



Geodaten-Austauschplattform für Blaulichtorganisationen

Bachelorarbeit

Abteilung Informatik

Autoren

Martin Eisenhammer
Josua Stähli

Betreuer

Prof. Stefan Keller

Industriepartner

Markus Honegger, honeyconsult, Wetzikon

Experte

Claude Eisenhut, Eisenhut Informatik AG, Burgdorf

Gegenleser

Prof. Dr. Markus Stolze

Einleitung

Seite 1 bis 10

Abstract, Management Summary, Inhaltsverzeichnis und Abbildungsverzeichnis

Teil I: Übersicht

Seite 11 bis 19

Allgemeine Projektübersicht und Aufgabenstellung

Teil II: Projektdokumentation

Seite 20 bis 73

Aus Software-Engineering-Sicht relevante Dokumente

Teil III: Projektmanagement

Seite 74 bis 92

Beschreibung zum Vorgehen während des Projekts

Teil IV: Softwaredokumentation

Seite 93 bis 114

Anleitungen zur Installation und Benutzung von BLUshare

Teil V: Anhang

Seite 115 bis 123

Anhänge und Verzeichnisse

Abstract

Diese Bachelorarbeit ist mit dem Ziel entstanden, den Austausch von Geodaten zwischen Blaulichtorganisationen (Rettungsdienste, Feuerwehr und Polizei) zu vereinfachen. Solche Geodaten können beispielsweise Baustelleninformationen, Points of Interest oder Detailpläne von grösseren Gebäudekomplexen sein. Es sind Daten, die den Einsatzkräften helfen, einen Einsatzort ohne Hindernisse zu erreichen und sich vor Ort schnell zurechtzufinden. Solche Daten erfasst eine Blaulichtorganisation für das eigene primäre Einsatzgebiet. Es kommt aber öfter vor, dass diese auch Einsätze in einem fremden Einsatzgebiet fahren müssen, wenn die lokale Organisation gerade voll ausgelastet ist. Nun wäre es wünschenswert, wenn auf die von der lokalen Organisation erfassten Daten zugegriffen werden könnte.

Für die Umsetzung einer solchen Datensynchronisations-Plattform wurden verschiedene Technologien evaluiert. Als am geeignetsten herausgestellt hat sich GeoNode in Kombination mit GeoServer und PostGIS. GeoNode ist ein Content Management System, welches auf dem Webframework Django basiert und auf die Verwaltung von Geodaten spezialisiert ist. Die primäre Programmiersprache für die Implementation ist Python.

Im Rahmen dieser Bachelorarbeit wurde eine Serverapplikation geschaffen, die es unterschiedlichen Benutzern erlaubt, untereinander Geodaten auszutauschen. Jedem Benutzer lässt sich ein Lese- und Erfassungsumfang zuweisen, welche die Schreib- und Leserechte geographisch eingrenzen. Aus Sicherheitsgründen kann ein normaler Benutzer nur die Daten bearbeiten, welche er selber erfasst hat.

Diese Synchronisationsfunktion wurde in BLUgis eingebaut, einem Geoinformationssystem für Rettungsdienste, welches sich bereits im Einsatz befindet. Innerhalb von diesem Tool erfolgt die Datensynchronisation mit BLUshare über einen einfachen Klick auf den Synchronisations-Button.

Management Summary

Ausgangslage

Damit Rettungsdienste, die Feuerwehr und Polizei einen Einsatzort schnell und ohne Umwege erreichen können, benötigen diese detailliertes Kartenmaterial, um Hindernisse unterwegs frühzeitig umfahren zu können und sich vor Ort schnell zurechtzufinden. Jeder Blaulichtorganisation ist ein primäres Einsatzgebiet zugeteilt, in welchem diese selber wichtige Daten wie Baustellen, Veranstaltungen, Points of Interest und Informationen zu Gebäudezufahrten erfassen. Tritt nun die Situation auf, dass beispielsweise ein Rettungsdienst gerade voll ausgelastet ist, dann wird vorübergehend ein Nachbar-Rettungsdienst für neue Einsätze in diesem Gebiet angeboten. Damit sich die Einsatzkräfte in dem neuen Gebiet zurechtfinden können, wäre es wünschenswert, dass diese Zugriff auf die bereits erfassten Daten des anderen Rettungsdienstes hätten. Bisher bestand jedoch noch keine Möglichkeit für einen unkomplizierten Geodaten austausch untereinander.

Vorgehen/Technologien

Das Ziel dieser Bachelorarbeit war es, eine Plattform zu schaffen, mit welcher Blaulichtorganisationen auf möglichst komfortable Art und Weise ihre erfassten Geodaten untereinander austauschen können. Als Grundlage für die Umsetzung wurde GeoNode verwendet, ein Content Management System, welches auf die Verwaltung von Geodaten spezialisiert ist. Darauf aufbauend wurde ein Synchronisationsalgorithmus entwickelt, welcher einen fehlerfreien und sicheren Datenaustausch zwischen verschiedenen Blaulichtorganisationen erlaubt. Sowohl der server- als auch clientseitige Teil wurden in der Programmiersprache Python umgesetzt. Für den Web Client, welcher die Administration des Servers erlaubt, wurde auf aktuelle Webtechnologien gesetzt. Die Geodaten werden serverseitig in einer PostGIS-Datenbank abgelegt.

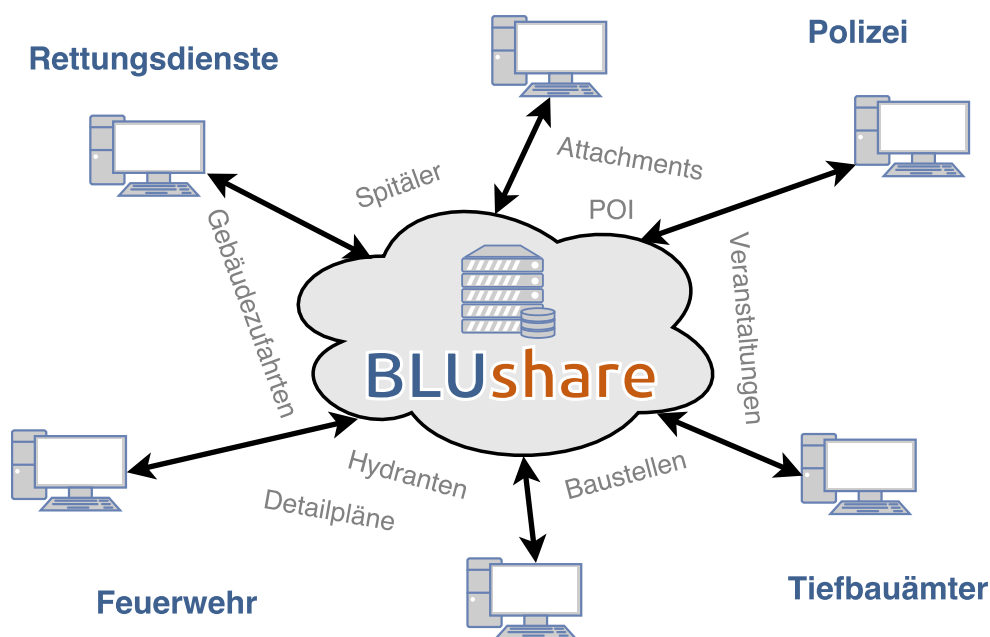


Abbildung 1: Struktur von BLUshare

Ergebnisse

Das Resultat der Arbeit ist eine Serverapplikation namens BLUshare, welche Geodaten von verschiedenen Blaulichtorganisationen entgegennimmt und gebündelt wieder zum Download bereitstellt. Die Synchronisationsfunktion wurde in BLUgis, einem bereits im Einsatz befindenen Geoinformationssystem für Rettungsdienste, eingebaut. Der Datenaustausch erfolgt in dieser Applikation über einen einfachen Klick auf den Synchronisations-Button, wodurch die eigenen Daten hochgeladen und die Daten der Nachbar-Organisationen heruntergeladen werden. Die selber erfassten Daten bleiben dabei geschützt vor unberechtigten Änderungen.

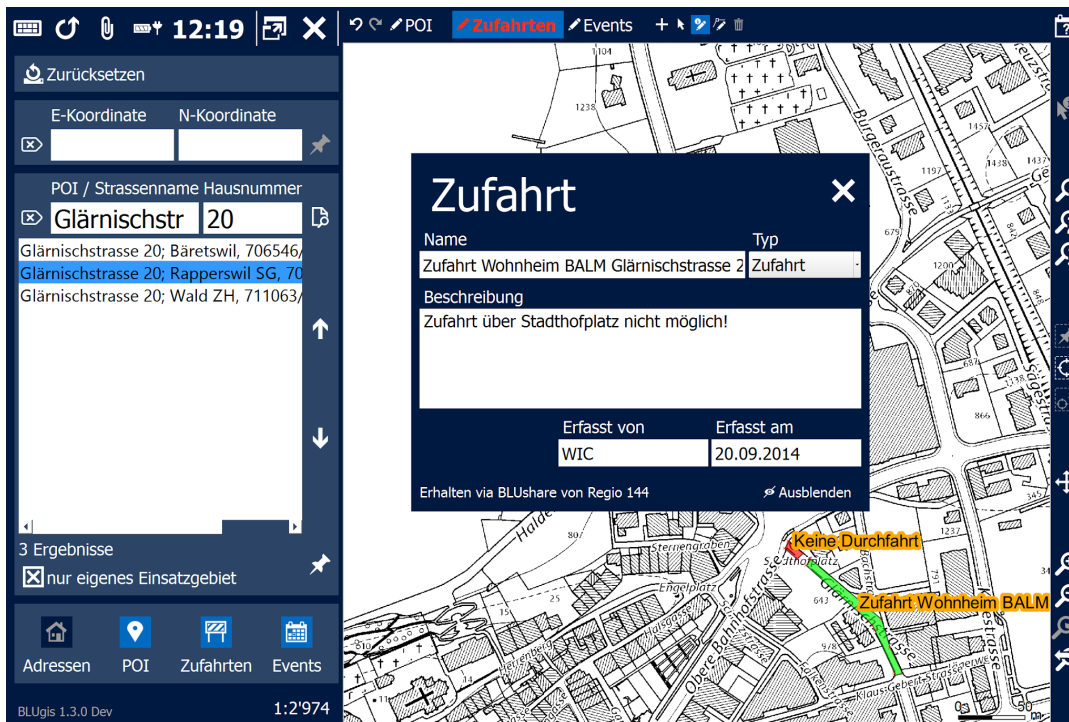


Abbildung 2: BLUgis mit einem via BLUshare erhaltenen Objekt

Ausblick

Die Serverapplikation kann um weitere Features erweitert werden, wie zum Beispiel die Möglichkeit bestimmte Geodaten für den Up- und Download zu filtern. Ausserdem könnte eine zentrale Baustellenverwaltung durch geeignete Imports realisiert werden, was den einzelnen Blaulichtorganisationen Arbeit ersparen würde. Als weitere Geodaten könnten beispielsweise auch Hydranten für die Feuerwehr hinzugenommen werden. Denkbar wäre auch eine Versionisierung der Geodaten in BLUshare, was das System noch besser vor Datenvandalismus und ungewollten Änderungen schützt.

Weitere Informationen: www.blugis.ch



Abbildung 3: BLUgis im Einsatz bei der Regio 144 in Rüti (ZH)

Inhalt

ABSTRACT	3
MANAGEMENT SUMMARY	4
AUSGANGSLAGE.....	4
VORGEHEN/TECHNOLOGIEN.....	4
ERGEBNISSE.....	5
AUSBlick.....	5
INHALT	7
ABBILDUNGSVERZEICHNIS	10

Teil I: Übersicht

1 PROJEKTbeschreibung	12
1.1 PROJEKTRAHMEN.....	12
1.2 BETEILIGTE PERSONEN.....	12
1.3 AUFGABENSTELLUNG.....	12
2 REALISIERUNG	16
2.1 VORGEHEN.....	16
2.2 ARBEITSAUFTEILUNG.....	17
3 RESULTATE	18
3.1 ZIELERREICHUNG.....	18
3.2 BEWERTUNG.....	18
4 AUSBLICK	19
4.1 VERWENDUNG.....	19
4.2 ERWEITERUNGSMÖGLICHKEITEN.....	19

Teil II: Projektdokumentation

5 VISION	21
5.1 EINLEITUNG.....	21
5.2 AUSGANGSLAGE.....	21
5.3 MOTIVATION.....	21
5.4 ZIEL.....	21
5.5 BEISPIELSZENARIO.....	22
6 ANFORDERUNGEN	23
6.1 FUNKTIONALE ANFORDERUNGEN.....	23
6.2 USE CASES.....	24
6.3 NICHTFUNKTIONALE ANFORDERUNGEN.....	31
7 DOMAINANALYSE	34
7.1 DATENMODELL BLUGIS (SQLITE).....	34
7.2 DATENMODELL BLUSHARE (POSTGRESQL).....	36

7.3	DATENSTRUKTUREN GEONODE (DJANGO)	36
8	TECHNOLOGIEANALYSE	38
8.1	GEODATENBANKEN	38
8.2	MAPSERVER	38
8.3	GEODATEN CMS	39
8.4	BENUTZERAUTHENTIFIZIERUNG	42
8.5	DATENÜBERTRAGUNG/ VERSIONISIERUNG	42
8.6	ÜBERTRAGUNGSFORMAT	43
8.7	FAZIT	45
9	GEONODE	46
9.1	EINLEITUNG	46
9.2	FUNKTIONEN	46
9.3	SYSTEMANFORDERUNGEN	47
9.4	UNTERSTÜTZTE WEBBROWSER (AUSWAHL)	47
9.5	INHALTE	47
9.6	BENUTZERMANAGEMENT	48
9.7	KOMPONENTEN	50
9.8	ARCHITEKTUR	51
9.9	LIZENZ (GPLV3)	52
9.10	WEBOBERFLÄCHE	53
9.11	VERKNÜPFUNG GEONODE UND GEOSERVER	53
9.12	GEONODE + QGIS SERVER?	53
9.13	DATENIMPORT	54
9.14	EINSCHRÄNKUNGEN/PROBLEME	55
9.15	COMMUNITY	55
9.16	FAZIT	55
10	DESIGN	56
10.1	AUSGANGSLAGE	56
10.2	ZIELE	57
10.3	PHYSISCHE ARCHITEKTUR	57
10.4	LOGISCHE ARCHITEKTUR	58
10.5	SYNCHRONISATION	61
10.6	SICHERHEIT	64
10.7	WEBCLIENT	65
11	TESTING	67
11.1	MANUELLE TESTS	67
11.2	ABNAHME	71
12	WEITERENTWICKLUNG	72

Teil III: Projektmanagement

13	VORGEHENSMODELL	75
13.1	SCRUM	75
14	ENTWICKLUNGSUMGEBUNG	78
14.1	INTEGRIERTE ENTWICKLUNGSUMGEBUNGEN	78

14.2	VERSIONSVERWALTUNG	78
14.3	DOKUMENTENVERWALTUNG	78
14.4	PROJEKTMANAGEMENTSOFTWARE.....	78
14.5	KOMMUNIKATION	79
15	PROJEKTPLANUNG	80
15.1	MEILENSTEINE	80
15.2	ANALYSEPHASE	80
15.3	SPRINT 1	81
15.4	SPRINT 2	83
15.5	SPRINT 3	84
15.6	SPRINT 4	86
15.7	ABSCHLUSSPHASE.....	88
15.8	PRÄSENTATION	88
15.9	MÜNDLICHE PRÜFUNG.....	88
16	RISIKOMANAGEMENT	89
16.1	RISIKEN	89
16.2	PRÄVENTIONSMASSNAHMEN	89
16.3	MASSNAHMEN BEI EINTRITT.....	90
16.4	EINGETRETENE RISIKEN.....	91
17	QUALITÄTSSICHERUNG	92

Teil IV: Softwaredokumentation

18	INSTALLATIONSANLEITUNG.....	94
18.1	VORAUSSETZUNGEN	94
18.2	INSTALLATION GEONODE MIT BLUSHARE	94
19	BENUTZERHANDBUCH.....	100
19.1	BLUSHARE-WEBCLIENT.....	100
19.2	BLÜGIS.....	104
20	SCHNITTSTELLEN	110
20.1	DATEN-UPLOAD.....	110
20.2	DATEN-DOWNLOAD	111
20.3	DATENFORMAT.....	112

Teil V: Anhang

PERSÖNLICHE BERICHTE	116
MARTIN EISENHAMMER	116
JOSUA STÄHLI	117
LITERATURVERZEICHNIS	118

Abbildungsverzeichnis

Abbildung 1: Struktur von BLUshare	4
Abbildung 2: BLUgis mit einem via BLUshare erhaltenen Objekt	5
Abbildung 3: BLUgis im Einsatz bei der Regio 144 in Rüti (ZH).....	6
Abbildung 4: Use-Case-Diagramm.....	24
Abbildung 5: Datenmodell von BLUgis vor den Anpassungen	34
Abbildung 6: Verändertes Datenmodell von BLUgis	36
Abbildung 7: Architektur von GeoNode	52
Abbildung 8: Layer aus GeoServer in GeoNode einfügen	54
Abbildung 9: Physische Architektur von BLUgis	56
Abbildung 10: Architektur von BLUgis mit BLUshare	58
Abbildung 11: Logische Architektur von BLUshare	59
Abbildung 12: Logische Architektur von BLUgis	60
Abbildung 13: Datensynchronisation zwischen BLUgis und BLUshare	62
Abbildung 14: Startseite von BLUshare	66
Abbildung 15: Burndown Diagramm Sprint 1.....	82
Abbildung 16: Burndown Diagramm Sprint 2.....	84
Abbildung 17: Burndown Diagramm Sprint 3.....	86
Abbildung 18: Burndown Diagramm Sprint 4.....	87
Abbildung 19: Menüansicht GeoNode	100
Abbildung 20: Hinzufügen eines Benutzers.....	101
Abbildung 21: Eingabe eines Benutzers	101
Abbildung 22: Möglichkeiten zum Speichern.....	101
Abbildung 23: Berechtigungen eines Benutzers in GeoNode	102
Abbildung 24: Hinzufügen einer Gruppe.....	103
Abbildung 25: Mitglieder und Rollen einer Gruppe	103
Abbildung 26: Liste der Gruppen.....	104
Abbildung 27: Erfasste Daten in BLUgis	104
Abbildung 28: BLUgis ohne Daten	105
Abbildung 29: Erhaltene Daten via BLUshare.....	106
Abbildung 30: BLUgis Bearbeitungsmodus.....	106
Abbildung 31: Detailfenster eines auszublendenden Objekts	107
Abbildung 32: BLUgis - Detailfenster des ausgeblendeten Objekts	108
Abbildung 33: BLUgis mit ausgeblendetem Event	108
Abbildung 34: BLUgis Editiermodus mit ausgeblendeten Objekten	109

Teil I: Übersicht

1 Projektbeschreibung

1.1 Projektrahmen

Dieses Projekt wird im Rahmen einer Bachelorarbeit an der HSR Hochschule für Technik Rapperswil umgesetzt. Die Bachelorarbeit beginnt am 22.2.2016, das Abgabedatum ist am 17.6.2016. Die aufgewendete Zeit muss pro Person mindestens 360 Stunden betragen.

1.2 Beteiligte Personen

Die Bachelorarbeit wird von Josua Stähli und Martin Eisenhammer umgesetzt.

Betreut wird die Arbeit von Prof. Stefan Keller, Leiter und Gründer des Geometa Labs, welches Teil vom Institut für Software an der HSR ist.

Industriepartner: Markus Honegger, honeyconsult markus honegger, Wetzikon

Experte: Claude Eisenhut, Eisenhut Informatik AG, Burgdorf

Gegenleser: Prof. Dr. Markus Stolze

1.3 Aufgabenstellung

Diese Bachelorarbeit wurde von Markus Honegger der Firma honeyconsult in Auftrag gegeben und soll anhand der folgenden Aufgabenstellung, welche zu Beginn der Bachelorarbeit vorgegeben wurde, durchgeführt werden. Darin enthalten sind auch die erwarteten Lieferobjekte von dieser Arbeit.

BLUshare: Geodaten-Austauschplattform für Blaulichtorganisationen

Finale Version vom 4.4.2016

- Bachelorarbeit im Frühjahrssemester 2016
- Autoren: Josua Stähli und Martin Eisenhammer
- Betreuer: Prof. Stefan Keller, Institut für Software, HSR
- Industriepartner: Markus Honegger, honeyconsult markus honegger

Ausgangslage

Mit BLUgis steht Blaulichtorganisationen eine GIS-Applikation zur Verfügung, mit welcher unter anderem Baustellen, Anlässe, Zufahrten und Points-Of-Interest verwaltet und auf dem Einsatzfahrzeug abrufbar gemacht werden können. Jede Organisation, die BLUgis einsetzt, pflegt diese Daten selbstständig in ihrer organisationseigenen Geodatenbank.

Um eine organisationsübergreifende Nutzung dieser manuell erfassten Daten zu ermöglichen, soll eine Online-Plattform für den kontrollierten Austausch dieser Geodaten aufgebaut werden.

Nebst einer engen Anbindung an BLUgis soll es insbesondere für Nicht-BLUgis-Anwender auch möglich sein, Daten über einen Web-Client zu pflegen. Ebenfalls soll ein Daten-Export über das Web-Frontend möglich sein.

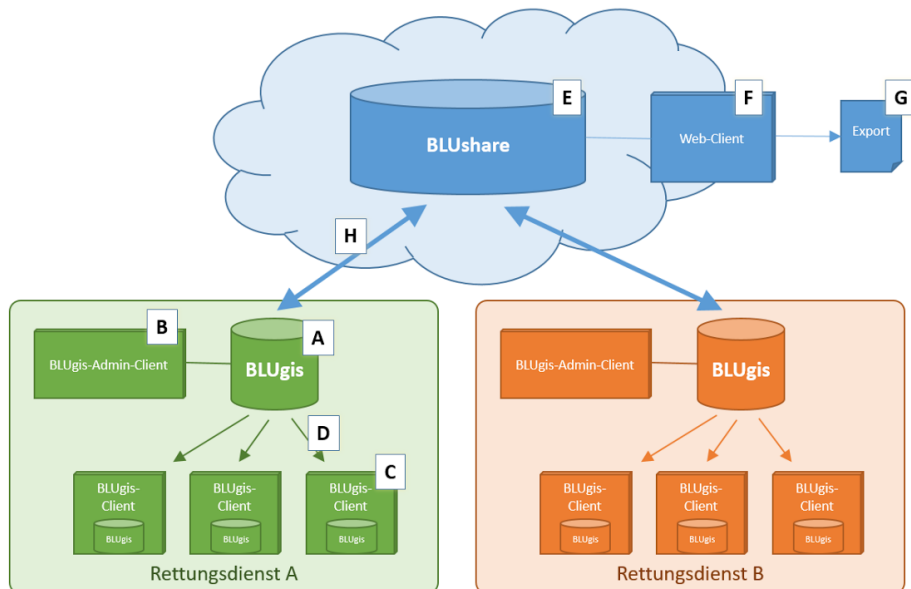
Rahmenbedingungen

- Es muss durchgehend Open Source-Software verwendet werden.
- Es sollen möglichst etablierte Software-Pakete eingesetzt werden.
- Als Desktop-GIS werden QGIS mit lokaler SQLite-Datenbank und einem Python-Plugin eingesetzt.
- Technologische Präferenzen sind Python, Django, PostGIS/PostgreSQL sowie HTML5/CSS.
- Die Nutzungsrechte bleiben wo möglich bei honeyconsult markus honegger (vgl. separate Vereinbarung), ansonsten gemäss den verwendeten Bibliotheken (GPL/LGPL)

Aufgabenstellung und Ziele

Erstellung einer Online-Plattform für den Austausch von Geodaten wie sie in BLUgis gehalten werden (sog. Bewegungsdaten¹) inklusive Integration in BLUgis gemäss folgendem Schema:

¹ Es handelt sich dabei um "Events" (Baustellen linien- und/oder flächenhaft), "Zufahrten" (flächenhaft) und "POI" (punkthaft). Im Folgenden werden diese der Einfachheit halber "Bewegungsdaten" genannt.



A	Organisationseigene BLUgis-Daten (vorbestehend); bestehend aus: <ul style="list-style-type: none"> • Eine QGIS-Projektdatei • Darstellungs-Konfigurations-Dateien (Icons) • Raster-Dateien als Basisdaten (Übersichtsplan, Swisstopo-Landeskarten) • Gebäudeadressen als Basisdaten (SpatialLite-DB) • Datenbank mit den manuell erfassten Bewegungsdaten (SpatialLite-DB) • Attachments zu den Bewegungsdaten (PDF, JPEG, PNG)
B	BLUgis-Applikation (vorbestehend, d.h. QGIS mit BLUgis-Plugin), im Admin-Mode betrieben (d.h. Erfassung von Bewegungsdaten möglich).
C	BLUgis-Client (vorbestehend) i.d.R. als Tablet auf dem Einsatzfahrzeug. Sämtliche BLUgis-Daten inkl. Raster-Dateien und SpatialLite-DBs werden lokal gespeichert um eine komplette Offline-Fähigkeit zu erreichen.
D	Daten (vorbestehend; insbesondere Bewegungsdaten, über Konfigurations-Datei genauer wählbar) werden über einen Funktionsaufruf im BLUgis-Admin-Client "B" in einem definierten Netzwerk-Share im Sinne einer Ein-Weg-Synchronisation bereitgestellt. SpatialLite-DBs werden dabei immer als gesamte Datei kopiert, es findet kein Abgleich auf Record-Ebene statt. Um die Daten auf den Clients zu übernehmen, muss auf diesen ebenfalls die Aktualisierungs-Funktion aufgerufen werden.
E	Zentrale Online-Datenbank für die Speicherung von Bewegungsdaten und deren Attachments. Auch wenn keine unmittelbaren Personendaten gespeichert werden, so handelt es sich dennoch um schätzenswerte Daten (z.B. Zugangshinweise zu Überbauungen, Telefonnummern, etc.).
F	Web-Client für die Darstellung und Bearbeitung der Bewegungsdaten, falls kein BLUgis zur Verfügung steht. Je nach angemeldetem Benutzer können auch auf diesem Weg nur Daten im zugewiesenen Perimeter erfasst bzw. nur eigene Daten geändert oder gelöscht werden. Als Kartenhintergrund sollen idealerweise der Übersichtsplan bzw. die Landeskarten verwendet werden.

G	Aus dem Web-Client sollen Daten exportiert werden können. In einem ersten Schritt soll ein einfacher Excel-Export der POI möglich sein (genaue Filterkriterien noch zu definieren).
H	<p>Ableich der Daten zwischen den einzelnen organisationseigenen BLUgis-Datenbanken und BLUshare. Der Vorgang wird seitens BLUgis-Admin-Client ausgelöst, zusammen mit dem Datenbereitstellungs-Vorgang für die BLUgis-Clients.</p> <p><i>Upload (BLUgis > BLUshare)</i> Es werden nur Bewegungsdaten (inkl. deren Attachments) zu BLUshare hinaufgeladen, die in der BLUgis-Datenbank entsprechend zur Freigabe markiert sind. BLUshare akzeptiert nur Daten aus dem Perimeter, für welchen der aktuelle Benutzer autorisiert ist.</p> <p><i>Download (BLUshare > BLUgis)</i> Es werden alle Daten (inkl. deren Attachments) heruntergeladen und in die lokale BLUgis-Datenbank integriert, die in dem aktuellen Benutzer zugewiesenen Perimeter liegen. Lokal gelöschte oder geänderte „fremde“ Daten sollen von der zukünftigen Synchronisation ausgenommen werden.</p>

Vorgehen und Lieferobjekte

Als Software-Entwicklungs-Projektmethode soll Scrum verwendet werden.

Erwartete Lieferobjekte:

- BLUshare-Plattform mit
 - Benutzer-Management
 - Anbindung an BLUgis
 - Web-Client mit Kartendarstellung inkl. Bewegungsdaten und Export/Download-Möglichkeit nach CSV/Excel
 - Deployed als Beta-Release
- Optional: Web-Client mit Create-Update-Delete-Möglichkeiten für Bewegungsdaten
- Software (Code englisch, GUI deutsch), einfach installierbar mit kurzer Installationsanleitung (deutsch)
- Projektdokumentation (deutsch).
- Weitere Dokumente (Kurzbeschreibung deutsch, Poster deutsch, Eigenständigkeitserklärung, Nutzungsrechte), Zitierweise und Termine (Abgabetermin, Präsentation der Bachelorarbeiten) gemäss Vorgaben des Studiengangs Informatik (vgl. HSR-Intranet) und nach Absprache mit dem Betreuer.

Abgabe: Alle Dokumente und Quellen der erstellten Software auf CD. Gebundene Dokumentation (2 Exemplare), inkl. je einer beschrifteten CD (plus eine CD für die Abteilung/Archivierung).

Diese definitive Aufgabenstellung, das Vorgehen und die Lieferobjekte wurden in den ersten Wochen nach Arbeitsbeginn gemeinsam verfeinert und finalisiert.

Bewertung

Es gelten die üblichen Regelungen zum Ablauf und zur Bewertung der Bachelorarbeit des Studiengangs Informatik (5 Aspekte zu je 1/6 plus 1/6 mündliche Schlussprüfung) mit besonderem Gewicht auf den Aspekt moderne Softwareentwicklung und eine lauffähige, getestete Software (inkl. Installation).

2 Realisierung

2.1 Vorgehen

Die Umsetzung des Projekts im Rahmen der Bachelorarbeit bestand aus 2 Phasen: Analysephase und Implementierung. Es wurde entschieden, dass die Implementierung mit SCRUM durchgeführt wird.

Zuerst erfolgte eine Einarbeitung in Python und in das Webframework Django. Nebenbei wurden diverse Tools eingerichtet. Es wurde ein virtueller Server (Server4You) für die Entwicklung gemietet und eingerichtet. Weiterhin wurde das Tool Bitbucket für die Versionskontrolle und Jira für das Projektmanagement eingerichtet.

Als nächstes erfolgte die Anforderungsanalyse. Es wurden funktionale und nicht funktionale Anforderungen dokumentiert und Use-Cases formuliert. Mögliche Risiken wurden grob analysiert.

Im nächsten Schritt ging es um die Analyse von brauchbaren Geo-CMS. Es wurden diverse Geodatenbanken, Map-Server und Geo-CMS angeschaut und dokumentiert. Die Funktionalitäten wurden beschrieben und Vor- und Nachteile der Technologien genannt. Es wurden Catara, GeoNode und Mapbender nach verschiedenen Kriterien evaluiert und entschieden welches Geo-CMS sich für das Projekt am Besten eignete. Für das Projekt hat sich GeoNode am besten geeignet, da es die Programmiersprache Python unterstützt und schon einiges an geforderter Funktionalität enthielt. Ein weiterer Grund für die Entscheidung war auch, dass GeoNode aktiv weiterentwickelt wird. GeoNode wurde anschliessend ausführlich beschrieben. Weiterhin wurde die Lizenz (GPLv3) von GeoNode geprüft. Dabei war unklar, ob der Quellcode von GeoNode zusammen mit der im Rahmen der Bachelorarbeit entwickelten Software herausgegeben werden muss. Schlussendlich kam heraus, dass die GPLv3 Lizenz nicht gilt, wenn die Software als Dienst auf einem Server läuft. Für die Übertragung von Geodaten wurden verschiedene Übertragungsformate angeschaut und anhand unterschiedlicher Kriterien verglichen. Dabei wurde festgestellt, dass ShapeFile zwar von GeoNode unterstützt wird, aber zu viele Einschränkungen hat. Es standen unter anderem SpatialLite, JSON und Geopackage zur Auswahl. Es wurde entschieden JSON als Übertragungsformat zu verwenden.

Nach der Technologieanalyse wurde mit der Implementierung begonnen. Die Entwicklung wurde als SCRUM-Prozess durchgeführt. Insgesamt wurde BLUshare in 4 Sprints entwickelt. Jeder Sprint dauerte 2 Wochen mit Ausnahme des zweiten Sprints (3 Wochen). Im ersten Sprint wurde die Benutzerschnittstelle für Datenuploads und -downloads spezifiziert. Weiterhin wurde die IDEs eingerichtet und der Quellcode von GeoNode installiert. Die Installation des Quellcodes hat etwas Zeit in Anspruch genommen, da zwischendurch Probleme auftraten. Weiterhin wurden die Sicherheitsanforderungen erarbeitet. Im anschließenden Sprint wurde die Benutzerverwaltung von GeoNode um das Erfassen von Lese- und Erfassungspereimetern erweitert. Für die Daten der SpatialLite Datenbank wurde eine eindeutige ID (UUID) eingeführt. Im Datenmodell von BLUgis wurden die Bewegungsdaten (Entry, Event und POI) um weitere Eigenschaften erweitert (Teilen via BLUshare und Information, dass die Daten von einem fremden BLUshare Benutzer via BLUshare kommen). Im dritten Sprint wurde BLUgis erweitert mit der Möglichkeit fremde Daten auszublenden. Dabei wurden die Bewegungsdaten mit einer zusätzlichen Eigenschaft «ausgeblendet» erweitert. Es wurde ein Icon (durchgestrichenes Auge) verwendet, um fremde Daten zu kennzeichnen. Die ausgeblendeten Daten wurden in einer Liste gesammelt, damit sie mit Hilfe eines Dialogs angezeigt und wieder eingeblendet werden können. Es stellte sich heraus, dass diese Lösung nicht brauchbar war, weil dadurch nicht ersichtlich war, wo sich dieses Feature befindet. Eine einfachere Lösung war es die Daten auf der Karte auszuwählen und dann zu sehen, dass sie ein- oder ausgeblendet werden können. Mit dieser Lösung wurde das Feature schlussendlich umgesetzt. Bei der Realisierung des Web-Clients wurde darauf Wert gelegt, dass Layers automatisiert in GeoNode und GeoServer hinzugefügt werden.

2.2 Arbeitsaufteilung

Die bevorstehenden Aufgaben wurden laufend im Team aufgeteilt, beispielsweise zu Beginn eines Sprints. Schwerpunktmässig ergab sich folgende Aufteilung:

Martin Eisenhammer:

- Dokumentation des verwendeten Vorgehensmodells
- Dokumentation der Entwicklungsumgebung
- Dokumentation Funktionale Anforderungen
- Dokumentation Use Cases
- Dokumentation Benutzermanagement GeoNode
- Domainanalyse
- Erstellung Datenmodell BLUgis
- Erweiterung GeoNode Datenmodell mit Erfassungssperimeter und Leseperimeter
- Erweiterung Bewegungsdaten und BLUgis UI um Checkbox «Export mit BLUshare» und Label «Erhalten via BLUshare von USER»
- Handling fremder Daten in BLUgis
- Dokumentation vervollständigen

Josua Stähli:

- Einrichten des virtuellen Servers und Jira
- Dokumentation nichtfunktionale Anforderungen und Architektur
- Spezifikation der Schnittstelle für Datenup- und -download
- Bestimmen der Security-Anforderungen
- GUID für Attachments und Daten
- Synchronisation von BLUgis mit BLUshare und Datenup- und -download
- Einbeziehen von Attachments in Up- und Download
- Standard Export nach Excel
- Webclient
- Dokumentation vervollständigen

3 Resultate

3.1 Zielerreichung

Es ist mit BLUshare eine Serverapplikation entstanden, die für Blaulichtorganisationen Geodaten verwaltet. BLUshare ist eine Ergänzung zum bestehenden BLUGis welches sich bereits bei Regio 144 im Einsatz befindet.

Ein Rettungsdienst kann erfasste Geodaten seines Einsatzgebietes benachbarten Rettungsdiensten, die dieses Einsatzgebiet übernehmen müssen, mittels BLUshare zur Verfügung stellen. Dazu wurde diese Funktionalität im bestehenden BLUGis eingebaut. Rettungsdienste können diese Daten von BLUshare auf ihr BLUGis Tablet laden. Bei den Daten handelt es sich beispielsweise um Zufahrten zu Gebäudekomplexen, zu umfahrende Baustellen, gesperrte Zufahrten oder Points of Interest (z.B. Spitäler oder Apotheken).

Ausserdem wurde ein Webclient entwickelt, der die Administration von BLUshare erlaubt, beispielsweise können neue Benutzer angelegt und ihnen Erfassungs- und Leseperimeter zugewiesen werden. Bei diesem Webclient wird GeoNode eingesetzt, ein auf Geodaten spezialisiertes Content Management System, welches umfangreiche Funktionen für die Benutzerverwaltung und zum Bearbeiten von Geodaten mitbringt.

3.2 Bewertung

Die in der Anforderungsanalyse gestellten Anforderungen wurden umgesetzt. Geodaten mit deren Anhängen werden vollständig von BLUGis zu BLUshare übertragen. Dabei wird die Sicherheit gewährleistet. Daten können vollständig von BLUshare zu BLUGis übertragen werden.

GeoNode und GeoServer werden für die Darstellung der Daten im Webclient eingesetzt. GeoNode bietet für die Benutzerverwaltung und für das Editieren der Daten zahlreiche Funktionen. Damit konnte in der Entwicklung viel Zeit gespart werden, da ansonsten noch eine Benutzerverwaltung selber implementiert werden müsste und dies mit viel Aufwand verbunden wäre.

Für die Datenübertragung wird JSON und Zip-Komprimierung genutzt. Damit wird die Dateigrösse klein gehalten und Daten können ohne Probleme client- und serverseitig exportiert werden.

Der Webclient erlaubt zwar das Editieren von Daten, jedoch kann ein Benutzer alle Daten auf einem Layer bearbeiten, nicht nur die eigenen. Deshalb eignet sich der Webclient von BLUshare bisher nur für Administrationaufgaben. Um diesen einem grösseren Benutzerkreis zugänglich machen zu können, müssten die Bearbeitungsrechte feiner gesteuert werden können.

4 Ausblick

4.1 Verwendung

BLUshare kann von Blaulichtorganisationen (Rettungsdienste, Feuerwehr oder Polizei) verwendet werden, um sich in einem fremden Einsatzgebiet Hindernisse (gesperrte Zufahrten, Baustellen, Events...) oder wichtige Orte (Point of Interest) auf einer Karte anzeigen zu lassen. Beispielsweise können Rettungsdienste, die in einem fremden Einsatzgebiet gebraucht werden, sich Informationen über dieses Gebiet mittels BLUshare auf ihre Tablets laden. Weiterhin können Rettungsdienste Daten ihres Einsatzgebietes benachbarten Rettungsdiensten zur Verfügung stellen, indem sie ihre Daten auf BLUshare laden. Damit haben Nachbarrettungsdienste Zugriff auf wichtige Daten, wenn sie in einem anderen Einsatzgebiet unterwegs sind. Weiterhin können Erfassungs- und Leseperimeter für Rettungsdienste festgelegt werden. Dies bedeutet, dass für Rettungsdienste Gebiete festgelegt werden können, wo nur auf Daten zugegriffen werden kann oder ob auch Daten erfasst werden können.

Mit dem Webclient kann zum Beispiel eine Einsatzzentrale, die BLUgis nicht besitzt, Geodaten erfassen und bearbeiten.

4.2 Erweiterungsmöglichkeiten

BLUshare bzw. BLUgis können beispielsweise mit weiteren Funktionen erweitert werden.

- Weitere Benutzerrechte für Admin
- Datenfilter für Synchronisation
- Download des Kartenmaterials von Swisstopo
- Zentrale Baustellenverwaltung durch geeignete Imports
- Weitere Imports, wie zum Beispiel Hydranten
- Versionierung der Geodaten
- Austausch des GeoServer durch QGIS-Server
- GeoSHAPE anstelle von GeoNode einsetzen
- Verwendung von GeoPackage als Übertragungsformat

Diese Punkte werden in späteren Kapiteln noch genauer beschrieben.

Teil II: Projektdokumentation

5 Vision

5.1 Einleitung

BLUshare soll Rettungsdiensten die Möglichkeit geben, wichtige Geodaten für eine rasche Anfahrt von Einsatzorten untereinander auszutauschen. Das Projekt wurde von der Firma honeyconsult markus honegger - Entwickler von BLUgis - in Auftrag gegeben und wird im Rahmen einer Bachelorarbeit an der HSR Rapperswil durchgeführt.

5.2 Ausgangslage

Bereits im Einsatz befindet sich die von der Firma honeyconsult entwickelte Software BLUgis. Diese kommt in Rettungsfahrzeugen in Form von Tablets zum Einsatz, wo sie den Rettungssanitätern Hindernisse wie beispielsweise Baustellen oder Sperrungen durch Veranstaltungen, sowie detaillierte Pläne für die Zufahrt bestimmter Gebäudekomplexe unterwegs anzeigt. Diese Geodaten werden im Stützpunkt vom Rettungsdienst erfasst und anschliessend auf die einzelnen Tablets in den Fahrzeugen verteilt.

Die Einsatzgebiete von Rettungsdiensten können sich überschneiden, insbesondere wenn ein Rettungsdienst gerade voll ausgelastet ist und somit ein anderer einspringen muss. In einem solchen Fall wäre es hilfreich, wenn der Rettungsdienst, welcher in einem «fremden» Einsatzgebiet unterwegs ist, Zugriff auf die für das jeweilige Gebiet erfassten Daten hat. Aktuell besteht keine Möglichkeit, die erfassten Daten einfach untereinander auszutauschen, selbst dann, wenn beide Rettungsdienste mit BLUgis arbeiten.

BLUshare soll genau diese Funktion anbieten, dass Rettungsdienste ihre Daten auf einen Server hochladen und im Gegensatz die Daten benachbarter Einsatzgebiete beziehen können.

5.3 Motivation

Der geplante Geodaten-Austauschserver ermöglicht ein komfortables Teilen von Geodaten zwischen Rettungsdiensten. Damit finden sich Rettungsteams auch dann zurecht, wenn sie in einem fremden Einsatzgebiet unterwegs sind. So wird auch verhindert, dass Daten von unterschiedlichen Rettungsdiensten doppelt erfasst werden.

Es sind auch weitere Einsatzmöglichkeiten denkbar, beispielsweise ein zentraler Import von Baustelleninformationen, welche dann über den BLUshare Server bezogen werden können.

5.4 Ziel

Das primäre Ziel ist die Entwicklung einer Serverapplikation, welche Geodaten verwaltet und eine Schnittstelle bietet um diese zu speichern, abzurufen und zu verändern. Die Zugriffe sollen dabei mit einem Berechtigungskonzept gesteuert werden können und genügend abgesichert sein. Die Funktionalitäten des BLUshare-Servers sollen zudem in das bereits existierende BLUgis integriert werden.

Ein weiteres Ziel ist die Entwicklung eines Webclients, welcher einen Zugriff auf die Daten in BLUshare unabhängig von BLUgis bietet. Der Webclient soll auch das Erfassen neuer Geodaten und die Bearbeitung bestehender Daten ermöglichen, sowie das Exportieren in bestimmte Formate erlauben.

5.5 Beispielszenario

Das folgende Szenario zeigt ein mögliches Beispiel, wo sich der Einsatz von BLUshare als nützlich erweist:

«Frau M. wohnt in einer Alterssiedlung in Uster. Mitten in der Nacht wacht sie mit Atemnot und einem stechenden Schmerz im Brustkorb auf. Mit grosser Mühe schafft sie es noch, per Telefon einen Notruf abzusetzen. Danach legt sie sich erschöpft auf den Boden neben dem Telefon.

Der Rettungsdienst Uster hat in dieser Nacht viel zu tun, alle Fahrzeuge befinden sich bereits im Einsatz, deshalb wird der benachbarte Rettungsdienst Regio 144 aufgeboten. Ein Notarzt und zwei Rettungssanitäter machen sich vom Stützpunkt Rüti aus sofort auf den Weg. Der Beifahrer im Rettungswagen nimmt das BLUgis-Tablet hervor, prüft, ob unterwegs mit Hindernissen zu rechnen ist, und sucht bereits den genauen Einsatzort heraus. Dank BLUshare kann der Beifahrer nun genaue Informationen zu der Alterssiedlung aufrufen, welche vom Rettungsdienst Uster erfasst wurden. Er erfährt, dass das Gebäude nachts verschlossen ist und nur über eine Seitentüre durch die Eingabe eines Zahlencodes zugänglich ist.

15 Minuten nach Erhalt des Notrufes erreicht das Rettungsteam die Alterssiedlung. Dank der Information im BLUgis finden sie sofort die richtige Eingangstüre und können diese mit dem Zahlencode öffnen. Als sie Frau M. in ihrer Wohnung vorfinden ist sie glücklicherweise noch bei Bewusstsein und kann in stabilem Zustand im Spital Uster abgeliefert werden. Wenn die Rettungskräfte für den Zugang in die Alterssiedlung viel Zeit verloren hätten, hätte die Geschichte auch ganz anders ausgehen können!»

6 Anforderungen

In diesem Kapitel geht es um die Anforderungsanalyse. Es werden die funktionalen Anforderungen beschrieben. Anschliessend folgen die Use Cases und die nichtfunktionalen Anforderungen.

6.1 Funktionale Anforderungen

Die funktionalen Anforderungen beschreiben die geforderte Funktionalität von BLUshare.

6.1.1 Persistente Speicherung der Geodaten und deren Anhänge

Um jederzeit Zugriff auf die Daten zu erhalten sollen diese mit Anhängen auf dem Server gespeichert werden.

6.1.2 Datenaustausch zwischen BLUgis und BLUshare

Es soll möglich sein, Daten zwischen BLUgis und BLUshare auszutauschen und abzugleichen.

Upload von zur Freigabe markierten Bewegungsdaten von BLUgis zu BLUshare

Bewegungsdaten (in BLUgis Events, Zufahrten und POI), die zur Freigabe markiert wurden, sollen von einem System mit BLUgis auf BLUshare hochgeladen werden können.

Download von BLUshare zu BLUgis

Daten sollen von BLUshare auf das BLUgis geladen werden können. Dabei soll geachtet werden, dass bestimmte Daten nur von bestimmten Benutzern bezogen werden können.

6.1.3 Darstellung und Veränderung von Daten mit einem Webclient

Wenn kein BLUgis vorhanden ist, sollen Daten über einen Webclient dargestellt und bearbeitet werden. Nur Daten im Besitz des Benutzers sollen dargestellt und bearbeitet werden. Es soll ein Übersichtsplan bzw. Landkarten als Hintergrund verwendet werden.

6.1.4 Export von Daten aus dem Webclient

Die Daten (insbesondere die POIs) sollen aus dem Webclient in ein geeignetes Format wie Excel oder csv exportiert werden können.

6.1.5 Editieren von Daten im Webclient (optional)

Optional können Daten im Webclient bearbeitet werden.

6.2 Use Cases

6.2.1 Übersicht

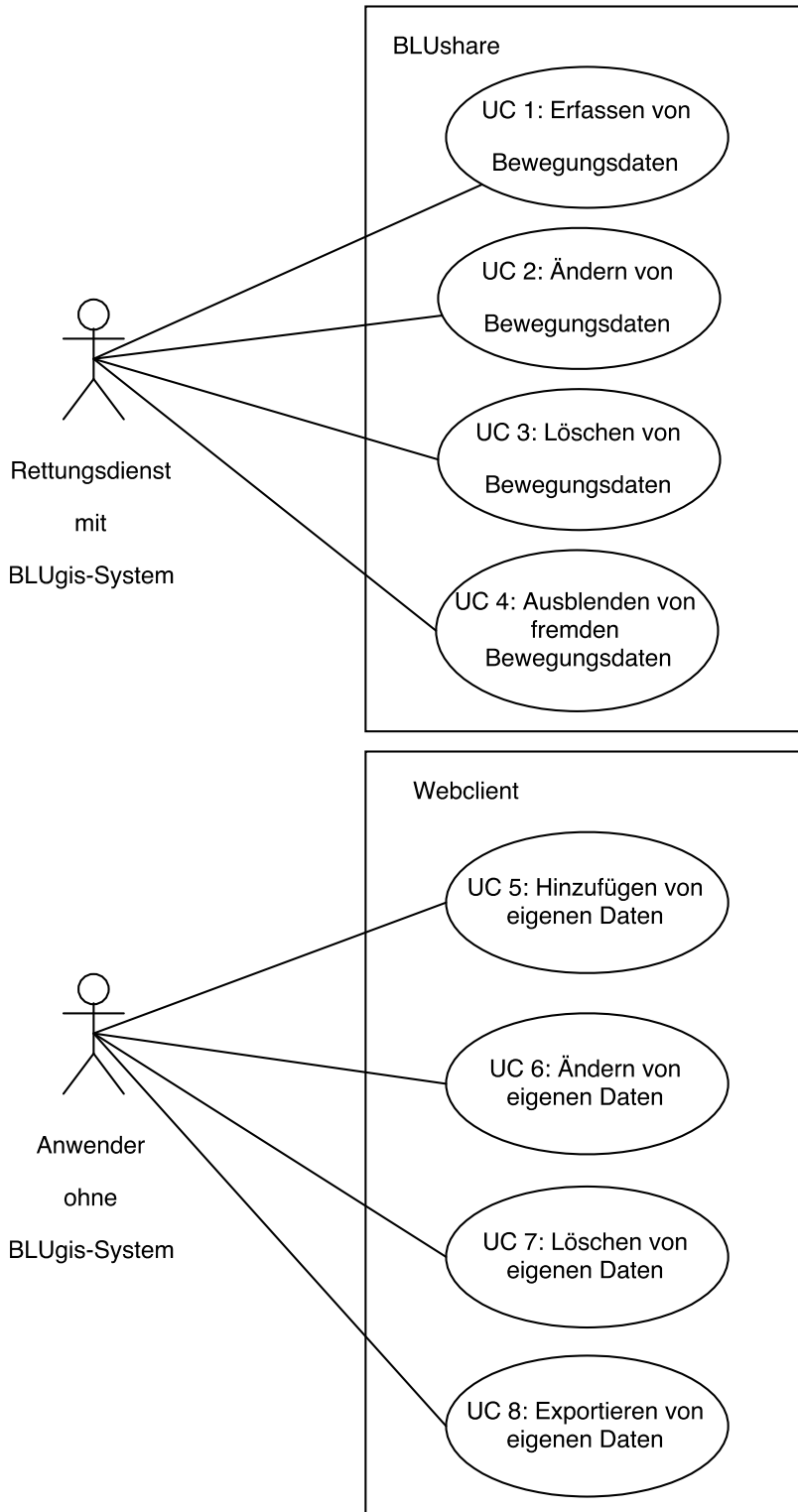


Abbildung 4: Use-Case-Diagramm

6.2.2 Akteure und Stakeholder

Ein Mitarbeiter im Rettungsdienst benutzt ein Tablet mit BLUgis und möchte Events, Zufahrten und POIs mittels BLUshare anderen Rettungsdiensten zur Verfügung stellen.

Die Einsatzzentrale möchte mit Hilfe eines Webclients auf Events, Zufahrten und POIs zugreifen ohne das System BLUgis besitzen zu müssen.

6.2.3 Beschreibung

UC 1: Erfassen von Bewegungsdaten

Name	Erfassung von Bewegungsdaten
Umfang	BLUshare, BLUgis
Ebene	BLUgis
Primär-Actor	Mitarbeiter im Rettungsdienst
Stakeholder und Interessen	Rettungsdienst möchte Bewegungsdaten, wie Events, Zufahrten und POI anderen Rettungsdiensten zur Verfügung stellen.
Vorbedingungen	Rettungsdienst muss Zugriff auf BLUshare haben.
Nachbedingungen	<ul style="list-style-type: none"> • Bewegungsdaten in BLUshare erfasst • Andere Rettungsdienste können darauf zugreifen
Standard-Szenario	<ol style="list-style-type: none"> 1. Der Benutzer gibt die Daten ins BLUgis ein. 2. Die Daten werden ins BLUshare geladen und gespeichert.
Erweiterungen	keine
Spezielle Anforderungen	Rettungsdienst darf Daten nur in seinem zugewiesenen Gebiet erfassen.
Technik und Datenvariationen	<ul style="list-style-type: none"> • BLUgis Tablet • Daten: Events (z.B. Baustellen), Zufahrten, POIs
Auftreten	mindestens einmal täglich
Verschiedenes	-

UC 2: Ändern von Bewegungsdaten

Name	Ändern von Bewegungsdaten
Umfang	BLUshare, BLUgis
Ebene	BLUgis
Primär-Actor	Mitarbeiter im Rettungsdienst
Stakeholder und Interessen	Rettungsdienst möchte ein Event, Zufahrt oder POI ändern
Vorbedingungen	Event, Zufahrt oder POI muss in BLUshare vorhanden sein.
Nachbedingungen	Event, Zufahrt oder POI im BLUshare wurde aktualisiert und steht anderen Rettungsdiensten zur Verfügung.
Standard-Szenario	<ol style="list-style-type: none"> 1. Benutzer klickt auf das Event, Zufahrt oder POI um es zu bearbeiten. 2. Es öffnet sich ein Detailfenster und der Benutzer macht seine Änderungen. 3. Die Änderungen werden gespeichert.
Erweiterungen	keine
Spezielle Anforderungen	Rettungsdienst darf Daten nur in seinem zugewiesenen Gebiet ändern.
Technik und Datenvariationen	<ul style="list-style-type: none"> • BLUgis Tablet • Daten: Events (z.B. Baustellen), Zufahrten, POIs
Auftreten	Ein- bis mehrmals täglich
Verschiedenes	-

UC 3: Löschen von Bewegungsdaten

Name	Löschen von Bewegungsdaten
Umfang	BLUshare, BLUgis
Ebene	BLUgis
Primär-Actor	Mitarbeiter im Rettungsdienst
Stakeholder und Interessen	Rettungsdienst möchte eine Zufahrt, ein Event oder ein POI löschen.

Vorbedingungen	Zufahrt, Event oder POI muss in BLUshare vorhanden sein.
Nachbedingungen	Zufahrt, Event oder POI wurde gelöscht.
Standard-Szenario	<ol style="list-style-type: none"> 1. Der Benutzer klickt auf das Event, Zufahrt oder POI. 2. Der Benutzer klickt auf «Löschen». 3. Die ausgewählten Daten werden gelöscht.
Erweiterungen	keine
Spezielle Anforderungen	Rettungsdienst darf Daten nur in seinem zugewiesenen Gebiet löschen.
Technik und Datenvariationen	<ul style="list-style-type: none"> • BLUgis Tablet • Daten: Event, Zufahrten und POIs
Auftreten	Ein- bis mehrmals täglich
Verschiedenes	-

UC 4: Ausblenden von fremden Bewegungsdaten

Name	Ausblenden von fremden Bewegungsdaten
Umfang	BLUshare, BLUgis
Ebene	BLUgis
Primär-Actor	Mitarbeiter im Rettungsdienst
Stakeholder und Interessen	Rettungsdienst möchte fremde Bewegungsdaten ausblenden
Vorbedingungen	Bewegungsdaten von einem fremden Rettungsdienst via BLUshare erhalten.
Nachbedingungen	Fremde Bewegungsdaten sind ausgeblendet
Standard-Szenario	<ol style="list-style-type: none"> 1. Der Benutzer klickt auf ein Icon, um die Daten auszublenden. 2. Die Daten werden ausgeblendet.
Erweiterungen	keine
Spezielle Anforderungen	keine
Technik und Datenvariationen	<ul style="list-style-type: none"> • BLUgis Tablet • Fremde Daten via BLUshare erhalten

Auftreten	Ein- bis mehrmals täglich
Verschiedenes	-

UC 5: Hinzufügen von eigenen Daten im Webclient (optional)

Name	Hinzufügen von eigenen Daten
Umfang	BLUshare
Ebene	BLUshare, Webclient
Primär-Actor	Benutzer ohne BLUgis
Stakeholder und Interessen	Benutzer ohne BLUgis möchte weitere Geodaten hinzufügen.
Vorbedingungen	Benutzer muss im Webclient eingeloggt sein.
Nachbedingungen	Geodaten wurden hinzugefügt.
Standard-Szenario	<ol style="list-style-type: none"> 1. Der Benutzer klickt im Webclient auf «Daten hinzufügen». 2. Der Benutzer gibt die Daten in eine Eingabemaske ein. 3. Der Benutzer klickt auf «Speichern». 4. Die Daten werden in BLUshare gespeichert.
Erweiterungen	keine
Spezielle Anforderungen	keine
Technik und Datenvariationen	
Auftreten	Ein- bis mehrmals täglich
Verschiedenes	-

UC 6: Ändern von eigenen Daten im Webclient (optional)

Name	Ändern von eigenen Daten im Webclient
Umfang	BLUshare
Ebene	BLUshare, Webclient
Primär-Actor	Benutzer ohne BLUgis
Stakeholder und Interessen	Benutzer ohne BLUgis möchte Geodaten ändern.

Vorbedingungen	<ul style="list-style-type: none"> Benutzer muss im Webclient eingeloggt sein Zu ändernde Geodaten müssen im BLU-share vorhanden sein
Nachbedingungen	Die aktualisierten Geodaten stehen zur Verfügung.
Standard-Szenario	<ol style="list-style-type: none"> Der Benutzer lässt die zu verändernden Geodaten im Webclient anzeigen. Der Benutzer klickt auf «Bearbeiten». Der Benutzer gibt die Änderungen in eine Eingabemaske ein. Der Benutzer klickt auf «Speichern». Die veränderten Daten werden im BLU-share gespeichert.
Erweiterungen	keine
Spezielle Anforderungen	keine
Technik und Datenvariationen	keine
Auftreten	Ein bis mehrmals täglich
Verschiedenes	-

UC 7: Löschen von eigenen Daten im Webclient (optional)

Name	Löschen von eigenen Daten im Webclient
Umfang	BLUshare
Ebene	BLUshare, Webclient
Primär-Actor	Benutzer ohne BLUgis
Stakeholder und Interessen	Benutzer ohne BLUgis möchte eigene Daten löschen.
Vorbedingungen	<ul style="list-style-type: none"> Benutzer muss im Webclient eingeloggt sein. Zu löschende Geodaten müssen in BLU-share vorhanden sein.
Nachbedingungen	Geodaten sind gelöscht
Standard-Szenario	<ol style="list-style-type: none"> Der Benutzer lässt die zu löschenden Geodaten im Webclient anzeigen. Der Benutzer klickt auf «Löschen». Die Geodaten werden gelöscht.

Erweiterungen	keine
Spezielle Anforderungen	keine
Technik und Datenvariationen	keine
Auftreten	Ein bis mehrmals täglich
Verschiedenes	-

UC 8: Exportieren von Daten als csv aus dem Webclient

Name	Exportieren von Daten als csv
Umfang	BLUshare
Ebene	Webclient
Primär-Actor	Benutzer ohne BLUgis
Stakeholder und Interessen	Benutzer ohne BLUgis möchte Geodaten als csv exportieren.
Vorbedingungen	<ul style="list-style-type: none"> ● Benutzer muss im Webclient eingeloggt sein. ● Zu exportierende Geodaten müssen im BLUshare vorhanden sein.
Nachbedingungen	Geodaten wurden als csv exportiert.
Standard-Szenario	<ol style="list-style-type: none"> 1. Der Benutzer lässt zu exportierende Geodaten im Webclient anzeigen. 2. Der Benutzer klickt auf «Exportieren». 3. Der Benutzer wählt einen Speicherort aus. 4. Die Daten werden als csv exportiert.
Erweiterungen	keine
Spezielle Anforderungen	keine
Technik und Datenvariationen	Export der Daten als csv
Auftreten	Ein bis mehrmals täglich
Verschiedenes	-

6.3 Nichtfunktionale Anforderungen

Die nichtfunktionalen Anforderungen beschreiben Nebenbedingungen sowie Anforderungen an die Qualität, mit welchen die geforderte Funktionalität umzusetzen ist.

6.3.1 Allgemein

Um einen langfristig stabilen Betrieb zu ermöglichen, sollen wo möglich auf etablierte Technologien und Standards gesetzt werden. Bevorzugt sind Open Source Lösungen.

6.3.2 Technologien

NFA 1: Programmiersprache

Weil das schon existierende Projekt BLUgis in Python umgesetzt ist und im Entwicklungsteam bereits einiges an Vorwissen in dieser Programmiersprache vorhanden ist, wird Python als primäre Implementationssprache für BLUshare bevorzugt. Damit kann unter Umständen Know-how aus dem BLUgis Projekt einfacher in diesem Projekt wiederverwendet werden.

Falls sich in der Evaluationsphase ein GeoCMS als geeignet erweist, welches auf einer anderen Programmiersprache basiert, kann auch eine andere Programmiersprache als die Primäre genommen werden.

NFA 2: Webframework

Django ist ein bewährtes Webframework, welches auf Python basiert. Im Team ist zudem schon Vorwissen zu diesem Framework vorhanden. Deshalb soll Django bevorzugt verwendet werden, es sei denn, ein anderes Framework wird sich im Verlauf der Arbeit als geeigneter herausstellen.

NFA 3: Datenbank

Als Datenbank ist PostgreSQL bzw. deren Erweiterung PostGIS vorgesehen.

NFA 4: Webtechnologien

Für die Umsetzung des Webclients sollen die aktuellen Webstandards HTML5, CSS3 und JavaScript zum Einsatz kommen.

NFA 5: Browserunterstützung

Der Webclient soll auf den während der Umsetzung aktuellen Versionen von Google Chrome, Mozilla Firefox, Microsoft Internet Explorer und Microsoft Edge lauffähig sein. Es handelt sich um folgende Versionen: Chrome 51, Firefox 46, Internet Explorer 11 und Edge 25.

Der Support für älterer Browser ist wünschenswert, jedoch nicht erforderlich.

Mobile Browser müssen nicht unterstützt werden, weil der Webclient primär mit einem Desktop PC verwendet wird.

NFA 6: Schnittstelle

Der Aufruf und das Abspeichern von Geodaten geschieht über eine API, welche als REST Webservice umgesetzt werden soll.

NFA 7: Karten

Für die Kartendarstellung im Webclient soll OpenLayers oder Leaflet zum Einsatz kommen, welche als bewährte Libraries gelten.

Die primäre Kartenquelle ist die Landeskarte der Swisstopo, als Alternative kann auch OpenStreetMap zum Einsatz kommen.

6.3.3 Benutzbarkeit

NFA 8: Bedienung

Beim Webclient soll grosser Wert auf eine einfache Bedienung gelegt werden, damit sich technisch weniger versierte Benutzer schnell damit zurechtfinden.

NFA 9: Installation

Für die Installation der Serverkomponenten soll eine Anleitung erstellt werden.

6.3.4 Wartbarkeit und Änderbarkeit

Nach Möglichkeit soll sichergestellt werden, dass sich das finale Produkt, insbesondere der Webclient, einfach in andere Sprachen übersetzen lässt. Zunächst kommt aber nur Deutsch zum Einsatz.

6.3.5 Sicherheitsanforderungen

Es gelten die folgenden allgemeinen Schutzziele aus der Informationssicherheit:

- **Vertraulichkeit (Confidentiality):** Die auf dem BLUshare Server abgelegten Daten sowie die Übertragung zwischen BLUshare und BLUgis sollen vor nicht autorisiertem Mitlesen geschützt werden.
- **Datenintegrität (Integrity):** Übertragene Daten dürfen unterwegs zwischen Client und Server nicht unbemerkt verändert werden.
- **Verfügbarkeit (Availability):** Systemausfälle sollen nach Möglichkeit verhindert werden.

Die Vertraulichkeit ist wichtig, weil die übertragenen Daten Personendaten und Informationen zu geheimen Gebäudezugängen enthalten können.

Eine ständige Verfügbarkeit des Servers ist zwar wichtig aber nicht kritisch, weil eine Blaulichtorganisation mit den offline verfügbaren Daten arbeiten kann, ohne dass eine Verbindung zum BLUshare-Server bestehen muss.

NFA 10: Verschlüsselung

Für die sichere Übertragung der Daten soll TLS (Transport Layer Security) als Datenübertragungsprotokoll eingesetzt werden. SSL sowie offiziell als unsicher geltende Chiffrensammlungen sollen serverseitig deaktiviert werden, wenn dadurch keine der zu unterstützenden Browserversionen (siehe NFA 5) von der Benutzung ausgeschlossen werden.

NFA 11: Benutzeridentifikation

Auf die Plattform dürfen nur authentifizierte Benutzer zugreifen.

Für alle gespeicherten Geodaten muss bekannt sein, von welchem Benutzer diese erstellt wurden.

NFA 12: Rechte

Für Geodaten soll auf Layerebene bestimmt werden können, welche Benutzer die Daten im Layer bearbeiten dürfen, nur lesen können oder gar keinen Zugriff darauf haben.

Einem Benutzer kann ein Erfassungs- und Leseperimeter in Form eines Polygons zugewiesen werden. Hochgeladene Daten, die sich ausserhalb vom Erfassungsperimeter eines Benutzers befinden sollen nicht im Server abgelegt werden. Zudem erhält der Benutzer bei einem Datendownload nur diejenigen Daten, welche sich innerhalb seines Leseperimeters befinden. Ein Benutzer darf nur die eigenen Daten bearbeiten.

NFA 13: Datenvandalismus

Es sollen geeignete Massnahmen vorgeschlagen werden, welche den Datenvandalismus durch unberechtigte Personen verhindern oder den daraus einhergehenden Schaden eindämmen.

NFA 14: SQL-Injections

SQL-Injections sollen mit geeigneten technologischen Massnahmen verhindert werden.

6.3.6 Mengenanforderungen

NFA 15: Speicherplatz

Auf dem Server sollen zum jetzigen Zeitpunkt für die BLUshare Software und die Geodaten (Vektordaten) mindestens 10 Gigabyte verfügbar sein.

NFA 16: Parallele Zugriffe

Da die Synchronisation mit BLUshare pro Client nur sporadisch stattfindet (z.B. alle zwei Stunden), sind parallele Synchronisationsvorgänge eher selten. Trotzdem soll der Server mit drei parallel stattfindenden Synchronisationen umgehen können, ohne dass ein Fehler auftritt und die Verarbeitungszeit darf sich dabei maximal verdoppeln.

6.3.7 Qualitätsanforderungen

Wo sinnvoll sollen automatisierte Tests zum Einsatz kommen. Ist das nicht möglich oder zu aufwendig, soll auf manuelle Tests gesetzt werden.

Bei der Programmierung in Python soll sich der Programmierstil an PEP 8 [1] orientieren.

7 Domainanalyse

In diesem Kapitel wird das in BLUgis bestehende Datenmodell beschrieben, sowie die Änderungen und Erweiterungen, die daran vorgenommen wurden. Ausserdem wird die Datenstruktur eines Layers in GeoNode beschrieben.

7.1 Datenmodell BLUgis (SQLite)

In BLUgis sind Adresdaten, Zufahrten, Events und POIs vorhanden. Diese Daten sind jeweils in einer sqlite-Datenbankdatei gespeichert.

In jeder sqlite-Datei gibt es eine Tabelle BLUGIS mit den Attributen key und value, welche das Abspeichern von sogenannten Properties ermöglicht. Aktuell ist die Property «schema_version» in Verwendung, welche zur Festlegung der Version des Datenschemas dient und so eine versionsabhängige Migration des Datenmodells erlaubt.

7.1.1 Bestehendes Datenmodell

In BLUgis werden die Daten in SpatiaLite-Datenbanken gespeichert, welche sich im Unterordner gisdata/spatialite befinden. Jede Tabelle befindet sich in einer sqlite-Datei.

Zu den Tabellen gibt es Indextabellen. (idx_..._GEOMETRY + _node/_parent/_rowid).

adr	EVENTS	poi
PK OGC_FID	PK pkuid	PK OGC_FID
GEOMETRY	geometrie	GEOMETRY
adr_street	event_file	poi_file
adr_number	event_name	poi_name
adr_postcode	event_type	poi_phone
adr_city	event_from_date	poi_type
adr_canton	event_from_time	poi_decription
adr_source	event_to_date	poi_record_source
	event_to_time	poi_record_by
	event_description	poi_record_date
	event_is_repeating	
	event_trobbdb_Export	
	event_record_by	
	event_record_date	

ZUFAHRT
PK pkuid
geometrie
entry_name
entry_type
entry_description
entry_record_by
entry_record_date

Abbildung 5: Datenmodell von BLUgis vor den Anpassungen

7.1.2 Anpassung des Datenmodells

Eindeutige ID

Die von QGIS verwendeten ID ist nur pro Layer bzw. Datenbanktabelle eindeutig. Darum soll eine UUID (Universally Unique Identifier) als ID eingeführt werden, welche über alle BLUshare-Benutzer hinweg eindeutig ist. Die UUID soll rein zufällig generiert werden (ohne Einbezug von Hardwareadressen), um Datenschutzbedenken vorzubeugen.

Eine UUID hat einen höheren Speicherplatzbedarf (128bit) [2] [3] beispielsweise eine Increment-ID, aber weil die Anzahl der Daten überschaubar bleibt (manuell erfasste Daten) stellt das kein Problem dar.

Attachments

Attachments konnten ursprünglich Events und POIs angefügt werden. Diese Funktion wurde nun auch für Zufahrten erweitert.

BLUshare

Für die Synchronisation zwischen BLUgis und BLUshare sind neben der eindeutigen ID (siehe oben) noch folgende Attribute hinzugefügt worden:

- boolean-Flag, welches anzeigt ob Daten auf den BLUshare hochgeladen werden sollen oder nicht.
- Ein Feld, welches den BLUshare-Benutzer angibt, von dem ein Objekt stammt. Lokal ist das Feld auf null gesetzt, falls das Objekt von dem aktuellen Benutzer selber stammt.
- Ein Timestamp, welcher auf die Zeit der letzten Bearbeitung eines Objektes gesetzt ist

Ausblenden

Daten, die von einem fremden User stammen, können auch ausgeblendet werden. Dazu braucht es ein boolean-Flag, welches anzeigt, in welchem Zustand sich das Objekt befindet. Über eine Filterfunktion innerhalb von QGIS werden die versteckten Elemente herausortiert.

Vereinheitlichungen

Um die Konsistenz des Datenmodells zu erhöhen, wurden einige Felder umbenannt.

- Für den Primary Key wird einheitlich der Name «pkuid» verwendet (Umbenennung von «OGC_FID»)
- Nur Verwendung von englischer Sprache (Umbenennung «geometrie»)
- Kleinschreibung von Attributen (Umbenennung GEOMETRY)
- Grossschreibung der Namen der eigenen Tabellen (um die selber erstellten Tabellen von den mit SpatiaLite automatisch erzeugten Tabellen abzuheben)

7.1.3 Neues Datenmodell

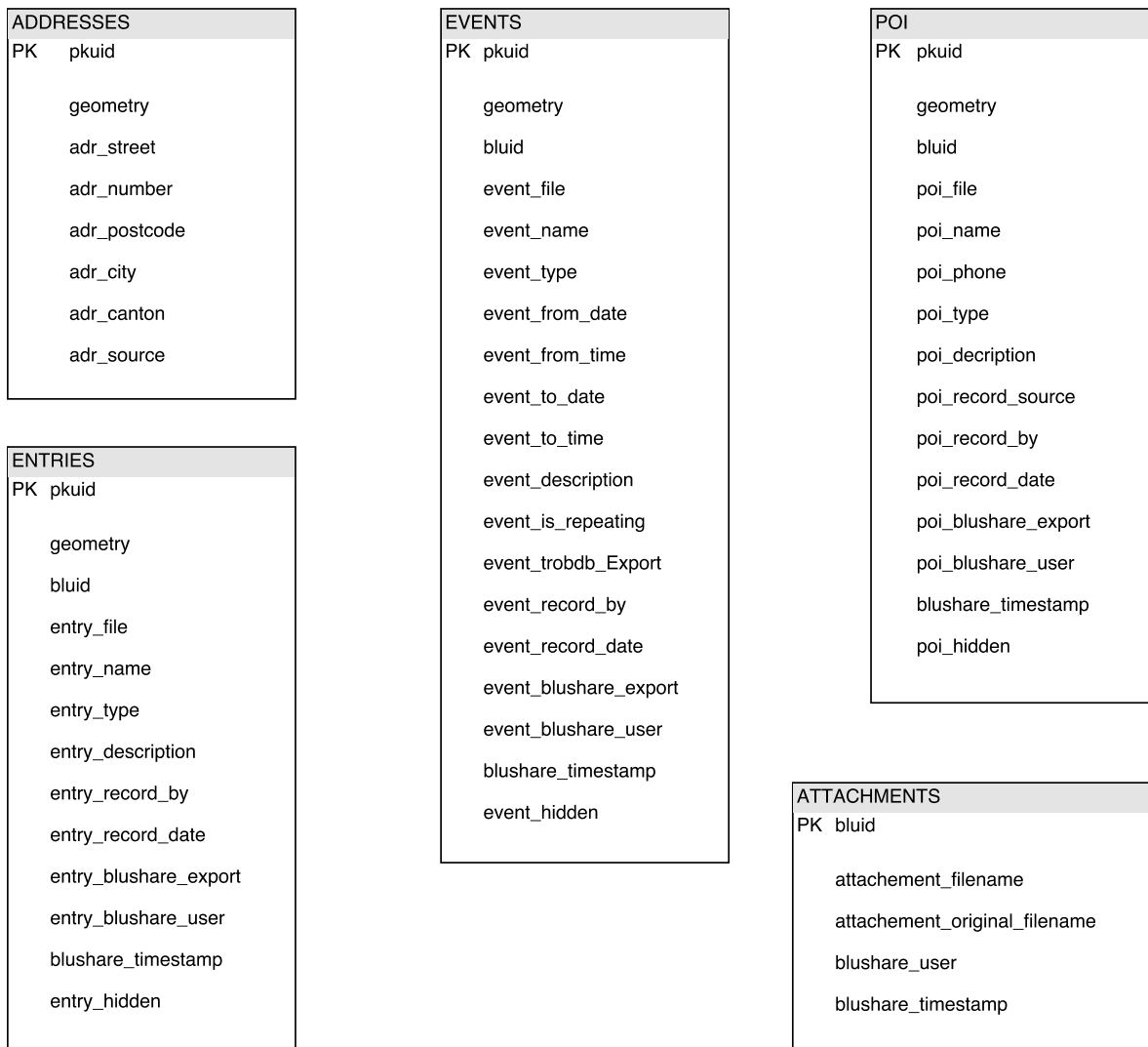


Abbildung 6: Verändertes Datenmodell von BLUgis

7.2 Datenmodell BLUshare (PostgreSQL)

Serverseitig werden die Daten in einer PostgreSQL-Datenbank abgelegt. Zusätzlich kommt PostGIS für die Speicherung der Geometrien zum Einsatz. Es wird das gleiche Datenmodell wie auf der Clientseite verwendet, wobei die pkuid nicht vorhanden ist. Diese ID wird lediglich von QGIS benötigt, serverseitig dient die bluid als Primary Key.

7.3 Datenstrukturen GeoNode (Django)

7.3.1 Layer

Layers können in Geonode Attribute und Styles haben. Attribute haben einen Namen, Beschreibung und einen Datentyp.

Die Klassen zur Verwaltung eines Layers befinden sich im Package `geonode.layers`.

Hier gibt es eine Klasse `Layer`, die Layers verwaltet. Weiterhin gibt es eine Klasse für Attribute. Diese Klasse bietet Funktionalität zum Anpassen von Attributen, Sortierungsreihenfolge und Sichtbarkeit.

7.3.2 Attachments

Die Attachments zu den Geodaten können als Attribute gesetzt werden.

In `geonode.layers.utils` und `geonode.layers.views` gibt es Methoden zum Upload eines Files.

In `geonode.layers.utils` gibt es eine Methode `file_upload`, die einen Dateinamen erwartet.

Mit einer anderen Methode (`upload`) können Verzeichnisse mit Geodaten hochgeladen werden. Es werden `.shp`, `.tif`, `.tar`, `.tar.gz` und `.zip`-Dateien unterstützt.

7.3.3 Erweiterung der GroupProfile-Datenstruktur

Im Quellcode von `GeoNode` (`geonode.groups/models.py`) muss die Klasse `GroupProfile` um folgende Eigenschaften erweitert werden:

- `region_with_create_permission`
- `region_with_read_permission`

Diese Eigenschaften ermöglichen es, dass festgelegt werden kann, in welchem Gebiet Daten erfasst werden dürfen (Erfassungssperimeter) und in welchem Gebiet Daten nur gelesen werden können (Leseperimeter).

8 Technologieanalyse

In diesem Kapitel werden verschiedene Technologien angeschaut und gegebenenfalls evaluiert, welche für das Projekt nützlich sein könnten.

8.1 Geodatenbanken

Unter einer Geodatenbank wird hier eine Datenbank verstanden, welche Geodaten in Form von Punkten, Linien und Polygonen speichern und effizient verwalten kann. Es gibt eine grosse Anzahl von Datenbanken mit Unterstützung von Geodaten. [4] Durch die Verwendung bestimmter Technologien wie MapServer oder Geo-CMS kann eine Datenbank bereits vorgeschrieben sein bzw. die Auswahl eingeschränkt. Deshalb werden hier lediglich Erweiterung zu Datenbanken angeschaut, welche etabliert und im Team bereits bekannt sind.

8.1.1 PostGIS

PostGIS ist eine Erweiterung für die objektrelationale Datenbank PostgreSQL. Das Projekt ist Open Source und steht unter der GPLv2 Lizenz. PostGIS unterstützt räumliche Indizierung sowie verschiedene räumliche Operatoren. [5]

8.1.2 Oracle Spatial and Graph

Oracle Spatial and Graph ist eine Komponente der Oracle Datenbank und steht damit unter einer kommerziellen Lizenz. Diese Datenbankeerweiterung erlaubt die Verwaltung und Analyse von Geodaten. [6]

8.1.3 SpatiaLite

SpatiaLite erweitert die SQLite Datenbank um Geodatenobjekte und geografische Funktionen. [7]

8.2 Mapserver

Als Mapserver wird eine Server-Software bezeichnet, welche Karten oder andere Geodaten bereitstellen, meistens in Form eines Webservices.

8.2.1 ArcGIS Server

ArcGIS Server ist kommerziell und wurde von der Firma Esri entwickelt. [8]

8.2.2 GeoServer

GeoServer ist Open Source und basiert auf Java. GeoServer ist OGC konform (WFS, WMS, ...). [9]

8.2.3 QGIS Server

QGIS Server ist Open Source und benutzt dieselben Libraries wie die QGIS Desktop Applikation. [10]

8.2.4 UMN Mapserver

UMN Mapserver ist Open Source und basiert auf C/C++. [11]

8.3 Geodaten CMS

Im Folgenden soll geprüft werden, ob für die geplante Geodaten-Austauschplattform bereits Applikationen existieren, welche die geplanten Funktionen teilweise oder ganz abdecken. Damit würde einiges an Zeit eingespart und die vorhandenen Ressourcen könnten anderweitig eingesetzt werden.

Anforderungen an CMS:

- Speicherung von Geodaten (Punkte, Linien und Polygone)
- Gute Erweiterbarkeit und Modifizierbarkeit oder gute Integrierbarkeit in andere Applikationen
- Verwendung etablierter Technologien
- OGC konform (WMS/WFS)
- Geeignete Lizenz
- Aktive Weiterentwicklung

8.3.1 Cartaro

Cartaro ist ein CMS für die Verwaltung von Geodaten. Diese Funktionalität ist im Drupal CMS integriert worden. Die ganze Infrastruktur von Drupal inklusive der Rechteverwaltung stehen darum auch zur Verfügung. Cartaro verwendet PostGIS, GeoServer, GeoWebCache und OpenLayers (sowie OpenLayers Editor). [12]

Funktionen: [13]

- Speichern von Geodaten
- Erstellen und verändern von Geodaten
- Visualisierung der Daten auf einer Karte
- Konform zu OGC Standards (WMS, WFS)
- Sicherheits- und Rechtesystem (von Drupal)
- Einfache Installation (wie Drupal)

8.3.2 GeoNode

GeoNode ist eine Webapplikation für das Erstellen von Geoinformationssystemen. Diese Applikation basiert auf Django und ist auf Erweiterbarkeit und Modifizierbarkeit ausgelegt. GeoNode kann in existierende Plattformen integriert werden. [14]

Funktionen: [15] [16] [17]

- Verwaltung von Rasterdaten, Vektordaten, Tabellen und Dokumenten
- Öffentliches oder geschütztes Teilen der Daten
- Rechtesteuerung auf Layer- und Objektebene
- Unterstützung von WMS, WFS, ...
- Geodaten-Editor mit Versionisierung
- Karten mit mehreren Layern
- Suchfunktion
- Administrationskonsole (Django)

GeoNode basiert auf Django-Apps wie avatar, dialogos, django_notification, django_pagination, django_taggit. [18]

8.3.3 GeoSHAPE

GeoSHAPE basiert auf GeoNode mit GeoServer, PostGIS und GeoGit. Zusätzlich bietet GeoSHAPE mit MapLoom einen verbesserten, auf OpenLayers 3 basierenden Map Viewer und Editor. Mit Arbiter steht zudem eine mobile App zur Verfügung, welche unterwegs das Sammeln und Bearbeiten von Daten ermöglicht. [19]

Vergleich mit GeoNode:

GeoSHAPE Vorteile:

- Arbeitet mit GeoGit → Versionisierung
- Mobile App
- Verbessertes GUI für Karteneditierung
- Karte unterstützt Touch-Gesten

GeoSHAPE Nachteile:

- Angewiesen, dass neue Versionen von GeoNode auch in GeoSHAPE übernommen werden
- Angewiesen, dass GeoSHAPE aktiv weiterentwickelt wird
- Zusätzliche Komplexität durch grösseren Funktionsumfang

8.3.4 Mapbender

Mapbender ist ein CMS für die Anzeige und Abfrage von Karten und Geodaten. Als Implementationssprachen werden PHP und JavaScript verwendet. Durch ein Authentifizierungs- und Autorisierungssystem lassen sich die Daten und Dienste gezielt für einzelne Benutzer freigeben. Die Weboberfläche lässt sich den eigenen Bedürfnissen anpassen bzw. durch eine eigene ersetzen. [20] Mapbender unterstützt auch WMS. [21]

Funktionen:

- Anzeige und Abfrage von Geodaten
- Authentifizierung, Autorisierung und Zugriffsprotokollierung
- Druckfunktion und Bildexport

8.3.5 OpenGeo Suite

OpenGeo Suite wird von Boundless betrieben und benutzt verschiedene Open Source Projekte, unter anderem auch GeoNode. [22] Der Download ist nach Angabe von persönlichen Daten möglich. [23] Dieser Ansatz wird deshalb nicht weiterverfolgt, weil auch andere, öffentlich zugängliche Projekte auf GeoNode basieren und GeoNode alleine verwendet werden kann.

8.3.6 Weitere Optionen

Die folgenden CMS und Frameworks, welche die Verarbeitung von Geodaten unterstützen, wurden während der Evaluationsphase gefunden, aber nicht weiterverfolgt:

- Degree [24]
- Geomajas [25]
- MapFish [26]
- GeoDjango [27]
- MapIgniter2 [28]

Weiter wurde die Möglichkeit in Betracht gezogen, GeoNode durch einen spezialisierten Dienstleister hosten zu lassen, beispielsweise von MapServerPro [29]. Ein GeoNode-Hosting ist da ab 100 Dollar pro Monat erhältlich. Während der Entwicklung würde sich diese Investition aber nicht lohnen, insbesondere, weil so vermutlich

auch die Flexibilität geringer wird als bei einem eigenen Hosting. Im Produktiveinsatz kann diese Option aber nochmals geprüft werden.

8.3.7 Vergleich

	Cartaro	GeoNode	Mapbender
Primäre Sprache	PHP	Python	PHP
Basiert auf	Drupal, GeoServer, GeoWebCache, Proj4js, OpenLayers Editor	Django, GeoServer, GeoGig, GDAL, pycsw, GeoExt	Symfony2, jQuery
Map Library	OpenLayers	OpenLayers	OpenLayers
Datenbank	PostGIS	PostGIS	SQLite (andere konfigurierbar)
Editierfunktionen	Ja	Ja	Nein
Rechteverwaltung	Ja	Ja	Ja
WMS, WFS	Ja	Ja	Ja (WMS)
Geodaten Editor	Ja	Ja	Ja
Suchfunktion	Ja (Filter)	Ja	Nein
Dokumentverwaltung	Ja	Ja	Nein
Versionisierung	Optional aktivierbar (Drupal Revisions)	Ja	Nein
Clustering	Ja	Ja	Ja
Styling	Ja (SLD)	Ja (SLD)	Nein
Druckfunktion	Ja	Ja	Ja
Export	Nein	12 Formate	JSON, YAML
Lizenz	Unbekannt (Drupal: GNU General Public License Version 2)	GNU General Public License Version 3	MIT
Datum letzter Release	29.10.2015	19.11.2015	04.02.2016

8.3.8 Fazit

Bei Drupal sind vor allem fehlende Exportfunktionen der Geodaten und bei Mapbender die fehlende Editierfunktion und fehlendes Styling der Daten via Webclient negativ aufgefallen, wodurch die Entscheidung gegen diese beiden Geo-CMS gefallen ist.

GeoNode hat sich wegen dem Funktionsumfang am vielversprechendsten herausgestellt. Nützlich ist beispielsweise die integrierte Editierfunktion oder die umfangreichen Möglichkeiten zur Rechtevergabe. Zudem steht hinter GeoNode eine aktive Community, welche das Projekt aktiv weiterentwickelt. Als Technologien werden Python und Django verwendet, welche gerade den für diese Bachelorarbeit bevorzugten Technologien entsprechen (siehe Kapitel Anforderungsanalyse).

Abzuklären bleibt, ob GeoNode in der Standardkonfiguration verwendet werden soll oder beispielsweise der GeoServer-Unterbau durch den QGIS Server mit akzeptablem Aufwand ausgetauscht werden kann. Dies hätte den Vorteil, dass sowohl server- als auch clientseitig mit QGIS gearbeitet würde und so beispielsweise das Styling einfacher übernommen werden könnte.

Eine weitere Option besteht darin, statt GeoNode ein Projekt zu verwenden, welches auf GeoNode aufbaut, beispielsweise GeoSHAPE. Für den Beginn wird jedoch auf das reine GeoNode gesetzt, ein Umstieg zu einem anderen auf GeoNode aufbauenden Projekt wird als mögliche Option offengelassen.

8.4 Benutzerauthentifizierung

Für die Benutzerauthentifizierung wird nach Möglichkeit eine bereits existierende Funktion in einem CMS oder Framework bzw. von einer Library verwendet.

8.5 Datenübertragung/ Versionisierung

Für die Datenübertragung zwischen dem BLUgis-Client und BLUshare-Server wird ein geeignetes Protokoll benötigt. Eine naheliegende Methode ist die Übertragung per HTTP POST. Geprüft wird noch, ob sich Git bzw. GeoGig für diesen Einsatzzweck eignen könnten.

8.5.1 REST API

Die Daten werden als File Up- bzw. Download übertragen. Der Datenabgleich findet nach Erhalt der Daten im Datenspeicher auf dem Server bzw. lokal im BLUgis statt.

8.5.2 Git

Es wäre denkbar, Git für die Datenübertragung sowie Versionisierung der Daten zu verwenden. Dabei würde der BLUshare Server sowie jeder Client lokal ein Git Repository besitzen.

Bei dieser Lösung könnten Merge Konflikte zu Problemen führen, diese müssten, wenn möglich, automatisiert behoben werden.

Der Nachteil bei dieser Methode ist, dass es schwieriger wird festzulegen, welche Benutzer in welchem Gebiet Daten ändern dürfen und welche sie sehen dürfen.

8.5.3 GeoGig

GeoGig unterstützt die verteilte Versionisierung von Geodaten und basiert auf Java. Es baut auf den Ideen von Git auf, um Geodaten dezentral zu verteilen. GeoNode bzw. der darunterliegende GeoServer unterstützt GeoGig.

Der Vorteil von GeoGig ist die eingebaute Versionisierung, womit Änderungen nachvollzogen werden können, wenn nötig, wieder rückgängig gemacht werden können. Aber wie bei Git an sich wird so die räumliche Einschränkung der Daten komplizierter, oder es müsste jeder Benutzer über sein eigenes Repository verfügen, was das Zusammenstellen der Daten aus allen Repositories aufwändiger macht.

8.6 Übertragungsformat

Für die Übertragung von Geodaten zwischen den BLUgis Clients und dem BLUshare Server soll ein geeignetes Format verwendet werden. Folgende Anforderungen muss das Format erfüllen:

- Das Format muss clientseitig (QGIS) und serverseitig (Python) importier- und exportierbar sein
- Bei der Konvertierung in das und aus dem Format dürfen keine relevanten Informationen verloren gehen

Wünschenswert:

- Binäre/Komprimierte Übertragung
- Keine Beschränkung der Dateigrösse
- Möglichst breite Unterstützung

8.6.1 Geobuf

Geobuf ist eine Library, welche GeoJSON-Daten verlustfrei komprimiert. Die Komprimierung ist dabei meist effizienter als mit gzip. [30]

8.6.2 GeoCSV

GeoCSV ist eine sich im Entwurf befindende Spezifikation für die Speicherung von Geodaten im CSV Format. [31]

8.6.3 GeoJSON

Das GeoJSON Format basiert auf JSON (JavaScript Object Notation) und ist ein offener Standard für die Darstellung von Geodaten und den dazugehörigen Attributen. [32] Dieses Format wird häufig für die Datenabfrage aus Webapplikationen verwendet.

8.6.4 GeoPackage

GeoPackage ist ein offener Standard für das Speichern von Geodaten und basiert auf einer SQLite Datenbank. [33]

8.6.5 Shapefile

Shapefile ist ein weit verbreitetes Format für Geodaten, welches von der Firma ESRI entwickelt wurde. Wegen seiner Verbreitung und breiten Unterstützung gilt Shapefile als Quasi-Standard im Desktop-GIS-Umfeld. [34] Das Format hat verschiedene Einschränkungen, beispielsweise: [35]

- Feldnamen dürfen maximal 10 Zeichen lang sein
- Eingeschränkte Unicode Unterstützung
- Dateigrösse darf 2 Gigabyte nicht überschreiten
- Rundungsfehler bei Fließkommazahlen möglich

8.6.6 SpatiaLite

SpatiaLite ist eine Erweiterung für die SQLite-Datenbank um geografische Objekte und Funktionen. Das SpatiaLite-Format wird bereits in BLUgis verwendet was eine Übertragung der Daten ohne Konvertierung ermöglichen würde. Ein Nachteil hingegen ist die grosse Dateigrösse bei diesem Format bzw. ein Overhead, z.B. durch den EPSG Datensatz. [36]

8.6.7 Andere Formate

Weitere Formate würden dann in Betracht gezogen werden, falls ein verwendetes Framework oder eine Library von einem bestimmten Format abhängt.

8.6.8 Vergleich

	Geobuf	GeoCSV	GeoJSON	GeoPackage	Shapefile	SpatiaLite
Attributnamen > 10 Zeichen	Ja	Ja	Ja	Ja	Nein	Ja
Dateigrösse > 2GB	Ja	Ja	Ja	Ja	Nein	Ja
1 Datei	Ja	Ja	Ja	Ja	Nein	Ja
Binär	Ja	Nein	Nein	Ja	Nein	Ja
>1 Geometrie pro Tabelle	Nein	Nein	Nein	Nein	Nein	Ja
MultiLineString/Polygon/...	Ja	Ja	Ja	Ja	Nein	Ja
Kreisbogen	Nein	Ja	Nein	Ja	Nein	?
Schema	Nein	Nein	(JSON Schema)	Nein	Nein	Ja
Verbreitung	Gering	Hoch	Sehr hoch	Mittel	Sehr hoch	Hoch
OGR Unterstützung	Nein	Ja (CSV)	Ja	Ja	Ja	Ja

Bemerkungen	Kleine Dateigrösse	Noch kein Standard	Besonders für Web-APIs geeignet	OGC Standard	Sehr hohe Verbreitung aber entscheidende Nachteile	BLUgis Datenbanken benutzen dieses Format
-------------	--------------------	--------------------	---------------------------------	--------------	--	---

Quelle: teilweise von [37]

8.6.9 Fazit Übertragungsformat

Shapefile ist historisch bedingt ein weit verbreitetes Format und ist von GeoNode sogar das einzige unterstützte Format für Geodaten. Die Einschränkungen von dem Format sind jedoch sehr gross, beispielsweise würde die beschränkte Attributnamenlänge eine Anpassung des Datenmodells erfordern.

Bei der Verwendung von SpatiaLite müssen die Daten clientseitig nicht in ein anderes Format konvertiert werden, weil es bereits im BLUgis Client verwendet wird. Jedoch ist die Dateigrösse durch die SpatiaLite-Erweiterungen unnötig gross.

GeoPackage als Übertragungsformat sieht vielversprechend aus, weil es sich möglicherweise in naher Zukunft als Standard etabliert. Aus QGIS heraus kann es auch direkt exportiert werden. Aufwendiger würde sich hingegen der Import auf der Serverseite gestalten, weil die Daten nicht direkt eingelesen werden können, sondern zuerst mit einem Tool wie ogr2ogr konvertiert werden müssen.

GeoJSON ist ein weit verbreitetes und einfach zu handhabendes Format. Als einziger Nachteil wurde hier die Dateigrösse ausgemacht, die bei anderen (binären) Formaten optimaler ist.

Geobuf ist sozusagen GeoJSON in effizient komprimierter Form. Dank einer Python Library kann das Format serverseitig direkt eingelesen werden. Hingegen unterstützt QGIS keinen direkten Export nach Geobuf.

GeoJSON hat sich am geeignetsten erwiesen, weil die Daten von QGIS problemlos exportiert und im Server ohne Konvertierung in Python eingelesen werden können. Die Daten werden für die Übertragung mit Zip komprimiert, um die Übertragungsgrösse möglichst klein zu halten. Das erwies sich als eine gute Kompromisslösung zwischen dem unkomprimierten GeoJSON und dem sehr effizient komprimierten Geobuf und macht besonders auch darum Sinn, weil für jeden Layer eine separate GeoJSON-Datei verwendet wird.

8.7 Fazit

In dieser Technologieanalyse wurden verschiedene Technologien angeschaut und die vielversprechendsten genauer evaluiert. Dabei hat sich besonders GeoNode als ein CMS herausgestellt, welches eine solide Grundlage für dieses Projekt darstellt und einige der erforderlichen Funktionalitäten bereits abdeckt. Deshalb wurde entschieden, das Projekt mit GeoNode als Basis umzusetzen.

GeoNode setzt auf das Django Webframework und benutzt im Hintergrund GeoServer. Als Datenspeicher wird PostgreSQL bzw. PostGIS verwendet.

Für die Übertragung von Geodaten zwischen BLUgis und BLUshare wird auf GeoJSON zusammen mit Zip-Komprimierung gesetzt. Dies stellte sich als ein guter Kompromiss zwischen kleiner Dateigrösse und einfacher Importier- und Exportierbarkeit auf Client- und Serverseite dar.

9 GeoNode

Die Technologieanalyse aus dem vorherigen Kapitel hat ergeben, dass sich GeoNode am besten als Grundlage für dieses Projekt eignet. In diesem Kapitel wird die Funktionalität von GeoNode weiter untersucht und geprüft, ob GeoNode in einer geänderten Grundkonfiguration verwendet werden soll, beispielsweise mit QGIS Server anstatt GeoServer als Unterbau.

9.1 Einleitung

GeoNode ist ein Content Management System für räumliche Daten. Das OpenSource Projekt erlaubt die Verwaltung und Veröffentlichung von Geodaten über eine Weboberfläche.

9.2 Funktionen

9.2.1 Allgemeine Funktionen

- Suche von Geodaten [38]
- Search-Engine basiert auf JSON REST API [39]
- Metadaten-Katalog [38]
- Importieren und Austausch von Geodaten [40]
- Unterstützung von Raster-, Vektordaten und Tabellendaten [40]
- Upload von ShapeFiles, GeoTIFF, KML, CSV [40]
- Open-Geospatial-Consortium-Services (OGC-Services): Web Feature Service (WFS), Web Map Service (WMS) [38] [40]
- WFS: Schnittstellen zum Abfragen und Editieren von räumlichen Geodaten [41] [42]
- WMS: HTTP-Schnittstelle zum Anfordern von Karten aus verteilten räumlichen Datenbanken [42] [43]
- weitere OGC-Services: Web Coverage Service (WCS), Catalogue Service for Web (CSW), Tile Mapping Service(TMS) [42]
- WCS: Schnittstelle zum Lesen und Schreiben von Rasterdaten [42]
- CSW: Schnittstelle zur Anwendung von GeoNode in anderen Webapplikationen (z. B. Suche im Katalog von GeoNode oder Suche von Layers) [42]
- Der Zugriff auf Karten kann für Benutzer beschränkt werden, z.B. nur bestimmte Nutzer haben Zugriff auf bestimmte Karten [40]
- Erstellen von interaktiven Karten, auch von Karten mit mehreren Layers [44]
- Einbinden von Karten in Webseiten [44]
- Editieren von grafischen Styles [44]

9.2.2 Funktionen speziell für Softwareentwickler

- GeoNode ist OpenSource
- Unterstützung von PostGIS-Datenbanken
- pycsw Metadaten Katalog
- Libraries für Python
- OpenLayers / GeoExt Web Mapping Libraries
- GeoNode benutzt Django
- GeoDjango Object-Relational Mapper
- Integrierte Schnittstelle für den Administrator
- GeoNode nutzt Bootstrap für das Frontend
- Django Templates

- Oberfläche (Themes) anpassbar mit LESS und Bootswatch
- verschiedene APIs z.B. GeoServer REST API oder GeoNode Search und REST APIs
- Integration mit Social Media Plattformen wie Facebook, Google+, Twitter
- Integration mit anderen Content Management Systemen wie WordPress oder Drupal

Quelle: [45]

9.3 Systemanforderungen

Für GeoNode werden folgende minimale Systemanforderungen gestellt: [46]

- 6 GB RAM
- Prozessor mit einer Taktrate von 2,2 GHz
- 1 GB Festplattenspeicher
- 64-bit Hardware wird empfohlen
- zusätzlicher Speicherplatz für Daten (es werden 100 GB empfohlen)

Es wird auch noch zusätzliche Software benötigt (Windows):

- Python Interpreter
- Java Runtime Environment
- ant (PATH muss gesetzt sein)
- maven (PATH muss gesetzt sein)
- git

9.4 Unterstützte Webbrowser (Auswahl)

Es werden unter anderem folgende Webbrowser unterstützt:

- Internet Explorer (ab Version 10)
- Mozilla Firefox
- Google Chrome
- Apple Safari

Quelle: [47]

9.5 Inhalte

9.5.1 Layers

Layer werden durch das Heraufladen eines Datensets von Geodaten erstellt, entweder in Form von Vektordaten (zurzeit nur Shapefile) und Rasterdaten (GeoTIFF).

Folgende Berechtigungen lassen für einzelne Benutzer und Gruppen festlegen: Layer ansehen, Layer herunterladen, Metadaten ändern, Daten bearbeiten, Style bearbeiten und Layer managen (update, delete, change permissions, publish/unpublish it).

Ein Layer kann bewertet und kommentiert werden.

9.5.2 Maps

Eine Map besteht aus maximal einer Hintergrundkarte (Base Map), sowie aus einer beliebigen Anzahl von Layern. Als Base Maps stehen standardmässig MapQuest Imagery, MapQuest OpenStreetMap, sowie OpenStreetMap selber zur Verfügung.

Folgende Berechtigungen lassen sich für einzelne Benutzer und Gruppen festlegen: Karte ansehen, Karte herunterladen, Metadaten ändern und Karte managen (update, delete, change permissions, publish/unpublish it).

9.5.3 Documents

Neben Geodaten kann GeoNode auch Dokumente wie PDFs verwalten. Diese werden entweder direkt hochgeladen oder per URL verlinkt.

Auch für Dokumente lassen sich die Berechtigungen anpassen: Dokument ansehen, Dokument herunterladen, Metadaten ändern und Dokument managen (update, delete, change permissions, publish/unpublish it).

9.5.4 People

Benutzer mit einem Account können sich in GeoNode anmelden und verfügen dadurch über die ihnen zugeordneten Rechte. Neue Benutzer können von einem Administrator durch Angabe von E-Mail und Benutzername eingeladen werden. Optional können neue Benutzer über das Django Administratorinterface erstellt werden.

Für einen Benutzeraccount sind folgende Angaben nötig: Benutzername und Passwort. Zusätzlich lassen sich folgende Informationen zuweisen: Vorname, Nachname, E-Mail, Organisation, Position, Telefon, Adresse, allgemeine Beschreibung und Stichwörter.

9.5.5 Groups

Einer Gruppe lassen sich Benutzer zuweisen. Diese Benutzer verfügen dann über die der Gruppe zugewiesenen Berechtigungen.

9.6 Benutzermanagement

9.6.1 Benutzer

Benutzer können sich in GeoNode registrieren. Es muss ein Benutzername, ein Passwort und eine E-Mail-Adresse angegeben werden. Nach der Registrierung wird eine Bestätigung E-Mail an den Benutzer gesendet. Der Benutzer muss dann sein Konto bestätigen.

Benutzer können ihr Profil bearbeiten. Es können die in «People» genannten Daten (Vorname, Nachname, E-Mail...) angegeben werden. In «Account-Settings» von GeoNode können die E-Mail-Adresse, Sprache und Zeitzone geändert werden.

Es kann festgelegt werden bei welchen Aktionen eine E-Mail gesendet werden soll, die dem Benutzer benachrichtigt, wenn ein gewisses Ereignis stattgefunden hat.

Benutzer können benachrichtigt werden, wenn eine Karte oder ein Layer oder ein Dokument erstellt, gelöscht, verändert, bewertet oder kommentiert wurde. Weiterhin kann der Benutzer entscheiden, ob er benachrichtigt werden möchte, wenn jemand seine Aktivitäten anschauen kann.

Ein Benutzer kann auch Informationen (zum Beispiel die Anzahl von Karten oder Dokumenten oder Layers) eines anderen Benutzers anschauen. Es können auch die Aktivitäten eines Benutzers angeschaut werden.

Es können die Aktivitäten eines Benutzers verfolgt werden (Follow).

Der Zugriff auf Karten, Layers oder Dokumente kann für Benutzer oder Benutzergruppen beschränkt werden.

Benutzer können registriert oder unregistriert sein. Nutzer, die noch keine Layers, Karten oder Dokumente hochgeladen haben, sind unregistriert. Nutzer, die schon Daten hochgeladen haben, sind registrierte Nutzer.

Ein registrierter Nutzer kann als Status «Staff», «Superuser» und «Active» haben.

Ein Superuser ist der Administrator und hat alle Zugriffsrechte. Er kann Daten hinzufügen, löschen oder verändern.

Ein Benutzer als Staff hat Zugriff auf das Django Admin Interface.

Active kennzeichnet, dass ein Benutzer verfügbar ist.

Ein Benutzer kann im Terminal nur als Superuser erstellt werden. Über das User Interface von GeoNode können normale Benutzer erstellt werden, die den Status «Active» bekommen.

Mit Hilfe des Django Administration Interfaces kann ein neuer Benutzer angelegt werden oder es kann der Status eines Benutzers geändert werden.

Es können auch Benutzergruppen erstellt werden. Benutzer können zur Gruppe hinzugefügt werden. Alle Benutzer einer Gruppe haben die gleichen Benutzerrechte.

Administratoren können Benutzergruppen erstellen und Benutzer zu einer Gruppe hinzufügen. Den Gruppen können Zugriffsrechte vergeben werden.

Quelle: [48]

9.6.2 Permissions

Layers und Dokumente

Ein Superuser oder der Besitzer eines Layers können Zugriffsrechte für einen Layer vergeben.

Es können folgende Rechte für jedermann, einen oder mehrere Benutzer, eine oder mehrere Benutzergruppen vergeben werden:

- Betrachten eines Layers
- Download eines Layers
- Ändern von Metadaten z.B. Titel, Datum, Sprache, Kategorie, URL...
- Ändern von Daten und Styles eines Layers
- Ändern, Löschen, Veröffentlichen und Ändern der Zugriffsrechte eines Layers

Karten (Maps)

Es können folgende Rechte geändert werden:

- Betrachten einer Karte
- Download
- Ändern von Metadaten
- Ändern, Löschen, Veröffentlichen und Setzen der Zugriffsrechte einer Karte

Veröffentlichen von Daten

Um Daten zu veröffentlichen muss ein Flag `RESOURCE_PUBLISHING` auf `True` gesetzt werden. Neue hochgeladene Daten sind aber unveröffentlicht.

Ein Benutzer, der ein Recht auf `base / ressource model` hat, kann ein Dokument, Layer oder eine Karte veröffentlichen. Es kann entschieden werden, ob die Daten veröffentlicht werden sollen oder nicht.

Quelle: [48]

9.7 Komponenten

9.7.1 Django

GeoNode baut auf Django auf, einem auf Python basierenden Web Application Framework. Django verwendet ein zu MVC ähnliches Pattern.

9.7.2 GeoServer

GeoServer ist ein auf Java basierender Mapserver und wird als Grundlage für GeoNode verwendet. GeoServer erlaubt es, OGC kompatible Web Services anzubieten (WMS, WFS, WCS, ...).

9.7.3 GeoExplorer

GeoExplorer unterstützt das Zusammensetzen von Karten und deren Veröffentlichung über OGC Web Services

9.7.4 PostGIS

PostGIS ist eine Erweiterung für PostgreSQL und wird von GeoNode bzw. GeoServer für die Speicherung der Geodaten verwendet.

9.7.5 pycsw

pycsw ist ein in Python implementierter OGC CSW Server. Damit werden Metadaten in einer standardisierten Form angeboten.

9.7.6 jQuery

jQuery ist eine JavaScript-Bibliothek, welche unter anderem die Selektion und Manipulation von DOM Elementen erlaubt. Innerhalb GeoNode dient jQuery dazu, das UI interaktiv und responsive zu machen.

9.7.7 Bootstrap

Bootstrap ist ein CSS-Framework und enthält eine grosse Anzahl an Gestaltungsvorlagen. Das Framework hilft dabei, dass GeoNode unabhängig vom Browser dasselbe Aussehen hat.

9.7.8 Weitere Libraries und Plugins

GeoNode benutzt eine Reihe weiterer Libraries für Python und Plugins für Django.

Zum Beispiel:

- GDAL
- GeoDjango

gsconfig:

- Python Library
- Erlaubt Änderungen an der lokalen GeoServer-Instanz

Zugriff auf Datastore mit gsconfig innerhalb GeoNode:

```
from django.conf import settings
from geoserver.catalog import Catalog
_user = settings.OGC_SERVER['default']['USER']
_password = settings.OGC_SERVER['default']['PASSWORD']
cat = Catalog(settings.OGC_SERVER['default']['LOCATION'] + "rest", _user, _password)
store = cat.get_store('...')
```

Quelle: [49]

9.8 Architektur

GeoNode ist aus mehreren Schichten aufgebaut, wie die folgende Grafik zeigt. Im Falle von BLUshare wird in der untersten Ebene eine PostGIS-Datenbank für die Speicherung der Geodaten verwendet.

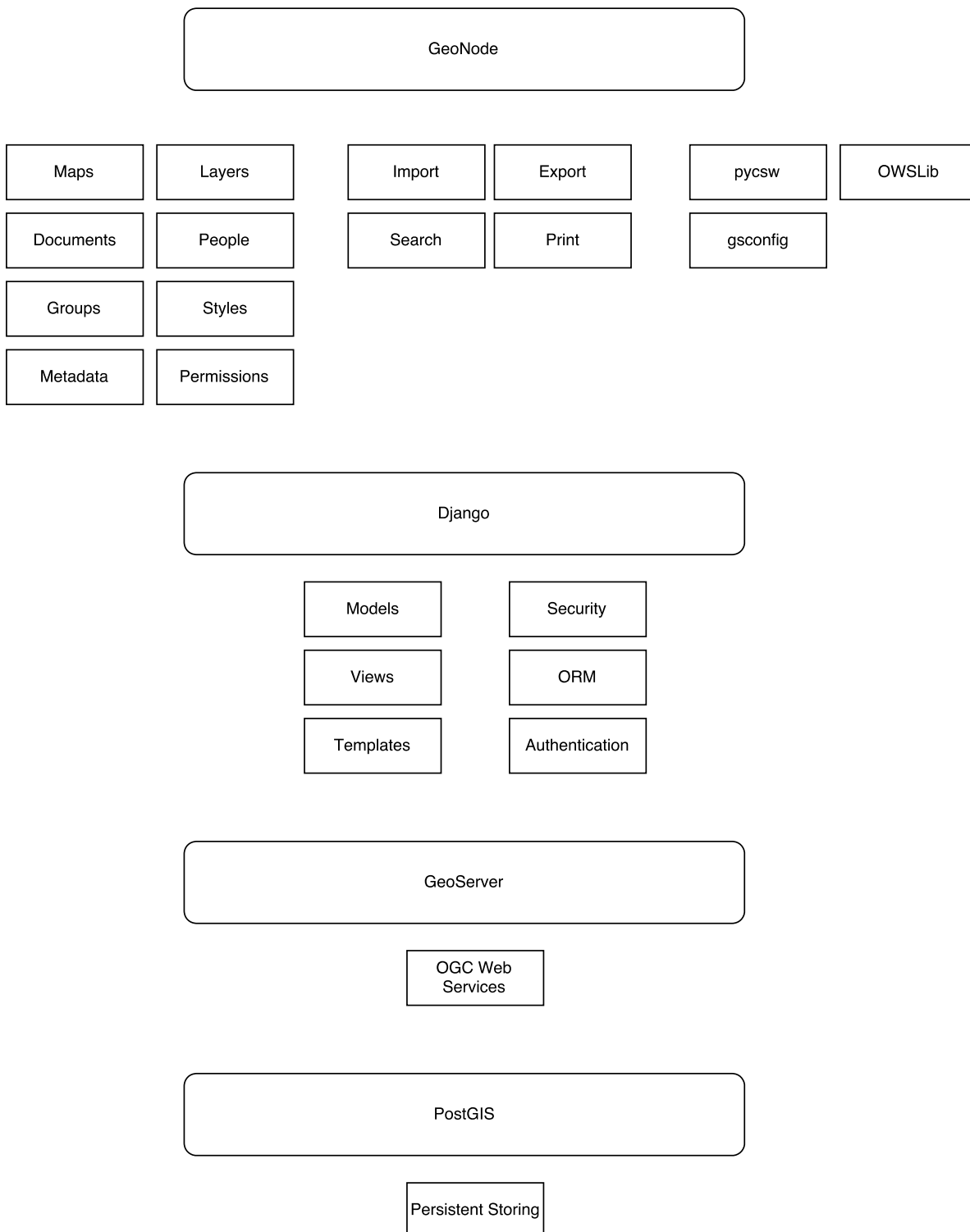


Abbildung 7: Architektur von GeoNode

9.9 Lizenz (GPLv3)

GeoNode kann kostenlos verwendet, erweitert und weiterverbreitet werden (unter der GNU Lizenz Version 3). [50]

Der Lizenztext kann unter dem folgenden Link gefunden werden: [51]

Lizenz von Geonode: [50]

Gemäss [52] soll gesamte Code veröffentlicht werden. Es ist unklar, ob mit dem gesamten Quellcode auch der Code der verwendeten Library veröffentlicht werden muss.

In [53] steht, dass sowohl der Quellcode der Software, die eine unter der GNU lizenzierte Library verwendet, als auch der Quellcode der Library veröffentlicht werden muss, wenn die Library modifiziert wurde.

Laut [53] muss der Quellcode auch veröffentlicht werden, wenn die Software eine unter der GNU lizenzierte Komponente enthält. Originalzitat in [53]: «You can distribute it (the GPL code), provided you make your source available.»

Jedoch steht gemäss [54] nicht im Lizenztext, dass der Quellcode der Software, die eine unter der GNU lizenzierte Komponente enthält, veröffentlicht werden muss.

Laut [55] und [56] greift die GPL erst, wenn die Software weitergegeben wird in Kombination mit der Affero GPL. Hier läuft die Software auf dem Server als Dienst (SaaS). Der Quellcode muss daher nicht veröffentlicht werden.

9.10 Weboberfläche

Das GUI wurde mithilfe von Django-Templates umgesetzt und kann beliebig modifiziert werden.

9.11 Verknüpfung GeoNode und GeoServer

Standardmässig verwendet GeoNode GeoServer als Mapserver. Es scheint, dass GeoNode und GeoServer stark miteinander verknüpft sind. Beispielsweise wird die Benutzerauthentifizierung von GeoNode in GeoServer übernommen. In der geoserver-Django-App befindet sich eine Art Proxy für den GeoServer Zugriff. Siehe auch nächster Abschnitt.

9.12 GeoNode + QGIS Server?

GeoNode verwendet standardmässig GeoServer als Unterbau. Weil auf der Clientseite mit BLUgis QGIS zum Einsatz kommt, kam die Idee auf, auch auf der Serverseite GeoNode zusammen mit QGIS Server anstatt GeoServer zu betreiben. Das hätte den Vorteil, dass bereits vorhandenes Know-how weiterbenutzt werden könnte und beispielsweise die Layer-Stile einfach übernommen werden könnten.

Eine Untersuchung des GeoNode Quellcodes [50] und eine Abklärung bei der GeoNode Community [57] hat ergeben, dass die Verbindung zwischen GeoNode und GeoServer sehr stark ist. Unter anderem würde das Rechtesystem verloren gehen, welches auf GeoServer aufbaut. Im Quellcode befinden sich im Unterordner «geonode/geoserver» Hilfsfunktionen für den Zugriff auf GeoServer. Ausgehend von diesen Hilfsfunktionen könnte eine Art Proxy geschaffen werden, welcher die Zugriffe anstatt an GeoServer an QGIS Server weiterleitet. Angesichts des mehrere Tausend Zeilen umfassenden Codes dieser Hilfsfunktionen wurde aber entschieden, dass sich dieser Aufwand nicht lohnt.

Es existiert bereits ein Projekt, welches genau das getan hat, also GeoServer in GeoNode durch QGIS Server ersetzt. [58] [59] Das Projekt befindet sich bei Kartoza, einem Open Source Dienstleister im Geoinformatik-Umfeld, in Arbeit und wird von der GFDRR (Global Facility for Disaster Reduction and Recovery, Teil der Weltbank) gesponsert. [57] [60] Bei diesem Projekt wird ausserdem in gewissen Teilen Leaflet anstatt OpenLayers

verwendet. Insgesamt funktioniert diese Umsetzung von GeoNode mit QGIS recht gut, bei den Tests sind aber auch einige Probleme aufgetreten:

- Layer können nicht zu einer Karte hinzugefügt werden
- Der Style-Editor von GeoNode kann nicht benutzt werden
- Die QGIS Server-Konfiguration lässt sich nicht aufrufen
- Die Druckfunktion funktioniert nicht

Aufgrund der noch vorhandenen Probleme in dem genannten Projekt und der höheren Abhängigkeit bei neuen Versionen von GeoNode, haben wir uns auch dagegen entschieden, auf diesem Projekt aufzubauen. Die Idee, QGIS Server anstatt GeoServer zu verwenden, wurde deshalb verworfen.

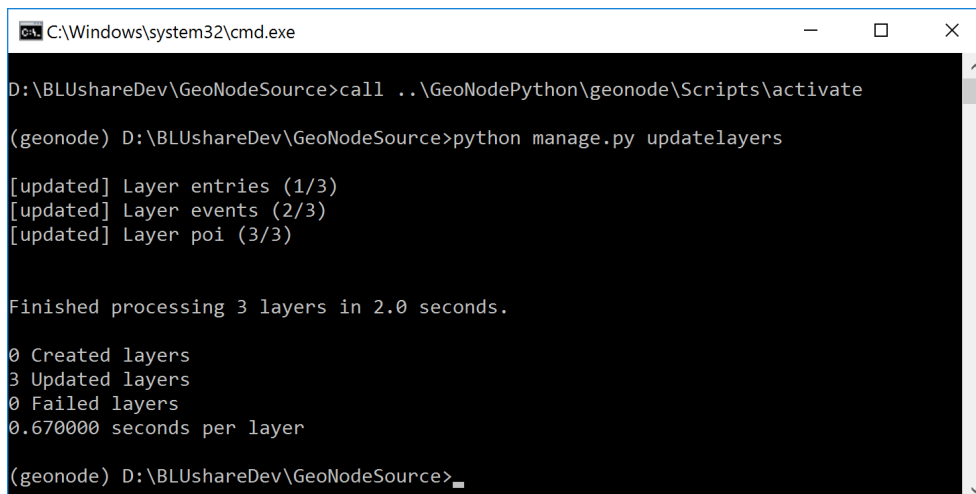
9.13 Datenimport

Für den Upload von Vektordaten unterstützt GeoNode nur Shapefiles. Dieses Format hat sich in der Technologieanalyse aber als ungeeignet herausgestellt, weshalb eine alternative Lösung für den Datenupload gefunden werden muss. Ein erster Ansatz war, die Upload-Methode innerhalb von GeoNode zu erweitern, damit mehr Formate unterstützt werden. Im Verlauf der Arbeit hat sich das aber in der Umsetzung als zu aufwändig herausgestellt, ausserdem müsste auch damit gerechnet werden, dass bei neuen Versionen die eigenen Änderungen wieder angepasst werden müssen. Dieser Ansatz wurde deshalb verworfen.

GeoServer, worauf GeoNode aufbaut, unterstützt hingegen deutlich mehr Formate, sogenannte Datenspeicher. Glücklicherweise bietet GeoNode die Option, Datenspeicher von GeoServer als Layer zu übernehmen, unabhängig von deren Format. Als Datenspeicher-Format wird nun, wie die Technologieanalyse ergeben hat, PostGIS verwendet.

Manuell lassen sich die Layer aus GeoServer in GeoNode bei aktivierter virtualenv über einen «Django Administrative Task» hinzufügen: [61]

```
$ python manage.py updatelayers [62]
```



```
C:\Windows\system32\cmd.exe
D:\BLUshareDev\GeoNodeSource>call ..\GeoNodePython\geonode\Scripts\activate
(geonode) D:\BLUshareDev\GeoNodeSource>python manage.py updatelayers
[updated] Layer entries (1/3)
[updated] Layer events (2/3)
[updated] Layer poi (3/3)

Finished processing 3 layers in 2.0 seconds.

0 Created layers
3 Updated layers
0 Failed layers
0.670000 seconds per layer

(geonode) D:\BLUshareDev\GeoNodeSource>_
```

Abbildung 8: Layer aus GeoServer in GeoNode einfügen

Für BLUshare wurde diese Aufgabe automatisiert und wird nach einem Daten-Upload aufgerufen. Innerhalb von GeoNode lässt sich das folgendermassen machen:

```
from geonode.geoserver.helpers import gs_slurp
gs_slurp()
```

9.14 Einschränkungen/Probleme

- GeoNode unterstützt nur Shapefile Upload [63]
- Deutsche Übersetzungsfehler / scheint grösstenteils automatisiert übersetzt worden zu sein [64]
- Bearbeitung kann nicht auf bestimmtes Gebiet beschränkt werden
- GeoNode ist als soziale Plattform ausgelegt (Bewertungen/ Kommentare). Diese Funktionen werden für dieses Projekt wahrscheinlich nicht benötigt.
- Dokumentation ist teilweise nicht auf dem Stand der aktuellen Software (z.B. kein Explore button / List View / Grid View, fehlende Funktionen wie «Follow» und «Block» User) [65]
- Nicht vorhandene verlinkte Seiten, z.B. einige bei [66]
- Voraussetzungen, damit beim Datenaustausch zwischen BLUgis/BLUshare keine Informationen verloren gehen?
- Benutzer können sich nicht selber registrieren. Nur der Administrator kann Benutzer erstellen.

9.15 Community

GeoNode wird aktiv weiterentwickelt. Der Quellcode befindet sich auf GitHub. [50] Auf GitHub können Fragen gestellt oder Bugs mittels Issues gemeldet werden.

Einerseits existiert ein Forum auf Google Groups, wo Fragen, Probleme oder Bugs gepostet werden können (siehe [67]). Andererseits gibt es ein Mailing-List Forum (siehe [68]).

9.16 Fazit

GeoNode hat sich als nützlich erwiesen, weil es einen grossen Funktionsumfang mitbringt. So ist beispielsweise bereits eine Editierfunktion vorhanden, Rechte lassen sich präzise setzen und Layerstile lassen sich über eine Weboberfläche anpassen.

Es wurde entschieden, GeoNode in seiner Standardkonfiguration, das heisst, mit GeoNode, GeoServer und OpenLayers zu verwenden. Dadurch wird es einfach, jeweils mit den aktuellsten Entwicklungen von GeoNode Schritt zu halten.

Eine entscheidende Einschränkung von GeoNode ist, dass nur Shapefile als Uploadformat von Vektordaten unterstützt wird. Diese Einschränkung wird durch die Benutzung von GeoServer und späterem Import in GeoNode umgangen.

Werden Anpassungen an GeoNode vorgenommen, bringt das den Nachteil mit sich, dass Änderungen an GeoNode durch die Community mit unseren Änderungen in Konflikt stehen könnten. Ein Umstieg auf eine neue Version von GeoNode kann deshalb mit Zusatzaufwand durch das Merging überschneidender Änderungen verbunden sein. Um diesen Aufwand gering zu halten, soll deshalb die Architektur von BLUshare so gewählt werden, dass die eigenen Änderungen möglichst getrennt bleiben vom originalen GeoNode-Quellcode.

10 Design

10.1 Ausgangslage

Der geplante BLUshare Server ist eine Erweiterung der bereits existierenden Software BLUgis. Die folgende Grafik zeigt die physische Architektur, welche bereits vorhanden ist. BLUgis wird einerseits in den Fahrzeugen auf Tablets verwendet und andererseits im Stützpunkt auf einem Desktop PC, auf welchem Daten erfasst und geändert werden.

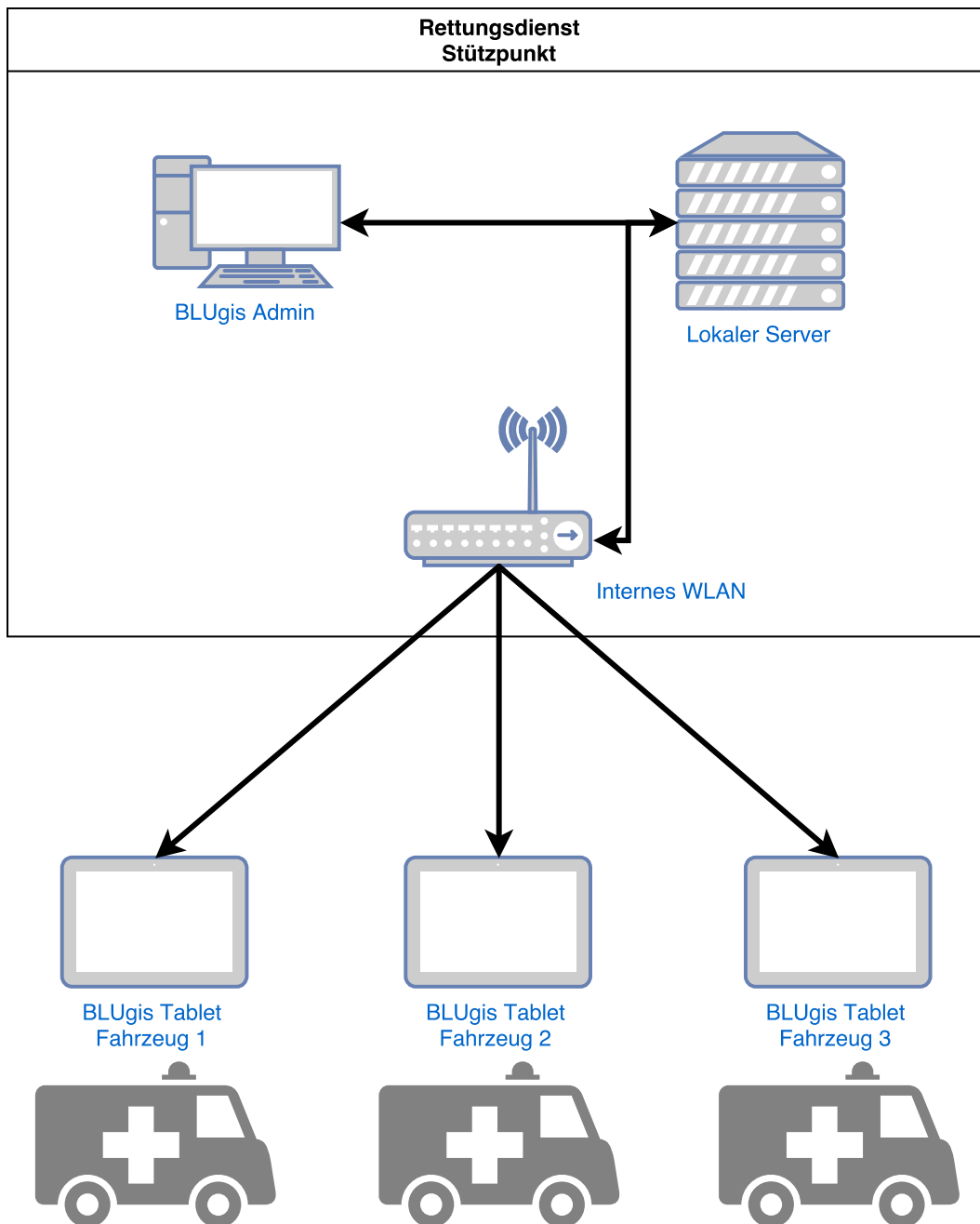


Abbildung 9: Physische Architektur von BLUgis

10.2 Ziele

BLUshare erweitert die vorhandene Architektur um einen externen Server, welcher den Datenaustausch zwischen verschiedenen Rettungsdiensten und anderen Blaulichtorganisationen ermöglichen soll.

Der BLUshare-Server baut auf GeoNode auf, eine Content Management System für Geodaten, welches wiederum auf dem Webframework Django aufbaut. Grundsätzlich soll sich die Architektur an dem bereits existierenden Aufbau von GeoNode bzw. an Django-Standards orientieren.

GeoNode wird von der Community aktiv weiterentwickelt. Damit BLUshare mit der Weiterentwicklung von GeoNode mithalten kann, sollen die nötigen Anpassungen nach Übernahme einer neuen Version möglichst geringgehalten werden. Dazu sollen die eigenen Anpassungen an bzw. Erweiterungen von GeoNode wo möglich nicht in den vorhandenen Sourcefiles vorgenommen werden, sondern in einem separaten Bereich.

Auf der Clientseite sind einige Erweiterungen in BLUgis vorgesehen. Hier wird die bereits existierende Architektur nur dann angepasst, wenn das durch die vorzunehmenden Modifikationen nötig wird.

10.3 Physische Architektur

Der neue BLUshare Server ist die zentrale Stelle für den Austausch von Geodaten. Die Kommunikation geschieht dabei nur über die BLUgis-Admin-Clients, für die Verteilung der Daten auf die Tablets kommt weiterhin ein lokaler Server zum Einsatz.

Auf die Daten des BLUgis Servers kann nicht nur mit BLUgis zugegriffen werden, sondern auch über einen Webclient. Dieser bietet zudem Exportmöglichkeiten in andere Formate.

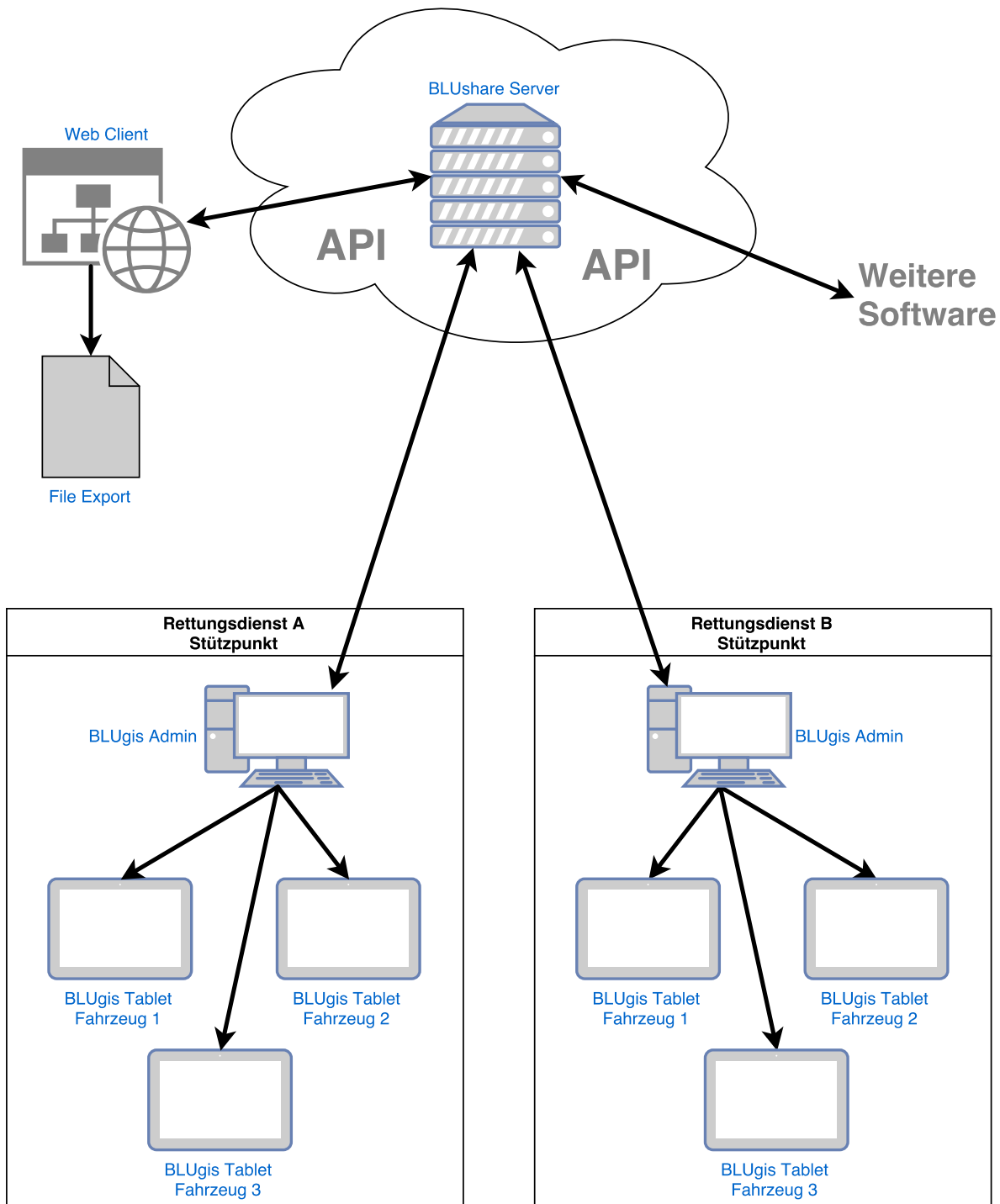


Abbildung 10: Architektur von BLUgis mit BLUshare

10.4 Logische Architektur

Die folgenden Diagramme zeigen den internen Aufbau vom BLUshare-Server und dem BLUgis-Client. Dabei sind die Stellen blau markiert, welche im Rahmen dies Projektes erstellt oder stark überarbeitet wurden. Die Packages stehen für Python-Module.

10.4.1 BLUshare

Die BLUshare-Erweiterung ist eine Django-App, welche eine Ergänzung zu den bereits existierenden Django-Apps von GeoNode ist.

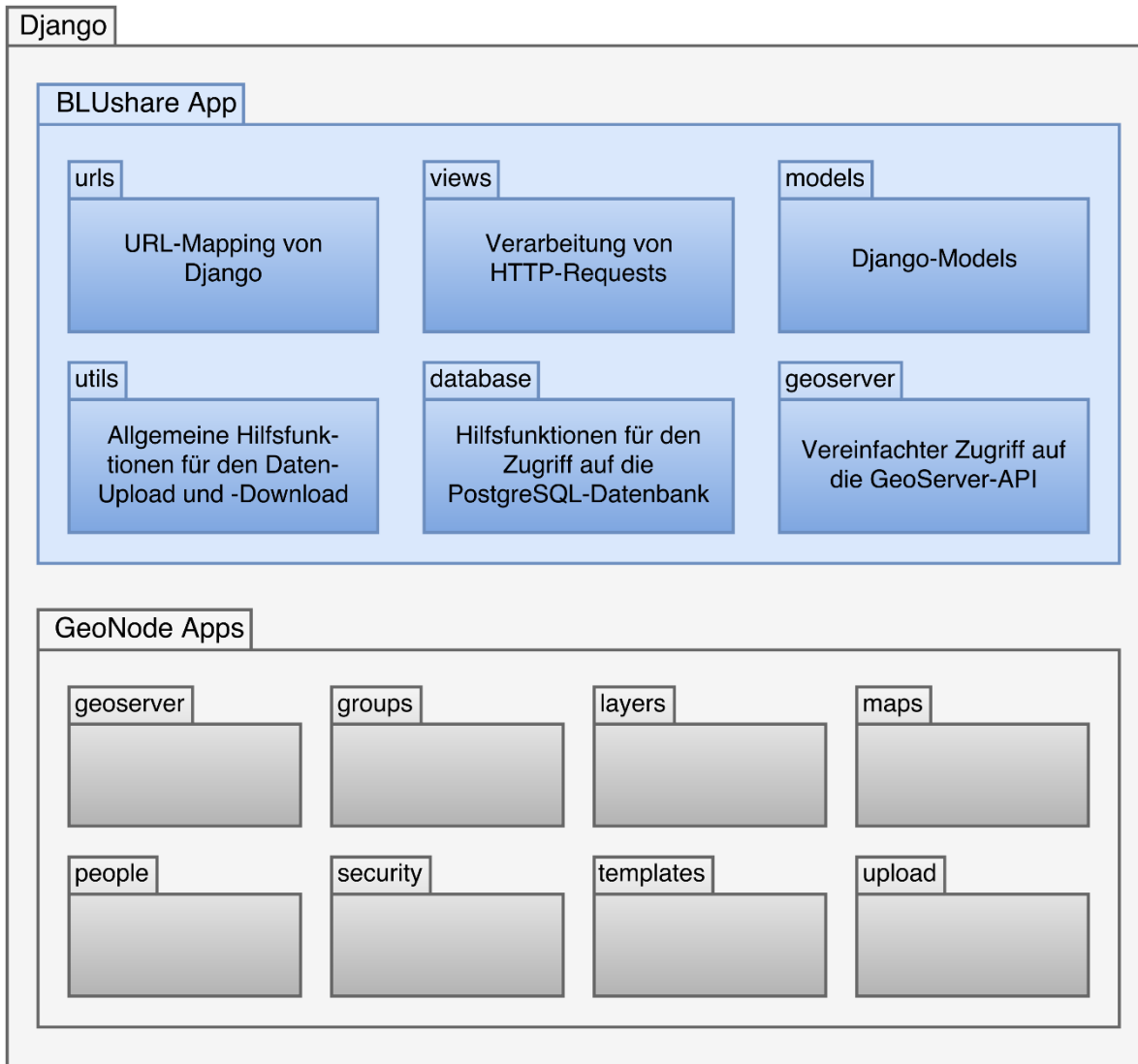


Abbildung 11: Logische Architektur von BLUshare

10.4.2 BLUgis

Der BLUgis-Client ist als Plugin für QGIS [69] in Python implementiert.

Folgende Haupt-Packages sind vorhanden:

- **actions**: Die «actionlibrary» definiert QAction's, ein Objekt aus der Qt Library, welches einem Button in Qt zugewiesen werden kann. [70] Die einzelnen *actions-Module und -Klassen legen die konkrete Aktion einer QAction fest.
- **forms**: Attribut-Dialog für die Anzeige und Bearbeitung von Eigenschaften der Objekte auf den Layern.
- **migration**: Erkennung der Schema-Version von Datenbanken und Datenverzeichnissen und Migration auf die aktuelle Version.
- **utils**: Allgemeine Hilfsfunktionen, die unabhängig vom restlichen Plugin sind.

QGIS Plugin

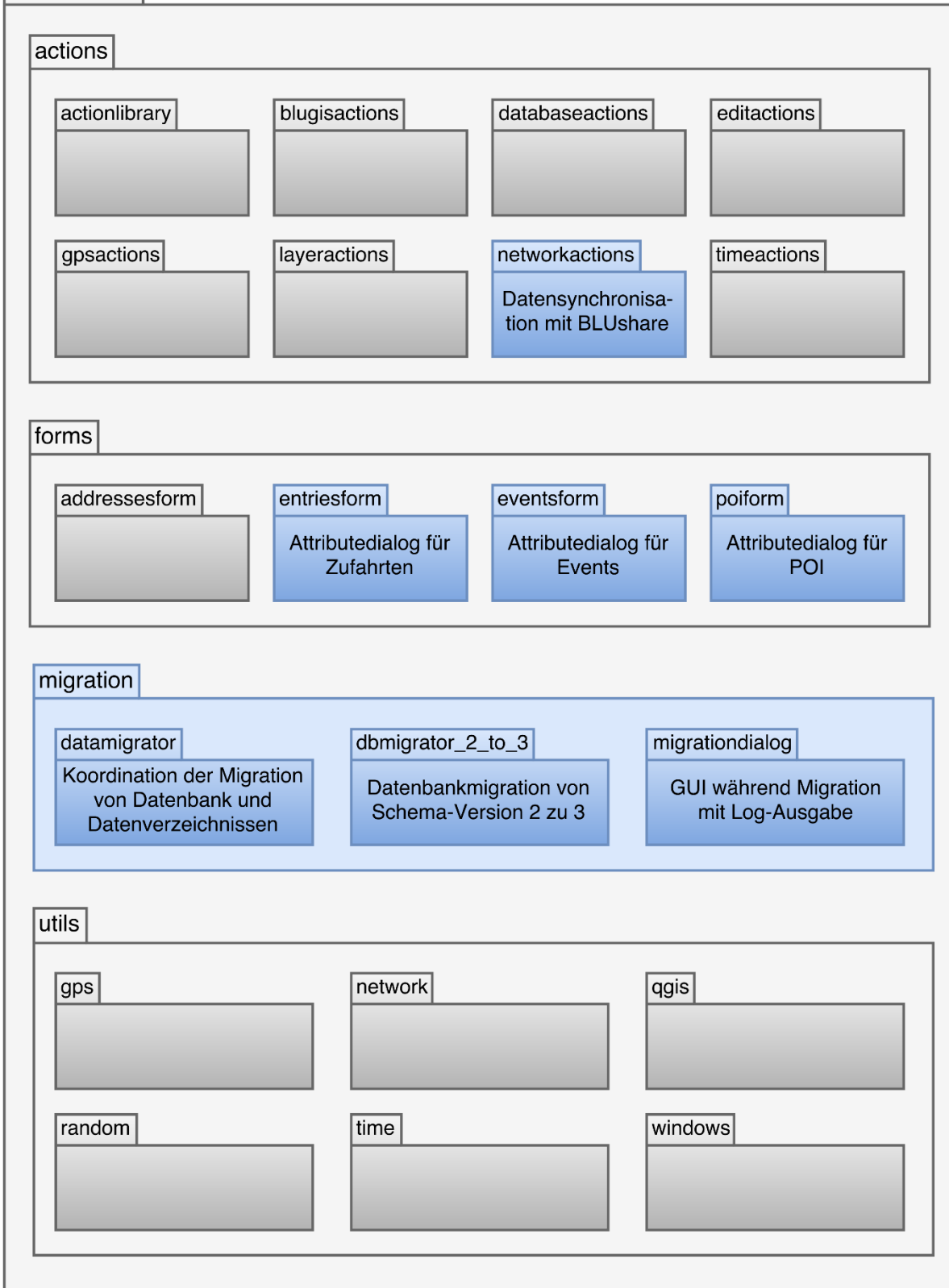


Abbildung 12: Logische Architektur von BLUshare

10.5 Synchronisation

10.5.1 Schnittstelle

Der BLUshare Server bietet eine REST Schnittstelle für den Up- und Download von Geodaten. Momentan können folgende Daten zwischen dem Client (BLUgis Administrator) und Server (BLUshare) synchronisiert werden:

- POI: Punkte (z.B. Standorte von Arztpraxen und Spitälern)
- Events: Polygone (z.B. Baustellen und Veranstaltungen)
- Zufahrten: Polygone (z.B. erlaubte und gesperrte Gebäudezufahrten)
- Attachments: Dateien (z.B. PDF-Anhang für POI, Event oder Zufahrt)

Für den Daten Up- und Download muss der Benutzer authentifiziert sein.

Als Antwort liefern die Up- und Download-Schnittstellen ein JSON-Objekt zurück, welches unter anderem mitteilt, ob die Verarbeitung geklappt hat. Wenn nicht, ist im Objekt eine vorgegebene Fehlermeldung enthalten. Nach einem Upload enthält das JSON zusätzlich eine Liste von Attachment-IDs, welche auf dem Server fehlen und vom Client heraufgeladen werden sollten.

Grundsätzlich wird der Standard-HTTP-Statuscode 200 zurückgegeben, auch bei Fehlern, die abgefangen werden können (z.B. wenn nicht authentifiziert oder bei einem Fehler bei der Datenverarbeitung). Lediglich bei nicht abgefangenen Fehlern wird entsprechender HTTP-Statuscode zurückgegeben (z.B. 500 bei Serverfehler).

10.5.2 BLUgis-Integration

Wird auf einem BLUgis-Admin-Client ein lokaler Datenbereitstellungs-Vorgang gestartet, wird davor automatisch auch der Datenaustausch mit dem BLUshare Server ausgelöst. Ist der Upload abgeschlossen, sendet BLUshare seinerseits die aktualisierten Datensätze, welche dann lokal integriert und schliesslich auf die Tablets verteilt werden.

10.5.3 Synchronisationsmechanismus

Für die Synchronisation wurde das Datenmodell von BLUgis erweitert. Folgende Attribute sind dazugekommen:

- BLUshare ID: ermöglicht durch Verwendung einer UUID eine eindeutige Identifizierung von Objekten über alle BLUshare-Benutzer hinweg.
- BLUshare User: wird serverseitig gesetzt und legt den Besitzer von einem Objekt fest. Im BLUgis-Client ist dieses Attribut für den aktuellen Nutzer auf «null» gesetzt, um die eigenen Objekte identifizieren zu können.
- BLUshare Timestamp: legt den Zeitpunkt fest, wann das Objekt zuletzt verändert wurde. Erlaubt Optimierungen bei der Synchronisation.
- Is-Hidden-Flag: ermöglicht das Ausblenden fremder Objekte.

Die Datensynchronisation aus BLUgis heraus läuft in mehreren Schritten ab:

1. Der BLUgis Client exportiert die Daten von Events-, POI- und Zufahrts-Layer in einzelne GeoJSON-Dateien. Es werden nur die Objekte exportiert, welche vom Benutzer selber erstellt wurden, also nicht diejenigen Daten, die von BLUshare stammen.
2. Die GeoJSON-Dateien werden in einem Zip verpackt und zum BLUshare-Server gesendet.
3. Der BLUshare-Server gleicht die erhaltenen Daten mit den bereits vorhandenen Daten ab. Dabei werden nur diejenigen Daten betrachtet, welche sich innerhalb vom Erfassungsbereich des aktuellen Benutzers befinden.

- a. Neue Daten, welche auf dem Server noch nicht vorhanden sind, werden übernommen.
 - b. Objekte, welche in der Server-Datenbank vorhanden sind und dem aktuellen Benutzer gehören, aber vom Client her nicht mehr übertragen wurden, werden aus der Datenbank gelöscht. Das ist der Fall, wenn der Client lokal bei sich ein Objekt gelöscht hat oder wenn sich der Erfassungssperimeter vom aktuellen Benutzer entsprechend verändert hat.
 - c. Veränderte Objekte werden übernommen, sofern das Objekt dem authentifizierten Benutzer gehört und der Timestamp neuer bzw. grösser ist als der vom bereits existierenden Objekt.
4. BLUshare sammelt von den Objekten des authentifizierten Benutzers eine Liste aller Attachment-IDs, und prüft, ob entsprechende Dateien auf dem Server vorhanden sind. Anschliessend wird eine Liste mit den fehlenden Attachments an den Client zurückgesendet.
 5. Der BLUgis-Client erhält die Liste mit den fehlenden Attachments und sucht entsprechende Dateien zusammen.
 6. Die Dateien werden in einem Zip verpackt und als Upload an den BLUshare-Server gesendet.
 7. Auf dem Server werden aus dem Zip diejenigen Dateien übernommen, welche in der Liste der fehlenden Attachments vorhanden sind.
 8. Der BLUgis-Client ruft die Download-URL vom Server auf.
 9. BLUshare sammelt alle Daten, welche sich innerhalb des Leseperimeters des authentifizierten Benutzers befindet und generiert daraus für jeden Layer eine einzelne GeoJSON-Datei.
 10. Die GeoJSON-Dateien werden in einem Zip verpackt und als Download dem BLUgis-Client gesendet.
 11. Der BLUgis-Client gleicht die erhaltenen Daten mit den bereits vorhandenen Daten ab.
 - a. Neue Daten, welche in der lokalen Datenbank noch nicht vorhanden sind, werden übernommen.
 - b. Objekte, welche in der lokalen Datenbank vorhanden sind, aber vom Server her nicht übertragen wurden, werden aus der Datenbank gelöscht, sofern diese einem fremden Benutzer gehören. Eigene Objekte werden zur Sicherheit durch den Synchronisationsvorgang nie gelöscht, selbst wenn das Objekt serverseitig gelöscht wurde.
 - c. Veränderte Objekte werden übernommen, sofern der Timestamp neuer bzw. grösser ist als der vom bereits existierenden Objekt.
 12. Der BLUgis-Client prüft anhand der Attachments-Datenbank, welche Dateien noch fehlen, generiert daraus eine Liste mit den Attachment-IDs und sendet diese an den BLUshare-Server.
 13. Der BLUshare-Server erhält die Liste mit den fehlenden Attachments und sucht entsprechende Dateien zusammen.
 14. Die Dateien werden in einem Zip verpackt und als Download an den BLUgis-Client gesendet.
 15. Der BLUgis-Client übernimmt aus dem Zip diejenigen Dateien, welche in der Liste der fehlenden Attachments aufgeführt sind.

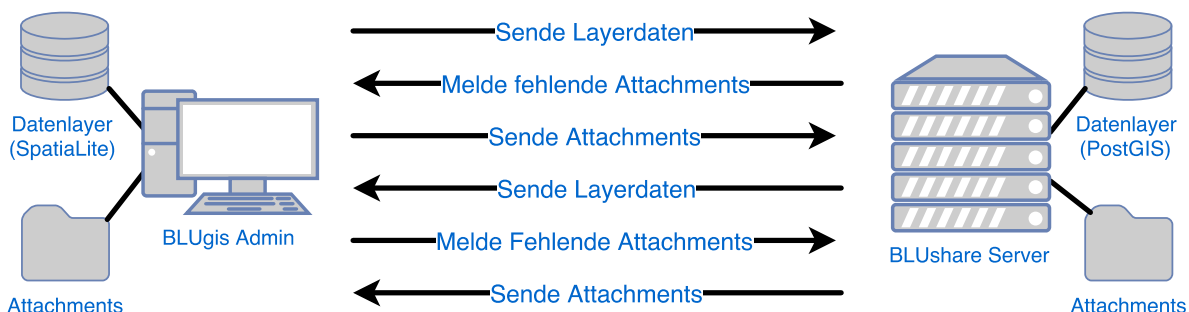


Abbildung 13: Datensynchronisation zwischen BLUgis und BLUshare

10.5.4 Korrektheitsprüfung

Um sicherzustellen, dass während der Datenübertragung keine relevanten Daten verloren gehen, wurden die nach BLUshare hochgeladenen Daten mit den heruntergeladenen verglichen. Dabei sind folgende Unterschiede aufgefallen:

- Nach der Synchronisation waren 2 Koordinatenstellen abgeschnitten, z.B.:
Vorher: [706287.47638887609, 235148.49444439198]
Nachher: [706287.476388876, 235148.494444392]
→ Die beiden Stellen gehen vermutlich während der Speicherung in der serverseitigen Datenbank verloren. Weil sich die daraus resultierenden Positions-Unterschiede im Nanometerbereich bewegen, wurde dieses Problem als unbedeutend eingestuft und die Ursache nicht weiter verfolgt.
- Umlaute werden umgewandelt, z.B.:
Vorher: ä
Nachher: \u00e4
→ Nach dem erneuten Import in QGIS werden die Umlaute korrekt dargestellt

Die allgemeine Funktionstüchtigkeit vom Synchronisationsmechanismus wird mit manuellen Tests im Kapitel «Testing» geprüft.

10.5.5 Einschränkungen

Ein BLUshare-Benutzer sollte nur auf genau einem BLUgis-Client verwendet werden, was in der Praxis auch kein Problem darstellen sollte, weil eine bestimmte Blaulichtorganisation im Normalfall BLUgis nur auf einem Rechner einsetzt. Wird das gleiche Login trotzdem auf verschiedenen Rechnern eingesetzt, bestehen folgende Einschränkungen:

- Verfügt der 2. BLUgis-Client bei der ersten Synchronisation noch über keine Daten, dann werden in BLUshare schon vorhandene Daten von dem jeweiligen Benutzer herausgelöscht.
- Bei Bearbeitungskonflikten wird stillschweigend die neuere Version genommen.
- Wird bei einem BLUgis-Client ein Objekt gelöscht und mit BLUshare synchronisiert, dann bleibt dieses Objekt bei dem anderen BLUgis-Client trotzdem noch erhalten und wird beim Synchronisieren in BLUshare sogar erneut erzeugt. Das Objekt müsste manuell bei beiden BLUgis-Clients gelöscht werden.

Zudem ist der Wechsel zwischen verschiedenen BLUshare-Benutzern innerhalb der gleichen BLUgis-Installation problematisch, sofern bereits Daten in den Layern vorhanden sind. Ein Benutzerwechsel würde manuelle Änderungen in der lokalen Datenbank mit sich ziehen, damit die Objektbesitzer korrekt hinterlegt sind. Wird hingegen der Benutzer verändert und sollen die Daten weiterhin «demselben Benutzer» gehören, dann müsste der Benutzername manuell in der serverseitigen Datenbank angepasst werden. In der Praxis wird ein Benutzerwechsel aber eher selten sein.

10.5.6 Optimierungsmöglichkeiten

Der Synchronisationsmechanismus kann in verschiedenen Bereichen noch verbessert werden, hier sind einige Ideen:

- Vor Synchronisation eine Liste der Objekt-IDs und Timestamps übertragen, damit anschliessend nur die Datensätze übertragen werden müssen, welche neu dazugekommen sind oder sich verändert haben.
- Erkennung von doppelten bzw. ähnlichen Objekten, damit beispielsweise eine Baustelle nicht ausversehen von zwei unterschiedlichen Blaulichtorganisationen erfasst wird (falls diese Überschneidungen beim Erfassungssperimeter haben)

- Ausschliessen von Events, welche bereits vergangen sind und deshalb gar nicht angezeigt würden (wobei bereits BLUgis die Möglichkeit bietet, solche Events herauszulöschen)

10.6 Sicherheit

10.6.1 Server

Der Server, auf dem BLUshare laufen soll, bietet verschiedene Angriffsmöglichkeiten:

- Knacken des SSH Zugangs
- Knacken des Plesk Zugangs
- Sicherheitslücken im Server
- Falsch konfigurierte Zugriffsrechte

Um diese Angriffsflächen zu verkleinern, werden folgende Massnahmen ergriffen:

- Verwendung sicherer Passwörter: länger als acht Zeichen, mindestens ein Grossbuchstabe, ein Kleinbuchstabe, eine Zahl und ein Sonderzeichen
- Ändern des Standard SSH Ports
- SSH Zugang für root User blockieren
- Automatisches Erkennen und Blockieren von Brute-Force-Attacken mit Fail2ban [71]
- Regelmässige Installation aktueller Sicherheitsupdates auf dem Server

10.6.2 Serverdatenbank

Weil BLUshare eine Datenbankintensive Anwendung ist und die einzufüllenden Daten vom Client her kommen, ist die Datenbank besonders gegen SQL-Injections zu schützen. Ein erster Schutz besteht darin, dass nur authentifizierte Benutzer Zugriff auf API-Funktionen haben, welche mit der Datenbank in Berührung kommen. Im Quellcode wird konsequent auf sogenannte «query placeholders» bzw. Prepared Statements gesetzt. Dieser Mechanismus verhindert SQL-Injections. [72]

10.6.3 Übertragung

Bei der Kommunikation mit dem Server besteht die Gefahr, dass vertrauliche Daten wie Passwörter oder schützenswerte Geodaten mitgelesen werden (beispielsweise Telefonnummern und Zugangshinweise zu Gebäuden). Als Massnahme wird durchgehend auf eine verschlüsselte Verbindung (TLS) bei der Übertragung von Passwörtern und Geodaten gesetzt.

10.6.4 Benutzerauthentifizierung

Für die Benutzerauthentifizierung werden die Mechanismen aus GeoNode verwendet, welches Benutzergruppen und ein Rechtesystem beinhaltet. [73] Das Authentifizierungssystem ist zudem mit GeoServer verknüpft, was beim Zugriff auf die GeoServer-Konfiguration kein erneutes Einloggen nötig macht.

10.6.5 Erfassungs- und Leseperimeter

Für jeden Benutzer lässt sich über ein Polygon ein Gebiet festlegen, in welchem er die Erfassungsrechte oder Leserechte über die darin enthaltenen Daten besitzt. Das bedeutet, dass der Benutzer in seinem Gebiet vorhandene Geodatenobjekte bearbeiten und löschen darf. Erfassungs- und Leseperimeter verschiedener Benutzer können sich auch überschneiden, in diesem Fall haben alle Benutzer die erweiterten Bearbeitungsrechte über die Daten in ihrem Erfassungs- bzw. Leseperimeter.

Ausserhalb von dem Perimeter sind die Bearbeitungsrechte eingeschränkt. Erstellt ein Benutzer ein Objekt ausserhalb von seinem Gebiet, bleibt dieses zwar lokal vorhanden, aber es wird nicht im BLUshare Server abgespeichert. Für Objekte, die von anderen Benutzern erstellt wurden und ausserhalb des eigenen Perimeters liegen, kann lediglich ausgewählt werden, dass diese nicht mehr angezeigt werden sollen. Die ausgeblendeten Objekte werden auf der Karte gesondert dargestellt. Im Eigenschaftendialog können diese Objekte wieder ein-geblendet werden.

10.6.6 Datenvandalismus

Mit Datenvandalismus ist der Fall gemeint, dass bewusst falsche Daten erfasst werden oder Objekte unrechtmässig gelöscht werden, um die Benutzung von BLUshare bzw. BLUgis zu stören. Datenvandalismus seitens der Blaulichtorganisationen ist sehr unwahrscheinlich. Trotzdem kann das nicht ausgeschlossen werden, beispielsweise dann nicht, wenn das BLUshare-Login geklaut und unberechtigterweise verwendet wird. Der BLUshare-Server ist soweit geschützt, dass nicht Daten von einem fremden Benutzer verändert werden können. Für den Fall, dass trotzdem Datenvandalismus eintritt, empfiehlt es sich, in regelmässigen Abständen ein Backup der vorhandenen Datensätze anzulegen, um den Schaden bei Eintritt von Datenvandalismus zu minimieren.

10.6.7 Offene Sicherheitsprobleme

Ein grosses Sicherheitsproblem stellen die Attachments dar. Enthalten diese einen Virus oder eine andere schädliche Software, würde dieser aktiviert werden, sobald das Attachment aufgerufen wird. Eine Einschränkung der erlaubten Datentypen auf beispielsweise PDF würde die Sicherheit auch nur beschränkt erhöhen, weil auch einem PDF ausführbarer Programmcode angehängt werden kann. [74] Die Gefahr kann etwas vermindert werden, indem server- und clientseitig ein Antivirenprogramm zum Einsatz kommt, welches zumindest bereits bekannt Schädlingsprogramme meistens zuverlässig erkennt. Die Auswirkungen von einem möglichen Schadensfall können vermindert werden, wenn die Geräte, auf denen BLUgis zum Einsatz kommt, innerhalb vom Netzwerk getrennt sind von der restlichen IT-Infrastruktur.

Eine weitere Gefahr geht von dem Diebstahl der Zugangsdaten eines BLUshare-Benutzers aus. Ein Angreifer könnte so Daten manipulieren, was besonders dann verheerend sein kann, wenn diese Manipulationen lange Zeit nicht auffallen. Im Augenblick ist diese Gefahr besonders darum gross, weil die Zugangsdaten für einen höheren Komfort clientseitig im Klartext abgespeichert werden (Alternative: Token-basiertes System). Dieser Gefahr kann entgegengewirkt werden indem der Zugang für ein BLUshare-Zugang auf bestimmte IP-Adressbereiche eingeschränkt wird, sofern die Organisation feste IPs zugewiesen hat.

10.7 Webclient

Als Webclient kommt GeoNode zum Einsatz, welches unter anderem das Betrachten, Ändern und Exportieren von Daten erlaubt. Im Rahmen der Bachelorarbeit wurde lediglich die Startseite angepasst.



Abbildung 14: Startseite von BLUshare

Nach einem Synchronisationsvorgang über BLUshare wird automatisch folgendes gemacht:

- Layer innerhalb von GeoServer erstellen, falls diese noch nicht existieren. Falls diese schon existieren, wird der Cache für diesen Layer geleert, damit die Änderungen sichtbar werden.
- Hinzufügen der Layer aus GeoServer in GeoNode.

11 Testing

Im folgendem Kapitel werden manuelle Tests zur BLUgis- und BLUshare-Applikation beschrieben.

11.1 Manuelle Tests

11.1.1 BLUshare Administration

Test 1: Erstellen eines Benutzers in GeoNode

Aufgabe: Ein Administrator will einen neuen Benutzer erstellen, welcher für den Upload und Download von Daten benutzt werden kann.

Voraussetzung: Der Administrator ist mit seinem Administrator-Konto in GeoNode eingeloggt.

Schritte zur Umsetzung:

1. Administrator klickt, nachdem er eingeloggt ist, auf seinen Benutzernamen. Es erscheint ein Menü.
2. Administrator klickt im Menü auf «Admin» und gelangt auf die Administrationsseite
3. Administrator scrollt zu dem Punkt «Group Profiles».
4. Administrator klickt auf «Add» neben «Group Profiles». Es erscheint das Formular zum Hinzufügen eines neuen Benutzers.
5. Administrator gibt Benutzername und Passwort ein und klickt auf «Save».
6. Anschliessend können optional weitere Felder wie Vorname, Nachname, E-Mail-Adresse ausgefüllt werden.

Erwartete Resultate:

Der neu erstellte Benutzer ist in der Liste aller Benutzer verfügbar. Mit einem Klick auf dem Benutzer können dessen Eigenschaften angesehen und gegebenenfalls bearbeitet werden. Die angezeigten Eigenschaften entsprechen den vorher erfassten.

Test 2: Zuweisung von Erfassung- und Leseperimeter an einem Benutzer

Aufgabe: Ein Administrator möchte eine neue Gruppe in GeoNode erstellen und dieser einen Erfassungsperimeter und Leseperimeter zuweisen.

Voraussetzung: Der Administrator ist mit seinem Administrator-Konto in GeoNode eingeloggt.

Schritte zur Umsetzung:

1. Administrator klickt, nachdem er eingeloggt ist, auf seinen Benutzernamen. Es erscheint ein Menü.
2. Administrator klickt im Menü auf «Admin» und gelangt auf die Administrationsseite.
3. Administrator scrollt zu dem Punkt «Users».
4. Administrator klickt auf «Add» neben «Users». Es erscheint das Formular zum Hinzufügen einer neuen Gruppe.
5. Administrator gibt im ersten Feld einen Namen für die Gruppe ein.
6. Administrator gibt im Feld «Slug» einen Slug (Bezeichnung für die Gruppe, die in der URL erscheinen soll, da URLs keine Leerzeichen bearbeiten können) für die Gruppe ein.
7. Administrator gibt im Feld «Description» eine kurze Beschreibung der Gruppe ein.
8. Administrator gibt im Feld «Email» optional eine E-Mail-Adresse an.

9. Administrator stellt mittels einer Auswahlbox ein, ob die Gruppe für registrierte Benutzer sichtbar ist.
10. Administrator legt den Erfassungspereimeter für die Gruppe fest, indem er auf der Karte «Region with create permission» die Region sucht, wo die Gruppe Daten erfassen darf, und in dieser Region ein Polygon in die Karte einzeichnet.
11. Administrator legt den Leseperimeter für die Gruppe fest, indem er auf der Karte «Region with read permission» die Region sucht, wo die Gruppe Daten lesen darf, und in dieser Region ein Polygon in die Karte einzeichnet.
12. Administrator gibt optional unter «Keywords» Schlüsselwörter an.
13. Administrator wählt aus einer Dropdownliste einen Benutzer aus, der zu der Gruppe gehören soll.
14. Administrator weist dem ausgewählten Benutzer eine Rolle zu. Dazu wählt er eine Rolle aus einer weiteren Dropdownliste.
15. Administrator klickt auf «Save» und die erstellte Gruppe wird einer Liste von Gruppen hinzugefügt.
 - 15a. Administrator klickt auf «Save and add another» um die Gruppe hinzuzufügen und eine weitere Gruppe zu erstellen.
 - 15b. Administrator klickt auf «Save and continue editing» um die Gruppe hinzuzufügen und weitere Änderungen an dieser Gruppe vorzunehmen.

Erwartete Resultate:

Die neu erstellte Gruppe ist in der Liste aller Gruppen verfügbar. Mit einem Klick auf die Gruppe können dessen Eigenschaften angesehen und gegebenenfalls bearbeitet werden. Die angezeigten Eigenschaften entsprechen den vorher erfassten.

11.1.2 BLUshare Synchronisation

Test 3: Erfasste Daten auf BLUshare laden und fremde Daten von BLUshare erhalten

Aufgabe: Der Benutzer möchte erfasste Daten anderen Benutzern via BLUshare zur Verfügung stellen und fremde Daten auf sein BLUgis laden.

Voraussetzung: Der Benutzer ist in BLUshare registriert und hat einen Erfassungs- und Leseperimeter zugeteilt.

Schritte zur Umsetzung:

1. Benutzer erfasst eine Zufahrt.
 - 1a. Benutzer erfasst einen Event (z.B. eine Baustelle)
 - 1b. Benutzer erfasst einen POI
2. Benutzer klickt auf den Synchronisations-Button worauf die eigenen Daten ins BLUshare übertragen und anschliessend die Daten von anderen Benutzern ins BLUgis geladen werden.

Erwartete Resultate:

- Die eigenen neuen Daten wurden in der serverseitigen Datenbank abgelegt, sofern diese innerhalb des Erfassungspereimeters des Benutzers sind.
- Die Daten von fremden Benutzern, welche sich innerhalb des Leseperimeters befinden, wurden im BLUgis hinzugefügt.

Test 4: Bestehende Daten ändern und synchronisieren

Aufgabe: Der Benutzer möchte bestehende Daten verändern. Andere Benutzer sollen die geänderten Daten auf ihr Tablet laden können.

Voraussetzung: Der Benutzer ist in BLUshare registriert und hat einen Erfassungs- und Leseperimeter zugeteilt. Es sind bereits Objekte im BLUgis und BLUshare vorhanden.

Schritte zur Umsetzung:

1. Benutzer verändert eine Zufahrt.
 - 1a. Benutzer verändert einen Event.
 - 1b. Benutzer verändert einen POI.
2. Benutzer klickt auf den Synchronisations-Button, worauf die geänderten Daten ins BLUshare übertragen werden.
3. Ein zweiter Benutzer, welche noch über die alten Daten des anderen Benutzers verfügt, klickt auf synchronisieren.

Erwartete Resultate:

- Geänderte hochgeladene Daten wurden in die Server-Datenbank übernommen, sofern ein bereits vorhandenes Objekt keinen neueren Timestamp hat als das hochgeladene Objekt.
- Bei dem zweiten Benutzer wurden die Änderungen übernommen.

Test 5: Bestehende Daten löschen und synchronisieren

Aufgabe: Der Benutzer möchte bestehende Daten löschen. Bei anderen Benutzern sollen die gelöschten Daten auch entfernt werden.

Voraussetzung: Der Benutzer ist in BLUshare registriert und hat einen Erfassungs- und Leseperimeter zugeteilt. Es sind bereits Objekte im BLUgis und BLUshare vorhanden.

Schritte zur Umsetzung:

1. Benutzer verändert eine Zufahrt.
 - 1a. Benutzer löscht einen Event.
 - 1b. Benutzer löscht einen POI.
2. Benutzer klickt auf den Synchronisations-Button, worauf die noch vorhandenen Daten ins BLUshare übertragen werden.
3. Ein zweiter Benutzer, welche noch über die Daten des anderen Benutzers verfügt, klickt auf synchronisieren.

Erwartete Resultate:

- Die im lokalen BLUgis gelöschten Daten werden auch in der Server-Datenbank gelöscht, sofern ein bereits vorhandenes Objekt keinen neueren Timestamp hat als das hochgeladene Objekt.
- Bei dem zweiten Benutzer wurden die gelöschten Daten auch aus dem BLUgis entfernt.

11.1.3 BLUgis

Test 6: Ausblenden fremder Daten

Aufgabe: Ein Benutzer hat eine Zufahrt, ein Event oder ein POI via BLUshare von einem fremden Benutzer bekommen und möchte es ausblenden.

Schritte zur Umsetzung:

1. Der Benutzer wechselt in den Editiermodus, in dem er auf «Zufahrten» klickt.
 - 1a. Der Benutzer wechselt in den Editiermodus, in dem er auf «Events» klickt.
 - 1b. Der Benutzer wechselt in den Editiermodus, in dem er auf «POIs» klickt.
2. Der Benutzer selektiert die fremde Zufahrt, die auf der Karte grau gefärbt ist.
3. Der Benutzer öffnet die Details des grau gefärbten Elements.
4. Der Benutzer sieht, von welchem fremden Benutzer er die Zufahrt, das Event oder POI erhalten hat.
5. Der Benutzer klickt auf das Symbol zum Ausblenden der Daten. Die Daten werden ausgeblendet (Polygone werden auf der Karte schraffiert dargestellt). Der Hintergrund der Detailansicht färbt sich von Blau zu Grau.
6. Der Benutzer klickt auf «Schliessen» um die Änderungen zu speichern.

Erwartete Resultate:

Der Hintergrund der Detailansicht ist grau. Ausgeblendete Polygone sind im Editiermodus auf der Karte schraffiert dargestellt mit grauer Hintergrundfarbe. Ausgeblendete POIs haben eine graue Hintergrundfarbe. Wird der Editiermodus beendet, ist das Objekt nicht mehr sichtbar.

Test 7: Einblenden fremder Daten

Aufgabe: Ein Benutzer möchte eine fremde Zufahrt, ein fremdes Event oder ein fremdes POI einblenden.

Voraussetzung: Das einzublendende Event, POI oder die einzublendende Zufahrt sind ausgeblendet.

Schritte zur Umsetzung:

1. Der Benutzer wechselt in den Editiermodus, in dem er auf «Zufahrten» klickt.

Es erscheinen die ausgeblendeten Zufahrten.

 - 1a. Der Benutzer wechselt in den Editiermodus, in dem er auf «Events» klickt.

Es erscheinen die ausgeblendeten Events.
 - 1b. Der Benutzer wechselt in den Editiermodus, in dem er auf «POIs» klickt.

Es erscheinen die ausgeblendeten POIs.
2. Der Benutzer selektiert die fremde Zufahrt, die auf der Karte hellgrau gefärbt ist.
3. Der Benutzer öffnet die Details des grau gefärbten Elements.
4. Der Benutzer klickt auf das Symbol, damit die Daten einblendend werden. Die Daten werden einblendend (Polygone werden auf der Karte grau dargestellt). Der Hintergrund der Detailansicht färbt sich von Grau zu Blau.
5. Der Benutzer klickt auf «Schliessen» um die Änderungen zu speichern.

Erwartete Resultate:

Der Hintergrund der Detailansicht wird blau dargestellt. Die Daten sind auf der Karte einblendend.

11.2 Abnahme

Eine Abnahme dieser manuellen Tests fand am 14.06.2016 durch den Industriepartner statt.

12 Weiterentwicklung

BLUshare bzw. BLUgis können mit weiteren Funktionen erweitert werden.

12.1.1 Weitere Benutzerrechte für Admin

Es könnten weitere Benutzerrechte für Administratoren realisiert werden. Zum Beispiel können Administratoren im Erfassungssperimeter eigene und fremde Daten verändern und löschen, jedoch nicht ausserhalb ihrer zugewiesenen Gebiete.

12.1.2 Datenfilter für Synchronisation

Für Up- und Download von Daten könnte eine Filterfunktion mit SQL für POIs, Events und Entries eingebaut werden.

12.1.3 Download des Kartenmaterials von Swisstopo

Es könnte eine Downloadmöglichkeit des Swisstopo Kartenmaterials eingebaut werden. Für die Verwendung der Karten von Swisstopo muss die Lizenz geprüft werden. Es kann auch ermöglicht werden ein anderes Kartenmaterial herunterzuladen.

12.1.4 Zentrale Baustellenverwaltung durch geeignete Imports

Es könnte eine zentrale Baustellenverwaltung durch geeignete Imports realisiert werden.

12.1.5 Weitere Imports, wie zum Beispiel Hydranten

Die Applikation könnte um Möglichkeiten zum Import von weiteren POIs, wie zum Beispiel Hydranten erweitert werden.

12.1.6 Versionierung der GeoDaten

Möglich wäre, dass Geodaten versioniert werden können mittels GeoNode und GeoGig.

12.1.7 Austausch GeoServer durch QGIS Server

An Stelle vom GeoServer könnte auch QGIS Server als Unterbau von GeoNode verwendet werden. Dies sollte mit akzeptablem Aufwand geschehen. Vorteil dieser Möglichkeit wäre, dass client- und serverseitig mit QGIS gearbeitet werden könnte. Ausserdem könnte das Styling von Geodaten vereinfacht werden.

12.1.8 Einsatz von GeoSHAPE anstelle von GeoNode

Es könnte auch ein auf GeoNode basiertes Projekt (GeoSHAPE) eingesetzt werden.

12.1.9 Verwendung von GeoPackage als Übertragungsformat

An Stelle von JSON könnte auch GeoPackage als Übertragungsformat verwendet werden, da GeoPackage in Zukunft etablierter Standard werden wird. QGIS bietet eine Exportmöglichkeit von Geodaten ins GeoPackage

Format. Der Import von GeoPackage kann sich als problematisch erweisen, da diese Daten erst ins JSON-Format umgewandelt werden müssten. Dazu braucht es ein externes Tool wie ogr2ogr.

12.1.10 Webclient-Editierung

GeoNode bietet zwar bereits Editierfunktionen an, aber damit diese für einen grösseren Benutzerkreis verwendet werden kann, müsste in GeoNode für jeden Benutzer ein eigener Layer angelegt werden, welcher nur seine eigenen Daten enthält. Das liesse sich beispielsweise mit Datenbank-Views umsetzen. [75]

Teil III: Projektmanagement

13 Vorgehensmodell

Dieser Abschnitt beschreibt das für die Entwicklung verwendete Vorgehensmodell.

Das Vorgehensmodell basiert auf dem agilen Manifest. [76]

Gemäss dem agilen Manifest sind folgende Punkte wichtig:

- Individuen und Interaktionen sind wichtiger als Prozesse und Werkzeuge
- Funktionierende Software ist wichtiger als umfassende Dokumentation
- Die Zusammenarbeit mit dem Kunden ist wichtiger als Vertragshandlungen
- Sich auf Änderungen einzustellen ist wichtiger als einem Plan zu folgen

Quelle: [76]

13.1 SCRUM

Als Vorgehensmodell wird SCRUM verwendet. Der Begriff SCRUM kommt aus dem Rugby und bedeutet «Ge- dränge». In Rugby ist SCRUM ein Spielzug. Das Vorgehensmodell SCRUM basiert auf agiler Softwareentwick- lung. Die Softwareentwickler arbeiten eng mit dem Auftraggeber zusammen und liefern in regelmäßigen Ab- ständen neue Funktionalität der zu entwickelnden Software. Die Beteiligten, die SCRUM durchführen, haben bestimmte Rollen.

13.1.1 Scrum-Rollen

Im SCRUM Prozess werden den beteiligten Personen verschiedene Rollen zugewiesen.

Scrum-Master:

Der Scrum-Master leitet den Prozess und sorgt dafür, dass der Prozess korrekt umgesetzt wird. Er beseitigt Probleme, die den Ablauf behindern. Die Scrum-Artefakte werden vom Scrum-Master im Auge behalten. [77]

Product-Owner:

Der Product-Owner ist der Auftraggeber und pflegt das Product-Backlog. Die Vorgaben im Product-Backlog werden vom Product-Owner priorisiert, sodass eine schnelle Auslieferung funktionierender Produktinkre- mente möglich wird. [77]

Entwicklerteam:

Das Team entwickelt die Software und besteht in diesem Projekt aus 2 Mitgliedern. In anderen Projekten kön- nen beispielsweise 5 oder mehr Personen zum Entwicklerteam gehören. Die Teammitglieder teilen die User- Stories in Tasks auf. Die Tasks werden den Entwicklern untereinander aufgeteilt. Jedes Teammitglied aktuali- siert täglich den restlichen Aufwand eines Tasks. Der Restaufwand eines Sprints wird in einem Burndown-Dia- gramm dargestellt. [77]

13.1.2 Scrum-Artefakte

Im SCRUM-Prozess werden verschiedene Artefakte verwendet.

Product-Backlog:

Das Product-Backlog listet alle Anforderungen der Software als User Stories auf.

Sprint-Backlog:

Das Sprint-Backlog enthält die User-Stories, die im Sprint umgesetzt werden.

Produktinkrement:

Das Produktinkrement ist das funktionierende Arbeitsresultat am Ende eines Sprints. Es wird an der Sprint-Demo vorgestellt.

13.1.3 Sprint und Sprint Plannings

Die Entwicklung findet in sogenannten Sprints statt. Ein Sprint ist ein Arbeitsabschnitt und dauert in diesem Projekt 2 Wochen. Die Vorgaben/Use Cases werden als User Stories verfasst. Die zu erledigenden User Stories werden in einem Product-Backlog verwaltet. Zu Beginn eines jeden Sprints findet ein Sprint Planning statt. Daran nehmen der Product Owner, der Scrum-Master und die Entwickler teil. Hier werden die User Stories für den Sprint im Sprint-Backlog festgelegt. Eine User Story wird in Tasks aufgeteilt. Die Entwickler schätzen mit Hilfe von Story Points den Entwicklungsaufwand einer User Story ab. Die Anzahl der Story-Points beschreiben die Komplexität der Anforderung. Es wird einer Story einen Wert zugewiesen und mit einer anderen User Story verglichen. Beispielsweise bekommt eine User Story 5 Story-Points. Es wird nun geschätzt, ob eine andere User Story komplexer ist oder weniger komplex. Die komplexere User-Story bekommt einen höheren Wert, zum Beispiel 8. Ist die Story beispielsweise weniger komplex als die Story mit 5 Story-Points, bekommt sie beispielsweise 3 Story-Points. Die Werte der Story-Points entsprechen den Fibonacci-Zahlen in abgewandelter Form: 1,2,3,5,8,13,20... [78]

13.1.4 Standup-Meeting

Zu Beginn eines Arbeitstags findet ein Standup-Meeting statt wo berichtet wird, was am letzten Tag gemacht wurde und was als nächstes gemacht wird. Anschließend werden möglich aufgetretene Probleme diskutiert. Am Standup-Meeting nehmen der Scrum-Master und das Entwicklerteam teil.

In diesem Projekt haben wir eine Beschreibung des Arbeitsstandes, Probleme und der anstehenden Aufgaben kurz vor dem wöchentlichen Meeting geschickt und am Meeting diskutiert.

13.1.5 Sprint Review

Am Ende eines Sprints findet eine Sprint-Review statt (aller 2 Wochen). Hier wird die Funktionalität der Software vorgestellt. Das Entwicklerteam zeigt an der funktionierenden Software was in dem Sprint realisiert wurde. Es wird nur fertig implementierte Funktionalität gezeigt. Daran kann der Entwicklungsfortschritt erkannt werden. Wurde eine im Sprint vorhergesehene User Story aus Zeitgründen nicht fertig gestellt, wird sie mit in den nächsten Sprint genommen. [77]

13.1.6 SCRUM-Rollen in diesem Projekt

Stefan Keller:

Prof. Stefan Keller ist Geoinformatiker und Dozent für Datenbanksysteme und Informationssysteme an der HSR.

Im Rahmen der Bachelorarbeit ist er der betreuende Dozent und übernimmt die Rolle des Scrum Masters. Er moderiert die Meetings und vermittelt bei Schwierigkeiten.

Markus Honegger:

Markus Honegger arbeitet als Rettungssanitäter bei Regio 144 in Rüti.

Er ist der Auftraggeber und übernimmt die Rolle als Product Owner. Markus Honegger steht für Rückfragen zur Verfügung.

Martin Eisenhammer:

Martin Eisenhammer ist Informatikstudent an der HSR und ist Teil des Entwicklerteams.

Josua Stähli:

Josua Stähli ist Informatikstudent an der HSR und gehört zum Entwicklerteam.

14 Entwicklungsumgebung

Dieser Abschnitt beschreibt die für die Entwicklung verwendeten Entwicklungsumgebungen, die eingesetzte Projektmanagement Software und die Kommunikation mit dem Betreuer und Industriepartner. Jedes Teammitglied arbeitet mit einer eigenen Entwicklungsumgebung.

14.1 Integrierte Entwicklungsumgebungen

Als integrierte Entwicklungsumgebungen werden Microsoft Visual Studio 2015 Enterprise und PyCharm 2016.1.4 verwendet.

14.1.1 PyCharm 2016.1.4

PyCharm von der Firma JetBrains ist eine kostenpflichtige IDE zur Entwicklung von Python Anwendungen. Es werden Syntax-Highlighting und Autocompletion unterstützt. Studenten können PyCharm kostenlos herunterladen. Es ist auch ein Code Inspection Tool eingebaut, welches den Quellcode auf PEP-8 Richtlinien überprüft.

14.1.2 Microsoft Visual Studio 2015 Enterprise

Die kostenpflichtige IDE Microsoft Visual Studio 2015 unterstützt die Programmiersprache Python und kann als Python-IDE verwendet werden. Studenten können diese Entwicklungsumgebung kostenlos bei Microsoft Dreamspark herunterladen. Für Python muss das Paket «Python Tools for Visual Studio» installiert werden. Dieses Paket ist kostenlos. Mit Visual Studio 2015 können in Python Konsolenanwendungen und Webanwendungen mit Django entwickelt werden. Das Paket «Python Tools for Visual Studio» unterstützt Syntax-Highlighting und Autocompletion.

Zur Umsetzung des Webfrontends ist die Projektvorlage «Django Web Project» geeignet.

14.2 Versionsverwaltung

Als Versionsverwaltung wird die Webapplikation Bitbucket (Git) verwendet. Es kann jeder seinen Quellcode in dieses Repository einchecken. Es sollte nur getesteter und fehlerfreier Quellcode eingecheckt werden. Der Quellcode sollte auf dem neuesten Stand sein.

Als Git-Client (grafische Benutzeroberfläche) wird SourceTree 1.8.3.0 von Atlassian eingesetzt.

14.3 Dokumentenverwaltung

Die Dokumentation der Bachelorarbeit wird auf Google Docs verwaltet. Jedes Projektmitglied hat eigene Zugangsdaten und kann Dokumente in Google Docs betrachten und wenn er das Bearbeitungsrecht zugewiesen bekommen hat, auch Dokumente bearbeiten. Durch die Verwaltung der Dokumente auf Google Docs entfällt eine regelmässige Datensicherung auf ein externes physisches Speichermedium (z.B. eine externe Festplatte).

14.4 Projektmanagementsoftware

Für das Projektmanagement wird die Webapplikation Jira benutzt. Hier werden alle User Stories und Tasks verwaltet. Für jede User-Story kann der Aufwand in Story-Points geschätzt werden. Es kann der Status eines Tasks geändert werden, wenn der Task gerade bearbeitet werden soll oder wenn der Task erledigt worden ist.

Bei den Tasks kann eine Zeitschätzung in Stunden angegeben werden. Ein Burndown-Diagramm zeigt die verbleibende geschätzte Zeit an.

14.5 Kommunikation

Zur Kommunikation wird hauptsächlich E-Mail genutzt.

Montags in der ersten Hälfte des Semesters bzw. mittwochs in der zweiten Hälfte des Semesters gibt es ein Treffen mit Herrn Keller bzw. Herrn Honegger, um den aktuellen Stand zu besprechen oder den laufenden Sprint abzuschließen und den nächsten Sprint zu planen.

15 Projektplanung

Dieses Dokument beschreibt die Meilensteine des Projekts und die einzelnen Projektphasen. Es werden die User Stories detailliert beschrieben. In der Entwicklungsphase (Sprints) sind auch die geschätzten und geleisteten Arbeitsaufwände pro User Story angegeben.

15.1 Meilensteine

Woche 2: Server für Entwicklung einrichten

Woche 4: Abschluss Anforderungsanalyse

Woche 6: Abschluss Technologieanalyse/-evaluation

Woche 8: Einarbeitung Frameworks

Woche 10: Programmschnittstelle für Up-/Download

Woche 12: Einbau BLUgis

Woche 15: Web Client

Woche 17: Abschluss Bachelorarbeit und Präsentation

15.2 Analysephase

Zeitraum: 22.02.2016 - 04.04.2016

In der Analysephase wurden folgende Aufgaben durchgeführt:

Einarbeitung in Python und Django

Dokumentation und Einrichten der Entwicklungsumgebungen und Projektmanagement-Tools:

- Dokumentation der Vorgehensweise in der Entwicklung (SCRUM) und der Entwicklungsumgebungen
- Einrichtung Server4You
- Einrichtung Jira
- Einrichtung Bitbucket

Anforderungsanalyse:

- Beschreibung der Vision
- Beschreibung der Architektur
- Dokumentation funktionaler Anforderungen und Use Cases
- Dokumentation nichtfunktionaler Anforderungen
- Risikoanalyse (grob)

Technologieanalyse:

- Beschreibung von GeoNode
- Beschreibung und Evaluation von Geo-CMS

15.3 Sprint 1

Zeitraum: 04.04.2016 - 18.04.2016 (2 Wochen)

15.3.1 User Story: BLS-7 Benutzermanagement (1 Story point)

Beschreibung:

- Permissions für Admin, Editor, Reader
- Logins für Client und API

Tasks:

- Dokumentieren des Benutzermanagements/Permissions in GeoNode

Zeitschätzung:

geschätzte Zeit: 6 Stunden

tatsächlich benötigte Zeit: 6 Stunden

15.3.2 User Story: BLS-8 Spezifikation der Benutzerschnittstelle für Datenup- und download. (3 Story points)

Tasks:

- Anforderungen an Schnittstelle festlegen
- Austauschformat und Übertragungsprotokoll festlegen
- Einbaumöglichkeit in GeoNode überprüfen und dokumentieren

Zeitschätzung:

geschätzte Zeit: 18 Stunden

tatsächlich benötigte Zeit: 14 Stunden

15.3.3 User Story: BLS-12 Datenmodell erstellen (3 Story points)

Beschreibung:

- Datenmodell an «BluGIS» angelehnt (POI, Zufahrt, Event)
- POINT-Geometrie
- POLYGON-Geometrie
- Handling Attachments

Tasks:

- Datenmodell von BLUgis beschreiben
- Zusätzlich benötigte Attribute überlegen
- Diagramm erstellen (UML oder ER)
- Prüfen, wie Datenstruktur in GeoNode erstellt wird

Zeitschätzung:

geschätzte Zeit: 13 Stunden

tatsächlich benötigte Zeit: 13 Stunden, 45 Minuten

15.3.4 User Story: BLS-20 Software Frameworks definieren und einrichten (3 Story points)

Beschreibung:

- Kriterien: kostenfrei, open source, etabliert, «Branchen Leader»

Tasks:

- Entwicklungsumgebung einrichten
- Git-Repository einrichten (Bitbucket)
- Dokumentation der verwendeten Frameworks
- GeoNode auf Server einrichten

Zeitschätzung:

geschätzte Zeit: 17 Stunden

tatsächlich benötigte Zeit: 28 Stunden

15.3.5 User Story: BLS-21 Security Anforderungen bestimmen (2 Story points)

Beschreibung:

- Welche/wieviel Sicherheit ist notwendig?

Tasks:

- Überlegen und Dokumentieren der Sicherheitsanforderungen
- Dokumentieren, welche Security Anforderungen in GeoNode schon vorhanden sind

Geschätzte Zeit: 6 Stunden

Tatsächlich benötigte Zeit: 4 Stunden

Burndown Diagramm

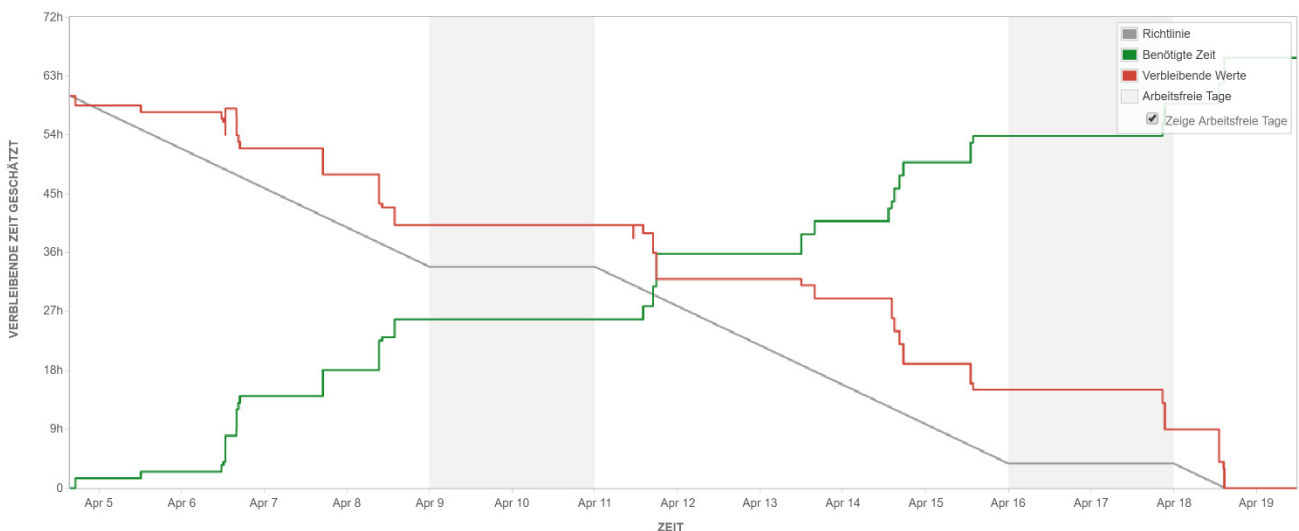


Abbildung 15: Burndown Diagramm Sprint 1

15.4 Sprint 2

Zeitraum: 20.04.2016 - 09.05.2016 (3 Wochen)

15.4.1 User Story: BLS-15 BLUshare-Benutzerverwaltung (3 Story points)

Beschreibung:

- pro Betrieb ein BLUshare-Login
- pro Login ein Erfassungs-Perimeter (EP) zuweisbar
- pro Login ein Lese-Perimeter (LP) zuweisbar
- Login dient sowohl für Up-/Download von/zu BLUgis als auch für Web-Login
- Jeder User kann in BLUshare nur eigene Daten ändern/löschen

Akzeptanzkriterien:

- Login-Verwaltung möglich
- Erfassung eines EP und LP pro Login möglich
- gegenseitiger Schutz der Daten

Tasks:

- Benutzerhandbuch für Login- und Permissionverwaltung
- Erweiterung User-Datenmodell mit EP und LP
- Erfassen von EP und LP ermöglichen

Zeitschätzung:

geschätzte Zeit: 18 Stunden

tatsächlich benötigte Zeit: 21 Stunden, 30 Minuten

15.4.2 User-Story: BLS-42 GUID einführen (3 Story points)

Beschreibung:

- GUID für alle Daten (POI, Event, Entry) einführen
- GUID für Datei-Attachments einführen

Tasks:

- Datenmodell der BLUgis-SpatialLite-DBs mit GUID erweitern
- in BLUgis neue DB für Attachments einführen
- Migrationsskript für neues DB-Modell
- Dokumentation anpassen
- Attachments für Zufahrten

Zeitschätzung:

geschätzte Zeit: 20 Stunden

tatsächlich benötigte Zeit: 32 Stunden

15.4.3 User-Story: Anpassungen BLUgis-GUI (2 Story points)

Beschreibung:

- Bei POI, Events und Entries zusätzliche Infos im GUI:
 - über BLUshare teilen (Checkbox)

- Infos, falls fremde Daten «erhalten über BLUshare, erfasst von %USER%»

Tasks:

- BLUgis DB-Modell erweitern um «über BLUshare teilen» und «BLUshare-Benutzer»
- Feature Forms um zusätzliche Felder erweitern (nur GUI)
- Die neuen Informationen im GUI anzeigen und Änderungen abspeichern

Zeitschätzung:

geschätzte Zeit: 18 Stunden

tatsächlich benötigte Zeit: 9 Stunden

Burndown-Diagramm

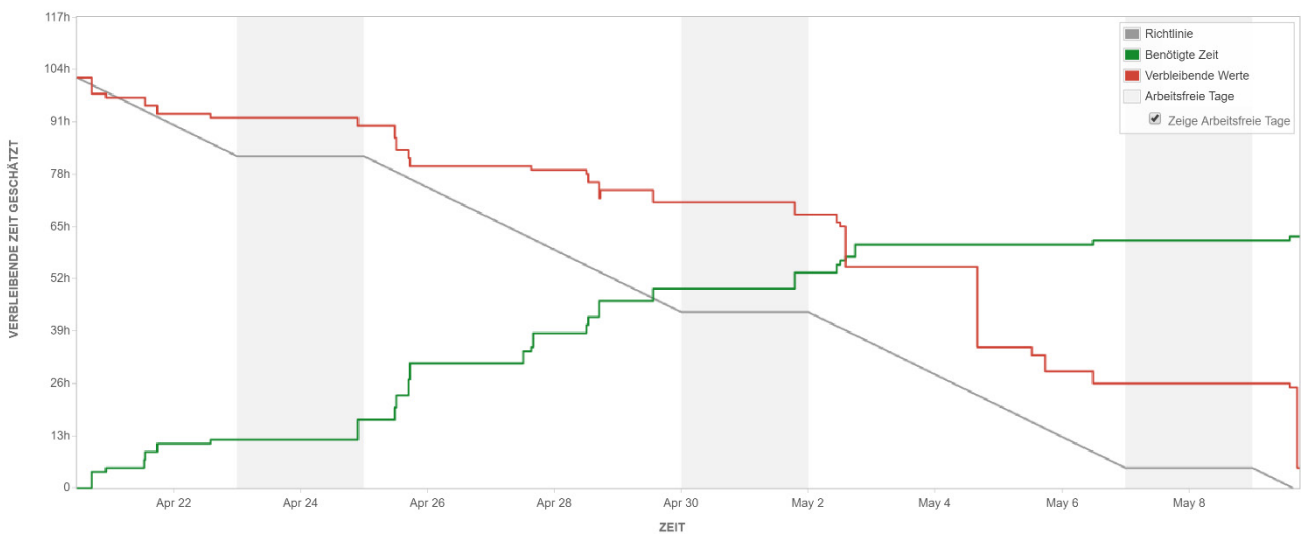


Abbildung 16: Burndown Diagramm Sprint 2

15.5 Sprint 3

Zeitraum: 09.05.2016 - 25.05.2016 (2 Wochen)

15.5.1 User-Story: BLS-41 Synchronisieren von BLUgis mit BLUshare (8 Story points)

Beschreibung:

- Der Synch-Vorgang wird manuell gestartet (zusammen mit dem bereits bestehenden Bereitstellungs-Vorgang).
- Daten werden ZUM und VOM BLUshare-Server geladen (Up- und Download).
- UPLOAD, d.h. Daten von BLUgis > BLUshare
- Alle meine Daten (POI, Event, Entry) mit gesetztem Flag "Synch mit BLUshare" werden zum Server geschickt (inkl. Attachment).
- im BLUgis geänderte/gelöschte eigene Daten werden auch im BLUshare geändert/gelöscht.
- BLUshare stellt sicher, dass jeder Benutzer nur SEINE Daten ändern/löschen kann. Fremde Daten können nicht geändert/gelöscht werden, auch wenn sie sich innerhalb des zugewiesenen EP's befinden.
- DOWNLOAD, d.h. Daten von BLUshare > BLUgis
- BLUgis erhält alle Daten von BLUshare, die innerhalb des dem verbindenden Benutzers zugewiesenen Lese-Perimeters (LP) liegen (inkl. Attachments).

Akzeptanzkriterien:

- Synch mit BLUshare möglich
- Daten inkl. Attachments aus EP kommen in BLUshare an
- Daten inkl. Attachments aus LP kommen in BLUgis an

Tasks:

- Upload-URL in GeoNode bereitstellen
- Datenspeicher in GeoServer erweitern
- Upload-Daten in GeoServer-Datenspeicher übernehmen
- Upload-Methode in BLUgis einbauen
- Download-Methode in BLUgis einbauen
- Download-URL in GeoNode bereitstellen
- Erfassungs-Perimeter bei Upload berücksichtigen
- Lese-Perimeter bei Download berücksichtigen
- Upload-Daten mit GeoServer-Datenspeicher abgleichen
- Attachments in Up- und Download einbeziehen

Zeitschätzung:

geschätzte Zeit: 36 Stunden

tatsächlich benötigte Zeit: 37 Stunden, 30 Minuten

15.5.2 User Story: Handling fremder Daten im lokalen BLUgis (5 Story points)

Beschreibung:

- Handhabung "fremder" Daten (POI, Event oder Entry) in BLUgis
- fremde Daten können nicht verändert werden (im GUI keine Möglichkeit)
- fremde Daten sind in der Detailansicht gekennzeichnet im Stil von "erhalten via BLUshare", "erfasst von [%USER%]"
- fremde Daten können am Admin-PC ausgeblendet werden --> benötigt wohl eine Anpassung im Datenmodell von BLUgis (boolean-Flag "ausgeblendet") z.B. über das Papierkorb-Icon (evtl. anderes Icon: durchgestrichenes Auge)
- ausgeblendete Daten sollen am Admin-PC wieder eingeblendet werden können
- ausgeblendete Daten werden auf allen Geräten (Tablets) in der Organisation ausgeblendet

Akzeptanzkriterien:

- fremde Daten können aus- und wieder eingeblendet werden
- fremde Daten sind in BLUgis gekennzeichnet (Herkunft BLUshare Erfasser)
- auch direkt in der DB geänderte fremde Daten (also nicht über BLUgis-GUI) werden von BLUshare nicht übernommen

Tasks:

- UI-Elemente readonly wenn fremde Daten in BLUgis
- Kennzeichnung der Daten, die von einem fremden User kommen (im GUI)
- Ausblenden von fremden Daten ermöglichen
- Icon für ausgeblendete Daten im GUI anzeigen lassen
- Ausgeblendete Daten werden auf allen Tablet-Geräten ausgeblendet

Zeitschätzung:

geschätzte Zeit: 27 Stunden

tatsächlich benötigte Zeit: 27 Stunden

Burndown-Diagramm

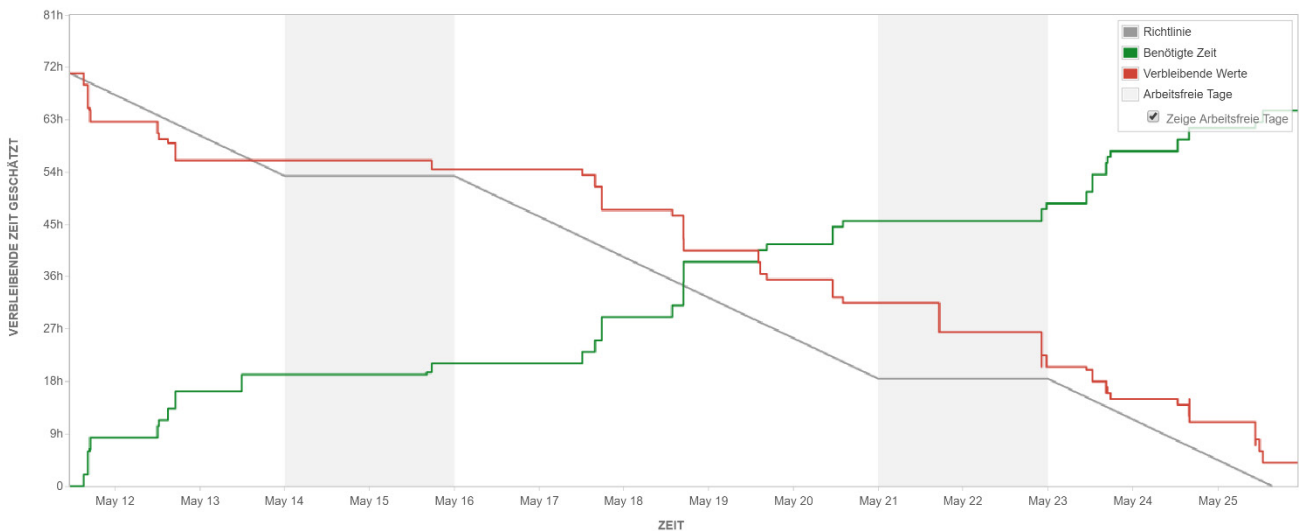


Abbildung 17: Burndown Diagramm Sprint 3

15.6 Sprint 4

Zeitraum: 25.05.2016 - 06.06.2016 (2 Wochen)

15.6.1 User-Story: Nachbesserungen vom Sprint 3

Tasks:

- Einblenden von ausgeblendeten Daten am Admin-PC ermöglichen
- Attachments in Up- und Download einbeziehen
- Ein-/Ausblenden fremder Daten über Attribut-Dialog
- Spezielle Darstellung fremder und ausgeblendeter Daten im Editiermodus

Zeitschätzung:

geschätzte Zeit: 19 Stunden

tatsächlich benötigte Zeit: 24 Stunden, 15 Minuten

15.6.2 User Story: Webclient

Beschreibung:

Bedienung im Webclient. Es soll möglichst gängige Standard Open-Source Software eingesetzt werden.

Tasks:

- Layer automatisiert in GeoServer hinzufügen
- Layer automatisiert in GeoNode hinzufügen
- GeoServer Cache leeren nach Datenupload
- Layer Permissions automatisiert setzen
- GeoNode-Editierfunktion aktivieren
- GeoNode Startseite anpassen

Zeitschätzung:

geschätzte Zeit: 15 Stunden

tatsächlich gebrauchte Zeit für die ersten 3 Tasks: 3 Stunden

15.6.3 User-Story: Standard-Export nach Excel

Beschreibung:

Als MEMBER (siehe BLS-24) kann ich mich im Web-Client anmelden und einen Download aller von mir erfassten Daten im CSV-Format machen. Dazu stehen mir durch den System-Admin vordefinierte Export-Definitionen zur Verfügung.

Die Export-Definition soll für den System-Admin eine gewisse Flexibilität vorsehen:

- mehrere Export-Definitionen erstellbar
- pro Export-Definition definierbare Spalten sowie deren Überschriften (für erste Export-Definition gem. angehängter Vorlage)
- eine dem Export vorgelagerte Filter-Möglichkeit (z.B. SQL) um die zu exportierenden Daten (ausgehend von allen eigenen Daten innerhalb des Erfassungs-Perimeters) weiter einzuschränken.

Akzeptanzkriterien:

- Als MEMBER kann ich im Web-Client vordefinierte CSV-Export-Definitionen auswählen.
- Als System-Admin kann ich Export-Definitionen erstellen, inkl. Spaltenwahl und SQL-Statement für Feinauswahl.

15.6.4 Tasks:

- Web-Client anpassen
- Export-Definition im Code ermöglichen
- PostGIS zu CSV Konverter implementieren

Diese User-Story konnte aus Zeitgründen in diesem Sprint nicht mehr bearbeitet werden.

Burndown-Diagramm

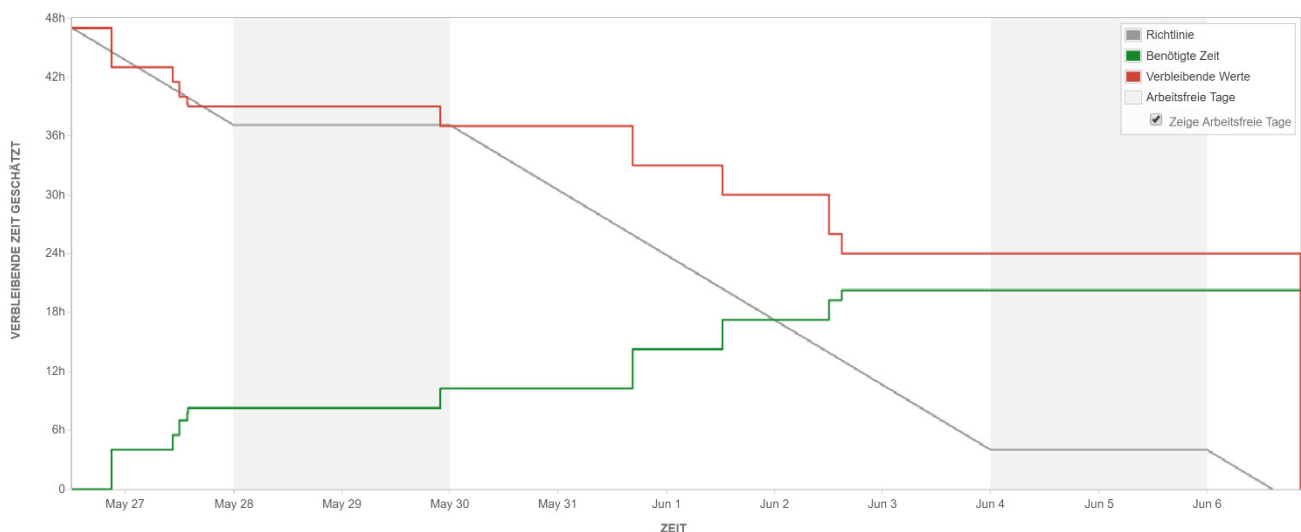


Abbildung 18: Burndown Diagramm Sprint 4

15.7 Abschlussphase

Zeitraum: 06.06.2016 - 17.06.2016

Fertigstellung Dokumentation

Code-Refactoring

Vorbereitung Präsentation

Abgabe des Berichts bis 17.06.2016 um 12.00 Uhr.

15.8 Präsentation

Datum: 17.06.2016 zwischen 16 und 20 Uhr.

15.9 Mündliche Prüfung

Die mündliche Prüfung zur Bachelorarbeit findet am 29.06.2016 statt.

16 Risikomanagement

Während dem Projekt können verschiedene Risikoszenarien eintreten, welche die Durchführung des Projektes behindern können oder den Ausgang negativ beeinflussen. Um mit solchen Risiken umgehen zu können, werden diese systematisch aufgelistet. Zudem werden Massnahmen überlegt, wie das Eintreten des jeweiligen Risikos frühzeitig verhindert werden kann bzw. wie bei Eintritt umgegangen wird.

16.1 Risiken

Im Folgenden sind verschiedene Risiken aufgeteilt in Kategorien aufgezählt, welche in diesem Projekt auftreten können.

16.1.1 Projektspezifisch

R1: Zu kompliziert zum Bedienen

R2: Gewählte Technologien erweisen sich als ungeeignet

R3: Datenbankschema genügt den Anforderungen nicht

R4: Ausfall HSR Infrastruktur

16.1.2 Mensch

R5: Ausfall Projektmitglied (z.B. Krankheit, Unfall...)

R6: Meinungsverschiedenheiten im Team

R7: Vergoldung: Zu viel Zeit in unwichtige Themen investiert / Überflüssiger Luxus

16.1.3 Infrastruktur

R8: Ausfall der Infrastruktur (Server mit Git/JIRA, Laptop)

R9: Datenverlust / Verlust von Dokumentation

16.1.4 Anforderungen

R10: Unklare Anforderungen

R11: Anforderungen werden falsch umgesetzt

16.1.5 Komplexität

R12: Lange Einarbeitungszeit / Schwierigkeiten mit neuen Technologien

R13: Fehleinschätzung Zeitaufwand

R14: Verwendung unreifer Technologie

16.2 Präventionsmassnahmen

R1: Frühzeitig Feedback durch Industriepartner einholen. Papierprototyp des User Interfaces frühzeitig erstellen, Testen an Hand des Papierprototypen.

- R2: Technologieanalyse möglichst früh im Projekt angehen
- R3: Priorisierung der Anforderungen, Machbarkeit prüfen
- R4: Regelmässig Backups erstellen, Daten sichern
- R5: -
- R6: -
- R7: Agile Vorgehensweise
- R8: Regelmässig Backups erstellen, Code regelmässig einchecken
- R9: Regelmässig Backups erstellen, Verwendung von Google Drive für die Dokumentationsverwaltung
- R10: Agile Vorgehensweise
- R11: Agile Vorgehensweise
- R12: Technologie evaluieren und Machbarkeit prüfen
- R13: Technologie evaluieren und Machbarkeit prüfen
- R14: Machbarkeit prüfen

16.3 Massnahmen bei Eintritt

- R1: Durch agile Vorgehensweise Bedienung schrittweise vereinfachen.
- R2: Möglichst bald geeignete Technologien finden, externe Personen um Rat fragen
- R3: Richtige Anforderungen als User Story im Backlog festlegen
- R4: -
- R5: Aktuelle Aufgaben des erkrankten Teammitgliedes delegieren
- R6: Schnell das Gespräch suchen. Wenn nötig Betreuer dazu nehmen.
- R7: Agile Vorgehensweise ermöglichen
- R8: Laptop neu aufsetzen, Wiederherstellung des letzten Commits des Repositories
- R9: -
- R10: Anforderungen klarer definieren und priorisieren
- R11: Umsetzung der Anforderungen im Backlog festlegen
- R12: Experten, die sich mit der Technologie auskennen, einbeziehen
- R13: Priorisierung mit Kunde, Verschieben von User Stories in den nächsten Sprint
- R14: Geeignete Technologien finden

16.4 Eingetretene Risiken

Während des Projekts sind folgende Risiken eingetreten:

16.4.1 R12: Komplexität - Schwierigkeiten mit neuen Technologien

Beschreibung: Die Analyse und Evaluation der Technologien hat viel Zeit gekostet.

Eintrittswahrscheinlichkeit: hoch

Schaden: mittel

Massnahmen: Beschränkung auf wenige Technologien und wesentliche Evaluationskriterien.

16.4.2 R5: Ausfall Projektmitglied

Beschreibung: Am Anfang des Projekts war ein Projektmitglied für eine Woche krank.

Eintrittswahrscheinlichkeit: mittel

Schaden: sehr gering (am Anfang des Projekts)

Massnahmen: Arbeitspensum für die Woche reduzieren, Aufgaben an anderes Teammitglied delegieren

16.4.3 R13: Fehleinschätzung Zeitaufwand / Komplexität

Beschreibung:

Während der Entwicklung von BLUshare wurde der Zeitaufwand von Tasks teilweise unter- oder überschätzt. Zum Beispiel war die Installation des Quellcodes von GeoNode zeitaufwändig, da es verschiedene zusätzliche Software-Libraries (fehlende Abhängigkeiten) brauchte um GeoNode vollständig zu installieren und zum Laufen zu bringen. Weiterhin traten unklare Fehlermeldungen auf. Teilweise mussten Schritte der Installation wiederholt werden, wenn etwas angepasst werden musste oder eine neue Version von GeoNode herauskam.

Im Sprint 2 konnte die User-Story «Synchronisation von BLUgis mit BLUshare» auf Grund der Zeit nicht abgeschlossen werden.

Eintrittswahrscheinlichkeit: mittel

Schaden: mittel

Massnahmen: User Stories in den nächsten Sprint verschieben

Die im 2. Sprint nicht mehr bearbeitete User Story wurde in den 3. Sprint verschoben und konnte im neuem Sprint abgeschlossen werden.

17 Qualitätssicherung

Damit ein Quellcode von einem anderen Entwickler, der möglicherweise die Software weiterentwickeln möchte, verstanden werden kann, ist es sehr empfehlenswert sich an gewisse Richtlinien für einen guten Programmierstil zu halten. Die Einhaltung solcher Richtlinien erhöhen die Lesbarkeit des Quellcodes.

Folgende relevante Richtlinien stammen aus den PEP-8 Codierrichtlinien: [79]

- Eine Zeile hat höchstens 79 Zeichen (empfohlen).
- Der Abstand zwischen Methoden in einer Klasse sollte 1 Leerzeile sein.
- Der Abstand zwischen Klassen in einem Modul sollte 2 Leerzeilen sein.
- Es sollte 1 Import pro Zeile stehen.
- Imports sollten in der Datei über globale Variablen des Moduls stehen.
- Bei den Imports sollten zuerst die Module aus Python importiert werden, dann Module aus Drittanbieter-Libraries und zum Schluss Module aus dem Projekt.
- Zwischen jeder Gruppe von Imports sollte sich eine Leerzeile befinden.
- Wildcard Imports z.B. `from <Module> import *` sollten vermieden werden, da damit nicht klar wird, welche Libraries aus dem Modul verwendet werden.

PyCharm hat ein bereits eingebautes Code Inspection Tool, welches die Einhaltung der PEP-8 Richtlinien überprüft. Verstöße gegen diese Richtlinien werden als Warnung ausgegeben.

Teil IV: Softwaredokumentation

18 Installationsanleitung

18.1 Voraussetzungen

Die allgemeinen Systemanforderungen sind im Kapitel «GeoNode» bzw. im Web [80] aufgelistet. Die folgende Installationsanleitung bezieht sich auf den Betrieb von BLUshare auf einem Ubuntu-Server in der Version 14.04, auf dem Plesk als Webhosting-Konfigurationstool eingesetzt wird. Eine allgemeinere Installationsanleitung ist in der Online-Dokumentation von GeoNode vorhanden. [81]

18.2 Installation GeoNode mit BLUshare

Die folgende Installationsanleitung für BLUshare wurde ausgehend von der Online-Anleitung von GeoNode [81] erstellt und in verkürzter Form aufgelistet. Die Online-Dokumentation enthält eine detailliertere Beschreibung der einzelnen Schritte. Zeilen die mit \$ beginnen enthalten eine Anweisung, welche in der Linux-Shell eingegeben werden soll.

18.2.1 Software-Abhängigkeiten installieren

Installationen über das Paketverwaltungssystem:

```
$ sudo apt-get install \
  python-virtualenv \
  build-essential \
  openssh-server \
  apache2 \
  gcc \
  gdal-bin \
  gettext \
  git-core \
  libapache2-mod-wsgi \
  libgeos-dev \
  libjpeg-dev \
  libpng-dev \
  libpq-dev \
  libproj-dev \
  libxml2-dev \
  libxslt-dev \
  openjdk-7-jre \
  patch \
  postgresql \
  postgis \
  postgresql-9.3-postgis-scripts \
  postgresql-contrib \
  python \
  python-dev \
  python-gdal \
  python-pycurl \
  python-imaging \
  python-pastescript \
  python-psycopg2 \
  python-support \
```

```
python-urlgrabber    \  
python-virtualenv    \  
tomcat7              \  
unzip                \  
zip
```

Installation von PostgreSQL-Version \geq 9.5: [82]

BLUshare verwendet an verschiedenen Stellen UPSERT-Befehle für das Einfügen von Daten in die Datenbank, welche erst ab PostgreSQL 9.5 verfügbar sind. [83] Diese Version ist aber standardmässig nicht im Paketverwaltungssystem von Ubuntu 14.04 enthalten, deshalb sind noch folgende Schritte notwendig:

```
$ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg  
main" >> /etc/apt/sources.list.d/pgdg.list'  
$ wget -q https://www.postgresql.org/media/keys/ACCC4CF8.asc -O - | sudo apt-key add -  
$ sudo apt-get update  
$ sudo apt-get install postgresql postgresql-contrib
```

Reparatur von PostGIS der PostgreSQL 9.3 Installation: [84]

```
$ apt-get remove postgresql-9.3-postgis-2.1 liblwgeom-2.1.2 postgis  
$ apt-get install postgresql-9.3-postgis-2.1
```

Installation von PostGIS für PostgreSQL 9.5:

```
$ sudo apt-get install postgresql-9.5-postgis-2.2
```

Im Folgenden wird angenommen, dass für PostgreSQL 9.5 der Port 5433 verwendet wird (wobei der Standardport 5432 von PostgreSQL 9.3 belegt wird). Wurde bei der Installation von PostgreSQL 9.5 ein anderer Port vergeben, muss das in den folgenden Schritten entsprechend angepasst werden.

18.2.2 BLUshare-Download

GeoNode-Systembenutzer erstellen:

```
$ sudo useradd -m geonode  
$ cd /home/geonode/
```

BLUshare-Download vom Git-Repository:

```
$ sudo git clone -b develop --recursive http://localhost:28080/scm/bls/blushare-dev.git
```

Installation von GeoNode-Abhängigkeiten:

```
$ cd /home/geonode/blushare-dev/GeoNodeSource/  
$ sudo pip install -e .  
$ sudo paver setup
```

18.2.3 Datenbankkonfiguration

Datenbankbenutzer erstellen:

```
$ sudo -u postgres createuser -P blushare -p 5433
```

(Passwort: blushare)

Datenbanken vorbereiten:

```
$ sudo -u postgres createdb -O blushare geonode -p 5433
```

```
$ sudo -u postgres createdb -O blushare geonode_data -p 5433
$ sudo su postgres
$ psql -c 'ALTER USER geonode SUPERUSER;'
$ psql -c 'ALTER USER blushare SUPERUSER;' -p 5433
$ psql -d geonode -c 'CREATE EXTENSION postgis;' -p 5433
$ psql -d geonode -c 'GRANT ALL ON geometry_columns TO PUBLIC;' -p 5433
$ psql -d geonode -c 'GRANT ALL ON spatial_ref_sys TO PUBLIC;' -p 5433
$ psql -d geonode_data -c 'CREATE EXTENSION postgis;' -p 5433
$ psql -d geonode_data -c 'GRANT ALL ON geometry_columns TO PUBLIC;' -p 5433
$ psql -d geonode_data -c 'GRANT ALL ON spatial_ref_sys TO PUBLIC;' -p 5433
$ exit
```

Zugriffskontrolle anpassen:

```
$ sudo vi /etc/postgresql/9.5/main/pg_hba.conf
```

Die Zeilen

```
-----
# "local" is for Unix domain socket connections only
local  all          all                                peer
-----
```

abändern zu

```
-----
# "local" is for Unix domain socket connections only
local  all          all                                trust
-----
```

```
$ sudo service postgresql restart
```

18.2.4 Plesk-Konfiguration

Domain in Plesk hinzufügen:

Plesk → Websites & Domains → Subdomain hinzufügen

Name der Subdomain: blushare

Übergeordnete Domain: blugis.ch

Apache konfigurieren:

Plesk → Websites & Domains → blushare.blugis.ch → Einstellungen für Apache & nginx

Zusätzliche Anweisungen für HTTP:

```
-----
Redirect / https://blushare.blugis.ch/
-----
```

Zusätzliche Anweisungen für HTTPS:

```
WSGIDAemonProcess geonode python-path=/home/geonode/blushare-dev/GeoNodeSource:/home/geonode/blushare-dev/.venvs/geonode/lib/python2.7/site-packages user=www-data threads=15 processes=2
```

```
WSGIProcessGroup geonode
WSGIPassAuthorization On
WSGIScriptAlias / /home/geonode/blushare-dev/GeoNodeSource/geonode/wsgi.py
```

```
Alias /static/ /home/geonode/blushare-dev/GeoNodeSource/geonode/static_root/
Alias /uploaded/ /home/geonode/blushare-dev/GeoNodeSource/geonode/uploaded/
```

```
<Directory "/home/geonode/blushare-dev/GeoNodeSource/geonode/">
    <Files wsgi.py>
        Order deny,allow
        Allow from all
        Require all granted
    </Files>

    Order allow,deny
    Options Indexes FollowSymLinks
    Allow from all
    IndexOptions FancyIndexing
</Directory>
```

```
<Directory "/home/geonode/blushare-dev/GeoNodeSource/geonode/static_root/">
    Order allow,deny
    Options Indexes FollowSymLinks
    Allow from all
    Require all granted
    IndexOptions FancyIndexing
</Directory>
```

```
<Directory "/home/geonode/blushare-dev/GeoNodeSource/geonode/uploaded/thumbs/">
    Order allow,deny
    Options Indexes FollowSymLinks
    Allow from all
    Require all granted
    IndexOptions FancyIndexing
</Directory>
```

```
<Directory "/home/geonode/blushare-dev/GeoNodeSource/geonode/uploaded/layers/">
    Order allow,deny
    Options Indexes FollowSymLinks
    Allow from all
    Require all granted
    IndexOptions FancyIndexing
</Directory>
```

```
<Proxy *>
    Order allow,deny
    Allow from all
</Proxy>
```

```
ProxyPreserveHost On
ProxyPass /geoserver http://127.0.0.1:8080/geoserver
ProxyPassReverse /geoserver http://127.0.0.1:8080/geoserver
```

Optional:

Plesk -> Websites & Domains -> blushare.blugis.ch -> File Manager

Die vorhandenen Dateien löschen, weil diese nicht benötigt werden.

18.2.5 GeoNode- und BLUshare-Vorbereitungen

GeoNode:

```
$ cd /home/geonode/blushare-dev/GeoNodeSource/
$ sudo python manage.py collectstatic
$ sudo mkdir -p /home/geonode/blushare-dev/GeoNodeSource/geonode/uploaded/thumbs
$ sudo mkdir -p /home/geonode/blushare-dev/GeoNodeSource/geonode/uploaded/layers
$ sudo chown -R geonode /home/geonode/blushare-dev/GeoNodeSource/
$ sudo chown geonode:www-data /home/geonode/blushare-dev/GeoNodeSource/geonode/static/
$ sudo chown -R geonode:www-data /home/geonode/blushare-dev/GeoNodeSource/geonode/up-
loaded/
$ chmod -Rf 777 /home/geonode/blushare-dev/GeoNodeSource/geonode/uploaded/thumbs
$ chmod -Rf 777 /home/geonode/blushare-dev/GeoNodeSource/geonode/uploaded/layers
$ sudo chown www-data:www-data /home/geonode/blushare-
dev/GeoNodeSource/geonode/static_root/
$ sudo chmod -R 777 /home/geonode/blushare-dev/GeoNodeSource/geonode/uploaded/thumbs
```

BLUshare:

```
$ sudo mkdir -p /home/geonode/blushare-dev/BLUshareSource/data
$ sudo chown geonode:www-data /home/geonode/blushare-dev/BLUshareSource/data
$ sudo chmod -R g+w /home/geonode/blushare-dev/BLUshareSource/data
```

18.2.6 GeoServer-Konfiguration

```
$ sudo service tomcat7 stop
$ sudo cp /home/geonode/blushare-dev/GeoNodeSource/downloaded/geoserver.war
/var/lib/tomcat7/webapps/
$ sudo service tomcat7 start
$ sudo service tomcat7 stop
$ sudo vi /var/lib/tomcat7/webapps/geoserver/WEB-INF/web.xml
```

Folgendes nach der Zeile "</display-name>" einfügen:

```
<context-param>
  <param-name>GEONODE_BASE_URL</param-name>
  <param-value>https://blushare.blugis.ch/</param-value>
</context-param>
```

```
$ sudo service tomcat7 start
```

18.2.7 GeoNode-Konfiguration

```
$ cd /home/geonode/blushare-dev/GeoNodeSource/geonode/  
$ sudo cp local_settings.py.sample local_settings.py  
$ sudo vi local_settings.py
```

Add after "SITEURL..."-line

```
-----  
ALLOWED_HOSTS = ['127.0.0.1', 'localhost', ':::1']  
PROXY_ALLOWED_HOSTS = ("127.0.0.1", 'localhost', ':::1')  
POSTGIS_VERSION = (2, 2, 2)  
-----
```

Konfiguration des Datenbank-Zugangs:

```
-----  
DATABASES = {  
    'default': {  
        'ENGINE': 'django.contrib.gis.db.backends.postgis',  
        ...  
    },  
    'datastore' : {  
        'ENGINE': 'django.contrib.gis.db.backends.postgis',  
        #'ENGINE': '', # Empty ENGINE name disables,  
        'NAME': 'geonode_data',  
        ...  
    }  
}  
OGC_SERVER = {  
    'default' : {  
        'BACKEND' : 'geonode.geoserver',  
        'LOCATION' : 'http://localhost:8080/geoserver/',  
        'PUBLIC_LOCATION' : 'https://blushare.blugis.ch/geoserver/',  
        ...  
    }  
}  
-----
```

Superuser erstellen und Datenbank initialisieren:

```
$ cd /home/geonode/blushare-dev/GeoNodeSource/  
$ sudo rm -R /home/geonode/blushare-dev/GeoNodeSource/geonode/people/migrations  
$ sudo rm -R /home/geonode/blushare-dev/GeoNodeSource/geonode/groups/migrations  
$ python manage.py migrate auth  
$ python manage.py migrate  
$ python manage.py createsuperuser
```

Benutzername, E-Mail und Passwort angeben.

```
$ sudo service apache2 restart
```

19 Benutzerhandbuch

19.1 BLUshare-Webclient

19.1.1 Hinzufügen eines Benutzers

Um einen Benutzer hinzuzufügen muss man als Administrator angemeldet sein.

Über das Administrator-Interface kann ein Benutzer hinzugefügt werden.

1. Loggen Sie sich als Administrator ein.
2. Klicken Sie auf ihren Benutzernamen. Es erscheint ein Menü. Klicken Sie auf «Admin», um das Administrator-Interface aufzurufen.

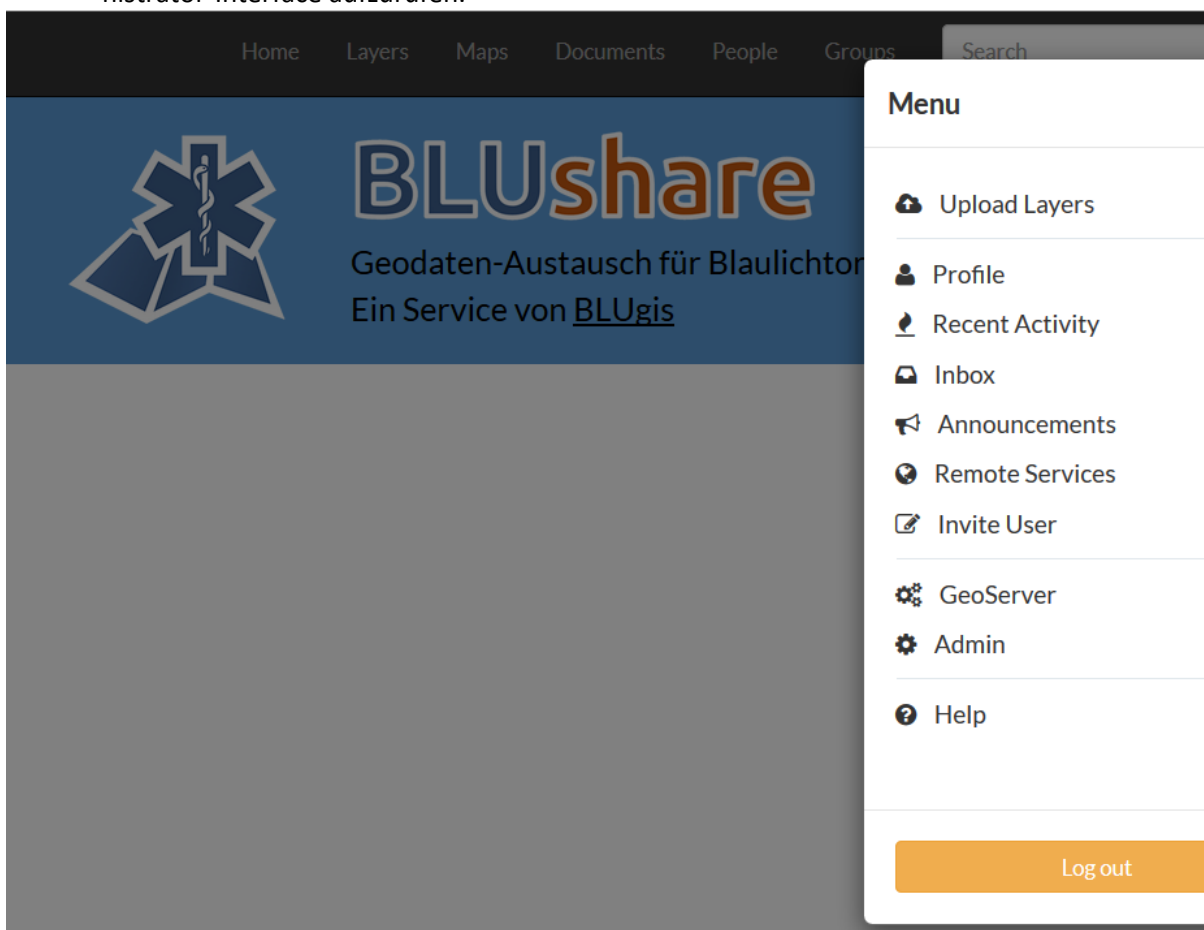


Abbildung 19: Menüansicht GeoNode

3. Scrollen Sie zu «People» und klicken Sie auf «Hinzufügen».



Abbildung 20: Hinzufügen eines Benutzers

4. Es erscheint die folgende Eingabemaske:

GeoNode Verwaltung

Startseite > People > Benutzer > Hinzufügen Benutzer

Benutzer hinzufügen

Zuerst einen Benutzer und ein Passwort eingeben. Danach können weitere Optionen für den Benutzer geändert werden.

Benutzer:	<input type="text"/>
	<small>Erforderlich. 30 Zeichen oder weniger. Nur alphanumerische Zeichen (Buchstaben, Ziffern und @/./+/-/_) sind erlaubt.</small>
Kennwort:	<input type="password"/>
Passwort bestätigen:	<input type="password"/>
	<small>Bitte das gleiche Passwort zur Überprüfung nochmal eingeben.</small>

Abbildung 21: Eingabe eines Benutzers

5. Geben Sie den Benutzernamen und ein Passwort ein. Das Passwort muss zur Bestätigung ein weiteres Mal eingegeben werden. Es sind maximal 30 Zeichen für den Benutzernamen erlaubt.
6. Es gibt 3 Möglichkeiten, um fortzufahren:

- «Sichern und neu hinzufügen»
- «Sichern und weiter bearbeiten»
- «Speichern»

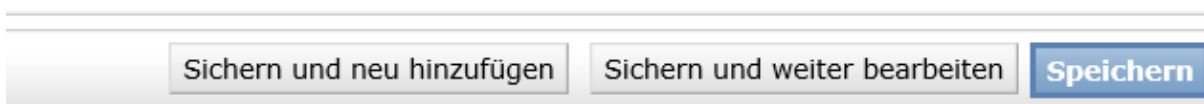


Abbildung 22: Möglichkeiten zum Speichern

7. Klicken Sie auf «Sichern und weiter bearbeiten», um weitere Daten hinzuzufügen.
8. Sie können den Vornamen, Nachnamen und die E-Mail-Adresse angeben.
9. Weiterhin können Sie dem Benutzer folgende Rechte zuweisen: Aktiv (Active), Redakteur-Status (Staff) und Administrator-Status (Superuser)

Berechtigungen

Aktiv
Legt fest, ob dieser Benutzer aktiv ist. Kann deaktiviert werden, anstatt Benutzer zu löschen.

Redakteur-Status
Legt fest, ob sich der Benutzer an der Administrationsseite anmelden kann.

Administrator-Status
Legt fest, dass der Benutzer alle Berechtigungen hat, ohne diese einzeln zuweisen zu müssen.

Abbildung 23: Berechtigungen eines Benutzers in GeoNode

- **Aktiv** legt fest, ob der Benutzer aktiv ist. Der Benutzer kann deaktiviert werden, aber wird nicht gelöscht.
- **Redakteur-Status** (Staff) legt fest, ob der Benutzer auf das Administrator Interface zugreifen kann.
- Mit einem **Administrator-Status** enthält ein Benutzer alle Rechte.

10. Unter «Gruppen» kann der Benutzer einer Gruppe hinzugefügt werden. Der Benutzer bekommt die gleichen Rechte, wie die Gruppe.

11. Unter «erweitertes Profil» können weitere Daten angegeben werden:

- Name der Firma
- Beschreibung des Benutzers
- Position des Benutzers innerhalb der Firma
- Telefonnummer
- Faxnummer
- E-Mail-Adresse
- Ort
- Kanton
- Postleitzahl
- Land
- Keywords

12. Wenn Sie fertig sind, klicken Sie auf Speichern. Weiterhin bestehen die Optionen «Sichern und neu hinzufügen» und «Sichern und weiter bearbeiten».

19.1.2 Ändern eines Benutzers

1. Klicken Sie bei «People» auf «Ändern». Es erscheint eine Liste mit den Benutzern.
2. Klicken Sie auf den Benutzernamen, um den Benutzer zu ändern. Es erscheint das Formular zum Ändern eines Nutzers. Es können sämtliche Eigenschaften (siehe «Benutzer hinzufügen») geändert werden.

19.1.3 Erstellen einer Benutzergruppe

Um eine Benutzergruppe zu erstellen müssen Sie als Administrator angemeldet sein.

1. Klicken Sie im Admin-Interface bei «Group profiles» auf «Hinzufügen». Es erscheint die folgende Eingabemaske:

Add group profile

Title:	<input type="text"/>
Slug:	<input type="text"/>
Logo:	<input type="button" value="Durchsuchen..."/> Keine Datei ausgewählt.
Description:	<div style="border: 1px solid #ccc; height: 100px;"></div>
Email:	<input type="text"/> <small>Email used to contact one or all group members, such as a mailing list, shared email, or exchange group.</small>
Access:	<input type="text" value="Public"/> <input type="button" value="v"/>

Abbildung 24: Hinzufügen einer Gruppe

2. Geben Sie einen Titel und einen Slug für die Gruppe ein. Sie können ein Logo hinzufügen.
3. Im Feld «Description» können Sie eine Beschreibung ihrer Gruppe hinzufügen.
4. Geben Sie eine E-Mail-Adresse Ihrer Gruppe an.
5. Legen Sie den Zugriff für diese Gruppe fest. Es gibt 3 Möglichkeiten:
 - öffentlich (public): Jeder registrierte Benutzer kann die Gruppe sehen und der Gruppe beitreten.
 - öffentlich (nur auf Einladung): Nur eingeladene Benutzer können der Gruppe beitreten.
 - privat
6. Legen Sie für die Gruppe einen Erfassungssperimeter (Region with Create permission) fest, indem Sie ein Polygon in die Karte zeichnen.
7. Legen Sie für die Gruppe einen Leseperimeter (Region with Read permission) fest, indem Sie ein Polygon in die Karte zeichnen.
8. Geben Sie Schlüsselwörter für diese Gruppe an.
9. Zum Schluss fügen Sie die Gruppenmitglieder der Gruppe hinzu und weisen Sie jedes Mitglied eine Rolle zu. Als Rolle können entweder «Manager» oder «Mitglied» ausgewählt werden.

Group members		
User	Role	Joined
<input type="text" value="....."/> <input type="button" value="v"/> <input type="button" value="+"/>	<input type="text" value="....."/> <input type="button" value="v"/>	Datum: <input type="text" value="2016-04-21"/> Heute <input type="button" value="📅"/> Zeit: <input type="text" value="04:04:20"/> Jetzt <input type="button" value="🕒"/>
<input type="text" value="....."/> <input type="button" value="v"/> <input type="button" value="+"/>	<input type="text" value="....."/> <input type="button" value="v"/>	Datum: <input type="text" value="2016-04-21"/> Heute <input type="button" value="📅"/> Zeit: <input type="text" value="04:04:20"/> Jetzt <input type="button" value="🕒"/>
<input type="text" value="....."/> <input type="button" value="v"/> <input type="button" value="+"/>	<input type="text" value="....."/> <input type="button" value="v"/>	Datum: <input type="text" value="2016-04-21"/> Heute <input type="button" value="📅"/> Zeit: <input type="text" value="04:04:20"/> Jetzt <input type="button" value="🕒"/>

Abbildung 25: Mitglieder und Rollen einer Gruppe

19.1.4 Ändern einer Benutzergruppe

1. Klicken Sie im Admin-Interface auf «Group Profiles». Es erscheint eine Liste mit den bestehenden Gruppen.

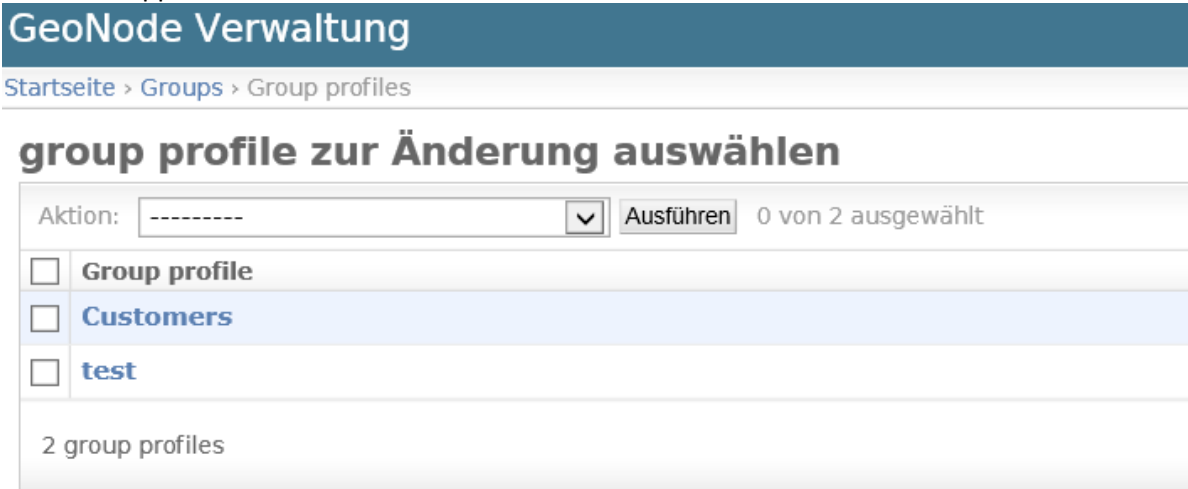


Abbildung 26: Liste der Gruppen

2. Klicken Sie auf eine Gruppe um sie zu ändern.

19.2 BLUgis

19.2.1 Synchronisieren von Daten mit BLUshare

Daten ins BLUshare heraufladen

Ausgangslage: Sie haben Daten in BLUgis erfasst. Diese Daten sollen auf BLUshare übertragen werden.

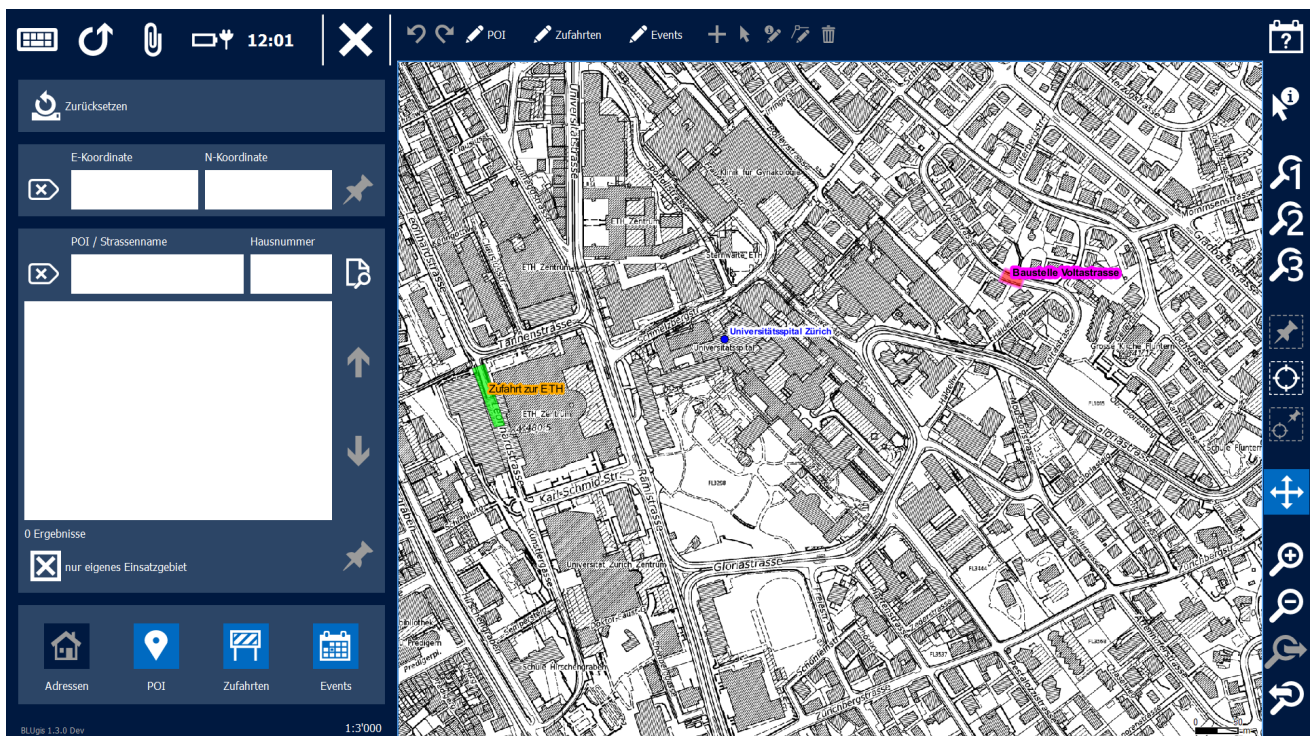


Abbildung 27: Erfasste Daten in BLUgis

1. Klicken Sie auf den Pfeil oben links in der obersten Leiste (siehe Abbildung 27), um die Daten auf BLUshare zu laden. Wenn der Upload erfolgreich war, wird dies bestätigt.

Daten von BLUshare zu BLUgis laden

Ausgangslage: Sie sind in einem fremden Einsatzgebiet unterwegs und möchten die Daten dieses Gebiets auf Ihr Tablet laden.

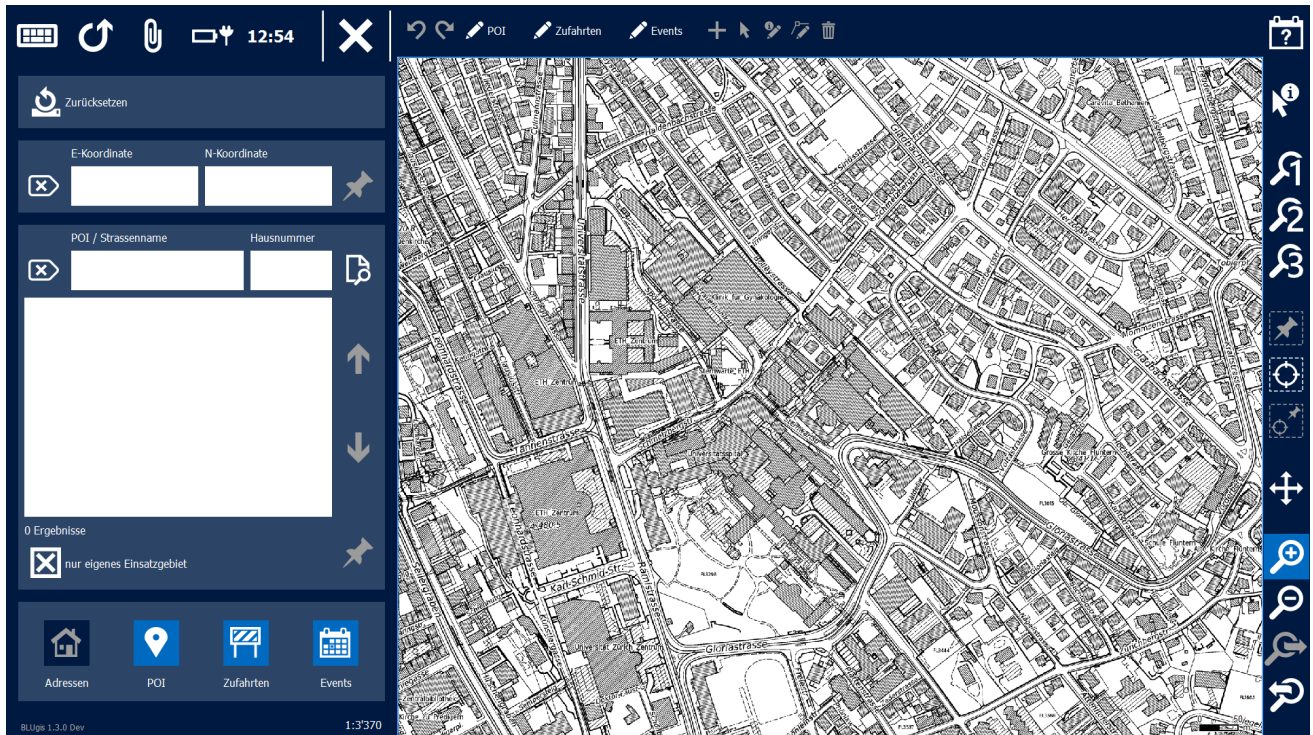


Abbildung 28: BLUgis ohne Daten

1. Klicken Sie auf dem Pfeil oben links, in der obersten Leiste (siehe Abbildung 28), um Daten von BLUshare auf Ihr BLUgis-Tablet herunterzuladen. Wenn der Download erfolgreich war, wird dies bestätigt. Die Daten erscheinen auf der Karte und sind grau gefärbt.

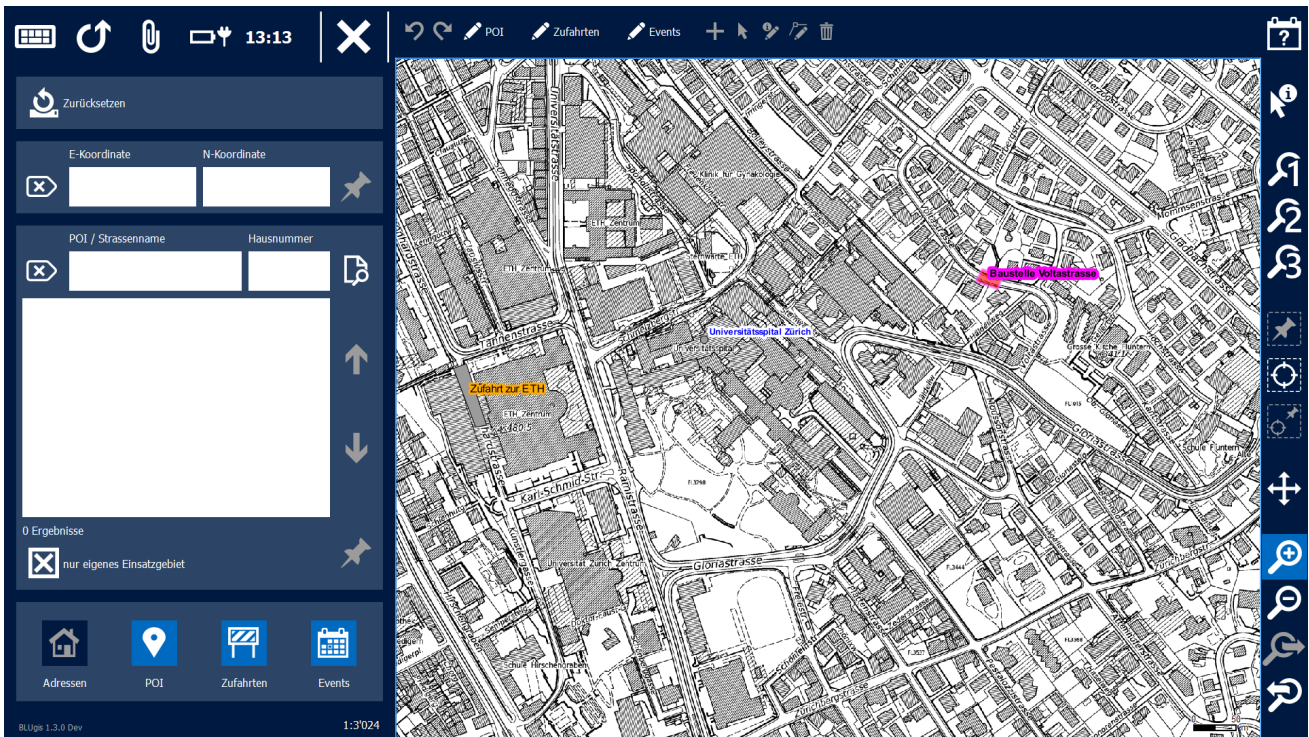


Abbildung 29: Erhaltene Daten via BLUshare

19.2.2 Ein- und Ausblenden von fremden Daten

Fremde Daten, die mit BLUshare erhalten wurden, können ausgeblendet und wieder eingeblendet werden.

Ausblenden fremder Daten

1. Wechseln Sie in den Editiermodus, indem Sie auf «POI», «Zufahrten» oder «Events» und dann auf das Symbol mit dem Stift und dem I-Symbol klicken.

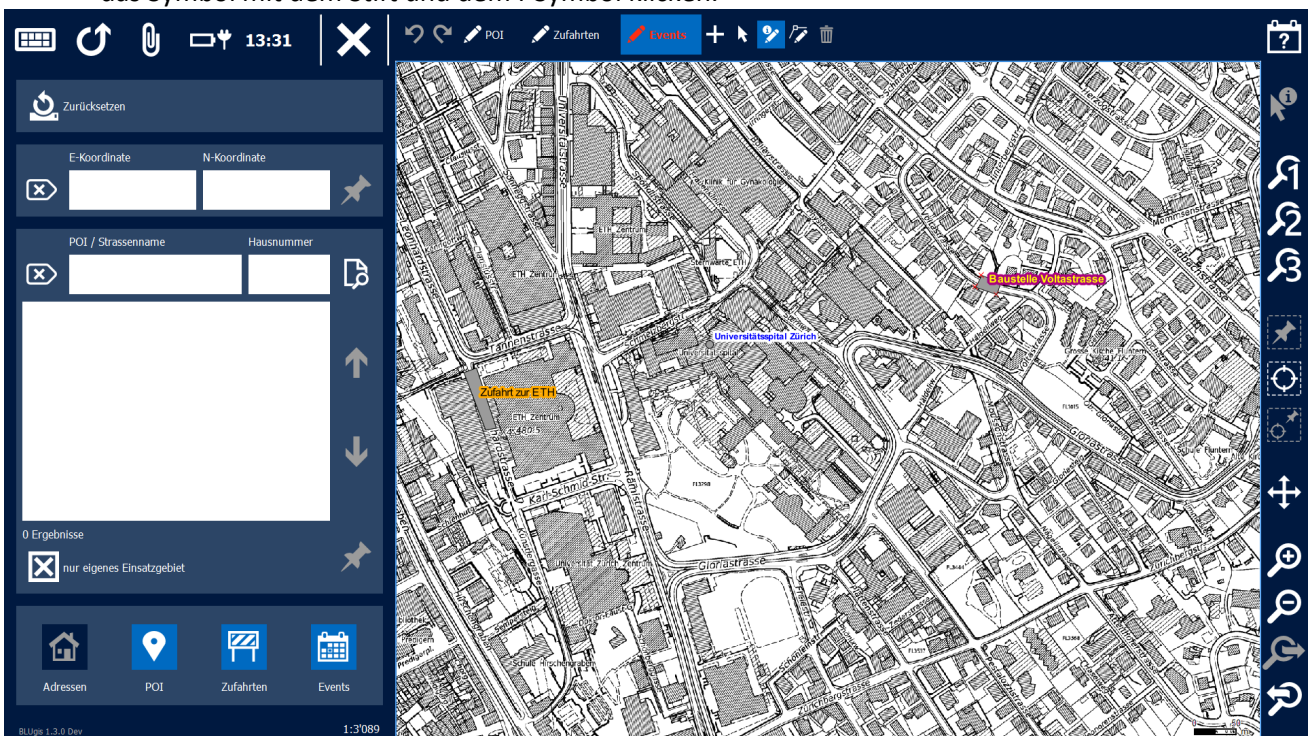


Abbildung 30: BLUgis Bearbeitungsmodus

2. Wählen Sie ein Element aus, das Sie ausblenden möchten. Es erscheint ein Fenster mit den Details des Elements. Aus dem Detailfenster wird ersichtlich, von welchem Benutzer das auszublenkende Objekt stammt (siehe Abbildung 29). Beachten Sie, dass im vorherigen Schritt auf «POI», «Zufahrten» oder «Events» geklickt werden muss (je nachdem welches Element Sie ausblenden möchten). Daher können Sie nur die Elemente des entsprechenden Typs ausblenden. Wenn Sie im vorherigen Schritt beispielsweise auf «Events» geklickt haben, können Sie nur «Events» ausblenden (siehe Abbildung 30).

3. Klicken Sie im Detailfenster auf «Ausblenden», um das Objekt auszublenden (siehe Abbildung 31). Das Objekt wird ausgeblendet und das Detailfenster färbt sich grau. Das ausgeblendete Objekt erscheint als schraffierte Fläche auf der Karte (siehe Abbildung 32).

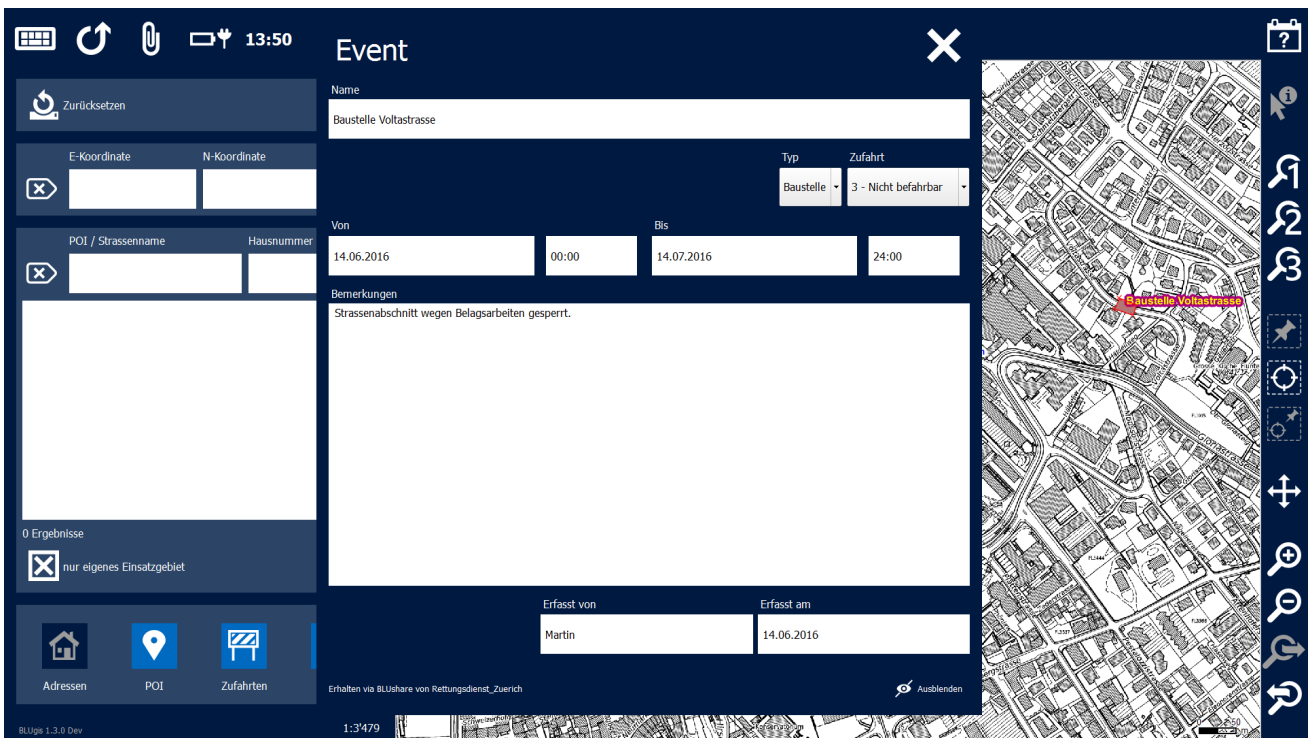


Abbildung 31: Detailfenster eines auszublendenden Objekts

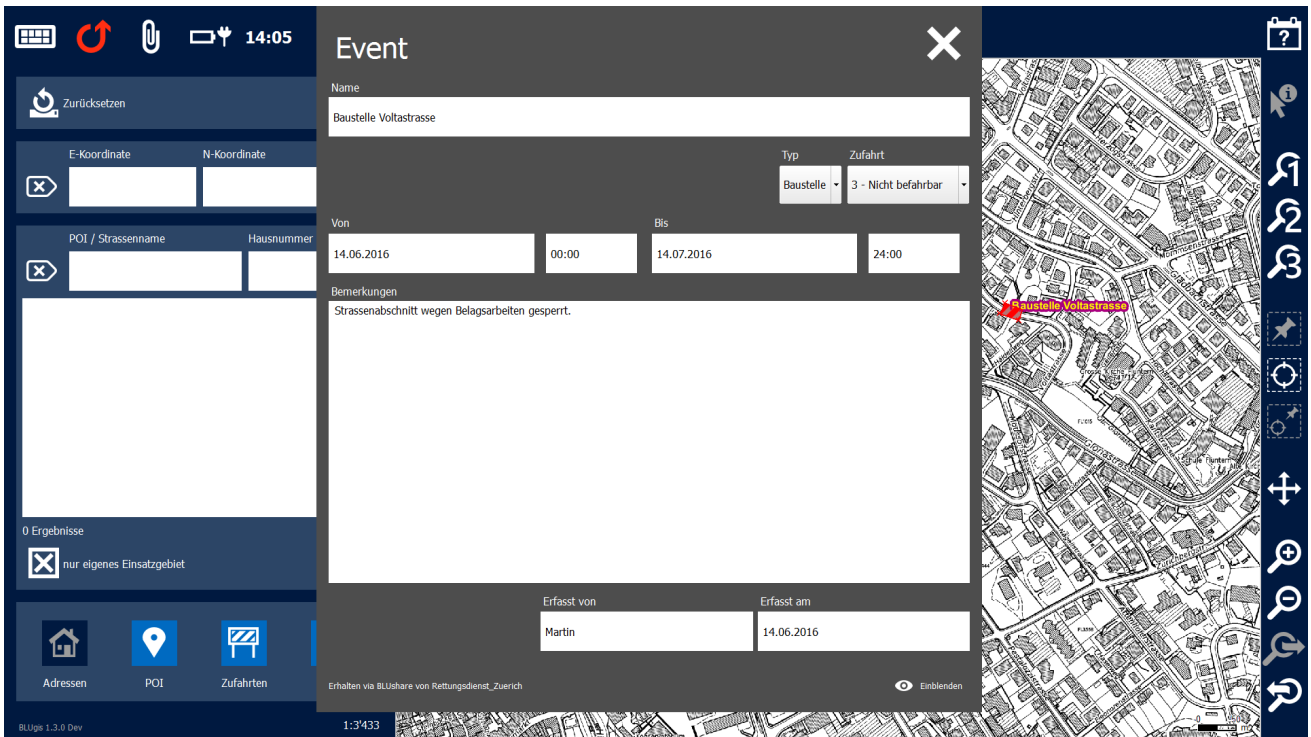


Abbildung 32: BLUgis - Detailfenster des ausgeblendeten Objekts

4. Schliessen Sie das Detailfenster und beenden Sie den Editiermodus. Das ausgeblendete Objekt verschwindet von der Karte (siehe Abbildung 33).

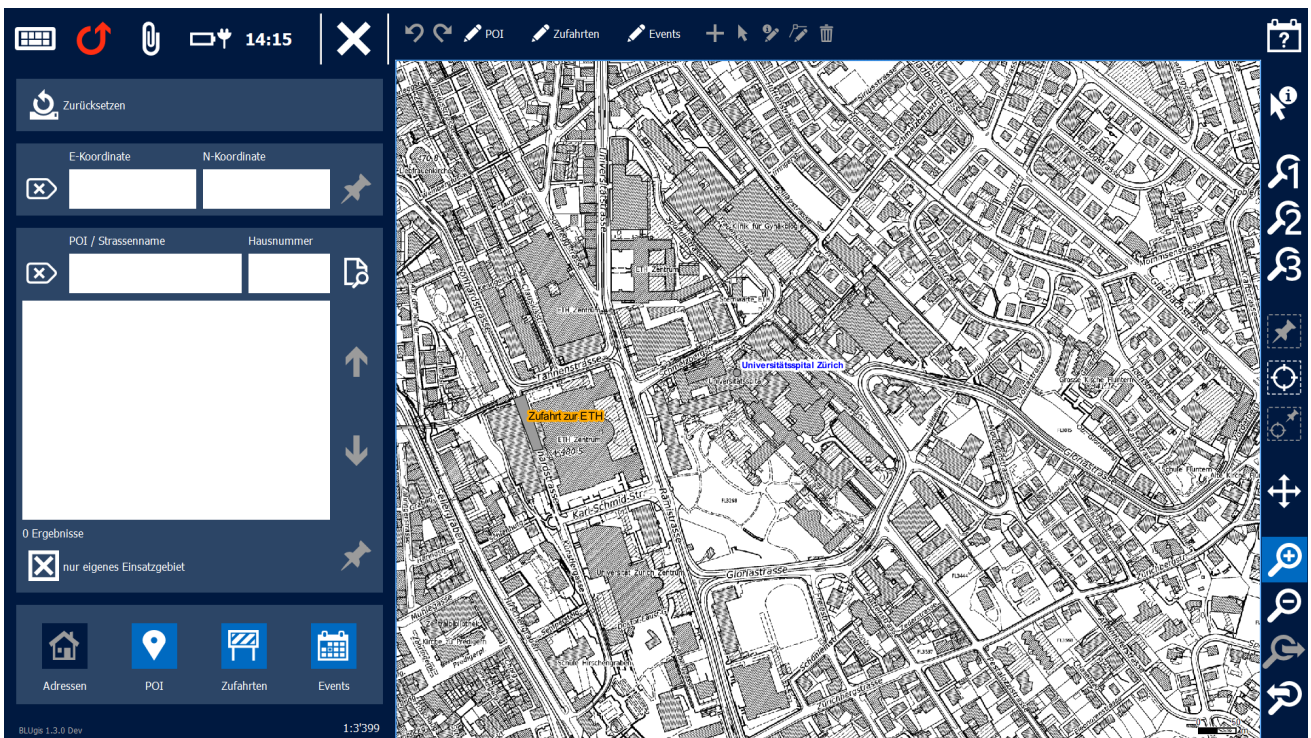


Abbildung 33: BLUgis mit ausgeblendetem Event

Einblenden fremder Daten

1. Wechseln Sie in den Editiermodus indem Sie auf «Zufahrten», «Events» oder «POI» klicken. Je nach Typ des Objekts erscheinen ausgeblendete Zufahrten, Events oder POIs auf der Karte.

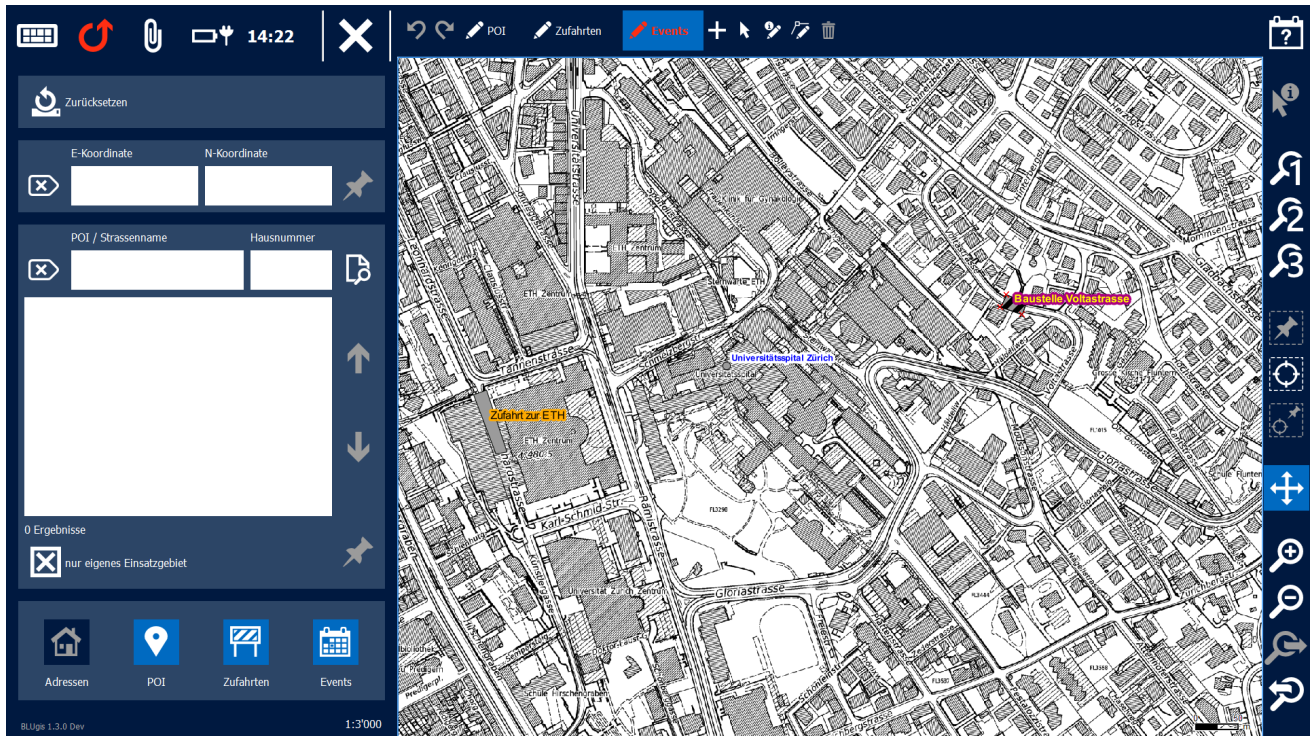


Abbildung 34: BLUgis Editiermodus mit ausgeblendeten Objekten

2. Klicken Sie auf das Symbol mit dem Stift und dem I-Symbol und wählen Sie das Objekt aus, welches Sie einblenden möchten. Es erscheint ein ausgegrautetes Detailfenster (siehe Abbildung 32).
3. Klicken Sie im Fenster auf «Einblenden», um das Objekt einzublenden. Es erscheint auf der Karte und das Detailfenster färbt sich blau (siehe Abbildung 31).
4. Schliessen Sie das Detailfenster und beenden Sie den Editiermodus. Das Objekt ist eingebildet.

20 Schnittstellen

Diese API-Beschreibung bezieht sich ausschliesslich auf die BLUshare-spezifischen Schnittstellen. Die Beschreibungen zu den von GeoServer und GeoNode angebotenen Schnittstellen sind in den entsprechenden Online-Dokumentationen zu entnehmen. [85] [86]

20.1 Daten-Upload

Der Daten-Upload erfolgt in zwei Schritten. Im ersten Schritt werden die Daten der Layer und Attachments-Information hochgeladen. Danach werden die eigentlichen Attachments übertragen. Das Format für den Upload der Daten wird im Abschnitt «Datenformat» genauer beschrieben.

20.1.1 Upload von Layer-Daten

URL	/blushare/upload
Beschreibung	Übertragung der Daten aus Attachments-, Events-, POI- und Zufahrts-Datenbank. Die Daten müssen in einem Zip verpackt sein, welches aus den folgenden Dateien «attachments.json», «entries.geojson», «events.geojson» und «poi.geojson» besteht. Voraussetzung ist zudem ein gültiges Login.
HTTP-Methode	POST
Parameter	<ul style="list-style-type: none"> • username: Benutzername des BLUshare-Benutzers • password: Passwort des BLUshare-Benutzers • dbfile: Zip-Datei mit den Layer-Daten
Rückgabewert	JSON mit den folgenden Attributen: <ul style="list-style-type: none"> • success (Boolean): Gibt an, ob die Verarbeitung der Daten erfolgreich war oder nicht. • missing_attachments (Liste): Enthält eine Liste der Attachments, welche in den hochgeladenen Daten angegeben waren, aber sich noch nicht auf dem Server befinden. • message (String): Enthält eine Nachricht, falls während dem API-Aufruf ein Fehler aufgetreten ist. Andernfalls wird dieses Attribut nicht gesetzt.
Beispielrückgabe 1:	<pre> { "success": true, "missing_attachments": ["b094cecf-774c-43e7-919b-7e68127c3c10", "dce52c33-698e-414b-8feb-b04552b9d3de", "e914ab61-9d6e-42b5-8c11-584973e4a296"] } </pre>

Beispielrückgabe 2:	<pre> { "success": false, "missing_attachments": [] "message": "Authentication failed" } </pre>
---------------------	---

20.1.2 Upload von Attachments

URL	/blushare/upload
Beschreibung	Übertragung der Attachments, welche in einem Zip verpackt sein müssen. Der Server übernimmt nur die Dateien, die er vorher als fehlend (missing_attachments) gemeldet hat. Voraussetzung ist zudem ein gültiges Login.
HTTP-Methode	POST
Parameter	<ul style="list-style-type: none"> • username: Benutzername des BLUshare-Benutzers • password: Passwort des BLUshare-Benutzers • attachments: Zip-Datei mit den Attachments
Rückgabewert	<p>JSON mit den folgenden Attributen:</p> <ul style="list-style-type: none"> • success (Boolean): Gibt an, ob die Verarbeitung der Attachments erfolgreich war oder nicht. • missing_attachments (Liste): Leere Liste. • message (String): Enthält eine Nachricht, falls während dem API-Aufruf ein Fehler aufgetreten ist. Andernfalls wird dieses Attribut nicht gesetzt.
Beispielrückgabe 1:	<pre> { "success": true, "missing_attachments": [] } </pre>
Beispielrückgabe 2:	<pre> { "success": false, "missing_attachments": [], "message": "Authentication failed" } </pre>

20.2 Daten-Download

Der Daten-Download erfolgt wie der Upload auch in zwei Schritten. Zuerst sendet der Server die Daten der Layer und Attachments-Information, danach kann der Client melden, welche Attachments ihm fehlen, worauf der Server diese sendet. Die Parameter beim POST-Aufruf müssen als JSON im Body vom Request übergeben werden. Das Format der gesendeten Daten wird im Abschnitt «Datenformat» genauer beschrieben.

20.2.1 Download von Layer-Daten

URL	/blushare/download
Beschreibung	Download der Daten aus Attachments-, Events-, POI- und Zufahrts-Datenbank. Die Daten werden in einem Zip verpackt welches aus den Dateien «attachments.json», «entries.geojson», «events.geojson» und «poi.geojson» besteht. Voraussetzung ist ein gültiges Login.
HTTP-Methode	POST
Parameter	<ul style="list-style-type: none"> • username: Benutzername des BLUshare-Benutzers • password: Passwort des BLUshare-Benutzers
Rückgabewert	<p>ZIP-Datei mit den den Layer-Daten. Gehört ein Objekt dem aktuell authentifizierten Benutzer, dann ist das blushare_user-Attribut von dem entsprechenden Objekt auf null gesetzt.</p> <p>Tritt während der Verarbeitung ein Fehler auf, wird der String «Error» zurückgegeben.</p>

20.2.2 Download von Attachments

URL	/blushare/download
Beschreibung	Download der Daten aus Attachments-, Events-, POI- und Zufahrts-Datenbank. Die Daten werden in einem Zip verpackt welches aus den Dateien «attachments.json», «entries.geojson», «events.geojson» und «poi.geojson» besteht. Voraussetzung ist ein gültiges Login.
HTTP-Methode	POST
Parameter	<ul style="list-style-type: none"> • username: Benutzername des BLUshare-Benutzers • password: Passwort des BLUshare-Benutzers • missing_attachments: Eine Liste mit den IDs der fehlenden Attachments
Rückgabewert	<p>ZIP-Datei mit den Attachments, welche im Attribut «missing_attachments» aufgeführt waren und auf dem Server vorhanden sind.</p> <p>Tritt während der Verarbeitung ein Fehler auf, wird der String «Error» zurückgegeben.</p>

20.3 Datenformat

Die Daten werden in einer Zip-Datei übertragen, welche vier Dateien beinhalten muss. Im Folgenden wird für diese Dateien ein Beispiel zur geforderten Struktur gezeigt.

20.3.1 attachments.json

```
[
{"bluid": "b5bac8c7-a228-4190-9a51-dee6c39c7614", "attachment_filename": ".pdf",
"b5bac8c7-a228-4190-9a51-dee6c39c7614": "Baustelle_Seestrasse.pdf",
"blushare_timestamp": 1465137628},
{"bluid": "35fd917f-72e2-4272-909c-82ef4a524c6d", "attachment_filename": "35fd917f-72e2-4272-909c-82ef4a524c6d.pdf", "attachment_original_filename": "Situationsplan.pdf",
"blushare_timestamp": 1465137532},
{...}
]
```

20.3.2 entries.geojson

```
{
"type": "FeatureCollection",
"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:EPSG::21781" } },
"features": [
{ "type": "Feature", "properties": { "bluid": "1ff333ab-85e0-408d-99b8-f1d686fa3e0c",
"entry_file": "35fd917f-72e2-4272-909c-82ef4a524c6d", "entry_name": "Zufahrt Wohnheim
BALM Glärnischstrasse 20", "entry_type": "Zufahrt", "entry_description": "Zufahrt über
Stadthofplatz nicht möglich!", "entry_record_by": "wic", "entry_record_date": "2014-09-
20", "blushare_timestamp": 1465137627 }, "geometry": { "type": "Polygon", "coordinates":
[ [ [ 704638.45205377298, 231749.07663396999 ], [ 704691.59146177396, 231703.88721614299
], [ 704697.86776980595, 231682.756979103 ], [ 704707.28223185299, 231663.50963447301 ],
[ 704701.215134089, 231662.04516259901 ], [ 704701.00592382101, 231664.34647554401 ], [
704686.57041534898, 231695.518805434 ], [ 704684.89673320705, 231699.07537998501 ], [
704681.34015865601, 231703.05037507199 ], [ 704635.31389975804, 231744.05558754501 ], [
704638.45205377298, 231749.07663396999 ] ] ] } },
{...}
]
}
```

20.3.3 events.geojson

```
{
"type": "FeatureCollection",
"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:EPSG::21781" } },
"features": [
{ "type": "Feature", "properties": { "bluid": "8de1e4d9-7d0f-47d8-9e7b-032a9df6d0dd",
"event_file": "b5bac8c7-a228-4190-9a51-dee6c39c7614", "event_name": "Sperrung, Wasser-
leitungsbau", "event_type": "achtung", "event_entry": "3", "event_from_date": "2014-11-
20", "event_from_time": "00:00", "event_to_date": "2014-11-26", "event_to_time":
"00:00", "event_description": "", "event_is_repeating": "0", "event_troddb_export": "1",
"event_record_by": "wic", "event_record_date": null, "blushare_timestamp": 14651376772
}, "geometry": { "type": "Polygon", "coordinates": [ [ [ 703309.343749956,
234882.14375013701 ], [ 703307.35937495402, 234877.778125132 ], [ 703317.28124996403,
234873.80937512801 ], [ 703319.26562496601, 234878.571875133 ], [ 703309.343749956,
234882.14375013701 ] ] ] } },
{...}
]
}
```

20.3.4 poi.geojson

```
{
  "type": "FeatureCollection",
  "crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:EPSG::21781" } },
  "features": [
    { "type": "Feature", "properties": { "bluid": "10af487f-6424-45cc-a809-b84d288d6f3e",
      "poi_file": null, "poi_name": "Regio 144", "poi_phone": null, "poi_type": "info",
      "poi_description": null, "poi_record_source": "manuell", "poi_record_by": "hom",
      "poi_record_date": "2016-06-05", "blushare_timestamp": 1465160625 }, "geometry": {
      "type": "Point", "coordinates": [ 706353.89459164103, 235216.095807896 ] } },
    {...}
  ]
}
```

Teil V: Anhang

Persönliche Berichte

Martin Eisenhammer

Das Thema «BLUshare - Geodaten austauschplattform für Blaulichtorganisationen» der Bachelorarbeit hat mich angesprochen, weil dies eine interessante und anspruchsvolle Aufgabe aus der Softwareentwicklung ist. Die Aufgabe besteht aus der Entwicklung einer Austauschplattform für Geodaten (Datenbankentwicklung), sowie eine Entwicklung eines Webclients (Webentwicklung). Weiterhin gab es Anpassungen des vorhandenen BLUgis vorzunehmen (User Interface einer Desktopanwendung, die auf einem Tablet läuft).

Das erste Meeting mit dem betreuenden Dozenten Stefan Keller und dem Industriepartner honeyconsult markus honegger Wetzikon (Markus Honegger) fand ich super. Stefan Keller kannte ich aus den Vorlesungen «Datenbanksysteme 2» und «Informationssysteme». Markus Honegger kannte ich vorher noch nicht.

Ich konnte mich in die vorgegebene Programmiersprache Python schnell einarbeiten. Ein Vorteil war, dass mein Teampartner Josua Stähli bereits mit Python vertraut war. Somit konnte ich ihn jederzeit etwas zu Python fragen. Während der Technologieanalyse habe ich neue verschiedene Technologien kennengelernt, wie beispielsweise GeoNode, ein CMS für Geodaten, welches in diesem Projekt für den Webclient eingesetzt wurde. QGIS kannte ich teilweise aus der Vorlesung «Informationssysteme». In diesem Projekt konnte ich mich mit Josua Stähli vertieft mit Geoinformationssystemen befassen. Die Anwendung von Geoinformationssystemen bei Blaulichtorganisationen finde ich interessant.

Die Technologieanalyse hat etwas viel Zeit im Anspruch genommen, da wir einige Technologien und GeoNode genauer unter die Lupe genommen haben. Insbesondere bei der Installation des Quellcodes gab es viele Abhängigkeiten zu installieren. Die Installation erfolgte mit Hilfe einer Windows-Konsole. Aus Erfahrung weiss ich, dass das Arbeiten mit einem Kommandozeileninterpreter etwas unhandlich ist. Bei einer neuen Version von GeoNode oder bei weiteren Anpassungen waren manchmal Schritte der Installation zu wiederholen. Bei Fehlern traten grösstenteils unklare Fehlermeldungen auf, wobei es manchmal Zeit gekostet hat, die Fehler zu beheben. Bei einer speziellen Fehlermeldung konnte ich herausfinden, dass im Quellcode von GeoNode ein falscher Downloadlink angegeben war. Ich habe diesen Bug der GeoNode-Community gemeldet und nach wenigen Tagen wurde dieser Bug gefixt. Somit konnte ich mit meinen Fähigkeiten etwas zur Community beitragen.

Bei der Implementierung war ich grösstenteils für Anpassungen des vorhandenen BLUgis verantwortlich. Dabei konnte ich mich mit der GUI-Library Qt befassen. Qt kannte ich vorher nur als C++-Library. Ein kleiner Nachteil war, dass ich öfters das Internet konsultieren musste, um eine bestimmte Methode oder Klasse zu finden, da meine Entwicklungsumgebung teilweise keine API-Auszüge für Python angezeigt hat.

Für mich war diese Bachelorarbeit lehrreich, da ich eine neue Programmiersprache und neue Technologien kennengelernt habe. Mir hat die Betreuung durch Prof. Stefan Keller und Markus Honegger und die Zusammenarbeit mit Josua Stähli sehr gut gefallen.

Josua Stähli

Mit dieser Bachelorarbeit standen wir vor der Aufgabe, eine Synchronisations-Plattform für Geodaten von Grund auf zu erstellen. Diese Synchronisationsfunktion mussten wir zudem in BLUgis integrieren, einem Geoinformationssystem für Rettungsdienste, welches bereits im Einsatz ist. Schon vor der Bachelorarbeit habe ich an BLUgis mitgearbeitet, wodurch ich beim clientseitigen Teil der Arbeit bereits Vorwissen mitbrachte.

Als sehr herausfordernd empfand ich die Umsetzung eines Synchronisations-Algorithmus, der mit einer grossen Zahl an Benutzern umgehen können muss, bei dem die Daten der Benutzer untereinander geschützt bleiben und möglichst nur veränderte Daten übertragen werden sollen. Zudem hat sich die Suche und Evaluation geeigneter Technologien, auf denen wir unsere Arbeit aufbauen wollen, als deutlich zeitintensiver herausgestellt als gedacht. Eine weitere langwierige Arbeit war die Serverkonfiguration, welche das Einrichten der Entwicklungsumgebung bis zum Deployment von BLUshare beinhaltete.

Das Projekt war für mich besonders darum sehr lehrreich, weil wir uns mit vielen verschiedenen Technologien befassen konnten. Das GeoNode-CMS alleine baut auf vielen Software-Bibliotheken und Tools auf, wobei wir im Verlaufe der Arbeit mit einigen davon in Berührung kamen. Ausserdem werden auf der Client- und Serverseite unterschiedliche Datenbanksysteme verwendet.

Aufgrund der grossen Zahl an gesammelten Erfahrungen ist diese Bachelorarbeit eine grosse Bereicherung für mich. Ausserdem hat die Zusammenarbeit mit meinem Projektpartner – Martin Eisenhammer – sehr gut geklappt, wodurch ein effizientes Arbeiten möglich war. Die Beendigung dieser Bachelorarbeit bedeutet für mich gleichzeitig auch den Abschluss meines Studiums an der Hochschule für Technik Rapperswil. Was ich über diese Bachelorarbeit sagen kann, gilt ebenso für das gesamte Studium an der HSR: es war eine herausfordernde und intensive, gleichzeitig aber auch sehr spannende und lehrreiche Zeit.

Ich möchte mich vielmals bei Prof. Stefan Keller für die kompetente Betreuung dieser Bachelorarbeit bedanken, welche ich sehr geschätzt habe und auch massgebend zur erfolgreichen Umsetzung des Projektes beigetragen hat. Ein grosser Dank geht auch an den Industriepartner Markus Honegger für die seinerseits investierte Zeit, die rasche Beantwortung aller auftretenden Fragen und für das laufende Feedback zu unserer Arbeit.

Literaturverzeichnis

- [1] „PEP 8 - Style Guide for Python Code,“ [Online]. Available: <https://www.python.org/dev/peps/pep-0008/>. [Zugriff am 15. Juni 2016].
- [2] „Stackoverflow: Advantages and disadvantages of GUID/UUID database keys,“ [Online]. Available: <http://stackoverflow.com/questions/45399/advantages-and-disadvantages-of-guid-uuid-database-keys>. [Zugriff am 13. Juni 2016].
- [3] „Stackoverflow: What’s your opinion on using UUIDs as database row identifiers, particularly in web apps?,“ [Online]. Available: <http://stackoverflow.com/questions/5949/whats-your-opinion-on-using-uuids-as-database-row-identifiers-particularly-in>. [Zugriff am 13. Juni 2016].
- [4] „Wikipedia: Spatial database,“ [Online]. Available: https://en.wikipedia.org/wiki/Spatial_database. [Zugriff am 15. Juni 2016].
- [5] „PostGIS: Spatial and Geographic objects for PostgreSQL,“ [Online]. Available: <http://postgis.net/>. [Zugriff am 15. Juni 2016].
- [6] „Oracle: Oracle Spatial and Graph,“ [Online]. Available: <http://www.oracle.com/technetwork/database-options/spatialandgraph/overview/spatialandgraph-1707409.html>. [Zugriff am 15. Juni 2016].
- [7] „Spatialite,“ [Online]. Available: <https://www.gaia-gis.it/fossil/libspatialite/index>. [Zugriff am 15. Juni 2016].
- [8] „Esri: ArcGIS for Server,“ [Online]. Available: <http://www.esri.com/software/arcgis/arcgisserver/>. [Zugriff am 15. Juni 2016].
- [9] „GeoServer,“ [Online]. Available: <http://geoserver.org/>. [Zugriff am 15. Juni 2016].
- [10] „QGIS: Ein freies Open-Source-Geographisches-Informationssystem,“ [Online]. Available: <http://qgis.org/>. [Zugriff am 15. Juni 2016].
- [11] „MapServer,“ [Online]. Available: <http://mapserver.org/>. [Zugriff am 15. Juni 2016].
- [12] „Catara: GeoSpatial CMS,“ [Online]. Available: <http://cartaro.org/>. [Zugriff am 15. Juni 2016].
- [13] „geOps - SpatialWeb: Catara features,“ [Online]. Available: <http://cartaro.org/features>. [Zugriff am 15. Juni 2016].
- [14] „GeoNode: Open Source Geospatial Content Management System,“ [Online]. Available: <http://geonode.org/>. [Zugriff am 15. Juni 2016].
- [15] „GeoNode - for users,“ [Online]. Available: http://geonode.org/user_features/. [Zugriff am 15. Juni 2016].

-
- [16] „GeoNode - for developers,“ [Online]. Available: http://geonode.org/dev_features/. [Zugriff am 15. Juni 2016].
- [17] „GeoNode - for Administrators,“ [Online]. Available: http://geonode.org/admin_features/. [Zugriff am 15. Juni 2016].
- [18] „Casagrande Luca, Corti Paolo, Dalmaso Simone: GeoNode 2.0, GFOSS Day 2013, Bologna, Italy, October 10th 2013,“ [Online]. Available: <http://de.slideshare.net/capooti/geonode-20>. [Zugriff am 15. Juni 2016].
- [19] „GeoSHAPE,“ [Online]. Available: <http://geoshape.org/>. [Zugriff am 15. Juni 2016].
- [20] „Mapbender 3,“ [Online]. Available: <https://mapbender3.org/>. [Zugriff am 15. Juni 2016].
- [21] „Mapbender3 Quickstart,“ [Online]. Available: <http://doc.mapbender3.org/en/book/quickstart.html>. [Zugriff am 15. Juni 2016].
- [22] „Boundless: Boundless GIS Platform,“ [Online]. Available: <http://boundlessgeo.com/products/product-overview/>. [Zugriff am 15. Juni 2016].
- [23] „Boundless: OpenGeo Suite,“ [Online]. Available: <http://boundlessgeo.com/products/opengeo-suite/>. [Zugriff am 15. Juni 2016].
- [24] „Deegree,“ [Online]. Available: <http://www.deegree.org/>. [Zugriff am 15. Juni 2016].
- [25] „Geomajas,“ [Online]. Available: <http://www.geomajas.org/>. [Zugriff am 15. Juni 2016].
- [26] „MapFish,“ [Online]. Available: <http://www.mapfish.org/>. [Zugriff am 15. Juni 2016].
- [27] „GeoDjango,“ [Online]. Available: <http://geodjango.org/>. [Zugriff am 15. Juni 2016].
- [28] „MapIgniter2,“ [Online]. Available: <https://github.com/taviraquai/mapigniter2>. [Zugriff am 15. Juni 2016].
- [29] „MapServerPro,“ [Online]. Available: <https://www.mapserverpro.com/>. [Zugriff am 15. Juni 2016].
- [30] „Geobuf (GitHub),“ [Online]. Available: <https://github.com/mapbox/geobuf>. [Zugriff am 15. Juni 2016].
- [31] „Keller, Stefan: GeoCSV,“ [Online]. Available: <http://giswiki.hsr.ch/GeoCSV>. [Zugriff am 15. Juni 2016].
- [32] „IETF Geographic JSON Workgroup: GeoJSON,“ [Online]. Available: <http://geojson.org/>. [Zugriff am 15. Juni 2016].
- [33] „OGC: GeoPackage,“ [Online]. Available: <http://www.geopackage.org/>. [Zugriff am 15. Juni 2016].
- [34] „ESRI - Shapefile technical description - An ESRI White Paper - July 1998,“ [Online]. Available: <https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf>. [Zugriff am 15. Juni 2016].
- [35] „Wikipedia: Shapefile,“ [Online]. Available: <https://de.wikipedia.org/wiki/Shapefile>. [Zugriff am 15. Juni 2016].

- [36] „SpatialLite,“ [Online]. Available: <https://www.gaia-gis.it/fossil/libspatialite/index>. [Zugriff am 15. Juni 2016].
- [37] „Keller, Stefan: TheShapeFileChallenge,“ [Online]. Available: <http://www.interlis.ch/general/docs/20160324-Spirtgartentreffen.zip>. [Zugriff am 15. Juni 2016].
- [38] „GeoNode: Spatial Data Discovery,“ [Online]. Available: http://docs.geonode.org/en/master/tutorials/overview_and_ref/user_features/spatial_data_discovery.html. [Zugriff am 15. Juni 2016].
- [39] „GeoNode: GeoNode 2.4 released,“ [Online]. Available: <http://geonode.org/blog/2015/11/19/geonode-2.4-released/>. [Zugriff am 15. Juni 2016].
- [40] „GeoNode: Import and Manage,“ [Online]. Available: http://docs.geonode.org/en/master/tutorials/overview_and_ref/user_features/import_and_manage.html. [Zugriff am 15. Juni 2016].
- [41] „OSGeoLive: Web Feature Service (WFS),“ [Online]. Available: http://live.osgeo.org/de/standards/wfs_overview.html. [Zugriff am 15. Juni 2016].
- [42] „GeoNode: Open Geospatial Consortium (OGC) Standards,“ [Online]. Available: http://docs.geonode.org/en/latest/tutorials/devel/devel_intro/standards.html#open-geospatial-consortium-ogc-standards. [Zugriff am 15. Juni 2016].
- [43] „OSGeoLive: Web Map Service (WMS),“ [Online]. Available: http://live.osgeo.org/de/standards/wms_overview.html. [Zugriff am 15. Juni 2016].
- [44] „GeoNode: Interactive Mapping,“ [Online]. Available: http://docs.geonode.org/en/master/tutorials/overview_and_ref/user_features/interactive_mapping.html. [Zugriff am 15. Juni 2016].
- [45] „GeoNode: for developers,“ [Online]. Available: http://geonode.org/dev_features/. [Zugriff am 15. Juni 2016].
- [46] „GeoNode: Custom Installation Guide,“ [Online]. Available: http://docs.geonode.org/en/master/tutorials/admin/install/custom_install.html?highlight=system%20requirements. [Zugriff am 15. Juni 2016].
- [47] „GeoNode: Supported Browsers,“ [Online]. Available: <http://docs.geonode.org/en/latest/reference/browsers.html#browsers>. [Zugriff am 15 Juni 2016].
- [48] „GeoNode: Permissions and GeoNode objects,“ [Online]. Available: <http://docs.geonode.org/en/latest/reference/security.html#permissions-and-geonode-objects>. [Zugriff am 15. Juni 2016].
- [49] „GeoNode: Components,“ [Online]. Available: http://docs.geonode.org/en/master/tutorials/overview_and_ref/reference_doc/components.html. [Zugriff am 15. Juni 2016].

- [50] „GeoNode (GitHub),“ [Online]. Available: <https://github.com/GeoNode/geonode>. [Zugriff am 15. Juni 2016].
- [51] „GNU General Public Licence Version 3, 29 June 2007,“ [Online]. Available: <http://www.gnu.org/licenses/gpl.html>. [Zugriff am 15. Juni 2016].
- [52] „Delphi Praxis: GNU v3 License für kommerzielle Projekte nutzbar?,“ [Online]. Available: <http://www.delhipraxis.net/158230-gnu-v3-license-fuer-kommerzielle-projekte-nutzbar.html>. [Zugriff am 15. Juni 2016].
- [53] „Programmers Stackexchange: Can I use GPL software in a commercial application?,“ [Online]. Available: <http://programmers.stackexchange.com/questions/47032/can-i-use-gpl-software-in-a-commercial-application>. [Zugriff am 15. Juni 2016].
- [54] „Kesper, Stephan: Verwendung GPL-lizenzierter Komponenten in kommerziellen Projekten,“ [Online]. Available: <https://blog.codecentric.de/2012/05/verwendung-gpl-lizenzierter-komponenten-in-kommerziellen-projekten/>. [Zugriff am 15. Juni 2016].
- [55] „Steinle, Thomas: GPLv3 Neue Nutzungsformen von Software wie z.B. Application Service Providing (ASP) und Software as a Service (SaaS),“ [Online]. Available: <http://www.it-rechtsanwalt.com/open-source-software/gplv3-neue-nutzungsformen-von-software-wie-z-b-application-service-providing-asp-und-software-as-a-service-saas-2711.php>. [Zugriff am 15. Juni 2016].
- [56] „Quora: GPL: Is it legal to host GPL3 licenced product after modification in SaaS mode and charge users?,“ [Online]. Available: <https://www.quora.com/GPL-Is-it-legal-to-host-GPL3-licenced-product-after-modification-in-SaaS-mode-and-charge-users>. [Zugriff am 15. Juni 2016].
- [57] „OSGeo.org: Possible to use different map server,“ [Online]. Available: <http://osgeo-org.1560.x6.nabble.com/Possible-to-use-different-map-server-td5257976.html>. [Zugriff am 15. Juni 2016].
- [58] „GitHub: kartoza/geonode,“ [Online]. Available: <https://github.com/kartoza/geonode>. [Zugriff am 15. Juni 2016].
- [59] „Kartoza,“ [Online]. Available: <http://geonode.kartoza.com/>. [Zugriff am 15. Juni 2016].
- [60] „GeoNode: Help The World Bank build a GeoNode to QGIS bridge,“ [Online]. Available: <http://geonode.org/blog/2012/09/17/help-the-world-bank-build-a-geonode-to-qgis-bridge/>. [Zugriff am 15. Juni 2016].
- [61] „Django documentation: django-admin and manage.py,“ [Online]. Available: <https://docs.djangoproject.com/en/1.8/ref/django-admin/>. [Zugriff am 15. Juni 2016].
- [62] „GeoNode: GeoServer data configuration,“ [Online]. Available: http://docs.geonode.org/en/master/tutorials/admin/loading_data_into_geonode/geoserver.html. [Zugriff am 15. Juni 2016].
- [63] „GeoNode: About GeoNode,“ [Online]. Available: <http://docs.geonode.org/en/master/about.html#geospatial-data-storage>. [Zugriff am 15. Juni 2016].

- [64] „Geonode: How to contribute to GeoNode’s translation,“ [Online]. Available: http://docs.geonode.org/en/master/organizational/contribute/contribute_to_translation.html#contribute-trans. [Zugriff am 15. Juni 2016].
- [65] „GeoNode: A Tour Of GeoNode,“ [Online]. Available: http://docs.geonode.org/en/master/tutorials/overview_and_ref/introduction/geonode_tour.html. [Zugriff am 15. Juni 2016].
- [66] „GeoNode: Frequently Asked Questions,“ [Online]. Available: <http://geonode.org/faqs/>. [Zugriff am 15. Juni 2016].
- [67] „GoogleGroups: GeoNode development,“ [Online]. Available: <https://groups.google.com/a/opengeo.org/forum/#!forum/geonode-dev>. [Zugriff am 15. Juni 2016].
- [68] „GeoNode devel,“ [Online]. Available: <http://osgeo-org.1560.x6.nabble.com/GeoNode-devel-f5185388.html>. [Zugriff am 15. Juni 2016].
- [69] „QGIS: Open-Source-Geographisches-Informationssystem,“ [Online]. Available: <http://qgis.org/>. [Zugriff am 15. Juni 2016].
- [70] „QAction Klasse in Qt,“ [Online]. Available: <http://doc.qt.io/qt-4.8/qaction.html>. [Zugriff am 15. Juni 2016].
- [71] „Fail2ban,“ [Online]. Available: <http://www.fail2ban.org/>. [Zugriff am 15. Juni 2016].
- [72] „Psycopg: Basic module usage,“ [Online]. Available: <http://initd.org/psycopg/docs/usage.html>. [Zugriff am 15. Juni 2016].
- [73] „GeoNode: Security and Permissions,“ [Online]. Available: <http://docs.geonode.org/en/master/reference/security.html>. [Zugriff am 15. Juni 2016].
- [74] „Stevens Didier: Escape from PDF,“ [Online]. Available: <https://blog.didierstevens.com/2010/03/29/escape-from-pdf/>. [Zugriff am 15. Juni 2016].
- [75] „GeoServer: SQL Views,“ [Online]. Available: <http://docs.geoserver.org/stable/en/user/data/database/sqlview.html>. [Zugriff am 15. Juni 2016].
- [76] „Manifesto for agile software development,“ [Online]. Available: <http://www.agilemanifesto.org/>. [Zugriff am 10. Juni 2016].
- [77] „ScrumMaster.de Agiles Projektmanagement,“ [Online]. Available: <http://scrum-master.de/Scrum-Rollen/>. [Zugriff am 10. Juni 2016].
- [78] „Kai Simons, Agile Coaching & Management: Story Points verständlich erklärt,“ [Online]. Available: <http://www.ksimons.de/2011/06/story-points-verstandlich-erklart/>. [Zugriff am 10 Juni 2016].
- [79] „PEP-8 Styleguide for Python,“ [Online]. Available: <https://www.python.org/dev/peps/pep-0008/>. [Zugriff am 15. Juni 2016].

-
- [80] „GeoNode: Custom Installation Guide,“ [Online]. Available:
http://docs.geonode.org/en/master/tutorials/admin/install/custom_install.html. [Zugriff am 15. Juni 2016].
- [81] „GeoNode (v2.4) installation on Ubuntu 14.04,“ [Online]. Available:
http://docs.geonode.org/en/master/tutorials/install_and_admin/geonode_install/index.html. [Zugriff am 15. Juni 2016].
- [82] „How to Install PostgreSQL 9.5 on Ubuntu 16.04 and 14.04 LTS,“ [Online]. Available:
<http://tecadmin.net/install-postgresql-server-on-ubuntu/>. [Zugriff am 15. Juni 2016].
- [83] „What's new in PostgreSQL 9.5,“ [Online]. Available:
https://wiki.postgresql.org/wiki/What's_new_in_PostgreSQL_9.5. [Zugriff am 15. Juni 2016].
- [84] „Postgis 2.1 error after update,“ [Online]. Available:
<http://gis.stackexchange.com/questions/97871/postgis-2-1-error-after-update>. [Zugriff am 15. Juni 2016].
- [85] „REST configuration API reference,“ [Online]. Available:
<http://docs.geoserver.org/stable/en/user/rest/api/>. [Zugriff am 15. Juni 2016].
- [86] „GeoNode APIs,“ [Online]. Available:
http://docs.geonode.org/en/master/tutorials/devel/geonode_apis/index.html. [Zugriff am 15. Juni 2016].