Urban Sprawl Metrics

QGIS Plugin Enhanced with Webservice

HS 2024: Studienarbeit

Nico Fehr, Kyra Maag

Ostschweizer Fachhochschule

OST
Ostschweizer
Fachhochschule

Urban Sprawl Metrics

QGIS Plugin Enhanced with 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, Bundesamt für Raumentwicklung (ARE), Bern
Projektdauer:	September, 2024 - Dezember, 2024
Fakultät:	Departement Informatik, Ostschweizer Fachhochschule OST

Cover:	Downtown Zürich, Switzerland - Image by Reto Stöckli, NASA Earth Observatory, based on Quickbird data copyright Digital-globe
Style:	OST Report Style, with modifications by Nico Fehr

Abstract

Das Thema der Zersiedelung ist nicht nur Bestandteil der Geographie, sondern auch eine gesellschaftliche Angelegenheit. Dies zeigt sich vor allem in der Schweiz, wo die Zersiedelung in den letzten Jahren stark zugenommen hat. Die "Zersiedlungsinitiaive"[1] vom 10. Februar 2019 bezeugt die Dringlichkeit der Thematik in der Politik. Unter Zersiedelung versteht man die turbulente Ausdehnung der Siedlungsfläche auf Kosten der Landwirtschafts- und Naturflächen[2]. Das Bundesamt für Raumentwicklung (ARE)[3] hat das Ziel, zu überprüfen, wie man in der Planung Zersiedelungswerte einbetten kann, um die Zersiedelung unter anderem zu messen und folglich zu begrenzen.

Bereits Jäger und Schwick begründeten in ihrem Buch "Zersiedelung messen und begrenzen"[4] Methoden zur Messung der Zersiedelung. Für ein erstes Werkzeug[5] der Eidgenössischen Forschungsanstalt für Wald, Schnee und Landschaft (WSL)[6] ist die kommerzielle Software ArcGIS[7] notwendig, worauf Horiguchi und Schwab (2020)[8] das quelloffene Plugin "USM Toolset"[9] für das Open Source Tool QGIS[10] entwickelten.

Dieses bestehende QGIS-Plugin soll um eine quelloffene Erweiterung ergänzt werden, die eine Weboberfläche bietet. Zusätzlich soll eine Schnittstelle (API) bereitgestellt werden. Darüber hinaus ist die effiziente und wartbare Implementierung der Berechnungen ein weiteres Ziel.

Im Rahmen dieser Arbeit wurden diverse Varianten einer effizienten Parallelisierung der Berechnung implementiert und anhand ihrer Performance evaluiert. Zusammen mit einer API-Schnittstelle, die es ermöglicht, einzelne Metriken zur Zersiedelung zu berechnen, wurde die Erweiterung realisiert.

Durch die Erweiterung können Metriken der Zersiedelung ohne QGIS berechnet werden. Die Kalkulation einer kleinen Gemeinde der Schweiz ist in wenigen Sekunden abgeschlossen. Mit dem Einsatz von Just-in-Time-Compilern und Parallelisierungsmethoden verbesserte sich die Performance, sodass Resultate für mittelgrosse Gemeinden in wenigen Sekunden berechnet werden können. Eine API-Schnittstelle nach der OpenAPI Spezifikation[11] ermöglicht die Integration in andere Anwendungen.

Die Berechnung der Metriken der Zersiedelung in einem Webservice kann effizient und wartbar implementiert werden. Dies bietet die Möglichkeit, diese Berechnungsmethoden einer breiteren Öffentlichkeit und Anwenderschaft zur Verfügung zu stellen. Raumplaner, Architekten, NGOs und politische Entscheidungsträger können so die Auswirkungen von geplanten Massnahmen auf die Siedlungsstruktur analysieren und visualisieren.

[1] https://www.admin.ch/zersiedelungsinitiative
[2] https://www.wsl.ch/de/landschaft/siedlung-und-raum/zersiedelung/
[3] https://www.are.admin.ch/are/de/home.html
[4] Schwick, Christian / Jäger, Jochen et. al. - Zersiedelung messen und begrenzen - https://www.haupt.ch/buecher/naturgarten/zersiedelung-messen-und-begrenzen.html
[5] https://www.wsl.ch/de/services-produkte/urban-sprawl-metrics-tool-usm/
[6] https://www.wsl.ch/de/
[7] https://www.arcgis.com/index.html
[8] Horiguchi, Ryan und Schwab, Joël - Bachelor Thesis, Ein offenes Werkzeug zur Messung der räumlichen Zersiedelung - https://eprints.ost.ch/id/eprint/869/
[9] https://plugins.qgis.org/plugins/usm_calculator-main
[10] https://www.qgis.org/
[11] https://swagger.io/specification

Management Summary

Ausgangslage

Die Zersiedelung der Landschaft (englisch ”Urban Sprawl”) ist das Abbild einer vehementen Ausbreitung der Siedlungsfläche. Dabei wird die Landwirtschafts- und Naturfläche durch Einfamilienhäuser, Gewerbe- und Industriezonen ersetzt. Dieses Bild der dünn besiedelten locker bebauten Gebiete, zeigt sich besonders in der Ferne der Ortskerne.

Messbar ist die Zersiedelung anhand einer Messgrösse Z (=Zersiedelung, englisch WUP für ”Weighted Urban Proliferation”), die nach Jäger und Schwick (2018) in der Literatur ”Zersiedelung messen und begrenzen” definiert wurde. Als Input für die Berechnung werden folgende Parameter benötigt: die Lage der Gebäude als Fläche der bebauten Gebiete, die Grenzen ebendieser Gebiete sowie die Anzahl der Einwohner und Beschäftigten des Untersuchungsgebiets. Nebenbei kann als optionaler Input der Anteil der Siedlungsfläche des Untersuchungsgebiets miteinbezogen werden. Ein bestehendes quelloffenes Plugin, das als Bachelorarbeit von Horiguchi und Schwab (2020) an der Ostschweizer Fachhochschule entwickelt wurde, ermöglicht die Berechnung von Z im Open Source Tool QGIS und soll als Grundlage übernommen werden. Eine Weboberfläche und eine API-Schnittstelle (Application Programming Interface) sollen als Erweiterung mit dieser Arbeit entwickelt werden.

Das Bundesamt für Raumentwicklung (ARE) hat zum Ziel, eine Methode zu finden wie man in der kommunalen Planung Zersiedlungswerte einbetten kann, um die Zersiedelung zu begrenzen und vorgängig zu messen. Die erarbeitete Webapplikation soll Raumplanern, Architekten und politischen Entscheidungsträgern die Möglichkeit geben, die Auswirkungen von geplanten Massnahmen auf die Siedlungsstruktur ad hoc zu analysieren und zu visualisieren.

Ziele, Vorgehen, Technologien

Im Wesentlichen hat diese Arbeit eine Performanceverbesserung zum Ziel, um die Berechnung der Zersiedelung zu beschleunigen. Dies ist relevant, um den Benutzern der Webapplikation eine schnelle Berechnung für mittelgrosse Schweizer Gemeinden zu ermöglichen. Bei der Berechnung des ”Sprawl at Index” (SI) werden die Abstände zu anderen bebauten Gebieten an einem bestimmten Punkt der Siedlungsfläche summiert und der Durchschnitt errechnet. Diese Berechnung ist ein ausschlaggebendes Element zur Bestimmung der Zersiedelung und soll performanter parallelisiert werden, um die totale Berechnungsdauer zu verkürzen. Ausserdem soll die Wartbarkeit der parallelisierten Lösung verbessert werden, wobei der Quellcode, der heute aus C++ Code besteht, vollständig in Python umgeschrieben wird. Nach einer Evaluation von verschiedenen Technologien wurde entschieden, die Parallelisierung mithilfe der Python-Bibliothek Numba zu implementieren.

Die Abhängigkeit zu QGIS soll entfernt und das Plugin um einen Webservice (Python-Backend mit Starlette) erweitert werden. Die Webapplikation (Frontend mit Vue.js und Ant Design UI Framework) soll die Resultate der Zersiedelungsberechnung mit einer Darstellung von Kartendaten (wie z.B. der Dispersion in Abb. 1) visuell mithilfe von Leaflet und Kartenmaterial von OpenStreetMap unterstützen.

Zur Erreichung der Ziele sollen eine Analyse des bestehenden Algorithmus, eine Evaluation verschiedener Parallelisierungsansätze und die Implementierung der gewählten Lösung durchgeführt werden. Zur Verifikation der Resultate des Webservices sollen die Berechnungen mit dem bestehenden Plugin für QGIS verglichen werden.

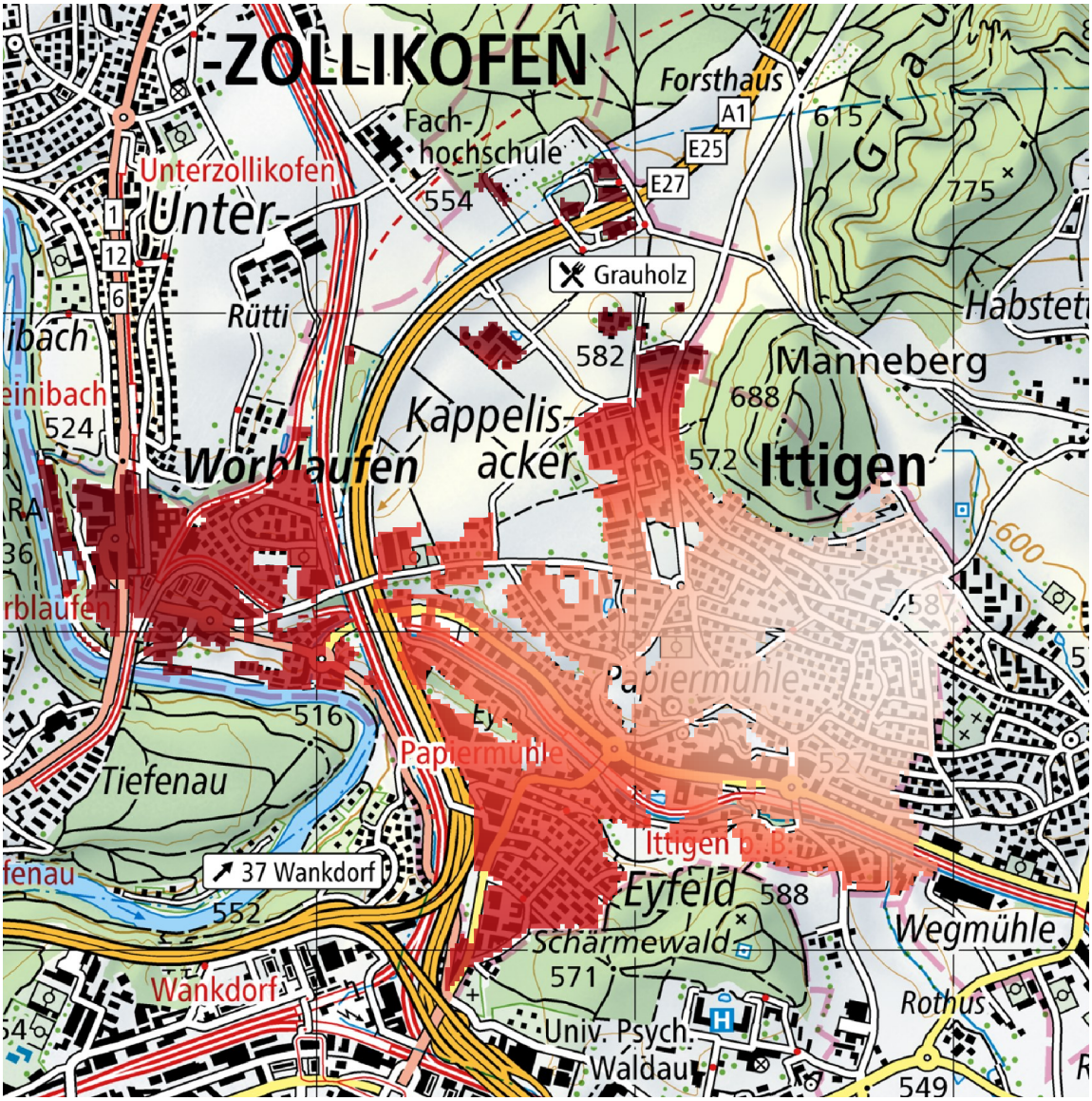

Abbildung 1: Kartographische Darstellung der Dispersion: Dunkelrote Gebiete zeigen eine grosse Zerstreuung der bebauten Flächen.

Ergebnisse

Das Resultat dieser Arbeit ist ein Webservice (Python-Backend mit Starlette) mit eigener intuitiver Benutzeroberfläche (Vue.js) als Erweiterung des bestehenden Plugins für QGIS (vgl. Abb. 2). Für die Integration in weitere Anwendungen steht eine standardisierte und dokumentierte REST-API (nach OpenAPI Spezifikation) zur Verfügung.

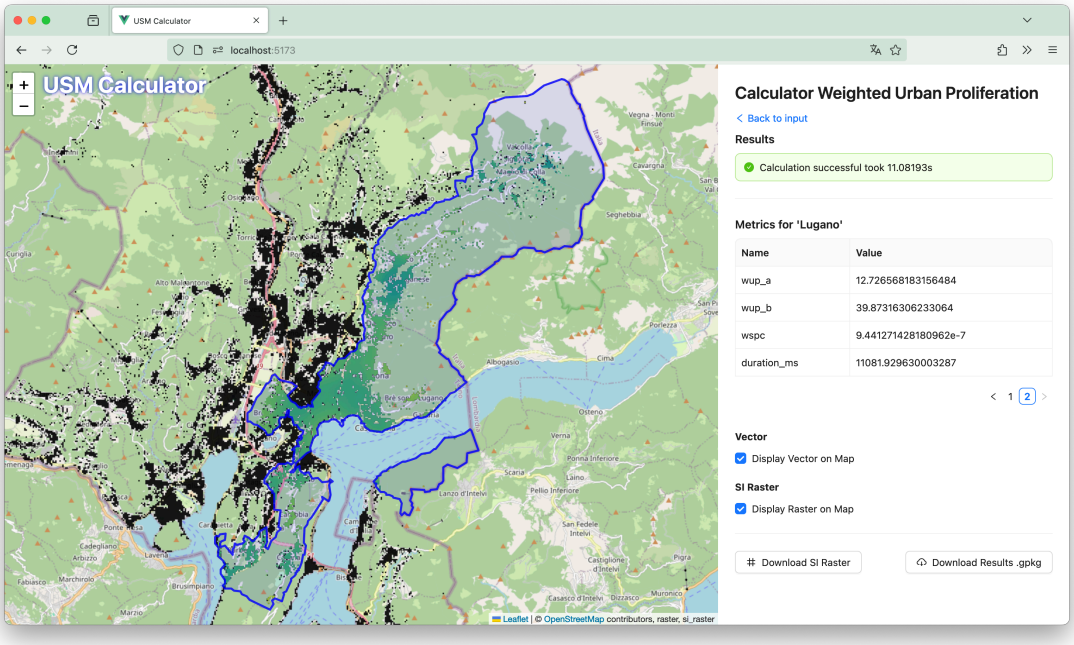

Abbildung 2: Benutzeroberfläche des Webservices: Die intuitive Oberfläche ermöglicht die einfache Berechnung der Zersiedelung.

Die Berechnung einer mittelgrossen Schweizer Gemeinde ist in wenigen Sekunden abgeschlossen (vgl. Abb. 3). Dadurch wurde die Anforderung, durch Parallelisierung die Berechnung zu beschleunigen und diese zu testen, erfüllt.

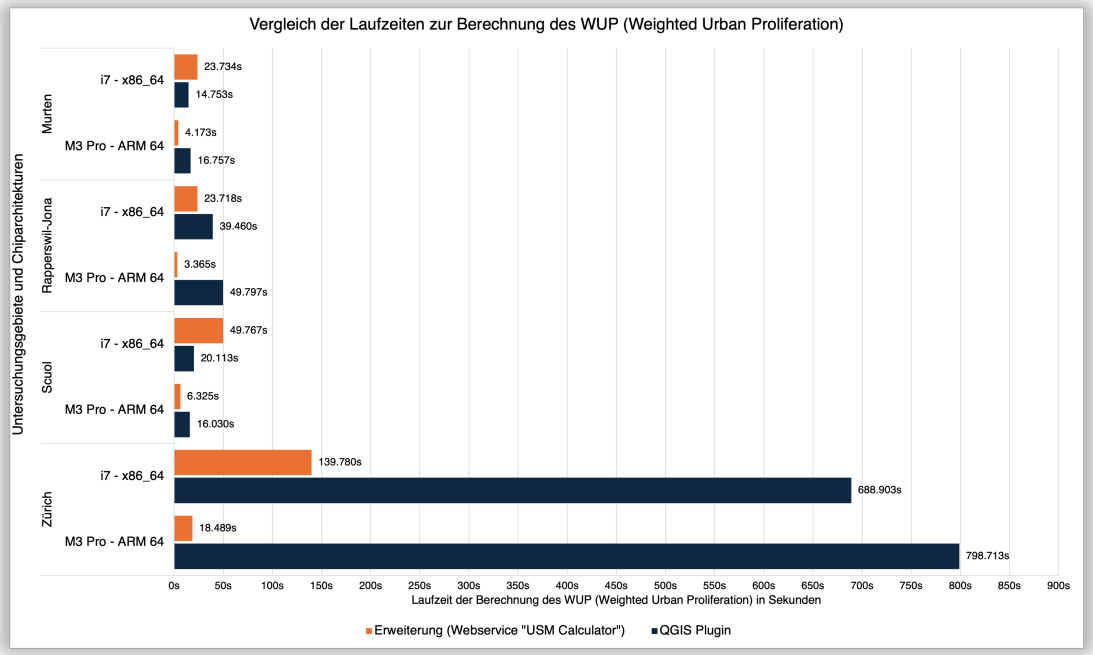

Abbildung 3: Benchmark der verschiedenen Technologien: Vergleich der Laufzeiten der Erweiterung mit dem bestehenden Plugin für QGIS.

Als Ergebnis dieser Arbeit kann Raumplanern ein benutzerfreundliches Werkzeug übergeben werden, dass die Zersiedelung in einem Untersuchungsgebiet effizient berechnet. In dieser Dokumentation wurden die Erfahrungen und Erkenntnisse, die aus der Entwicklung der Erweiterung dieses Projekts gewonnen wurden, für die Weiterentwicklung festgehalten.

Ausblick

Die Hauptziele verbesserte Parallelisierung und Webservice als API mit einem eigenen Frontend wurden in dieser Arbeit erreicht.

Zusätzlich können in weiteren Schritten die Konfigurationsmöglichkeiten der Berechnung erweitert werden. Dazu gehören Aspekte wie der Radius der Berechnung zur Bestimmung des SI (Sprawl at index), die Grenzen des Untersuchungsgebiets oder die Auswahl der zu berechnenden Metriken.

Für bestimmte Anwender, wie z.B. Raumplanern oder Architekten von Bauprojekten, kann die Möglichkeit zur Anpassung des Rasters der bebauten Gebiete sehr interessant sein. Ad-hoc-Berechnungen für spezifische Fragestellungen zu Projekten können mithilfe der Benutzeroberfläche effizienter und mit visueller Unterstützung durchgeführt werden. Dadurch können die Auswirkungen von geplanten Massnahmen auf die Siedlungsstruktur direkt analysiert werden.

Inhaltsverzeichnis

Aufgabenstellung

Hintergrund und Problemstellung

Die Herausforderung: Die Zersiedelung (”urban sprawl”) des bebauten Gebiets nimmt zu, aber keiner weiss, wie viel... Die Lösung: Eine Messmethode – eine Metrik, die sogenannten ”Urban Sprawl Metrics” (USM). Diese Metrik wurde in einem QGIS Plugin bereits implementiert. QGIS ist ein freies und quelloffenes Geographisches Informationssystem (GIS) zur Erfassung, Analyse und Darstellung räumlicher Daten. Allerdings bringt die Abhängigkeit vom QGIS-Ökosystem auch Nachteile mit sich, wie zum Beispiel eine geringere Geschwindigkeit.

Aufgaben

Als Hauptfokus der Arbeit soll ein Webservice entwickelt werden, welcher es den Planenden ermöglicht, einfach, effizient und zeitnah die Zersiedelungsmetrik zu berechnen. Deshalb ist der Kern des Webservice die Beschleunigung der Berechnung der Zersiedelung. Das Ganze ist nicht mehr an das Ökosystem von QGIS gebunden, dementsprechend können verschiedene Arten der Parallelisierung und Optimierung geprüft werden.

Für die einfache Verwendung der Planer soll noch eine einfache WebApp als Frontend implementiert werden.

Umsetzung des Projekts

Die Arbeit kann wie folgt eingeteilt werden:

- Ausarbeitung der Architektur des Webservices
- Ausarbeitung der Client - Server Kommunikation
- Evaluieren der Parallelisierung und Optimierungsmöglichkeiten
- Realisierung des Webservices und des Frontends
- Der Fokus liegt auf der Zersiedelungsberechnung d.h. diese muss nur für ein Gebiet möglich sein.

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 Zersiedelung ist die turbulente Ausdehnung der Siedlungsfläche auf Kosten der Landwirtschafts- und Naturflächen. Besonders zeigt sie sich in der Zunahme von Einfamilienhäusern, Gewerbe- und Industriezonen und Infrastrukturen ausserhalb der Siedlungszentren.

Dabei stellt die Zersiedelung nicht nur ein Problem für die Geographie und Raumplanung dar, sondern ist auch ein gesellschaftliches Problem. Das Bundesamt für Raumentwicklung (ARE) hat das Ziel Indikatoren zu prüfen. Die Formeln zur Berechnung der Zersiedelungsmetriken wurden von Jäger und Schwick (2018) in der Literatur "Zersiedelung messen und begrenzen" [1] definiert. Die Bachelor Thesis von Horiguchi und Schwab (2020) [2] hat ein Plugin für QGIS entwickelt, das die Berechnung der Metriken der Zersiedelung mithilfe der Vorlage von Jäger und Schwick ermöglicht. Das Plugin ist quelloffen und kann von der Community weiterentwickelt werden. Im Rahmen dieser Arbeit soll das Plugin um eine Weboberfläche erweitert werden, um die Berechnung der Metriken der Zersiedelung ohne QGIS zu ermöglichen. Als Grundlage der Erweiterung diente somit der quelloffene Code in Python und C++ des bestehenden Plugins.

Planungsbehörden, Raumplaner, Architekten und politische Entscheidungsträger sollen die Möglichkeit erhalten, die Auswirkungen von geplanten Massnahmen auf die Siedlungsstruktur zu analysieren und visualisieren. Sie stellen das Zielpublikum der Erweiterung dar.

1.2. Vision

Metriken zur Zersiedelung sollen ohne QGIS direkt über eine Weboberfläche berechnet werden können. Weder die Installation der Software noch ein genaues technisches Verständnis ihrer Anwendung soll notwendig sein. Berechnungen der Metriken sollen in wenigen Sekunden für mittelgrosse Schweizer Gemeinden möglich sein. Grundsätzlich wird wie beim bestehenden Plugin das besiedelte Gebiet als Fläche definiert, die Anzahl von Einwohnern und Beschäftigten beigemessen und in der Folge die Berechnung gestartet. Die Resultate demonstrieren den Planenden die Auswirkungen ihrer planerischen Massnahmen auf die Zersiedelung.

1.3. Ziele

Das Hauptziel liegt in der Entwicklung eines Webservices zur effizienten und zeitnahen Berechnung der Zersiedelungsmetriken. Dabei soll die Berechnung der Zersiedelung beschleunigt werden, wobei dies die Grundlage zur Verfügbarkeit des Webservices darstellt. Die Lösung soll in Zukunft ohne die Installation von QGIS funktionieren. Für die Optimierung der Berechnung werden verschiedene Ansätze der Parallelisierung geprüft und umgesetzt.

Es soll eine simple Webapplikation entstehen, die Raumplanern die Berechnung der Zersiedelung zugänglicher macht und erleichert.

1.4. Performance

In dieser Arbeit wird der Fokus auf eine Verbesserung der Performance der bestehenden Lösung gelegt. Die Berechnung des Sprawl at Index (SI) soll performanter parallelisiert werden, um die Berechnungsdauer zu verkürzen. Die bestehende Lösung berechnet den SI momentan mithilfe einer Dynamic Link Library (DLL) in C++ [2]. Um die Wartbarkeit der Lösung zu verbessern, soll die Berechnung des SI in Python implementiert werden.

1.5. Erweiterung der bestehenden Lösung

Die bestehende Lösung wird um eine API, für die einfache Integration der Algorithmen in Applikationen, erweitert. Weiter wird eine Weboberfläche entwickelt, auf welcher die Berechnung der Zersiedelungsmetriken ohne Installation von QGIS möglich ist. Der Einsatz einer API und Weboberfläche setzt voraus, dass die Berechnung des SI möglichst performant und in Python implementiert ist. Aus diesem Grund werden verschiedene Ansätze der Parallelisierung geprüft und eine geeignete Lösung implementiert.

1.6. Rahmenbedingungen

Zu Beginn des Projekts wurden folgende Rahmenbedingungen festgelegt:

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

Tabelle 1.1: Rahmenbedingungen des Projekts.

Des Weiteren wurden in einem späteren Schritt die nicht-funktionalen Anforderungen definiert, die im Kapitel 6 auf Seite 11 zu finden sind.

1.7. Vorgehen

Das Projekt wird nach dem Rational Unified Process (RUP) [3] durchgeführt. Eine genauere Beschreibung des Vorgehens ist im Kapitel 11 auf Seite 47 zu finden. Da für die Versionsverwaltung der Software mit Git gearbeitet wird, erfolgt der Prozess der Softwareentwicklung nach dem Git Feature Branch Workflow [4] [5]. Das Projekt wird in Zusammenarbeit mit dem Bundesamt für Raumentwicklung ARE [6] durchgeführt, das als Auftraggeber fungiert.

2

Stand der Technik

2.1. Metriken der Zersiedelung nach Schwick und Jäger

In der Raumplanung ist die Messung der Zersiedelung kein neues Anliegen. Die Berechnung der Zersiedlung erfordert einen aufwändigen Algorithmus, der die Untersuchungsfläche, die bebaute Fläche und die Bevölkerungszahl (inklusive Arbeitsplätze) berücksichtigt. Des Weiteren kann der Anteil der bebauten Fläche an der Gesamtfläche mit in die Berechnung einbezogen werden. Diesen Algorithmus haben Schwick und Jäger 2018 in der Literatur "Zersiedelung messen und begrenzen" [1] definiert. Diese Methodik wird von den bestehenden Softwarelösungen verwendet und bildet auch in diesem Projekt die Grundlage für die Berechnung der Zersiedelung.

2.2. USM Toolset für QGIS

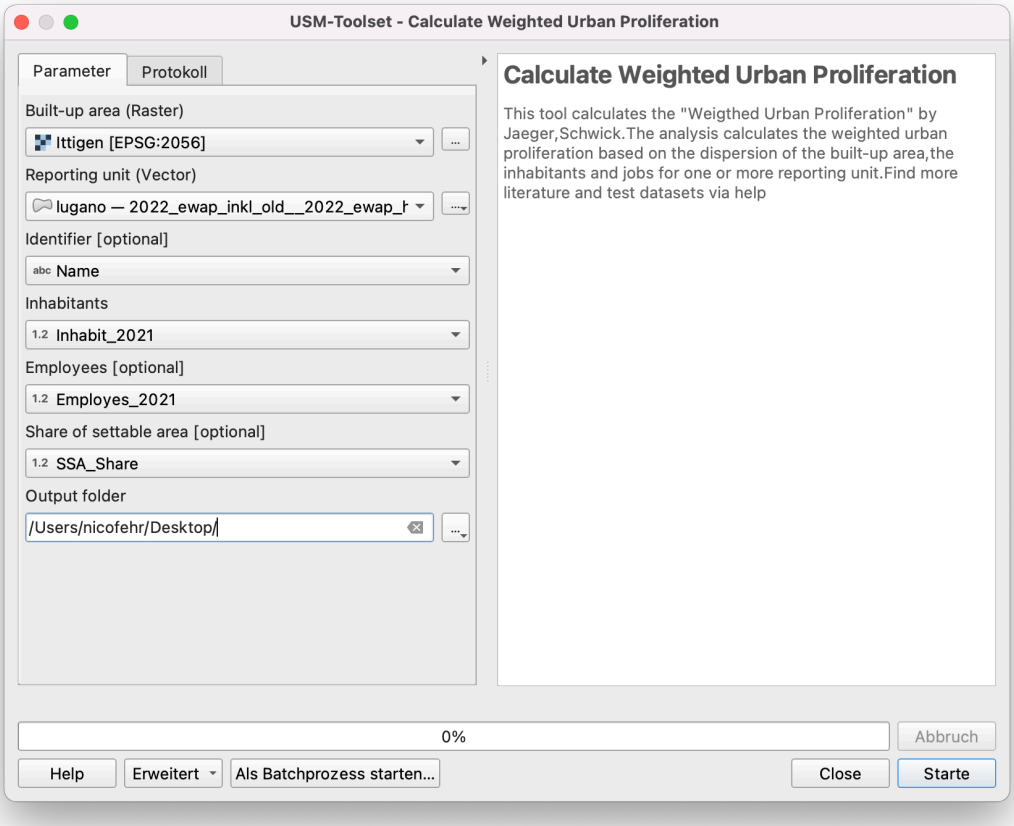

Abbildung 2.1: Screenshot des USM Toolset für QGIS. Die Input-Parameter können aus der "Reporting-Unit" geladen werden.

Für die Berechnung der Zersiedelung wird bisher das USM Toolset für QGIS [7] (vgl. Screenshot aus Abbildung 2.1) verwendet. Das Toolset wurde im Rahmen einer Bachelorarbeit von Horiguchi und Schwab 2020 entwickelt, vom IFS der Ostschweizer Fachhochschule OST [8] weiterentwickelt und ist als Plugin für QGIS unter der GNU General Public License v2.0 [9] verfügbar [10]. Die Arbeit wurde in Zusammenarbeit mit dem Bundesamt für Raumentwicklung (ARE) durchgeführt und soll die Berechnung mit Open Source

Software ermöglichen, da zuvor nur ein Tool des WSL mit der proprietären Software ArcGIS verfügbar war. Das Plugin berechnet die Metriken der Zersiedelung für eine gegebene Siedlungsfläche und stellt die Resultate als Attribute in einer Geopackage-Datei (`.gpkg`) in QGIS dar.

Diese Arbeit hat das Ziel, das USM Toolset für QGIS um eine API und ein Web-Frontend zu erweitern.

2.3. Defizite der bestehenden Lösungen

Mit dem QGIS Plugin treten nur einzelne Defizite auf. Für mittelgrosse Schweizer Gemeinden funktionieren die Berechnungen wie erwartet und Resultate werden in einer absehbaren Zeit zurückgegeben. Im Zeitraum von 10 Sekunden bis zu einer Minute sind die Resultate für die meisten Gemeinden verfügbar. Für grössere Ballungsgebiete wie zum Beispiel Zürich, dauern die Berechnungen länger und nehmen mehr Zeit (mehrere Minuten im einstelligen Bereich) in Anspruch. Um die Lösung als Webservice anzubieten, ist die Performance der Berechnung ein entscheidender Faktor und soll in dieser Arbeit verbessert werden.

Ein weiteres Defizit der bestehenden Lösung ist die Abhängigkeit von QGIS. Die Installation von QGIS ist für die Berechnung der Zersiedelung notwendig, was die Lösung für den Endanwender kompliziert macht. Vor allem für Anwender, die keine Erfahrungen mit QGIS mitbringen, ist die Installation und Bedienung der Software eine Hürde. In dieser Arbeit wird deshalb eine Lösung ohne den Einsatz des Geotools angestrebt.

Weiter soll die Berechnung des SI performanter parallelisiert werden, um die Berechnungsdauer zu verkürzen. Momentan ist dieser Teil der Berechnung in C++ implementiert und wird als Dynamic Link Library (DLL) in Python eingebunden. Um die Wartbarkeit zu verbessern, soll dieses Modul in Python implementiert werden. Dazu werden verschiedene Methoden einer solchen Umsetzung analysiert und die geeignetste Lösung implementiert.

3
Bewertung

In diesem Kapitel wird die Bewertung des Resultats beschrieben. Um Entscheidungen im Hinblick auf die Auswahl einer geeigneten Methode für die Parallelisierung der SI Berechnung zu treffen, wurden verschiedene Kriterien definiert. Die objektive Bewertung erfolgt über qualitative und quantitative Kriterien.

3.1. Kriterien

Hauptkriterium zur Entscheidungsfindung ist die Performance der Lösung. Als höchste Performance wird die kürzeste Berechnungsdauer im Schnitt von mindestens drei Berechnungsdurchläufen auf verschiedenen Geräten definiert.

Die Geräte sind in dieser Bewertung von unterschiedlicher Leistungsfähigkeit und sollen die Bandbreite der möglichen Hardware abbilden. Ausserdem soll die Chiparchitektur der Geräte unterschiedlich sein, um die Performance der Lösung auf verschiedenen Plattformen zu testen. Die Berechnung des SI ist aufgrund seiner Art nicht arbeitsspeicherintensiv, sondern rechenintensiv, belastet also primär die CPU. Weil für jedes Pixel die Abstände zu anderen im Ausschnitt des Rasters unabhängig voneinander berechnet werden, kann man die Berechnung des Sprawl at Index (SI) als ein ”Embarrassingly Parallel Problem” [11] betrachten.

Zusammengefasst sind die Kriterien für die Bewertung der gewählten Methode zur Parallelisierung der Berechnung des SI folgende:

Kriterium	**Art**	**Beschreibung**
Performance	Quantitativ	Die Performance der Lösung wird anhand der Berechnungsdauer des (SI) gemessen.
Geräte	Qualitativ	Die Berechnungsdauer wird auf verschiedenen Geräten getestet.
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 Plugins manuell verglichen.

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

3.2. Schlussfolgerung

Die Bewertung der Performance der Lösung erfolgt anhand der Berechnungsdauer des SI auf verschiedenen Geräten. Die Wartbarkeit der Lösung wird anhand der Lesbarkeit des Codes und der Dokumentation gemessen.

Die Messung der Performance erfolgt durch die Berechnung des SI auf verschiedenen Geräten und die Bestimmung der durchschnittlichen Berechnungsdauer und ist im Kapitel 10.2.3 beschrieben. Um die Korrektheit der Resultate zu bewerten, werden die Kennzahlen der Zersiedelung mit denen des bestehenden Plugins verglichen.

4

Umsetzungskonzept

4.1. Beschreibung

Gemäss der Aufgabenstellung wird das bestehende QGIS Plugin durch einen Webservice erweitert. Der Webservice wird die selben funktionalen Anforderungen des Plugins erfüllen und die Berechnung der Zersiedelung ermöglichen.

Für die Umsetzung soll dabei möglichst viel Code aus dem bestehenden Plugin wiederverwendet werden. Der Webservice wird in Python implementiert und soll asynchrone Operationen unterstützen. Verschiedene Frameworks kommen für die Implementation infrage, darunter Flask, Starlette und Django. Die erste Möglichkeit, die in Betracht gezogen wird, ist die Verwendung des Flask-Frameworks. Flask ist ein leichtgewichtiges Web-Framework für Python, das sich gut für die Entwicklung von simplen REST-APIs eignet. Flask (WSGI, Web Server Gateway Interface) besitzt aber den Nachteil, dass das Framework keine asynchrone Unterstützung bietet im Vergleich zu Starlette (ASGI, Asynchronous Server Gateway Interface). Ausserdem ist Starlette als leichtgewichtiges Framework, optimal geeignet, wenn es um Performance geht. Als dritte Möglichkeit wird Django in Betracht gezogen. Im Vergleich zu den anderen Frameworks ist Django überdimensioniert. Es ist ein Full-Stack-Framework, das viele Funktionen und Bibliotheken für die Entwicklung von Webanwendungen bietet, die für die Umsetzung dieses Projekts nicht von Nutzen sind.

Nach der Evaluierung ist die Entscheidung auf Starlette gefallen, weil es ein leichtgewichtiges Framework ist, das asynchrone Operationen unterstützt. Viele der bestehenden Funktionen können dank des simplen Designs von Starlette, ohne grossen Aufwand in eine REST-API eingebettet werden. Durch die asynchrone Unterstützung von Starlette gibt es auch keine Blockierungen bei der Berechnung der Zersiedelung.

Der Softwaredokumentation zur Implementation im Kapitel 10 können genauere Angaben der Entscheidungsfindung und Begründungen entnommen werden.

5
Resultate

5.1. Zielerreichung

Die Berechnung der Zersiedelung und die Visualisierung der Resultate ist mittels der Webapplikation "USM-Calculator" unkompliziert und effizient möglich. Durch die Integration einer verbesserten Lösung für die parallelisierte Berechnung des SI konnte die Berechnungsdauer erheblich reduziert werden. Das erarbeitete Open Source Produkt ist frei verfügbar und kann offen weiterentwickelt werden. Beliebige Anwender und Entwickler von Geodaten haben die Option, die Software zu erweitern und anzupassen.

Neben des offenen Zugangs zur Software und der verbesserten Berechnungsdauer steht auch eine REST-API zur Verfügung. Die API-Schnittstelle (Application Programming Interface) ermöglicht es, die Funktionalitäten (Berechnungsmethoden) in beliebige internetfähige Anwendungen zu integrieren. Diese Schnittstelle ist ein weiterer Schritt in Richtung einer offenen und erweiterbaren Softwarelösung für die Berechnung der Zersiedelung.

Sämtliche Codeänderungen wurden im Git-Verlauf des Projekts dokumentiert. Diese Änderungen werden zurückgesetzt und mit einem Commit abgebildet, weil es viele Prototypen gab, die fehlgeschlagen sind und nicht öffentlich sein sollen.

5.2. Ausblick: Weiterentwicklung

Die Codestruktur und Architektur der Erweiterung erlauben es, die Weiterentwicklung möglichst effizient voranzutreiben. Für die Weiterentwicklung können weitere Aspekte in Betracht gezogen werden:

5.2.1. Parallelisierung der Features

Die Parallelisierung, wie im Unterabschnitt 10.2.3 beschrieben, hat bereits die Berechnungsdauer für eine Untersuchungsfläche reduziert. Will ein Benutzer aber die Zersiedelung für mehrere Untersuchungsflächen berechnen, was die Webapplikation bereits sequenziell unterstützt, so kann die Berechnungsdauer sehr hoch werden. Eine weitere Parallelisierung der Untersuchungsgebiete ("Features") könnte die Berechnungsdauer weiter reduzieren und würde Benutzern mit grossen Datensätzen zugutekommen.

5.2.2. Ad-hoc-Raster- und Statistikdatenanpassungen in der Webapplikation

Die Webapplikation könnte um eine Funktion erweitert werden, die es ermöglicht, die Rastergrösse und die Rasterauflösung ad hoc anzupassen. Momentan sind weitere Tools wie QGIS oder ArcGIS notwendig, um die Rastergrösse und die Rasterauflösung zu verändern und die Berechnung erneut durchzuführen. Durch die Integration dieser Funktion in die Webapplikation könnte die Abhängigkeit zu weiteren Tools reduziert und mühsame Arbeitsschritte vereinfacht werden.

Weiter könnte die Webapplikation um eine Funktion erweitert werden, die es ermöglicht, die Statistikdaten direkt anzupassen. Momentan können Benutzer diese Daten nur über die Parameter der Vektor-Datei (GeoJSON) auswählen, somit muss man die GeoJSON-Datei anpassen und die Berechnung erneut durchführen.

5.2.3. Speicherung der Resultate in einer Datenbank

Das Speichern von Resultaten ist in der aktuellen Lösung nicht möglich. Zukünftig könnten die Resultate und die dazugehörigen Eingabeparameter in einer Datenbank gespeichert werden. Bereits durchgeführte Berechnungen könnten so wiederverwendet und weiterverarbeitet werden. Dazu bräuchte es eine Anbindung an eine Datenbank und eine entsprechende Schnittstelle in der Webapplikation. Des Weiteren müsste man sich Gedanken über die Sicherheit der Daten machen und entsprechende Massnahmen

treffen, wie z.B. Verschlüsselung, den Einsatz von Zugriffsrechten und gesicherten Verbindungen.

Teil II

SW-Projektdokumentation

6

Anforderungsspezifikation

6.1. Funktionale Anforderungen (Use Cases)

Die funktionalen Anforderungen des Projekts werden in Form von Szenarien beschrieben. Akteure der Szenarien sind zwei verschiedene Personas, die in Tabelle 6.1 vorgestellt werden.

6.1.1. Personas

	Architekt - Max Weber	**Raumplanerin - Lisa Meier**
Technisches Wissen	Mittelmässig. Er ist vertraut mit CAD-Software und Geoinformationssystemen (GIS).	Hoch. Lisa ist als Raumplanerin routiniert im Umgang mit Datenanalysen und GIS-Software.
Ziel	Max möchte die Zersiedelungsmetriken für ein geplantes Bauprojekt berechnen. Das Bauprojekt ist ein neues Wohnquartier in einem ländlichen Zürcher Gebiet ausserhalb der Agglomeration. Um eine nachhaltige Bauweise sicherzustellen und rechtlichen Anforderungen (z.B. Zonenpläne, Dichtevorgaben) zu genügen, benötigt er detaillierte Daten zur Zersiedelung in dem betroffenen Gebiet. Max sucht nach einem Tool, das ihm zeigt, welche Gebiete bereits stark zersiedelt sind und wo noch Potenzial für Verdichtung besteht.	Um nationale Raumplanungsstrategien zu bewerten und anzupassen, benötigt Lisa einen Überblick über die Zersiedelung in der Schweiz. Dabei steht der Fokus auf den statistischen Daten und langfristigen Trends. Das Tool möchte sie nutzen, um Berichte zu erstellen, die sie politischen Entscheidungsträgern präsentieren kann.
Erwartungen	Max erwartet eine benutzerfreundliche Oberfläche mit Kartenansichten und klaren Visualisierungen. Er will mit gängigen Datenformaten, wie zum Beispiel GeoJSON und Geopackages (GPKG) zur Zersiedelungsdichte, potenziellen Bauflächen und bestehenden Infrastrukturen arbeiten. Nebenbei braucht Max Exportmöglichkeiten für Berichte, die er seinen Kunden und der Baubehörde vorlegen kann.	Lisa erwartet, dass die Resultate der Berechnung präzise sind und umfassende Zersiedelungsindikatoren enthalten, die sie in ihren Berichten verwenden kann. Die Raumplanerin erwartet eine Downloadfunktion für Rohdaten und Diagramme in einem gängigen Format, wie zum Beispiel Geopackage (GPKG), um die Daten in GIS-Software weiterverarbeiten zu können.

Tabelle 6.1: Personas für die funktionalen Anforderungen.

6.1.2. Szenarien

Szenario 1 - Max Weber, Architekt
Max Weber hat den Auftrag, ein neues Wohnquartier zu planen. Er hat kein Tool auf seinem PC für die Berechnung der Zersiedelungsmetriken, deswegen sucht er im Internet nach einer Lösung und stösst auf das Webtool. Die benötigten GIS-Daten des geplanten Baugebiets hat Max von der Gemeinde erhalten. Diese enthalten Informationen über bestehende Gebäude, Strassen und Grünflächen. Ihn interessiert die momentane Zersiedelung des Gebiets und wie die Auswirkungen der verschiedenen Möglichkeiten sind, die er hat. Er berechnet die Zersiedelung mit verschiedenen Zahlen, um sie zu vergleichen. Max kann auf einer Karte visualisieren, wie verschiedene Bebauungsszenarien das Gebiet verändern würden. Max exportiert einen Bericht mit den berechneten Zersiedelungsdaten. Dieser Bericht hilft ihm, das Projekt der Gemeinde und potenziellen Investoren vorzustellen, und dient als Grundlage für die Genehmigungsprozesse.

Szenario 2 - Lisa Meier, Raumplanerin
Lisa Meier arbeitet an einem nationalen Bericht über die Entwicklung der Zersiedelung. Sie möchte insbesondere für die Zentralschweiz analysieren, wie sich die Zersiedelung in den letzten zehn Jahren entwickelt hat. Lisa hat Zugang zu nationalen Datenbanken mit Zersiedelungsstatistiken. Sie lädt zusätzlich spezifische Daten zu Siedlungsflächen und Einwohnerzahlen der letzten Jahre aus einer kantonalen Quelle herunter. Sie importiert die Daten in das Webtool und berechnet die Zersiedelungsmetriken für die Zentralschweiz. Aus ihren Daten kann sie die verschiedenen Jahre auswählen und berechnen lassen. Auf der Karte wird visualisiert, wie sich die Zersiedelung in den letzten Jahren entwickelt hat. Lisa lädt die Raster und Berechnungen herunter, um einen Bericht mit Diagrammen, Karten und Tabellen zu erstellen, welche die Ergebnisse klar darstellen. Der Bericht dient als Grundlage für politische Diskussionen und gibt Empfehlungen zur Anpassung der Raumplanungsstrategien auf nationaler Ebene.

6.2. Nicht-Funktionale Anforderungen

In den folgenden Tabellen sind die für unser Projekt relevanten, nicht-funktionalen Anforderungen (NFR - non-functional requirements) aufgelistet. Die Anforderungen sind nach dem ISO-Standard ISO/IEC 25010:2024 [12] definiert.

6.2.1. NFR-01: Open Source

ID	**NFR-01**
Thema	Der Quellcode der Applikation (Front- und Backend) soll offen zur Verfügung stehen. Die Open Source Community soll die Applikation weiterentwickeln und abwandeln können.
Anforderung	Wartbarkeit - Open Source, Erweiterbarkeit
Verifikation	• 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.
Priorität	Sehr hoch
Resultat	Das Projekt wurde unter der GNU General Public License v3.0 [9] 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: Exaktheit der Berechnung

ID	**NFR-02**
Thema	Die berechneten Werte der Zersiedelung sind korrekt und stimmen mit den Resultaten der Referenzsoftware überein.
Anforderung	Funktional - Exaktheit
Verifikation	In einem ersten Schritt werden die berechneten Werte mit den Resultaten des bestehenden USM Toolset QGIS Plugins getestet. Es werden dabei die Werte einer Gemeinde berechnet. In einem zweiten Schritt werden die Werte der Zersiedelung mittels einem oder mehreren automatisierten Tests durchgeführt. Dabei wird ein Mockup Test-Raster erstellt, das die bebauten Gebiete kennzeichnet und eine Gemeindegrenze darüberlegt, um die zu berechnende bebaute Fläche für weitere Berechnungen zu ermitteln. Ein automatisierter Test ermöglicht die Reproduzierbarkeit der Werte in anderen Umgebungen und erlaubt es anderen Entwicklern, die Ergebnisse zu verifizieren.
Priorität	Sehr hoch
Resultat	Der Testablauf im Abschnitt "Sicherstellung der Datenqualität" 10.2.2 auf Seite 37 wurde durchgeführt und die Resultate stimmen mit den Referenzwerten überein. Des Weiteren wurden automatisierte Tests (vgl. Abschnitt 10.3) implementiert, welche die Berechnung der Zersiedelung von verschiedenen Gemeinden replizieren und die Ergebnisse verifizieren.

6.2.3. NFR-03: Datenschutz und Datensicherheit

ID	NFR-03
Thema	Die Datenübertragung findet verschlüsselt statt und die Daten werden nach Verarbeitung auf dem Server wieder gelöscht. Nach Beenden der Session werden die Daten wieder entfernt.
Anforderung	Funktional - Sicherheit
Verifikation	Die Datenübertragung, dabei ist vor allem die Interaktion zwischen dem Benutzer und des Webservices gemeint, soll über ein sicheres Protokoll wie HTTPs stattfinden. Des Weiteren müssen alle temporär angelegten Dateien nach Beenden der Session durch den Benutzer oder durch einen Fehler gelöscht werden
Priorität	Hoch
Resultat	Die Daten auf dem Server werden verschlüsselt über HTTPs übertragen und nach Beenden der Session wieder gelöscht. Kapitel 8 auf Seite 21 beschreibt die Datenschutzmassnahmen, die in der Applikation implementiert wurden.

6.2.4. NFR-04: Modularität und Wartbarkeit

ID	NFR-04
Thema	Separieren des Front- und Backends um die Wartbarkeit zu maximieren und um die Skalierbarkeit und Erweiterbarkeit sicherzustellen.
Anforderung	Wartbarkeit - Modularität
Verifikation	• Frontend und Backend sind getrennt. • Im Frontend wird ein Komponenten-basiertes Framework angewendet, um die Wiederverwendbarkeit und Wartbarkeit des Codes und der Benutzeroberfläche zu erhöhen. • Alle Komponenten und Module folgen dem Single Responsibility Principle (SRP) [13]. Dadurch grenzt sich jedes Modul beziehungsweise Komponente durch eine isolierte Funktionalität ab. • Die Modularität wird durch regelmässige Code Reviews mit dem Ziel des Decouplings und erhöhter Kohäsion überprüft.
Priorität	Mittel
Resultat	Dieses Kriterium wurde zu Ende der Elaboration Phase erfüllt, wobei die Architektur des Projekts die Trennung von Front- und Backend vorsieht. Im Kapitel 9.1 auf Seite 22 wird die Umsetzung der Architektur beschrieben.

6.2.5. NFR-05: Kompatibilität

ID	NFR-05
Thema	Kompatibilität mit den gängigsten Webbrowsern und Betriebssystemen der Gegenwart.
Anforderung	Kompatibilität - Interoperabilität
Verifikation	• Die Applikation soll auf den neuesten Versionen der Browser Chrome, Firefox, Safari und Edge korrekt gerendert und angezeigt werden können. • Es sollen keine Fehlermeldungen in der Konsole der Browser erscheinen. • Elemente der Benutzeroberfläche und Funktionen haben eine konsistente Darstellung und Funktionalität auf den verschiedenen Browsern.
Priorität	Tief
Resultat	Die Applikation wurde auf den definierten, gängigen Browsern getestet und erfüllt die Kriterien. Die Anwendung rendert korrekt und es treten keine funktionalen Fehler auf. Weil nur selbst-zertifizierte Zertifikate verwendet werden, kann es zu Warnungen bei der Verbindung kommen. Dies ist jedoch kein Fehler der Applikation, sondern eine Sicherheitsfunktion des Browsers.

6.2.6. NFR-06: Usability

ID	NFR-06
Thema	Die ”Usability” der Webapp muss sichergestellt werden. Eine intuitive Benutzeroberfläche und eine klare Strukturierung der Funktionen sind essenziell. Neben der Webapp soll auch die API-Schnittstelle des Tools nach OpenAPI Spezifikationen dokumentiert werden, um die Benutzung und Integration in andere Systeme zu erleichtern.
Anforderung	Usability
Verifikation	• Die Benutzeroberfläche ist klar strukturiert und intuitiv bedienbar. • Die Benutzerführung ist selbsterklärend und führt den Benutzer durch die Applikation. • Die Benutzeroberfläche ist responsiv und passt sich an verschiedene Bildschirmgrössen an. • Die API-Schnittstelle ist nach OpenAPI Spezifikationen dokumentiert und enthält Beispiele für die Verwendung der Endpunkte.
Priorität	Tief
Resultat	Die Usability der Webapplikation wurde durch den Einsatz einer klaren Struktur und einer intuitiven Benutzeroberfläche sichergestellt. Die API-Schnittstelle wurde nach OpenAPI Spezifikationen dokumentiert und enthält Beispiele für die Verwendung der Endpunkte. Eine Dokumentation der API-Schnittstelle ist im Git-Repository des Projekts verfügbar.

7 Analyse

7.1. Domain Model

In diesem Abschnitt wird kein eigentliches Domain Model erstellt. Vielmehr werden hier die einzelnen Berechnungsschritte des Algorithmus für die Berechnung der Zersiedelung beschrieben. Die Berechnungsschritte können der Dokumentation des USM Toolsets für QGIS [7] entnommen werden, das die Metriken der Zersiedelung nach Schwick und Jäger [1] berechnet. Die einzelnen Schritte lassen sich in mehrere Algorithmen unterteilen.

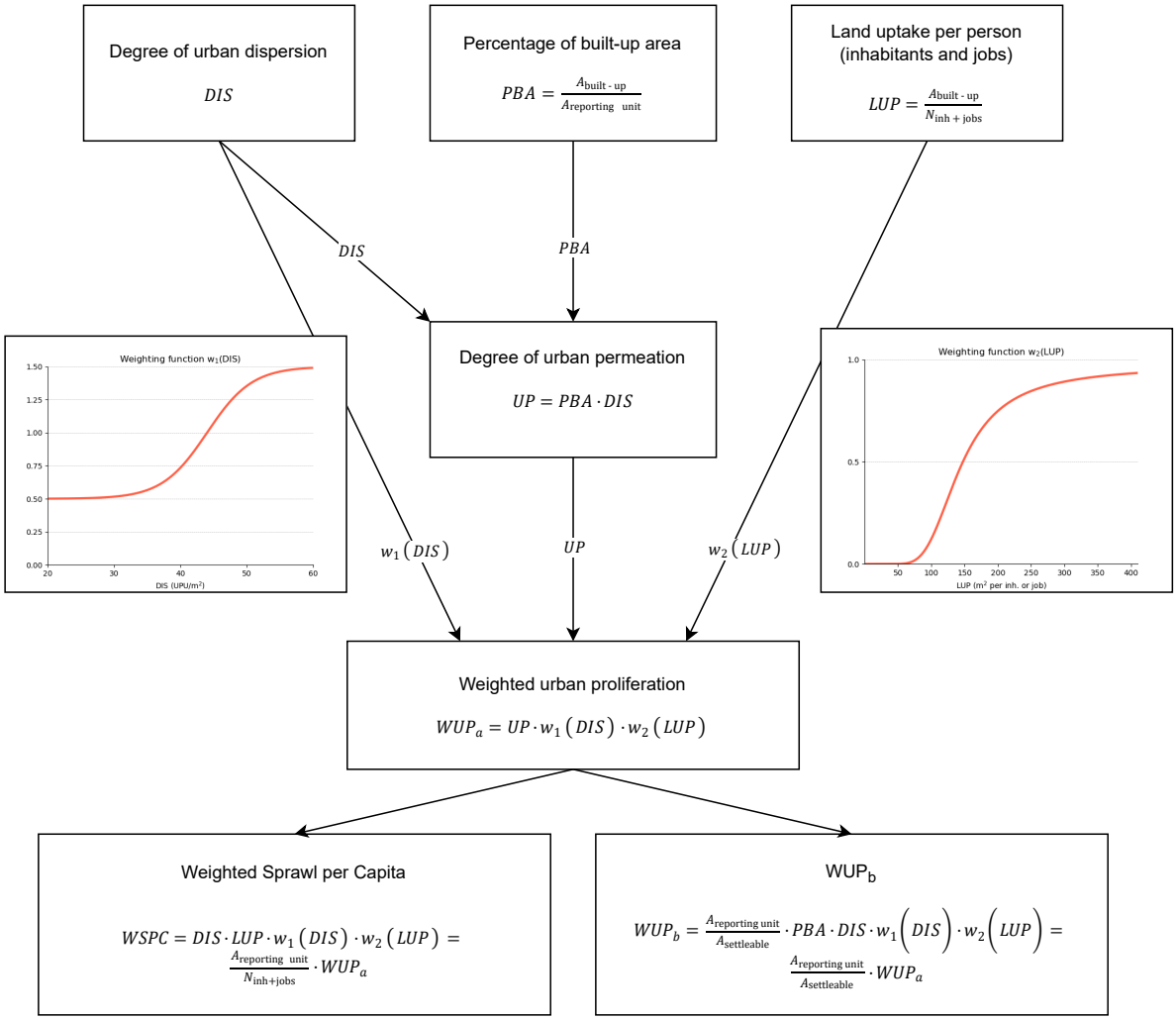

Abbildung 7.1: Zusammenhänge und Berechnungsschritte der Messwerte der Zersiedelung nach Schwick und Jäger [1]

7.2. Berechnungsschritte

Um ein Beispiel zu geben, kann man sich die Berechnung für die Metriken der Zersiedelung aus Abbildung 7.1 wie folgt vorstellen:

Als Grundlagen zur Berechnung dienen die Metriken DIS (Degree of urban dispersion), PBA (Percentage of built-up area) und LUP (Land uptake per person). PBA ist die Proportion der bebauten Fläche zu der zu berechnenden Fläche in Prozent. LUP ist die Flächeninanspruchnahme pro Person der bebauten Fläche und wird in Quadratmetern pro Person ausgedrückt. Der DIS misst die Streuung von bebauten Gebieten anhand der Entfernungen zwischen zwei beliebigen Punkten innerhalb des bebauten Gebietes in der Einheit UPU (Urban Permeation Units) pro Quadratmeter der bebauten Fläche.

Sind diese Metriken bekannt, kann man daraus weitere berechnen. So kann man die Metrik UP (Urban Permeation) berechnen, welche die Durchdringung einer Landschaft durch bebaute Gebiete misst und in urbanen Durchdringungseinheiten pro Quadratmeter der Landschaft (UPU/m^2) ausgedrückt wird. Des Weiteren ergibt sich die Metrik WUP (Weighted Urban Proliferation = Zersiedelung Z), welche die Zersiedelung quantifiziert und in urbanen Durchdringungseinheiten pro Quadratmeter der Landschaft (UPU/m^2) angibt. Die Metrik WUP ist das Produkt aus UP, dem Ergebnis einer Gewichtungsfunktion für DIS und dem Ergebnis einer Gewichtungsfunktion für LUP.

Die Gewichtungsfunktionen sind so definiert, dass stärker zerstreute bebaute Gebiete ein höheres Gewicht erhalten als weniger zerstreute bebaute Gebiete. Dies ist insofern sinnvoll, da kompakter bebaute Gebiete niedrigere DIS-Werte haben als stärker zerstreut bebaute Gebiete [14].

Aus der Metrik WUP lassen sich zwei weitere Metriken ableiten: WSPC (Weighted Sprawl per Capita) [15] und WUP_b (Weighted Urban Proliferation of the settleable part of the study area) [16].

7.2.1. Metriken der Zersiedelung

Die Weighted Urban Proliferation (WUP) besteht aus drei Komponenten: Degree of urban dispersion (DIS), Proportion of built-up areas (PBA) und Land Uptake per person (LUP) (oder Utilization Density, UD) [1]. Neben den Hauptkomponenten sind die Gewichtungsfunktionen w_1, w_2 für DIS und LUP definiert. Zum Schluss wird die Metrik Total Sprawl (TS) erläutert, die mithilfe der Webapplikation berechnet werden kann.

DIS

Name der Metrik	Degree of Urban Dispersion *Dispersion*
Beschreibung	Die Metrik misst die Streuung von bebauten Gebieten anhand der Entfernungen zwischen zwei beliebigen Punkten, die innerhalb des bebauten Gebietes liegen. Die Grundlage des DIS wird in dieser Arbeit als SI-Raster bezeichnet. Dieses Raster beinhaltet für jeden Pixel (bebaute Fläche) die gemittelte Summe der Distanzen (Euklidischer Abstand [17]) 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. Daher haben kompakter bebaute Gebiete niedrigere DIS-Werte als stärker zerstreut bebaute Gebiete.
Formel [1]	$DIS(b) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{n_i}\left(\sum_{j=1}^{n_i}\left(\sqrt{\frac{2\cdot d_{ij}}{1m}+1}-1\right)\frac{DSE}{m^2}+ZIB(b)\right)$ $ZIB(b)$ 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 (oder 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 der Metrik	Proportion of Built-up Areas *Anteil der Siedlungsfläche (ASF)*
Beschreibung	Die Metrik beschreibt das Verhältnis zwischen der Grösse der bebauten Flächen und der Grösse des Untersuchungsgebiets.
Formel	$PBA = A_{built-up}/A_{reporting-unit}$
Einheit	%

LUP

Name der Metrik	Land Uptake per person *Flächeninanspruchnahme pro Person (FI)*
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 eigentlich der Kehrwert von UD: $LUP = 1/UD$
Formel	$LUP = A_{built-up}/N_{inh+job}$
Einheit	m^2 pro Einwohner oder Arbeitsplatz

UD

Name der Metrik	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 der Metrik	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 Untersuchungsgebiets ausgedrückt.
Formel	$UP = PBA \cdot DIS$
Einheit	UPU/m^2 des Untersuchungsgebiets

WUP

Name der Metrik	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 Werte zwischen 0,5 und 1,5 annimmt, 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 Werte zwischen 0 und 1 annimmt, 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 der Metrik	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 [7].
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 der Metrik	Weighted Sprawl per Capita *Gewichtete Zersiedelung pro Kopf*
Beschreibung	Diese Metrik 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 der Metrik	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

8

Sicherheit

8.1. Allgemein

Die Gewährleistung der Datensicherheit stellt einen wesentlichen Aspekt bei der Erweiterung dar. Dies ist insbesondere darauf zurückzuführen, dass für die Berechnung der Metriken Statistikdaten verwendet werden, die potenziell als heikel eingestuft werden können. Darüber hinaus ist sicherzustellen, dass die Daten des Benutzers nicht ohne dessen Kenntnis weitergegeben werden.

Konkret handelt es sich in diesem Fall um Daten, die als Eingangsgrösse zur Berechnung der Zersiedelung durch den Benutzer bereitgestellt werden.

8.2. Persistenz im Webserver

Der USM-Calculator speichert weder Benutzereingaben noch Resultate, die aus den Eingaben errechnet werden, über die Dauer der Sitzung des Benutzers hinaus ab. Während der Berechnung werden jedoch in einigen Schritten Zwischenresultate gespeichert. Die Vektordatei zur Begrenzung des Rasters wird als Shapefile (`.shp`) zwischengespeichert. Dieses zwischengespeicherte Polygon (Shapefile) wird danach verwendet, um das Raster entlang der Shape-Grenzen zuzuschneiden, wobei dieser Ausschnitt des Rasters ebenfalls gespeichert wird (`.tif`). Diese beiden temporären Dateien und die Resultate werden nach Fertigstellung der Berechnung nach vorzeitigem Abbruch durch Auftreten eines Fehlers oder Beenden der Sitzung durch den Benutzer gelöscht. Somit wird sichergestellt, dass die Dateien in keinem Fall auf dem Server verbleiben.

8.3. Verantwortlichkeit

Die Verantwortung der Daten liegt beim Benutzer unseres Webservices, weil die Eingabedaten als Grundlage zur Berechnung von ihm zur Verfügung gestellt werden.

9
Design

9.1. Architektur

In der Umsetzung der Erweiterung durch einen Webservice wird eine Clean Architectur des Codes nach Robert C. Martin angestrebt [18]. Die Clean Architecture ist ein Architekturmuster, das darauf abzielt, die Abhängigkeiten zwischen den einzelnen Schichten zu minimieren und die Geschäftslogik von der Infrastruktur zu trennen (vgl. Abbildung 9.1). Das Backend ist aufgeteilt in die Schichten Controller, Services und Models. Eine wichtige Abgrenzung ist in der Clean Architektur definiert: Die `api` Boundary für die REST API, die zwischen Vue.js Frontend und Python Backend kommuniziert. Aufgrund der Trennung mit der `api` Boundary kann das Frontend unabhängig vom Backend entwickelt werden.

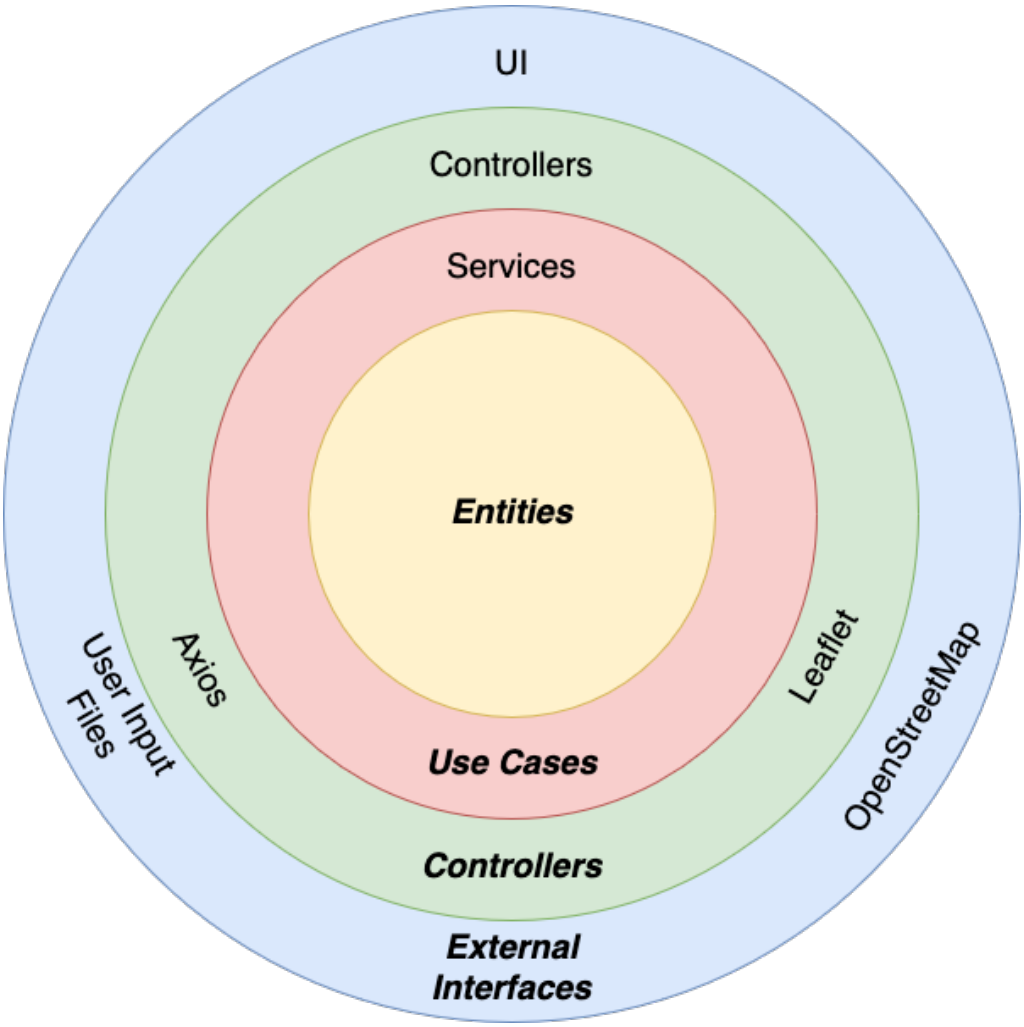

Abbildung 9.1: Clean Architektur Diagramm nach Robert C. Martin. (Eigene Darstellung)

9.2. Technologien und Designentscheidungen

Das Backend, bestehend aus einer REST API, wurde in Python mit dem Starlette-Framework entwickelt. Python wurde für das Backend gewählt, weil die Grundlage des Projekts, das QGIS-Plugin, ebenfalls in Python geschrieben ist. Die Familiarität mit der Programmiersprache und der Umfang an Bibliotheken für Geodatenverarbeitung in Python wie GDAL waren weitere Gründe für die Wahl von Python.

Das Frontend wurde in Typescript mit dem Vue.js-Framework entwickelt. Gewählt wurde Vue.js, weil es ein modernes und populäres Framework für die Entwicklung von Single-Page-Webanwendungen ist, wobei die Vue Komponenten in zwei verschiedenen API-Designs entwickelt werden können: Options API und Composition API [19]. Die Options API ist die traditionelle Art, Vue-Komponenten zu entwickeln, während die Composition API eine neue Art ist, Komponenten zu entwickeln, die in Vue 3 eingeführt wurde. Ausserdem wurde der Ansatz der Single-File-Components verfolgt, wobei HTML, CSS und Typescript in einer Datei zusammengefasst werden [20]. Für dieses Projekt wurde die Composition API verwendet, da sie eine flexiblere Code-Struktur ermöglicht und die Wiederverwendbarkeit von Code erhöht [19]. Für die Darstellung der interaktiven Karte wurde Leaflet inklusive einer Bibliothek für eine vereinfachte Integration in eine Vue-Applikation ausgewählt, eine Open-Source Bibliothek für die Darstellung von Kartenmaterial in Webanwendungen. Die Tile-Layer der Karte stammen von OpenStreetMap, einer freien Weltkarte, die von einer großen Gemeinschaft von Freiwilligen erstellt wird.

Um die effizienteste Berechnung der Zersiedelungsmetriken zu ermöglichen, wurde Numba, eine Bibliothek zur Beschleunigung von Python-Code durch Just-in-Time-Kompilierung[21], verwendet. Nebenbei bietet Numba auch Funktionen zur Parallelisierung von Code, was die Berechnung der Metriken beschleunigt [22]. Für die Evaluation der Berechnungsperformance für die Parallelisierung wurden auch andere Libraries wie Joblib und die Standardbibliothek multiprocessing von Python integriert.

In der Entwicklung des Projekts wurde auf den Einsatz einer Datenbank verzichtet, um die Komplexität des Projekts zu reduzieren und datenschutztechnische Probleme zu vermeiden.

9.2.1. Tools

Nachfolgend sind alle wichtigen Tools aufgelistet, die eine Anwendung in der Entwicklung des Projekts finden. Für das Projekt wurden nach Möglichkeit jeweils die neuesten stabilen Versionen der Tools verwendet.

- **Python** [23]: Eine interpretierte, objektorientierte Programmiersprache, die für ihre einfache Syntax und Lesbarkeit bekannt ist.
- **GDAL** [24]: Eine Open-Source-Bibliothek für Geodatenverarbeitung, die viele Funktionen für die Verarbeitung von Raster- und Vektordaten bietet.
- **Starlette** [25]: Ein schnelles, flexibles und leichtgewichtiges Web-Framework für Python, das auf ASGI basiert.
- **Numba** [26]: Eine Bibliothek zur Beschleunigung von Python-Code durch Just-in-Time-Kompilierung und Funktionen zur Parallelisierung von Code.
- **Joblib** [27]: Eine Bibliothek zur Parallelisierung von Code in Python.
- **Numpy** [28]: Eine Bibliothek für numerische Berechnungen und Operationen mit großen, mehrdimensionalen Arrays und Matrizen.
- **Uvicorn** [29]: Ein ASGI-Server.
- **Pytest** [30]: Ein Framework für das Testen von Python-Code.
- **Poetry** [31]: Ein Tool zur Verwaltung von Python-Abhängigkeiten und virtuellen Umgebungen.
- **Typescript** [32]: Typensichere Programmiersprache, die zu Javascript kompiliert wird.
- **Vite** [33]: Ein Build-Tool und Development-Webserver für moderne Webanwendungen, das auf Rollup, ein Javascript Modul Bundler, basiert.
- **Vue.js** [34]: Ein populäres JavaScript-Framework zur Entwicklung von Webanwendungen.
- **Leaflet** [35]: Eine Open-Source JavaScript-Bibliothek zur Darstellung interaktiver Karten. Konkret wird die Erweiterung "Vue Leaflet" [36] verwendet, die einfach mit Vue.js verwendet werden kann.
- **Ant Design** [37]: Eine ursprüngliche React-Komponentenbibliothek, für die es auch eine Vue-Version gibt.
- **Axios** [38]: 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** [39]: 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** [40]: 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** [41]: Ein JavaScript-Bibliothek, die das Lesen von Rasterdaten ermöglicht und diese Daten in der Leaflet Karte visuell darstellen kann.
- **Flake8** [42]: Ein Tool zur statischen Codeanalyse in Python, das die Einhaltung des PEP8-Standards[43], ein Style Guide für Python, überprüft.
- **Pylint** [44]: Ein weiteres Tool zur statischen Codeanalyse in Python, das die Codequalität überprüft.
- **Mypy** [45]: Ein statischer Typenchecker für Python.
- **Eslint** [46]: Ein Linter für Typescript, um die Codequalität und Konsistenz zu gewährleisten.
- **QGIS** [47]: Ein Open-Source Geoinformationssystem, das zum Anzeigen und Bearbeiten von Geodaten verwendet wird. Unterschiedliche Plugins erweitern die Funktionalität von QGIS. QGIS wurde in diesem Projekt verwendet, um die Zersiedelungsmetriken zu berechnen und die Resultate manuell mit den berechneten Metriken des Webservices zu vergleichen.

9.2.2. Designentscheidungen

In diesem Abschnitt werden die Designentscheidungen für das Projekt und die Begründungen zu den jeweils verwendeten Technologien erläutert.

MVVM Pattern
Im Frontend wurde das MVVM (Model-View-ViewModel) [48] Pattern verwendet, um die Trennung von Logik und Darstellung zu gewährleisten. Durch den Einsatz von Vue.js und der Composition API [19] ist das MVVM Pattern vordefiniert und einfach umzusetzen. In Vue.js werden `.vue`-Dateien verwendet, die die Darstellung (View) und die Logik (ViewModel) in einer Datei zusammenfassen. Dabei sind die Darstellung und die Logik klar getrennt, was die Wartbarkeit und Erweiterbarkeit des Codes erhöht. Die Darstellung befindet sich im `<template>`-Tag, die Logik im `<script>`-Tag und die Styles im `<style>`-Tag.

RESTful API
RESTful-APIs besitzen drei Hauptvorteile: Skalierbarkeit, Flexibilität und Unabhängigkeit [49]. Dabei ist die RESTful-API zustandslos, was bedeutet, dass der Server keine Informationen über die Client-Sitzung speichert. Dadurch kann eine einfache horizontale Skalierung des Servers ermöglicht werden, weil der Server den Zustand des Clients nicht verfolgen muss. Dies entlastet den Server, weil er keine Sitzungsinformationen speichern muss.

Flexibilität und Unabhängigkeit werden durch die strikte Trennung von Client (Webapplikation) und Server erreicht. Der Client könnte ausgetauscht werden, ohne den Server zu verändern und umgekehrt. Ausserdem können Client und Server in verschiedenen Programmiersprachen geschrieben sein und trotzdem miteinander kommunizieren. Ein weiterer Vorteil ist, dass die Technologien unabhängig voneinander aktualisiert werden können, wodurch das System flexibler und leichter zu warten ist.

Verzicht einer Datenbank
Um die Komplexität des Projekts zu reduzieren, wird auf den Einsatz einer Datenbank verzichtet. Hauptgrund für den Verzicht ist jedoch, dass eine Speicherung der Daten datenschutztechnisch kompliziert ist. Da es sich bei den Input-Daten und den Resultaten um teils sensible Statistikdaten handelt, ist eine Speicherung nur mit entsprechenden Massnahmen wie API-Keys, Verschlüsselung und Zugriffsbeschränkungen möglich, wobei eine Umsetzung den Projektumfang sprengen würde.

Monorepo
Das Projekt wird in einem Monorepo [50] verwaltet. Wir haben uns für ein Monorepo entschieden, weil wir dadurch das Front- und Backend in einem Repository verwalten können. Dies ermöglicht eine einfachere Verwaltung der Abhängigkeiten möglich und erleichtert nebenbei die Entwicklung und das Deployment.

9.3. Deployment

Mittels Docker [51] Compose kann eine Gruppe von Containern zusammen gestartet werden. Dabei wurde für Front- und Backend jeweils ein `Dockerfile` erstellt. Diese Datei beschreibt die Laufzeitumgebung der Schicht und kreiert ein Image zur einfachen Replikation.

Zwischen den beiden Services (Frontend und Backend) wird ein traefik [52] Load-Balancer erstellt. Mit dem Load-Balancer kann durch Hinzufügen weiterer Replikas z.B. ein zweites Backend, die Ausfallsicherheit und die Zuverlässigkeit der Applikation erhöht werden. Fällt ein Backend aus, kann ein zweiter Container zum Einsatz kommen.

9.3.1. Deployment Diagramm

Im Deployment Diagramm (Abbildung 9.2) ist der Aufbau des Docker Compose ersichtlich. Ausserdem sieht man, wie die einzelnen Container miteinander kommunizieren und wie sie deployed werden. Das Deployment beginnt mit dem Erstellen eines Releases auf Gitlab, was manuell erfolgt. Der Gitlab-Deplyoment-Prozess erstellt ein Docker-Image und pusht dieses auf das Gitlab-Registry. Danach wird das Docker-Image auf den Server deployed und gestartet.

Ein Gitlab Runner übernimmt die Aufgabe des Deployments und startet die Container. Die Schritte des Deployments aus Abbildung 9.2 sind nachfolgend aufgelistet:

1. Der Gitlab Runner scannt das Repository für neue CI/CD Jobs.
2. Wird ein neuer Job gefunden, startet der Gitlab Runner eine neue Gitlab Runner Instanz als Docker Container. In diesem Container wird der Job für das Deployment auf den Server ausgeführt.
3. Der Deployment Job etabliert eine SSH-Verbindung zum Server und kopiert das `docker-compose.yml` File auf den Server. Die Datei enthält die Konfiguration für das Deployment (traefik, Frontend, Backend).
4. Docker Compose instruiert den Server, die benötigten Images (traefik) von Docker Hub und die eigenen Images von der Gitlab Registry zu laden.

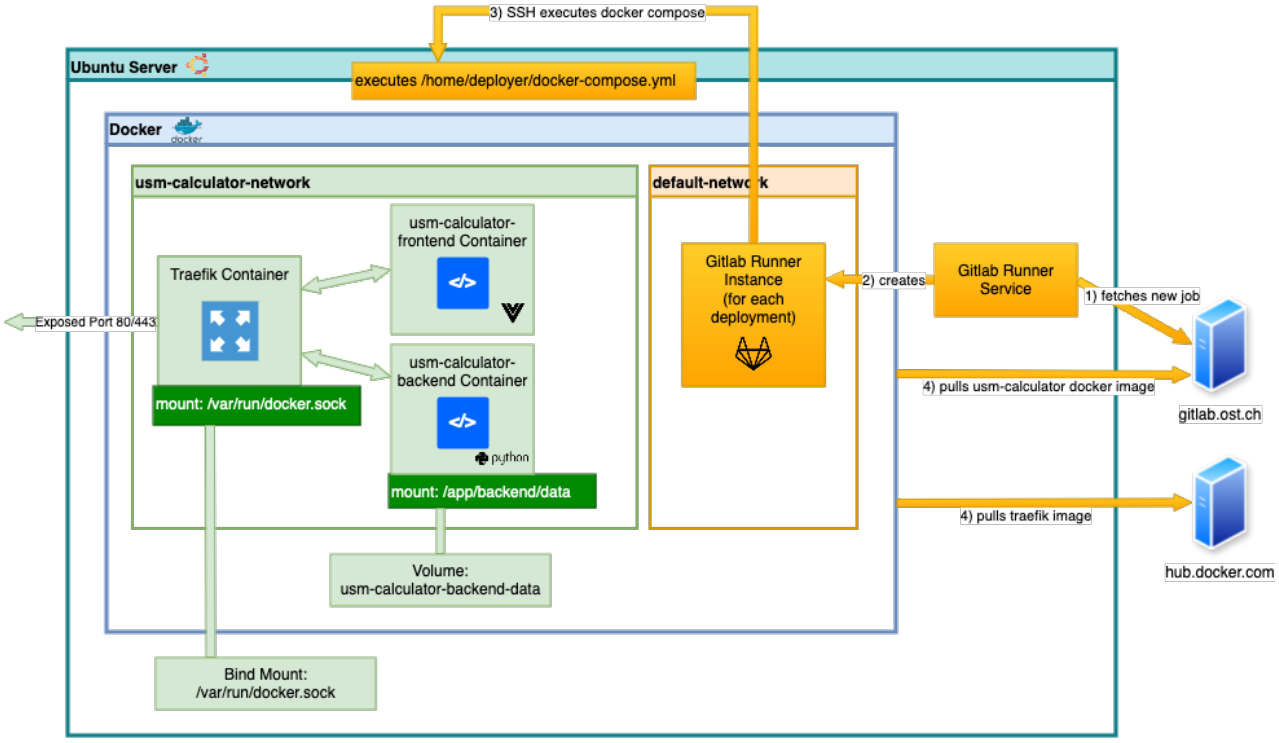

Abbildung 9.2: Deployment Übersicht des Systems.

9.4. Paket- und Modulstrukturen

9.4.1. Frontend (Typescript-Module)

Die Typescript-Module des Frontends sind in die drei Hauptbereiche `components`, `services` und `views` unterteilt. Die `components` enthalten die Vue-Komponenten, die für die Darstellung der Benutzeroberfläche zuständig sind. Die `services` enthalten die Logik für die Kommunikation mit dem Backend. Die `views` enthalten die Vue-Komponenten, die die Seiten der Webapplikation darstellen.

9.4.2. Backend (Python Packages)

Das Paketdiagramm in Abbildung 9.3 dient zur Übersicht der Packages im Python-Backend. Gestrichelte Linien und offene Pfeilen zeigen die Richtung der Abhängigkeiten (wird verwendet von). Durchgezogene Linien und leere, geschlossene Pfeile demonstrieren die Richtung der Vererbung (erbt von).

Nahezu der gesamte Teil des Python-Codes befindet sich im Unterverzeichnis `/app`. Nur der Einstieg in die Applikation, welcher das Starlette Framework anwendet, `main.py`, befindet sich im Wurzelverzeichnis. Im Verzeichnis `/backend` befinden sich die Teile des Codes, die mit der Starlette-Framework arbeiten. Dabei handelt es sich um eine ”Relaxed Layer Architecture”, wie Robert C. Martin sie beschreibt [18]. Das bedeutet, dass die Abhängigkeiten von aussen nicht direkt auf das nächstinnere Paket, sondern auch auf tiefere Pakete zeigen können.

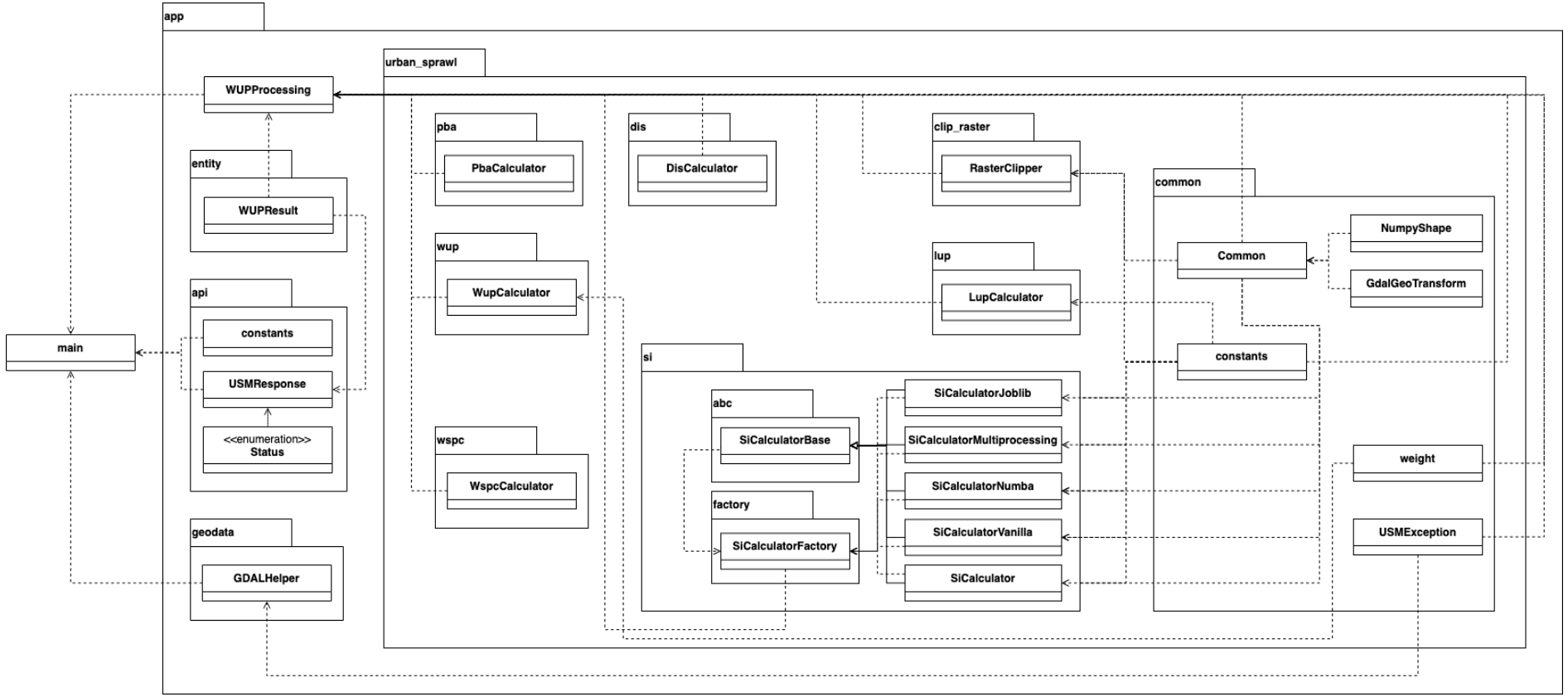

Abbildung 9.3: Python-Packagestruktur des Backends. Pfeile mit gestrichelten Linien zeigen Abhängigkeiten zwischen den Packages.

9.5. Sequenzdiagramme

In folgendem Sequenzdiagramm in Abbildung 9.4 ist die vollständige Berechnung der Zersiedelung der einzelnen Features aus dem Vektor-Layer der Untersuchungsfläche dargestellt.

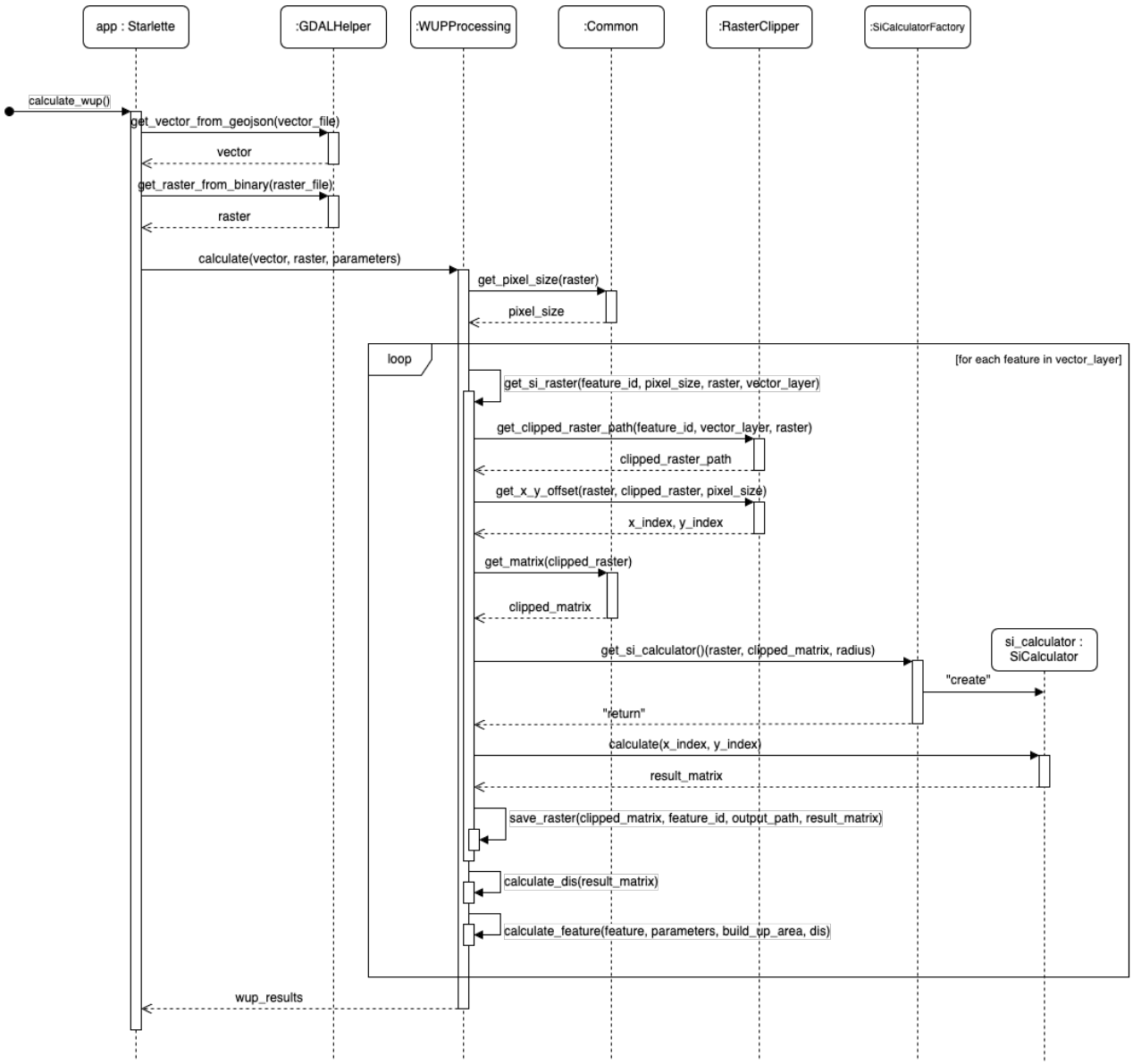

Abbildung 9.4: UML Sequenzdiagramm der vollständigen Berechnung der Zersiedelung der einzelnen Features in einem Vektor-Layer.

In dem nachfolgenden Sequenzdiagramm in Abbildung 9.5 ist die Interaktion des Benutzers mit der API über die Benutzeroberfläche der Webapplikation dargestellt. Dabei wird der gesamte Prozess der Benutzereingaben bis zur Berechnung der Zersiedelung und der Rückgabe der Resultate dargestellt. Die Funktionen im Diagramm sind Pseudo-Code und entsprechen nicht dem tatsächlichen Code, sie stellen nur die Interaktionen dar.

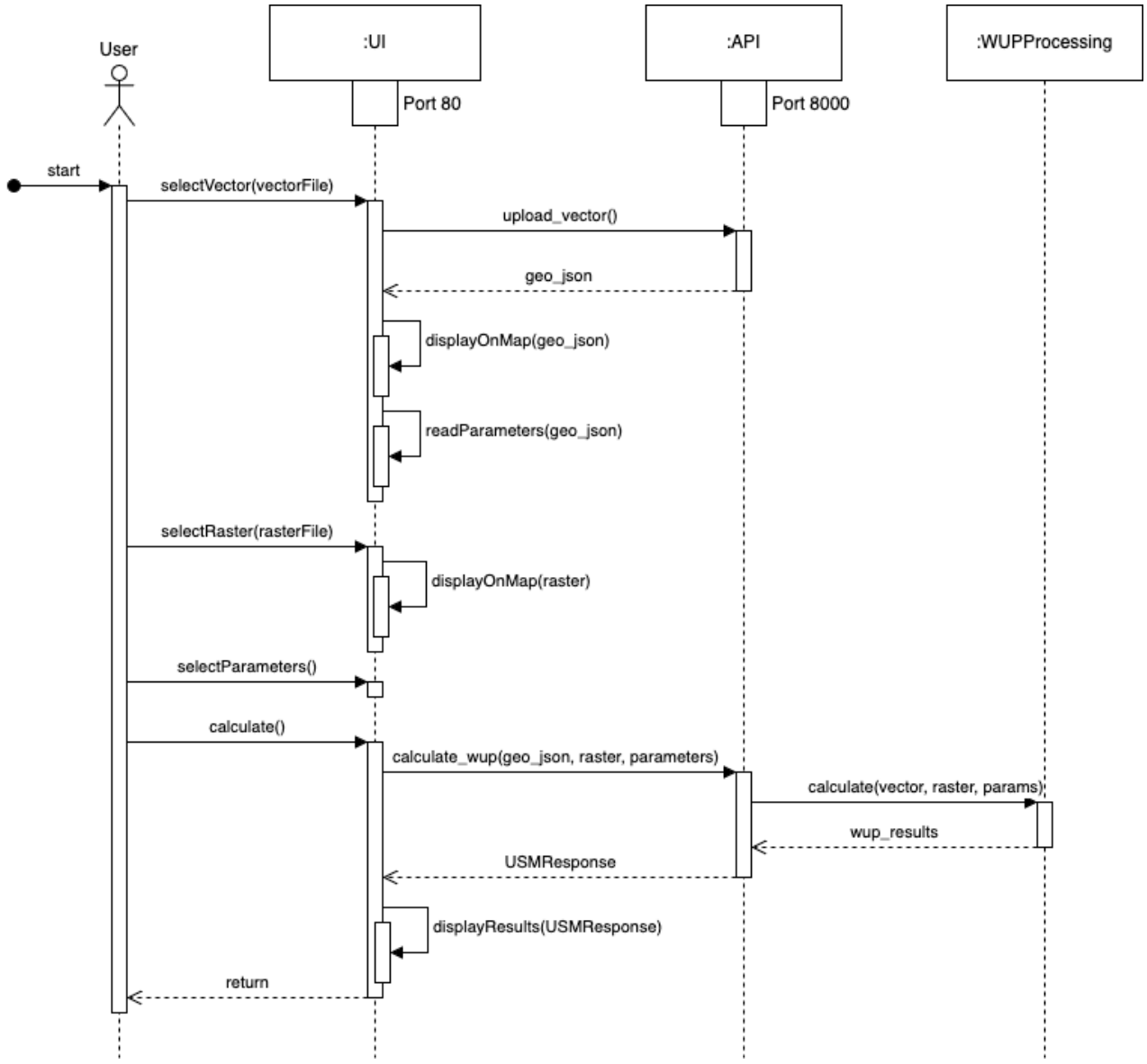

Abbildung 9.5: UML Sequenzdiagramm der Interaktion des Benutzers mit der API über die Benutzeroberfläche der Webapplikation.

9.6. UI Design

Für das Styling der Komponenten des Frontends wurde eine Testversion der Software Balsamiq Mockups [53] verwendet. Balsamiq bietet eine grosse Palette an vordefinierten Komponenten zur Benutzereingabe und für das Layout der Webseite.

Das Layout ist einfach gehalten und stellt die Karte und somit das Raster und die Vektordatei in den Vordergrund. Weil diese Dateien als wichtigste Grundlage der Berechnung dienen, wurde eine Karte mittels Leaflet eingebunden. Durch diese visuelle Stütze, ist es für den Benutzer einfacher sich vorzustellen, wie das Raster aussieht und welche Bereiche davon berechnet werden. Die Leaflet Erweiterung ist ein exzellentes Tool für die Darstellung von Polygonen (Shapes), Flächen und Punkten. Durch die ausführliche Dokumentation, lässt sich die Karte simpel interaktiv gestalten.

Momentan werden für die Darstellung des Kartenmaterials Tiles aus OpenStreetMap [54] verwendet. Diese Darstellung könnte in einer Weiterentwicklung dynamisch ausgetauscht werden, z.B. mit einer Möglichkeit des Benutzers auf Swisstopo Tiles (z.B. mit dem Leaflet Plugin "Leaflet.TileLayer.Swiss" [55]) zu wechseln.

9.6.1. Skizzen der Benutzeroberfläche

Die Benutzeroberfläche der Webapplikation wird sehr einfach gehalten. Nur die nötigsten Felder für die Berechnung der Zersidelungsmetriken sind vorhanden. In der Abbildung 9.6 ist ein erster Entwurf der Benutzeroberfläche des Webtools zu sehen.

Erster Entwurf der Benutzeroberfläche
In einem ersten Entwurf (Abbildung 9.6) ist die Oberfläche sehr einfach gehalten. Das Design orientiert sich stark an der Benutzerschnittstelle des bestehenden QGIS Plugins.

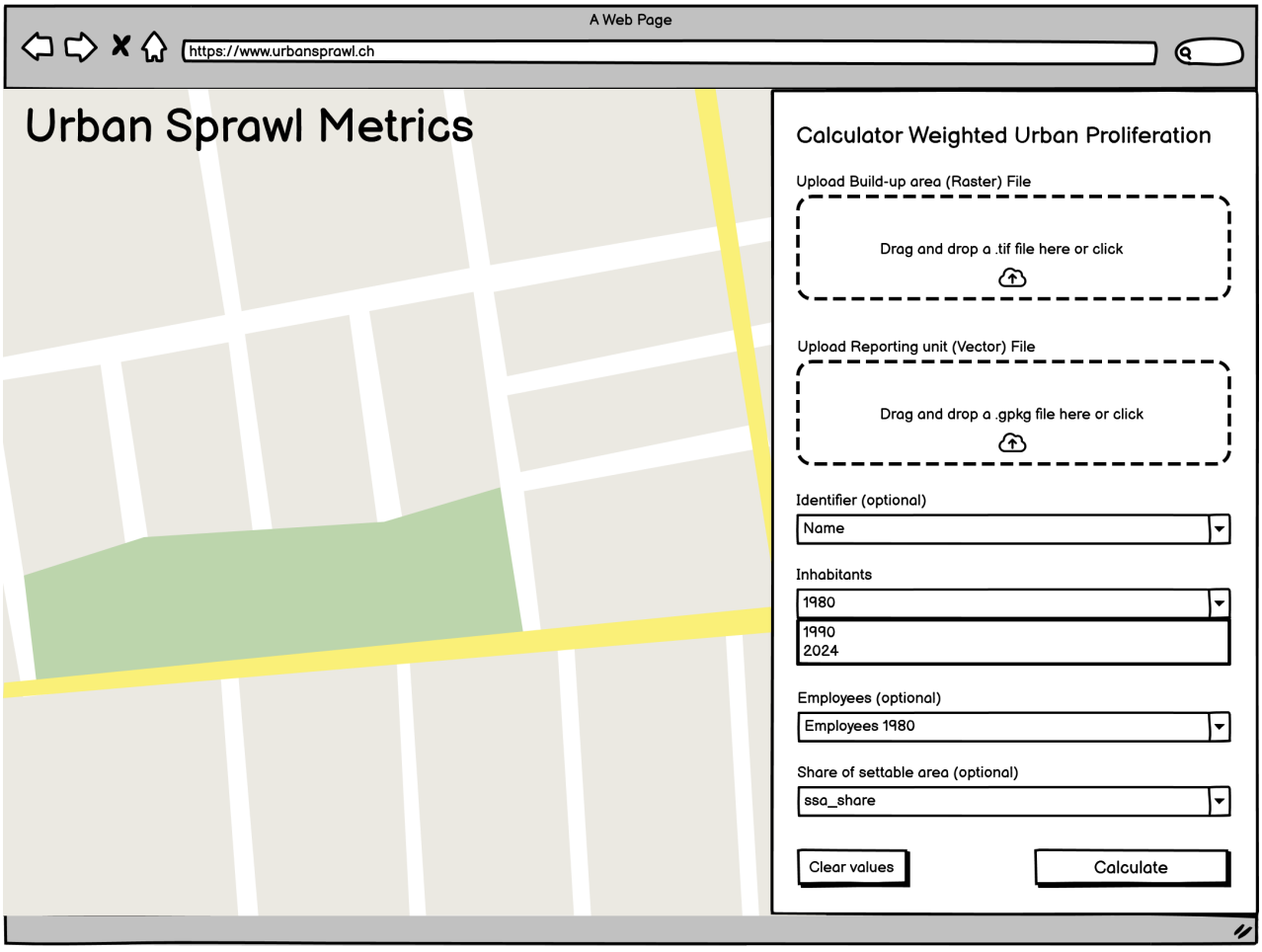

Abbildung 9.6: Ein erster Entwurf der Benutzeroberfläche des Webtools.

9.6.2. Weitere Entwürfe im Laufe des Projekts

Nachstehende Mockups in den Abbildungen 9.7, 9.8 und 9.9 zeigen weitere Entwürfe, die im Laufe des Projekts entstanden sind. Der Fokus lag dabei auf einer einfachen und intuitiven Benutzeroberfläche, welche die Berechnung der Zersiedelungsmetriken ermöglicht.

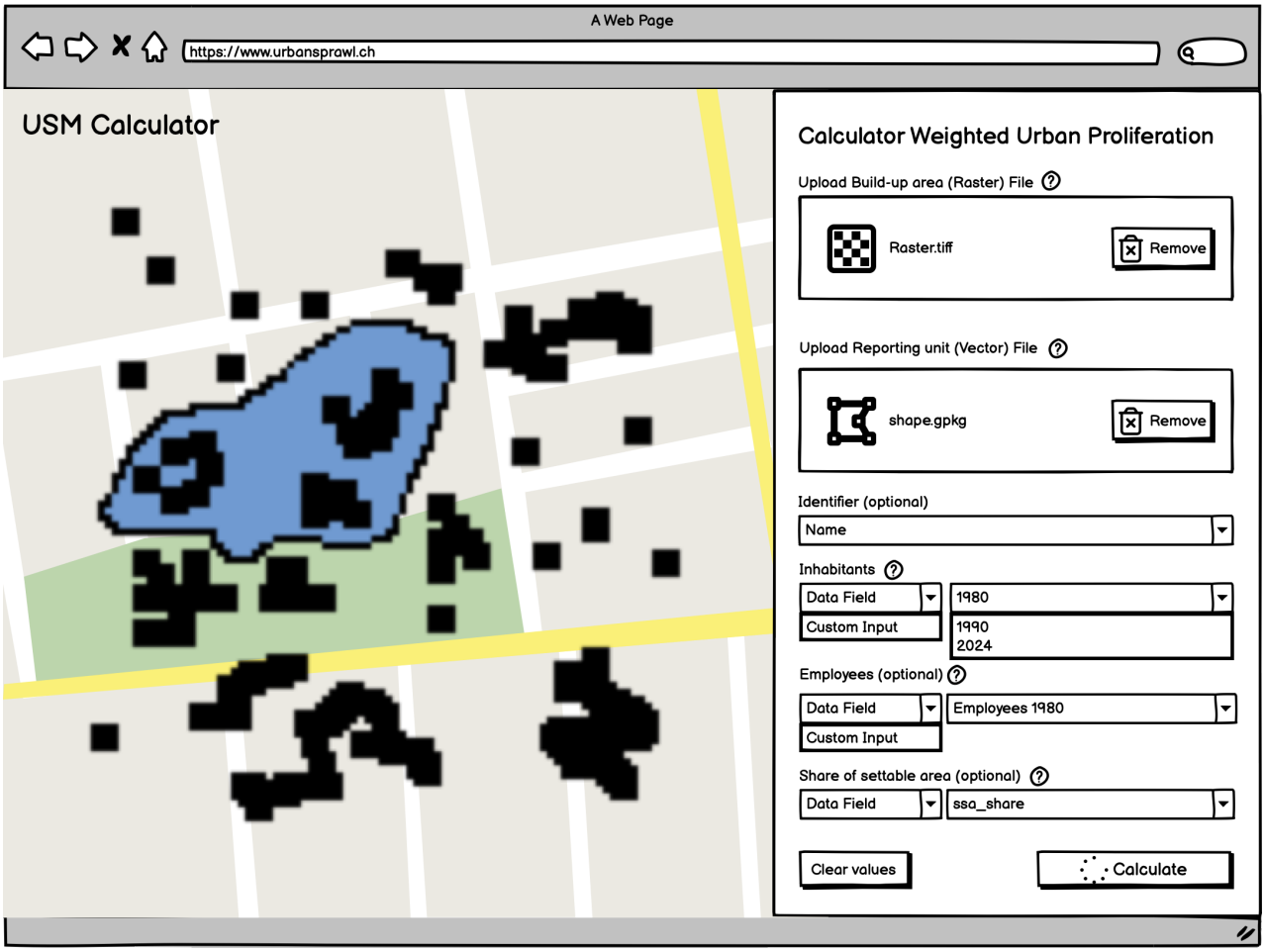

Abbildung 9.7: Die Karte auf der linken Seite dient der Darstellung des Raster und des Vektor-Layers. Auf der rechten Seite sind die Eingabefelder für die Berechnung der Zersiedelungsmetriken.

In der Abbildung 9.9 kann die Tabelle Resultate für mehrere Features in einem Vektor-Layer anzeigen. Mit einem Klick auf die Pfeile oberhalb der Tabelle können die einzelnen Resultate der jeweiligen Features angezeigt werden. Diese Funktionalität wurde nicht im Rahmen des Projekts implementiert, kann aber in einer zukünftigen Version des Webtools hinzugefügt werden.

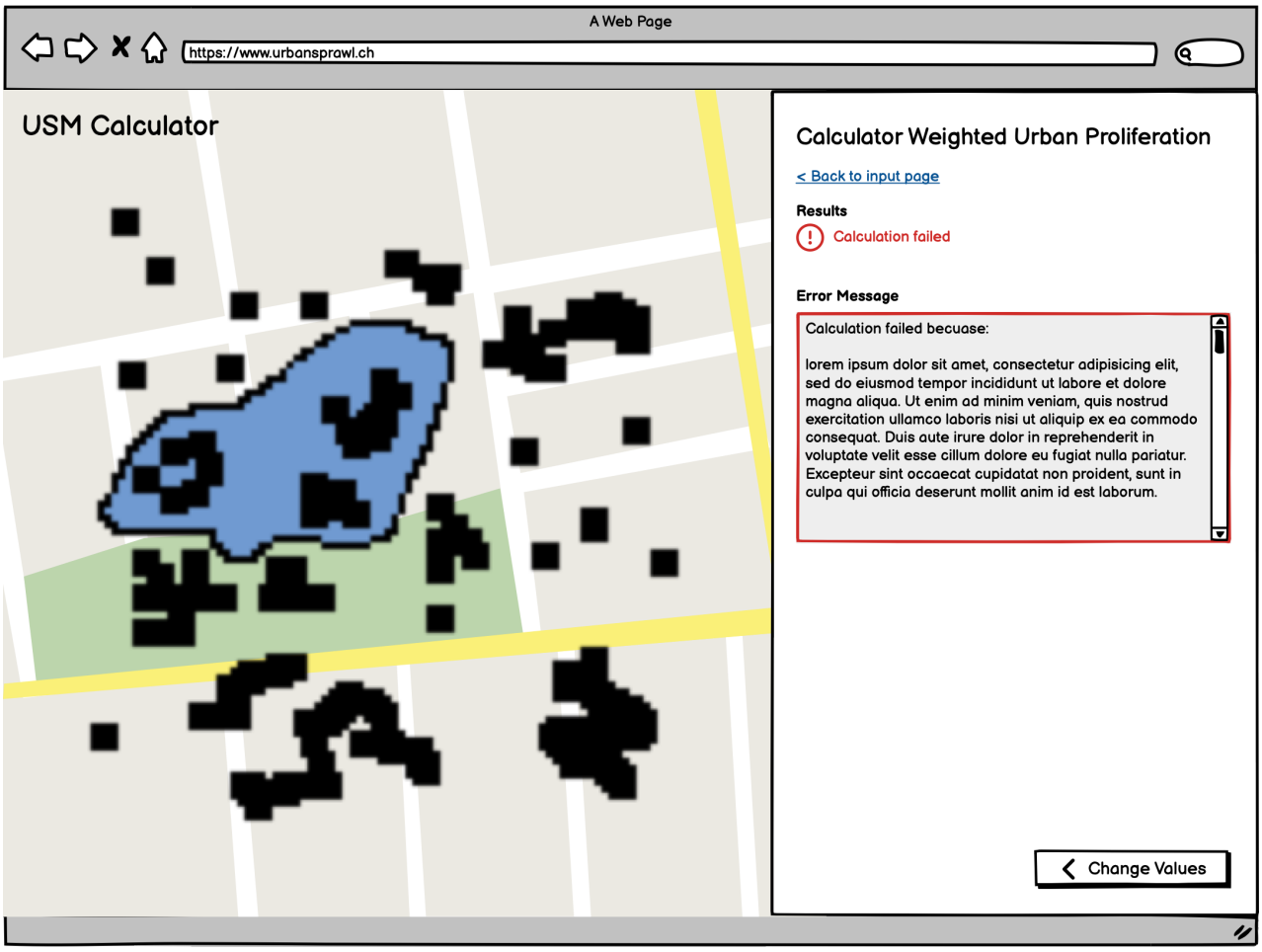

Abbildung 9.8: Eine sinnvolle Fehlermeldung wird angezeigt, wenn die Berechnung fehlschlägt.

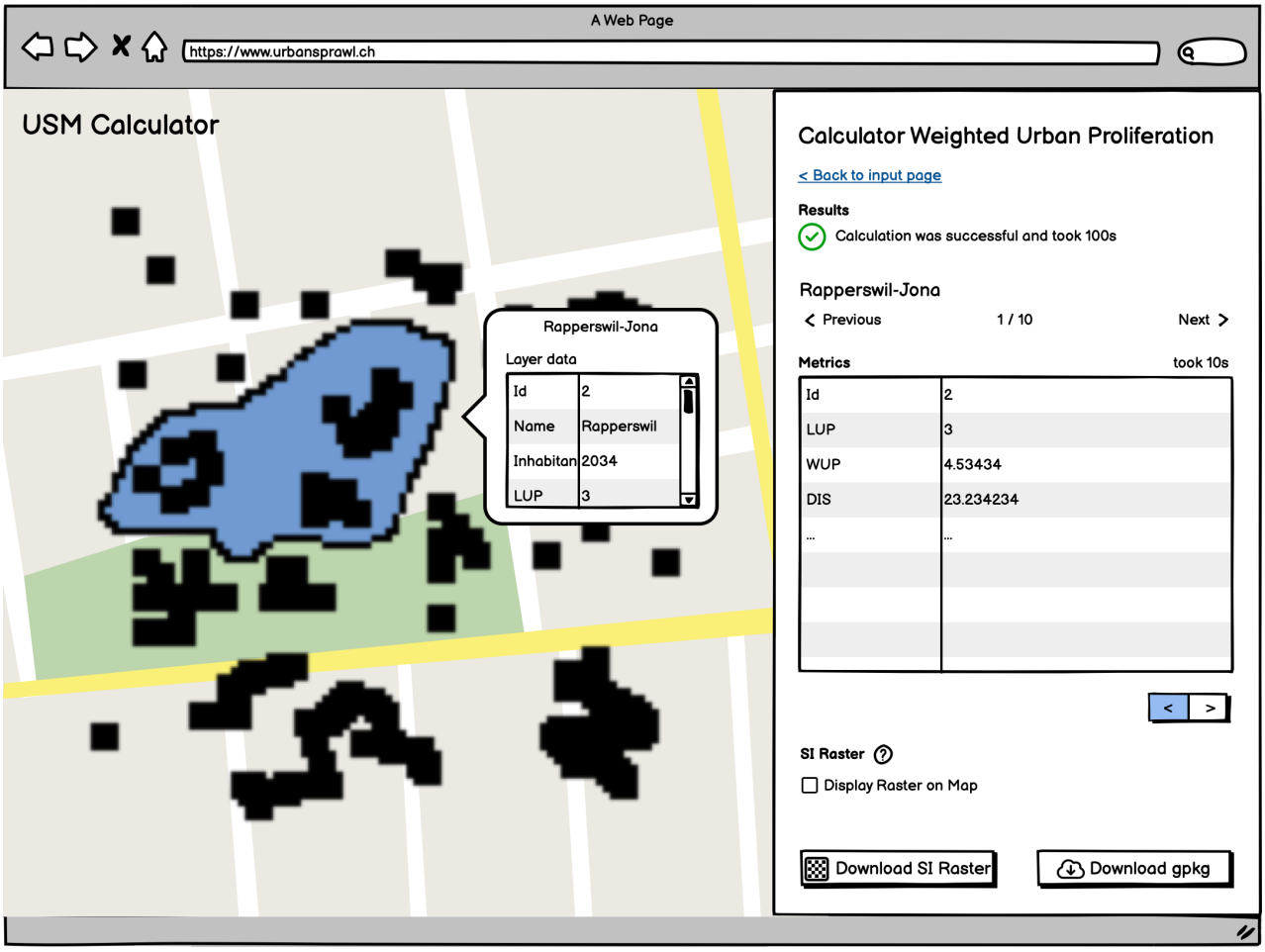

Abbildung 9.9: Die Resultate werden in einer Tabelle dargestellt. Auf der Karte können diese Resultate in einem Popup des Vektor-Layers angezeigt werden.

9.6.3. Endgültiges Design

In den Abbildungen 9.10 und 9.11 ist das endgültige Design der ”USM-Calculator” Webapplikation zu sehen. Das Umsetzen des Designs erfolgte mit dem Vue.js Framework und der Ant Design Komponentenbibliothek. Die geplante Funktionalität, um die Input-Parameter für die Berechnung manuell abzuändern, um von den Feature-Attributen abzuweichen, wurde aus Zeitgründen nicht umgesetzt. Eine Weiterentwicklung des Webtools könnte diese Funktionalität hinzufügen.

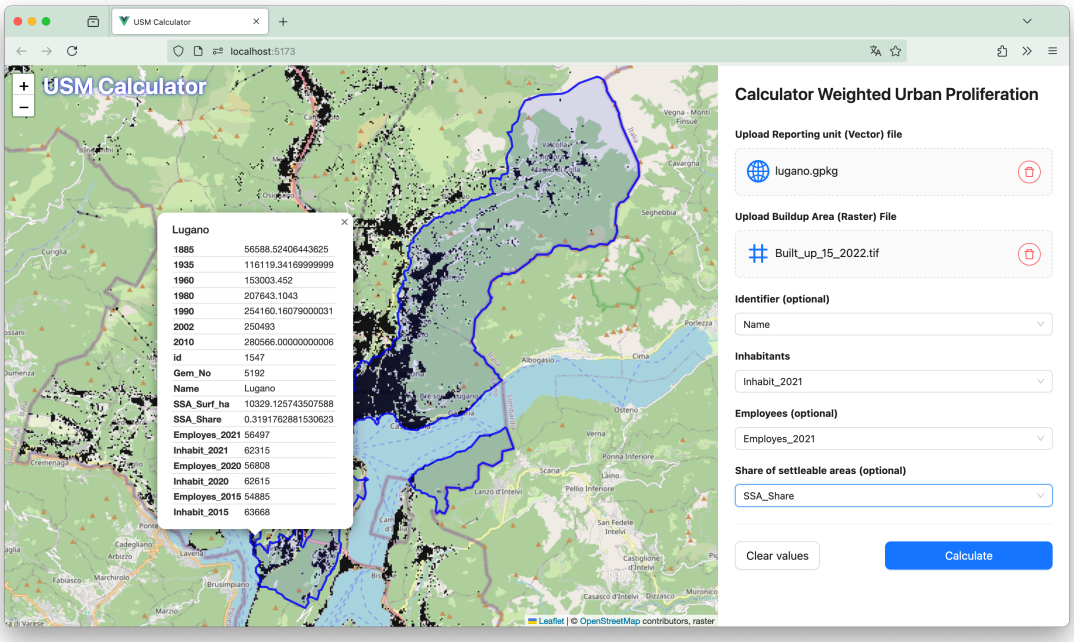

Abbildung 9.10: Der Benutzer kann in einem übersichtlichen Formular das Raster und die Vektordatei hochladen, die Input-Parameter für die Berechnung der Zersiedelungsmetriken eingeben und die Berechnung starten.

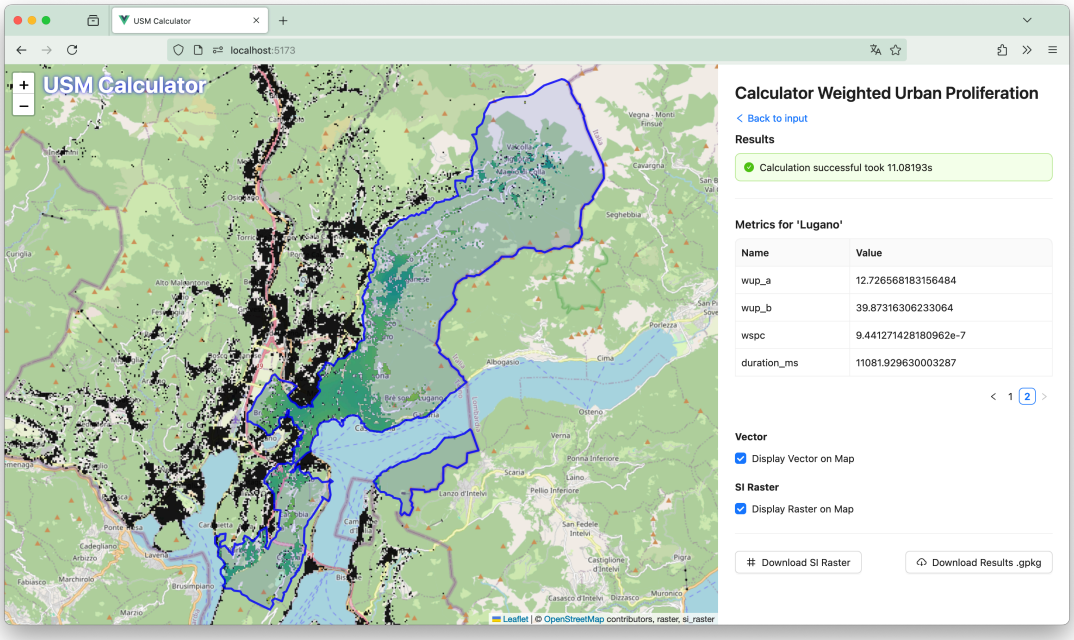

Abbildung 9.11: Das Resultat der Weighted Urban Proliferation (WUP) und weitere Metriken werden in einer Tabelle dargestellt. Die Dispersion lässt sich visuell auf der Karte angezeigen.

10 Implementation

10.1. Erweiterung

10.1.1. Webservice

Der Webservice wurde mittels Starlette-Framework in Python implementiert. Starlette ist ein schnelles Web-Framework, das auf ASGI (Asynchronous Server Gateway Interface) basiert.

Aufbau der RESTful API
Die API ist in verschiedene Endpunkte unterteilt, wobei jeder Endpunkt ein spezifischer Rechenschritt des gesamten Algorithmus darstellt.

Die Endpunkte sind, wie in Tabelle 10.1 dargestellt, definiert. Des Weiteren sind die Endpunkte gemäss OpenAPI Spezifikiation im Code-Repository dokumentiert. Dazu wurde die Klasse `SchemaGenerator` [56] des Moduls `starlette.schemas` verwendet, welche die automatische Generierung der OpenAPI Spezifikation ermöglicht. Mit Hilfe von Swagger UI, welches das Schema von Starlette lesen kann, hat man die Möglichkeit die API zu testen und mit verschiedenen Parametern aufzurufen. Die benötigten Parameter und die Struktur der Antwort sind in der Spezifikation definiert.

Für die Erklärung der einzelnen Metriken und deren Berechnungsschritte wird auf den Abschnitt 7.2 verwiesen. Die vollständige Dokumentation der API kann dem Open-Source Repository des Projekts (`https://gitlab.ost.ch/sa-urban-sprawl-metrics/usm-calculator`) entnommen werden.

Befehl (HTTP-Methode)	Endpoint	Beschreibung
POST	`/calculate`	Berechnet alle folgenden Metriken und gibt die Resultate als JSON-Objekt für jedes einzelne `Feature` des GeoJSON-Inputs (`FeatureCollection`) zurück.
POST	`/calculate/dis`	Berechnet den Degree of Urban Dispersion (DIS), die Streuung (Dispersion) von bebauten Gebieten anhand der Entfernungen zwischen zwei beliebigen Punkten innerhalb des bebauten Gebiets.
POST	`/calculate/lup`	Berechnet den Land Uptake per Person (LUP), die Flächeninanspruchnahme pro Person (Einwohner und Arbeitnehmer) der bebauten Fläche.
POST	`/calculate/pba`	Berechnet die Proportion of Built-up Area (PBA), den Anteil der bebauten Fläche an der gesamten Fläche (Untersuchungsgebiet).
POST	`/calculate/si`	Berechnet den Sprawl at Index (SI) für das gesamte Untersuchungsgebiet. Die Resultate werden in Form einer `tif`-Datei (TIFF) zurückgegeben.
POST	`/calculate/ts`	Berechnet den Total Sprawl (TS), der durschnittlichen Summe der gewichteten Distanzen zwischen allen Punkten innerhalb des bebauten Gebiets.
POST	`/calculate/ud`	Berechnet die Utilization Density (UD), welche die Anzahl Einwohner und Arbeitsplätze pro Quadratkilometer bebauter Fläche misst.
POST	`/calculate/up`	Berechnet den Wert der Urban Permeation (UP), welcher die Durchdringung einer Landschaft durch bebaute Gebiete misst.
POST	`/calculate/wdis`	Berechnet den gewichteten Wert der Dispersion von bebauten Gebieten (Weighted Degree of Urban Dispersion, WDIS).
POST	`/calculate/wspc`	Gibt den Wert des Anteils der Zersiedelung pro Person (Einwohner und Arbeitnehmer) im Untersuchungsgebiet (Weighted Sprawl per Capita, WSPC) zurück.
POST	`/calculate/wud`	Gibt die gewichtete Utilization Density (WUD) zurück (gewichteter Wert der Anzahl Einwohner und Arbeitsplätze pro Quadratkilometer bebauter Fläche).
POST	`/calculate/wup-a`	Berechnet den Weighted Urban Proliferation (WUP, deutsch: Zersiedelung Z) für das gesamte Untersuchungsgebiet.
POST	`/calculate/wup-b`	Berechnet den Weighted Urban Proliferation (WUP) für den besiedelbaren Teil des Untersuchungsgebiets.
POST	`/vector`	Ermöglicht das Hochladen einer Geopackage-Datei. Gibt die Datei als GeoJSON, reprojiziert im geodätischen Referenzsystem EPSG:4326 [57], zurück.
POST	`/vector/download`	Konvertiert das hochgeladene GeoJSON in ein Geopackage und streamt die Datei zurück.

Tabelle 10.1: API Endpunkte.

10.2. Berechnung

10.2.1. Grundlagen

Die Berechnung der Zersiedelung erfolgt in mehreren Schritten, wie in Kapitel 7 im Abschnitt "Berechnungsschritte" 7.2 beschrieben. Mit der API können die einzelnen Schritte der Berechnung separat aufgerufen werden.

Input-Daten und Parameter
Wenn die gesamte Berechnung durchgeführt wird, sind folgende Eingabeparameter notwendig:

- Ein Vektor Polygon mit Gebietsgrenzen (Untersuchungsgebiet)
- Ein Raster des bebauten Gebiets
- Anzahl Bewohner des Untersuchungsgebiets
- Anzahl Arbeitnehmer im Untersuchungsgebiet (optional)
- Wert für den SSA (Share of settleable Area), den Anteil der besiedelbaren Fläche am Untersuchungsgebiet (optional)

Für die Berechnung ist der Radius konstant auf den Wert von 2'000 Metern gesetzt, ansonsten müssten Gewichtungsfunktionen grundlegend angepasst werden.

Ausschneiden des Rasters an die Gebietsgrenzen
Das Raster des bebauten Gebiets wird an die Gebietsgrenzen des Untersuchungsgebiets angepasst. Dazu wird das Raster mit dem Vektor-Polygon des Untersuchungsgebiets geschnitten. Die Funktion `Warp` der GDAL-Library wurde verwendet, um das Raster zuzuschneiden. Als Zwischenschritt wird die Vektor-Datei (GeoJSON) in ein Shapefile konvertiert, um die Funktion `Warp` anwenden zu können. Das geschnittene Raster wird schlussendlich in eine Numpy-Matrix umgewandelt, um die Berechnung durchzuführen.

SI Berechnung
Für das bebaute Gebiet der Untersuchungsfläche wird eine Pixel-Matrix erstellt. Für jedes dieser Pixel, das als bebaut markiert ist, wird die Distanz zu allen anderen Pixeln, die bebaut und in einem bestimmten Radius sind, aufsummiert und gemittelt. Die Resultate werden in einem sogenannten SI-Raster gespeichert, das die Dispersion der bebauten Flächen im Untersuchungsgebiet darstellt. Anhand des SI-Rasters kann der Wert Dispersion (DIS), vgl. Kapitel 7.2, berechnet werden.

10.2.2. Sicherstellung der Datenqualität

Zur Überprüfung der Datenqualität der Berechnungsresultate wurden mehrere Tests durchgeführt. Als Referenz wurden die Resultate des bestehenden USM Calculator QGIS Plugins verwendet. Bereits in der Entwicklung des QGIS Plugins wurden die Resultate von einem Experten des ARE überprüft und validiert.

Zur Validierung der Resultate wurde als Referenz (vgl. Abbildung 10.1) die Gemeinde Rapperswil-Jona ausgewählt.

Das Vorgehen zur Validierung der Resultate war wie folgt:

1. Die Berechnung der Zersiedelungsmetriken für die Gemeinde Rapperswil-Jona wurde mit dem QGIS Plugin durchgeführt.
2. Dieselben Metriken wurden mit den selben Eingabeparametern mit dem Webservice berechnet.
3. Die Resultate wurden mit den Resultaten des QGIS Plugins verglichen.

Damit dieses Vorgehen auch für alle Entwickler und Benutzer der Software reproduzierbar ist, wurde ein automatisierter Test implementiert. Dieser Test vergleicht fixe Resultate, mit den Resultaten, die durch die Berechnung eines Testdatensatzes (Open Data) erzeugt wurden.

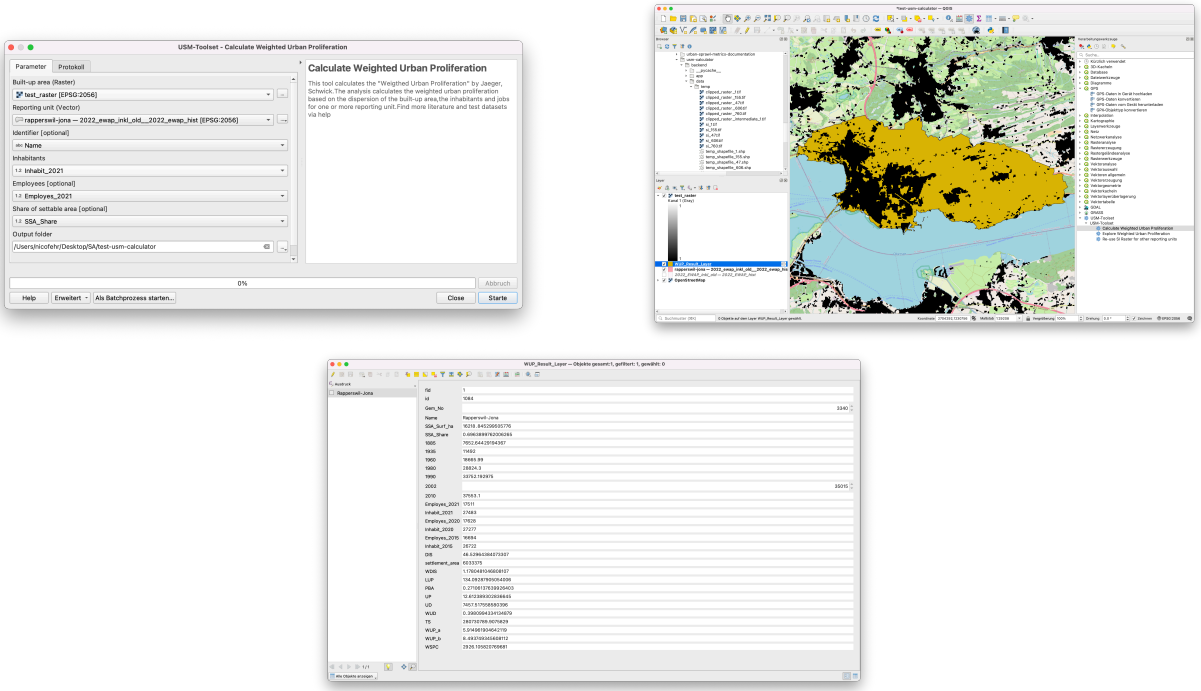

Abbildung 10.1: Berechnung der Zersiedelung mittels QGIS für die Gemeinde Rapperswil-Jona für die Referenzdaten.

10.2.3. Parallelisierung der Sprawl at Index (SI) Berechnung

Die Berechnung der SI für die Bestimmung der Dispersion, vergleiche Unterunterabschnitt 10.2.1, ist ein zeitaufwändiger Prozess, der durch die Parallelisierung der Berechnung beschleunigt werden kann. Für die Erweiterung wurden verschiedene Libraries zur Parallelisierung der Berechnung in einem empirischen Verfahren evaluiert.

Für die Evaluation wurde ein Python-Skript angelegt und die Berechnung des SI für verschiedene Gemeinden durchgeführt. Grundsätzlich wurden die Libraries Multiprocessing, Joblib und Numba getestet. Ein Github-Repository zeigt verschiedene Ansätze und zählt umsetzbare Varianten für Parallelisierungsaufgaben mit Python auf [58].

Nach der Abwägung der verschiedenen Technologien nach den Kriterien Performance, Einfachheit der Implementation und Wartbarkeit wurde die Technologie Numba ausgewählt. Numba ist eine Open-Source-Bibliothek, die es ermöglicht, den Python-Code zu beschleunigen, indem sie Funktionen in Maschinencode kompiliert. Mittels Just-In-Time (JIT) Kompilierung wird der Python-Code in Maschinencode übersetzt und ausgeführt. Die Performance der Numba-Bibliothek ist vergleichbar mit der von C-Code, jedoch ist die Implem einfacher und schneller als bei reinem C-Code, weil die Funktionen in Python geschrieben werden können.

Zu Beginn wurden zwei weitere Libraries, Dask [59] und Ray [60], getestet, jedoch war die Implementation im Vergleich zu Numba komplexer und es entsteht bei der Verwendung ein Overhead, weil die Tools für den Einsatz der Parallelisierung in verteilten Systeme ausgelegt sind.

Evaluation der Parallelisierung
Zur Evaluation der verwendeten Technologie zur Parallelisierung der SI-Berechnung wurde ein Benchmark durchgeführt. Als weitere Referenz wurde die Berechnung mit dem bestehenden QGIS Plugin durchgeführt.

Die Berechnungen wurden auf zwei verschiedenen Rechnern, die in der Tabelle 10.2 aufgelistet sind, durchgeführt, um die Performance auf unterschiedlichen Hardwarekonfigurationen und Chiparchitekturen zu testen. Zum einen wurde ein Apple MacBook Pro mit einem M3 Pro Prozessor und zum anderen ein HP Laptop mit einem Intel i7 Processor verwendet.

	Macbook Pro [61]	**HP Elite Dragonfly G3 Notebook** [62]
Veröffentlicht	2023	2022
Prozessor	Apple M3 Pro	12th Gen Intel Core i7-1255U, 1700 MHz, 10-Core
Architektur	ARM64	x86_64
Arbeitsspeicher	18 GB	16 GB
OS (Betriebssystem)	macOS Sonoma 14.3	Microsoft Windows 11 Pro - Version 10.0.22631

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

Für jede Methode der Parallelisierung wurde jeweils für vier verschiedene Schweizer Gemeinden (Tabelle 10.3) die Berechnung des SI mit den verschiedenen Technologien durchgeführt. Die Gemeinden, wie in Abbildung 10.2 dargestellt, wurden so gewählt, dass sie unterschiedliche Gemeindeflächen, bebaute Flächen und Strukturen aufweisen.

Folgende Gemeinden, wie in Tabelle 10.3 aufgeführt, wurden für den Benchmark ausgewählt. Die Daten zur Gemeindefläche entstammen der Applikation der Schweizer Gemeinden, die vom Bundesamt für Statistik (BFS) bereitgestellt wird [63]. Das Bundesamt für Statistik (BFS) verwendet für die Bestimmung von Flächen die ”Punktfläche” [64]. Ebenfalls die Einwohnerzahl der Geminden stellt das BFS im Dokument ”Ständige Wohnbevölkerung nach Staatsangehörigkeitskategorie, Geschlecht und Gemeinde, definitive Jahresergebnisse, 2023” zur Verfügung [65].

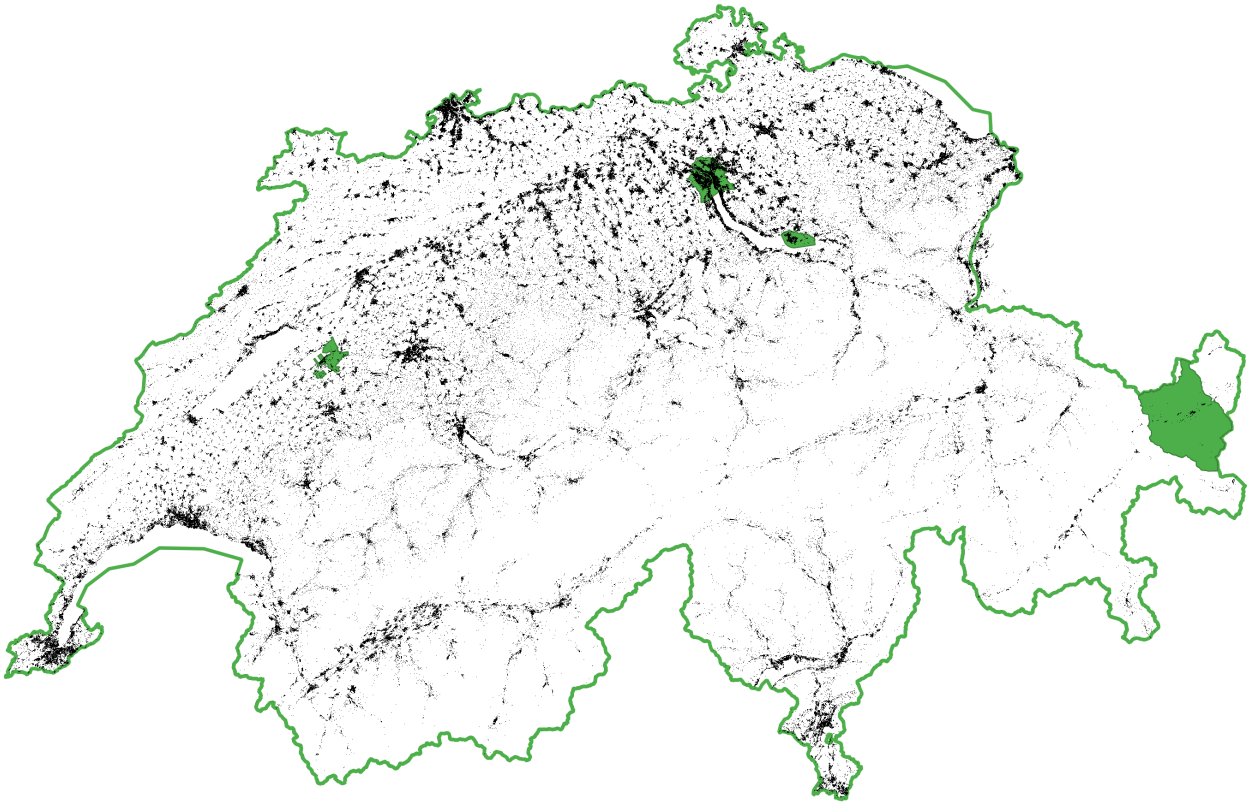

Abbildung 10.2: Auswahl der Gemeinden für den Benchmark. Die grüne Fläche repräsentiert die Gemeindefläche und die schwarze Fläche die bebaute Fläche. (Erstellt mit Geodaten des ARE und von ©swisstopo.)

Gemeinde (Stadt)	**Besonderheit**	**Weitere Informationen**		
		Fläche	*Einwohner*	*Einwohnerdichte* *(Einwohner pro km^2)*
Zürich	Die Stadt mit den meisten Einwohnern der Schweiz [65]	87.91km^2	433’989	4936.7
Murten	Eine Gemeinde, die aus keiner einzelnen Fläche besteht, sondern aus mehreren Teilen [66]	36.41km^2	9531	261.8
Scuol	Die flächenmässig grösste Gemeinde der Schweiz [67]	438.76km^2	4572	10.4
Rapperswil-Jona	Zweitgrösste Stadt im Kanton St. Gallen und Standort eines Campus der OST [68]	22.15km^2	28’640	1293

Tabelle 10.3: Auswahl der Gemeinden für den Benchmark. (Quellen: BFS, Gemeindeplan Murten, Gemeinde Scuol, Stadt Rapperswil-Jona)

Resultate der Benchmarks

In den nachfolgenden Tabellen 10.4, 10.5, 10.6 und 10.7 sind die Resultate der Benchmarks für die verschiedenen Gemeinden aufgeführt.

Zürich

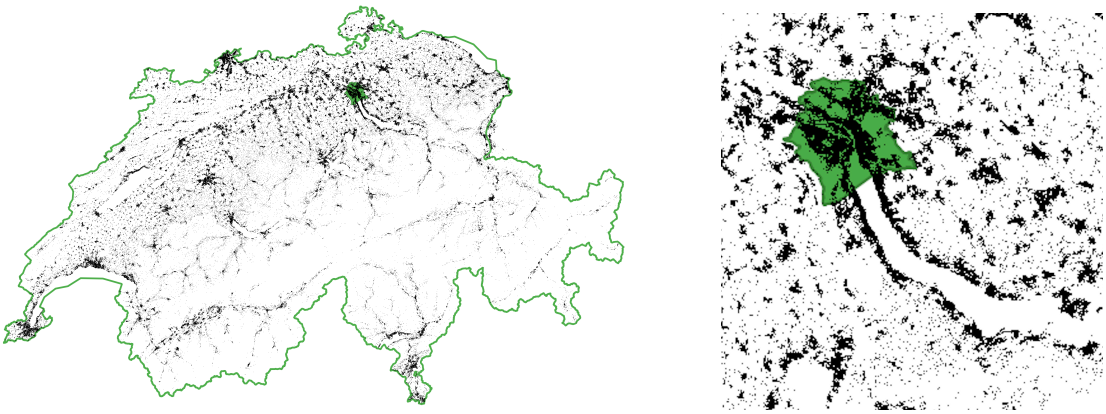

Abbildung 10.3: Fläche der Stadt Zürich in grün und bebaute Fläche in schwarz. (Erstellt mit Geodaten des ARE und von ©swisstopo.)

Methode		Dauer (ms)		System	Details und Bemerkungen
Multiprocessing	1.	2 761 097	(≈ 46 min 1 s)	M3 Pro	N_PROCESSES = 4
Joblib	1.	1 816 250	(≈ 30 min 16 s)		N_JOBS = Anzahl der Prozessoren
	2.	2 142 409	(≈ 35 min 42 s)		
	3.	1 998 784	(≈ 33 min 19 s)		
Numba	1.	21 516	(= 21.516 s)		inkl. JIT Kompilierung
	2.	15 310	(= 15.310 s)		
	3.	18 640	(= 18.640 s)		
QGIS Plugin	1.	636 190	(≈ 10 min 36 s)		
	2.	1 048 420	(≈ 17 min 28 s)		
	3.	711 530	(≈ 11 min 52 s)		
Numba	1.	121 383	(≈ 2 min 1 s)	Intel i7	inkl. JIT Kompilierung
	2.	120 087	(≈ 2 min)		
	3.	177 870	(≈ 2 min 58 s)		
QGIS Plugin	1.	918 120	(≈ 15 min 18 s)		
	2.	536 590	(≈ 8 min 57 s)		
	3.	612 000	(≈ 10 min 12 s)		

Tabelle 10.4: Resultate des Benchmarks für die Gemeinde Zürich.

Murten

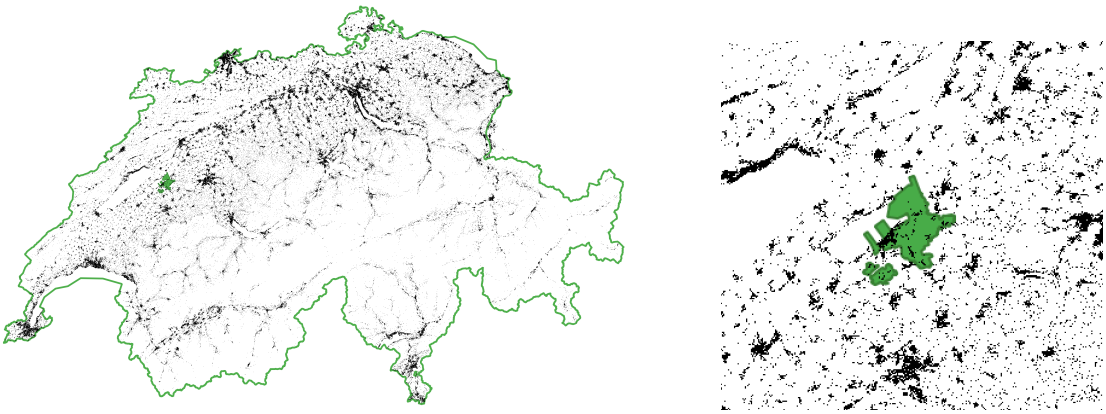

Abbildung 10.4: Fläche der Gemeinde Murten in grün und bebaute Fläche in schwarz. (Erstellt mit Geodaten des ARE und von ©swisstopo.)

Methode		Dauer (ms)		System	Details und Bemerkungen
Multiprocessing	1.	231 041	(≈ 3 min 51 s)	M3 Pro	N_PROCESSES = 4
	2.	229 586	(≈ 3 min 50 s)		
	3.	231 167	(≈ 3 min 51 s)		
Joblib	1.	123 307	(≈ 2 min 3 s)		N_JOBS = Anzahl der Prozessoren
	2.	123 765	(≈ 2 min 3 s)		
	3.	123 270	(≈ 2 min 3 s)		
Numba	1.	5341	(= 5.341 s)		inkl. JIT Kompilierung
	2.	3466	(= 3.466 s)		
	3.	3713	(= 3.713 s)		
QGIS Plugin	1.	16 580	(= 16.58 s)		
	2.	16 710	(= 16.71 s)		
	3.	16 980	(= 16.98 s)		
Joblib	1.	469 829	(≈ 7 min 50 s)	Intel i7	N_JOBS = Anzahl der Prozessoren
	2.	465 272	(≈ 7 min 45 s)		
	3.	487 909	(≈ 8 min 8 s)		
Numba	1.	29 613	(= 29.613 s)		inkl. JIT Kompilierung
	2.	22 414	(= 22.414 s)		
	3.	19 174	(= 19.174 s)		
QGIS Plugin	1.	15 290	(= 15.29 s)		
	2.	14 360	(= 14.36 s)		
	3.	14 610	(= 14.61 s)		

Tabelle 10.5: Resultate des Benchmarks für die Gemeinde Murten.

Scuol

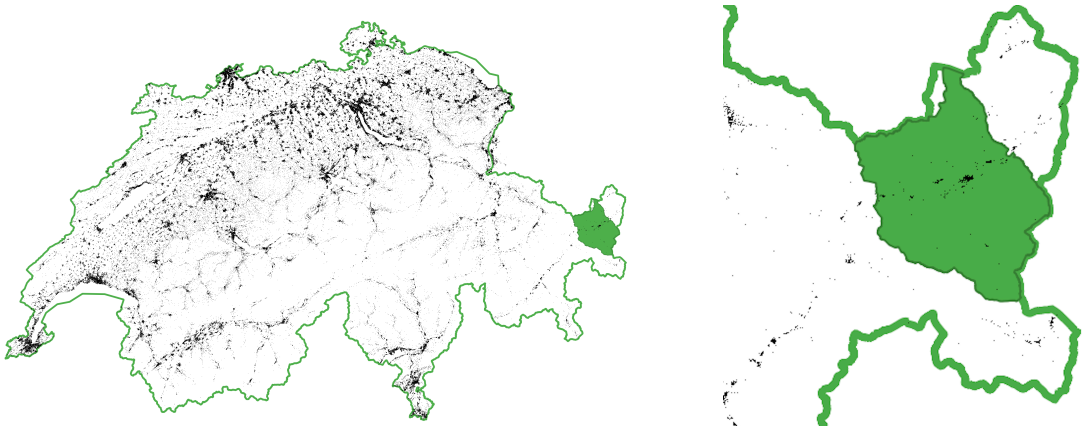

Abbildung 10.5: Fläche der Gemeinde Scuol in grün und bebaute Fläche in schwarz. (Erstellt mit Geodaten des ARE und von ©swisstopo.)

Methode		Dauer (ms)		System	Details und Bemerkungen
Multiprocessing	1.	297 931	(≈ 4 min 58 s)	M3 Pro	N_PROCESSES = 4
	2.	294 516	(≈ 4 min 54 s)		
	3.	294 731	(≈ 4 min 55 s)		
Joblib	1.	125 033	(≈ 2 min 5 s)		N_JOBS = Anzahl der Prozessoren
	2.	123 512	(≈ 2 min 3 s)		
	3.	123 629	(≈ 2 min 4 s)		
Numba	1.	6788	(= 6.788 s)		inkl. JIT Kompilierung
	2.	6066	(= 6.066 s)		
	3.	6121	(= 6.121 s)		
QGIS Plugin	1.	16 120	(= 16.12 s)		
	2.	15 940	(= 15.94 s)		
	3.	16 030	(= 16.03 s)		
Joblib	1.	403 360	(≈ 6 min 43 s)		N_JOBS = Anzahl der Prozessoren
	2.	407 242	(≈ 6 min 47 s)		
	3.	402 969	(≈ 6 min 43 s)		
Numba	1.	53 380	(= 53.380 s)		inkl. JIT Kompilierung
	2.	51 804	(= 51.804 s)		
	3.	44 117	(= 44.117 s)		
QGIS Plugin	1.	19 870	(= 19.87 s)		
	2.	20 280	(= 20.28 s)		
	3.	20 190	(= 20.19 s)		

Tabelle 10.6: Resultate des Benchmarks für die Gemeinde Scuol.

Rapperswil-Jona

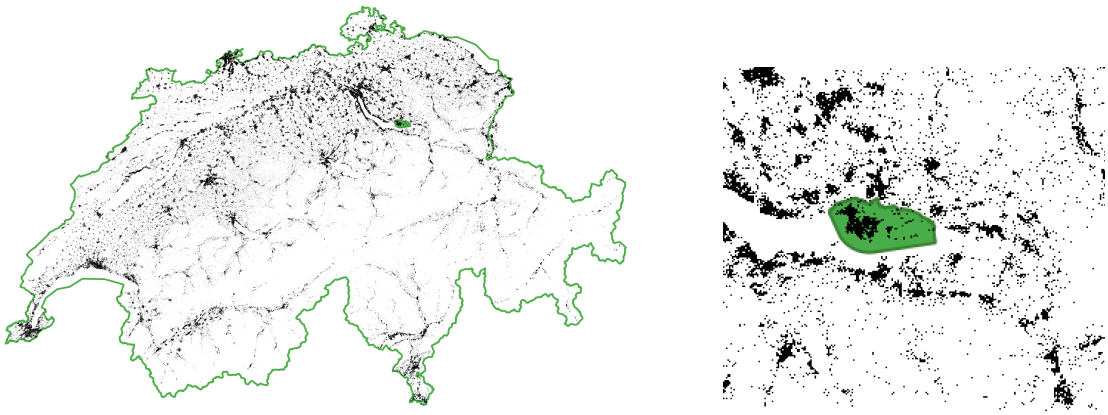

Abbildung 10.6: Fläche der Gemeinde Rapperswil-Jona in grün und bebaute Fläche in schwarz. (Erstellt mit Geodaten des ARE und von ©swisstopo.)

Methode		Dauer (ms)		System	Details und Bemerkungen
Multiprocessing	1.	349 507	(≈ 5 min 50 s)	M3 Pro	N_PROCESSES = 4
	2.	361 009	(≈ 6 min 1 s)		
	3.	361 085	(≈ 6 min 1 s)		
Joblib	1.	224 747	(≈ 3 min 45 s)		N_JOBS = Anzahl der Prozessoren
	2.	227 629	(≈ 3 min 47 s)		
	3.	226 329	(≈ 3 min 46 s)		
Numba	1.	3866	(= 3.866 s)		inkl. JIT Kompilierung
	2.	3130	(= 3.130 s)		
	3.	3100	(= 3.100 s)		
QGIS Plugin	1.	49 440	(= 49.44 s)		
	2.	49 880	(= 49.88 s)		
	3.	50 070	(= 50.07 s)		
Joblib	1.	868 350	(≈ 14 min 28 s)		N_JOBS = Anzahl der Prozessoren
	2.	872 072	(≈ 14 min 32 s)		
	3.	872 324	(≈ 14 min 32 s)		
Numba	1.	30 074	(= 30.074 s)		inkl. JIT Kompilierung
	2.	18 337	(= 18.337 s)		
	3.	22 743	(= 22.743 s)		
QGIS Plugin	1.	39 230	(= 39.23 s)		
	2.	39 260	(= 39.26 s)		
	3.	39 890	(= 39.89 s)		

Tabelle 10.7: Resultate des Benchmarks für die Gemeinde Rapperswil-Jona.

Fazit

Eine Zusammenfassung der Resultate des Benchmarks aller Varianten sind in Tabelle 10.8 aufgeführt. Einfachheitshalber wird jeweils die minimale, maximale und durchschnittliche Dauer der Berechnung durch jedes System für die vier Gemeinden angegeben.

Die wichtigste Erkenntnis aus dem Benchmark ist, dass der Einsatz von Numba die Berechnung um etwa den Faktor 6.1, im Vergleich zum QGIS Plugin, schneller durchführt. Auf leistungsschwächeren Rechnern, wie es bei dem i7 Prozessor der Fall ist, schneidet das QGIS Plugin teilweise besser ab als die Numba Variante. Vor allem bei der Berechnung von grösseren Untersuchungsgebieten mit kleiner bebauter Fläche, wie Scuol oder Murten, ist dies der Fall. Dies ist darauf zurückzuführen, dass verschiedene Zwischenschritte, wie die JIT-Kompilierung und Umwandlungen der Geodaten für GDAL notwendig sind, um die Berechnung durchzuführen und diese Schritte bei grösseren Flächen mehr Zeit in Anspruch nehmen können.

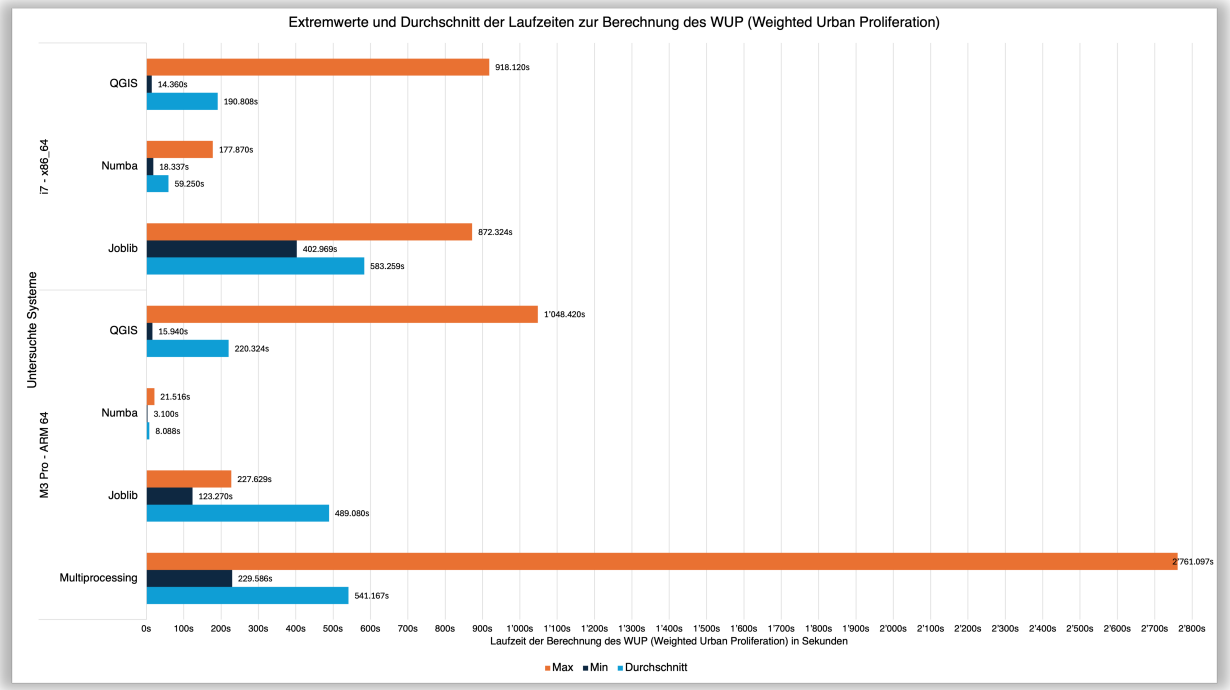

Abbildung 10.7: Benchmarkresultate nach System und Technologie.

System	Technologie	Min. Dauer (ms)	Max. Dauer (ms)	∅ Dauer (ms)
M3 Pro	Multiprocessing	229 586	2 761 097	541 167
	Joblib	123 270	227 629	489 080
	Numba	3100	21 516	8088.083
	QGIS Plugin	15 940	1 048 420	220 324.167
Intel i7	Joblib	402 969	872 324	583 258.556
	Numba	18 337	177 870	59 249.667
	QGIS Plugin	14 360	918 120	190 807.5

Tabelle 10.8: Extrem- und Durchschnittswerte der Benchmarks für die Berechnung der Zersiedelungsmetriken.

10.2.4. Probleme

Probleme traten bei der effektiven Umsetzung der Parallelisierung mit Numba keine auf. Es gab jedoch bei der Implementation der verschiedenen Varianten Schwierigkeiten mit der Multiprocessing Standard Library. Bei leistungsschwächeren Rechnern, wie dem i7 Rechner, war die Berechnung mit Multiprocessing unmöglich für stärker besiedelte Untersuchungsgebiete wie z.B. Zürich. Die Berechnung endete oft in einem ”Broken Pipe” Fehler, der auftritt, wenn die Kommunikation zwischen den Prozessen unterbrochen wird. Dies kann geschehen, wenn die Prozesse über keinen freien Arbeitsspeicher verfügen können.

10.3. Automatisierte Testverfahren

Im Backend-Teil des Projekts wurden automatisierte Tests implementiert, um vor allem die Korrektheit der Berechnungen sicherzustellen. Die Tests können mit der pytest [30] Library ausgeführt werden. Für jede Funktion, die eine Berechnung durchführt, wurde ein Unit-Test erstellt.

Als Testdaten für das Repository wurden offen verfügbare Geodaten verwendet. Von swisstopo [69] wurden Geodaten für die Gemeindeflächen der Schweiz bezogen. Die Daten sind im Geopackage-Format und enthalten Informationen über die Einwohnerzahl der Gemeinden. Über Opendata.swiss stehen Geodaten des Bundesamts für Zivilluftfahrt (BAZL) für die bebauten Flächen der Schweiz zur Verfügung [70].

Mit dieser Aufstellung können Open Source Mitwirkende die Tests zur Zersiedelungsberechnung selbst durchführen und die Resultate unabhängig überprüfen.

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. Unser Plan ist in Abbildung 11.1 dargestellt. In den grünen Boxen sind jeweils die Nummern der Meilensteine aus Abschnitt 11.3 eingetragen.

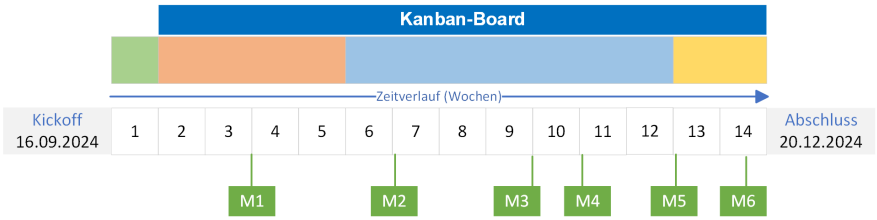

Abbildung 11.1: Projektplan nach RUP.

Abbildung 11.2 zeigt einen genaueren Zeitplan für die einzelnen Phasen des Projekts. Wir haben Tätigkeiten definiert und in einer Matrix dargestellt, wie viel Zeit jeweils pro Tätigkeit aufgewendet wird.

		Inception	Elaboration				Construction							Transition	
Tätigkeit	Anz. Wochen	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Project Management	12 - 14														
Project Planning	1 - 3														
Tooling / Infrastructure	2 - 4														
Requirements Engineering	4 - 6														
Software Architecture	3 - 5														
Admin (meetings, reviews, refinement)	12 - 14														
Zersiedelung für eine Gemeinde berechnen (Prototyp)	2 - 4														
Parallelisierung der SI Berechnung implementieren und evaluieren	3 - 5														
In Formular hochladen und eine Gemeinde berechnen	2 - 4														
Karte mit GeoJSON und Raster Anzeige-Funktion in Webapp integrieren	2 - 4														

Abbildung 11.2: Detailierterer Zeitplan der Tätigkeiten.

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 [71] organisiert und überwacht.

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	Detailierte 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	Initiales Projektsetup bereit	06.10.2024	Das Projektsetup ist bereit für die Entwicklung. Die Berechnung der Dispersion / Streuung der Siedlungsfläche ist verstanden.
M2	Prototyp	27.10.2024	Berechnung der Zersiedelung und technischer Durchstich.
M3	Parallelisierung SI Berechnung	17.11.2024	Parallelisierung der SI als Grundlage für DIS Berechnung für eine Gemeinde.
M4	Frontend mit Formular	24.11.2024	Ein Frontend für die Inputs der Berechnung inklusive Ausgabe der Resultate existiert.
M5	Frontend mit Kartenintegration	08.12.2024	Im Frontend ist eine Karte integriert, wobei ein Benutzer Ausschnitte der Karte auswählen kann, die als Input für die Berechnung dienen.
M6	Finale Abgabe	18.12.2024	Die Dokumentation und das Produkt ist gemäss dem definierten Umfang fertiggestellt und bereit zur Abgabe.

Tabelle 11.2: Meilensteine des Projekts.

11.4. Rollen

In unserem Projekt gibt es folgende Rollen:

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

Tabelle 11.3: Rollen im Projekt und deren Interesse.

Der ”Product Owner”, auch PO genannt, ist verantwortlich für die Einhaltung der Deadlines und er hat die Aufgabe, die Übersicht des Projekts zu behalten. Alle Teammitglieder sind für die Entwicklung und Umsetzung des Projekts zuständig und unterstützen den PO bei der Entscheidungsfindung und administrativen Aufgaben. 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.

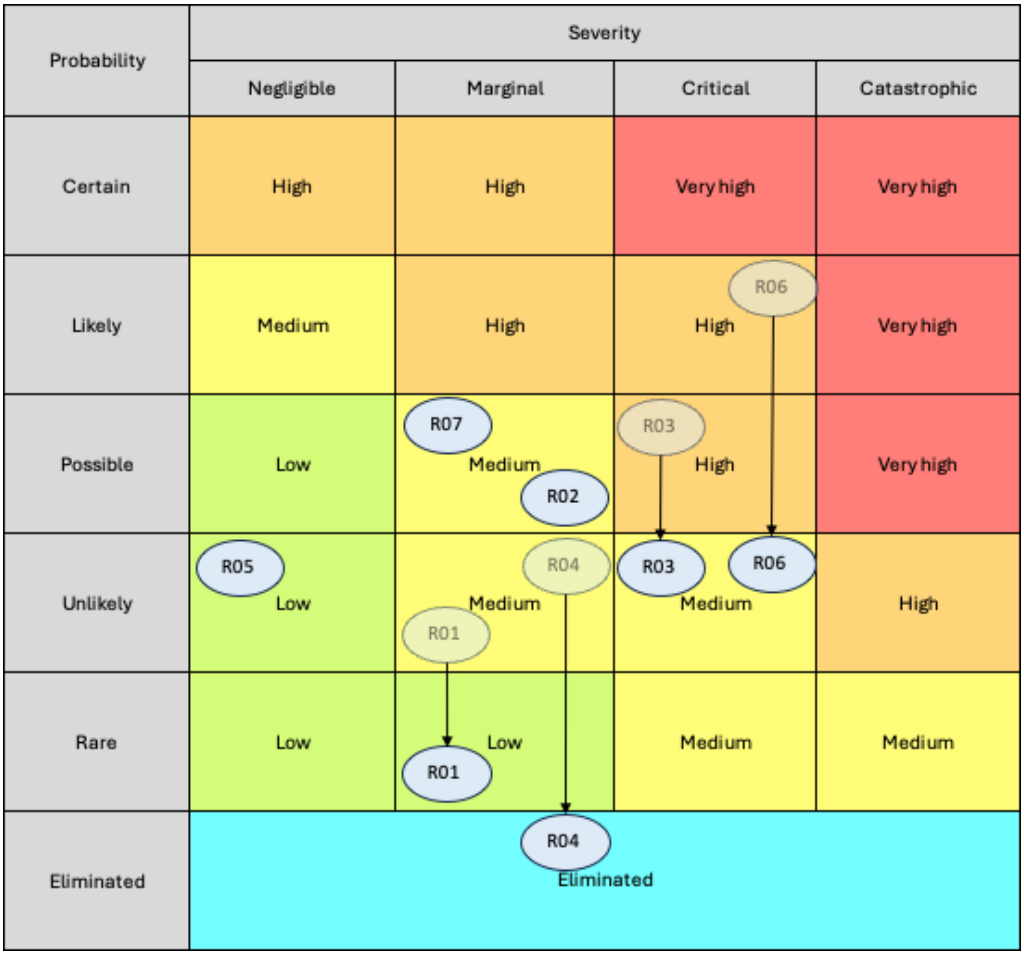

Abbildung 11.3: Risikomatrix basierend auf der Risikoliste.

11.5.2. Risikoliste

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

ID	Risiko	Wahrscheinlichkeit	Schwere	Risikolevel	Kosten / Aufwand	Strategie
R01	Berechnung der Zersiedelung ist unklar.	Rare	Marginal	Low	Low - 5h	Bei Fragen zur Berechnung Betreuer und Industriepartner informieren und stärker miteinbeziehen. Fragerunden
R02	Parallelisierung der Berechnungen ist technisch nicht oder nur schwer umsetzbar.	Possible	Marginal	Medium	Medium - 10h	Anwenden von anderen Programmiersprachen und Technologien, deren Parallelisierung effizient funktioniert. Eruierung weiterer Libraries.
R03	Testdaten für den Input sind ungenügend, unvollständig.	Unlikely	Critical	Medium	Medium - 8h	Weitere Testdaten werden beim Auftraggeber eingeholt. Die Daten werden mit bestehenden Tools (z.B. USM Plugin) überprüft.
R04	Karte kann nicht in Frontend eingebunden werden.	Eliminated	-	Eliminated	Medium - 16h	Mögliche Libraries für die Integration der Karte werden untersucht und ausgewertet. Auf diesem Web finden sich passende Kandidaten und man kann Alternativen finden.
R05	Komplexität der Frontendtechnologien wird unterschätzt.	Unlikely	Negligible	Low	Low - 4h	Die Teammitglieder informieren sich ausreichend über die Frontendtechnologien und eruieren Möglichkeiten der Umsetzung. Falls technische Unmöglichkeiten bestehen wird das Frontend vereinfacht.
R06	Der Datenschutz kann nicht gewährleistet werden.	Unlikely	Critical	Medium	High - 32h	Das Entwicklerteam muss mit dem Auftraggeber die Beschaffenheit und die Art der schützenswerten Daten genau untersuchen und kategorisieren. Die Datenbank(en) und Schnittstellen werden entsprechend Best-Practices geschützt.
R07	Teammitglieder, die Quereinsteiger sind, brauchen länger an gewissen Aufgaben	Possible	Marginal	Medium	Medium - 24h	Software Engineering Practices wie Extreme Programming und Pair-Programming werden angewendet. Dies erhöht den Lerneffekt stark (Learning by Doing). Auch Erfahrene Entwickler können durch die Weitergabe der Erfahrung dazulernen.

Tabelle 11.4: Phasen des Projekts nach RUP.

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

Das Änderungsprotokoll in der unten stehenden Tabelle 11.5 wird verwendet, um Änderungen in der Risikoeinschätzung zu dokumentieren.

Datum	Risiko ID	Änderung	Grund	Neues Risikolevel
27.10.2024	R01	Reduzierte Wahrscheinlichkeit von ”Unlikely” zu ”Rare”	Die Berechnungen wurden, bei der Anwendung in der Sofware, klarer im Laufe des Projekts.	Low
18.11.2024	R03	Reduzierte Wahrscheinlichkeit von ”Possible” zu ”Unlikely”	Genügend Daten wurden für zuverlässige Tests gesammelt. Die Berechnungen lassen sich mit den verfügbaren Daten verifizieren.	Medium
31.11.2024	R06	Reduzierte Wahrscheinlichkeit von ”Likely” zu ”Unlikely”	Das Risiko wurde verringert, indem keine persistente Speicherung von Daten vorgenommen wird und alle temporär angelegten Daten nach der Berechnung gelöscht werden. Während der Berechnung werden noch immer temporäre Dateien angelegt.	Medium
08.12.2024	R04	Risiko konnte vollständig eliminiert werden	Die Karte konnte erfolgreich in das Frontend eingebunden werden.	Eliminated

Tabelle 11.5: Änderungsprotokoll für Risiken während der Projektlaufzeit.

11.6. Qualitätssicherung

Aktivitäten zur Qualitätssicherung werden in verschiedenen Bereichen durchgeführt. Die nachfolgend beschriebenen Massnahmen sind sehr verbreitet in der Softwareentwicklung und stellen sicher, dass die Qualität des Codes und des Produkts gewährleistet ist. Dies ist vor allem wichtig, da das Produkt in der Praxis eingesetzt wird und die Berechnungen korrekt sein müssen. Nebenbei steht der Quellcode offen zur Verfügung und kann von beliebigen Entwicklern eingesehen und weiterentwickelt werden. Somit ist es wichtig, dass die Qualität des Codes hoch ist und die Entwickler sich an die Best Practices halten.

11.6.1. Code-Richtlinien

Das Frontend wird in Typescript und das Backend in Python verfasst. Im Folgenden werden die wesentlichen Punkte der Code-Richtlinien dargelegt:

- 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) [13] - Eine Funktion macht nur eine Sache
- Selbsterklärende Fehlermeldungen werden verwendet
- Typescript Types werden verwendet, um Typsicherheit zu gewährleisten
- ”Keep it simple, stupid” (KISS) [72] - Einfachste Lösung ist die beste Lösung

11.6.2. Gitlab CI/CD

Zur bestmöglichen Qualitätssicherung des Codes wird von der Gitlab CI/CD Pipeline Gebrauch gemacht. Als Teil der Versionskontrolle, können Aktionen wie Tests, Linting und Deployment automatisiert werden. Über die Datei `.gitlab-ci.yml` wird die Pipeline konfiguriert und definiert, welche Schritte durchgeführt werden sollen. In diesem Projekt werden für Python drei statische Code-Analyse-Tools verwendet. Für Typscript wird ein Type-Checker und ein Linter verwendet. Diese Tools helfen frühzeitig Unschönheiten und Fehler im Code zu erkennen.

Nach der Linting-Stage wird die Test-Stage ausgeführt.

flake8
Mittels Flake8 [42] werden Pogrammierfehler und Code Smells in Python-Code erkannt. Dabei wird auf die Einhaltung der PEP8-Richtlinien [43] geachtet. Die Konfiguration in der Pipeline hat die maximale Zeilenlänge auf 140 Zeichen gesetzt.

pylint
Pylint [44] überprüft ebenfalls auf Unstimmigkeiten im Code. Ausserdem schlägt das Tool Verbesserungen für das Refactoring vor. Auch hier wurde eine Konfiguration in der Pipeline hinterlegt, die die maximale Zeilenlänge auf 140 Zeichen, die Anzahl der Argumente auf 10 und die Anzahl der Statements pro Funktion auf 100 beschränkt.

mypy
Mypy [45] ist ein Type-Checker für Python. Weil Python eine dynamisch typisierte Sprache ist, werden Typen im Sprachgebrauch oft nicht explizit definiert. Diese Lücke schliesst Mypy in Kombination mit der Python-Standartbibliothek `typing` [73], indem es Typen überprüft und Fehler meldet, wenn Typen nicht korrekt verwendet werden. Mypy ist in der Pipeline so konfiguriert, dass die Typenüberprüfung des Codes der Tests ignoriert wird.

pytest
Pytest [30] ist ein Framework für Python, das das Testen von Codes vereinfacht. Es wird in der Pipeline verwendet, um die Unit-Tests für Python auszuführen.

vue-tsc
Vue-tsc [74] ist ein TypeScript-Compiler für Vue.js. In der Pipeline wird dieser verwendet, um die Typen des Frontend-Codes, vor dem Build des Frontends, zu überprüfen. Explizite Typen in TypeScript helfen, Fehler frühzeitig zu erkennen und die Code-Qualität zu erhöhen.

eslint
Mittels ESLint [46] werden Programmierfehler und Code Smells in TypeScript-Code erkannt. Die Konfiguration des Linters wurde um Vue.js- und Typescript-Regeln erweitert.

11.6.3. Git Feature Branch Workflow

Für das Git-Repository wird der Feature Branch Workflow angewendet. Das bedeutet, dass für jede neue Funktionalität oder Bugfix ein neuer separater Branch erstellt wird. Ist die Funktionalität fertig entwickelt, wird ein Merge Request erstellt und der Code wird von einem anderen Entwickler überprüft. Nach dem Review-Prozess wird der Branch in den ”main” Branch ”gemerged”. Der ”main” Branch ist geschützt und kann nur durch den Merge Request-Prozess verändert werden. Somit wird sichergestellt, dass der Code des Hauptzweigs immer funktionsfähig und jederzeit auslieferbar ist.

11.6.4. Testkonzept

Die Qualitätssicherung des Projekts wird durch verschiedene Tests der Software sichergestellt. Die Teststrategie soll so aufgebaut sein, dass Qualität, Performance, Funktionalität und Kompatibilität des Produkts stets gewährleistet sind.

Unit-Tests
Die Unit-Tests sind unter den automatisierten Tests in Kapitel 10 im Abschnitt 10.3 beschrieben. Die Tests verifizieren die Funktionalität der einzelnen Komponenten und Funktionen, in diesem Fall die einzelnen Berechnungsschritte des ”USM-Calculators”.

System-Tests
Ein System-Test soll die korrekte Funktionalität des gesamten Systems als Ganzes, überprüfen. Angestrebt wird dabei eine Umgebung, die der Produktionsumgebung möglichst nahe kommt. Ein Testprotokoll wurde erstellt, um die System-Tests zu dokumentieren. Nach jeder Umsetzung einer neuen Funktionalität oder eines Bugfixes werden manuelle System-Tests durchgeführt.

Vor der Durchführung der System-Tests wird ein Smoke-Test durchgeführt, um sicherzustellen, dass die grundlegenden Funktionen des Systems funktionieren. Konkret wird dabei überprüft, ob die Webapplikation startet und ob die Startseite korrekt angzeigt wird und die API antwortet.

Die API konnte während der Entwicklungsphase dank der Integration von Swagger [75] manuell getestet werden. Swagger ist ein Tool, das die Dokumentation von RESTful APIs vereinfacht und die Interaktion mit der API erleichtert.

11.6.5. Testprotokoll

Das Testprotokoll deckt die finale Durchführung der System-Tests ab. Für jedes Feature im Frontend wurde ein Testfall definiert und mit dem Resultat dokumentiert.

ID	Beschreibung	Ergebnis	Status
T01	Die Webapplikation startet.	Erfolgreich	✓
T02	Die Startseite wird korrekt angezeigt.	Erfolgreich	✓
T03	Ein Geopackage kann hochgeladen werden und die Fläche wird mit allen Informationen auf der Karte korrekt angezeigt.	Erfolgreich	✓
T04	Ein Geojson kann hochgeladen werden und die Fläche wird mit allen Informationen auf der Karte korrekt dargestellt.	Erfolgreich	✓
T05	Die Rasterdatei im TIFF-Format kann ausgewählt werden und wird korrekt in der Karte dargestellt.	Erfolgreich	✓
T06	Im ”Identifier”-Dropdown sind alle Parameter auswählbar.	Erfolgreich	✓
T07	In allen Dropdowns, ausser ”Identifier”, sind alle Parameter mit Zahlenwerten auswählbar.	Erfolgreich	✓
T08	Die Berechnung funktioniert korrekt, wenn alle Input-Werte gesetzt und korrekt sind.	Erfolgreich	✓
T09	Die Berechnung funktioniert korrekt, wenn nur ”Inhabitants” korrekt gesetzt ist.	Erfolgreich	✓
T10	Die Berechnung schlägt fehl und zeigt eine Fehlermeldung, wenn eine Zahl < 1 für ”Inhabitants” gesetzt wird.	Erfolgreich	✓
T11	Die Berechnung der Zersiedelung schlägt fehl und zeigt eine Fehlermeldung, wenn das Raster sich ausserhalb des Untersuchungsgebiets (Vektor) befindet.	Erfolgreich	✓
T12	Die Berechnung schlägt fehl und zeigt eine Fehlermeldung, wenn der Wert in den ”Inhabitants”- und ”Employees”-Parametern 0 beträgt.	Erfolgreich	✓
T13	Die temporär angelegten Dateien werden nach der Berechnung gelöscht.	Erfolgreich	✓
T14	Das Herunterladen des Geopackages funktioniert und die Resultate sind vorhanden.	Erfolgreich	✓
T15	Das Herunterladen des SI-Rasters funktioniert fehlerlos und die Datei kann in QGIS geöffnet werden.	Erfolgreich	✓

Tabelle 11.6: Testprotokoll für System-Tests.

12

Projektmonitoring

12.1. Soll-Ist-Zeitvergleich

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

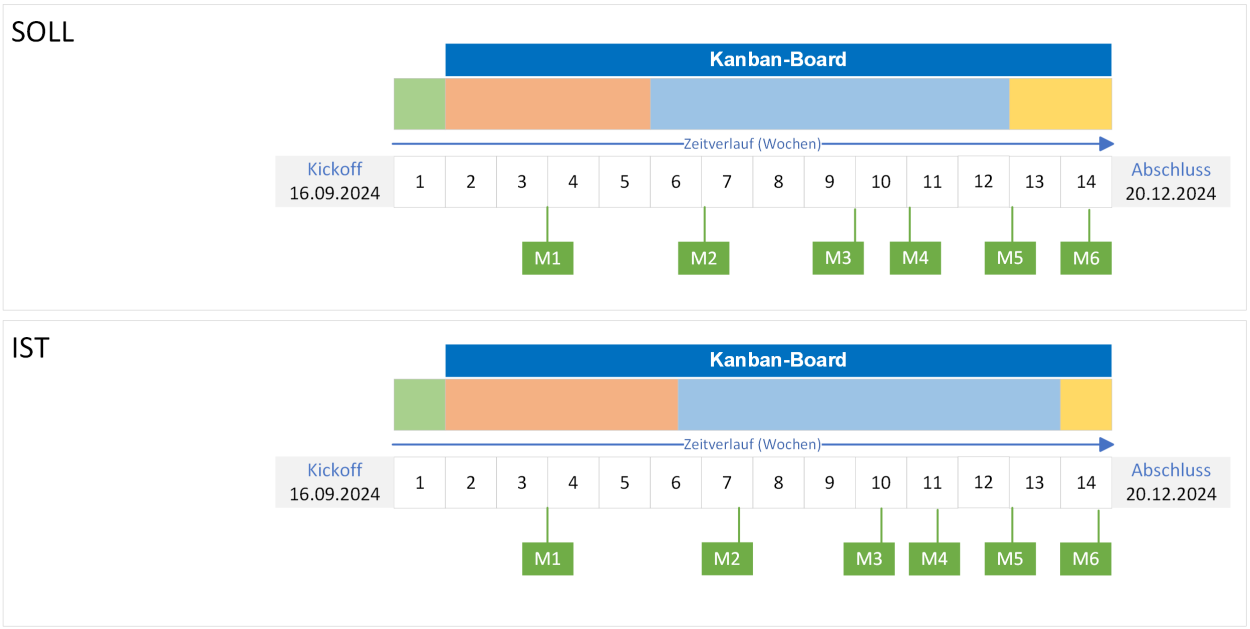

Abbildung 12.1: Vergleich Soll und Ist der Projektphasen nach RUP

Die Elaboration Phase hat sich um eine halbe Woche verzögert, weil noch nicht alle Nicht-Funktionalen Anforderungen definiert waren. Der Meilenstein für die Erstellung des Prototypen, wurde um knapp eine Woche überschritten, weil die Verknüpfung des Backends mit dem Frontend länger dauerte. Ausserdem verlängerte sich die Construction Phase um eine Woche, weil das Frontend mit dem Testing und Bugfixing mehr Zeit in Anspruch genommen hat.

Total wurde 486.5 Stunden für das gesamte Projekt aufgewendet. Davon entfallen 326 Stunden auf die Entwicklung und 160.5 Stunden auf die Dokumentation.

12.2. Zeitaufwand pro Person

Pro Person werden pro ECTS-Punkt 30h Aufwand erwartet. Daraus ergibt sich ein Soll-Aufwand von 240 Stunden pro Person. Der Totalauwand in Stunden pro Person beläuft sich auf somit $8 \cdot 30h = 240h$ Stunden.

Name	Zeitaufwand
Kyra Maag	224.1h
Nico Fehr	262.4h

Tabelle 12.1: Zeitaufwand pro Person, Stand 19.12.2024

12.3. Codestatistik

Im Rahmen dieser Arbeit wurden **2904** Zeilen Code geschrieben. Davon wurden **1951** Zeilen in Python und **762** Zeilen in TypeScript inklusive Vue.js geschrieben.

Für die Entwicklung der Erweiterung wurden **61** CI/CD Jobs mit Gitlab pipelines erstellt, die Durchschnittlich **3** Minuten dauerten.

In SonarQube [76] wurde der Code analysiert und die Qualität des Codes überwacht, wobei die Wartbarkeit mit einem **A** bewertet wurde. 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 einfachste Art der Installation geschieht mit dem Einsatz von Docker. Dazu muss auf dem Entwicklungscomputer Docker Desktop oder Docker auf einem Server installiert sein.

Mit einem einfachen Befehl können Back- und Frontend mithilfe von Docker-Compose gestartet werden. Zu diesem Zweck befindet sich im Root-Verzeichnis des Projekts die Datei `docker-compose.yml`. Mit dem untenstehenden Befehl wird die Anwendung aus dem Verzeichnis gestartet, in dem sich die Datei befindet.

```
1    docker-compose up
```

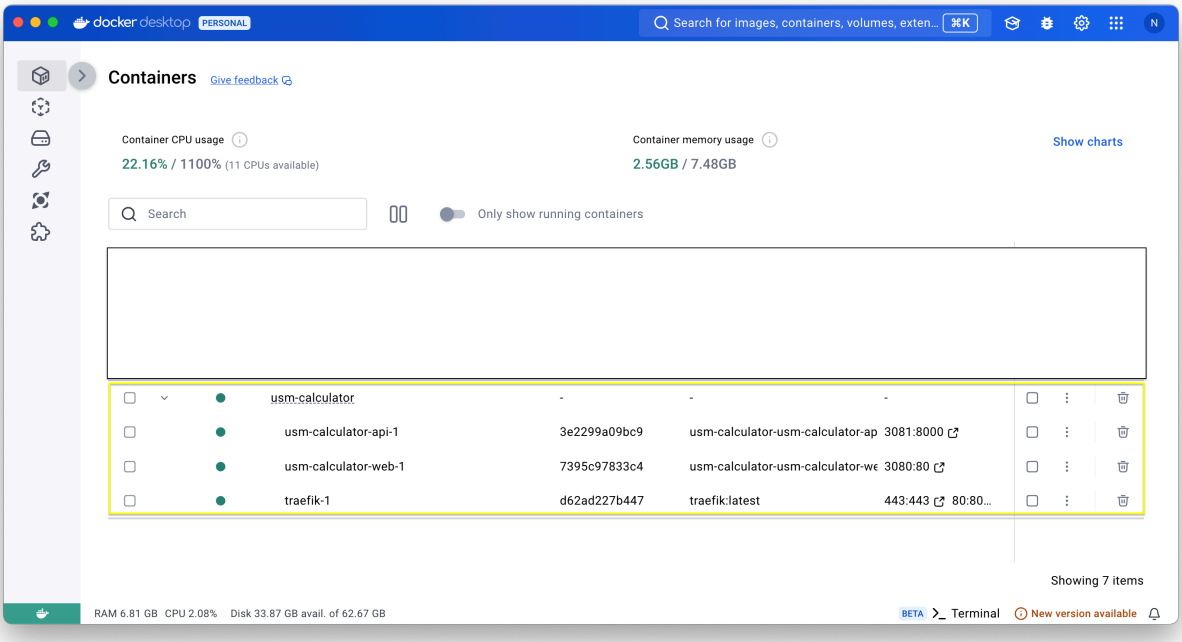

Abbildung 13.1: Screenshot aus Docker Desktop: Docker-Compose startet die Anwendung bestehend aus Front-, Backend und Loadbalancer.

Weitere Befehle für Docker sind der README-Datei des Projekts zu entnehmen.

Wie man die Anwendung ohne Docker startet, ist im README des Projekts beschrieben. Um dies zu tun, müssen sämtliche Abhängigkeiten des Projekts manuell installiert werden.

13.2. Bedienungsanleitung

13.2.1. Startseite

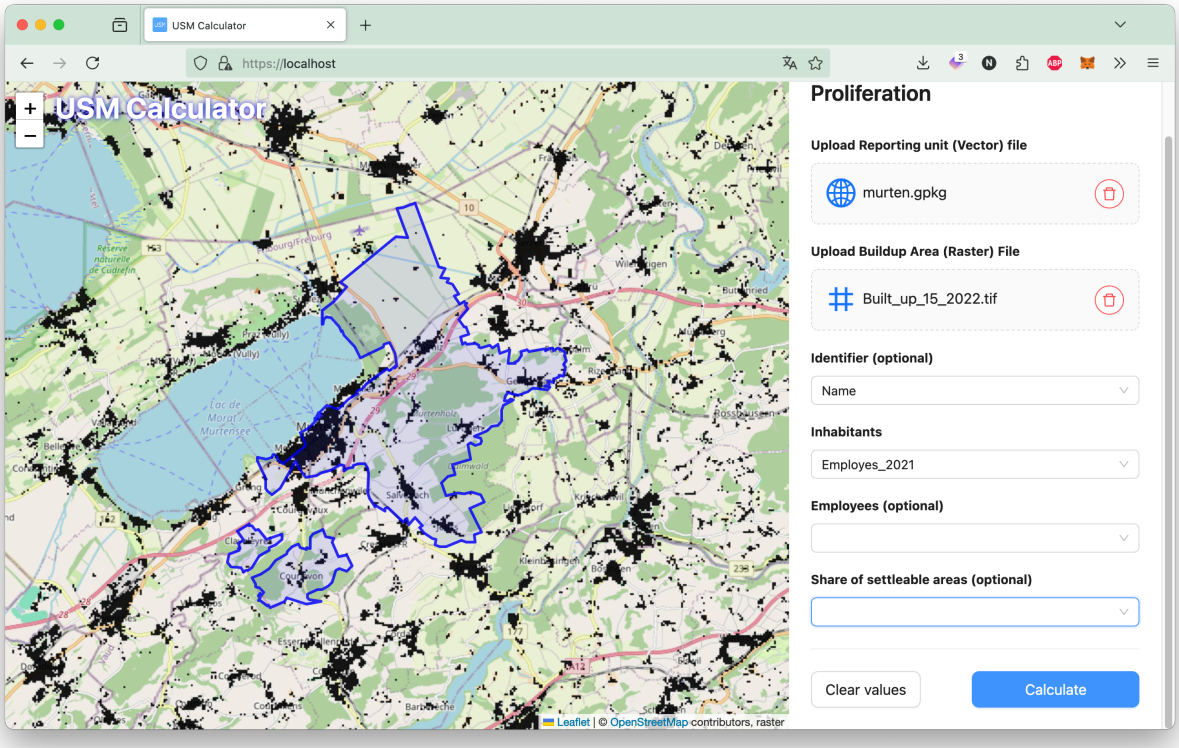

Abbildung 13.2: USM-Calculator Startseite mit Eingabefeldern für die Berechnung der WUP.

Reporting Unit (Vector file): Das Untersuchungsgebiet als GeoJSON- oder Geopackage-Datei.

Buildup Area (Raster file): Das Siedlungsgebiet als TIFF-Datei.

Aus den Parametern der ”Reporting Unit” lassen sich die nachfolgenden Informationen in einem Dropdown auswählen.

Identifier (optional): Ein eindeutiger Name für das Untersuchungsgebiet. Wird kein Name angegeben, werden die einzelnen Features der Vektor-Datei durchnummeriert.

Inhabitants: Die Anzahl der Einwohner im Untersuchungsgebiet.

Employees (optional): Die Anzahl der Beschäftigten im Untersuchungsgebiet.

Share of settleable areas (optional): Der Anteil der besiedelbaren Fläche im Untersuchungsgebiet.

13.2.2. Resultat

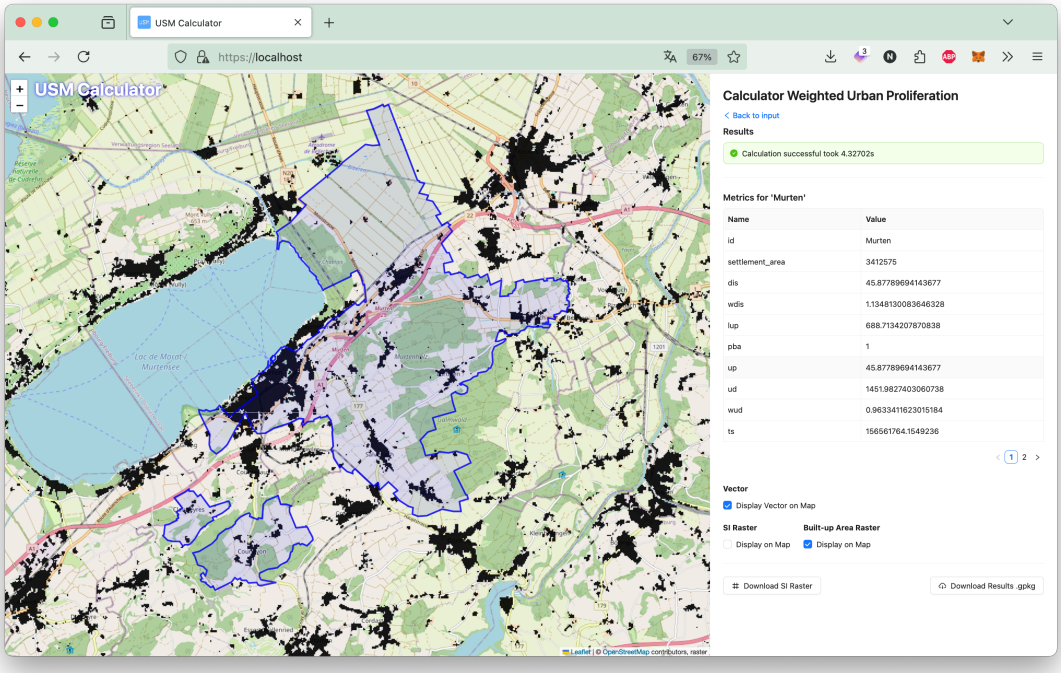

Abbildung 13.3: USM-Calculator Resultat

Mit den beiden Buttons am unteren rechten Rand der Seite kann das Resultat als Geopackage-Datei heruntergeladen werden. Ausserdem lässt sich das SI-Raster als TIFF-Datei herunterladen.

13.2.3. Fehlermeldungen

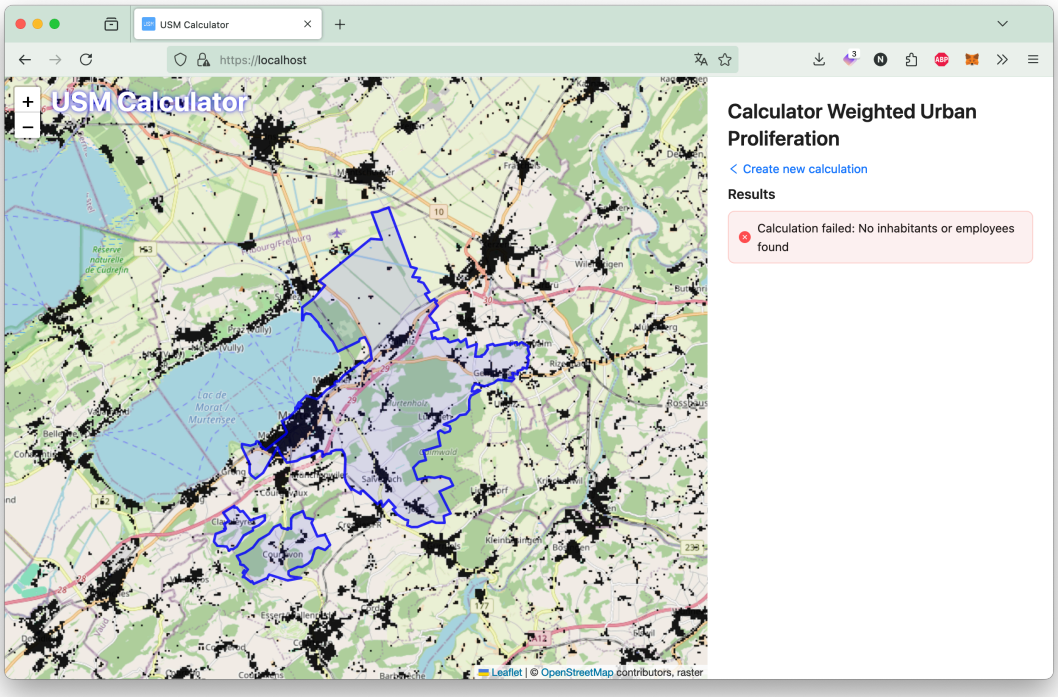

Abbildung 13.4: Fehlermeldung im Frontend.

13.2.4. API-Dokumentation

Die Swagger API-Dokumentation lässt sich über `/api/docs` aufrufen.

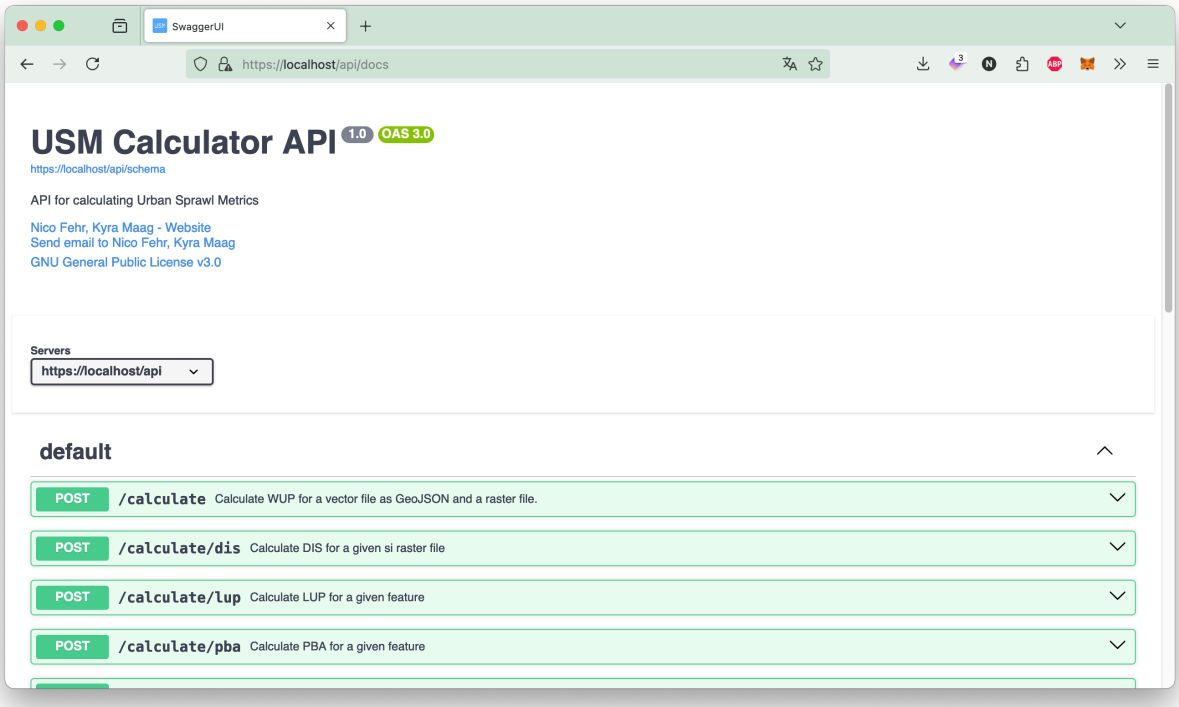

Abbildung 13.5: API Dokumentation mit Swagger.

13.3. Weiterentwicklung mit VSCode

Es wird empfohlen, die Weiterentwicklung des Projekts mit Visual Studio Code durchzuführen. Eine entsprechene Anleitung zur Einrichtung der IDE ist im Repository des Projekts zu finden. Durch die Anwendung der VS Code Development Container Extension wird eine einheitliche Entwicklungsumgebung für alle Entwickler gewährleistet [77].

A

Persönliche Berichte

A.1. Kyra Maag

Die Durchführung des Projekts war für mich sehr lehrreich und interessant. Zu Beginn des Projekts war ich aufgeschlossen, ein spannendes und anspruchsvolles Frontend zu entwickeln. Das Thema der Zersiedelung und wie man diese misst, war mir zu Beginn unbekannt und ich musste mich zuerst einlesen. Zusammen konnten wir den bestehenden Algorithmus und die Funktionalität schrittweise verstehen. Mit QGIS konnte ich bereits in einem Praktikum Erfahrungen sammeln und konnte dabei Nico bei der Erarbeitung von Testdaten und der Verifizierung unterstützen.

Das Ausarbeiten des Frontends, vor allem das Design zu entwerfen, machte mir Spass. Eine Herausforderung war für mich eher die Verknüpfung zum Backend, weil die Daten zum Teil speziell "verpackt" werden müssen (Formdata). Dazu gehören unter anderem die Funktionalitäten für den Upload und Download von beliebigen Dateiformaten, in unserem Fall speziell Geopackages, GeoJSON und TIFF.

Ich bin sehr zufrieden mit dem Ergebnis dieses Projektes. Trotz ein paar Schwierigkeiten mit der Berechnung hat die Parallelisierung und somit die Beschleunigung der Berechnung funktioniert. Der Benutzer kann nun in einem intuitiven Frontend von dieser schnelleren Berechnung profitieren.

Die Zusammenarbeit mit Nico war sehr angenehm. Wir haben schon viele Projekte gemeinsam gemacht, somit wussten wir, was die Stärken und Schwächen des anderen sind und konnten so gut die Arbeit danach aufteilen. Die einzige Herausforderung war, da wir beide Teilzeitstudenten mit Nebenanstellung sind und auch unterschiedliche Stundenpläne hatten, gemeinsame Zeit zu finden, um am Projekt zu arbeiten.

A.2. Nico Fehr

Die Thematik und die Umsetzung des Projekts fand ich sehr interessant und ansprechend. Zu Beginn war ich mir unsicher, ob der Algorithmus für uns leicht verständlich sein könnte. Persönlich fand ich das Einlesen in die Thematik der Zersiedelung sehr interessant. Die Methoden mögen einem sehr technisch und uninteressant erscheinen, aber sie betreffen im Endeffekt alle Personen, die in wachsenden Gemeinden leben. Eine Arbeit zu implementieren, die auch in der Praxis zum Einsatz kommen könnte, war für mich wichtig.

Für mich war es primär eine Herausforderung, die Programmiersprache Python im Zusammenhang mit Geodaten einzusetzen. Ich hatte bereits mit Numpy und mehrdimensionalen Arrays gearbeitet, aber für die Arbeit mit den verschiedenen Dateiformaten und geodätischen Referenzsystemen musste ich mich sehr intensiv mit den Technologien auseinandersetzen. Mir gefiel die Arbeit mit den Geodaten und den verschiedenen Dateiformaten wie TIFF, Geopackage und GeoJSON sehr und kann mir gut vorstellen weitere spannende Projekte zu verfolgen.

Ich bin zufrieden mit der Laufzeit, die wir für den Algorithmus verbessern konnten. Es erforderte viel Arbeit, eine geeignete Methode zu finden und diese auch noch zu implementieren. Nun können auch unerfahrenere Benutzer schnell und einfach ihre gewünschten Untersuchungsgebiete analysieren und schneller Erkenntnisse gewinnen.

Mit Kyra zusammenzuarbeiten war lehrreich und wirksam. Wir hatten bereits an mehreren Projekten gemeinsam gearbeitet, wobei sie stets sehr strukturiert und zielorientiert arbeitet. Als Quereinsteigerin in die Softwareentwicklung hat sie nicht dieselben Erfahrungen wie ich, weswegen sie bei gewissen Aufgaben Unterstützung brauchte. Wir haben versucht möglichst oft den "Pair Programming" Ansatz zu wählen. Das erhöhte den Lerneffekt und manchmal kommt man schneller an ein gemeinsames Ziel, weil man sich direkt austauschen kann. Ich bin sehr zufrieden mit dem Resultat dieser Arbeit und hoffe,

dass wir mit unserem Resultat zu einer besseren Zugänglichkeit der Zersiedelung und ihrer Bemessung beitragen können.

B

Abbildungs- und Tabellenverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

C

Literatur- und Quellenverzeichnis

[1] Christian 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.

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

[3] GeeksforGeeks. *What is RUP (Rational Unified Process) and its Phases?* URL: https://www.geeksforgeeks.org/rup-and-its-phases (besucht am 14. 12. 2024).

[4] Atlassian Git Tutorial. *Git Feature Branch Workflow*. URL: https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow (besucht am 06. 11. 2024).

[5] 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. 11. 2024).

[6] admin.ch. *Bundesamt für Raumentwicklung ARE*. URL: https://www.are.admin.ch/are/de/home.html (besucht am 14. 12. 2024).

[7] 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/.

[8] OST - Ostschweizer Fachhochschule. *IFS Institut für Software*. URL: https://www.ost.ch/de/forschung-und-dienstleistungen/informatik/ifs-institut-fuer-software (besucht am 14. 12. 2024).

[9] GNU. *GNU General Public License*. URL: https://www.gnu.org/licenses/gpl-3.0.html (besucht am 18. 12. 2024).

[10] QGIS Python Plugins Repository. *Urban Sprawl Metrics (USM) Toolset - QGIS Plugin*. URL: https://plugins.qgis.org/plugins/usm_calculator-main (besucht am 13. 11. 2024).

[11] Nir Herlihy Maurice; Shavit. *The Art of Multiprocessor Programming, Revised Reprint*. p. 14.: "Some computational problems are 'embarrassingly parallel': they can easily be divided into components that can be executed concurrently." Elsevier, 2020. ISBN: 9780123977953.

[12] ISO25000.com. *ISO/IEC 25010*. URL: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010 (besucht am 18. 12. 2024).

[13] Robert C. Martin. *The Single Responsibility Principle*. URL: https://blog.cleancoder.com/uncle-bob/2014/05/08/SingleReponsibilityPrinciple.html (besucht am 11. 12. 2024).

[14] Jochen A.G. Jaeger und Christian Schwick. "Improving the measurement of urban sprawl: Weighted Urban Proliferation (WUP) and its application to Switzerland". In: *Ecological Indicators* 38 (2014), S. 294–308.

[15] Martin Behnisch, Tobias Krüger und Jochen A. G. Jaeger. "Rapid rise in urban sprawl: Global hotspots and trends since 1990". In: *PLOS Sustainability and Transformation* 1.11 (2022), S. 1–30.

[16] Ernest I. Hennig u. a. "Multi-scale analysis of urban sprawl in Europe: Towards a European de-sprawling strategy". In: *Land Use Policy* 49 (2015), S. 483–498.

[17] Wolfram MathWorld. *Distance*. URL: https://mathworld.wolfram.com/Distance.html (besucht am 16. 12. 2024).

[18] Robert C. Martin. *The Clean Architecture*. Available at https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html, Accessed on 30.05.2024. Aug. 2012. URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html.

[19] Vue.js. *Composition API FAQ - Why Composition API*. URL: https://vuejs.org/guide/extras/composition-api-faq.html#why-composition-api (besucht am 11. 12. 2024).

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

[21] Numba. *Numba User Manual - Compiling Python code with @jit*. URL: https://numba.readthedocs.io/en/stable/user/jit.html (besucht am 11. 12. 2024).

[22] Numba. *Numba User Manual - Automatic paralleization with @jit*. URL: https://numba.readthedocs.io/en/stable/user/parallel.html#numba-parallel (besucht am 11. 12. 2024).

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

[24] GDAL. *GDAL Documentation*. URL: https://gdal.org/ (besucht am 18. 12. 2024).

[25] Starlette. *Starlette - The little ASGI framework that shines.* . URL: https://www.starlette.io/ (besucht am 13. 11. 2024).

[26] Numba. *Numba - A High Performance Python Compiler*. URL: https://numba.pydata.org/ (besucht am 11. 12. 2024).

[27] Joblib. *Joblib: running Python functions as pipeline jobs*. URL: https://joblib.readthedocs.io/en/latest/ (besucht am 18. 12. 2024).

[28] NumPy. *NumPy Documentation*. URL: https://numpy.org/doc/stable/ (besucht am 18. 12. 2024).

[29] Uvicorn. *Uvicorn Documentation*. URL: https://www.uvicorn.org/ (besucht am 18. 12. 2024).

[30] pytest.org. *pytest: helps you write better programs*. URL: https://docs.pytest.org/en/latest/ (besucht am 15. 12. 2024).

[31] Poetry. *Poetry Documentation*. URL: https://python-poetry.org/docs/ (besucht am 18. 12. 2024).

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

[33] Vite. *Getting Started*. URL: https://vitejs.dev/guide/ (besucht am 18. 12. 2024).

[34] Vue.js. *Vue.js Introduction*. URL: https://v3.vuejs.org/guide/introduction.html (besucht am 18. 12. 2024).

[35] Leaflet. *Leaflet - a Javascript library for interactive maps*. URL: https://leafletjs.com/index.html (besucht am 13. 11. 2024).

[36] Vue Leaflet. *Vue Leaflet - Harness the power of Leaflet in VueJs*. URL: https://vue2-leaflet.netlify.app/ (besucht am 11. 12. 2024).

[37] Ant Design Vue. *Ant Design Vue Documentation*. URL: https://antdv.com/docs/vue/introduce (besucht am 18. 12. 2024).

[38] Axios. *Axios Docs - Getting Started*. URL: https://axios-http.com/docs/intro (besucht am 18. 12. 2024).

[39] Turf.js. *Turf.js Documentation*. URL: https://turfjs.org/docs/intro (besucht am 18. 12. 2024).

[40] D3.js. *The JavaScript library for bespoke data visualization*. URL: https://d3js.org/ (besucht am 18. 12. 2024).

[41] GeoTIFF/georaster. *Github - georaster*. URL: https://github.com/GeoTIFF/georaster (besucht am 18. 12. 2024).

[42] Flake8. *Flake8 - Your Tool For Style Guide Enforcement*. URL: https://flake8.pycqa.org/en/latest/ (besucht am 15. 12. 2024).

[43] peps.python.org. *PEP 8 - Style Guide for Python Code*. URL: https://peps.python.org/pep-0008/ (besucht am 11. 12. 2024).

[44] Pylint. *Pylint - code analysis for Python*. URL: https://www.pylint.org/ (besucht am 15. 12. 2024).

[45] mypy 1.13.0 documentation. *mypy - Welcome to mypy documentation!* URL: https://mypy.readthedocs.io/en/stable/ (besucht am 15. 12. 2024).

[46] ESLint - Pluggable JavaScript linter. *Find and fix problems in your JavaScript code - ESLint*. URL: https://eslint.org/ (besucht am 15. 12. 2024).

[47] QGIS Web Site. *QGIS Overview*. URL: https://www.qgis.org/project/overview/ (besucht am 13. 11. 2024).

[48] GeeksforGeeks. *Introduction to Model-View-View Model (MVVM)*. URL: https://www.geeksforgeeks.org/introduction-to-model-view-view-model-mvvm/ (besucht am 16. 12. 2024).

[49] Amazon AWS. *What is RESTful API? - RESTful API Explained - AWS — aws.amazon.com*. https://aws.amazon.com/what-is/restful-api/?nc1=h_ls. [Aufgerufen am 16.12.2024].

[50] Monorepo.tools. *Monorepo Explained*. URL: https://monorepo.tools/ (besucht am 06. 11. 2024).

[51] Docker. *Docker Documentation*. URL: https://docs.docker.com/ (besucht am 18. 12. 2024).

[52] Traefik Labs. *Documentation*. URL: https://doc.traefik.io/ (besucht am 18. 12. 2024).

[53] Balsamiq. *Balsamiq: Fast, focused wireframing for teams and individuals*. URL: https://balsamiq.com/ (besucht am 15. 12. 2024).

[54] OpenStreetMap. *OpenStreetMap*. URL: https://www.openstreetmap.org/about (besucht am 15. 12. 2024).

[55] Leaflet.TileLayer.Swiss. *Overview - Leaflet.TileLayer.Swiss*. URL: https://leaflet-tilelayer-swiss.karavia.ch/ (besucht am 15. 12. 2024).

[56] Starlette. *Starlette - API Schemas*. URL: https://www.starlette.io/schemas/ (besucht am 13. 11. 2024).

[57] EPSG.io. *EPSG:4326 - WGS 84*. URL: https://epsg.io/4326 (besucht am 16. 12. 2024).

[58] yngvem (Github). *Parallelising Python Code*. URL: https://github.com/yngvem/parallelising-python (besucht am 16. 12. 2024).

[59] Dask. *Dask is a Python library for parallel and distributed computing*. URL: https://docs.dask.org/en/stable/ (besucht am 16. 12. 2024).

[60] Ray. *An open source framework to build and scale your ML and Python applications easily*. URL: https://docs.ray.io/en/latest/ (besucht am 16. 12. 2024).

[61] 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 14. 12. 2024).

[62] HP Inc. Support. *HP Elite Dragonfly 13.5 inch G3 Notebook PC specifications*. URL: https://support.hp.com/bg-en/document/ish_6020545-6020637-16 (besucht am 14. 12. 2024).

[63] Bundesamt für Statistik BFS. *Applikation der Schweizer Gemeinden - Geographische Kennzahlen*. URL: https://www.agvchapp.bfs.admin.ch/de/kennzahlen/query (besucht am 16. 12. 2024).

[64] Bundesamt für Statistik BFS. *Flächenberechnung*. URL: https://www.bfs.admin.ch/bfs/de/home/statistiken/raum-umwelt/erhebungen/area/datenauswertung/flaechenberechnung.html (besucht am 16. 12. 2024).

[65] Bundesamt für Statistik BFS. *Ständige Wohnbevölkerung nach Staatsangehörigkeitskategorie, Geschlecht und Gemeinde, definitive Jahresergebnisse, 2023*. URL: https://www.bfs.admin.ch/bfs/de/home/statistiken/bevoelkerung/stand-entwicklung.assetdetail.32229143.html (besucht am 05. 12. 2024).

[66] Gemeinde Murten Morat. *Gemeindeplan*. URL: https://www.murten-morat.ch/_docn/4720765/Gemeindeplan.pdf (besucht am 05. 12. 2024).

[67] Cumün Scuol - Gemeinde Scuol. *Bainvgnü in der Gemeinde Scuol*. URL: https://www.scuol.net/de.html/251 (besucht am 05. 12. 2024).

[68] Stadt Rapperswil-Jona. *Stadtporträt*. URL: https://www.rapperswil-jona.ch/diestadt (besucht am 05. 12. 2024).

[69] Swisstopo. *Swisstopo - Bundesamt für Landestopografie*. URL: https://www.swisstopo.admin.ch/de (besucht am 18. 12. 2024).

[70] Bundesamt für Zivilluftfahrt BAZL. *Opendata.swiss - Bebaute Gebiete nach Schweizer Luftfahrtrecht*. URL: https://opendata.swiss/de/dataset/bebaute-gebiete-nach-schweizer-luftfahrtrecht/resource/9dea5960-17fb-450f-b3b9-f7cfba1db840 (besucht am 18. 12. 2024).

[71] Atlassian. *Kanban - How the kanban methodology applies to software development*. URL: https://www.atlassian.com/agile/kanban (besucht am 16. 12. 2024).

[72] Interaction Design Foundation. *Keep It Simple, Stupid (KISS)*. URL: https://www.interaction-design.org/literature/topics/keep-it-simple-stupid (besucht am 11. 12. 2024).

[73] Python Documentation (3.13.1). *typing - Support for type hints*. URL: https://docs.python.org/3/library/typing.html (besucht am 15. 12. 2024).

[74] 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 15. 12. 2024).

[75] Swagger. *Swagger Documentation*. URL: https://swagger.io/docs/ (besucht am 18. 12. 2024).

[76] Sonar. *Better Code - Better Software*. URL: https://www.sonarsource.com/ (besucht am 18. 12. 2024).

[77] 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 18. 12. 2024).

D

Glossar und Abkürzungsverzeichnis

DIS Dispersion (Streuung) der Siedlungsflächen, (gemessen in UPU/m^2). 17–20, 36, 37, 48

GDAL Geospatial Data Abstraction Library, eine Bibliothek für Geodaten. 3, 23, 37, 45

IFS Institut für Software, Institut der OST Ostschweizer Fachhochschule. i, 4

LUP Land Uptake per Person (LUP), de: Flächeninanspruchnahme pro Person (FI). 17–19, 36

monorepo Ein Monorepo ist ein Repository, in dem der gesamte Code eines Projekts verwaltet wird. 24

PBA Proportion of Built-up Areas (PBA), de: Anteil der besiedelbaren Fläche (ASF). 17, 19, 36

Pixel Punkt in einem Raster, der die kleinste Einheit bildet. Meistens im SI-Raster 15x15 Meter gross. 6, 18, 37

SI Sprawl at Index (SI), de: Zersiedelungsmass an einem Punkt. ii, v, 2, 3, 5, 6, 8, 18, 24, 36, 37, 39, 48, 54, 59

TIFF Tagged Image File Format, ein Dateiformat für Rastergrafiken. Diese können geodätische Informationen enthalten. 36, 54, 58, 59

TS Total Sprawl (TS), de: Gesamtdurchsiedelung. 17, 36

UD Utilization Density (UD), de: Ausnützungsdichte. 17, 18, 36

UP Urban Proliferation (UP), de: Urbane Durchdringung. 17, 19, 36

UPU Urban Permeation Units (UPU), de: Durchsiedlungseinheiten (DSE). 17–19

USM-Calculator Webapplikation zur Berechnung der Zersiedelung. 8, 34, 53, 58, 59, 65

WSPC Weighted Sprawl Per Capita (WSPC), de: Gewichtete Zersiedelung pro Kopf. 17, 36

WUP Weighted Urban Proliferation (WUP), de: Zersiedelung (Z), gemessen in UPU/m^2. ii, 17, 20, 34, 36, 58, 64, 65

Z Zersiedelung. ii, 19