

geobeet

Gartenplanung mit Schweizer Geodaten, vom Luftbild bis zum massstäblichen Plan

Eine Webanwendung, die amtliche Schweizer Geodaten mit der Praxis des Gemüsegartens verbindet. Flächen werden auf dem swisstopo-Luftbild erfasst oder mit dem Bandmass ausgemessen, Kulturen über die Saison geplant, die Fruchtfolge geprüft, und daraus entsteht ein massstäblicher, druckbarer Gartenplan. Konzept, Geodaten, Backend, Frontend und Planzeichnung in einer Hand.

946

Zeilen Python

22

typisierte API-Endpunkte

36

Kulturen im Katalog

10 cm

Auflösungsgrenze, gemessen

0.00003 mm

maximaler Kantenfehler

2

Koordinatensysteme, sauber getrennt

Gabriel Suter · Einzelumsetzung · Konzept, Geodaten, Backend, Frontend, Planzeichnung

Python · FastAPI · Pydantic · SQLite · Shapely · pyproj · Leaflet · Juli bis September 2026

kontakt@gabrielsuter.dev · gabrielsuter.dev · Persönliches Projekt, Gemüsegarten in Däniken SO

1 Kurzfassung

geobeet ist ein Werkzeug für die Planung eines realen Gemüsegartens in Däniken SO. Es verbindet drei Dinge, die sonst selten zusammenkommen: amtliche Geodaten, präzise Flächenberechnung und gärtnerische Fachlogik. Flächen entstehen gezeichnet oder aus gemessenen Massen, Pflanzungen werden pro Saison geführt, die Fruchtfolge prüft mit, und die Planansicht gibt das Ergebnis als druckbares A4 aus.

Der Kern in einem Satz

Geometrien werden in **WGS84** gespeichert, aber jede Berechnung läuft über eine Transformation nach **LV95 (EPSG:2056)**: nur so stimmen Quadratmeter und Kantenlängen in der Schweiz auf den Millimeter.

2 Ausgangslage und Ziel

Ausgangslage war ein realer Gemüsegarten, geplant auf Papier und im Kopf. Gesucht war kein Gartentagebuch, sondern ein Werkzeug, das mit den tatsächlichen Massen des Grundstücks rechnet und die Entscheidungen einer Saison trägt.

Fünf Dinge sollten möglich werden: Beete abrufen und einsehen, Pläne für Woche, Monat und Jahr erstellen, per Text mit einer KI planen, eine ungenutzte Fläche verplanen, und eine Steinplatte verplanen, auf der früher ein Gewächshaus stand. Daraus ergab sich die Richtung: amtliche Geodaten als Grundlage statt einer leeren Zeichenfläche, und Messwerte statt Schätzungen.

Im Verlauf kam eine Einsicht dazu, die das Eingabeverfahren veränderte: Das Luftbild endet bei 10 cm pro Pixel, und ein Bandmass ist zehnmal genauer. Wie es dazu kam, steht in Abschnitt 5. Nebenziel ist die Verwendung als Portfolio-Arbeit für Python, Schnittstellenarbeit und Frontend.

Meine Rolle

- **Konzept:** Anforderungen aus der eigenen Gartenpraxis, bewusste Abgrenzung gegen Funktionen ohne konkreten Bedarf.
- **Geodaten:** swisstopo und Kanton Solothurn über WMTS, WMS, WFS und STAC; Koordinatentransformation nach LV95, Flächenberechnung, Konstruktion aus Messwerten.
- **Backend:** FastAPI mit Pydantic-Schemata, SQLite-Schema mit Migration, 22 typisierte Endpunkte, Fruchtfolge- und Kalenderlogik.
- **Frontend:** Leaflet-Karte und generierte SVG-Planzeichnung.
- **Recherche und Prüfung:** Auflösungsmessung gegen die echten Dienste, Genauigkeitstests der Geometrie, Layout- und Verdrahtungsprüfung.

3 Funktionsumfang

- **Flächen erfassen:** auf dem swisstopo-Luftbild zeichnen oder aus gemessenen Massen konstruieren; zehn Flächentypen vom Beet bis zum Gebäude.
- **Gehölze setzen:** Bäume und Sträucher als Punkt mit Kronendurchmesser, auf der Karte als Kreis in echter Kronengrösse.
- **Kulturen planen:** Katalog mit 36 Kulturen, Pflanzungen pro Fläche und Saison, automatisch gerechneter Erntetermin.

- **Fruchtfolge prüfen:** Warnung, wenn dieselbe Pflanzenfamilie zu kurz hintereinander auf derselben Fläche steht; die Prüfung läuft live bei der Eingabe und über den gesamten Bestand.
- **Plan erzeugen:** massstäblicher A4-Gartenplan in der Bildsprache der Landschaftsarchitektur, druckbar und als SVG exportierbar.
- **Bearbeiten:** Name, Typ, Notiz und Geometrie jeder Fläche ändern, Bepflanzung im Detailfenster führen.

4 Architektur

Bereich	Technologie
Sprache	Python 3.13
Web	FastAPI 0.141, uvicorn 0.52, Pydantic 2.13 für Schemata und Validierung
Geometrie	Shapely 2.1 und pyproj 3.7, Transformation zwischen WGS84 und LV95
Datenhaltung	SQLite 3.51 aus der Standardbibliothek, Schema als SQL-Datei mit Migration
Karte	Leaflet 1.9 mit Leaflet.draw, swisstopo-Luftbild als WMTS
Plan	Generiertes SVG, rough.js für den handgezeichneten Strich

Projektstruktur

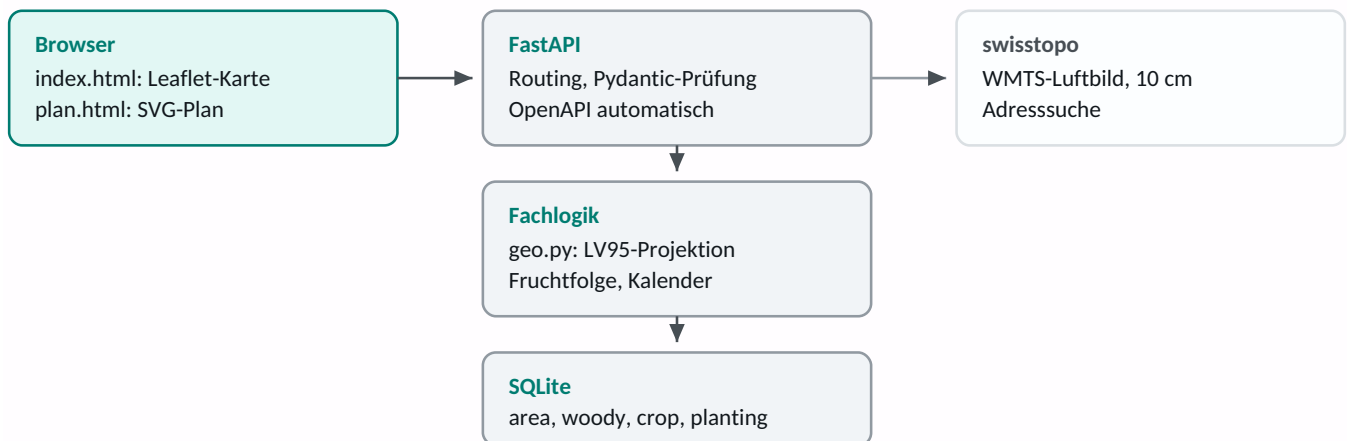
Datei	Aufgabe
<code>geobeet/main.py</code>	FastAPI-App, 22 Endpunkte
<code>geobeet/db.py</code>	SQLite-Verbindung, Migration, CSV-Import
<code>geobeet/geo.py</code>	Projektion, Flächenberechnung, Rechteckkonstruktion
<code>geobeet/models.py</code>	Pydantic-Schemata
<code>geobeet/static/</code>	<code>index.html</code> für die Karte, <code>plan.html</code> für die Planansicht
<code>data/</code>	<code>schema.sql</code> als Datenbankschema, <code>crops.csv</code> als Kulturenkatalog

Bewusst schlank gehalten

Kein Build-Schritt, kein Frontend-Framework, kein Datenbankserver. Beide HTML-Seiten sind self-contained, die Anwendung startet mit einem Befehl und legt ihre Datenbank selbst an. Postgres mit PostGIS wurde geprüft und verworfen, solange keine echten räumlichen Abfragen nötig sind. Die Geometrien liegen als GeoJSON-Text in SQLite, der Umstieg bleibt später ein überschaubarer Schritt.

Schichten

Schicht	Aufgabe
Browser	Leaflet-Karte und SVG-Planansicht, beide ohne Framework. Spricht das Luftbild direkt an.
FastAPI	Routing und Pydantic-Prüfung, die OpenAPI-Spezifikation entsteht automatisch.
Fachlogik	<code>geo.py</code> für die LV95-Projektion, dazu Fruchtfolge und Kalender.
SQLite	Tabellen <code>area</code> , <code>woody</code> , <code>crop</code> , <code>planting</code> .
swisstopo	WMTS-Luftbild mit 10 cm Auflösung und Adresssuche, ohne Zugangsdaten.



Der Browser spricht das Luftbild direkt an, die Anwendung selbst braucht dafür weder Zugangsdaten noch einen Proxy.

Wenn der Dienst nicht tut, was die Doku sagt

Die Amtliche Vermessung kommt als Vektor über den WFS des Kantons Solothurn. Der antwortet in **Version 2.0.0** mit **UnknownError**, und zwar unabhängig von der Abfrage; dasselbe passiert, sobald ein Ausgabeformat gesetzt wird, GeoJSON wie GML. Brauchbar ist der Dienst nur mit **VERSION=1.1.0**, dem Parameter **TYPENAME** statt **TYPENAMES** und ganz ohne **OUTPUTFORMAT**: Dann liefert er GML von selbst. Gefunden wurde das nicht in der Dokumentation, sondern indem alle Kombinationen der Reihe nach abgefragt wurden.

Schnittstelle

22 Endpunkte, vollständig typisiert. FastAPI erzeugt daraus automatisch eine OpenAPI-Spezifikation und eine interaktive Dokumentation unter `/docs`.

Endpunkt	Zweck
<code>GET /areas</code>	alle Flächen
<code>POST /areas</code>	Fläche anlegen, berechnet <code>area_m2</code>
<code>PATCH /areas/{id}</code>	Teiländerung, nur gesetzte Felder
<code>DEL /areas/{id}</code>	Fläche löschen
<code>POST /geometry/rectangle</code>	Rechteck aus Massen, rechnet nur
<code>GET /woodies</code>	Bäume und Sträucher
<code>GET /crops</code>	Kulturenkatalog
<code>POST /crops/import</code>	CSV neu einlesen
<code>GET /plantings</code>	Pflanzungen, filterbar nach Fläche und Saison
<code>POST /plantings</code>	Pflanzung anlegen, rechnet den Erntetermin
<code>GET /rotation/check</code>	verletzt diese Kultur die Fruchtfolge?
<code>GET /rotation/warnings</code>	alle Verstösse im Bestand
<code>GET /calendar</code>	Jahreskalender mit Aussaatfenster und Ernte
<code>GET /plan/data</code>	Flächen und Gehölze in Metern, auf den Planursprung verschoben

Auszug, dazu kommen Einzelabfragen, Löschrouten und die HTML-Seiten.

5 Technische Kernprobleme

Fünf Stellen, an denen das Projekt über «Formular speichert in Datenbank» hinausging. Jede davon ist gemessen oder recherchiert, nicht geraten.

KOORDINATEN

Zwei Koordinatensysteme sauber trennen

Problem

Leaflet und GeoJSON arbeiten in WGS84, also in Grad. Flächen und Distanzen lassen sich darin nicht sinnvoll rechnen: Ein Längengrad ist in Zürich kürzer als am Äquator.

Lösung

Die durchgehende Konvention lautet **speichern in WGS84, rechnen in LV95 (EPSG:2056)**. Sie gilt für die Flächenberechnung, die Rechteckkonstruktion und die Planansicht gleichermassen. Die Transformation ist an einer einzigen Stelle gekapselt (`geo.py`), die Transformer-Objekte werden einmalig gebaut und wiederverwendet.

Warum es zählt

Eine Konvention, an die man sich bei jedem Aufruf erinnern muss, ist keine Konvention, sondern eine Fehlerquelle mit Verzögerung. Sie hält nur, wenn der Code sie erzwingt und der falsche Weg gar nicht erst offensteht. Erschwerend kommt hier dazu, dass ein Fehler in der Projektion nicht abstürzt, sondern eine plausibel aussehende Zahl liefert. Solche Fehler fallen erst auf, wenn jemand mit dem Bandmass nachmisst.

MESSEN STATT ANNEHMEN

Die Auflösungsgrenze und die richtige Konsequenz

Problem

Ausgangswunsch war ein schärferes Luftbild, um Beetkanten genauer abzeichnen zu können. Statt zu optimieren wurde zuerst gemessen.

Untersuchung

- **WMTS-Zoomstufen 17 bis 23:** Kacheln bis z20, rund 10 cm pro Pixel, ab z21 HTTP 400.
- **Spektralanalyse Orthofoto 2024:** relative Leistung bei 10 cm Wellenlänge $2 \cdot 10^{-6}$, bestätigt dieselbe Grenze.
- **GeoTIFF-Header swissimage-dop10 per HTTP-Range:** intern JPEG-komprimiert Faktor 4.7, kein Gewinn.
- **Jahrgänge 2012 bis 2024 verglichen:** 2024 am schärfsten.
- **LiDAR-Oberflächenmodell:** 52 cm Rasterweite, fünfmal gröber.

Konsequenz

10 cm ist die physische Aufnahmeauflösung, schärfer existiert nicht. Also wurde das Ziel gewechselt: statt das Bild zu verbessern, wurde die **Eingabe** verbessert. Ein Bandmass ist bei einem 3-Meter-Hochbeet rund zehnmal genauer als das beste verfügbare Luftbild. Daraus entstand die Funktion «Ausmessen statt Abmalen». Was am Bild machbar war, wurde trotzdem gemacht: Oberhalb der nativen Zoomstufe schaltet die Karte auf `image-rendering: pixelated` um und legt eine Unsharp-Mask darüber, deren Radius mit dem Vergrößerungsfaktor mitwächst. Als Mass dient der mittlere Gradientenbetrag über das Graustufenbild, also die durchschnittliche Helligkeitsänderung von Bildpunkt zu Bildpunkt. An einem realen Ausschnitt steigt er von 1.76 auf **4.45**. Der erste Versuch mit festem 3×3-Kernel brachte nur acht Prozent. Bei achtfacher Vergrößerung wirkt ein so kleiner Kernel innerhalb eines einzigen Quellpixels und damit praktisch gar nicht.

Warum es zählt

Wer optimiert, bevor er misst, verbessert womöglich die falsche Sache. Die Grenze war hier physikalisch, nicht technisch, und die richtige Antwort deshalb kein besserer Filter, sondern ein anderes Eingabeverfahren.

GEOMETRIE

Rechtecke aus gemessenen Massen

Problem

Ein rechteckiges Hochbeet lässt sich auf einem 10-cm-Luftbild nur schätzen. Gefragt war ein Weg, gemessene Masse direkt in eine exakte Geometrie zu überführen.

Lösung

Der Nutzer gibt Länge, Breite und Ausrichtung ein und klickt die Position auf der Karte. Das Rechteck wird in LV95 konstruiert: Einheitsvektoren aus der Kompasspeilung, Ecken über die halben Kantenlängen. Erst die fertigen Ecken gehen zurück nach WGS84, dadurch stimmen die Kantenlängen unabhängig vom Breitengrad. Vorschau und Speicherung sind bewusst getrennt: `POST /geometry/rectangle` rechnet nur, damit das Frontend die Vorschau live nachführen kann, während der Nutzer am Ausrichtungsregler dreht. Gemessene Genauigkeit über sieben Testfälle: Kantenfehler maximal **0.00003 mm**, Flächenfehler **0 cm²**, Innenwinkel exakt 90°.

Warum es zählt

Ein Messwerkzeug ist nur so viel wert wie das Vertrauen in seine Anzeige. Getrennte Rechenwege für Anzeige und Ergebnis sind die klassische Stelle, an der dieses Vertrauen unbemerkt bricht: Die Abweichung ist klein, sieht nach nichts aus und zeigt sich erst, wenn die Reihe im Beet nicht aufgeht. Eine gemeinsame Berechnung nimmt diese Fehlerklasse ganz aus dem Spiel, statt sie hinterher zu testen.

DARSTELLUNG

Massstäblicher Plan als SVG

Problem

Ein Gartenplan muss gedruckt nachmessbar sein und zugleich wie eine Zeichnung aussehen, nicht wie ein Datenbankauszug.

Lösung

Die Planansicht erzeugt einen A4-Gartenplan in der Bildsprache der Landschaftsarchitektur. Der Trick für die Massstabstreue: **eine SVG-Einheit entspricht einem Millimeter Papier**. Bei 1:100 ist ein Meter damit genau 10 mm, und der Ausdruck lässt sich mit dem Lineal nachmessen. Jede Fläche wird dreilagig gezeichnet: farbiger Untergrund mit Schlagschatten, darüber die Materialschraffur, zuoberst eine ruhige Kontur. Der Schatten liegt bewusst nur auf dem Untergrund, läge er auf der Schraffur, bekäme jede einzelne Linie einen eigenen Schatten. Gezeichnet wird nach Flächentyp geschichtet, Boden unten, Bauten darüber, Beete und Gehölze zuoberst. Blattausrichtung und Massstab wählt die Anwendung selbst.

Warum es zählt

Der Unterschied zwischen einem Bild und einem Messinstrument liegt nicht im Aussehen, sondern darin, ob man ihm eine Zahl entnehmen darf. Ein hübscher Gartenplan, dessen Abstände nur ungefähr stimmen, verleitet dazu, im Frühling danach zu pflanzen und im Sommer festzustellen, dass die Reihe zu kurz war. Solange die Zeichnung an die gerechnete Geometrie gebunden bleibt, ist sie auch auf Papier eine Aussage über den echten Garten. Der Massstab ist deshalb keine Dekoration, sondern die eigentliche Funktion.

FACHLOGIK**Fruchtfolge: warnen statt blockieren****Problem**

Kohl- und Nachtschattenkrankheiten überdauern im Boden, deshalb gilt für dieselbe Pflanzenfamilie eine Anbaupause von drei Jahren. Eine Software kann das prüfen, aber sie kennt den Einzelfall nicht.

Lösung

Die Prüfung läuft live bei der Eingabe und zusätzlich über den gesamten Bestand. Sie **warnt, sie blockiert nicht**, und nennt Vorkultur und Jahr, damit die Warnung nachvollziehbar ist.

Warum es zählt

Ein Beet kann trotz Warnung die richtige Wahl sein. Das Fachurteil gehört zum Nutzer, nicht zur Software, und eine Warnung mit Begründung hilft dabei mehr als ein Verbot.

6 Datenmodell

Tabelle	Zweck und Besonderheit
<code>area</code>	Jede Fläche im Garten. <code>kind</code> deckt zehn Typen ab: Beet, Rasen, Terrasse, Weg, Steinplatte, Gebäude, Hecke, Sträucher, Wasser und ungenutzt. So kann die Planansicht einen ganzen Garten darstellen und nicht nur Bodenflächen. <code>area_m2</code> wird beim Speichern aus der Geometrie berechnet.
<code>woody</code>	Baum oder Strauch, bewusst Punkt statt Fläche , mit Kronendurchmesser. So werden Gehölze vermessen und so zeichnet die Landschaftsarchitektur sie.
<code>crop</code>	Eigener Kulturenkatalog aus <code>data/crops.csv</code> , 36 Einträge fürs Mittelland: Familie als Basis der Fruchtfolge, Pflanzabstand, Aussaatfenster, Kulturdauer, Zehrerkategorie.
<code>planting</code>	Konkrete Pflanzung auf einer Fläche in einer Saison. Historie und Grundlage für Kalender und Fruchtfolge.

Warum ein eigener Katalog statt einer Pflanzen-API

Die naheliegende Fremdquelle steht unter CC BY-SA 4.0 und würde abgeleitete Werke unter dieselbe Lizenz zwingen. Ein eigener CSV-Katalog ist lizenzfrei, passt zum konkreten Standort und lässt sich im Texteditor pflegen.

Schema-Migration in SQLite

Die Erweiterung von vier auf zehn Flächentypen erforderte eine Änderung der `CHECK`-Bedingung. Das kann SQLite nicht per `ALTER TABLE`, die Tabelle muss neu gebaut werden. Zwei Fallstricke wurden dabei gelöst:

- **Fremdschlüssel:** Ohne `PRAGMA legacy_alter_table = ON` schreibt SQLite beim Umbenennen die Referenz in `planting` auf die alte Tabelle um. Sie zeigt danach ins Leere.
- **Einmaligkeit:** Die Migration erkennt am gespeicherten Schema, ob sie bereits gelaufen ist, und läuft danach nie wieder.

Verifiziert mit vorheriger Datensicherung: Datenbestand unverändert, Fremdschlüssel intakt, `PRAGMA foreign_key_check` ohne Befund.

7 Qualität und offene Punkte

Das Projekt hat noch **keine automatisierte Testsuite**. Verifiziert wurde bisher durch gezielte Messung gegen die echten Schnittstellen und die echte Datenbank:

Prüfung	Ergebnis
Rechteckgeometrie, sieben Fälle	Kantenfehler 0.00003 mm, Fläche exakt, Winkel 90°
Vorschau gegen Gespeichertes	identische Geometrie und Fläche
Fruchtfolge über Jahresgrenzen	2024 sperrt 2025 bis 2027, gibt 2028 frei
Eingabevalidierung	ungültige Werte 422, fehlende Objekte 404, leerer Patch 400
Schema-Migration	Daten erhalten, Fremdschlüssel geprüft
Blattlayout des Plans	alle Punkte im Zeichenfeld, Legende ohne Überlauf
Frontend-Verdrahtung	HTML parst, keine fehlenden Element-IDs

Ein Befund, der zur Regel wurde

Edge und Chrome invertieren helle Seiten automatisch zu Dunkelmodus. Ein Gartenplan ist aber immer hell, deshalb steht in beiden Seiten `color-scheme: only light`. Solche Stolpersteine werden in der Projektnotiz festgehalten, damit sie nicht zweimal Zeit kosten.

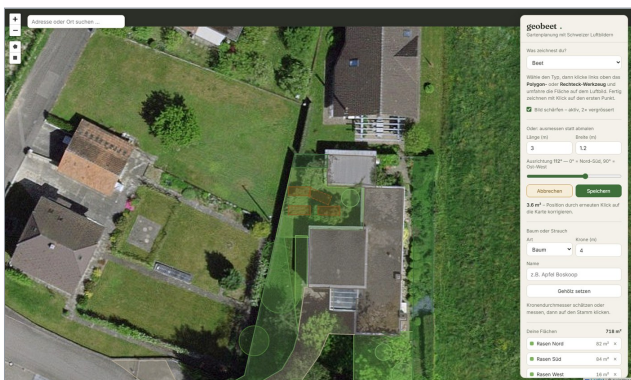
Offen

- Automatisierte Testsuite mit pytest.
- Kalenderseite mit Jahresbalken, die Daten liefert die API bereits.
- Beschriftungen im Plan prüfen sich nicht gegenseitig auf Kollision, dicht liegende Beete laufen bei kleinem Massstab ineinander.
- Als nächstes Feature geplant: Sonnenstunden pro Beet aus dem LiDAR-Oberflächenmodell, Datenquellen sind bereits verifiziert.
- Danach ein KI-Chat mit Tool-Use auf die eigene Datenbank, damit Fakten aus den echten Daten kommen statt aus dem Gedächtnis des Modells.
- Deployment, bisher läuft die Anwendung lokal.

8 Einblicke



Flächenerfassung auf dem swisstopo-Luftbild, zehn Flächentypen.



Ausmessen statt Abmalen: Länge, Breite und Kompasspeilung ergeben ein Rechteck mit Live-Vorschau.



Masstäblicher A4-Gartenplan mit Materialschraffuren, Nordpfeil und Legende, erzeugt aus den erfassten Flächen.

9 Was das Projekt zeigt

Fähigkeit	Wo sie sichtbar wird
API-Entwurf	22 typisierte Endpunkte, saubere Trennung von Rechnen und Speichern, automatische OpenAPI-Dokumentation
Arbeit mit fremden Schnittstellen	WMTS, WMS, WFS, STAC und die Abfrage-API von geo.admin.ch, inklusive Umgang mit undokumentiertem Verhalten wie dem nur in Version 1.1.0 nutzbaren WFS des Kantons
Fachdomäne durchdringen	Koordinatensysteme, Massstabstreue, Fruchtfolge und die Darstellungskonventionen der Landschaftsarchitektur
Entscheiden statt annehmen	Die Auflösungsfrage wurde gemessen, nicht geschätzt, und das führte zum Verwerfen des ursprünglichen Lösungswegs
Datenbankpflege	Schema-Migration mit Datenerhalt und Fremdschlüsselprüfung
Frontend ohne Framework	Karte und druckfähiger SVG-Plan in reinem JavaScript
Dokumentieren	Eine gepflegte Projektnotiz hält Entscheidungen, verifizierte Fakten und Stolpersteine fest

Stand	September 2026 · Karte, Ausmessen, Gehölze, Pflanzungen und Planansicht fertig, Weiterführung geplant
Kontakt	Gabriel Suter · kontakt@gabrielsuter.dev · gabrielsuter.dev
Quellcode	Privates GitHub-Repository, Einsicht und Vorführung auf Anfrage
Betrieb	Läuft lokal mit einem Befehl, Deployment offen
Kartengrundlage	© swisstopo, Amtliche Vermessung Kanton Solothurn