellung erneut aufgeführt werden.

- Verwenden von aussagekräftigen Variablen- und Funktionsnamen
- Selbstbeschreibender Code - Kommentare für komplexe Funktionen
- Einhaltung der PEP8 Richtlinien für Python
- CamelCase für Variablen- und Funktionsnamen in Typescript
- Verwendung von Linters für Typescript und Python
- Code wird so geschrieben, dass er einfach und modular testbar ist (Unit-Tests).
- Kein kopierter Code - Wiederverwendung von Funktionen
- Single Responsibility Principle (SRP) [86] - Eine Funktion macht nur eine Sache
- Selbsterklärende Fehlermeldungen werden verwendet
- Typescript Types werden verwendet, um Typsicherheit zu gewährleisten
- ”Keep it simple, stupid” (KISS) [87] - Einfachste Lösung ist die beste Lösung

11.6.2. Gitlab CI/CD

Zur Umsetzung der Qualitätssicherung wird die Gitlab CI/CD Pipeline verwendet. Aktionen wie Tests, Linting und Deployment können automatisiert werden, so dass der Code nach jeder Änderung im Repository überprüft wird. Drei statische Code-Analyse-Tools werden in der Pipeline verwendet. Diese Tools dienen dazu, frühzeitig Fehler und Unstimmigkeiten im Code mittels vordefinierten Regeln zu erkennen.

In der Pipeline wird nach der Linting-Stage die Test-Stage ausgeführt.

flake8
Flake8 [55] erkennt Pogrammierfehler und Code Smells in Python-Code. Dabei wird auf die Einhaltung der PEP8-Richtlinien [56] geachtet. Die maximale Codezeilenlänge beträgt 140 Zeichen.

pylint
Pylint [57] übernimmt die selbe Aufgabe wie Flake8, kann aber zusätzlich Verbesserungsvorschläge anbieten. Eine Konfiguration in der Pipeline setzt die maximale Zeilenlänge auf 140 Zeichen, die Anzahl der Argumente auf 10 und die Anzahl der Statements pro Funktion auf 100.

mypy
Mypy [58] dient als Type-Checker für Python. Python ist eine dynamisch typisierte Sprache, was bedeutet, dass Typen zur Laufzeit überprüft werden. Diese Lücke schliesst Mypy, indem es statische Typenüberprüfung mit der Python-Standardbibliothek `typing` [88] ermöglicht. Tests werden in der Pipeline von Mypy ignoriert.

pytest
Pytest [46] ist ein beliebtes Test-Framework für Python, das die Ausführung von Unit-Tests vereinfacht.

vue-tsc
Der Typescript-Compiler für Vue.js, Vue-tsc [89], wird in der Pipeline verwendet, um die Typen des Frontend-Codes vor dem Build zu überprüfen. In Typescript werden explizite Typen verwendet, um die Code-Qualität zu erhöhen und Fehler bei Datenzugriffen frühzeitig zu erkennen.

eslint
Code Smells und Programmierfehler im Frontend-Code werden mittels ESLint [59] erkannt.

11.6.3. Git Feature Branch Workflow

Gearbeitet mit dem Code wird nach dem Git Feature Branch Workflow [20]. Jede neue Funktionalität oder Bugfix wird in einem separaten Branch entwickelt. Nach dessen Fertigstellung wird ein Merge Request erstellt, um den Code in den ”develop” Branch zu integrieren. Nur ein Merge-Request kann den geschützten ”develop” sowie ”main” Branch verändern. Zum Schluss der Arbeit wurde der ”develop” Branch in den ”main” Branch gemerged.

11.6.4. Testkonzept

Mittels einer Teststrategie, welche die Qualität, Performance, Funktionalität und Kompatibilität der Software sicherstellt, wird die Qualitätssicherung des Projekts gewährleistet.

Unit-Tests
Im Kapitel 10 werden die Automatisierten Tests 10.6 beschrieben. Die Unit-Tests sind ein wichtiger Bestandteil der Qualitätssicherung und werden mit dem Test-Framework Pytest [46] umgesetzt. Sie überprüfen die Funktionalität einzelner Komponenten im Backend und die Berechnungsschritte des ”USM Calculators”.

System-Tests
Ein System-Test überprüft die Funktionalität eines gesamten Systems. Die Umgebung für die Tests soll dabei möglichst nahe der Produktionsumgebung sein. Um die System-Tests zu dokumentieren, wurde ein Testprotokoll erstellt. Die System-Tests werden manuell durchgeführt und überprüfen die Funktionalität des Frontends und der API.

Mittels Smoke-Tests wird sichergestellt, dass die grundlegenden Funktionen des Systems funktionieren. Dabei wird vor allem überprüft, ob Front- und Backend erreichbar sind.

Mittels in der Swagger UI eingebauten Funktion um Endpunkte abzufragen, wird die API manuell getestet. Swagger vereinfacht die Interaktion mit der API und standartisiert die Dokumentation visuell.

11.6.5. Testprotokoll

Das Testprotokoll zeigt die Ergebnisse und Aufstellung der System-Tests. Für jede Funktionalität, die ein Benutzer im Frontend nutzen kann, wurde ein Testfall definiert.

ID	Beschreibung	Ergebnis	Status
T01	Das Frontend startet und die Startseite wird korrekt angezeigt.	Erfolgreich	✓
T02	Das Backend startet und die API ist erreichbar.	Erfolgreich	✓
T03	Ein Geopackage mit mehreren Regionen kann hochgeladen werden und die Auswahl der Gebiete wird korrekt angezeigt.	Erfolgreich	✓
T04	Ein Geojson mit mehreren Regionen kann hochgeladen werden und die Auswahl der Gebiete wird korrekt angezeigt.	Erfolgreich	✓
T05	Ein Geopackage mit mehreren Regionen kann hochgeladen werden und ein Klick auf eine Region zeigt die dazugehörigen Statistikdaten an.	Erfolgreich	✓
T06	Ist ein Geopackage hochgeladen, können die Statistikdaten einer angeklickten Region im Frontend bearbeitet werden.	Erfolgreich	✓
T07	Werden die Statistikdaten einer Region im Frontend bearbeitet, kann in einer bestehenden Zeile ein Wert geändert werden.	Erfolgreich	✓
T08	Werden die Statistikdaten einer Region im Frontend bearbeitet, kann eine neue Zeile hinzugefügt werden.	Erfolgreich	✓
T09	Werden die Statistikdaten einer Region im Frontend bearbeitet, kann eine Zeile gelöscht werden.	Erfolgreich	✓
T10	Ein Raster im TIFF-Format kann hochgeladen werden und die Fläche wird korrekt in der Karte angezeigt.	Erfolgreich	✓
T11	Ein Raster im TIFF-Format kann hochgeladen werden und über den ”Edit”-Button wird der Bearbeitungsmodus aktiviert.	Erfolgreich	✓
T12	Im Bearbeitungsmodus des Rasters können die Pixel mit einem Klick auf die Karte hinzugefügt und entfernt werden.	Erfolgreich	✓
T13	Im Bearbeitungsmodus des Rasters können gemachte Veränderungen mit einem Klick auf den ”Save”-Button gespeichert werden.	Erfolgreich	✓
T14	Ist ein Geopackage hochgeladen, kann mit dem ”Merge”-Button ein neues Fenster geöffnet werden, in dem die Regionen zusammengefasst werden können.	Erfolgreich	✓
T15	Im ”Merge”-Fenster werden mit einem Klick auf ”Merge” alle Regionen zusammengefasst und auf der Karte angezeigt.	Erfolgreich	✓
T16	Im ”Merge”-Fenster können die Statistikdaten mittels der definierten Strategien (z.B. ”Mittelwert”) zusammengefasst werden.	Erfolgreich	✓
T17	Im ”Merge”-Fenster können die Statistikdaten manuell zusammengefasst werden.	Erfolgreich	✓
T18	Die Berechnung der Zersiedelung kann gestartet werden, wenn alle erforderlichen Parameter gesetzt sind.	Erfolgreich	✓
T19	Die Berechnung der Zersiedelung kann gestartet werden, wenn nur ”Inhabitants” gesetzt ist.	Erfolgreich	✓
T20	Eine gestartete Berechnung der Zersiedelung zeigt den Fortschritt in Prozent an.	Erfolgreich	✓
T21	Die Resultate der Berechnung der Zersiedelung werden im Frontend angezeigt.	Erfolgreich	✓

T22	Die Resultate der Berechnung der Zersiedelung können als Geopackage heruntergeladen werden.	Erfolgreich	✓
T23	Die Resultate der Berechnung der Zersiedelung können als SI-Raster im TIFF-Format heruntergeladen werden.	Erfolgreich	✓
T24	Die Resultate der Berechnung der Zersiedelung eines Gebiets mit mehreren Regionen können als Zip-Datei mit Inhalt aller SI-Raster im TIFF-Format heruntergeladen werden.	Erfolgreich	✓

Tabelle 11.6: Testprotokoll der System-Tests.

12

Projektmonitoring

12.1. Soll-Ist-Zeitvergleich

Während des gesamten Projekts wurden die in jeder Woche aufgewendeten Stunden in einem Excel-Sheet erfasst. Dabei wurde zwischen dem Aufwand für die Entwicklung und dem Aufwand für die Dokumentation unterschieden. Abbildung 12.1 zeigt den Soll-Ist-Zeit-Vergleich der Projektlaufzeit mit den Phasen nach RUP.

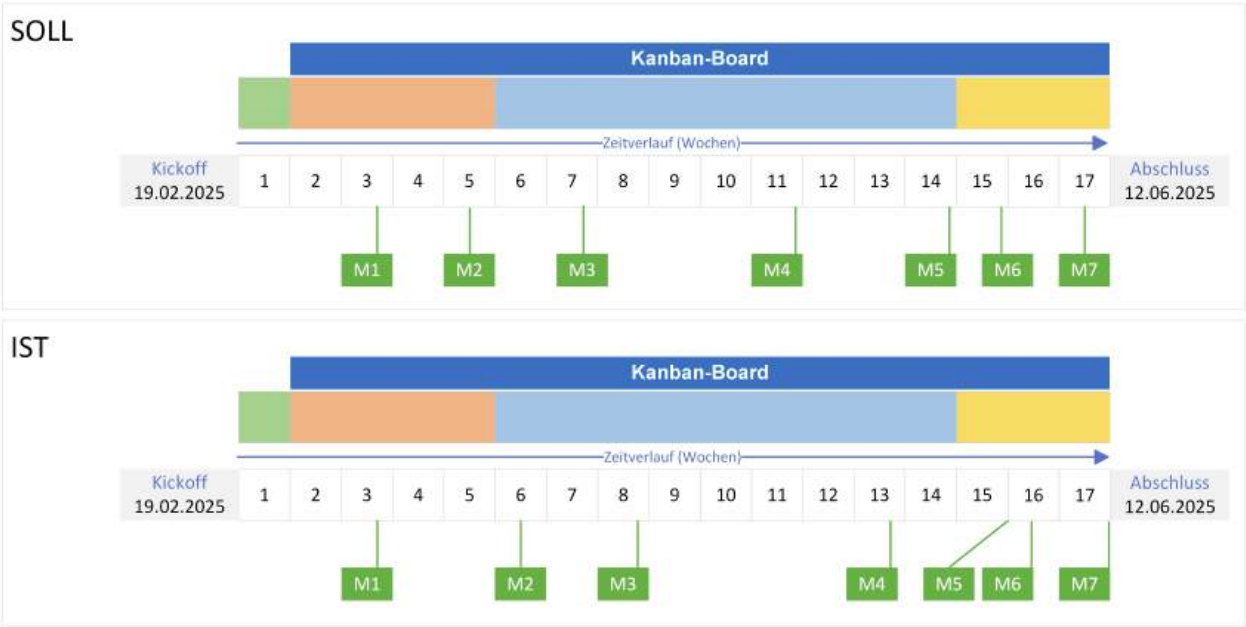


Abbildung 12.1: Vergleich Soll und Ist der Projektphasen nach RUP, Stand 11.06.2025

Der Abschluss des zweiten Meilensteins verzögerte sich um eine Woche, weil die Evaluation für die Rasterbearbeitung länger dauerte als geplant. Deren Umsetzung nahm ebenfalls mehr Zeit in Anpruch, wodurch sich der vierte Meilenstein um zwei Wochen verzögerte. Daraus resultierte eine Verschiebung des fünften Meilensteins um eine Woche. Dieser wurde jedoch schneller erreicht als geplant, da die Implementierung der Gebietsverschmelzung weniger Zeit in Anspruch nahm als erwartet.

Insgesamt wurden 737.6 Stunden für das gesamte Projekt aufgewendet. Davon entfielen 491.5 Stunden auf die Entwicklung und 246.1 Stunden auf die Dokumentation.

12.2. Zeitaufwand pro Person

Pro Person und ECTS-Punkt wird ein Aufwand von 30 Stunden erwartet. Daraus ergibt sich ein Soll-Aufwand von X Stunden pro Person. Der Gesamtaufwand in Stunden pro Person beläuft sich somit auf $12 \cdot 30h = 360h$. Der tatsächliche Aufwand pro Person ist in Tabelle 12.1 dargestellt.

Name	Zeitaufwand
Kyra Maag	352.7h
Nico Fehr	384.9h

Tabelle 12.1: Zeitaufwand pro Person, Stand 11.06.2025

12.3. Codestatistik

Im Rahmen dieser Arbeit wurden **3959** Zeilen Code verfasst. Davon wurden **1996** Zeilen in Python und **1963** Zeilen in TypeScript inklusive Vue.js erstellt.

Für die Entwicklung der Erweiterung wurden **43** CI/CD-Jobs mit GitLab pipelines erstellt, die durchschnittlich **2** Minuten dauerten.

In SonarQube [90] wurde der Code analysiert und die Qualität überwacht. Die Wartbarkeit wurde dabei mit einem **A** bewertet. SonarQube ist ein Tool zur statischen Codeanalyse, das die Qualität des Codes, die Testabdeckung, Vulnerabilities, Code Smells und Bugs überwacht.

13

Softwaredokumentation

13.1. Installation

Die Installation des USM-Calculators erfolgt am einfachsten über Docker. Auf einem Server oder dem Entwicklungscomputer muss die Containersoftware Docker installiert sein. Für Desktopumgebungen wird empfohlen, Docker Desktop zu verwenden.

Dank Docker Compose kann die Anwendung (Frontend und Backend) mittels eines einzigen Befehls gestartet werden. Im Root-Verzeichnis des Quellcodes befindet sich die Datei `docker-compose.yml`, welche die Konfiguration der Container-Komposition enthält. Der Befehl für den Start der Anwendung lautet wie folgt:

```
1    docker-compose up
```

In der README-Datei des Repositories ist eine ausführlichere Anleitung zur Installation vorhanden. Zu finden sind dort auch weitere Befehle, die eine Ausführung der Anwendung ohne Docker ermöglichen, wobei jedoch Abhängigkeiten manuell installiert werden müssen.

Für die Entwicklung des ”USM Calculators” Webservice wird die Verwendung des Development Containers, das sich im Projekt im Unterverzeichnis `.devcontainer` befindet, empfohlen. Ein möglicher Einsatz wird im Abschnitt 13.3 beschrieben.

13.2. Bedienungsanleitung

Die Bedienungsanleitung aus der Studienarbeit [8] wurde aktualisiert und erweitert. Die folgenden Abschnitte beschreiben die neuen Funktionalitäten, die im Rahmen dieser Arbeit implementiert wurden. Die Parameter zur Berechnung der Zersiedelung sind gleich geblieben.

13.2.1. Statistikdaten der ''Reporting Unit'' bearbeiten

Mit einem Klick auf ein bestimmtes Gebiet nach dem Hochladen der ”Reporting Unit” wird ein ”Popup” geöffnet, in dem die Statistikdaten des Gebiets bearbeitet werden können. Die Tabelle zeigt immer Maximal 10 Zeilen an, die mit den Pfeilen am unteren Rand der Tabelle durchgeblättert werden können.

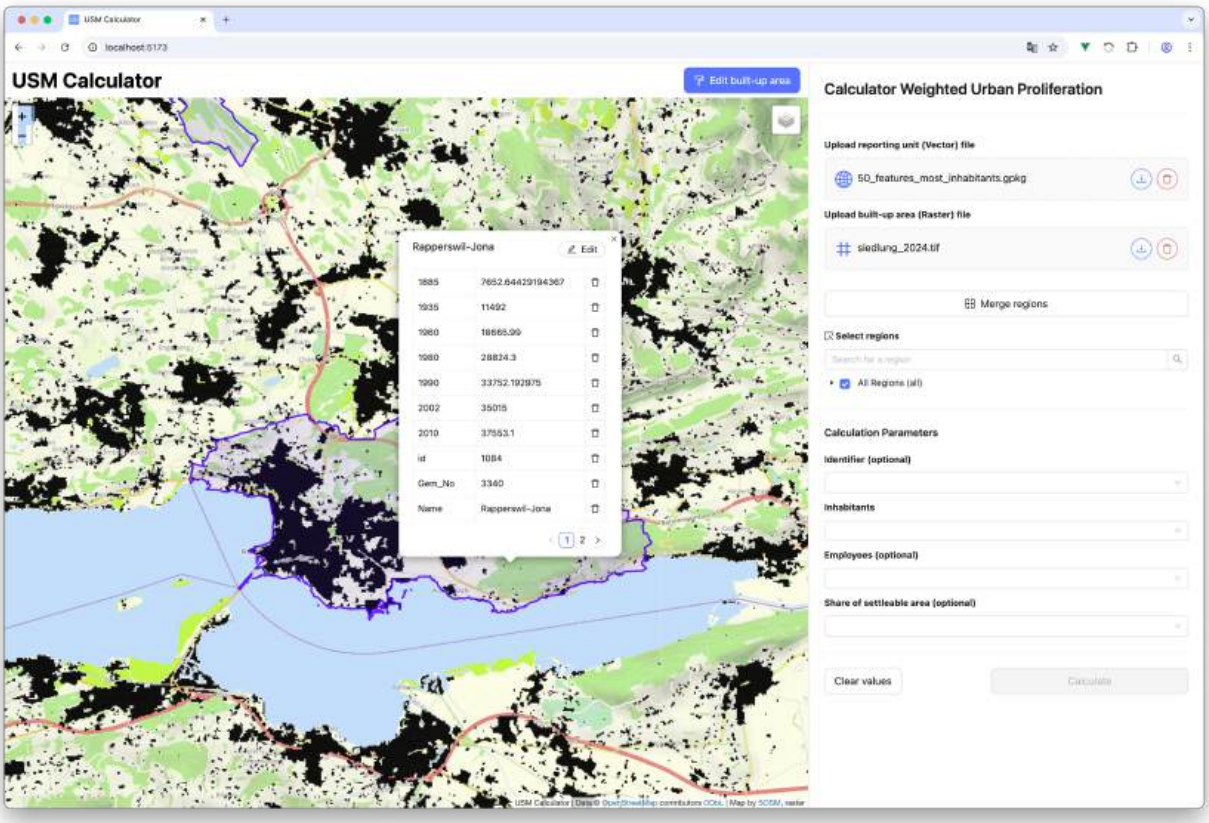


Abbildung 13.1: Anzeige des ”Popup”-Fensters mit den Statistikdaten der Gemeinde Rapperswil-Jona.

Im Bearbeitungsmodus, der durch einen Klick auf den ”Edit”-Button aktiviert wird, können die Statistikdaten der ”Reporting Unit” bearbeitet werden. Dabei können jeweils der Attributename und der Attributwert zusammen geändert werden. Mit einem Klick auf den ”Save”-Button werden die Änderungen gespeichert. Die Änderungen werden in der ”Reporting Unit” gespeichert und können bei der Berechnung der Zersiedelung berücksichtigt werden.

Eine zusätzliche Funktionalität bietet die Auswahl bestimmter Gebiete, um diese in der Berechnung einzubeziehen. Diese Selektion geschieht über die Checkboxen rechts neben der Karte und im Teil ”Select regions”. Mit der Suchleiste können Gebiete gesucht werden, die dann rot hervorgehoben werden.

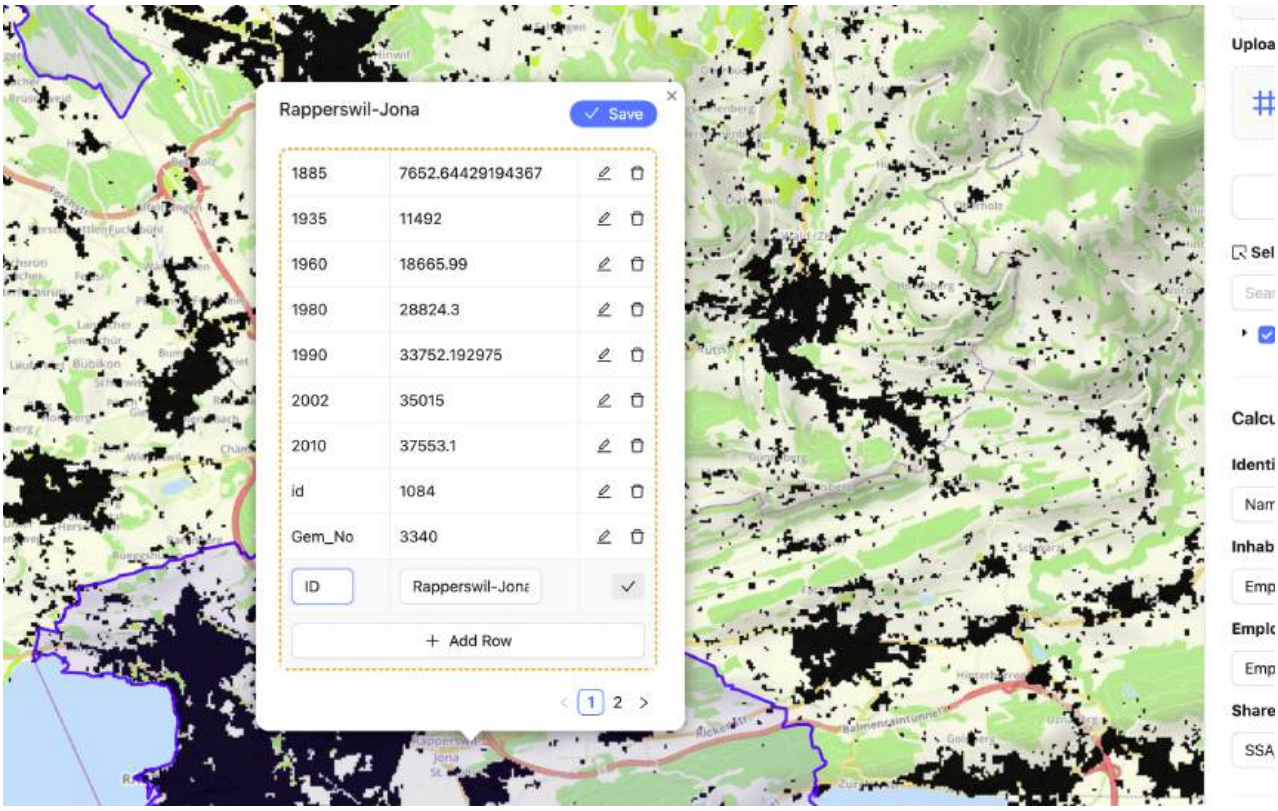


Abbildung 13.2: Im Bearbeitungsmodus können die Statistikdaten der "Reporting Unit" angepasst werden.

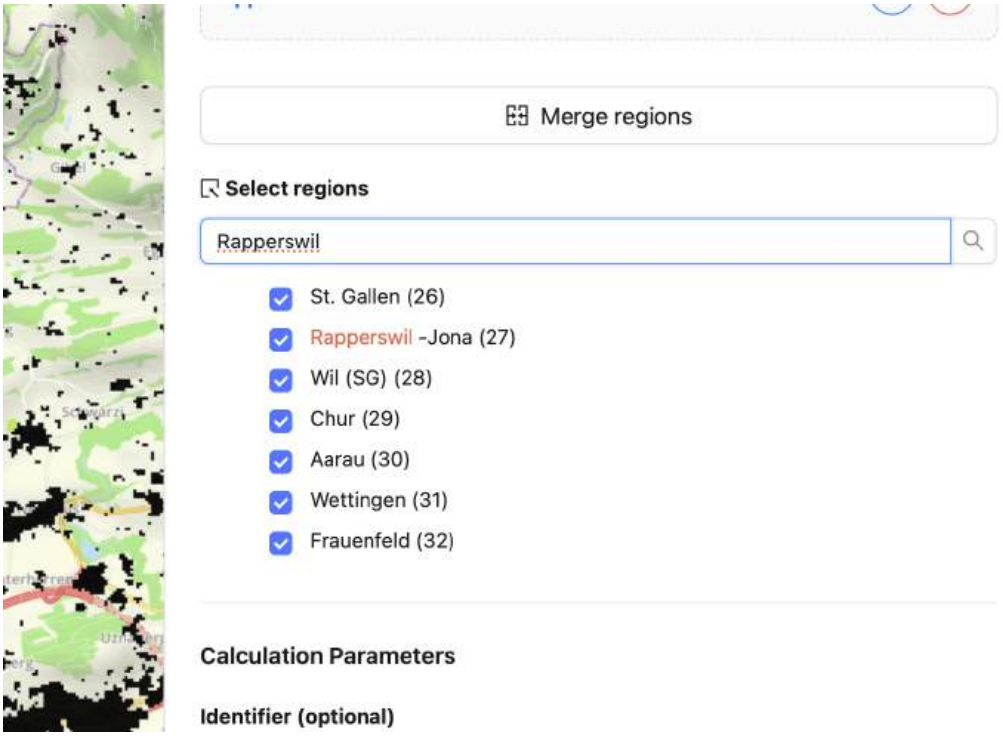


Abbildung 13.3: Gesuchte Gebiete werden rot hervorgehoben.

13.2.2. Rasterdaten bearbeiten

Um die Rasterdaten bearbeiten zu können, muss zuerst ein Siedlungsraster hochgeladen werden. Das Siedlungsraster muss im GeoTIFF-Format vorliegen und darf nur die Werte 0 (keine Siedlung) und 1 (Siedlung) enthalten. Ist das Siedlungsraster hochgeladen, kann es über den Button "Edit built-up area" im Applikationsheader bearbeitet werden. Im Bearbeitungsmodus können die Pixel mittels Mausklicks auf der Karte geändert werden. Eine orangefarbene Box zeigt beim überfahren der Karte an, wo die Pixel geändert werden.

Es gibt zwei Modi, die über die Buttons "Add" und "Remove" ausgewählt werden können. Im "Add"-Modus werden die Pixel, auf die geklickt wird, zu Siedlungsgebieten hinzugefügt und grün markiert. Im "Remove"-Modus werden die Pixel, auf die geklickt wird, von den Siedlungsgebieten entfernt und rot markiert.

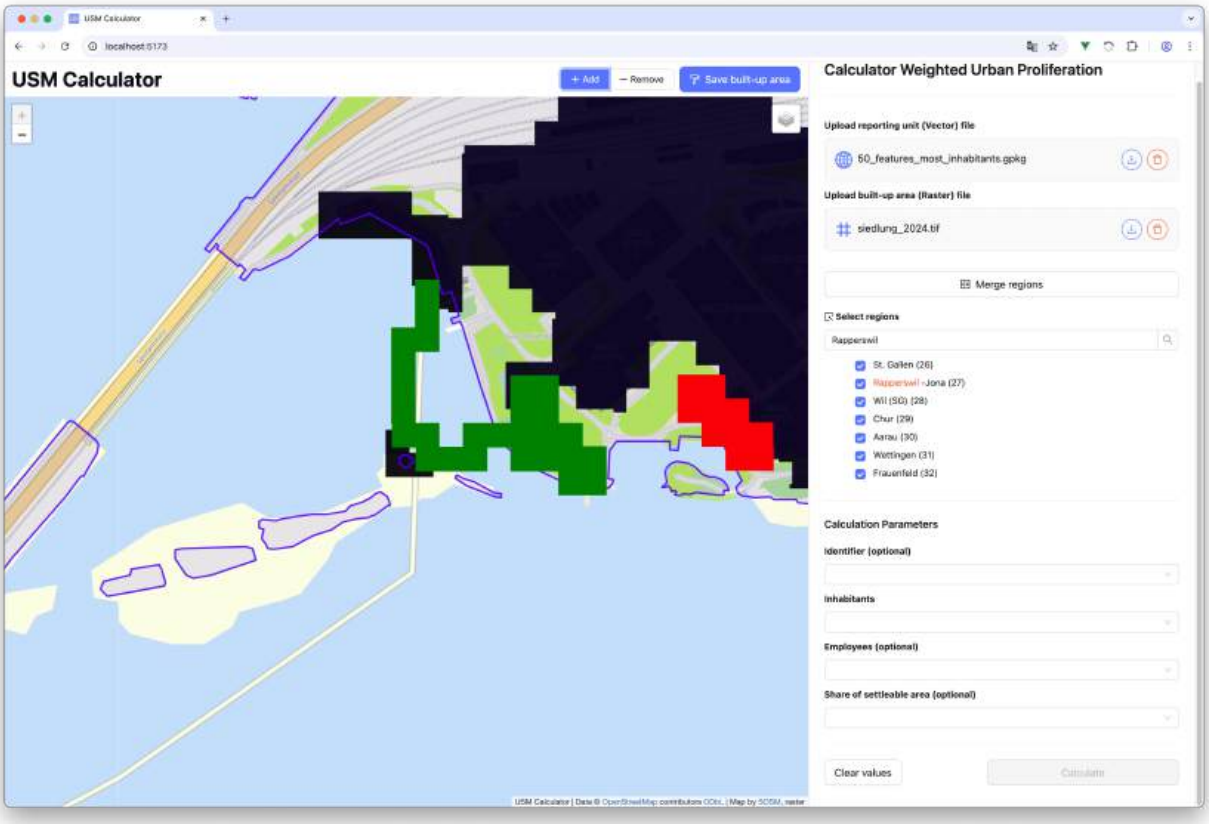


Abbildung 13.4: Das Siedlungsraster wird bearbeitet und die Pixel werden hinzugefügt oder entfernt.

Mit einem Klick auf den Button "Save built-up area" werden die Änderungen gespeichert. Die Änderungen werden nach erfolgreichem Speichern direkt in der Karte angezeigt und das neue Siedlungsraster kann für die Berechnung der Zersiedelung verwendet werden.

Abbildung 13.5: Die Veränderungen am Siedlungsraster sind auf der Karte sichtbar.

13.2.3. Verschmelzen von Vektordaten der ''Reporting Unit''

Die ”Reporting Unit” kann über den Button ”Merge regions” im Menü auf der rechten Seite der Karte bearbeitet werden. Ein neues Fenster öffnet sich, das zeilenweise alle Gebiete der ”Reporting Unit” anzeigt. In den Spalten befinden sich die Namen der Attribute und ihre jeweiligen Werte. Die Spalten ”ID” und ”Name” (nur wenn im Datensatz vorhanden) werden ganz links angeheftet.

In den zwei untersten Zeilen befinden sich in der ersten Zeile Dropdown-Menüs, in denen eine Strategie zum Zusammenführen der Gebiete ausgewählt werden kann und in der zweiten Zeile der daraus resultierende Wert, wenn die Strategie angewendet wird. Wird die Strategie ”Manual value” ausgewählt, kann in der zweiten Zeile ein Wert manuell eingegeben werden.

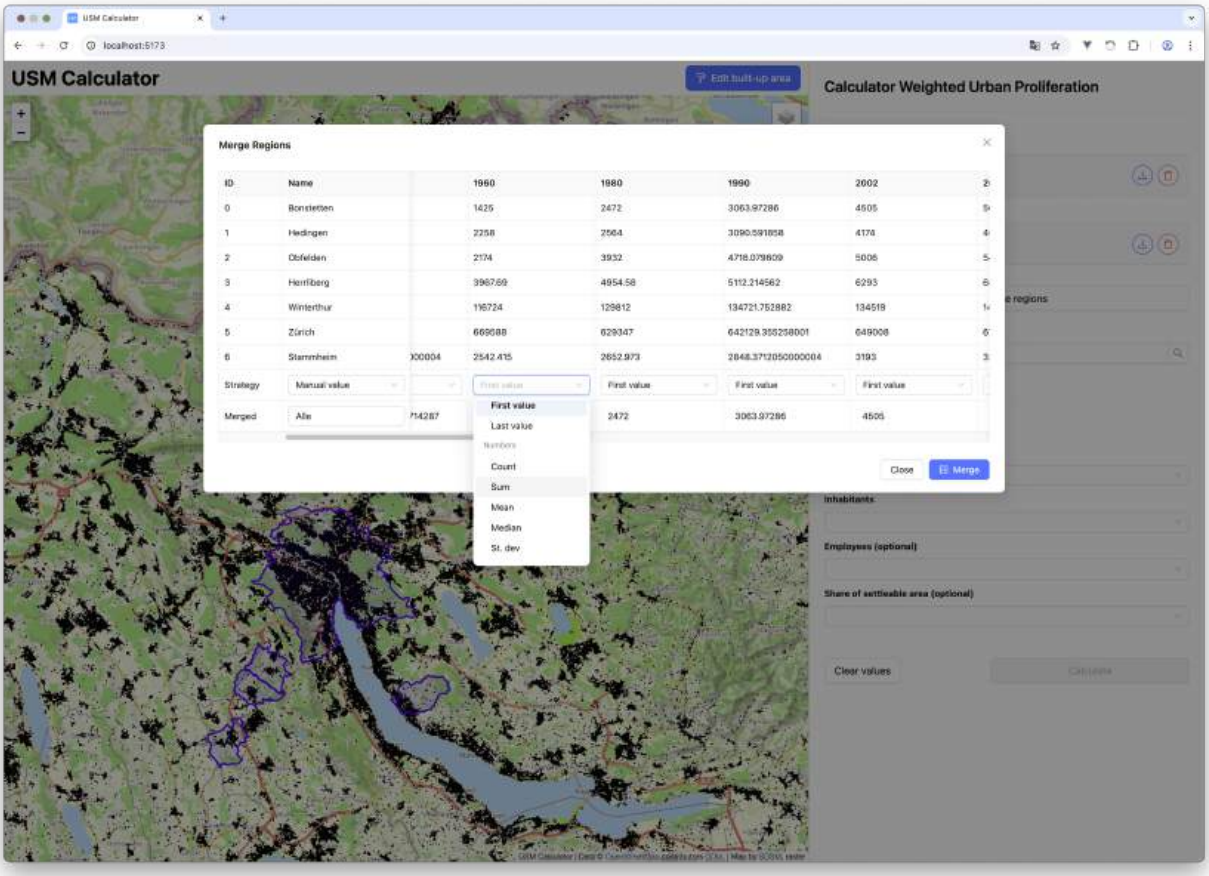


Abbildung 13.6: Anzeige des Fensters zum Zusammenführen von Vektordaten der ”Reporting Unit” mit ausgeklapptem Dropdown-Menü zur Auswahl der Zusammenführungsstrategie.

Wird am Schluss auf den Button ”Merge” geklickt, werden die Gebiete der ”Reporting Unit” zusammengeführt und direkt in der Karte angezeigt. Das Resultat kann anschliessend exportiert werden, wie in der Abbildung 13.7 dargestellt. Dasselbe gilt ebenfalls für das angepasste Siedlungsraster, das ebenfalls über einen ähnlichen Button jederzeit heruntergeladen werden kann.

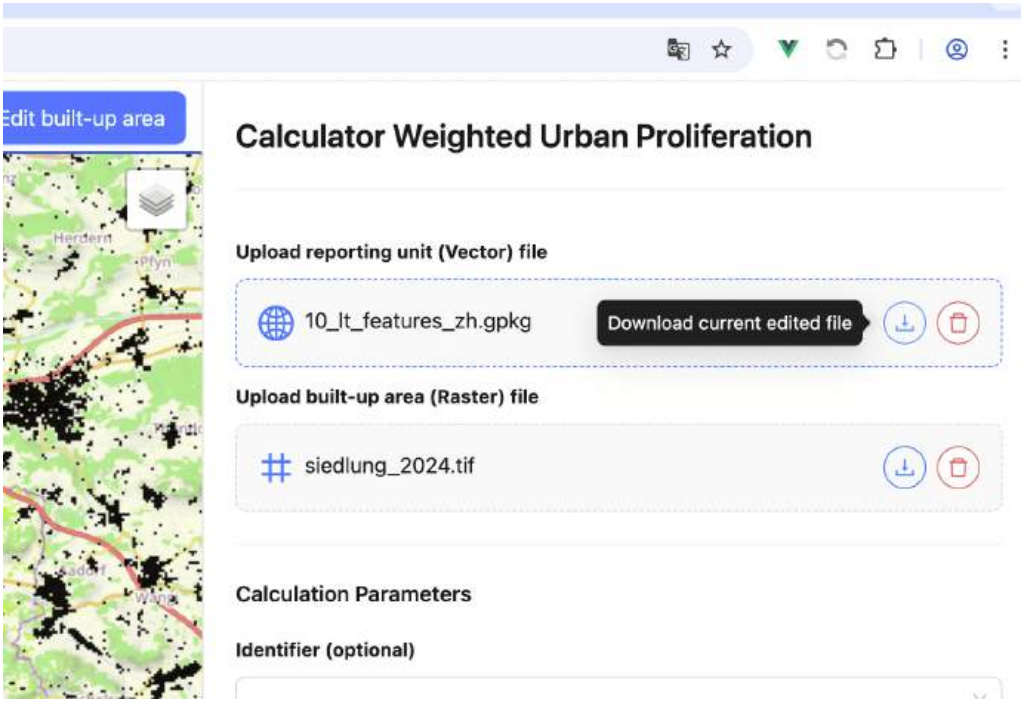


Abbildung 13.7: Die zusammengeführten Regionen können heruntergeladen werden.

13.2.4. Berechnung und Resultate

Wird die Berechnung gestartet, werden alle Eingabefelder deaktiviert, um eine nachträgliche Änderung der Eingaben zu verhindern. Der Status der Berechnung wird in einer Statusleiste, wie in Abbildung 13.8 ersichtlich, angezeigt.

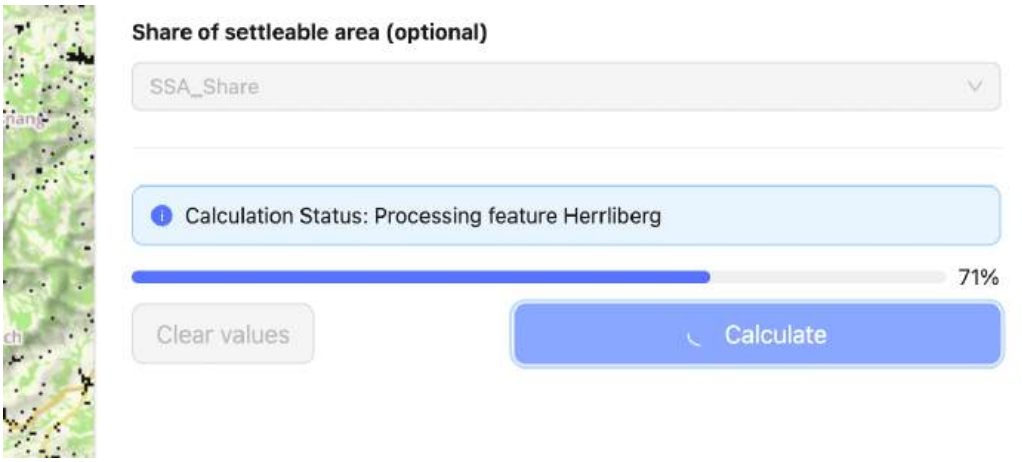


Abbildung 13.8: Statusleiste während der Berechnung.

Am Ende der Berechnung wird eine Zusammenfassung der Ergebnisse, wie in Abbildung 13.9 dargestellt, angezeigt. Die Ergebnisse werden in einer Tabelle dargestellt, wobei in einem durchnummerierten Menü die Daten der einzelnen Gebiete ausgewählt werden können. Das SI-Raster kann pro Gebiet ein- und ausgeblendet werden.

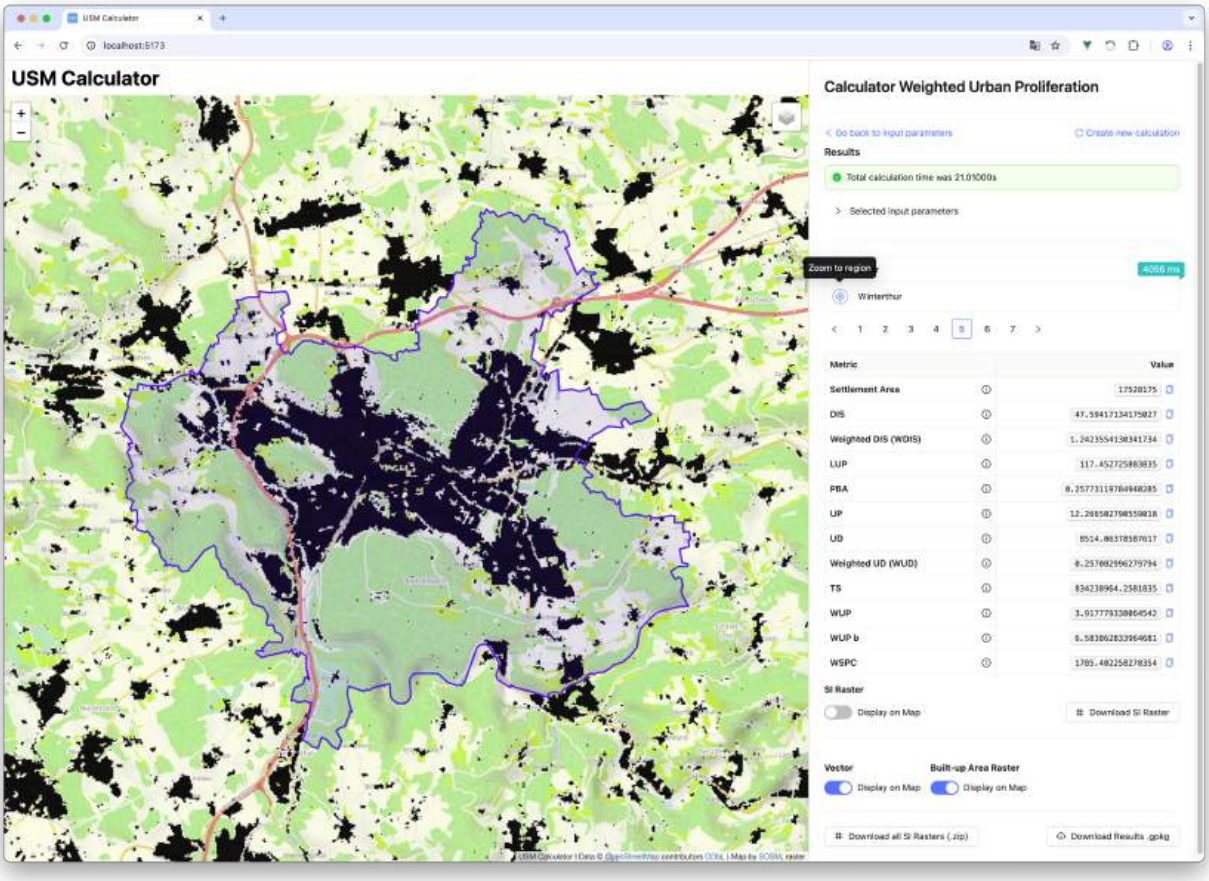


Abbildung 13.9: Anzeige der Ergebnisse nach erfolgreicher Berechnung.

Zur besseren Darstellung des SI-Rasters lassen sich das Vektor-Polygon der ”Reporting Unit” und das Siedlungsraster ebenfalls ein- und ausblenden. In der Abbildung 13.10 sind beide Layer auf der OSM-Baiskarte ausgeblendet.

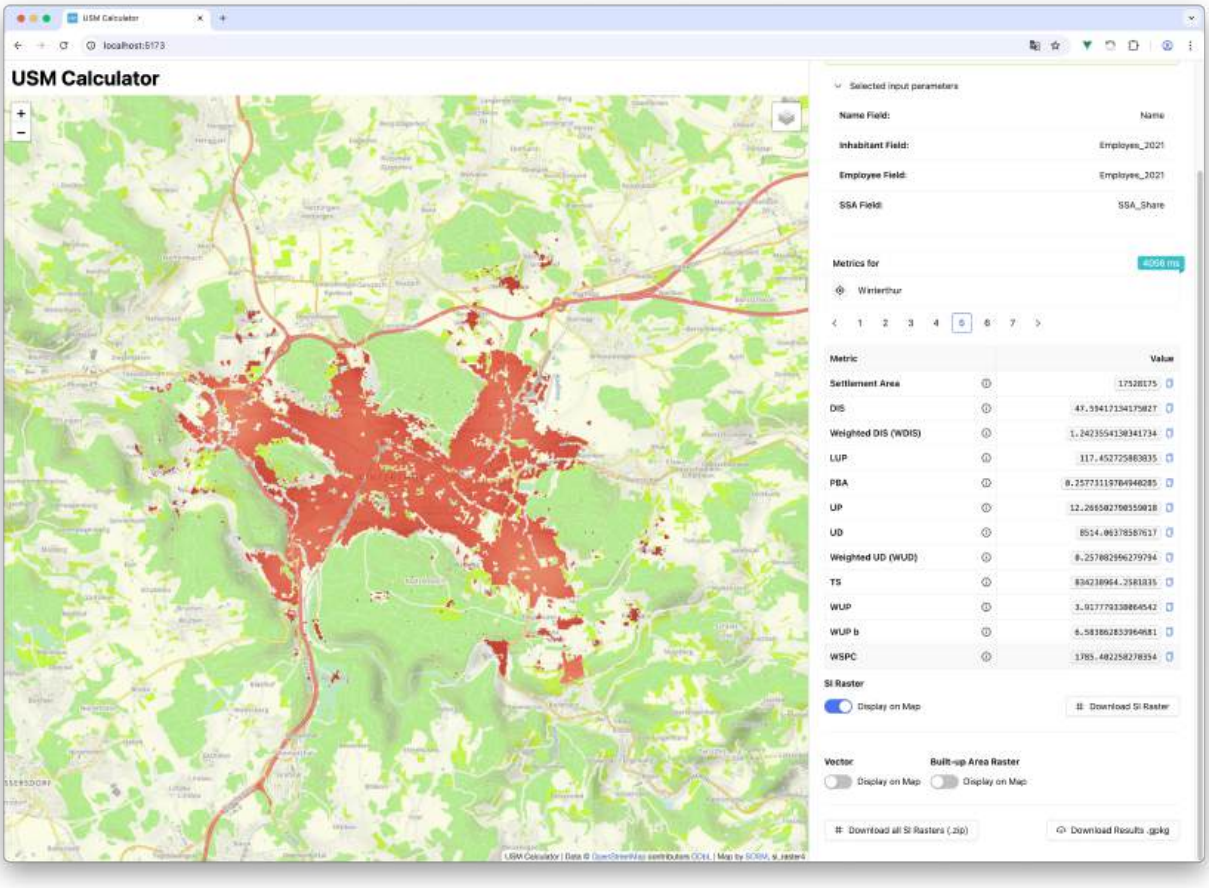
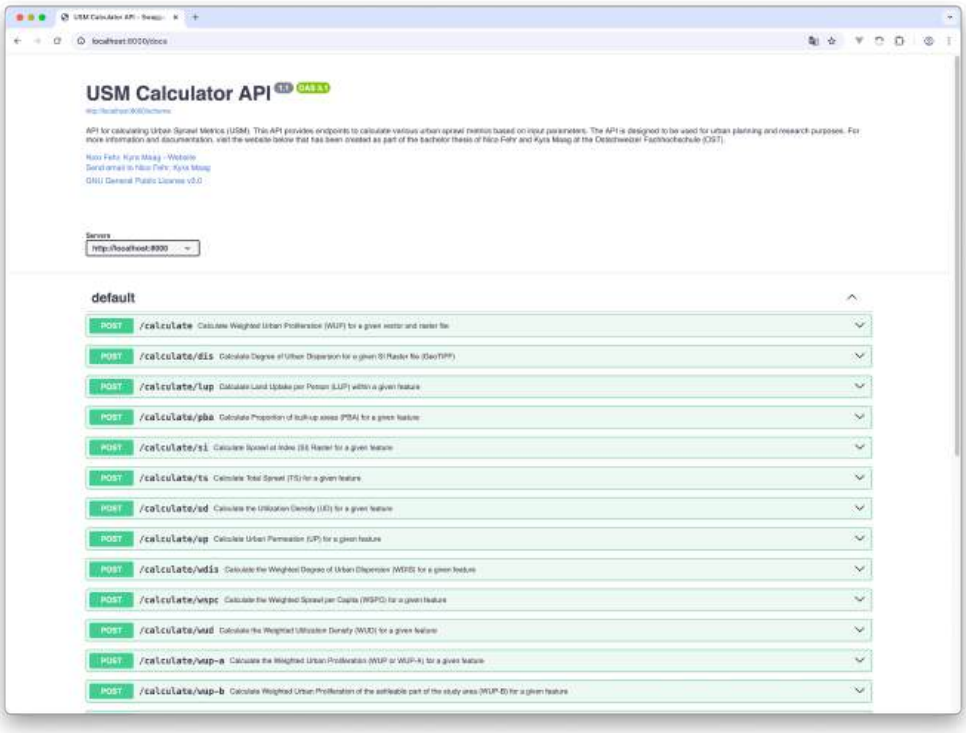


Abbildung 13.10: Ausgewähltes SI-Raster für die Gemeinde Winterthur auf der OSM-Basiskarte.

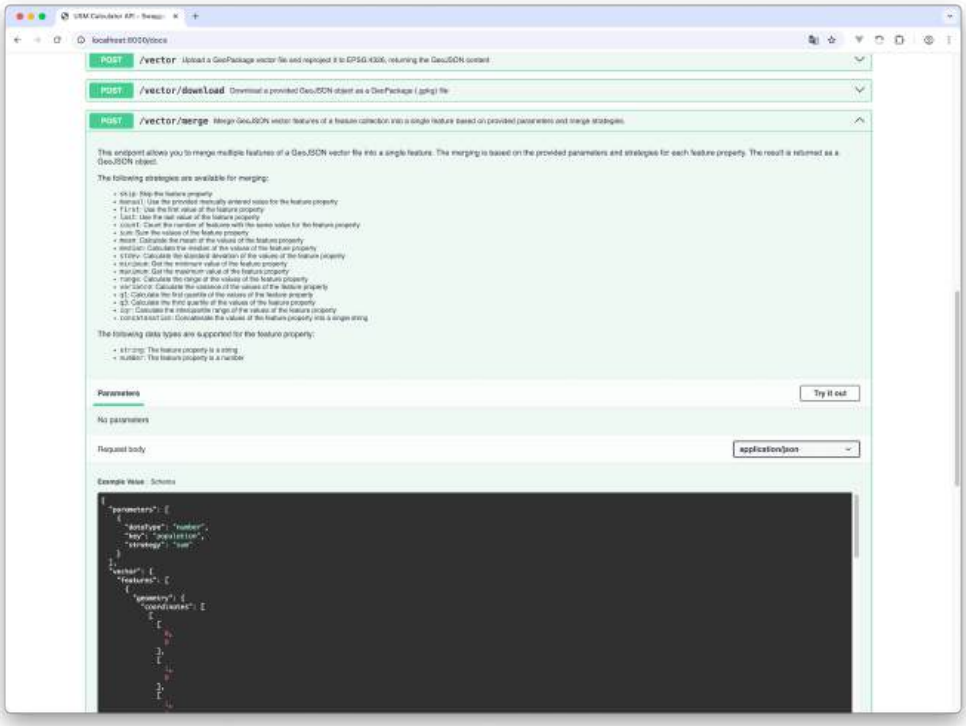
Über mehrere Buttons können einerseits das SI-Raster als GeoTIFF-Datei und andererseits die ”Reporting Unit” als GeoJSON-Datei mit den Resultaten als Attribute heruntergeladen werden. Werden mehrere Gebiete berechnet, können alle SI-Raster in einem ZIP-Archiv heruntergeladen werden.

13.2.5. API-Dokumentation

Über den Pfad `/api/docs` des Backends ist die Swagger API-Dokumentation des USM-Calculators erreichbar. Im Laufe dieser Arbeit wurde die API-Dokumentation aktualisiert und erweitert.



(a) Titel und Einleitung der Dokumentation.



(b) Der ausführlich dokumentierte Endpunkt zum Zusammenführen von Vektordaten.

Abbildung 13.11: Swagger API-Dokumentation.

13.3. Weiterentwicklung mit VSCode

Empfohlen wird die Weiterentwicklung des USM-Calculators mit Visual Studio Code (VS Code). Eine Anleitung zur Einrichtung der Entwicklungsumgebung befindet sich im Repository des Projekts. Mit dem Einsatz der VS Code Development Container Extension wird eine einheitliche und reproduzierbare Entwicklungsumgebung für alle Entwickler sichergestellt [91].

A

Persönliche Berichte

A.1. Kyra Maag

Die Weiterführung des Projekts blieb nicht nur interessant, sondern wurde sogar noch spannender, da wir nun vertieftere Einblicke in die Thematik erhalten konnten. In der Weiterführung unseres Projekts konzentrierte ich mich auf die Verbesserung der Benutzeroberfläche (UI) und die Ergänzung des Frontends. Dank meiner Erfahrungen mit UX-Tests aus dem Modul "Human Centered Design" konnte ich gute UX-Tests vorbereiten und durchführen. Das Erarbeiten der UX-Tests machte mir viel Spass. Die Verknüpfung des Frontends mit dem Backend stellte mich teilweise noch immer vor Herausforderungen, da gewisse Statistikdaten zunächst in einem nicht kompatiblen Typ gespeichert wurden. Ich musste dies erst ändern, um sie dann mit der Funktion verknüpfen zu können. Bei solchen Herausforderungen unterstützten Nico und ich uns gegenseitig mit Pair Programming, um diese schneller zu überwinden. Die Zusammenarbeit mit Nico war sehr gut. Wir konnten unsere Semesterarbeit gemeinsam weiterführen, was vieles in der Planung und Organisation erleichterte. Wir kannten uns bereits mit dem Thema aus und hatten uns eine gewisse Arbeitsstruktur und einen Arbeitsrhythmus angeeignet. Ich bin mit dem Endergebnis sehr zufrieden. Trotz anfänglicher Schwierigkeiten mit der Rasterbearbeitung konnten wir dieses Feature erfolgreich umsetzen, ebenso wie die anderen. Der Nutzer kann nun mit unserem Endergebnis die Zersiedelung schnell berechnen, das Raster sowie die Statistikdaten bearbeiten, Gebiete miteinander verschmelzen und mit den veränderten Daten die Berechnung durchführen. Am meisten hat mir an dem Projekt unser Meeting mit Yves Maurer vom ARE gefallen. Er gab uns einen Einblick in die Wichtigkeit und Aktualität unseres Projekts für die Schweiz. Als Erweiterung wäre es nun spannend, noch herauszufinden, ob sich die Verbesserungen der Berechnung auch in das ursprüngliche QGIS-Plugin einfügen lassen.

A.2. Nico Fehr

Die Weiterarbeit an unserem Projekt war für mich durchgehend interessant und lehrreich. Da wir bereits über ein gutes Vorwissen verfügten, fiel uns die Thematik leichter und wir konnten uns besser auf die Implementierung konzentrieren. Bei der Rasterbearbeitung hatten wir anfangs einige Schwierigkeiten, da sich die Bearbeitung des Rasters als komplizierter herausstellte als gedacht. Wir mussten uns intensiver mit der Darstellung der Rasterdaten mit Leaflet auseinandersetzen, da wir uns damit anfangs nicht so gut auskannten. Die Zusammenarbeit mit Kyra schätzte ich sehr und ich denke, dass wir eine gute gemeinsame Arbeitsmoral hatten. Wir nutzten häufig Pair Programming, um uns während der Implementierung gegenseitig zu unterstützen. Die Weiterführung der Semesterarbeit erleichterte uns die Planung und Organisation, da wir bereits eine gute Grundlage erarbeiten konnten. Fehler, die wir in der Semesterarbeit gemacht hatten, konnten wir nun korrigieren. In den letzten Wochen konnten wir sehr produktiv arbeiten und die Zusammenführung der Regionen schneller als gedacht umsetzen. Ich fand es sehr schön, dass unser Projekt auch bei Yves Maurer vom ARE Anklang gefunden hat. Es ist etwas Besonderes zu erfahren, dass die eigene Arbeit von Experten geschätzt wird und es mögliche Einsatzgebiete in der Praxis gibt. Dadurch fühlt man sich sehr wertgeschätzt und in seiner Arbeit bestätigt. Mit dem Endergebnis bin ich sehr zufrieden und freue mich auf zukünftige Anwendungsmöglichkeiten.

B

Abbildungs- und Tabellenverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

C

Literatur- und Quellenverzeichnis

[1] Schnee und Landschaft WSL - Eidgenössische Forschungsanstalt für Wald. *Zersiedelung*. URL: https://www.wsl.ch/de/landschaft/siedlung-und-raum/zersiedelung/ (besucht am 05. 06. 2025).

[2] admin.ch. *Bundesamt für Raumentwicklung ARE*. URL: https://www.are.admin.ch/are/de/home.html (besucht am 06. 06. 2025).

[3] C. Schwick u. a. *Zersiedelung messen und begrenzen : Massnahmen und Zielvorgaben für die Schweiz, ihre Kantone und Gemeinden*. ger;eng. Bristol-Schriftenreihe Band 57. Bern: Haupt Verlag, 2018. ISBN: 3258080860.

[4] Y. Maurer Weisbrod u. a. "Ein neuer Geodatensatz für das Monitoring der Siedlungsentwicklung". In: *Geomatik Schweiz - Géomatique Suisse - Geomatica Svizzera* 123 (1-2) (2025), S. 4–10.

[5] R. Horiguchi und J. Schwab. *Ein offenes Werkzeug zur Messung der räumlichen Zersiedelung*. ger. https://eprints.ost.ch/id/eprint/869. Bachelorarbeit. 2020.

[6] QGIS Web Site. *QGIS Overview*. URL: https://www.qgis.org/project/overview/ (besucht am 10. 06. 2025).

[7] QGIS Python Plugins Repository. *Urban Sprawl Metrics (USM) Toolset - QGIS Plugin*. URL: https://plugins.qgis.org/plugins/usm_calculator-main (besucht am 10. 06. 2024).

[8] N. Fehr und K. Maag. *Urban Sprawl Metrics - QGIS Plugin Enhanced with Webservice*. ger. Studienarbeit (other thesis). 2024. URL: https://eprints.ost.ch/id/eprint/1268.

[9] Balsamiq. *Balsamiq: Fast, focused wireframing for teams and individuals*. URL: https://balsamiq.com/ (besucht am 11. 06. 2025).

[10] Python Software Foundation. *Python Documentation*. URL: https://docs.python.org/3/ (besucht am 18. 12. 2024).

[11] Starlette. *Starlette - The little ASGI framework that shines.* URL: https://www.starlette.io/ (besucht am 10. 06. 2024).

[12] Swagger. *Swagger Documentation*. URL: https://swagger.io/docs/ (besucht am 10. 06. 2025).

[13] Dask. *Dask is a Python library for parallel and distributed computing*. URL: https://docs.dask.org/en/stable/ (besucht am 10. 06. 2025).

[14] Joblib. *Joblib: running Python functions as pipeline jobs*. URL: https://joblib.readthedocs.io/en/latest/ (besucht am 10. 06. 2025).

[15] Python Software Foundation. *multiprocessing - Process-based parallelism*. URL: https://docs.python.org/3/library/multiprocessing.html (besucht am 10. 06. 2025).

[16] Vue.js. *Vue.js Introduction*. URL: https://v3.vuejs.org/guide/introduction.html (besucht am 10. 06. 2025).

[17] Leaflet. *Leaflet - a Javascript library for interactive maps*. URL: https://leafletjs.com/index.html (besucht am 11. 06. 2025).

[18] Leaflet. *Leaflet - Image Overlay*. URL: https://leafletjs.com/reference.html#imageoverlay (besucht am 08. 06. 2025).

[19] GeeksforGeeks. *What is RUP (Rational Unified Process) and its Phases?* URL: https://www.geeksforgeeks.org/rup-and-its-phases (besucht am 06. 06. 2024).

[20] Atlassian Git Tutorial. *Git Feature Branch Workflow*. URL: https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow (besucht am 06. 06. 2025).

[21] Mathieu Poissard. *Feature branch: A quick walk through git workflow*. URL: https://blog.mergify.com/feature-branch-a-quick-walk-through-git-workflow/ (besucht am 06. 06. 2025).

[22] Parnian Pourtaherian u. a. *Urban Sprawl Metrics (USM) Toolset - User Manual for QGIS - First Edition.* Manual. Aug. 2023. URL: https://spectrum.library.concordia.ca/id/eprint/992680/.

[23] QGIS Python API. *Welcome to the QGIS Python API documentation project*. URL: https://qgis.org/pyqgis/master/ (besucht am 10. 06. 2025).

[24] Ant Design Vue. *Ant Design Vue Documentation*. URL: https://antdv.com/docs/vue/introduce (besucht am 11. 06. 2025).

[25] GDAL. *GDAL Documentation*. URL: https://gdal.org/ (besucht am 10. 06. 2025).

[26] NumPy. *NumPy Documentation*. URL: https://numpy.org/doc/stable/ (besucht am 10. 06. 2025).

[27] Numba. *Numba - A High Performance Python Compiler*. URL: https://numba.pydata.org/ (besucht am 10. 06. 2025).

[28] Numba. *Numba User Manual - Automatic paralleization with @jit*. URL: https://numba.readthedocs.io/en/stable/user/parallel.html#numba-parallel (besucht am 10. 06. 2025).

[29] QGIS Documentation. *Merge selected features*. URL: https://docs.qgis.org/3.40/en/docs/user_manual/working_with_vector/editing_geometry_attributes.html#merge-selected-features (besucht am 01. 06. 2025).

[30] O. Zimmermann. *Olaf Zimmermann (ZIO): Architectural Decisions — The Making Of*. URL: https://ozimmer.ch/practices/2020/04/27/ArchitectureDecisionMaking.html (besucht am 06. 06. 2025).

[31] Python Software Foundation. *asyncio - Asynchronous I/O*. URL: https://docs.python.org/3/library/asyncio.html (besucht am 10. 06. 2025).

[32] Cucumber. *Gherkin Reference*. URL: https://cucumber.io/docs/gherkin/reference/ (besucht am 03. 06. 2025).

[33] ISO25000.com. *ISO/IEC 25010*. URL: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010 (besucht am 03. 06. 2025).

[34] GNU. *GNU General Public License*. URL: https://www.gnu.org/licenses/gpl-3.0.html (besucht am 03. 06. 2025).

[35] Wolfram MathWorld. *Distance*. URL: https://mathworld.wolfram.com/Distance.html (besucht am 09. 06. 2025).

[36] Bundesamt für Raumentwicklung ARE - geocat.ch R. Giezendanner. *Geodatensatz "Siedlung"*. URL: https://www.geocat.ch/geonetwork/srv/api/records/4229c353-e780-42d8-9f8c-298c83920a3a?language=all (besucht am 01. 06. 2025).

[37] EPSG.io. *EPSG:2056 - CH1903+ / LV95*. URL: https://epsg.io/2056 (besucht am 06. 06. 2025).

[38] QGIS Documentation. *Rasterize (vector to raster)*. URL: https://docs.qgis.org/3.40/en/docs/user_manual/processing_algs/gdal/vectorconversion.html#rasterize-vector-to-raster (besucht am 01. 06. 2025).

[39] Robert C. Martin. *The Clean Architecture*. Available at https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html, Accessed on 30.05.2025. Aug. 2012. URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html.

[40] Vue.js. *Composition API FAQ - Why Composition API*. URL: https://vuejs.org/guide/extras/composition-api-faq.html#why-composition-api (besucht am 11. 06. 2025).

[41] Vue.js. *Single-File Components - Why SFC*. URL: https://vuejs.org/guide/scaling-up/sfc.html#why-sfc (besucht am 11. 12. 2024).

[42] GitHub Repository - Valkey. *A flexible distributed key-value database that is optimized for caching and other realtime workloads.* URL: https://github.com/valkey-io/valkey (besucht am 03. 06. 2025).

[43] valkey-py dev documentation. *valkey-py - Python Client for Valkey*. URL: https://valkey-py.readthedocs.io/en/latest/ (besucht am 03. 06. 2025).

[44] IETF. *The WebSocket Protocol - RFC 6455*. URL: https://datatracker.ietf.org/doc/rfc6455/?include_text=1 (besucht am 10. 06. 2025).

[45] Uvicorn. *Uvicorn Documentation*. URL: https://www.uvicorn.org/ (besucht am 10. 06. 2025).

[46] pytest.org. *pytest: helps you write better programs*. URL: https://docs.pytest.org/en/latest/ (besucht am 10. 06. 2025).

[47] Poetry. *Poetry Documentation*. URL: https://python-poetry.org/docs/ (besucht am 10. 06. 2025).

[48] TypeScript. *TypeScript Documentation*. URL: https://www.typescriptlang.org/docs/ (besucht am 18. 12. 2024).

[49] Vite. *Getting Started*. URL: https://vitejs.dev/guide/ (besucht am 11. 06. 2025).

[50] Vue Leaflet. *Vue Leaflet - Harness the power of Leaflet in VueJs*. URL: https://vue2-leaflet.netlify.app/ (besucht am 11. 06. 2025).

[51] Axios. *Axios Docs - Getting Started*. URL: https://axios-http.com/docs/intro (besucht am 10. 06. 2025).

[52] Turf.js. *Turf.js Documentation*. URL: https://turfjs.org/docs/intro (besucht am 10. 06. 2025).

[53] D3.js. *The JavaScript library for bespoke data visualization*. URL: https://d3js.org/ (besucht am 10. 06. 2025).

[54] GeoTIFF/georaster. *Github - georaster*. URL: https://github.com/GeoTIFF/georaster (besucht am 10. 06. 2025).

[55] Flake8. *Flake8 - Your Tool For Style Guide Enforcement*. URL: https://flake8.pycqa.org/en/latest/ (besucht am 10. 06. 2025).

[56] peps.python.org. *PEP 8 - Style Guide for Python Code*. URL: https://peps.python.org/pep-0008/ (besucht am 11. 06. 2025).

[57] Pylint. *Pylint - code analysis for Python*. URL: https://www.pylint.org/ (besucht am 10. 06. 2025).

[58] mypy 1.13.0 documentation. *mypy - Welcome to mypy documentation!* URL: https://mypy.readthedocs.io/en/stable/ (besucht am 10. 06. 2025).

[59] ESLint - Pluggable JavaScript linter. *Find and fix problems in your JavaScript code - ESLint*. URL: https://eslint.org/ (besucht am 10. 06. 2025).

[60] Valkey. *Valkey Python Client Documentation*. URL: https://valkey.io/docs/clients/python/ (besucht am 03. 06. 2025).

[61] Python Software Foundation. *cProfile - Python's built-in profiler*. URL: https://docs.python.org/3/library/profile.html#module-cProfile (besucht am 11. 06. 2025).

[62] GitHub Repository - gprof2dot. *gprof2dot - Converts profiling output to a dot graph.* URL: https://github.com/jrfonseca/gprof2dot (besucht am 01. 06. 2025).

[63] Graphviz. *Graphviz - Graph Visualization Software*. URL: https://graphviz.org/ (besucht am 11. 06. 2025).

[64] Pypi - memory-profiler. *A module for monitoring memory usage of a python program*. URL: https://pypi.org/project/memory-profiler/ (besucht am 01. 06. 2025).

[65] GitHub Repository - GeoBlaze. *Blazing Fast JavaScript Raster Processing Engine*. URL: https://github.com/GeoTIFF/geoblaze (besucht am 02. 06. 2025).

[66] geotiff.js. *Check out GeoTIFFs in your browser!* URL: https://geotiffjs.github.io/geotiff.js/ (besucht am 10. 06. 2025).

[67] Docker. *Docker Documentation*. URL: https://docs.docker.com/ (besucht am 11. 06. 2025).

[68] Traefik Labs. *Documentation*. URL: https://doc.traefik.io/ (besucht am 11. 06. 2025).

[69] Robert C. Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. 1st. USA: Prentice Hall Press, 2017. ISBN: 0134494164.

[70] Starlette. *Starlette - Background Tasks*. URL: https://www.starlette.io/background/ (besucht am 31. 05. 2025).

[71] OpenStreetMap. *OpenStreetMap*. URL: https://www.openstreetmap.org/about (besucht am 31. 05. 2025).

[72] S. Keller, OST/SOSM (contact) with awesome help from C. Brodmann, M. Peloso u. a. *OpenStreetMap Resources for Web Developers*. URL: https://md.coredump.ch/s/S17YgVfzX# (besucht am 01. 06. 2025).

[73] EPSG.io. *EPSG:4326 - WGS 84*. URL: https://epsg.io/4326 (besucht am 06. 06. 2025).

[74] GitHub Repository - georaster-layer-for-leaflet. *Display GeoTIFFs and soon other types of raster on your Leaflet Map*. URL: https://github.com/GeoTIFF/georaster-layer-for-leaflet (besucht am 10. 06. 2025).

[75] Vue.js. *Guide - Reusability - Composables*. URL: https://vuejs.org/guide/reusability/composables (besucht am 11. 06. 2025).

[76] Linbug - A blog about data science. *You too can parallelise in Python*. URL: https://linbug.github.io/python/programming/data%20science/parallelisation/2017/09/21/You-too-can-parallelise-in-python/ (besucht am 10. 06. 2025).

[77] GDAL. *Multi-threading*. URL: https://gdal.org/en/stable/user/multithreading.html (besucht am 10. 06. 2025).

[78] Apple Inc. Support. *MacBook Pro (14-inch, M3 Pro or M3 Max, Nov 2023) - Tech Specs*. URL: https://support.apple.com/en-us/117736 (besucht am 11. 06. 2025).

[79] Bundesamt für Statistik BFS. *Die 2148 Gemeinden der Schweiz am 1.1.2022*. URL: https://www.bfs.admin.ch/asset/de/20604220 (besucht am 11. 06. 2025).

[80] Joblib. *Setting up joblib's backend*. URL: https://joblib.readthedocs.io/en/latest/parallel.html#parallel-config-backend (besucht am 11. 06. 2025).

[81] Geoinformatik - Universität Rostock. *UTM-Projektion*. URL: https://www.geoinformatik.uni-rostock.de/einzel.asp?ID=1719 (besucht am 10. 06. 2025).

[82] Swisstopo. *Schweizerische Kartenprojektionen*. URL: https://www.swisstopo.admin.ch/de/schweizerische-kartenprojektionen (besucht am 11. 06. 2025).

[83] Swisstopo. *Swisstopo - Bundesamt für Landestopografie*. URL: https://www.swisstopo.admin.ch/de (besucht am 11. 06. 2025).

[84] Figma. *Figma - Das kollaborative Designtool für Benutzeroberflächen*. URL: https://www.figma.com/de-de/ (besucht am 01. 06. 2025).

[85] Atlassian. *Kanban - How the kanban methodology applies to software development*. URL: https://www.atlassian.com/agile/kanban (besucht am 10. 06. 2025).

[86] Robert C. Martin. *The Single Responsibility Principle*. URL: https://blog.cleancoder.com/uncle-bob/2014/05/08/SingleReponsibilityPrinciple.html (besucht am 10. 06. 2025).

[87] Interaction Design Foundation. *Keep It Simple, Stupid (KISS)*. URL: https://www.interaction-design.org/literature/topics/keep-it-simple-stupid (besucht am 10. 06. 2025).

[88] Python Documentation (3.13.1). *typing - Support for type hints*. URL: https://docs.python.org/3/library/typing.html (besucht am 10. 06. 2025).

[89] npm vue-tsc. *vue-tsc - Vue 3 command line Type-Checking tool base on IDE plugin Volar*. URL: https://www.npmjs.com/package/vue-tsc (besucht am 11. 06. 2025).

[90] Sonar. *Better Code - Better Software*. URL: https://www.sonarsource.com/ (besucht am 11. 06. 2025).

[91] Visual Studio Code. *Developing inside a Container using Visual Studio Code Remote Development*. URL: https://code.visualstudio.com/docs/devcontainers/containers#_quick-start-try-a-development-container (besucht am 11. 06. 2025).

D

Glossar und Abkürzungsverzeichnis

DDoS Distributed Denial of Service, eine Form von Cyberangriff, bei dem mehrere Systeme gleichzeitig versuchen, einen Dienst zu überlasten. 30

DIS Dispersion (Streuung) der Siedlungsflächen, (gemessen in UPU/m^2). 24–28, 53, 54, 97

GDAL Geospatial Data Abstraction Library, eine Bibliothek für Geodaten. 3, 6, 32, 54, 57, 61–63

GeoTIFF Eine Erweiterung des TIFF-Formats, die geodätische Informationen enthält. viii, 6, 18, 19, 33, 53, 54, 56, 63, 64, 77, 87, 92

IFS Institut für Software, Institut der OST Ostschweizer Fachhochschule. i, 5

LUP Land Uptake per Person (LUP), de: Flächeninanspruchnahme pro Person (FI). 24, 26, 27, 53, 97

PBA Proportion of Built-up Areas (PBA), de: Anteil der besiedelbaren Fläche (ASF). 24, 26, 53, 63, 97

Pixel Punkt in einem Raster, der die kleinste Einheit bildet. Meistens im SI-Raster 15x15 Meter gross. 17, 22, 25, 54, 55, 77, 80, 87, 98

QGIS Eine freie und quelloffene Geoinformationssystem-Software. ii, vii, x, 2, 5, 7, 9, 18, 23, 28, 29, 33, 54, 61, 97

SI Sprawl at Index (SI), de: Zersiedelungsmass an einem Punkt. vi, 6, 11, 12, 19, 25, 30, 33, 53, 54, 63, 81, 91, 92, 98

TIFF Tagged Image File Format, ein Dateiformat für Rastergrafiken. Diese können geodätische Informationen enthalten. 80, 81

TS Total Sprawl (TS), de: Gesamtdurchsiedelung. 28, 53

UD Utilization Density (UD), de: Ausnützungsdichte. 26, 53

UI User Interface, die Benutzeroberfläche einer Software. ii, viii, 3, 6, 21, 44–52, 56, 79, 97

UP Urban Proliferation (UP), de: Urbane Durchdringung. 24, 26, 27, 53

UPU Urban Permeation Units (UPU), de: Durchsiedlungseinheiten (DSE). 24–26

USM Calculator Auch USM Calculator Webservice, Webapplikation zur Berechnung der Zersiedelung. ii, iv, vi, 10, 11, 25, 30, 37, 45–49, 54, 61, 65, 79, 84, 97

UUID Universally Unique Identifier, ein eindeutiger Bezeichner, der in der Softwareentwicklung verwendet wird, um Objekte eindeutig zu identifizieren. 30, 43, 53

WSPC Weighted Sprawl Per Capita (WSPC), de: Gewichtete Zersiedelung pro Kopf. 24, 28, 53

WUP Weighted Urban Proliferation (WUP), de: Zersiedelung (Z), gemessen in UPU/m^2. ii, 5, 24, 27, 53

Z Zersiedelung. ii, 5, 24, 27, 36, 52–54, 57, 60, 61, 64, 97