



FH Aachen

**Fachbereich
Elektrotechnik und Informationstechnik
Studiengang Informatik**

Bachelorarbeit

Konzeption und prototypische Entwicklung einer webbasierten Applikation
zur Wertschöpfung und Bereitstellung von Geodaten
(Data-Konnector für Geodaten)

Tuan Anh Cong Nguyen
Matr.-Nr.: 3517392

Referent: Prof. Dr. rer. nat. Heinrich Faßbender

In Zusammenarbeit mit Ausbildungsbetrieb:
ahu GmbH Wasser Boden Geomatik

Externer Betreuer: Dr. David Loibl

Selbstständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Aachen, den 12. Mai 2025

.....

Inhaltsverzeichnis

Abstract	5
1 Einleitung	6
1.1 Motivation der Arbeit	6
1.2 Ziel der Arbeit	6
2 Some examples to get started	7
2.1 How to create Sections and Subsections	7
2.2 How to include Figures	7
2.2.1 How to add Tables	7
2.3 How to add Comments and Track Changes	7
2.4 How to add Lists	7
2.5 How to write Mathematics	8
2.6 How to change the margins and paper size	8
2.7 How to change the document language and spell check settings	8
2.8 How to add Citations and a References List	8
2.9 Good luck!	9
3 Übersicht von Technologien	10
3.1 Vaadin	10
3.2 Spring Framework	11
3.3 Spring Boot	11
3.4 Maven	11

Abkürzungsverzeichnis

g	Gravitation in Nähe der Erdoberfläche
Nu	Nußelt-Zahl
ν_{Luft}	Kinematische Viskosität von Luft
Pr	Prandtl-Zahl
\dot{Q}	Wärmestrom
Ra	Rayleigh-Zahl
ρ_{Luft}	Dichte von Luft
T	Temperatur
T_{∞}	Umgebungstemperatur

Abstract

In einer zunehmend digitalisierten und automatisierten Welt wächst täglich die Menge an neu generierten Geodaten – insbesondere Mess- und räumlichen Daten.

Die Suche nach und Einbindung von OGC-Geodatendiensten wie WMS (Web Map Service) und WFS (Web Feature Service) in Geoinformationssysteme wie QGIS kann tatsächlich schwierig sein. Die Suche erfordert den Anwender oft das Durchforsten verschiedener Quellen und die manuelle Überprüfung von Dienstbeschreibung. Eine automatisiert und effiziente Lösung kann teilweise helfen, diesen Prozess zu verbessern.

Die Suche nach Messdaten wie Grundwasserständen, Pegelständen und Klimadaten (Niederschlag, Temperatur) und deren Umformatierung in ein einheitliches Format gestalten sich als schwierig und kompliziert, da es eine Vielzahl von Datenanbietern mit unterschiedliche Datenstrukturen, Formen, etc. gibt.

Die Bereitstellung dieser Rohdaten erfolgt meist im CSV-Format oder als individuell strukturierte ASCII-Dateien. Im Kontext von der Datenkonsum, gibt es bei der Ahu den sogenannten ahuManager-Client, der diese Daten konsumiert. Bevor diese konsumieren können, müssen die Rohdaten in ein kompatibles Format umformatiert werden, was bisher teilweise von Hand erledigt werden (in Excel o.ä.). Diese manuelle Vorgehen kann perspektivisch zum Teil oder ganz automatisiert werden, was die Produktivität steigert und die Fehleranfälligkeit vermeidet.

Daher befasst sich diese Arbeit mit dem Thema, wie eine webbasierte Applikation zur Wertschöpfung und Bereitstellung von rohen Geodaten im Kontext von der Abteilung Geomatik von Ahu GmbH entwickelt werden kann.

1 Einleitung

1.1 Motivation der Arbeit

Diese Bachelorarbeit wird in Zusammenarbeit mit dem Bereich Geomatik der Ahu GmbH verfasst. Der Bereich beschäftigt sich mit Konzeption und Softwareentwicklung der Monitoringsysteme und Web-Anwendungen, die es Dienstleitern, Betreibern und Aufsichtsbehörden ermöglichen, ein kosteneffizientes Management für Geodaten (Grundwasser, Oberflächenwasser, Boden, etc.) umzusetzen.

Momentan besteht es Bedarf für ein zentrales System zur automatisierten Suche und Verwaltung der Geodaten aus verschiedenen Datenanbietern und diese Daten in einer homogen Struktur zusammenzubringen und einen vorhandenen Client (z.B. ahumanager) bereitzustellen. Unter anderem wird auch die Einbindung von OGC-Geodatendiensten für besseren und effizienten Prozess zur Durchsuche und Verwaltung gesorgt.

1.2 Ziel der Arbeit

Die Hauptaufgabe dieser Arbeit besteht darin, eine Konzeption für die prototypische Entwicklung einer webbasierten Applikation zu erstellen und umzusetzen. Die Ziele der Applikation enthalten eine übergreifende Suche von Geodaten und deren Umformatierung in ein passendes vorgegebenes Zielformat. Das weitere Ziel ist die Einbindung von mindestens einer OGC-Geodatendiensten.

Durch die Umsetzungen dieser Ziele soll perspektivisch eine Software entstehen, die die offenen Problematik abdeckt.

2 Some examples to get started

2.1 How to create Sections and Subsections

2.2 How to include Figures

Note that your figure will automatically be placed in the most appropriate place for it, given the surrounding text and taking into account other figures or tables that may be close by. You can find out more about adding images to your documents in this help article on [including images on Overleaf](#).

2.2.1 How to add Tables

Use the `table` and `tabular` environments for basic tables — see Table 2.1, for example. For more information, please see this help article on [tables](#).

2.3 How to add Comments and Track Changes

Track changes are available on all our [premium plans](#), and can be toggled on or off using the option at the top of the Review pane. Track changes allow you to keep track of every change made to the document, along with the person making the change.

2.4 How to add Lists

You can make lists with automatic numbering ...

1. Like this,
2. and like this.

...or bullet points ...

Item	Quantity
Widgets	42
Gadgets	13

Tabelle 2.1: An example table.

2.5 How to write Mathematics

L^AT_EX is great at typesetting mathematics. Let X_1, X_2, \dots, X_n be a sequence of independent and identically distributed random variables with $E[X_i] = \mu$ and $\text{Var}[X_i] = \sigma^2 < \infty$, and let

$$S_n = \frac{X_1 + X_2 + \dots + X_n}{n} = \frac{1}{n} \sum_i^n X_i$$

denote their mean. Then as n approaches infinity, the random variables $\sqrt{n}(S_n - \mu)$ converge in distribution to a normal $\mathcal{N}(0, \sigma^2)$.

2.6 How to change the margins and paper size

If however you're using a more general template, such as this one, and would like to alter the margins, a common way to do so is via the geometry package. You can find the geometry package loaded in the preamble at the top of this example file, and if you'd like to learn more about how to adjust the settings, please visit this help article on [page size and margins](#).

2.7 How to change the document language and spell check settings

[international language support](#).

2.8 How to add Citations and a References List

You can simply upload a `.bib` file containing your BibTeX entries, created with a tool such as JabRef. You can then cite entries from it, like this: **greenwade93**. Just remember to specify a bibliography style, as well as the filename of the `.bib`. You can find a [video tutorial here](#) to learn more about BibTeX. If you have an [upgraded account](#), you can also import your Mendeley or Zotero library directly as a `.bib` file, via the upload menu in the file-tree.

2.9 Good luck!

We hope you find Overleaf useful, and do take a look at our [help library](#) for more tutorials and user guides! Please also let us know if you have any feedback using the Contact Us link at the bottom of the Overleaf menu — or use the contact form at <https://www.overleaf.com/contact>.

3 Übersicht von Technologien

3.1 Vaadin

Vaadin ist ein serverseitiges Web-Framework, das dem Entwickler erlaubt, moderne Webanwendung in Java zu entwickeln, ohne dass explizit HTML, CSS oder JavaScript geschrieben werden muss. Vaadin verfügt über eine große Komponentenbibliothek, die eine Vielzahl an vorgefertigten UI-Elementen wie Buttons, Tabellen, Formulare, Dialoge und Layouts bietet.

Vaadin verfolgt einen serverseitigen Rendering-Ansatz. Während eine AJAX-basierte Vaadin Client-Side Engine dafür sorgt, dass die Benutzeroberfläche im Browser durch ein Widget-Set gerendert wird. Die Benutzeroberfläche kann aus eingebauten Komponenten, Add-Ons und benutzerdefinierten Komponenten bestehen. Diese Client-Side Engine kommuniziert über HTTP oder WebSockets mit dem Server. Die gesamte UI-Logik wird dann auf dem Server ausgeführt. Die Vaadin Servlet empfängt Client-Anfragen und aktualisiert die Benutzeroberfläche. Die UI-Komponenten werden serverseitig erstellt und verwaltet. Änderungen in der UI werden durch die Vaadin Client-Side Engine an den Browser zurückgespielt.

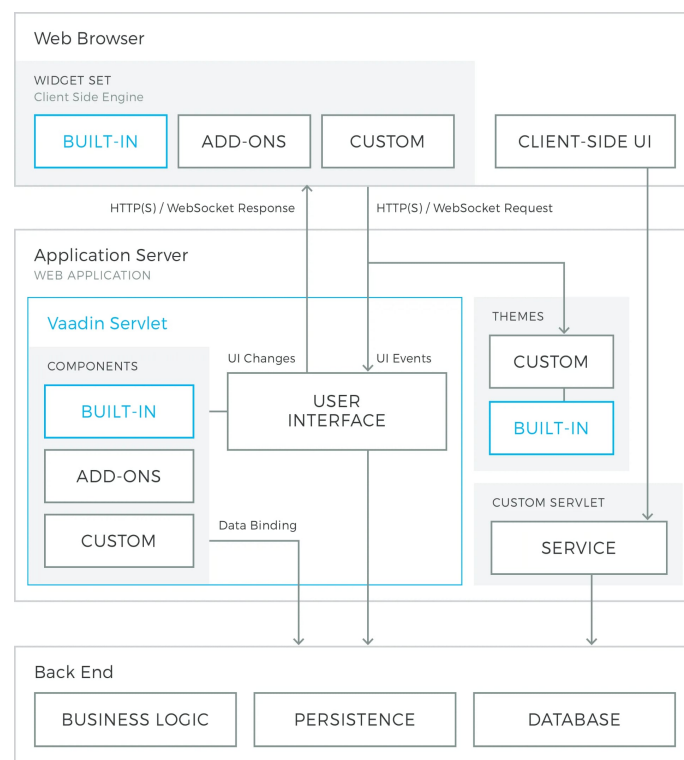


Abbildung 3.1: Übersicht Vaadin Architektur **architecture21**

3.2 Spring Framework

Spring Framework ist ein Java-Framework, das um 2003 als Reaktion auf die damals noch zu komplizierte J2EE-Plattform entwickelt wurde. Spring ermöglicht eine einfachere und unkompliziertere Entwicklung von Enterprise-Applikation in Java. Eines der wichtigsten Konzepte von Spring war die Inversion of Control (IoC). IoC ist ein Prinzip, bei dem der Kontrollfluss an eine externe Quelle(z.B. ein Framework) übergeben wird. Das Framework ist dann gemäß einer Spezifikation für die Erstellung und Löschung der Objekte und den Aufruf von Methoden verantwortlich.

Spring führt das Konzept des Bean-Containers ein. Beans sind Java-Objekte, die von Spring instanziiert und verwaltet sind. Wenn die Haupt-Bean von der Hilfs-Bean abhängig ist, stellt Spring sicher, dass die Hilfs-Bean vor der Haupt-Bean initialisiert wird. Spring injizierte außerdem die Instanz der Hilfs-Bean in die Haupt-Bean, sodass die Haupt-Bean nicht mehr nach ihren eigenen Abhängigkeiten suchen muss. Dieses Muster wird als Dependency Injection bezeichnet.

Damals musste der Entwickler die Bean-Konfiguration in XML schreiben, die Spring anweist, wie die Beans zu konstruieren sind. Inzwischen wird diese mithilfe einer Kombination aus Java-Annotation, Java-Code und Konventionen erstellt.

3.3 Spring Boot

Mit dem Wachstum der Spring-Plattform nahm auch die Komplexität der Entwicklung von Spring-Anwendungen zu. Gleichzeitig verbreitet sich der Einsatz von Microservices und containerisierten Umgebungen in der Softwareentwicklung. Entwickler wünschten sich einfach Methode, um schlanke Webanwendungen zu erstellen, die unabhängig als eigenständige Dienst ausgeführt werden können, anstatt auf einem dedizierten Anwendungsserver zu laufen.

Spring Boot wurde als Reaktion auf diese Anforderungen entwickelt. Es erleichtert die Erstellung von Spring-Anwendungen durch sinnvolle Standard-Einstellungen, Starter-Abhängigkeiten und produktionsreife Funktionen für Konfiguration und Überwachung. Durch die Einführung eingebetteter Servlet-Container wurde es möglich, Anwendungen als eigenständige, ausführbare JAR-Dateien zu verpacken.

3.4 Maven

Maven ist ein Build-Management- und Projektverwaltungswerkzeug, das häufig in der Java-Entwicklung eingesetzt wird. Es automatisiert Aufgaben wie das Herunterladen von Abhängigkeiten, das Kompilieren von Quellcode und das Ausführen von Builds sowie Tests. Maven verwendet eine zentrale Konfigurationsdatei, die `pom.xml`, um alle Aspekte des Projekts wie Libraries,

Plugins und Build-Prozesse zu steuern.

Die Funktionsweise von Maven basiert auf einem deklarativen Ansatz. Das bedeutet, dass der Entwickler alle relevanten Informationen über das Projekt in einer zentralen Konfigurationsdatei, der sogenannten `pom.xml` (Project Object Model) angibt, wie z.B. Abhängigkeiten, Build-Prozesse und Plugins : **deinhard24**.

1. Die `pom.xml`-Datei (Project Object Model)

Die `pom.xml` ist das Herzstück eines Maven-Projekts. Sie enthält Informationen wie:

- Projektinformationen: Name des Projekts, Version, Beschreibung.
- Abhängigkeiten: Welche Bibliotheken und Frameworks das Projekt benötigt.
- Plugins: Zusätzliche Werkzeuge, die den Build-Prozess erweitern (z.B. Compiler-Plugins, Test-Frameworks).
- Repositories: Woher Maven externe Abhängigkeiten herunterladen soll, typischerweise das Maven Central Repository.
- Build-Spezifikationen: Kompilierungsanweisungen, Testkonfigurationen und Deployment-Optionen.

2. Build-Lifecycle

Maven besitzt einen vordefinierten Build-Lifecycle, der aus mehreren Phasen besteht. Zu den wichtigsten Phasen gehören:

- `validate`: Überprüft, ob alle erforderlichen Informationen im Projekt vorhanden sind.
- `compile`: Kompiliert den Quellcode.
- Führt automatisierte Tests aus.
- `package`: Verpackt den kompilierten Code in ein Distributionsformat, typischerweise eine JAR- oder WAR-Datei.
- `install`: Installiert das Paket in das lokale Maven-Repository, damit es in anderen Projekten verwendet werden kann.

3.5 Vaadin Initializer als Einstieg

Gegenüber dem üblichen [Spring Initializer](#), der ein vorkonfiguriertes Spring-Projekt bereitstellt, bietet der [Vaadin Initializer](#) den Vorteil, dass er speziell auf Vaadin-Projekte zugeschnitten ist und eine vereinfachte und schnellere Projektgenerierung ermöglicht, insbesondere wenn man Spring Boot als Backend verwendet.

Mit dem Vaadin Initializer kann man direkt ein Vaadin Flow-basiertes Projekt mit einem Spring Boot-Backend erstellen, während man mit dem Spring Initializer ein allgemeines Spring Boot-Projekt generiert und dann manuell die Vaadin-Abhängigkeiten hinzufügen muss.

3.6 Liquibase

Dank Versionsverwaltung wie Git kann man in den meisten Projekten den Weg einer Code-Änderung von der Entwicklung über Test bis hin zum produktiven Deployment richtig verfolgen und nachvollziehen. Was für den Code gilt, sollte auch für die Datenbank Anpassungen gelten. Liquibase ist eine Bibliothek um Änderungen an einem Datenbankschema verfolgen, verwalten und anwenden zu können. Mit folgendem Eintrag in `pom.xml` kann man in ein Spring-Projekt integrieren.

```
<dependency>
  <groupId>org.liquibase</groupId>
  <artifactId>liquibase-core</artifactId>
</dependency>
```

Abbildung 3.2: Liquibase-Dependency

Für die Speicherung der Daten wird die PostgreSQL-Datenbank entschieden. PostgreSQL selbst eine leistungsstarke, offene Datenbank, die kompatibel zu Liquibase passt. Die Aktivierung der Datenbank wird in der Datei `application.properties` innerhalb der Spring-Applikation wie folgt konfiguriert:

```
spring.datasource.url=jdbc:postgresql://192.168.252.140:5432/opendataconn_ng
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.username=opendataconn_ng_usr
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
```

Abbildung 3.3: Datenbank-Konfiguration

Für die Verwendung von Liquibase sind folgende Dateien notwendig:

1. Liquibase Konfiguration - `liquibase.properties`

In dieser Datei werden allgemeine Einstellungen wie der Datenbankzugang und die zu verwendenden Changelog-Konfiguration festgelegt:

2. Changelog Konfiguration - changelog-root.xml

Ein so genannter Root-ChangeLog wird erstellt, welcher in XML geschrieben ist und eine output-changelog.xml inkludiert, welche im Ordner „versions“ zu finden ist. Diese Datei folgt einer Empfehlung der Liquibase-Dokumentation.

2. Changelog Konfiguration - output-changelog.xml

Diese Datei enthält sämtliche Datenbanksänderungen und die Änderungen werden in Form von ChangeSet beschrieben.

2. Changelog Konfiguration - liquibase-new-diff-changelog.xml

Diese Datei enthält nur bestimmte Änderungen von bestimmten Tabellen wenn man das Schema der Datenbank modifiziert. Wie in out-changelog.xml werden Änderungen auch in Form von ChangeSet beschrieben.

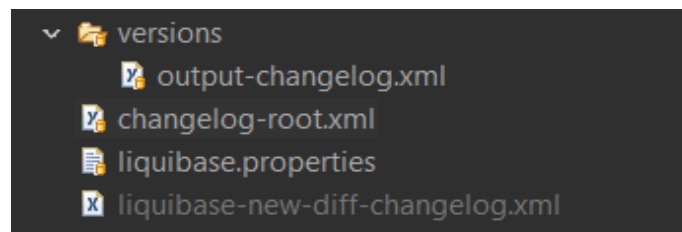


Abbildung 3.4: Liquibase-Konfiguration