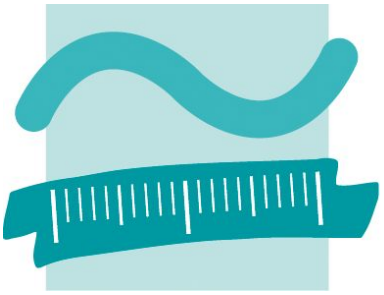

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN

University of Applied Sciences

Fachbereich VI - Informatik und Medien

Datenanalyse und interaktive Visualisierung am Beispiel Studierender in Deutschland

Bachelorarbeit
zur Erlangung des akademischen Grades Bachelor of Science

Studiengang:	Medieninformatik Bachelor
Bearbeiter:	Ekaterina Anisimova Matrikelnummer: 848220
Eingereicht am:	01.04.2019
Betreuer:	Prof. Dr. Agathe Merceron
Gutachter:	Prof. Dr. Heike Ripphausen-Lipa

Inhaltsverzeichnis

Glossar

Agile Softwareentwicklung	Agile Softwareentwicklung bezeichnet Ansätze im Softwareentwicklungsprozess, die die Transparenz und Flexibilität erhöhen und zu einem schnelleren Einsatz der entwickelten Systeme führen sollen, um so Risiken im Entwicklungsprozess zu minimieren. Dr. Markus Siepermann. Seite "Agile Softwareentwicklung". In: Gabler Wirtschaftslexikon. URL: https://wirtschaftslexikon.gabler.de/definition/agile-softwareentwicklung-53460 [26.03.2019]
Ajax	Ajax (auch AJAX; Akronym von englisch Asynchronous JavaScript and XML) bezeichnet ein Konzept der asynchronen Datenübertragung zwischen einem Browser und dem Server. Dieses ermöglicht es, HTTP-Anfragen durchzuführen, während eine HTML-Seite angezeigt wird, und die Seite zu verändern, ohne sie komplett neu zu laden. Seite "Ajax (Programmierung)". In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 08.12.2018. URL: https://de.wikipedia.org/w/index.php?title=Ajax_(Programmierung)&oldid=183515309 [26.03.2019]
Ausländische Studierende	Ausländische Studierende sind weibliche oder männliche Personen mit einer anderen als der deutschen Staatsangehörigkeit (oder Staatenlose), die an einer deutschen Hochschule immatrikuliert sind. Seite "Ausländische Studierende". In: Wissenschaft weltoffen. URL: http://www.wissenschaftweltoffen.de/glossar/a07_html [26.03.2019]
Benutzerschnittstelle	Die Benutzerschnittstelle (engl. User Interface, kurz UI) ist die Stelle oder Handlung, mit der ein Mensch mit einer Maschine in Kontakt tritt. Seite "Benutzerschnittstelle". In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 18.03.2019. URL: https://de.wikipedia.org/w/index.php?title=Benutzerschnittstelle&oldid=186721522 [26.03.2019]

Bibliothek	Eine Programmbibliothek (kurz Bibliothek) bezeichnet in der Programmierung eine Sammlung von Unterprogrammen/-Routinen, die Lösungswege für thematisch zusammengehörende Problemstellungen anbieten. Seite “Programmbibliothek”. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 30.01.2019. URL: https://de.wikipedia.org/w/index.php?title=Programmbibliothek&oldid=185213072 [26.03.2019]
Bounding Box	Bounding Box oder Hüllkörper ist eine einfache geometrische Figur, die ein komplexes Objekt umschließt.
CDN	Ein Content Delivery Network (CDN), oder auch Content Distribution Network genannt, ist ein Netz regional verteilter und über das Internet verbundener Server, mit dem Inhalte – insbesondere große Mediendateien – ausgeliefert werden. Ein CDN stellt skalierende Speicher- und Auslieferungskapazitäten zur Verfügung und gewährleistet auch bei großen Lastspitzen einen optimalen Datendurchsatz. Seite “Content Delivery Network”. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 28.02.2019. URL: https://de.wikipedia.org/w/index.php?title=Content_Delivery_Network&oldid=186127464 [26.03.2019]
Choroplethenkarte	Eine Choroplethenkarte (auch Flächenkartogramm oder Flächenwertstufenkarte) ist eine thematische Karte, bei der die Gebiete im Verhältnis zur Verteilungsdichte des thematischen Objektes eingefärbt sind. Seite “Choroplethenkarte”. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 15.02.2019. URL: https://de.wikipedia.org/w/index.php?title=Choroplethenkarte&oldid=185717852 [26.03.2018]
conda	Conda ist ein Open Source Paket- und ein Umgebungsverwaltungssystem für die Programmiersprachen Python, R, Ruby, Lua, Scala, Java, JavaScript, C/ C++, FORTRAN, das unter Windows, Mac OS und Linux ausgeführt werden kann. Conda installiert, ausführt und aktualisiert Pakete und deren Abhängigkeiten. Seite “Conda”. In: Conda documantation. URL: https://conda.io/en/latest/ [10.02.2019]

Dekorateur	Ein Dekorateur in Python ist ein beliebiges aufrufbares Python-Objekt welches zur Modifikation einer Funktion oder einer Klasse genutzt wird. Bernd Klein. Seite “Einführung in Dekorateure”. In: Python-Tutorial. URL: https://www.python-kurs.eu/dekorateure.php [25.02.2019]
DOM	Das Document Object Model (DOM, engl. für Dokumenten-Objekt-Modell) ist eine Spezifikation einer Programmierschnittstelle, welche HTML- oder XML-Dokumente als eine Baumstruktur darstellt, in der jeder Knoten ein Objekt ist, welches einen Teil des Dokumentes repräsentiert. Seite “Document Object Model”. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 18.10.2018. URL: https://de.wikipedia.org/w/index.php?title=Document_Object_Model&oldid=181918466 [26.03.2019]
Geokodierung	Unter dem Vorgang der Georeferenzierung, Geokodierung, Geotagging oder Verortung versteht man die Zuweisung raumbezogener Informationen, der Georeferenz, zu einem Datensatz. Inverse Geokodierung (auch „reverse geocoding“ genannt) bezeichnet das Gegenteil: mittels Geokoordinaten werden textuelle Lokationsangaben gesucht, also etwa Städtenamen, Straßennamen und so weiter. Seite “Georeferenzierung”. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 12.03.2018. URL: https://de.wikipedia.org/w/index.php?title=Georeferenzierung&oldid=174938538 [26.03.2019]
GIS	Geoinformationssysteme, Geographische Informationssysteme (GIS) oder Räumliche Informationssysteme (RIS) sind Informationssysteme zur Erfassung, Bearbeitung, Organisation, Analyse und Präsentation räumlicher Daten. Geoinformationssysteme umfassen die dazu benötigte Hardware, Software, Daten und Anwendungen. Seite “Geoinformationssystem”. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 11.12.2018. URL: https://de.wikipedia.org/w/index.php?title=Geoinformationssystem&oldid=183596912 [26.03.2019]

Großkreis	Ein Großkreis ist ein größtmöglicher Kreis auf einer Kugeloberfläche. Sein Mittelpunkt fällt immer mit dem Mittelpunkt der Kugel zusammen und ein Schnitt auf dem Großkreis teilt die Kugel in jedem Fall in zwei („gleich große“) Hälften. Seite “Großkreis”. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 29.03.2018. URL: https://de.wikipedia.org/w/index.php?title=Gro%C3%9Fkreis&oldid=175521580 [26.03.2019]
Framework	Ein Framework (englisch für Rahmenstruktur) ist ein Programmiergerüst, das in der Softwaretechnik verwendet wird. Es stellt für Applikationen eine wiederverwendbare, gemeinsame Struktur zur Verfügung. Seite “Framework”. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 10.08.2018. URL: https://de.wikipedia.org/w/index.php?title=Framework&oldid=179886028 [26.03.2019]
KBS	Ein Koordinatenreferenzsystem oder Koordinatenbezugsystem (KBS), engl. coordinate reference system, international mit CRS abgekürzt, ist ein Koordinatensystem, das durch Verknüpfung mit einem Datum auf die reale Welt bezogen ist. Zur Beschreibung der Erde wird das Geodätische Datum oder Vertikale Datum verwendet. Seite “Koordinatenreferenzsystem”. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 09.03.2018. URL: https://de.wikipedia.org/w/index.php?title=Koordinatenreferenzsystem&oldid=174845462 [26.03.2019]
Model-View-Presenter	Model-View-Presenter (MVP) ist ein Entwurfsmuster in der Softwareentwicklung, das aus dem Model-View-Controller (MVC) hervorgegangen ist. Es beschreibt einen Ansatz, um das Modell (engl. model) und die Ansicht (engl. view) komplett voneinander zu trennen und über einen Präsentierer (engl. presenter) zu verbinden. Seite “Model-View-Presenter”. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 07.01.2019. URL: https://de.wikipedia.org/w/index.php?title=Model-View-Presenter&oldid=184499822 [26.03.2019]

Modul	Modul in der Programmiersprache Python ist eine Datei, die Definitionen, Anweisungen und Funktionen beinhaltet. Diese Funktionen können später in mehreren Programmen verwendet werden, ohne sie erneut definieren zu müssen.
ORM	Objektrelationale Abbildung (englisch object-relational mapping, ORM) ist eine Technik der Softwareentwicklung, mit der ein in einer objektorientierten Programmiersprache geschriebenes Anwendungsprogramm seine Objekte in einer relationalen Datenbank ablegen kann. Seite "Objektrelationale Abbildung". In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 24.09.2018. URL: https://de.wikipedia.org/w/index.php?title=Objektrelationale_Abbildung&oldid=181202941 [26.03.2019]
pip	pip ist ein rekursives Akronym für Pip Installs Python und ist das Standardverwaltungswerkzeug für Python-Module. Seite "pip". In: Wiki ubuntuusers.de. Bearbeitungsstand: 03.06.2018 . URL: https://wiki.ubuntuusers.de/pip/ [26.03.2019]
Studierende	An deutschen Hochschulen in einem Fachstudium eingeschriebene (immatrikulierte) weibliche und männliche Personen ohne Beurlaubte, Studienkollegiaten und Gasthörern". Seite "Studierende". In: Wissenschaft weltoffen. URL: http://www.wissenschaftweltoffen.de/glossar/s09_html [26.03.2019]
Technologie Stack	Technologie Stack bezeichnet eine Reihe von Technologien (Softwarekomponenten), die in einem Projekt benutzt werden.
Web GIS	Unter dem Begriff Web GIS wird im Allgemeinen eine GIS-Anwendung verstanden, deren Kernfunktionen auf für Geodaten spezialisierte Webservices (Geodienste) zurückgreifen. Seite "Web GIS". In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 13.03.2018, 15:42 UTC. URL: https://de.wikipedia.org/w/index.php?title=Web_GIS&oldid=174978293 [26.03.2019]

Wireframe

Wireframe ist einen sehr frühen konzeptionellen Entwurf einer Website oder eines Software-Frontends. Dabei spielt die Gestaltung und Funktion noch keine Rolle. Das Augenmerk liegt auf der Anordnung von Elementen und der Benutzerführung.
Seite “Wireframe”. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 27.09.2016. URL: https://de.wikipedia.org/w/index.php?title=Wireframe&oldid=158260420 [26.03.2019]

WSGI

Das Python Web Server Gateway Interface (WSGI) ist eine Schnittstellen-Spezifikation für die Programmiersprache Python, die eine Schnittstelle zwischen Webservern und Webframeworks festlegt, um die Plattformunabhängigkeit von Webanwendungen auf unterschiedlichen Webservern zu fördern.
Seite “Web Server Gateway Interface”. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 15.01.2019. URL: https://de.wikipedia.org/w/index.php?title=Web_Server_Gateway_Interface&oldid=184764686 [26.03.2019]

Abkürzungen

IDE	Integrierte Entwicklungsumgebung
DS1	Datensatz “Studierendenzahl nach Bundesland, Nationalität und Geschlecht WS 1998/99 - WS 2016/17”
DS2	Datensatz “Hochschulen nach Gründungsjahr von 1386 bis 2017”
DS3	Datensatz “Studierende nach Geschlecht, Land des Studienortes und Land des Erwerbs der Hochschulzugangsberechtigung WS 2006/07 - WS 2017/18”
GDF	Datensatz mit den Koordinaten der Bundesländer im GeoJSON Format
GDT	Tabelle mit den Daten für die Tooltips für den Datensatz DS1
DS3_T	Tabelle mit den Daten für die Tooltips für den Datensatz DS3

Abbildungsverzeichnis

Listingverzeichnis

1. Einleitung

Moderne Informations- und Kommunikations- Technologien generieren heutzutage jeden Tag mehrere Exabytes (10^{18} Byte) von Daten. Laut dem Statistik-Portal Statista[1] im Jahr 2018 war das gesamte Volumen von digitalen Datenmengen 33 Zettabytes und im Jahr 2025 soll sich das Datenaufkommen auf 175 Zettabyte erhöhen. Solche enorme Mengen von Daten können von dem menschlichen Gehirn nicht mehr adäquat verarbeitet und gespeichert werden. Die kognitive Belastung wird immer höher und man versucht verschiedene Methoden einzusetzen um den wachsenden Belastungsgrad zu reduzieren. Eine von solchen Methoden ist die visuelle Darstellung der Informationen. Die Visualisierung hilft beim Erkunden und Verstehen komplexer Sachverhalte und Prozesse und verfolgt das Ziel abstrakte Daten grafisch darzustellen. Bilder und Grafiken sind ein wichtiger Bestandteil bei der Wissensaufnahme, da sie schneller und einfacher von dem Gehirn erfasst werden[2].

Es gibt verschiedene Arten der Visualisierungen, wie bspw. Logo, Diagramm, Schema, Grafik, Tabelle, Karte u.v.a. Je nach Anwendungsfall wird die eine oder andere Art verwendet. Eine grundlegende Methode ortsbezogene Daten zu visualisieren heißt Social Mapping. Dadurch lassen sich Karten erstellen, die einen Überblick über räumliche Strukturen und Prozesse erlauben. Neben dem schlichten Datenwert wird es die zusätzliche Information, wo sich dieser Wert befindet angezeigt. Hier werden die sozialen Geodaten und Geoinformationssysteme verwendet. Soziale Daten sind Daten, die mit der Person in Verbindung stehen. Geodaten[3] sind digitale Informationen, denen auf der Erdoberfläche eine bestimmte räumliche Lage zugewiesen werden kann. Die Visualisierung solcher Daten liefert nicht nur ein besseres Verständnis und tiefere Einsicht in den Forschungsbereich, sondern ermöglicht auch Trends und Phänomene frühzeitig zu erkennen.

Es gibt viele hervorragende Beispiele dafür wie große Datenmengen effizient auf einer Karte dargestellt werden können. Unter anderen kann man folgende Projekte hervorheben.

Die Webseite[4], bei der die Rohdaten des Fahrradverleihs in eine lesbare, nutzbare und schöne Visualisierung umgesetzt wurden. Aktuell werden Hunderte Millionen Daten von Radfahrern aus 25 Städten in Taiwan, den Vereinigten Staaten, Kanada, Großbritannien, Deutschland, Norwegen und Mexiko analysiert.

Ein anderes Projekt[5] aus der USA stellt die weltweite Geschichte und die Ursachen von Fluchtbewegungen seit 1975 bis 2017 vor.

---

[1] https://de.statista.com/statistik/daten/studie/267974/umfrage/prognose-zum-weltweit-generierten-datenvolumen [18.01.2019]

[2] [NRW]

[3] https://de.wikipedia.org/wiki/Geodaten [18.01.2019]

[4] https://www.visualization.bike/ [07.02.2019]

[5] http://www.therefugeeproject.org [07.02.2019]

Die letzten zwei Beispiele kommen von der Berliner Morgenpost. Im Projekt “Berlin an deiner Linie”[6] werden soziodemographische Merkmale und die Infrastruktur der Stadt entlang den Linien der öffentlichen Verkehrsmitteln untersucht.

Auf der Seite “Bundestagswahl 2017 im Detail”[7] befindet sich eine interaktive Karte, die detaillierte Ergebnisse der Bundestagswahl 2017 - für jede Partei, in jedem Ort zeigt.

Diese hervorragenden Beispiele sind inspirierend und wecken das Interesse sich näher mit dem Thema zu befassen. Der Aufwand der mit dem Entwicklungsprozess einhergeht und die Arbeit die damit verbunden ist führen zu einer intensiven Beschäftigung mit dem Thema.

[6] https://interaktiv.morgenpost.de/berlin-an-deiner-linie/ [07.02.2019]
[7] https://interaktiv.morgenpost.de/gemeindekarte-bundestagswahl-2017/ [07.02.2019]

2. Aufgabenstellung

Das Ziel dieser Arbeit ist es, eine Vorgehensweise zur anschaulichen Visualisierung von Geodaten zu identifizieren und umzusetzen. Dabei werden unterschiedliche Technologien und Bibliotheken untersucht und nach den Anforderungen entsprechend ausgewählt.

Im Rahmen des Projektes sollen drei Karten-Visualisierungen erstellt werden. Dabei wird beispielhaft das Thema "Studierende in Deutschland" behandelt und folgende Aspekte genauer betrachtet:

- Verteilung der Studierenden nach Bundesländern
- Anteil von ausländischen Studierenden
- Entstehung der Hochschulen in Deutschland

Der Schwerpunkt des Projektes liegt im Umgang mit Daten und der Erstellung einer Webanwendung mit einer interaktiven Visualisierung. Die Daten werden auf einer Karte von Deutschland dargestellt mit der Möglichkeit verschiedene Ansichten und Zeitperioden auswählen zu können.

Die Arbeit kann in vier Teile gegliedert werden:

- Recherche nach offenen Datenquellen, die Auswahl und die Vorbereitung von Daten
- Analyse und Auswahl von existierenden Programmbibliotheken und Frameworks
- Entwicklung einer Webanwendung
- Deployment

Der Umgang mit den Daten ist der wesentliche und umfangreiche Teil dieser Arbeit. Schon bevor das Thema der Bachelorarbeit feststand, wurde eine Recherche nach offenen Datenquellen durchgeführt. Das entscheidende Kriterium für die Themenauswahl war die Verfügbarkeit der Daten. Folgende Datenquellen werden in dieser Arbeit verwendet:

- DESTATIS[8] Statistisches Bundesamt: Das Statistische Bundesamt ist der führende Anbieter qualitativ hochwertiger statistischer Informationen über Deutschland.
- OffeneDaten.de[9] ist ein Datenkatalog für offene Daten in Deutschland. Die Seite wird von der Open Knowledge Foundation Deutschland betrieben.
- GovData[10], das Datenportal für Deutschland, bietet einen einheitlichen, zentralen Zugang zu Verwaltungsdaten aus Bund, Ländern und Kommunen.
- Das Amt für Statistik Berlin-Brandenburg[11] bietet im Internet kostenfrei Daten für das Land Berlin und seine Bezirke sowie für das Land Brandenburg, seine Kreise und kreisfreien Städte.

[8] https://www.destatis.de/DE/Startseite.html [14.01.2019]
[9] https://offenedaten.de/ [14.01.2019]
[10] https://www.govdata.de [14.01.2019]
[11] https://www.statistik-berlin-brandenburg.de [14.01.2019]

- Die Internetseite "Wissenschaft weltoffen"[12] ergänzt die gleichnamige Publikation im W. Bertelsmann Verlag und stellt zur Verfügung die relevante Daten zur Internationalität von Studium und Lehre in Deutschland.

Außerdem sollen die Datensätze unbedingt folgende Attribute wie Jahr und Bundesland beinhalten. Die Attribute Geschlecht, Nationalität, Altersgruppe sind optional, aber wünschenswert.

[12] http://www.wissenschaft-weltoffen.de/ [14.01.2019]

3. Fachliches Umfeld

Dieses Kapitel beschäftigt sich mit den Grundlagen zum Thema der Bachelorarbeit. Es bietet einen Überblick in das fachliche Umfeld und präsentiert ein breites Spektrum von vorhandenen Bibliotheken und Frameworks, die unter verschiedenen Aspekten betrachtet werden. Im Anschluss werden vorgestellte Entwicklungswerkzeuge nach Kriterien, wie Lizenz, Einstiegsschwierigkeiten, Merkmalen, Anwendungsbeispiele und Dokumentation bewertet.

Technisch kann die Visualisierung nach verschiedenen Verfahren umgesetzt werden. Mit HTML und CSS können elementare Grafiken gestaltet werden. SVG, oder Scalable Vector Graphics gewährleistet die Skalierbarkeit der Abbildungen ohne Qualitätsverlust. Diese Herangehensweise hat ihre Schwächen, weil eine anspruchsvolle Abbildung mehrere tausende Punkte beinhaltet, die zum Leistungsverlust des Systems führen können. Mit Hilfe von Canvas werden größere Datenmengen visualisiert. Ein Canvas-Element ist ein Bereich mit angegebenen Höhen und Breiten, in den per JavaScript gezeichnet werden kann. WebGL (Web Graphics Library) ist eine JavaScript-Programmierschnittstelle, mit deren Hilfe 3D-Grafiken hardwarebeschleunigt im Webbrowser ohne zusätzliche Erweiterungen dargestellt werden können.

Für eine Webanwendung, der eine interaktive Visualisierung zugrunde liegt, ist die Verwendung von diesen Verfahren nicht effizient. In der Praxis werden moderne Visualisierungsbibliotheken eingesetzt. Sie erleichtern und beschleunigen den Entwicklungsprozess, da viele Komponenten oder Bedienungselemente bereits zur Verfügung stehen.

3.1. Datenvorbereitung

Erster Schritt in der Arbeit mit Daten ist die Vorbereitung von Daten. Die Datensätze sollen in ein bestimmtes Format umgewandelt werden, welches die ausgewählte Bibliothek anfordert. Für die Datenvorbereitung werden zwei Programmiersprachen am häufigsten verwendet.

R[13] ist die Programmiersprache der Wissenschaft, die für statistische Berechnungen und Grafiken entwickelt wurde. Diese Sprache eignet sich gut für die Datenverarbeitung, hat aber eine hohe Einarbeitungsschwelle und lässt sich schwer mit anderen Programmiersprachen kombinieren.

Python[14] ist eine weit verbreitete Programmiersprache, die einfach zu erlernen ist. Diese ist sowohl für Skripte als auch für schnelle Anwendungsentwicklung (Rapid Application Development) hervorragend geeignet. Python besitzt eine riesige Community, die für ständige Weiterentwicklung und die Aktualisierung von Bibliotheken sorgt. Das

[13] [RPR]
[14] [PYT]

Python Ökosystem bietet **Jupyter Notebook**[15], ein interaktives Open-Source Browser basiertes Werkzeug. Jupyter Notebook kann für schnelle Experimente mit Daten und für ausführliche Verarbeitung, Transformation und Visualisierung von Datensätzen verwendet werden. Um die Entwicklungsumgebung zu definieren wird das Paket- und Umgebungsverwaltungssystem **Anaconda**[16] benutzt. Anaconda ist kostenlos und systemunabhängig. Bei der Installation erstellt Anaconda eine Basis-Entwicklungsumgebung mit den vorinstallierten Python-Modulen[17]. Anaconda bietet die Möglichkeit mehrere Umgebungen zu definieren. Diese Umgebungen können unterschiedliche Versionen von Python und installierten Paketen haben. Wenn eine neue Projektumgebung erstellt wird, enthält sie noch keine Python-Module, welche über die eigene Paketverwaltung conda oder einen Paketmanager pip installiert werden können. Die separate Projektumgebung für jedes Projekt ist nötig, damit die Konflikte zwischen unterschiedlichen Versionen von Softwarepaketen und ihren Abhängigkeiten vermieden werden können.

Für die Datenvorbereitung werden folgende Bibliotheken benutzt.

Pandas[18] ist eine Programmbibliothek für die Programmiersprache Python, die Hilfsmittel für die Verwaltung von Daten und deren Analyse anbietet. Insbesondere enthält sie Datenstrukturen und Operatoren für den Zugriff auf numerische Tabellen und Zeitreihen.

Da viel mit der Pandas Bibliothek gearbeitet wird, sollen die meistgenutzten Datenstrukturen und Datentypen erklärt werden. Pandas zugrunde liegen zwei Datenstrukturen: Series und DataFrame. Eine Series entspricht einem eindimensionalen Array. Die Datenstruktur DataFrame entspricht einer Excel-Tabelle. Ein DataFrame besitzt Spalten- und Zeilenindex. Spaltenindex entspricht den Spaltennamen, das heißt jede Spalte kann mit dem Namen angesprochen werden. Außerdem können die verschiedenen Spalten des DataFrame Objektes verschiedene Datentypen haben. Die Übersicht der unterstützten Datentypen und ihre Entsprechung in Python sind in der Abbildung 3.1.vorgestellt.

Pandas dtype	Python type	NumPy type	Usage
object	str	string_, unicode_	Text
int64	int	int_, int8, int16, int32, int64, uint8, uint16, uint32, uint64	Integer numbers
float64	float	float_, float16, float32, float64	Floating point numbers
bool	bool	bool_	True/False values
datetime64	NA	NA	Date and time values
timedelta[ns]	NA	NA	Differences between two datetimes
category	NA	NA	Finite list of text values

Abb. 3.1: *Übersicht der Datentypen in Python und Pandas*[19]

[15] [JUP]
[16] [ANA]
[17] Die Liste von vorinstallierten Modulen https://docs.anaconda.com/anaconda/packages/pkg-docs/ [26.02.2019]
[18] [PAN]
[19] Quelle: https://pbpython.com/pandas_dtypes.html [19.03.2019]

GeoPandas[20] ist ein Open Source-Projekt, das die Arbeit mit Geodaten in Python vereinfacht. GeoPandas erweitert die von Pandas verwendeten Datentypen, um räumliche Operationen an geometrischen Typen zu ermöglichen. Die wichtigsten Datenstrukturen in GeoPandas sind GeoSeries und GeoDataFrame. Eine GeoSeries ist ein Vektor, wobei jedes Vektorelement eine Menge von Geometrieformen oder nur eine Form ist. Die unterstützten Geometrietypen sind in der Tabelle 1 aufgeführt. Ein GeoDataFrame ist ein DataFrame, der eine GeoSeries Spalte beinhaltet. Diese Spalte wird als "geometry" bezeichnet und dient zur räumlichen Abbildung der Daten.

NumPy[21] ist eine Programmbibliothek für die Programmiersprache Python, die eine einfache Handhabung von Vektoren, Matrizen oder generell großen mehrdimensionalen Arrays ermöglicht. Neben den Datenstrukturen bietet NumPy auch effizient implementierte Funktionen für numerische Berechnungen an.

json[22] ist eine Programmbibliothek für die Programmiersprache Python, die als JSON-Encoder und Decoder dient.

xlrd[23] ist eine Programmbibliothek für die Programmiersprache Python, die die Extraktion von Daten aus Excel-Tabellen (.xls und .xlsx, Version 2.0 und höher) auf einer beliebigen Plattform anbietet.

Die Autorin ist bereits mit der Programmiersprache Python vertraut und darin bestrebt ihre Python-Kenntnisse zu vertiefen. Die Datenverarbeitung wird deswegen mithilfe von Python vollzogen.

3.2. Backend

Python ist auch für die Entwicklung der Webanwendungen sehr gut geeignet. Es gibt mehrere Bibliotheken, die eine schnelle Lösung für die Einrichtung eines Webservers liefern. Davon werden folgende drei Frameworks am häufigsten benutzt.

Django[24] ist ein in Python geschriebenes, quelloffenes Webframework, das einem Model-View-Presenter-Schema (MVP) folgt. Django wurde für die schnelle Entwicklung komplexer Webanwendungen entwickelt. Dieses Framework beinhaltet alle Werkzeuge, die zu schneller Implementierung und Entwicklung von skalierbaren, zuverlässigen und wartbaren Webanwendungen benötigt werden. Django hat eine steile Lernkurve, die von den unerfahrenen Entwicklern mehr Zeit für die Einarbeitung benötigt. Die Umsetzung von Django lohnt sich für folgende Arten von Anwendungen:

- ein Onlineshop, da Django bereits die E-Commerce-Module und Object-Relational Mapping (ORM) Möglichkeiten in sich enthält.

[20] [GEO]
[21] [NUM]
[22] [JSO]
[23] [XLR]
[24] [DJA]

- Online-Medien, da das Framework ursprünglich dafür entwickelt wurde. Django ist sehr verbreitet und wird in den großen bekannten Projekten, wie Pinterest, Disqus, Eventbrite, Instagram, Nextdoor u.a. angewendet.

Flask[25] ist ein in Python geschriebenes Webframework. Der Fokus von Flask liegt auf guter Dokumentation und Erweiterbarkeit. Die einzigen Abhängigkeiten sind Jinja2, eine Template-Engine, und Werkzeug, eine Bibliothek zum Erstellen von Web Server Gateway Interface (WSGI)-Anwendungen. Eine der größten Stärken von Flask ist der Minimalismus und die Einfachheit. Flask bietet eine große Freiheit und Auswahl von externen Bibliotheken, was die Implementierung ganz nach eigenen Anforderungen möglich macht. Der Einsatz von Flask lohnt sich für die Entwicklung einer einfachen Anwendung mit statischem Inhalt. Flask wird in folgenden bekannten Projekten umgesetzt: Twilio, Netflix, Uber, LinkedIn.

Bottle[26] ist ein schnelles, einfaches und leichtes WSGI- Micro-Web-Framework für Python. Es wird als einzelnes Dateimodul verteilt und hat keine anderen Abhängigkeiten als die Python-Standardbibliothek. Bottle eignet sich eher für kleine und anspruchslose Projekte.

Um ein den Projektanforderungen passendes Framework auszuwählen, werden die grundlegende Merkmale zusammengefasst.

Django	
Lizenz:	BSD-Lizenz[27]
Einstiegsschwierigkeiten:	hoch
Merkmale:	enthält die meisten Bibliotheken und Frameworks verfügt über eine eigene Template-Engine liegt viel Wert auf die Sicherheit hohe Skalierbarkeit enthält Admin-Framework
Anwendungsbeispiele:	Pinterest, Disqus, Eventbrite, Instagram, Nextdoor
Dokumentation:	ausführliche Dokumentation, große Entwickler-Gemeinschaft 39484 Sterne auf Github, 1700 Mitwirkende [14.02.2019]
Flask	
Lizenz:	BSD-Lizenz
Einstiegsschwierigkeiten:	niedrig
Merkmale:	enthält erforderliches Minimum an Bibliotheken basiert auf Jinja2 Template-Engine sehr flexibel
Anwendungsbeispiele:	Twilio, Netflix, Uber, LinkedIn

[25] [FLA]
[26] [BOT]
[27] BSD-Lizenz bezeichnet eine Gruppe von freizügigen Open-Source-Lizenzen. Der Urtyp der Lizenz stammt von der University of California, Berkeley (UCB), worauf das Akronym BSD hinweist: Berkeley Software Distribution https://de.wikipedia.org/wiki/BSD-Lizenz [14.02.2019]

Dokumentation:	sehr gut dokumentiert große Entwickler-Gemeinschaft 41932 Sterne auf Github, 503 Mitwirkende [14.02.2019]
Bottle	
Lizenz:	MIT[28]
Einstiegsschwierigkeiten:	niedrig
Merkmale:	minimalistisch verfügt über eine integrierte Template-Engine namens SimpleTemplate Engine
Anwendungsbeispiele:	kleine, private Projekte
Dokumentation:	gute Dokumentation kleine Entwickler-Gemeinschaft 5892 Sterne auf GitHub, 154 Mitwirkende [14.02.2019]

In dieser Arbeit wird für die Einrichtung vom Webserver Flask benutzt, da dieses Framework eine niedrige Eintrittsschwelle, einen hohen Flexibilitätsgrad, eine gute Dokumentation und eine große Entwickler-Gemeinschaft hat. Es erfüllt alle Anforderungen des Projektes und ist auf dem Markt sehr gefragt, sodass die Erlernung dieses Frameworks für den weiteren Berufsweg einen wesentlichen Pluspunkt bringt.

3.3. Frontend

Der andere wichtige Bestandteil jeder Webanwendung ist die Benutzerschnittstelle, oder auf englisch User Interface (UI). Die Benutzerschnittstelle soll möglichst simpel und intuitiv gestaltet werden, so dass die Benutzung von Oberfläche zu keiner Verwirrung führt. Außerdem soll die Webanwendung in allen Browsern gleich aussehen, ohne dass Elemente verrutschen oder unsichtbar werden. Dabei ist wichtig zu beachten, dass die Bedienungselemente auf jeder Seite der Webanwendung gleich aussehen. Um diese Anforderungen zu erfüllen, kommen UI-Frameworks den Entwicklern zu Hilfe. Das am meisten verwendete heißt Bootstrap, ein Frontend Framework von Twitter.

Bootstrap[29] ist ein freies Frontend-CSS-Framework. Es enthält auf HTML und CSS basierende Gestaltungsvorlagen, sowie zusätzliche, optionale JavaScript-Erweiterungen. Bootstrap ermöglicht eine schnelle Entwicklung von responsiven Webseiten.

Für die Manipulation von HTML-Elementen wird sowohl das reine JavaScript als auch jQuery benutzt.

jQuery[30] ist eine freie browserübergreifende JavaScript-Bibliothek, die Navigation durch HTML-Seiten und die Manipulation von Document Object Model (DOM)-Elementen vereinfacht. jQuery stellt die Ajax-Schnittstelle (Asynchronous JavaScript and XML) zur

[28] Die MIT-Lizenz, auch X-Lizenz oder X11-Lizenz genannt, ist eine aus dem Massachusetts Institute of Technology stammende Freizügige Open-Source-Lizenz https://de.wikipedia.org/wiki/MIT-Lizenz [14.02.2019]
[29] [BOO]
[30] [JQU]

Verfügung. Ajax ermöglicht die einzelne Bereiche der Webseite austauschen, ohne die Seite neu zu laden.

3.4. Visualisierung

Der Fokus dieser Arbeit liegt auf der Visualisierung von Geodaten. Die Daten können effizient auf einer Karte dargestellt werden, da diese eine räumliche Anordnung besitzen. Für die Visualisierung von raumbezogenen quantitativen Daten eignet sich gut die Choroplethenkarte. Mit den Farbabstufungen kann die Verteilung von Daten für die jeweilige Region der Karte dargestellt werden. Manchmal ist die kartografische Darstellung von Daten nicht ausreichend. In diesen Fällen können klassische Diagramme, wie Linien- oder Balkendiagramme angewendet werden, um die Kartendarstellung mit der zusätzlichen Information zu bereichern.

Für die Visualisierung im Browser nimmt JavaScript Platz Eins unter den Programmiersprachen ein, da diese speziell für die Kommunikation mit HTML-Elementen entwickelt wurde. JavaScript hat zahlreiche Bibliotheken für die Visualisierung.

D3.js[31] (oder D3, engl. Data-Driven Documents) ist eine JavaScript-Bibliothek zum Erstellen dynamischer, interaktiver Datenvisualisierungen im Webbrowser. Es verwendet die SVG- , HTML5- und CSS- Standards. Im Gegensatz zu vielen anderen Bibliotheken ermöglicht D3.js eine gute Kontrolle über das endgültige visuelle Ergebnis. Gleichzeitig ist sie sehr umfangreich und setzt aufgrund ihrer Komplexität und granulären Kontrollmöglichkeiten einen großen Zeitaufwand für die Einarbeitung voraus.

Plotly.js[32] ist eine Open-Source JavaScript-Bibliothek zum Erstellen von Diagrammen und Dashboards. Sie ist auf D3.js (und stack.gl) aufgebaut. Der Hauptunterschied zwischen D3.js und Plotly.js ist, dass Plotly.js eine Bibliothek speziell für die Erstellung von Diagrammen ist.

Leaflet[33] ist eine leichte Open-Source JavaScript-Bibliothek, mit der WebGIS-Anwendungen erstellt werden können. Die Bibliothek verwendet HTML5, CSS3 und unterstützt somit die meisten Desktop- und Mobil-Browser. Sie verfügt über vielfältige Mapping-Funktionen und lässt sich mit vielen Plugins erweitern. Leaflet hat eine gut strukturierte und dokumentierte API.

Da für die Entwicklung der Webanwendung im Rahmen dieser Bachelorarbeit Python als Hauptprogrammiersprache ausgewählt wurde, wäre es konsequent auch im Frontend möglichst viel mit Python zu realisieren.

Python bietet eine große Auswahl von Visualisierungsbibliotheken an.

Folium[34] ist eine Open-Source Python-Bibliothek, die auf Leaflet.js aufgebaut ist. Sie ermöglicht die Anwendung von Leaflet Funktionalitäten in Python.

---

[31] [D3J]
[32] [PLO]
[33] [LEA]
[34] [FOL]

Bokeh[35] ist eine Open-Source Python-Bibliothek, die die Erstellung von interaktiven Diagrammen und Grafiken ermöglicht. Sie hat eine ausführliche Dokumentation mit zahlreichen Beispielen. Bokeh unterscheidet sich von anderen Bibliotheken darin, dass sie den Benutzern die Interaktion mit den übermittelten Daten liefert.

Diese Bachelorarbeit beschäftigt sich hauptsächlich mit kartografischen Visualisierungen, deswegen soll das Datenformat GeoJSON[36] erläutert werden. Die Visualisierung von Geodaten fordert die Anbindung der geografischen Daten. Es gibt mehrere Dateiformate, die die Bearbeitung von geografischen Daten auf einem Computer ermöglichen. In dieser Arbeit wird ein GeoJSON Format verwendet, ein offenes Format zum Kodieren geografischer Daten. GeoJSON unterstützt die folgenden Geometrietypen: Point, LineString, Polygon, MultiPoint, MultiLineString, und MultiPolygon. Geometrische Objekte können zusätzlichen Eigenschaften enthalten, diese heißen Feature Objekte.

Beispiele[37] der unterstützten Geometrietypen sind in der Tabelle 1 aufgeführt:

Point (Punkt)	`{ "type": "Point",`
	`  "coordinates": [30, 10]`
	`}`

LineString
(Linie)
	`{ "type": "LineString",`
	`  "coordinates": [`
	`  [30, 10], [10, 30], [40, 40]`
	`  ]`
	`}`

Polygon	`{ "type": "Polygon",`
	`  "coordinates": [`
	`  [[30, 10], [40, 40], [20, 40], [10, 20], [30, 10]]`
	`  ]`
	`}`

MultiPoint	`{ "type": "MultiPoint",`
	`  "coordinates": [`
	`  [10, 40], [40, 30], [20, 20], [30, 10]`
	`  ]`
	`}`

MultiLineString	`{ "type": "MultiLineString",` `  "coordinates": [`

[35] [BOK]
[36] http://geojson.org/ [21.01.2019]
[37] Quelle: https://de.wikipedia.org/wiki/GeoJSON [21.01.2019]

	[[10, 10], [20, 20], [10, 40]], [[40, 40], [30, 30], [40, 20], [30, 10]]] }
MultiPolygon	{ "type": "MultiPolygon", "coordinates": [[[[40, 40], [20, 45], [45, 30], [40, 40]]], [[[20, 35], [10, 30], [10, 10], [30, 5], [45, 20], [20, 35]], [[30, 20], [20, 15], [20, 25], [30, 20]]]] }

Tabelle 1: *GeoJSON Geometrietypen*

3.5. Technologie Stack

Die Softwarekomponenten aus den vorherigen Abschnitten dieses Kapitels ergeben folgenden Technologie Stack.

Datenaufbereitung:	Anaconda 4.5.12, Jupyter Notebook 5.7.4, pandas 0.24.0, geopandas 0.4.0, numpy 1.15.4, json 0.13.1, xlrd 1.2.0
Backend:	Python 3.6.6, Flask 1.0.2,
Frontend:	jQuery 3.3.1, Twitter Bootstrap 4.2.1
Visualisierung:	Folium 0.7.0, Leaflet 1.4.0 und verschiedene Plugins, Bokeh 1.0.4
IDE:	PyCharm 2017.3.2 (Professional Edition)

PyCharm[38] ist eine integrierte Entwicklungsumgebung (IDE) des Unternehmens JetBrains für die Programmiersprache Python. Es gibt eine kostenlose, quelloffene Community Version sowie eine Professional Version. Die Studierende an einer anerkannten Hochschule können eine kostenlose Lizenz beantragen.

Die Entwicklung im Rahmen dieser Bachelorarbeit erfolgt unter dem Betriebssystem macOS Mojave 10.14.2. Als Versionsverwaltungssystem wird Github benutzt. Der Link zum Repositorium: https://github.com/ekatja/bachelor-beuth-2019

[38] https://www.jetbrains.com/pycharm/ [06.02.2019]

4. Zielbestimmung

Das Thema dieser Bachelorarbeit wurde von der Autorin selber ausgesucht und formuliert. Das Projekt ist mit keinen strengen Anforderungen und äußeren Einschränkungen belastet aus diesem Grund verfügt das Projekt über einen hohen Grad an Freiheit und Kreativität. Begrenzt ist die Arbeit von den Daten, die aus offenen Datenquellen gewonnen wurden. Für solche Projektarten mit explorativer Zielsetzung lohnt sich die Entwicklung nach der agilen Methodik. Laut agilen Entwicklungsmethoden reichen nur allgemeine Vorgaben an das ganze Projekt. Es musste kein detailliertes Pflichtenheft erarbeitet werden. Diese Vorgehensweise unterscheidet sich durch ihre Flexibilität und den iterativen Entwicklungsprozess. Die Projektanforderungen werden in kleinen abgeschlossenen Zyklen bearbeiten, sodass nach jeder Iteration ein minimales funktionsfähiges Projekt entsteht. Agile Methodik erlaubt Änderung, Ergänzung und Abschaffung von Projektanforderungen im Laufe des Entwicklungsprozesses. Das erlaubt jederzeit die Anpassungen vorzunehmen und flexibel zu bleiben.

Im Rahmen dieser Bachelorarbeit wird eine Webanwendung entwickelt, welche die Visualisierung erfassten Daten mittels Choroplethenkarten ermöglicht. Die Datensammlung besteht aus drei Datensätzen, für jeden davon wird eine eigene Seite gemäß der gemeinsamen Vorlage erstellt. Die einzelnen Seiten werden sich nur durch das Vorhandensein bestimmter Bedienelemente voneinander unterscheiden. Daraus lassen sich folgende allgemeine Anforderungen an das Projekt definieren.

4.1. Muss-Kriterien

M1. Datensatz auswählen

Der Benutzer hat die Möglichkeit folgende Datensätze auf der Webseite anzeigen zu lassen:

1. Studierende (Anzahl) nach Bundesland, Nationalität und Geschlecht WS 1998/99 - WS 2016/17[39]
2. Hochschulen nach Gründungsjahr von 1386 bis 2017[40]
3. Studierende nach Geschlecht, Land des Studienortes und Land des Erwerbs der Hochschulzugangsberechtigung WS 2006/07 - WS 2017/18[41]

Gleichzeitig kann nur ein Datensatz angezeigt werden. Dabei soll jede Visualisierung eine eigene URL besitzen.

[39] Quelle: © Statistisches Bundesamt (Destatis), 2018 https://www-genesis.destatis.de/genesis/online/
[40] Quelle: https://www.hochschulkompass.de/hochschulen/downloads.html
[41] Quelle: Statistisches Bundesamt, 2018 http://www.datenportal.bmbf.de/portal/de/Tabelle-2.5.32.csv „dl-de/by-2-0" (https://www.govdata.de/dl-de/by-2-0)

M2. Zeitraum auswählen

Der Benutzer kann das Jahr abhängig von den ausgewählten Datensätzen auswählen. Die Zeitraumauswahl erfolgt durch eine Auswahlliste oder einen Schieberegler.

M3. Filtern nach Geschlecht

Der Benutzer kann die Daten nach Geschlecht filtern, dabei gibt es drei Möglichkeiten: männlich, weiblich, insgesamt.

M4. Filtern nach Nationalität

Der Benutzer kann die Daten nach Nationalität filtern, dabei gibt es drei Möglichkeiten: Deutsche, Ausländer, insgesamt

M5. Karte skalieren

Der Benutzer kann die Karte vergrößern, verkleinern und/oder verschieben

M6. Zusätzliche Information durch Pop-Ups/Tooltips

Wenn der Benutzer mit der Maus über einen bestimmten Bereich (Bundesland, Stadt, Hochschule u. a.) auf der Karte fährt oder auf einen Marker klickt, soll eine zusätzliche Information über diesen Bereich angezeigt werden.

Die Auswahl des Datensatzes und anderen Parametern erfolgt durch einen Bereich mit den Bedienungselementen, wie Auswahlliste, Radio-Buttons und Schieberegler.

4.2. Kann-Kriterien

K1. Diagramm und/oder Tabelle anzeigen

Je nach dem ausgewählten Datensatz kann auf der Webseite ein Diagramm und/oder eine Tabelle mit zusätzlicher Information angezeigt werden.

K2. Animation abspielen

Je nach dem ausgewählten Datensatz kann der Benutzer eine Zeitanimation abspielen.

4.3. Abgrenzungskriterien

A1. Im Rahmen der Bachelorarbeit wird nur die Desktop-Version entwickelt

A2. Die Sprache der Benutzeroberfläche ist Deutsch

5. Systementwurf und Implementierung

In diesem Kapitel werden die Vorbereitungsschritte für den Entwicklungsprozess und die Implementierung erläutert. Im Kapitel 4. Zielbestimmung wurde bereits erwähnt, dass diese Arbeit dem Prinzip agiler Entwicklung folgt. Dementsprechend wird das Projekt zuerst in größere Abschnitte nach Themen/Datensätzen eingeteilt. Als Erstes wird der Datensatz "Studierendenzahl nach Bundesland, Nationalität und Geschlecht WS 1998/99 - WS 2016/17" (DS1), dann der Datensatz "Hochschulen nach Gründungsjahr von 1386 bis 2017" (DS2) und als Letztes der Datensatz "Studierende nach Geschlecht, Land des Studienortes und Land des Erwerbs der Hochschulzugangsberechtigung WS 2006/07 - WS 2017/18" (DS3) bearbeitet. Weiterhin wird jeder Abschnitt in kleinere Aufgaben unterteilt, die iterativ bearbeitet werden. Einige Aufgaben, wie die Erstellung eines Wireframes oder die Einrichtung einer Entwicklungsumgebung sind für das ganze Projekt relevant, andere, wie Vorbereitung und Visualisierung von Daten sind für jeden Datensatz spezifisch.

Die in dieser Arbeit verwendeten Datensätze wurden durch die Forschung der oben genannten Datenquellen erhoben. Die meisten Datenportale bieten die Daten als CSV- oder Excel-Tabelle an. Diese Datensätze sind für die Menschen verständlich, aber ihre Datenstruktur ist für die unmittelbare Bearbeitung durch ausgewählte Algorithmen ungeeignet. Darüber hinaus müssen die Daten für die weitere Verwendung auf Vollständigkeit, Einheitlichkeit und Fehler geprüft und danach aufbereitet werden. Die Vorbereitung und die Bereinigung der Daten sowie die Erstellung der ersten Visualisierungs-Prototypen wird in Jupyter Notebooks durchgeführt, was später im entsprechenden Abschnitt ausführlich erläutert wird.

5.1. Wireframe

Bei jedem Projekt ist die Benutzeroberfläche ein wichtiger Bestandteil des Entwicklungsprozesses. Diese soll möglichst intuitiv zu bedienen sein und selbsterklärend und ansprechend gestaltet werden. Gleichzeitig dürfen die visuellen Elemente nicht den Fokus vom Inhalt weg auf sich selbst lenken. Da jede Seite der Webanwendung nach einer Vorlage aufgebaut wird, reicht es aus nur ein Wireframe zu erstellen (siehe Abb. 5.1). Dafür wird das browserbasierte Tool draw.io[42] benutzt.

[42] draw.io ist eine kostenlose Online-Diagrammsoftware für die Erstellung von verschiedenen Diagrammtypen und Grafiken https://www.draw.io/ [11.03.2019]

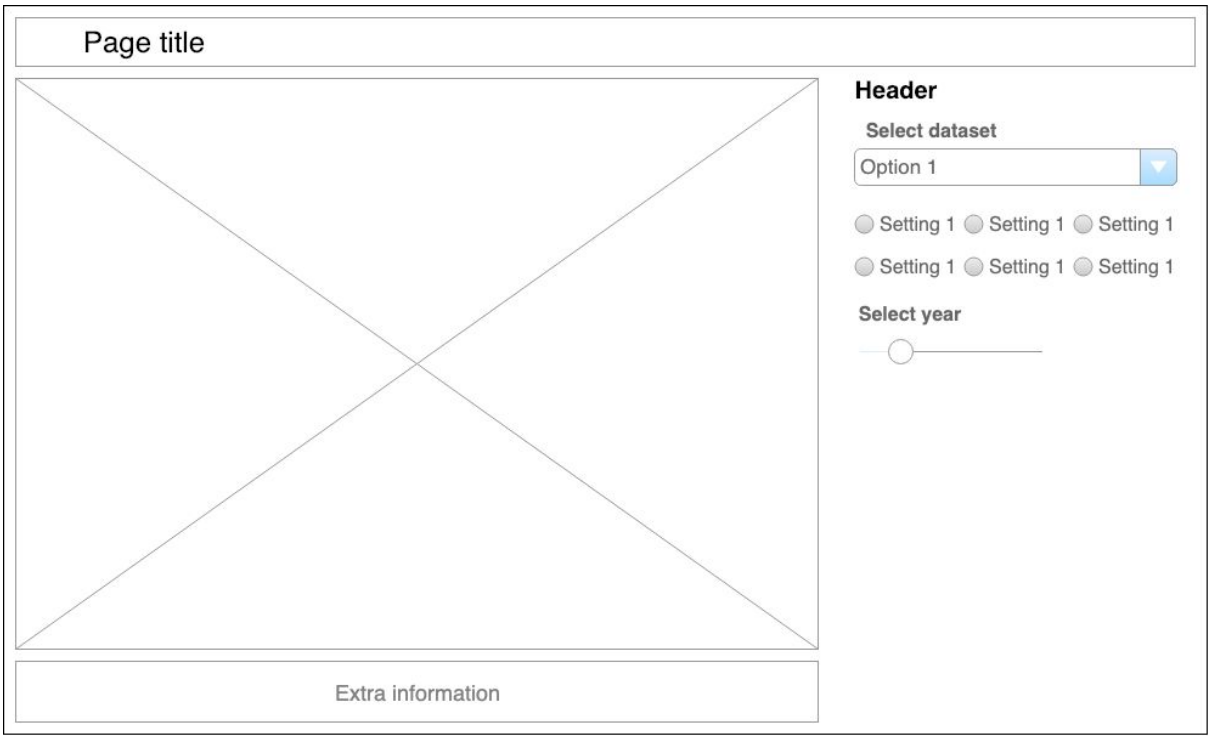

Abb. 5.1: *Wireframe der Benutzeroberfläche*

Im zentralen Bereich der Webseite befindet sich eine Karte, auf der die ausgewählten Daten dargestellt werden. Unter der Karte befindet sich eine Leiste. Mit dem Klick darauf transformiert sie sich in einen ausklappbaren Bereich mit zusätzlicher Information. Die Spalte rechts daneben beinhaltet die Bedienungselemente.

5.2. Entwicklungsumgebung

Für jedes Projekt wird die Entwicklungsumgebung separat eingerichtet, um Konflikte zwischen verschiedenen Versionen von Softwarepaketen und ihren Abhängigkeiten zu vermeiden[43]. Dementsprechend wird für diese Bachelorarbeit ein neues Projekt mit einer neuen Projektumgebung angelegt und die benötigten Bibliotheken installiert. Der Paketmanager conda ermöglicht unter anderem die Erstellung einer neuen Projektumgebung. Dafür sollen im Terminal folgende Befehle ausgeführt werden.

1. Als Erstes wird eine neue Umgebung mit dem Befehl `conda create -n <Umgebungsname>` eingerichtet

```
conda create -n bachelor3.6
```

[43] https://docs.python.org/3/tutorial/venv.html [11.03.2019]

2. Dann wird mit dem Befehl `source activate <Umgebungsname>` die bereits erstellte Umgebung als Standard festgelegt

```
source activate bachelor3.6
```

3. Die Installation von benötigten Paketen und Bibliotheken erfolgt mit dem Befehl `conda install <Paketname>`

```
conda install pandas
```

Nachdem die Projektumgebung vollständig eingerichtet wurde, wird ein neues Projekt in der integrierten Entwicklungsumgebung (IDE) PyCharm erstellt. Die endgültige Anwendung wird aus mehreren Dateien bestehen. So wird eine eindeutige Projektstruktur (siehe Abb. 5.2) angelegt.

```
flaskapp/
|----dataset/
|----static/
|    |----css/
|    |----js/
|----templates/
|----application files
```

Abb. 5.2: *Projektstruktur in der IDE PyCharm*

Es wird ein Hauptordner `flaskapp` für das ganze Projekt eingerichtet. Der Ordner `dataset` ist für die Datensätze vorgesehen. Der Ordner `static` wird CSS-, JavaScript- und Bild-Dateien beinhalten. HTML-Dateien und Jinja2-Vorlagen werden im Ordner `templates` liegen.

5.3. Backend

Das Backend kümmert sich um das Routing, das Rendering der HTML-Seiten, die Vorbereitung von Daten für die Visualisierung und die Kommunikation mit dem Frontend via Ajax-Anfragen. Wie im Kapitel 3 bereits geschrieben wurde, ist die Entwicklung eines Webservers mit dem Framework Flask zu realisieren.

5.3.1. Routing und Ansichten

Laut Anforderungen, die im Kapitel 4 definiert wurden, besteht die Webanwendung aus drei Seiten - eine Seite pro Datensatz, so dass jede Visualisierung eine separate URL hat. Flask ermöglicht durch den Einsatz eines `route()` Decorators, die URL an eine Funktion zu

binden. Mit dem Argument `methods` kann die unterstützte HTTP-Methode für die HTTP-Anfrage definiert werden. Als Standard ist die GET-Methode definiert.

Nachstehend wird dieses Verfahren an einem Beispiel (siehe Listing 5.1) aus dem Quellcode erläutert.

```
@app.route('/map/')
def map(year='1998/99'):
...
return render_template('students-state.html',
                       year='Wintersemester'+' '+year,
                       years=YEARS,
                       gender='',
                       page_title='Studierende nach Bundesländer',
                       ds="st_bd")
```

Listing 5.1: *Anbindung einer Funktion an einen `route()` Decorator*

Vom Browser wird eine GET-Anfrage an die URL `/map/` gesendet. Die Funktion `map()` nimmt diese HTTP-Anfrage an und gibt eine HTTP-Antwort in Form einer HTML-Seite zurück.

Mit `render_template()`Methode erfolgt das Rendering der HTML-Seite. Dafür ist Jinja2 zuständig. Als Eingangsparameter sollen der Name der Vorlage und optionale Variablen in Form einer Schlüssel-Wert-Paar übergeben werden. Die übergebenen Variablen werden beim Rendering der HTML-Seite eingesetzt. Die von Jinja2 erstellte HTML-Seite wird an Flask zurückgegeben und dann von Flask an den Web-Browser weitergeleitet.

In diesem Beispiel (Listing 5.1) ist “students-state.html” der Name der Vorlage. Die anderen Parameter sind die Variablen, die an die Vorlage übergeben werden. Listing 5.2 illustriert den Einsatz von übergebenen Variablen für die Erstellung einer Auswahlliste auf der Webseite. Dieses Konzept wird in dem Abschnitt 5.3.2 detaillierter erklärt.

```
<select>
  {% for k in years %}
     <option value="{{ k }}"
      {{ 'selected' if k == selected_year }} >
      {{ k }}
     </option>
  {% endfor %}
</select>
```

Listing 5.2: *Einsetzen von Variablen in die Vorlage*

Dementsprechend werden die Ansichten für die weiteren Datensätze definiert.

5.3.2. Vorlage

Unter einer Vorlage kann man sich ein leeres Dokumentformular vorstellen. Die Vorlage bietet ein Grundgerüst einer Webseite, die an vordefinierten Stellen mit bestimmten Inhalten befüllt werden kann. Die Nutzung von Vorlagen ermöglicht den Code sauber und übersichtlich zu halten und somit dem DRY[44] Prinzip zu folgen. Flask nutzt eine Jinja2 Template Engine. Eine wichtige Besonderheit von Jinja2 ist die Vererbung von Vorlagen. Da alle Seiten der Webanwendung die gleiche Struktur haben, lohnt es sich eine Seitenvorlage zu erstellen, die als Grundgerüst für die Seiten verwendet wird. Die Struktur der Basisvorlage, von der alle Seiten erben werden, ist in der Abbildung 5.3 vorgestellt.

In dem obigen Beispiel (siehe Listing 5.2) werden die Variablen und die Kontrollstrukturen eingebaut, die mithilfe Jinja2 Template Engine durch die übergebene Werte ersetzt werden. Zu den Kontrollstrukturen gehören if-else Abfragen, for-Schleifen, Makros u.a. Dabei wird folgende Syntax verwendet.

`{{ ... }}` für die Variablen
`{% ... %}` für die Kontrollstrukturen

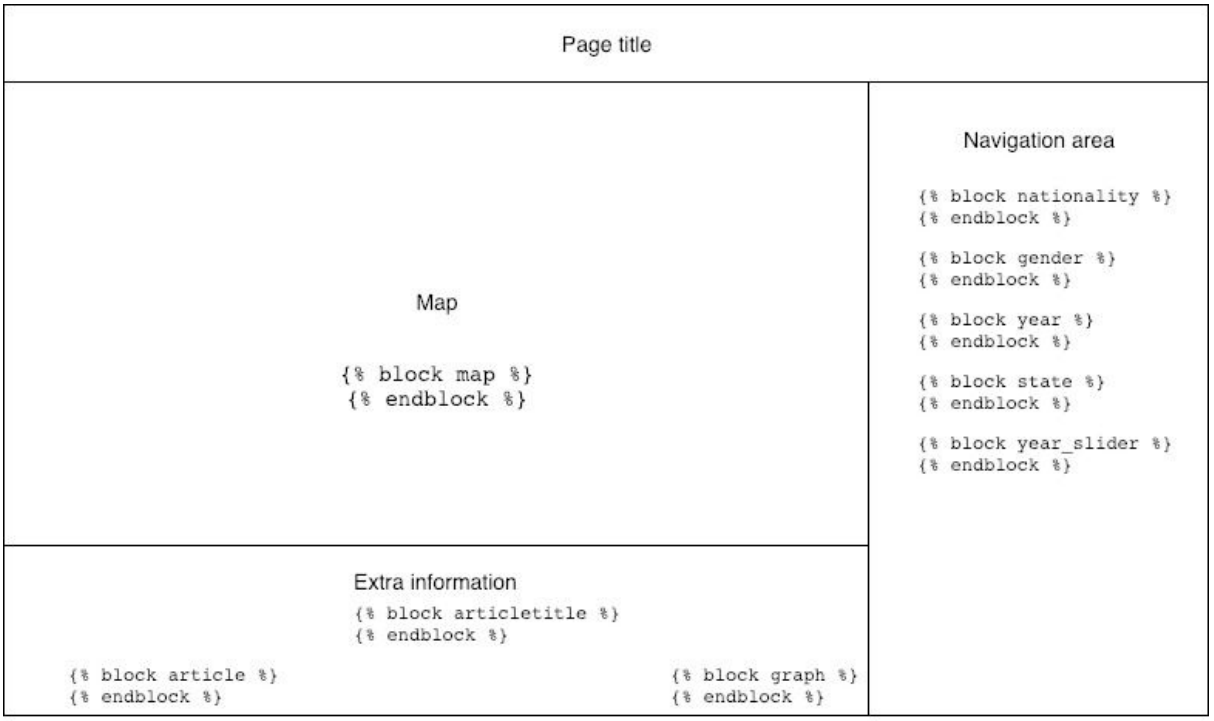

Abb. 5.3: *Die Struktur der Basisvorlage*

Jede Seite der Webanwendung besteht aus gleichen Komponenten: ein Titel oben, eine Karte mit visualisierten Daten in der Mitte, ein Navigationsbereich mit den Bedienungselementen rechts und eine ausklappbare Leiste mit zusätzlicher Information unten. Um die gemeinsamen Komponenten auf allen Seiten beizubehalten, ohne den Code zu

[44] Don't repeat yourself (DRY, englisch für „wiederhole dich nicht")

wiederholen, wird die Vererbung von Vorlagen verwendet. Es wird zuerst eine Basisvorlage mit allen nötigen Komponenten konzipiert. Die Anweisung `{% extends 'base.html' %}` erweitert eine Vorlage um diese Vorlage. Mit der Block-Anweisung

```
{% block <name> %}
{% endblock %}
```

wird die Vererbung von Komponenten organisiert. Jeder Block soll einen eindeutigen Namen haben, damit Jinja2 weiß, welche Blöcke übereinstimmen. Mit der Anweisung `super()` innerhalb eines Blockes kann die Komponente der Basisvorlage in der abgeleiteten Vorlage gerendert werden. Die komplette Vererbungsarchitektur ist auf der Abbildung 5.4 im Anhang dargestellt.

5.4. Visualisierung

Die Python Bibliothek Folium liefert zwei Möglichkeiten eine Choroplethenkarte zu erstellen. Es kann eine statische oder eine dynamische Choroplethenkarte erzeugt werden. Unter "dynamischer Karte" versteht die Autorin die Möglichkeit eine Zeitperiode mit dem Schieberegler auszuwählen, ohne die Karte neu zu laden. Eine dynamische Choroplethenkarte kann mit der Klasse `TimeSliderChoropleth` erstellt werden. Für die statische Karte wird eine Klasse `Choropleth` benutzt. Die Visualisierung des ersten Datensatzes "Studierende (Anzahl) nach Bundesland, Nationalität und Geschlecht WS 1998/99 - WS 2016/17" und des dritten Datensatzes "Studierende nach Geschlecht, Land des Studienortes und Land des Erwerbs der Hochschulzugangsberechtigung WS 2006/07 - WS 2017/18" wird mithilfe der Klasse `Choropleth` erstellt. Der zweite Datensatz "Hochschulen nach Gründungsjahr von 1386 bis 2017" wird anhand der modifizierten Klasse `TimeSliderChoropleth` visualisiert. Für die Erstellung einer Choroplethenkarte werden neben Sachdaten noch die geografischen Daten gebraucht. Außer Choroplethenkarten werden die Daten in tabellarischer Form und mittels Linien- und Balkendiagramm dargestellt.

Der Visualisierungsprozess besteht aus drei Etappen:

1. Datenvorbereitung
2. Erstellung einer statischen Visualisierung
3. Hinzufügen von Interaktionen (Auswahl der Zeitperiode, des Geschlechts und der Nationalität)

Die Vorbereitung aller Datensätze für die weitere Visualisierung erfolgt anhand Jupyter Notebooks. Die ersten zwei Etappen sind für jeden Datensatz spezifisch und werden einzeln behandelt. Die dritte Etappe ist für alle Datensätze gleich und wird deswegen nur einmal vorgestellt.

5.4.1. Datensatz “Studierendenzahl nach Bundesland, Nationalität und Geschlecht WS 1998/99 - WS 2016/17”

Die Implementierung beginnt mit dem Datensatz “Studierendenzahl nach Bundesland, Nationalität und Geschlecht WS 1998/99 - WS 2016/17” (DS1). Dieser Datensatz liegt in der Form einer CSV-Tabelle vor und beinhaltet die Information über die Anzahl von Studenten für jedes Bundesland unterteilt nach Geschlecht, Nationalität und Jahresspanne von 1998 bis 2017. Zur Choroplethenkarte werden Tooltips hinzugefügt, die den Namen des Bundeslandes und die Anzahl von Studierenden zeigen (siehe Abb. 5.5). Der ausklappbare Bereich unten enthält eine Tabelle, die die visualisierten Werte zusammenfasst. Zusätzlich stellt sie die Bevölkerungszahl und die relative Zahl der Studierenden dar.

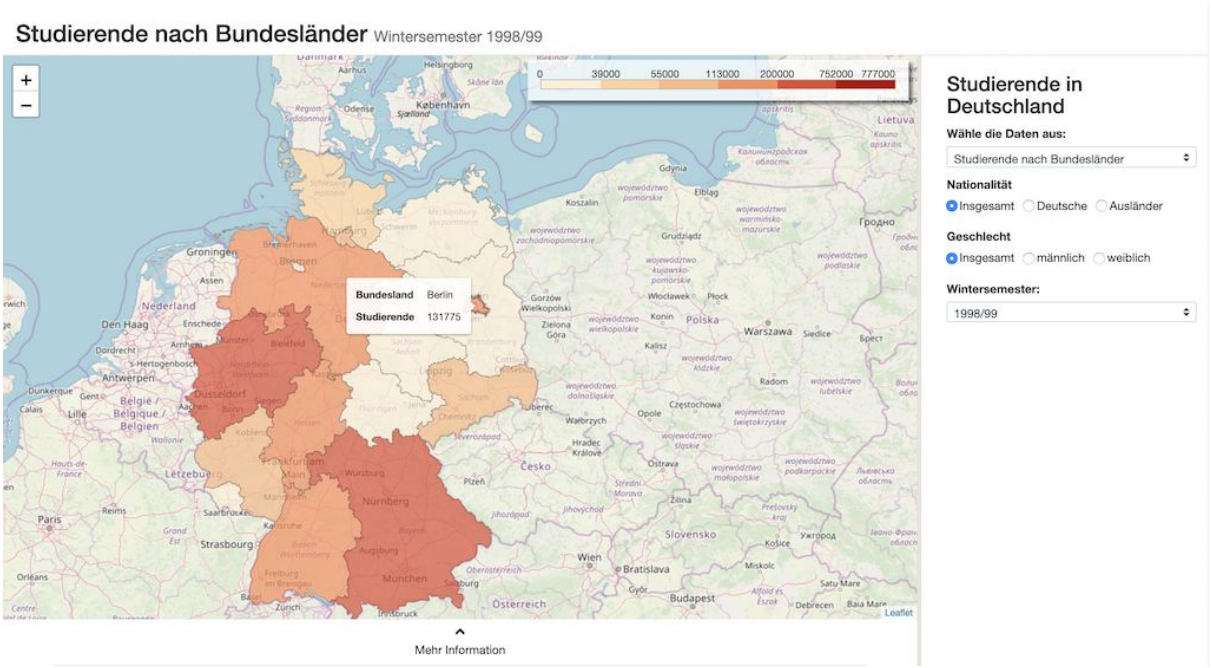

Abb. 5.5: *Visualisierung des Datensatzes DS1 mittels Choroplethenkarte*

Datenvorbereitung

Es werden vier Datensätze für die erste Visualisierung benötigt: der Datensatz DS1 selbst, die Koordinaten der Bundesländer (GDF), die Tabelle mit Tooltips (GDT) und der Datensatz mit der Anzahl der Einwohner in jedem Bundesland (DSB).

Der Datensatz DS1 wird anhand der Pandas Funktion `read_csv()` eingelesen (siehe Abb. 5.6). Es hat sich gezeigt, dass dieser Datensatz 11 Spalten und 313 Zeilen beinhaltet. Man kann sehen, dass darunter auch leere und unnötige Zeilen und Spalten vorhanden sind, die für die weitere Auswertung bereinigt werden sollen.

	GENESIS-Tabelle: 21311-0005	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10
0	Studierende: Bundesländer, Semester, Nationali...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Geschlecht	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Statistik der Studenten	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	Studierende (Anzahl)	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	Deutsche	Deutsche	Deutsche	Ausländer	Ausländer	Ausländer	Insgesamt	Insgesamt	Insgesamt
5	NaN	NaN	männlich	weiblich	Insgesamt	männlich	weiblich	Insgesamt	männlich	weiblich	Insgesamt
6	Baden-Württemberg	WS 1998/99	97587	69435	167022	11624	9866	21490	109211	79301	188512
7	Baden-Württemberg	WS 1999/00	94622	70791	165413	12781	10961	23742	107403	81752	189155
306	Thüringen	WS 2013/14	24041	22603	46644	2732	2625	5357	26773	25228	52001
307	Thüringen	WS 2014/15	23117	21847	44964	3059	2892	5951	26176	24739	50915
308	Thüringen	WS 2015/16	22297	21467	43764	3335	3064	6399	25632	24531	50163
309	Thüringen	WS 2016/17	22379	21530	43909	3338	3273	6611	25717	24803	50520
310	_____	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
311	(C)opyright Statistisches Bundesamt (Destatis)...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
312	Stand: 28.08.2018 - 16:17:22	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

313 rows × 11 columns

Abb. 5.6: *Datensatz "Studierendenzahl nach Bundesland, Nationalität und Geschlecht WS 1998/99 - WS 2016/17" vor der Vorbereitung*

Die Spaltennamen der Tabelle sind unkorrekt, die richtigen Namen stehen in den Zeilen 4 und 5. Die erste und die zweite Spalte haben keine Namen und sollen extra benannt werden. Die ersten vier (0 - 3) und letzten drei (310 -312) Zeilen besitzen keine relevante Information. Die NaN-Werte entsprechen leeren Zellen der CSV-Datei.

Die Methode `info()` aus der Bibliothek Pandas gibt eine Information über die Spaltennamen, die Anzahl nicht leerer Zeilen und den Spaltentyp aus.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 313 entries, 0 to 312
Data columns (total 11 columns):
GENESIS-Tabelle: 21311-0005    311 non-null object
Unnamed: 1                     304 non-null object
Unnamed: 2                     306 non-null object
Unnamed: 3                     306 non-null object
Unnamed: 4                     306 non-null object
Unnamed: 5                     306 non-null object
Unnamed: 6                     306 non-null object
Unnamed: 7                     306 non-null object
Unnamed: 8                     306 non-null object
Unnamed: 9                     306 non-null object
Unnamed: 10                    306 non-null object
dtypes: object(11)
```

Abb. 5.7: *Beschreibung des Datensatzes DS1*

Die Abbildung 5.7 bestätigt die Annahme, dass die Ausgangstabelle ohne Bearbeitung nicht weiter verwendet werden kann. Es gibt folgende Probleme:

- unkorrekte Spaltennamen
- Zeilen mit NaN-Werten
- leere Zeilen
- unkorrekter Datentyp

Für die Erstellung einer Choroplethenkarte werden alle Werte des Datensatzes DS1 gebraucht, deswegen ist es wichtig, die oben genannte Probleme zu beheben. Listing 5.3 stellt den Quellcode der Datenvorbereitung vor.

```
df =
pd.read_csv('data/Studierende_BundesländerSemesterNationalitätGesc
hlecht.csv', sep=';', encoding='latin-1')
# cut all empty rows
df = df.iloc[4:-3]
# rename columns
df.columns.values[0] = 'Bundesland'
df.columns.values[1] = 'Semester'
for i in range(2,11):
    df.columns.values[i] = df.iloc[0,i] + ', ' + df.iloc[1,i]
df = df.reset_index(drop = True)
# cut all empty rows
df = df.iloc[2:]
# convert object type into int type
df[df.columns[2:11]] = df[df.columns[2:11]].astype(int)
# leave in column only numeric values
df.Semester.replace(regex='WS ', value='', inplace=True)
```

Listing 5.3: *Vorbereitung des Datensatzes DS1*

Die Manipulation von Daten erfolgt anhand der Python-Bibliothek Pandas. Im ersten Schritt wird die CSV-Tabelle mit der Funktion `read_csv()` eingelesen. Standardmäßig benutzt diese Funktion ein Komma als Trennzeichen und nutzt das utf-8-Format für die Zeichendekodierung. Wenn die Datensätze andere Trennzeichen verwenden oder in einem anderen Format kodiert sind, müssen sie explizit angegeben werden. Im zweiten Schritt werden die oberen vier Zeilen und die unteren drei Zeilen weggelassen. Im dritten Schritt bekommen die Spalten korrekte Namen. Die ersten zwei Spalten werden nach ihrem Inhalt benannt. Die Spalten 3 bis 11 beinhalten den Namen in den ersten zwei Zeilen. Nachdem alle Spalten korrekt benannt wurden, sollen die Zeilen 1 und 2 gelöscht werden. Wie in der Abbildung 5.7 zu sehen ist, sind alle Werte von Typ object. Im vierten Schritt werden alle Spalten mit numerischen Werten vom Typ object zu einem Integer umgewandelt. Die Werte

in der Spalte “Semester” beinhalten noch die Abkürzung “WS ”. Diese wird nicht gebraucht und wird im letzten Schritt mit der Funktion `replace()` herausgenommen.

Die Tabelle nach der Bereinigung ist in der Abbildung 5.8 dargestellt.

Out[38]:

	Bundesland	Semester	Deutsche, männlich	Deutsche, weiblich	Deutsche, Insgesamt	Ausländer, männlich	Ausländer, weiblich	Ausländer, Insgesamt	Insgesamt, männlich	Insgesamt, weiblich	Insgesamt, Insgesamt
0	Baden-Württemberg	1998/99	97587	69435	167022	11624	9866	21490	109211	79301	188512
1	Baden-Württemberg	1999/00	94622	70791	165413	12781	10961	23742	107403	81752	189155
2	Baden-Württemberg	2000/01	95519	73934	169453	13977	12256	26233	109496	86190	195686
3	Baden-Württemberg	2001/02	97923	79056	176979	15160	14062	29222	113083	93118	206201
4	Baden-Württemberg	2002/03	100344	84873	185217	16785	15782	32567	117129	100655	217784
5	Baden-Württemberg	2003/04	105687	90327	196014	17949	17359	35308	123636	107686	231322
6	Baden-Württemberg	2004/05	109208	94037	203245	18219	18330	36549	127427	112367	239794
7	Baden-Württemberg	2005/06	111530	96707	208237	18093	18288	36381	129623	114995	244618
	...										
295	Thüringen	2008/09	24092	23250	47342	1607	1775	3382	25699	25025	50724
296	Thüringen	2009/10	24795	23978	48773	1803	1946	3749	26598	25924	52522
297	Thüringen	2010/11	25305	24168	49473	2033	2081	4114	27338	26249	53587
298	Thüringen	2011/12	25707	23667	49374	2092	2202	4294	27799	25869	53668
299	Thüringen	2012/13	25085	23358	48443	2398	2390	4788	27483	25748	53231
300	Thüringen	2013/14	24041	22603	46644	2732	2625	5357	26773	25228	52001
301	Thüringen	2014/15	23117	21847	44964	3059	2892	5951	26176	24739	50915
302	Thüringen	2015/16	22297	21467	43764	3335	3064	6399	25632	24531	50163
303	Thüringen	2016/17	22379	21530	43909	3338	3273	6611	25717	24803	50520

304 rows × 11 columns

Abb. 5.8: *Tabelle DS1 nach der Bereinigung*

Die Ausführung der Pandas-Methode `info()` zeigt, dass alle Spalten korrekte Namen haben. Es gibt keine Leerzeilen mehr und die Zahlenwerte haben einen korrekten Typ.

Bei der Manipulation von Daten muss man sehr ordentlich sein. Nachlässigkeit kann dabei zur Datenfälschung führen. Dementsprechend darf man die Wichtigkeit von Tests nicht unterschätzen. Es ist ratsam die Zeilen- und Spaltenzahl in der Tabelle zu überprüfen damit sichergestellt wird, dass bei der Bereinigung keine Zeile oder Spalte verloren gegangen ist. Die Ausgangstabelle beinhaltete die Daten über einen Zeitraum von 19 Jahren für 16 Bundesländer, was 304 Zeilen ergibt und mit der Zeilenanzahl nach der Bereinigung übereinstimmt. Die Spaltenzahl ist unverändert geblieben. Die Werte selbst in der Tabelle wurden nicht manipuliert. So kann man sicher sein, dass keine Werte verfälscht wurden.

Um die Anzahl von Studierenden für jedes Bundesland auf der Deutschlandkarte darzustellen, werden geografische Daten für die Bundesländer im GeoJSON-Format gebraucht. Dieser Datensatz[45] (GDF) enthält die Geokoordinaten, die die Grenzen jedes Bundeslandes beschreiben (siehe Abb. 5.9). Daneben hat jedes Bundesland die zusätzlichen Eigenschaften, wie Id, Name und Stil. Mit der Id oder mit dem Namen können die Geodaten

[45] Quelle: https://github.com/isellsoap/deutschlandGeoJSON/tree/master/2_bundeslaender [23.02.2019]

mit den Sachdaten gemappt werden. Das Attribut `'style'` deklariert wie die Grenzlinien auf der Karte aussehen sollen. Dieser kann bei Bedarf verändert werden (siehe Listing 5.8). Mit dem Attribut `'highlight'` wird das Aussehen der Geometrie bestimmt, wenn ein Benutzer mit der Maus über das Bundesland fährt.

```
{'type': 'FeatureCollection',
 'features': [{'type': 'Feature',
   'id': 0,
   'properties': {'ID_0': 86,
    'ISO': 'DEU',
    'NAME_0': 'Germany',
    'ID_1': 1,
    'NAME_1': 'Baden-Württemberg',
    'NL_NAME_1': None,
    'VARNAME_1': None,
    'TYPE_1': 'Land',
    'ENGTYPE_1': 'State',
    'style': {'weight': 1,
     'opacity': 0.2,
     'color': 'black',
     'fillOpacity': 0.7,
     'fillColor': '#b30000'},
    'highlight': {'weight': 3, 'fillOpacity': 0.8999999999999999}},
   'geometry': {'type': 'MultiPolygon',
    'coordinates': [[[[9.650460243225211, 49.7763404846192],
       [9.656839370727539, 49.761451721191406],
       [9.640399932861612, 49.75014114379883],
       [9.652028083801326, 49.742759704589844],
       [9.646539688110352, 49.738990783691406],
       ...
```

Abb. 5.9: *Auszug aus dem Datensatz GDF*

Laut dem Muss-Kriterium M6 aus dem Kapitel 4, muss die begleitende Information mittels Tooltips angezeigt werden, wenn ein Benutzer mit der Maus über die Choroplethenkarte fährt. Die Tooltips sollen den Namen des Bundeslandes und die Anzahl der Studierenden zeigen. Da die Tooltips einem bestimmten Bereich auf der Karte zugewiesen sind, wird eine weitere Datentabelle für die Erstellung von Tooltips gebraucht. Diese soll neben den Sachdaten noch die Koordinaten der Bundesländer beinhalten. Der Datensatz DS1 hat nur die Information über die Anzahl von Studierenden, die Koordinaten jedes Bundeslandes sind in dem Datensatz GDF gespeichert. Infolgedessen soll der DS1 Datensatz um die fehlende Information bereichert werden. Das heißt, die Tabelle mit Geodaten GDF und die Tabelle mit Sachdaten DS1 sollen zusammengeführt werden. Dabei wird der Name des Bundeslandes als Schlüssel benutzt. Die resultierende Tabelle soll alle Spalten der DS1 Tabelle und drei zusätzliche Spalten “geometry”, “highlight” und “style” aus

dem Datensatz GDF beinhalten. Diese Operation wird mit Jupyter Notebook erledigt. Beide Tabellen werden eingelesen (siehe Listing 5.4). Damit man mit Geodaten weiterarbeiten kann, werden sie aus dem GeoJson-Format zum GeoDataFrame-Format mithilfe der Python-Bibliothek GeoPandas umgewandelt.

```
DS1 =
pd.read_pickle('clean_data/students_bundesland_gender_foreigner_ws
1998_99_ws2016_17.pkl')

with open('geodata/geo_germany.geojson') as data_file:
    state_geo = json.load(data_file)
GDF = geopandas.GeoDataFrame.from_features(state_geo['features'])
```

Listing 5.4: *Einlesen der Datensätze*

Zur Veranschaulichung werden in der Abbildung 5.10 und 5.11 die Tabelle DS1 beziehungsweise die Tabelle GDF gezeigt.

	Bundesland	Semester	Deutsche, männlich	Deutsche, weiblich	Deutsche, Insgesamt	Ausländer, männlich	Ausländer, weiblich	Ausländer, Insgesamt	Insgesamt, männlich	Insgesamt, weiblich	Insgesamt, Insgesamt
0	Baden-Württemberg	1998/99	97587	69435	167022	11624	9866	21490	109211	79301	188512
1	Baden-Württemberg	1999/00	94622	70791	165413	12781	10961	23742	107403	81752	189155
2	Baden-Württemberg	2000/01	95519	73934	169453	13977	12256	26233	109496	86190	195686
3	Baden-Württemberg	2001/02	97923	79056	176979	15160	14062	29222	113083	93118	206201
4	Baden-Württemberg	2002/03	100344	84873	185217	16785	15782	32567	117129	100655	217784
5	Baden-Württemberg	2003/04	105687	90327	196014	17949	17359	35308	123636	107686	231322
6	Baden-Württemberg	2004/05	109208	94037	203245	18219	18330	36549	127427	112367	239794
7	Baden-Württemberg	2005/06	111530	96707	208237	18093	18288	36381	129623	114995	244618

Abb. 5.10: *Tabelle DS1*

	ENGTYPE_1	ID_0	ID_1	ISO	NAME_0	NAME_1	NL_NAME_1	TYPE_1	VARNAME_1	geometry	highlight	style
0	State	86	1	DEU	Germany	Baden-Württemberg	None	Land	None	{'type': 'Polygon', 'coordinates': [[[9.650460243225211, 49.7763404846192,...	{'weight': 3, 'fillOpacity': 0.8999999999999999}	{'weight': 1, 'opacity': 0.2, 'color': 'black'...
1	State	86	2	DEU	Germany	Bayern	None	Land	Bavaria	{'type': 'Polygon', 'coordinates': [[[10.13385963439958, 50.54999923706066,...	{'weight': 3, 'fillOpacity': 0.8999999999999999}	{'weight': 1, 'opacity': 0.2, 'color': 'black'...
2	State	86	3	DEU	Germany	Berlin	None	Land	None	{'type': 'Polygon', 'coordinates': [[[13.16180992126476, 52.59442138671869,...	{'weight': 3, 'fillOpacity': 0.8999999999999999}	{'weight': 1, 'opacity': 0.2, 'color': 'black'...
3	State	86	4	DEU	Germany	Brandenburg	None	Land	None	{'type': 'Polygon', 'coordinates': [[[13.87950801849371, 53.50106811523443,...	{'weight': 3, 'fillOpacity': 0.8999999999999999}	{'weight': 1, 'opacity': 0.2, 'color': 'black'...
4	State	86	5	DEU	Germany	Bremen	None	Land	None	{'type': 'Polygon', 'coordinates': [[[8.98544883728033, 53.12821960449224, ...	{'weight': 3, 'fillOpacity': 0.8999999999999999}	{'weight': 1, 'opacity': 0.2, 'color': 'black'...
5	State	86	6	DEU	Germany	Hamburg	None	Land	None	{'type': 'Polygon', 'coordinates': [[[10.07161712646484, 53.71823120117182,...	{'weight': 3, 'fillOpacity': 0.8999999999999999}	{'weight': 1, 'opacity': 0.2, 'color': 'black'...
6	State	86	7	DEU	Germany	Hessen	None	Land	Hesse	{'type': 'Polygon', 'coordinates': [[[9.498769760131779, 51.63151931762695,...	{'weight': 3, 'fillOpacity': 0.8999999999999999}	{'weight': 1, 'opacity': 0.2, 'color': 'black'...

Abb. 5.11: *Tabelle GDF*

Jede Zeile der Tabelle DS1 soll die entsprechenden Koordinaten aus der GDF Tabelle beinhalten. Mit der Pandas-Funktion `merge()` werden beide Tabellen zusammengeführt (siehe Listing 5.5). Die Parameter `left_on` und `right_on` legen die Schlüssel für die linke und die rechte Tabelle fest. Die resultierende Tabelle GDT hat einen DataFrame Datentyp. Dieser muss zum GeoDataFrame Datentyp umgewandelt und mit den Daten des Koordinatenreferenzsystems (KBS, engl. CRS) bereichert werden, damit die geometrischen Objekte auf der Kartenoberfläche gezeichnet werden können.

```
result = DS1.merge(GDF, left_on='Bundesland', right_on='NAME_1')
GDT = geopandas.GeoDataFrame(result)
GDT.crs = fiona.crs.from_epsg(4326)
```

Listing 5.5: *Datenvorbereitung für die Erstellung von Tooltips*

Wie in der Abbildung 5.12 zu sehen ist, beinhaltet die Tabelle GDT nach der Bearbeitung die Sach- und die Geodaten, die für die Erstellung von Tooltips notwendig sind.

	Bundesland	Semester	Deutsche, männlich	Deutsche, weiblich	Deutsche, Insgesamt	...	geometry	highlight	style
0	Baden-Württemberg	WS_1998_99	97587	69435	167022	...	(POLYGON ((9.650460243225211 49.7763404846192,...	{'weight': 3, 'fillOpacity': 0.8999999999999999}	{'weight': 1, 'opacity': 0.2, 'color': 'black'...
1	Baden-Württemberg	WS_1999_00	94622	70791	165413	...	(POLYGON ((9.650460243225211 49.7763404846192,...	{'weight': 3, 'fillOpacity': 0.8999999999999999}	{'weight': 1, 'opacity': 0.2, 'color': 'black'...
2	Baden-Württemberg	WS_2000_01	95519	73934	169453	...	(POLYGON ((9.650460243225211 49.7763404846192,...	{'weight': 3, 'fillOpacity': 0.8999999999999999}	{'weight': 1, 'opacity': 0.2, 'color': 'black'...
3	Baden-Württemberg	WS_2001_02	97923	79056	176979	...	(POLYGON ((9.650460243225211 49.7763404846192,...	{'weight': 3, 'fillOpacity': 0.8999999999999999}	{'weight': 1, 'opacity': 0.2, 'color': 'black'...

Abb. 5.12: *Tabelle GDT, die für Tooltips benutzt wird*

Um sicher zu gehen, dass diese Tabelle vollständig ist und keine Fehler beinhaltet, wird sie nach folgenden Kriterien untersucht.

- Zeilenanzahl: Die Tabelle GDT soll 304 Zeilen beinhalten. Dies entspricht 16 Bundesländer je 19 Zeitangaben.
- Die minimalen und maximalen Werte jeder numerischen Spalte der GDT Tabelle sollen mit den Werten der DS1 Tabelle übereinstimmen.

- Die Spalte “geometry” muss exakt 16 einzigartige Werte haben. Ein Wert pro Bundesland.

Zusätzlich können beide Tabellen anhand der Pandas-Methode `describe()` verglichen werden. Als Ergebnis erhält man für jede Spalte die Anzahl nicht NaN-Werten, den Mittelwert, die Standardabweichung, den minimalen und maximalen Wert und die Quartile.

Ein weiteren Datensatz, der für die Visualisierung benötigt wird, ist der Datensatz “Einwohner in Deutschland nach Bundesländern”[46] DSB (siehe Abb. 5.23 im Anhang). Der wird nach dem gleichen Prinzip, wie der Datensatz DS1 vorbereitet. Die relativen Werte von Studierenden in jedem Bundesland lassen sich nach der Formel *(1)* berechnen.

$$S_{rel} = \frac{S_{abs} * 100\%}{P} \qquad (1)$$

S_{rel} - Anzahl von Studierenden im Prozentsatz in einem Bundesland,

S_{abs} - Anzahl von Studierenden in absoluten Zahlen in einem Bundesland,

P - Einwohnerzahl in einem Bundesland

Nachdem alle Datensätze bereinigt und auf Korrektheit getestet wurden, können sie für den weiteren Gebrauch gespeichert werden. Die Verwendung des Excel- oder CSV-Formats wird nicht empfohlen, da in diesem Fall die Typisierung von Werten oder die zusätzliche Metainformation verloren gehen kann. Um das zu vermeiden, werden die vorbereiteten Datensätze im pickle-Format gespeichert. Das Pickle[47] Modul ermöglicht die Serialisierung und Deserialisierung von Objekten, so dass die Daten persistent abgespeichert werden können. Die Bibliothek Pandas hat zwei Methoden um die Dateien in pickle-Format einzulesen und zu speichern. Die heißen `read_pickle()` beziehungsweise `to_pickle()`.

Statische Visualisierung

Nachdem die Daten vorbereitet wurden, folgt als nächster Schritt der Visualisierungsprozess. Dazu gehören die Erstellung einer Karte von Deutschland, einer Choroplethenkarte und die Tooltips. Außerdem wird eine Tabelle mit der Anzahl von Studierenden zu dem unteren ausklappbaren Bereich hinzugefügt (siehe Abb. 5.3).
Zuerst wird die Karte von Deutschland, mittels Folium Klasse `Map` erstellt, siehe Listing 5.6.

```
m = folium.Map(location=[52, 13], tiles="Openstreetmap",
zoom_start=6)
```

Listing 5.6: *Deutschlandkarte erstellen*

[46] Quelle: Statistisches Bundesamt, Fortschreibung des Bevölkerungsstandes
https://www.statistik-bw.de/VGRdL/tbls/tab.jsp?rev=RV2014&tbl=tab20&lang=de-DE
[47] [PIC]

Mit dem Attribut `location` wird das Zentrum der Karte übergeben, mit `tiles` wird das Aussehen der Karte definiert. Es gibt eine Liste von Standard-Kartenkacheln (Tilesets), oder man kann eine benutzerdefinierte Kartenkachel hinzufügen. Das Attribut `zoom_start` bestimmt die Anfangszoomstufe der Karte.

Dann wird eine Choroplethenkarte erstellt, die als zusätzliche Ebene auf die Karte hinzugefügt wird, siehe Listing 5.7. Dafür wird die Folium Klasse `Choropleth` benutzt.

```
folium.Choropleth(
        geo_data=GDF,
        name='choropleth',
        data=DS1,
        columns=['Bundesland', 'Insgesamt, Insgesamt'],
        key_on='feature.properties.NAME_1',
        fill_color='OrRd',
        fill_opacity=0.7,
        line_opacity=0.2,
        legend_name=legend,
        bins=bins,
        highlight=True
    ).add_to(m)
```

Listing 5.7: *Choroplethenkarte erstellen*

An diese Klasse sollen zwei Datensätze übergeben werden. Die Parameter `geo_data` und `data` stehen für die Geodaten im GeoJSON-Format beziehungsweise die Sachdaten. In diesem Fall werden die Datensätze GDF und DS1 benutzt. Mit dem Parameter `key_on` wird der Schlüssel übergeben, an den die beiden Datensätze gemappt werden sollen. Der Parameter `columns` bestimmt welche Spalten der DS1-Tabelle genommen werden sollen um die Choroplethenkarte zu erstellen. Dabei muss die erste Spalte als Schlüssel dienen und die zweite als Wert. Andere Parameter sind optional.

Die Choroplethenkarte soll außer farblicher Kodierung der Daten auch die Möglichkeit bieten zusätzliche Information mithilfe von Tooltips zu erkunden. Wenn der Benutzer mit der Maus über die Geometrie des Bundeslandes fährt, wird ihm ein kleines Rechteck mit dem Namen des Bundeslandes und der Anzahl der Studierenden angezeigt (siehe Abb. 5.14).

Abb. 5.14: *Tooltips auf der Choroplethenkarte*

Standardmäßig bietet die `Choropleth`-Klasse keine Möglichkeit Tooltips zur Karte hinzufügen. Diese Option ist in der Folium Klasse `GeoJson` vorhanden, die seinerseits dafür eine Instanz der `GeoJsonTooltip`-Klasse benutzt. Die `GeoJson`-Klasse erstellt aus übergebenen GeoJson-Dateien geometrische Objekte, die auf der Karte angezeigt werden. Laut definierten Anforderungen soll der Tooltip die Information über den Namen des Bundeslandes und die Anzahl der Studierenden beinhalten.

```
folium.GeoJson(data=GDT,
               style_function=lambda x:
                 {'fillColor': '#00000000', 'color': '#00000000'},
               highlight_function=lambda x:
                 {'weight': 3, 'color': 'black'},
               tooltip=folium.features.GeoJsonTooltip(
                    fields=['Bundesland', 'Insgesamt, Insgesamt'],
                    aliases=['Bundesland', 'Studierende'],
                    labels=True,
                    sticky=True))
```

Listing 5.8: *Tooltips erstellen*

Der Codeauszug in Listing 5.8 illustriert die Erstellung von Tooltips. Mit dem Parameter `data` wird ein bereits vorbereiteter Datensatz GDT übergeben. Tooltips sind mit Geometrien auf der Karte verbunden. Das heißt, damit die Tooltips zur Karte hinzugefügt werden können, müssen noch einmal die Geometrien der Bundesländer erstellt werden. Damit diese auf der Karte nicht zu sehen sind, werden die Grenzen mittels Parameter `style_function` unsichtbar gemacht.

Um einen Überblick der Zahlen zu bekommen werden die Daten auch in tabellarischer Form dargestellt (siehe Abb. 5.15).

Bundesland	Bevölkerung	Studierende, abs.	Studierende, rel.
Baden-Württemberg	10,297,400	188,512	1.83%
Bayern	12,013,000	233,116	1.94%
Berlin	3,346,200	131,775	3.94%
Brandenburg	2,565,700	27,531	1.07%
Bremen	667,400	25,978	3.89%
Hamburg	1,672,900	65,141	3.89%
Hessen	5,995,800	148,907	2.48%
Mecklenburg-Vorpommern	1,793,600	23,900	1.33%
Niedersachsen	7,800,600	153,641	1.97%
Nordrhein-Westfalen	17,856,100	515,678	2.89%
Rheinland-Pfalz	4,017,600	80,418	2.00%
Saarland	1,071,100	21,063	1.97%
Sachsen-Anhalt	4,473,700	76,678	1.71%
Sachsen	2,673,900	32,894	1.23%
Schleswig-Holstein	2,745,500	42,061	1.53%
Thüringen	2,455,500	33,358	1.36%

Abb. 5.15: *Tabelle mit absoluten und relativen Werten der Studierendenzahl*

Die Tabelle in der Abbildung 5.15 wird mit HTML und Jinja2-Kontrollstrukturen erstellt. Mit einer for-Schleife werden die Zeilen generiert und gleichzeitig mit entsprechenden Werten befüllt (siehe Listing 5.9).

```
{% for key in table %}
  <tr>
    <th scope="row">{{ key[0]|e }}</th>
    <td id="population">{{ "{:,}".format(key[1])|e }}</td>
    <td id="students_a">{{ "{:,}".format(key[2])|e }}</td>
    <td id="students_r">{{ "{:.2%}".format(key[2]/key[1])|e}}</td>
  </tr>
{% endfor %}
```

Listing 5.9: *Erstellung einer Tabelle mit Jinja2*

Hinzufügen von Interaktionen

Nachdem die statische Visualisierung fertig ist, sollen im nächsten Schritt Interaktionen hinzugefügt werden. Laut der Projektanforderungen, die im Kapitel 4 definiert wurden, muss eine Möglichkeit bestehen die Zeitperiode, die Nationalität und das Geschlecht auszuwählen. Dementsprechend muss die Choroplethenkarte aktualisiert werden. Die Benutzer-Anfragen werden vom System serverseitig bearbeitet. Als Ergebnis wird eine neue Kartenansicht generiert und angezeigt. Ein Sequenzdiagramm in der Abbildung 5.16 illustriert die Reaktion des Systems auf das Benutzerverhalten.

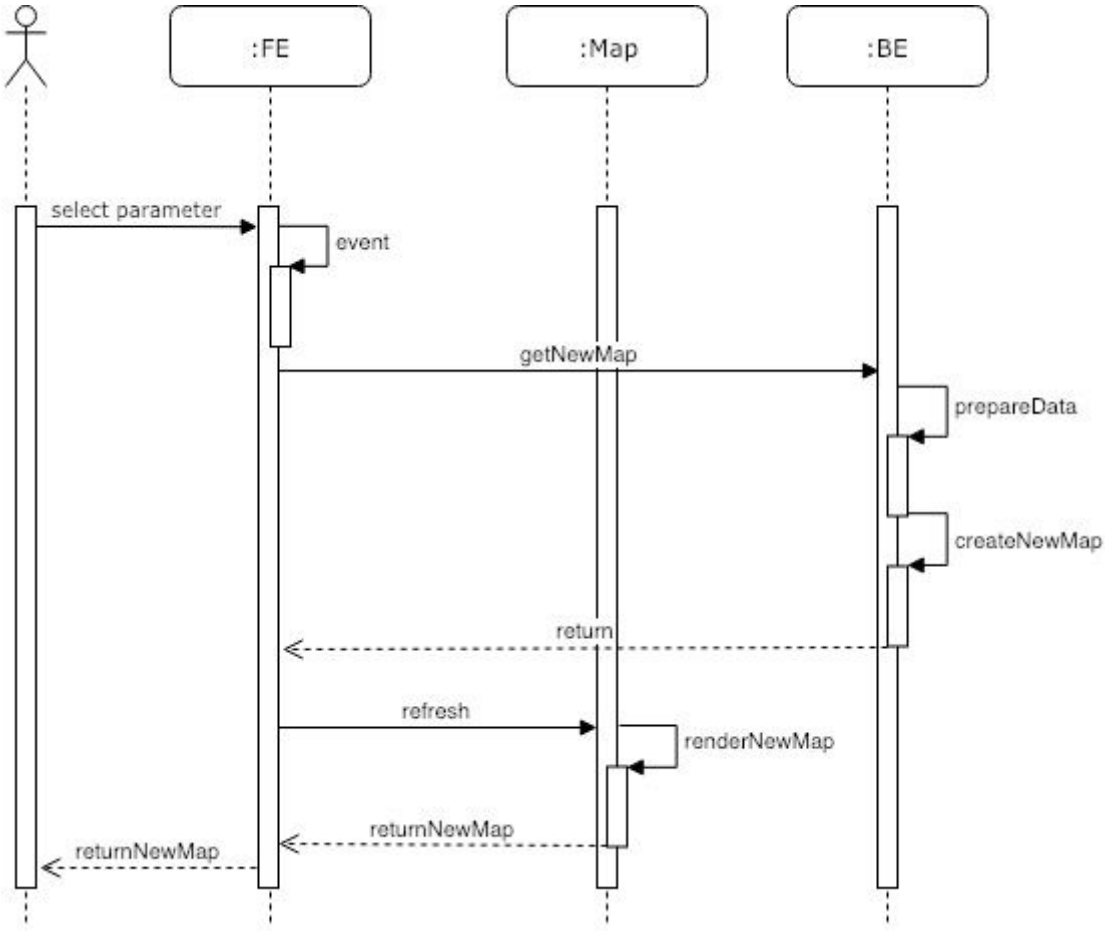

Abb. 5.16: *Sequenzdiagramm Kartenaktualisierung*

Die Auswahl eines Parameters auf der Benutzeroberfläche löst ein Ereignis beim HTML-Element aus. Dieses wird mithilfe von jQuery abgefangen. Dabei wird eine Ajax-Anfrage mit den von dem Benutzer ausgewählten Parametern an das Backend geschickt. Im Backend werden dementsprechend die Daten vorbereitet und eine neue Choroplethenkarte generiert, die als Ajax-Antwort zurück an das Frontend geschickt wird. Die Ajax-Antwort wird im Frontend geparst und eine neue Karte wird gerendert.

Die Daten in tabellarischer Form und die Diagramme werden nach der gleichen Logik aktualisiert.

5.4.2. Datensatz “Hochschulen nach Gründungsjahr von 1386 bis 2017”

Der zweite Datensatz (DS2) ist in der Form einer Excel-Tabelle vorhanden. Der beinhaltet die Informationen über die Hochschulen in Deutschland. Die Hauptidee der Visualisierung dieses Datensatzes ist es, die Gründung der Hochschulen im Zeitverlauf zu zeigen (siehe Abb. 5.17). Durch das Verschieben eines Zeitschiebereglers werden dementsprechend die Hochschulen auf der Karte angezeigt. Mit dem Klick auf den Punkt, der die Hochschule symbolisiert, wird ein Popup mit dem Namen und dem Gründungsjahr der Hochschule angezeigt. Daneben im ausklappbaren Bereich werden die Daten in tabellarischer Form und in der Form eines Liniendiagramms dargestellt (siehe Abb. 5.21).

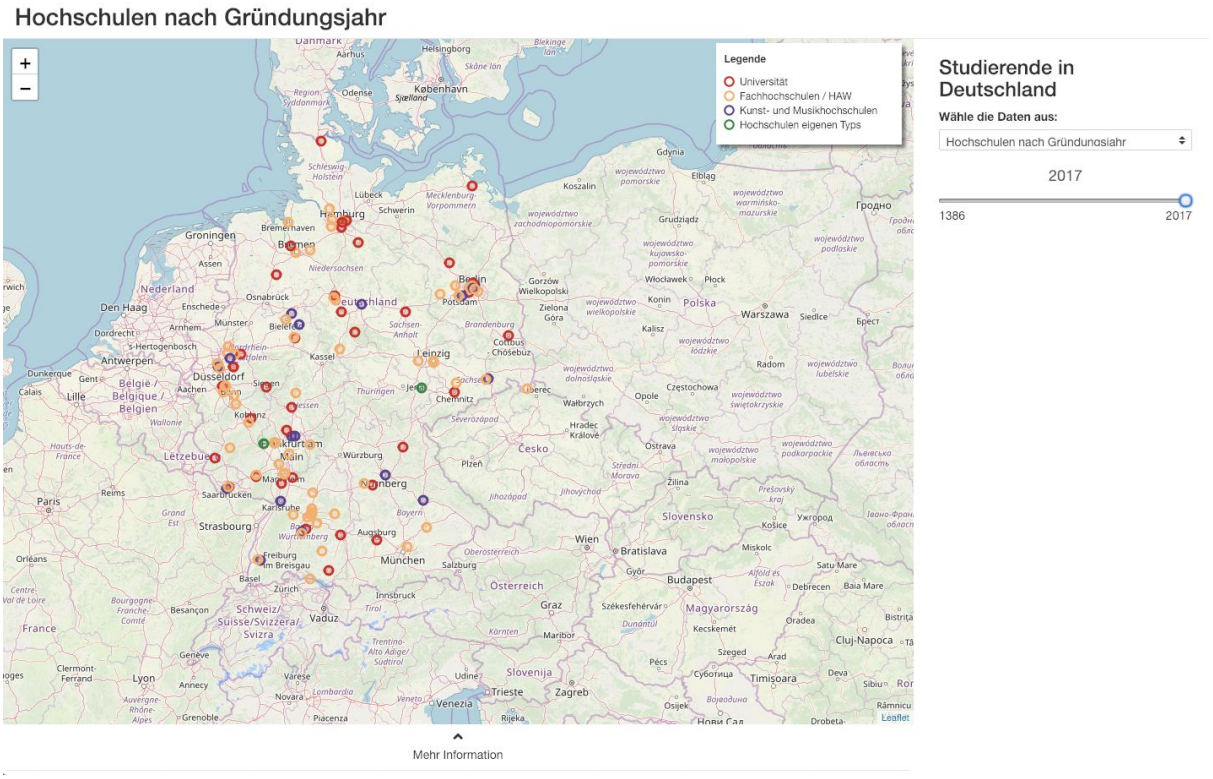

Abb. 5.17: *Visualisierung “Hochschulen nach Gründungsjahr von 1386 bis 2017”*

Datenvorbereitung

Diese Visualisierung fordert zwei Datensätze: die Sachdaten - der Datensatz DS2 und die Geodaten - der Datensatz GDF. Die ursprüngliche Tabelle DS2 ist in der Abbildung 5.13 im Anhang dargestellt. Sie beinhaltet 21 Spalten und 396 Zeilen, wobei einige Zeilen leer sind. Für die Visualisierung sind nicht alle Spalten relevant, wichtig sind Hochschulname, Hochschultyp, Gründungsjahr, Straße, PLZ, Ort. Während der Untersuchung dieser Tabelle wurde festgestellt, dass der Datensatz 393 Hochschulen enthält. Bei 4 Hochschulen fehlen die Werte in der Spalte Gründungsjahr. Diese wurden manuell ergänzt.

```
unis = pd.read_excel('dataset/hs_liste.xlsx')
unis_clean = unis[['Hochschulname', 'Hochschulkurzname',
                   'Hochschultyp', 'Bundesland',
                   'Gründungsjahr', 'Ort', 'PLZ', 'Strasse']]
unis_clean = unis_clean.iloc[:-3]
unis_clean.PLZ = unis_clean.PLZ.astype(int)
unis_clean.Gründungsjahr = unis_clean.Gründungsjahr.astype(int)
```

Listing 5.10: *Bereinigung der Tabelle DS2*

Listing 5.10 illustriert den Vorgang der Bereinigung des Datensatzes DS2. Von 21 Spalten werden nur 7 benötigt. Die unteren drei Zeilen sind leer und werden abgeschnitten. Beim Einlesen der Tabelle wurden die Spalten "PLZ" und "Gründungsjahr" als float-Typ erkannt und werden zum int-Typ umgewandelt. Nach der Bereinigung sieht die Tabelle folgendermaßen aus (siehe Abb. 5.18)

	Hochschulname	Hochschulkurzname	Hochschultyp	Bundesland	Gründungsjahr	Ort	PLZ	Strasse
0	Fachhochschule Aachen	Aachen FH	Fachhochschulen / HAW	Nordrhein-Westfalen	1971	Aachen	52066	Bayernallee 11
1	Rheinisch-Westfälische Technische Hochschule A...	Aachen TH	Universitäten	Nordrhein-Westfalen	1870	Aachen	52062	Templergraben 55
2	Hochschule Aalen - Technik und Wirtschaft	Aalen H	Fachhochschulen / HAW	Baden-Württemberg	1962	Aalen	73430	Beethovenstrafle 1
3	Hochschule Albstadt-Sigmaringen	Albstadt-Sigmaringen H	Fachhochschulen / HAW	Baden-Württemberg	1971	Sigmaringen	72488	Anton-Günther-Strafle 51
4	Alanus Hochschule	Alfter HfK	Kunst- und Musikhochschulen	Nordrhein-Westfalen	1973	Alfter	53347	Villestrafle 3
5	Ostbayerische Technische Hochschule Amberg-Weiden	Amberg-Weiden OTH	Fachhochschulen / HAW	Bayern	1994	Amberg	92224	Kaiser-Wilhelm-Ring 23
6	Hochschule Anhalt - Anhalt University of Appli...	Anhalt H	Fachhochschulen / HAW	Sachsen-Anhalt	1991	Köthen	6366	Bernburger Strafle 55

Abb. 5.18: *Tabelle DS2 nach der Bereinigung*

Die Hochschulen sollen auf der Karte mit Punkten dargestellt werden. Ein Punkt auf der Karte entspricht dem Standort der Hochschule, außerdem besitzt er die Information über das Gründungsjahr, den Hochschultyp und den Hochschulnamen. In dem Datensatz gibt es die Angabe über die Zuordnung der Hochschule zum Bundesland und die Adresse aber keine Geokoordinaten. Um diese fehlende Information zu gewinnen, wird der Vorgang der Geokodierung angewendet (siehe Listing 5.11). Mit der Python-Bibliothek Geocoder[48] und HERE API[49] können die Adressen in Geokoordinaten umgewandelt werden.

[48] [HER]
[49] https://developer.here.com/documentation/geocoder/topics/what-is.html [04.03.2019]

```
lat = []
lon = []
for street, code, city in zip(unis.Strasse, unis.PLZ, unis.Ort):
    address = street + ', ' + str(code) + ', ' + city
    g = geocoder.here(address,
                      app_id='<app_id>',
                      app_code='<app_code>')
    lat.append(g.lat)
    lon.append(g.lng)
```

Listing 5.11: *Geokodierung*

Der Visualisierung dieses Datensatzes zugrunde liegt eine Klasse `TimeSliderChoropleth` aus der Python-Bibliothek Folium, die aber nicht allen Anforderungen entspricht und aus diesem Grund dementsprechend angepasst wird. Diese Klasse fordert die Daten im dictionary-Format (siehe Abb. 5.19), wobei als Schlüssel eine Zeitangabe dient und als Wert eine Liste von Wörterbüchern mit den Daten, die dieser Zeitangabe entsprechen.

```
[{1971: [{'lat': 50.75551,
          'lon': 6.09605,
          'name': 'Fachhochschule Aachen',
          'typ': 'Fachhochschulen / HAW'},
 ...
```

Abb. 5.19: *Datenstruktur für die Visualisierung des Datensatzes DS2*

Die zusätzliche Tabelle “Anzahl der Hochschulen nach Gründungsjahr” im ausklappbaren Bereich besteht aus drei Spalten: Hochschultyp, Anzahl der Hochschulen im ausgewählten Jahr und Gesamtanzahl der Hochschulen zu diesem ausgewählten Jahr. Um diese Tabelle zu befüllen, sollen die Daten in der entsprechenden Form vorbereitet werden (siehe Listing 5.12). Aus dem Datensatz DS2 werden drei Spalten benötigt. Der Name der Hochschule ist für diese Tabelle irrelevant. Die Werte der Spalte “Hochschulname” werden von qualitativ in quantitativ umgewandelt, damit sie summiert werden können. Da die Hochschulen in der Tabelle nach dem Hochschultyp unterteilt sind, wird eine entsprechende Pivot-Tabelle erstellt. Die leeren Zeilen der Pivot-Tabelle werden mit Nullen befüllt. Damit man weiter im Code auf die Spaltennamen verweisen kann, werden diese umbenannt und die Tabelle selbst ins dictionary-Format umgewandelt.

```
df = DS2[['Hochschulname', 'Hochschultyp', 'Gründungsjahr']]
df.Hochschulname = 1
table = pd.pivot_table(df, values='Hochschulname',
                       index='Gründungsjahr',
                        columns=['Hochschultyp'], aggfunc=np.sum)
table = table.fillna(0)
table = table.rename(columns={"Fachhochschulen / HAW": "hs",
                              "Hochschulen eigenen Typs": "other",
                              "Kunst- und Musikhochschulen":"kmh",
                              "Universitäten": "uni"})
table = table.astype(int)
table.index = table.index.astype(int)
dict = table.to_dict('index')
```

Listing 5.12: *Datenvorbereitung für die Tabelle "Anzahl der Hochschulen nach Gründungsjahr"*

Statische Visualisierung

Die Bibliothek Folium bietet keine fertige Lösung für die Umsetzung der Visualisierungsidee für den Datensatz DS2. Über die naheliegende Funktionalität verfügt die Klasse `TimeSliderChoropleth`, deren Rückgabe in der Abbildung 5.20 zu sehen ist. Basierend auf der Funktionalität dieser Klasse wird eine neue Klasse entwickelt, die es ermöglicht statt zur Choroplethenkarte die Punkte (Marker) zur Basiskarte hinzufügen.

Abb. 5.20: *Schematische Abbildung einer Choroplethenkarte mit dem Schieberegler aus der Klasse TimeSliderChoropleth*

Um ein Objekt der Klasse `TimeSliderChoropleth` zu erstellen, sollen zwei erforderliche Eingangsparameter `data` und `styledict` übergeben werden. Mit dem ersten Parameter werden die Geokoordinaten im GeoJSON-Format übergeben. Mit dem zweiten Parameter werden die Stile im Format eines Wörterbuches übergeben. Als Schlüssel dienen feature ids aus dem Datensatz mit den Geokoordinaten (siehe Abb. 5.9) und als Werte eine Liste von Wörterbüchern im Format {Zeit : Stil}. Wobei die Zeit standardmäßig in Nanosekunden Format benutzt wird.

Um das gewünschte Resultat zu erzielen werden folgende Änderungen an der Klasse `TimeSliderChoropleth` vorgenommen. Erstens, die Position des Schiebereglers wird geändert. Zweitens, die Zeit wird nicht mehr in Nanosekunden-Format übergeben. Drittens, die Choroplethenkarte soll durch Punkte (Markers) ersetzt werden.

Die Untersuchung des Quellcodes hat ergeben, dass der Schieberegler standardmäßig zum HTML body-Element hinzugefügt wird. Daraus folgt, dass seine Position durch die Übergabe einen anderen HTML-Tag geändert werden kann.

Die zweite Einschränkung liefert ein weiteres Hindernis, da die Werte der Zeitspalte "Gründungsjahr" des Datensatzes DS2 vom int-Typ sind. Die Konvertierung nach datetime-Typ, der Nanosekunden Format unterstützt, ist wegen spezifischen Einschränkungen der Python-Bibliothek Pandas nicht für alle Daten möglich. Laut offizieller Dokumentation[50] verwendet Pandas 64 Bit um die Zeitmarken in Nanosekundenauflösung zu speichern, so ist die Zeitspanne auf circa 584 Jahre begrenzt (siehe Listing 5.13).

```
In [90]: pd.Timestamp.min
Out[90]: Timestamp('1677-09-21 00:12:43.145225')
In [91]: pd.Timestamp.max
Out[91]: Timestamp('2262-04-11 23:47:16.854775807')
```

Listing. 5.13: *Limit der Zeitspanne der Python-Bibliothek Pandas*

Diese Begrenzung ist kritisch, da der minimale Wert in der Spalte "Gründungsjahr" in der Tabelle DS2 - 1386 ist und deshalb nicht in Nanosekunden-Format umgewandelt werden kann. Im Quellcode der Klasse `TimeSliderChoropleth` wird die Zeitvariable aus dem Nanosekunden-Format in das String-Format umgewandelt. Da die Werte in der Spalte "Gründungsjahr" vom Typ int sind, kann diese Typumwandlung nicht mehr angewendet werden. Um dieses Problem zu lösen wird der Quellcode der Klasse `TimeSliderChoropleth` modifiziert. Die Umwandlung der Zeitvariable aus Nanosekunden-Format wird weggelassen (siehe Listing 5.14).

[50] http://pandas-docs.github.io/pandas-docs-travis/user_guide/timeseries.html#timestamp-limitations [22.01.2019]

```
// vor der Änderung
var datestring = new Date(parseInt(current_timestamp)*1000)
                         .toDateString();
// nach der Änderung
var datestring = current_timestamp.toString();
```

Listing 5.14: *Typumwandlung der Zeitvariable*

Für die dritte Änderung wird aus dem Quellcode der Klasse `TimeSliderChoropleth` ein Teil des Templates, der für die Erstellung der Choroplethenkarte zuständig ist, komplett weggenommen. An dessen Stelle werden zwei neue Funktionen geschrieben. Eine für das Erzeugen und Hinzufügen von Markern und Tooltips zur Karte. Andere, um die erzeugten Elemente zu löschen.

Die Klasse `TimeSliderChoropleth` besitzt standardmäßig keine Legende. So wird die Legende mit HTML erstellt und in das Template eingefügt. In der Legende verwendete Symbole (siehe Abb. 5.17) wurden mit der Font Awesome[51] Bibliothek erstellt. Jede Seite, die Font Awesome Symbole verwendet, soll im `head` Bereich ein Link auf die zugehörige CSS-Datei beinhalten.

Die Entstehung der Hochschulen wird zusätzlich in tabellarischer Form und mit dem Liniendiagramm dargestellt (siehe Abb. 5.21).

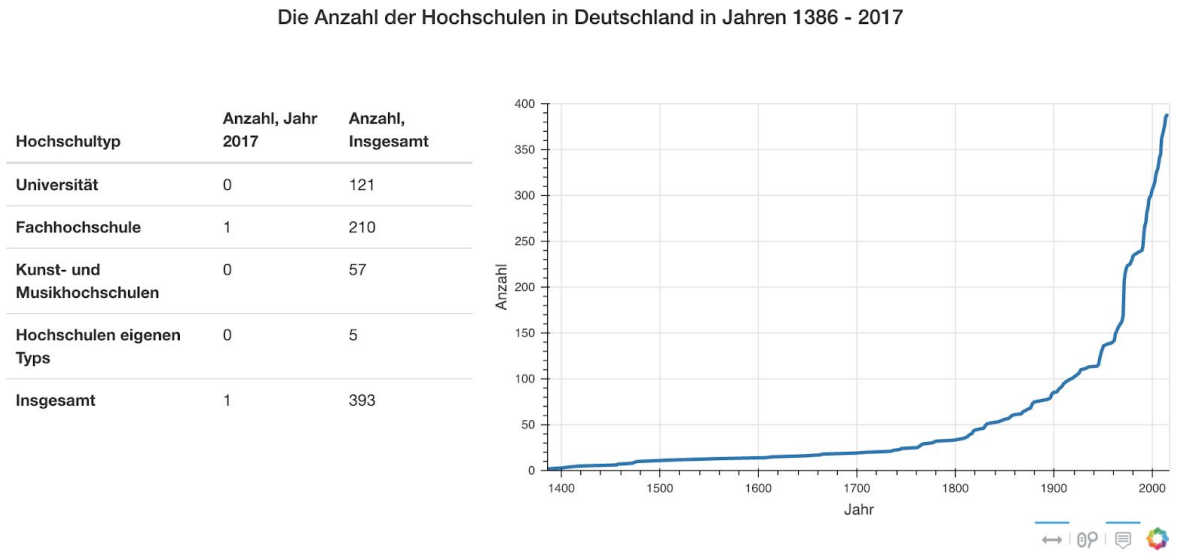

Abb. 5.21: *Tabelle und Liniendiagramm: Anzahl der Hochschulen nach Gründungsjahr*

Die Tabelle, wie oben beschrieben wurde, wird mit HTML und Jinja2-Kontrollstrukturen aufgebaut. Das Liniendiagramm wird mit der Python-Bibliothek Bokeh erstellt (siehe Listing 5.15) und dann in die Seite integriert. Für das Liniendiagramm

[51] https://fontawesome.com [31.01.2019]

sollen drei Komponenten definiert werden: Datenquelle (engl. source), eine Form (engl. figure) und eine Linie (engl. line). Die Form legt grundlegende Parameter der Grafik fest, wie die Größe, den Name u.v.a. Die Linie bestimmt die Werte für x- und y-Koordinaten, die Linienstärke u.v.a.

```
source = ColumnDataSource(data=dict(year=x, quantity=y),
                          name='students')
plot = figure(plot_height=400, plot_width=600,
              toolbar_location='below',
              tools='hover, xwheel_zoom, xpan',
              tooltips=TOOLTIPS,
              x_range=(1386, 2017),
              y_range=(0, 400))
plot.line(x='year', y='quantity', source=source, line_width=3)
script, div = components(plot)
```

Listing 5.15: *Codeauszug: das Erstellen des Liniendiagramms*

Um dieses Diagramm in die Seite einzubetten, wird die `components()`-Funktion der Bokeh-Bibliothek benutzt. Diese Funktion gibt zwei Komponenten zurück. Ein Script, das die Daten für das Diagramm enthält und ein -Tag, in welches die Diagrammansicht geladen wird. Beide Komponenten werden an die Vorlage als Parametern übergeben (siehe Listing 5.16).

```
@app.route('/university-foundation-year/')
def timemap(year=1386):
    ...
    return render_template('uni-year.html',
                           page_title='Hochschulen nach Gründungsjahr',
                           ds="university-foundation-year",
                           script=script, div=div)
```

Listing 5.16: *Funktion für Rendering der Webseite*

5.4.3. Datensatz “Studierende nach Geschlecht, Land des Studienortes und Land des Erwerbs der Hochschulzugangsberechtigung WS 2006/07 - WS 2017/18”

Der Datensatz “Studierende nach Geschlecht, Land des Studienortes und Land des Erwerbs der Hochschulzugangsberechtigung WS 2006/07 - WS 2017/18”, weiter als DS3 genannt, beinhaltet die Information der Studentenmigration innerhalb Deutschlands. Der

Visualisierung zugrunde liegt eine Choroplethenkarte. Die Linien über die Karte symbolisieren die Migration aus dem Land des Erwerbs der Hochschulzugangsberechtigung nach das Land des Studienortes. Mit den Tooltips werden der Name des Studienortes und die Studierendenzahl angezeigt (siehe Abb. 5.22). Das Balkendiagramm in dem ausklappbaren Bereich repräsentiert die prozentuelle Verteilung der Studierenden nach Studienorten.

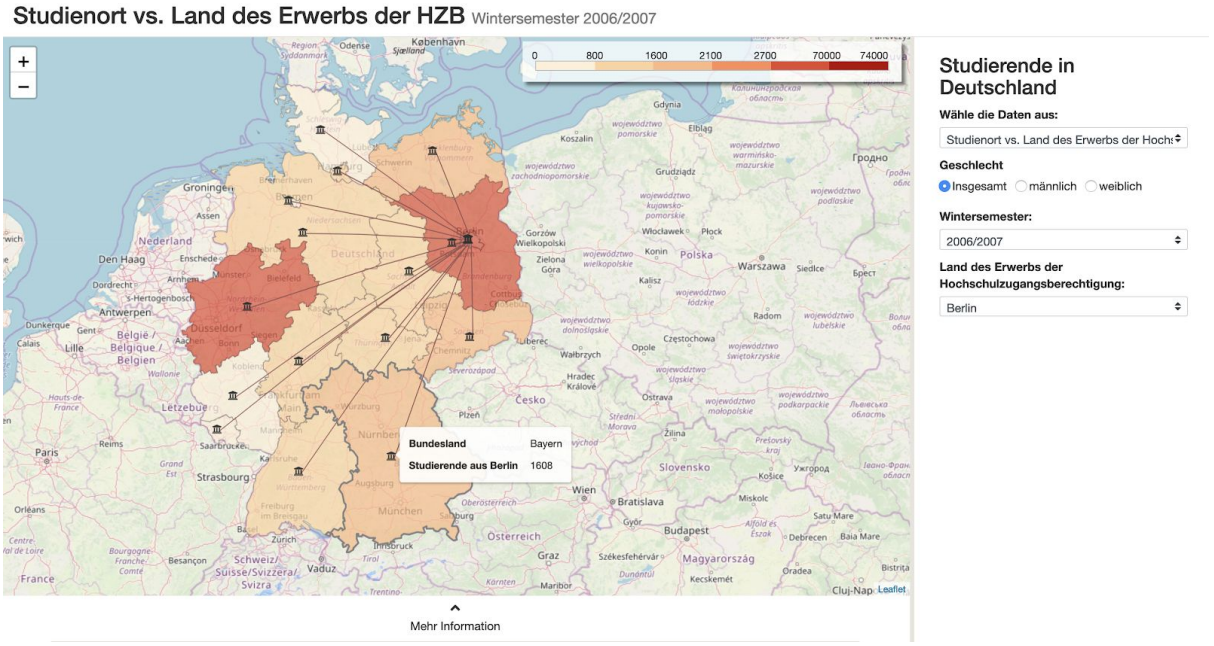

Abb. 5.22: *Visualisierung des Datensatzes DS3*

Datenvorbereitung

Für die Visualisierung des Datensatzes DS3 wird der Datensatz selbst, der Datensatz mit den Koordinaten der Bundesländer (GDF) und die Tabelle mit den Daten für die Tooltips benötigt.

Der Datensatz DS3 liegt als Excel-Tabelle vor (siehe Abb. 5.23 im Anhang). Die Ausgangstabelle besteht aus 22 Spalten und 629 Zeilen. Die Spalten dieser Tabelle beinhalten die Information, wann und in welchem Bundesland Studierende die Hochschulzugangsberechtigung bekommen haben, die Zeilen sagen, in welchem Bundesland sie studieren. Im Studienort werden die Studierenden zusätzlich nach Geschlecht unterteilt. Wie in der Abbildung 5.23 zu sehen ist, ist die Ausgangstabelle für die weitere Verwendung ohne Bereinigung nicht geeignet.

Bei diesem Datensatz handelt es sich um folgende Probleme:

- unkorrekte Spaltennamen
- komplett leere Zeilen
- fehlende Werte in den Spalten
- unkorrekter Datentyp, was mittels Pandas-Funktion `info()` festgestellt wurde

```
# rename columns
df.rename(columns={ df.columns[0]: "WS",
                    df.columns[1]: "Bundesland_Studienort",
                    df.columns[2]: "Geschlecht" }, inplace=True)
for i in range(3,22):
    df.columns.values[i] = df.iloc[4,i]
df.rename(columns={"Nieder-\nsachsen": "Niedersachsen"},
          inplace=True)
# cut out first six empty rows
df = df.iloc[6:-14]
# fill values forward in first and second columns
df[['WS','Bundesland_Studienort']] =
          df[['WS','Bundesland_Studienort']].ffill()
# change type to int
df[df.columns[3:-2]] = df[df.columns[3:-2]].astype(int)
```

Listing 5.17: *Bereinigung der Tabelle DS3*

Der Code im Listing 5.17 demonstriert den Vorgang der Vorbereitung des Datensatzes DS3. Zuerst werden die Spalten korrekt benannt. Dabei wurde festgestellt, dass ein Spaltenname fehlerhaft ist und wurde korrigiert. Dann werden leere Zeilen weggelassen. Danach werden fehlende Werte in den Spalten “WS” und “Bundesland_Studienort” ersetzt. Als letztes erfolgt die Typumwandlung. Die Abbildung 5.24 im Anhang stellt die Tabelle DS3 nach der Bereinigung dar.

Der Datensatz GDF braucht keine Bearbeitung. Der Vorgang der Datenvorbereitung für die Tooltips (DS3_T) ist analog mit der Vorbereitung des Datensatzes GDT.

Statische Visualisierung

Für die Visualisierung des Datensatzes DS3 werden zwei Darstellungsverfahren zusammengesetzt: die Choroplethenkarte und die Verbindungskarte. Die Verbindungskarte entsteht aus Punkten, die mit geraden oder gebogenen Linien verbunden sind. Die Verbindungskarte eignet sich gut, um die Beziehungen zwischen verschiedenen Objekten/Punkten darzustellen. In diesem Fall wird dieses Darstellungsverfahren eingesetzt, um die Migration der Studierenden innerhalb Deutschlands anzuzeigen. Wenn man sich für die Visualisierung mit gebogenen Linien entscheidet, dann wird meistens ein Großkreisprinzip angewendet. Die Linien, die nach diesem Prinzip gezeichnet werden, bilden die kürzeste Verbindung zwischen zwei Punkten auf einer Kugeloberfläche. Die JavaScript Bibliothek Leaflet.js hat ein Plugin Leaflet.Arc[52] für die Erstellung solchen Linien. Dieses Plugin ist in Folium nicht vorhanden und wird im Rahmen dieser Bachelorarbeit in Python implementiert.

[52] [ARC]

Der Visualisierungsprozess beginnt mit der Erstellung einer Basiskarte. Dann wird eine Choroplethenkarte hinzugefügt. Danach werden die Verbindungen zwischen dem ausgewählten Bundesland und allen anderen Bundesländern erstellt und zur Karte hinzugefügt. Als letztes wird die Karte mit Tooltips ergänzt.

Die Erstellung einer Basis- und einer Choroplethenkarte, sowie der Tooltips ist völlig analog der oben vorgestellten Verfahren. Um die Verbindungslinien zwischen Bundesländern nach dem Großkreisprinzip zu zeichnen, wird eine neue Folium-Klasse entwickelt. Wie schon erwähnt wurde, benutzt Folium das Jinja2-Template-Engine für das Rendering von HTML-Seiten. Jinja2 erlaubt die Erstellung von HTML-Seiten anhand Funktionen. Diese Funktionen werden in Jinja2 Makros genannt. Ein großer Vorteil von Makros ist die Möglichkeit innerhalb der Python Klasse HTML mit darin enthaltenem JavaScript zu erstellen.

```
{% macro script(this, kwargs) %}
// Berechnung eines Ausgangspunktes
   try {
       var from_poly = L.polygon({{this.location[0]}})
                        .addTo({{this._parent.get_name()}});
       var from = from_poly.getBounds().getCenter();
       L.marker([from.lng, from.lat])
        .addTo({{this._parent.get_name()}});
   }
   catch(err) {
       console.log('location[0] is not a point', err);
       var from = {{this.location[0]}};
   }
...
// Erstellung einer Verbindungslinie
   {{this.get_name()}} = L.polyline(
                           L.Polyline.Arc(
                             [from.lng, from.lat],
                             [to.lng, to.lat]
                           )._latlngs, {{ this.options }})
                         .addTo({{this._parent.get_name()}})
                         .snakeIn();
{% endmacro %}
```

Listing 5.18: *Makros: die Erstellung einer Verbindungslinie nach dem Großkreisprinzip*

Für die Verbindungslinie werden die Koordinaten des Ausgangs- und des Zielpunktes benötigt. Der Mittelpunkt jedes Bundeslandes mit seinen Koordinaten kann dazu dienen. Der Mittelpunkt kann anhand Bounding Box berechnet werden. Dann wird eine Verbindungslinie

zwischen diesen Punkten gezeichnet. Die JavaScript-Bibliothek Leaflet.js stellt dafür die Methode `L.Polyline.Arc()` zur Verfügung. Zur Anschaulichkeit werden die Verbindungslinien anhand der Funktion `snakeIn()` des Leaflet.Polyline.SnakeAnim[53] Plugins animiert. Ein Codeauszug dafür ist im Listing 5.18 vorgestellt.

5.5. Frontend

Die Frontend-Entwicklung erfolgt anhand der JavaScript Bibliothek jQuery und dem Frontend-Framework Bootstrap. jQuery vereinfacht die Manipulation von HTML-Elementen und die Verarbeitung von Benutzereingaben. Bootstrap ermöglicht die Erstellung von responsiven Webseiten und bringt eine Reihe von UI-Komponenten mit.

Es gibt zwei Möglichkeiten für die Einbindung von jQuery und Bootstrap in das Projekt:

1. Die Quelldateien können heruntergeladen und lokal verwendet werden.
2. Man kann das Content Delivery Network (CDN) benutzen. In diesem Fall sollen die Links für CSS- und JavaScript-Dateien im head-Bereich jeder HTML-Seite eingefügt werden.

In dieser Arbeit wird die zweite Variante der Einbindung verwendet.

Die Benutzeroberfläche wird anhand Bootstrap erstellt. Obwohl mit Bootstrap die Entwicklung von responsiven Webseiten möglich ist, wird im Rahmen dieser Bachelorarbeit nur die Desktop-Version implementiert. Die Unterstützung von mobilen Geräten fordert die spezielle Anordnung von Elementen auf der Seite und die Bearbeitung von mobile spezifischen Gesten[54]. Die Implementierung dieser zusätzlichen Aufgaben braucht mehr Zeit, als im Rahmen der Bachelorarbeit vorhanden ist.

Bootstrap zugrunde liegt ein responsives Gittersystem. Das Gittersystem besteht aus Containern, Reihen und Spalten. Die Elemente der Webseite müssen in Spalten angeordnet werden. Die Zeilen können nur die Spalten beinhalten. Die Container ermöglichen das Layout der Elemente zu bestimmen. Damit ein Container die ganze Breite der Displaygröße annimmt, wird ihm eine Bootstrap-Klasse `.container-fluid` zugewiesen. Ein Gitter Layout besteht aus maximal 12 Spalten. Die Bootstrap-Klassen `.col` und `.row` definieren eine Spalte beziehungsweise eine Reihe.

Die Basisvorlage `base.html`, die im Abschnitt 5.3.2. vorgestellt wurde, wird mit Bootstrap folgendermaßen aufgebaut (siehe Abb. 5.26).

[53] [SNA]
[54] [WIK]

Abb. 5.26: *Die Bootstrap Gitter Layout der Basisvorlage base.html*

Alle Seitenelemente werden in einem Container platziert. Dieser beinhaltet zwei Reihen. Dafür werden zwei div-Elemente mit der Klasse `.row` erstellt. die erste Reihe ist für die Seitenüberschrift vorgesehen, die 100% der Breite des Containers einnimmt. Die zweite Reihe besteht aus zwei Spalten und ist für alle anderen Elemente der Seite vorgesehen. Um eine Spalte zu erstellen, wird eine Klasse `.col-x` benutzt. Die erste Spalte, die für die Karte vorgesehen ist, beinhaltet außerdem noch eine Leiste mit dem ausklappbaren Bereich. Um die Karte und die Leiste untereinander anzuordnen, werden zwei Zeilen zur Spalte hinzugefügt. An sich besteht die Leiste aus zwei weiteren Elementen, jedes davon wird in eine eigene Zeile platziert. Die zweite Spalte ist für die Bedienungselemente vorgesehen.

Die Stile auf der Webseite werden hauptsächlich mit den Bootstrap-Klassen definiert. Die Bootstrap-Klassen können überschrieben und ergänzt werden. Dafür muss die benutzerdefinierte CSS-Datei nach den Bootstrap CSS-Dateien im head-Bereich der HTML-Seite eingebunden werden.

Für die Verarbeitung der Benutzereingaben werden jQuery-Ereignisse an die Bedienungselemente der Webseite gebunden. Dabei handelt es sich um die folgenden Bedienungselemente: Auswahlliste, Radiobutton, Schieberegler und Linkbutton. Die Interaktion mit den Auswahllisten und Radiobuttons wird anhand des `change`-Ereignis verfolgt. Die Auswahl eines Datensatzes erfolgt durch die Auswahlliste. Dabei wird ein Ereignis ausgelöst und der Benutzer wird zum ausgewählten Datensatz weitergeleitet.

Eine Auswahlliste wird mit dem <select>-Tag erstellt. Die Einträge der Auswahlliste sind mit dem <option>-Tag definiert. Jeder Eintrag besitzt ein `value`-Attribut, welches den Wert dieses Eintrags bestimmt. In Listing 5.19 wird der ausgewählte Wert der Auswahlliste ermittelt. Je nach Wert wird die Weiterleitung auf die entsprechende URL ausgeführt. Wobei das Kürzel `'st_bd'` der Wert des `value`-Attributs ist, welcher dem Datensatz DS1 entspricht.

```
$('#data-selector-form').on('change', function (e) {
    switch ($('#data-selector').val()) {
        case 'st_bd': {
            window.location = "/map/";
            break;
        }
        ...
    }
});
```

Listing 5.19: *jQuery change-Ereignis für die Auswahl eines Datensatzes*

Bei der Veränderung zusätzlicher Optionen wie Geschlecht, Nationalität, Semester und/oder Bundesland wird der Seiteninhalt mittels Ajax aktualisiert (siehe Listing 5.20).

```
$('#options-selector-form').on('change', function (e) {
  ...
    $.ajax({
       data: {
          dataframe: $('#data-selector').val(),
          year: $('#year-selector').val(),
          nationality: $('input[name=nationality]:checked').val(),
          gender: $("input[name=gender]:checked").val()},
       type: 'POST',
       url: '/mapupdate/'})
    .done(function (data, statusText, xhr) {
       $('#ws').text("Wintersemester "+data.year+','
            +data.nationality+', '+data.gender);
       let $new_map = $(data.map);
             ...
       $('#folium-map').append($new_map);
     });
  ...
});
```

Listing 5.20: *Aktualisierung des Seiteninhaltes beim change-Ereignis*

Die ausgewählten Parameter werden mit der Ajax-Anfrage an das Backend übermittelt. Anhand der Parameter wird im Backend eine neue Karte generiert, die mit Ajax-Antwort an das Frontend zurückgeliefert wird. Auf die gleiche Weise wird der Seitenüberschrift aktualisiert.

Auf der Seite mit dem zweiten Datensatz DS2 befindet sich statt der Auswahlliste ein Schieberegler für die Zeitauswahl. Die Veränderung des Wertes beim Schieberegler wird mit dem input-Ereignis verfolgt. Im Gegensatz zum `change`-Ereignis wird das `input`-Ereignis synchron bei der Änderung des `value`-Attributs des <input>-Tags ausgelöst. Dabei werden mittels Ajax-Calls die Tabelle und das Liniendiagramm aktualisiert.

Mit dem Klick auf die untere Leiste wird ein ausklappbarer Bereich mit der zusätzlichen Information angezeigt. Mit einem weiteren Klick wird dieser Bereich zugemacht. Dafür wird ein jQuery `click`-Ereignis an diese Leiste gebunden.

5.6. Deployment

Nachdem die Implementierungsphase abgeschlossen wurde, kommt der letzte Schritt - die Bereitstellung der Webanwendung auf dem Server, damit sie für alle im Internet zugänglich ist. Die Voraussetzungen dafür sind das Vorhandensein eines Domainnames und eines Hosting-Servers. Es gibt mehrere Domain-Anbieter, welche Domainnamen kostenpflichtig oder kostenlos registrieren. Da die Autorin bereits den Domainnamen anisimova.de besitzt, wurde es nicht nötig einen Neuen zu registrieren. Für die Webanwendung wird eine Subdomain students.anisimova.de erstellt. Es wird ein Hosting-Server mit dem Betriebssystem Linux, Version Ubuntu 16.04, des Cloud-Server Anbieters Scaleway[55] benutzt. Scaleway ist ein europäisches Unternehmen mit Rechenzentren in den Niederlanden und Frankreich. Es stellt preiswert mehrere Konfigurationen von Webservern zur Verfügung und bietet unbegrenzt Traffic. Für diese Arbeit wird folgende Konfiguration verwendet:

Kerne	1 x86-64
Arbeitsspeicher	1 GB
SSD-Speicher	25 GB
Bandbreite	100 MBit/s

Wenn eine Domain und ein Hosting-Server vorhanden sind, sollen sie miteinander verbunden werden. Dafür wird zur Domain eine IP-Adresse des Servers hinzugefügt.

Die Einrichtung eines Servers besteht aus mehreren Komponenten. Es werden ein Webserver, ein WSGI-Server und die Python-Umgebung mit zugehörigen Paketen installiert. Anschließend wird der Quellcode der Webanwendung aus dem Git-Repository auf den Server heruntergeladen und installiert. Ein Webserver bekommt die HTTP-Anfragen vom Client, z.B. einem Webbrowser und sendet eine Antwort zurück. Als Webserver wird Caddy

[55] https://www.scaleway.com/ [19.03.2019]

[56] benutzt. Caddy ist ein kleiner HTTP/2-Webserver mit automatischer HTTPS-Unterstützung. Er liefert für jede definierte Domain ein Let's Encrypt-Zertifikat. Caddy identifiziert alle HTTP-Anfragen und verteilt sie zwischen den entsprechenden Anwendungen auf dem Server.

Ein WSGI-Server ist ein Webserver für Python-Anwendungen. Er stellt eine Schnittstelle zwischen dem Webserver und der Python-Anwendung zur Verfügung. Als solcher wird der WSGI HTTP-Server Gunicorn[57] benutzt. Da auf den Produktionsserver gleichzeitig mehr als ein Client zugreifen kann, muss es möglich sein gleichzeitig mehr als eine Kopie der Anwendung auf dem Server starten zu können. Der WSGI HTTP-Server Gunicorn kümmert sich darum. Gunicorn startet mehrere Arbeiter-Prozesse, welche die Anfragen an den Server abarbeiten.

Für die Erstellung einer Python-Umgebung und die Verwaltung von benötigten Python-Paketen wird auf dem Server miniconda[58] installiert. Miniconda ist eine leichtgewichtige Version von Anaconda.

Die Installation von Caddy und Gunicorn wird anhand des Tutorials[59] durchgeführt. Die Kommunikation zwischen verschiedenen Komponenten des Servers ist in der Abbildung 5.27 vorgestellt.

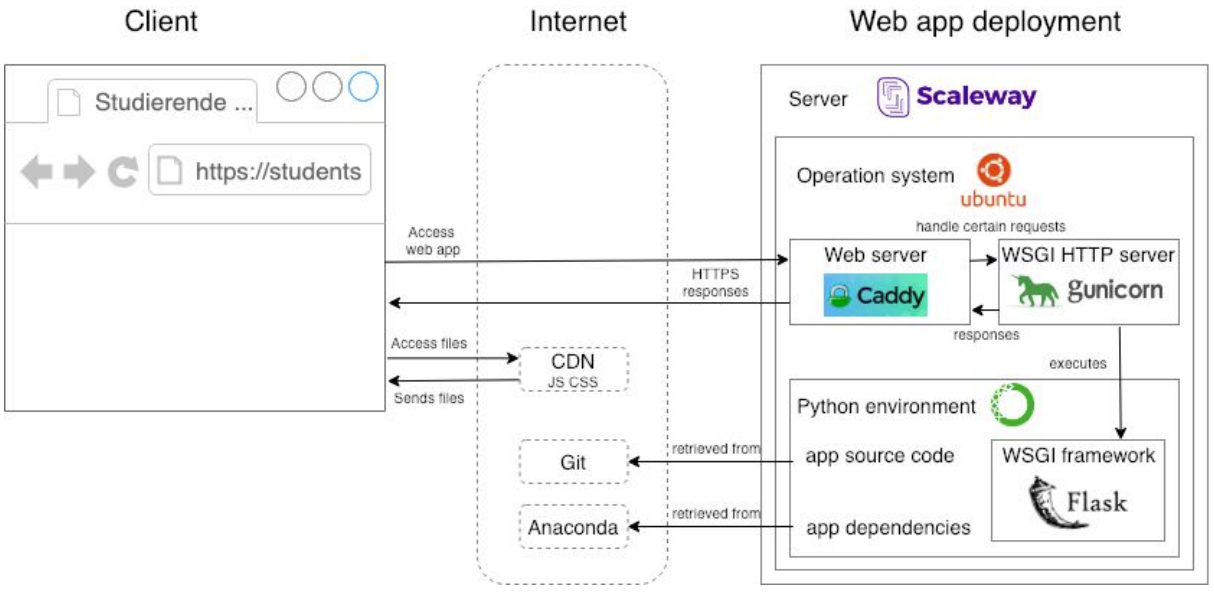

Abb. 5.27: *Das Schema der Kommunikation zwischen den Serverkomponenten*[60]

[56] [CAD]
[57] [GUN]
[58] [MIN]
[59] [EHT] [SHT]
[60] Nach der Abbildung auf der Seite "Deployment". In: Full Stack Python. URL: https://www.fullstackpython.com/deployment.html [27.03.2019]

6. Zusammenfassung und Ausblick

Im Rahmen dieser Bachelorarbeit wurde eine Webanwendung entwickelt, die drei Visualisierungen zum Thema “Studierende in Deutschland” umfasst. Die Visualisierungen informieren über die Anzahl und die Verteilung der Studierenden in Deutschland, sowie über die Gründung von Hochschulen im Zeitraum von 1386 bis 2017.

Die Arbeit an dem Projekt hat mit der Recherche nach den offenen Datenquellen angefangen, dessen Daten der Anwendung zugrunde liegen. Erst danach wurde der Schwerpunkt der Bachelorarbeit festgelegt und die Anforderungen an das Projekt formuliert. Ein wesentlicher Teil dieser Arbeit besteht aus der Vorbereitung der Daten. Die ursprüngliche Struktur der Datensätze musste an die ausgewählten Algorithmen angepasst werden. Da die Daten eine räumliche Anordnung besitzen, wird die Choroplethenkarte als Hauptvisualisierungsmethode ausgewählt. Außerdem werden Linien- und Balkendiagramme eingesetzt, um einen tieferen Einblick in das Thema zu ermöglichen. Ausgehend von ausgewählten Visualisierungsmethoden und Programmiersprachen wurde die Analyse der vorhandenen Visualisierungsbibliotheken durchgeführt. Die Python-Bibliothek Folium stellt viele Möglichkeiten für kartenbasierte Visualisierungen. Trotzdem bestand ein Bedürfnis eine eigene Komponente zu entwickeln, um die Gründung von Hochschulen im Zeitverlauf zu visualisieren. Nach Abschluss der Entwicklungsphase, wurde ein Server eingerichtet, damit die Webanwendung für alle offen zugänglich ist. Das Projekt ist unter die URL https://students.anisimova.de/map/ veröffentlicht.

Aus zeitlichen Gründen wurde das Kann-Kriterium K2 “Animation abspielen” nicht implementiert. Inhaltlich wurde das Projekt dadurch nicht beeinträchtigt.

Die Arbeit beschränkt sich auf die Desktopversion der Webanwendung, obwohl die Softwarekomponenten, die für die Frontend-Entwicklung ausgewählt wurden, ein responsives Layout unterstützen. Die Entwicklung für mobile Geräte stellt viel mehr Anforderungen als nur die Anordnung von Seitenelementen. Die Interaktion mit dem Seiteninhalt erfolgt auf mobilen Geräten mit Gesten, die nicht immer Eins-zu-eins mit der Bedienung durch die Maus zugeordnet werden können. Die Verarbeitung von Touch-Ereignissen erfordert einen höheren Aufwand als die Unterstützung von Maus-Ereignissen.

Es wurde für die Benutzeroberfläche die deutsche Sprache ausgewählt. Bei Bedarf können weitere Sprachen, z.B. durch die Flask-Erweiterung Flask-Babel[61], hinzugefügt werden. Flask-Babel stellt die Werkzeuge für die Erstellung von mehrsprachigen Anwendungen zur Verfügung. Die Übersetzung wird aber nicht automatisch generiert. Die Texte sollen manuell von Übersetzern in eine andere Sprache übertragen werden.

Alle Muss-Kriterien, die an dem Projekt gestellt wurden, sind in der Bachelorarbeit implementiert worden. Es gibt noch viele weitere Ideen, die das Programm ergänzen können. Das Thema kann weiter erforscht werden, z.B. unter den folgenden Aspekten:

- Anteil von ausländischen Studierenden in Deutschland

[61] [BAB]

- Deutsche Studierende im Ausland

Weiterhin kann man sich mit der Vorhersage beschäftigen in dem man beispielsweise ein Vorhersagemodell für die Studierendenzahl in Deutschland aufbaut. Das kann zum Beispiel ein Modell sein, das die Studierendenzahl für jedes Bundesland für die nächsten 3-5 Jahre vorhersagt. Damit ein Modell eine hohe Genauigkeit besitzt, werden noch weitere Daten benötigt. Als solche Daten können beispielsweise die Bevölkerungszahl mit der Altersangabe, das Alter der Studierenden, die Anzahl der Studienplätze sein. Die Vorhersage kann für die Planung der Wohnplätze für Studierende, der Anzahl von Lehrkräften, der Belastung der Mensen und v.a. benutzt werden. Diese Forschung kann einer besseren Ressourcenprognose zugrunde liegen.

Literaturverzeichnis

[ANA] Anaconda.com. Anaconda. URL: https://www.anaconda.com/ [06.02.2019]

[ARC] Mad-gooze. Leaflet.Arc. URL: https://github.com/mad-gooze/Leaflet.Arc [04.02.2019]

[BAB] Pythonhosted.org. Flask-Babel. URL: https://pythonhosted.org/Flask-Babel/ [22.03.2019]

[BOK] Bokeh.Pydata.org. Bokeh documentation. URL: http://bokeh.pydata.org/en/latest/ [16.02.2019]

[BOO] Getbootstrap.com. Bootstrap. URL: https://getbootstrap.com/ [16.02.2019]

[BOT] Marcel Hellkamp. Bottle: Python Web Framework. Bearbeitungsstand: 26.03.2019. URL: https://bottlepy.org/docs/dev/ [26.03.2019]

[CAD] Caddyserver.com. Caddy User Guide. URL: https://caddyserver.com/docs [19.03.2019]

[D3J] Mike Bostock. D3.js - Data-Driven Documents. URL: https://d3js.org/ [09.02.2019]

[DJA] Djangoproject.com. The web framework for perfectionists with deadlines. URL: https://www.djangoproject.com/ [10.03.2019]

[EHT] Justin Ellingwood. How To Serve Flask Applications with Gunicorn and Nginx on Ubuntu 16.04. Bearbeitungsstand: 19.05.2016. URL: https://www.digitalocean.com/community/tutorials/how-to-serve-flask-applications-with-gunicorn-and-nginx-on-ubuntu-16-04 [19.03.2019]

[FLA] Armin Ronacher. Flask (A Python Microframework). URL: http://flask.pocoo.org/ [10.03.2019]

[FOL] Python-visualization.github.io. Folium documentation. URL: https://python-visualization.github.io/folium/ [09.02.2019]

[GEO] Geopandas.org. GeoPandas documentation. URL: http://geopandas.org/ [10.02.2019]

[GTF] Miguel Grinberg. The Flask Mega-Tutorial Part XIII: I18n and L10n. Bearbeitungsstand: 27.02.2018. URL: https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-xiii-i18n-and-l10n [22.03.2019]

[GUN] Gunicorn.org. Gunicorn documentation. URL: https://gunicorn.org/#docs [19.03.2019]

[HER] Denis Carriere. HERE - Geocoder. URL: https://geocoder.readthedocs.io/providers/HERE.html [04.03.2019]

[JUP] Jupyter.org. Project Jupyter. URL: https://jupyter.org/ [06.02.2019]

[JQU] Jquery.com. Jquery. URL: https://jquery.com/ [26.02.2019]

[JSO] Python.org. JSON encoder and decoder library. URL: https://docs.python.org/3/library/json.html [06.02.2019]

[LEA] Vladimir Agafonkin. Leaflet.js - a JavaScript library for interactive maps. URL: https://leafletjs.com/ [09.02.2019]

[MIN] Conda.io. Miniconda documentation. URL: https://docs.conda.io/en/latest/miniconda.html [19.03.2019]

[MFS] Matt Makai. Full Stack Python. Deployment. URL: https://www.fullstackpython.com/deployment.html [20.03.2019]

[NKS] Cole Nussbaumer Knaflic. Storytelling mit Daten: Die Grundlagen der effektiven Kommunikation und Visualisierung mit Daten. Vahlen, 2017

[NRW] Douglas Nelson, Valerie S. Reed, John R. Walling. (1976). Pictorial superiority effect. Journal of experimental psychology. Human learning and memory. 2. 523-8. 10.1037/0278-7393.2.5.523. URL: https://www.researchgate.net/publication/22152101_Pictorial_superiority_effect [10.03.2019]

[NUM] Numpy.org. Numpy. http://www.numpy.org/ [06.02.2019]

[PAN] Pandas.Pydata.org. Python Data Analysis Library. URL: https://pandas.pydata.org/ [06.02.2019]

[PIC] Python.org. Pickle library. URL: https://docs.python.org/3/library/pickle.html [27.02.2019]

[PLO] Plot.ly. Plotly Javascript Open Source Graphing Library. URL: https://plot.ly/javascript/ [09.02.2018]

[PYT] Python.org. About Python. URL: https://www.python.org/ [16.02.2019]

[RPR] R-Project.org. The R Project for Statistical Computing. URL: https://www.r-project.org/ [14.02.2019]

[SHT] Tom Stoneham. How To Host a Website with Caddy on Ubuntu 16.04. Bearbeitungsstand: 12.04.2018. URL: https://www.digitalocean.com/community/tutorials/how-to-host-a-website-with-caddy-on-ubuntu-16-04#prerequisites [19.03.2019]

[SNA] IvanSanchez. Leaflet.Polyline.SnakeAnim. URL: https://github.com/IvanSanchez/Leaflet.Polyline.SnakeAnim [11.02.2019]

[TCR] Silas Toms, Paul Crickard, Eric van Rees. Mastering Geospatial Analysis with Python: Explore GIS processing and learn to work with GeoDjango, CARTOframes and MapboxGL-Jupyter. Packt Publishing Ltd, 2018

[WIK] Wiki.selfhtml.org. JavaScript Tutorials Mouse and More. Bearbeitungsstand: 05.06.2018. URL: https://wiki.selfhtml.org/wiki/JavaScript/Tutorials/Mouse_and_More#Auswahl_der_geeigneten_Events [09.03.2018]

[XLR] PyPi.org. xlrd library. URL: https://pypi.org/project/xlrd/ [06.02.2019]

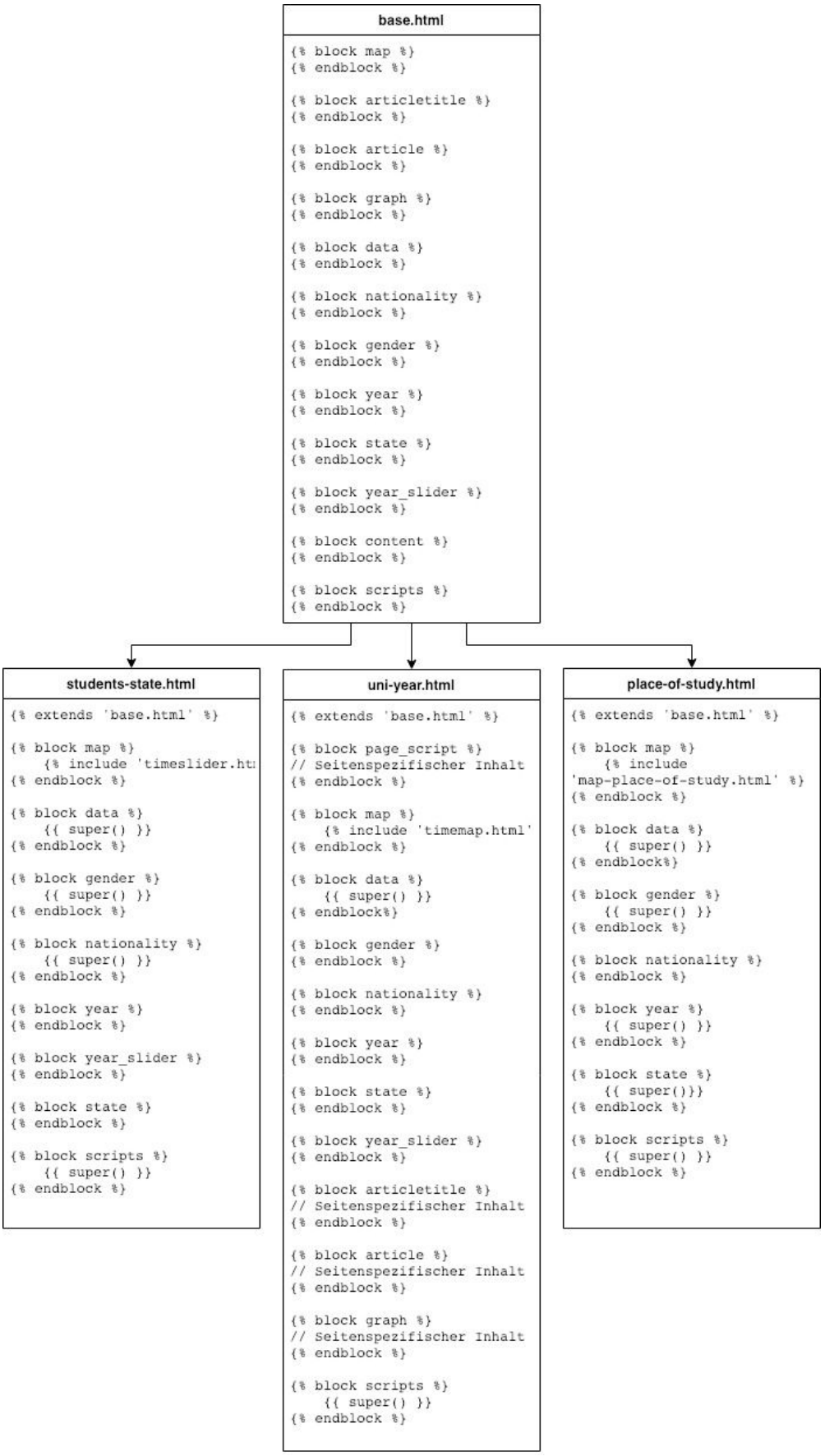

Abb. 5.4: *Die Vorlagenvererbung*

	Hs-Nr.	Hochschulkurzname	Hochschulname	Hochschultyp	Trägerschaft	Bundesland	Anzahl Studierende	Gründungsjahr	Promotionsrecht	Habilitationsrecht	...	PLZ	Ort	Postfach	Postleitzahl (Postanschrift)	Ort (Postanschrift)	Telefonvorwahl	Telefon	Fax	Home Page	Mitglied HRK
0	2	Aachen FH	Fachhochsch...	Fachhochsch...	öffentlich-...	Nordrhein-W...	14083.0	1971.0	Nein	Nein	...	52066.0	Aachen	100660	52005.0	Aachen	241	6009-0	6009-51090	http://www....	1.0
1	1	Aachen TH	Rheinisch-W...	Universitäten	öffentlich-...	Nordrhein-W...	45403.0	1870.0	Ja	Ja	...	52062.0	Aachen	NaN	52056.0	Aachen	241	80-1	80-92312	http://www....	1.0
2	3	Aalen H	Hochschule ...	Fachhochsch...	öffentlich-...	Baden-Württ...	5928.0	1962.0	Nein	Nein	...	73430.0	Aalen	1728	73428.0	Aalen	7361	576-0	576-2250	http://www....	1.0
3	4	Albstadt-Si...	Hochschule ...	Fachhochsch...	öffentlich-...	Baden-Württ...	3545.0	1971.0	Nein	Nein	...	72488.0	Sigmaringen	1254	72481.0	Sigmaringen	7571	732-0	732-8229	www.hs-albs...	1.0
4	349	Alfter HfK	Alanus Hoch...	Kunst- und ...	privat, sta...	Nordrhein-W...	1575.0	1973.0	Ja	Nein	...	53347.0	Alfter	NaN	NaN	NaN	2222	9321-0	9321-21	https://www....	0.0
5	280	Amberg-Weid...	Ostbayerisc...	Fachhochsch...	öffentlich-...	Bayern	3096.0	1994.0	Nein	Nein	...	92224.0	Amberg	1462	92204.0	Amberg	9621	482-0	482-4991	http://www....	1.0
6	5	Anhalt H	Hochschule ...	Fachhochsch...	öffentlich-...	Sachsen-Anhalt	7159.0	1991.0	Nein	Nein	...	6366.0	Köthen	1458	6354.0	Köthen	3496	67-1000	67-1099	http://www....	1.0

Abb. 5.13: *Ausgangstabelle DS2*

	Einwohner in Deutschland nach Bundesländern¹)	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unnamed: 11	Unnamed: 12	Unnamed: 13	Unnamed: 14	Unnamed: 15	Unnamed: 16	Unnamed: 17
0	NaN	1.000 Personen (Jahresdurchschnitt)	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Jahr	BW	BY	BE	BB	HB	HH	HE	MV	NI	NW	RP	SL	SN	ST	SH	TH	D
2	1991	9904	11517.8	3436.2	2559.8	682.4	1658.7	5797.9	1907.2	7427.7	17421.3	3792.1	1074.5	4719.5	2847.7	2636.1	2590.6	79973.4
3	1992	10050.4	11668.8	3444.4	2540.2	684	1673	5872.2	1876.5	7515.1	17568.6	3850.3	1079	4653.6	2807.4	2660.2	2555.9	80499.8
4	1993	10149.3	11792.7	3450.7	2535.5	683.3	1685.9	5931.7	1851.2	7594.3	17676	3902.1	1081.9	4613.8	2782.8	2680.8	2534.4	80946.5
5	1994	10195.3	11859.9	3445.1	2530.5	679.9	1690.3	5955.4	1833.6	7656.2	17728.4	3936.9	1081.2	4581.5	2762.1	2692.9	2518.3	81147.5
6	1995	10223.1	11917	3434.2	2530.6	677.8	1689	5971.7	1822.2	7714.6	17779.9	3962.4	1080.3	4556.6	2740.7	2706	2501.7	81307.7
7	1996	10260.1	11970.3	3418.4	2537.1	676.2	1686.2	5990.3	1813.3	7756.9	17830.8	3986.4	1079.5	4532.8	2721	2721.1	2486.4	81466.4
8	1997	10285.1	11999.3	3386	2550.2	672.8	1680.8	5996.5	1804.2	7782.2	17856.9	4005.9	1076.9	4506.2	2700.4	2735.1	2471.5	81509.9
9	1998	10297.4	12013	3346.2	2565.7	667.4	1672.9	5995.8	1793.6	7800.6	17856.1	4017.6	1071.1	4473.7	2673.9	2745.5	2455.5	81446
10	1999	10323.5	12049.7	3316.8	2577.2	661.6	1668.9	6001.2	1783	7820.4	17853.8	4023.6	1065.7	4438.1	2645.5	2754.4	2439.1	81422.4

Abb. 5.23: *Tabelle DSB*

	Tab.2.5.32	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 12	Unnamed: 13	Unnamed: 14	Unnamed: 15	Unnamed: 16	Unnamed: 17	Unnamed: 18	Unnamed: 19	Unnamed: 20	Unnamed: 21
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Studierende nach Geschlecht, Land des Studieno...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	Land des Erwerbs der Hochschulzugangsberechtigung	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	Land des Erwerbs der Hochschulzugangsberechtigung	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	Insgesamt	Baden-Württemberg	Bayern	Berlin	Brandenburg	Bremen	Hamburg	...	Nieder-sachsen	Nordrhein-Westfalen	Rheinland-Pfalz	Saarland	Sachsen	Sachsen-Anhalt	Schleswig-Holstein	Thüringen	Ohne Angabe	Ausland
5	Winter-semester	Land des Studienortes	Geschlecht	Anzahl	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	Anzahl	NaN	NaN	NaN	NaN
6	2006/2007	Baden-Württemberg	Insgesamt	245143	158162	12929	1518	1177	470	920	...	4408	8897	10450	1478	2018	1000	1554	1573	54	29364
7	NaN	NaN	Männlich	129654	86446	7149	672	486	255	471	...	2322	4646	5420	643	836	406	850	686	15	13956
8	NaN	NaN	Weiblich	115489	71716	5780	846	691	215	449	...	2086	4251	5030	835	1182	594	704	887	39	15408
9	NaN	Bayern	Insgesamt	257898	18875	179532	1608	1368	332	800	...	4044	7426	2330	496	4583	1303	1329	3788	8	23850
10	NaN	NaN	Männlich	131650	9367	95030	750	626	159	405	...	2063	3866	1244	236	1777	593	719	1546	8	10145

Abb. 5.24 *Ausgangstabelle DS3*

	WS	Bundesland	Studienort	Geschlecht	Insgesamt	Baden-Württemberg	Bayern	Berlin	Brandenburg	Bremen	Hamburg	...	Niedersachsen	Nordrhein-Westfalen	Rheinland-Pfalz	Saarland	Sachsen	Sachsen-Anhalt	Schleswig-Holstein	Thüringen	Ohne Angabe	Ausland
6	2006/2007	Baden-Württemberg	Insgesamt	245143	158162	12929	1518	1177	470	920	...	4408	8897	10450	1478	2018	1000	1554	1573	54	29364
7	2006/2007	Baden-Württemberg	Männlich	129654	86446	7149	672	486	255	471	...	2322	4646	5420	643	836	406	850	686	15	13956
8	2006/2007	Baden-Württemberg	Weiblich	115489	71716	5780	846	691	215	449	...	2086	4251	5030	835	1182	594	704	887	39	15408
9	2006/2007	Bayern	Insgesamt	257898	18875	179532	1608	1368	332	800	...	4044	7426	2330	496	4583	1303	1329	3788	8	23850
10	2006/2007	Bayern	Männlich	131650	9367	95030	750	626	159	405	...	2063	3866	1244	236	1777	593	719	1546	8	10145
11	2006/2007	Bayern	Weiblich	126248	9508	84502	858	742	173	395	...	1981	3560	1086	260	2806	710	610	2242	-	13705
12	2006/2007	Berlin	Insgesamt	132822	6055	4302	59524	14222	726	1353	...	4967	7477	1295	331	3169	2590	1927	1693	1	16800
13	2006/2007	Berlin	Männlich	66840	3228	2263	31681	6858	360	630	...	2513	3763	676	167	1271	1102	999	708	1	7589
14	2006/2007	Berlin	Weiblich	65982	2827	2039	27843	7364	366	723	...	2454	3714	619	164	1898	1488	928	985	-	9211
15	2006/2007	Brandenburg	Insgesamt	42331	1012	690	10089	16792	111	265	...	1034	1281	234	38	2169	1316	386	560	2	4866

Abb. 5.25 *Tabelle DS3 nach der Bereinigung*