Stat(e)hopper

Dokumentation

Alexander Prange, Daniel Grevismühl, Melanie Windrich

13. Juli 2015

Inhaltsverzeichnis

1 Einleitung

Diese Dokumentation beschreibt das Konzept des Spiels „Statehopper“, sowie den Entwicklungsprozess eines Prototypen dieses Spiels. Beides ist im Rahmen des Moduls „Theoretical Foundations and Applications of Interactive Multimedia“ von Prof. Dr. habil. Ansgar Scherp entstanden. Dieses Kapitel beschreibt anfangs kurz die Aufgabenstellung und anschließend grob die Idee des Spiels und seine Zielgruppe. In Kapitel 2 wird das Konzept dann mit Hilfe eines Storyboars und einer Navigation Map näher erläutert. Anschließend gibt Kapitel 3 einen Einblick in die Entwicklung des Prototypen. Hierzu werden insbesondere die genutzten Werkzeuge kurz dargestellt und die verwendeten Medienassets und Datenquellen aufgeführt. In Kapitel 4 wird dann kurz zusammengefasst, inwiefern der entwickelte Prototyp den Anforderungen gerecht wird und einige Ideen für eine Weiterentwicklung präsentiert.

1.1 Anforderungen

Die Anforderungen an das Projekt wurden nur sehr grob formuliert. Im wesentlichen bestanden sie daraus, dass eine Anwendung entwickelt werden soll, welche drei verschiedene Kriterien erfüllt:

Interaktivität

Multimedial Es sollten verschiedene Arten von Medien verwendet werden. Insbesondere sollte mindestens ein diskretes Medium (Text, Bild) und mindestens ein kontinuierliches Medium (Video, Audio, Animation) zum Einsatz kommen.

Datenvisualisierung Zweck der Anwendung sollte es sein, beliebige Daten zu visualisieren, wobei es erwünscht war, Linked Open Data zu verwenden

1.2 Exposé

Im Projekt Statehopper wurde der Prototyp eines Spiels entwickelt, in dem von einem Land zu einem anderen gereist werden muss. Um das zu erreichen, wird das Wissen des Spielers über die verschiedenen Länder auf die Probe gestellt. Der Spieler startet in einem zufälligen Land Europas und muss nun mit möglichst wenig Hops zum ebenfalls zufälligen Zielland springen.

Dabei kann er jedoch nicht beliebig in die Nachbarländer springen, sondern muss verschiedene Eigenschaften der Länder hierfür ausnutzen. Für das aktuelle Land werden dem Spieler fünf Eigenschaften des Landes (mit numerischem Wertebereich) gezeigt.

Nun hat er die Möglichkeit zu entscheiden, welche dieser Eigenschaften betrachtet werden soll und ob diese im gewünschten Nachbarland wohl höher oder niedriger als im aktuellen Land ist. Die Wahl des Spielers bestimmt den nächsten Hop in ein Nachbarland. Dies geschieht nach zwei Kriterien:

1. Die gewählte Eigenschaft muss im Vergleich zum aktuellen Land in die gewählte Richtung abweichen und

2. falls mehrere Länder in diese Richtung abweichen, wird dasjenige gewählt, dass die geringste Differenz aufweist.

Die Eigenschaften stammen aus verschiedenen Themengebieten. Diese sind z.B. Demographie, Wirtschaft oder Verkehr. In Kapitel 3 findet sich eine Liste der Eigenschaften inklusive deren Quellen (siehe Tabelle 3.2).

Für jedes Spiel können dann aus der Liste aller Eigenschaften beliebig viele (mindestens fünf) verschiedene aktiviert werden, sodass ein Pool von Eigenschaften vorhanden ist, aus denen dann am Spielstart fünf Eigenschaften gewürfelt werden, welche dann für diese Spielrunde aktiv sind.

Neben dem eigentlichen Spielmodus gibt es auch noch einen freien Modus. In diesem können sämtliche Länder bereist und die zugehörigen, bereits freigespielten Daten betrachtet werden. Die Darstellung erfolgt in Form eines Radarplots.

Nach dem erfolgreichen Beenden des Spiels können alle gewonnenen Daten noch einmal in Balkendiagrammen begutachtet werden. Außerdem wird die zurückgelegte Strecke auf der Karte visualisiert.

1.3 Zielgruppe

Im Prinzip kann jede Person zur Zielgruppe gezählt werden, die ein Interesse aufweist, Wissen über die Länder Europas zu erlangen. Das System ist hierdurch gut als Lernspiel geeignet. Durch den Spielcharakter ist das Spiel auch gut für ältere Kinder geeignet. Der aktuelle Prototyp kann theoretisch mit beliebigen Eigenschaften und anderen Kontinenten erweitert werden, sodass das Spiel auf verschiedene Zielgruppen angepasst werden könnte.

2 Skript

Vor der Implementierung des Prototypen wurde ein Skript angefertigt, welches neben dem Exposé (siehe Abschnitt 1.2) insbesondere aus einem Storyboard und einer Navigation Map besteht.

2.1 Storyboard

Die folgenden Skizzen zeigen das Storyboard, welches als Grundlage für den entwickelten Prototypen diente.

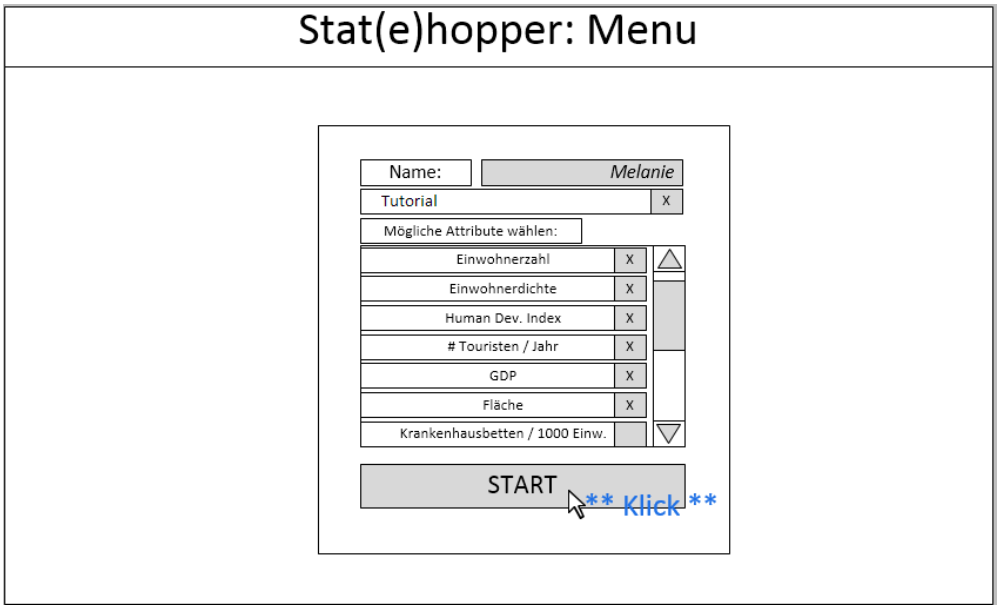

Abbildung 2.1: Nach der Startseite kann der Benutzer in einem Menü seinen Namen angeben und die zur Verfügung stehenden Eigenschaften eingrenzen. Anschließend kann das Spiel gestartet werden (wenn mindestens 5 Eigenschaften ausgewählt wurden). In einer erweiterten Version wäre noch eine Auswahl des zu bespielenden Kontinents denkbar. Dies wurde im entwickelten Prototyp aber nicht implementiert.

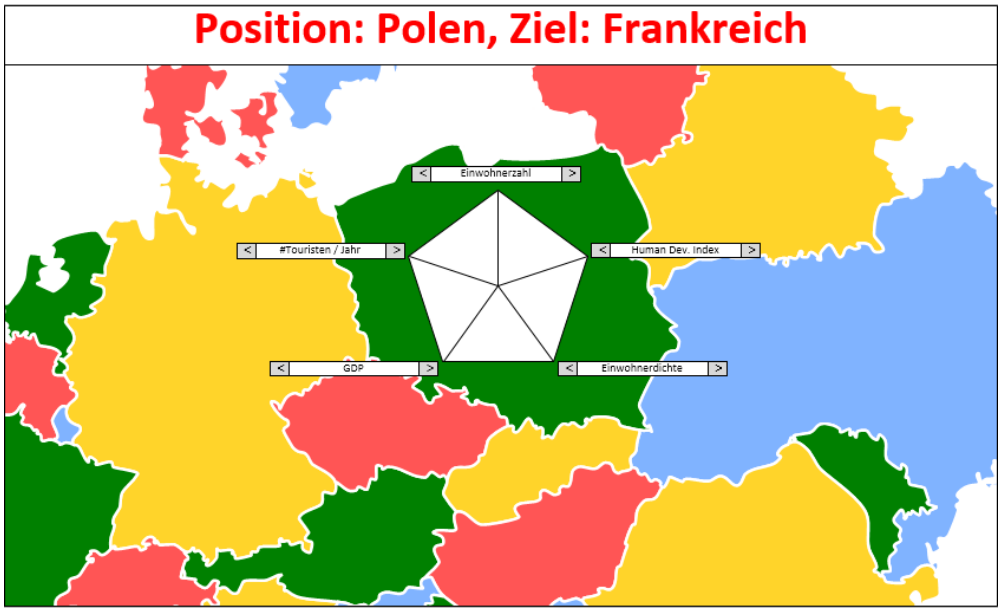

Abbildung 2.2: Das eigentliche Spiel beginnt in einem zufällig ausgewählten Land (hier Polen). Auf diesem Land wird ein Radarplot mit fünf der ausgewählten Eigenschaften dargestellt. Am oberen Bildschirmrand wird dem Spieler das ebenfalls zufällig ausgewählte Zielland (hier Frankreich) angezeigt.

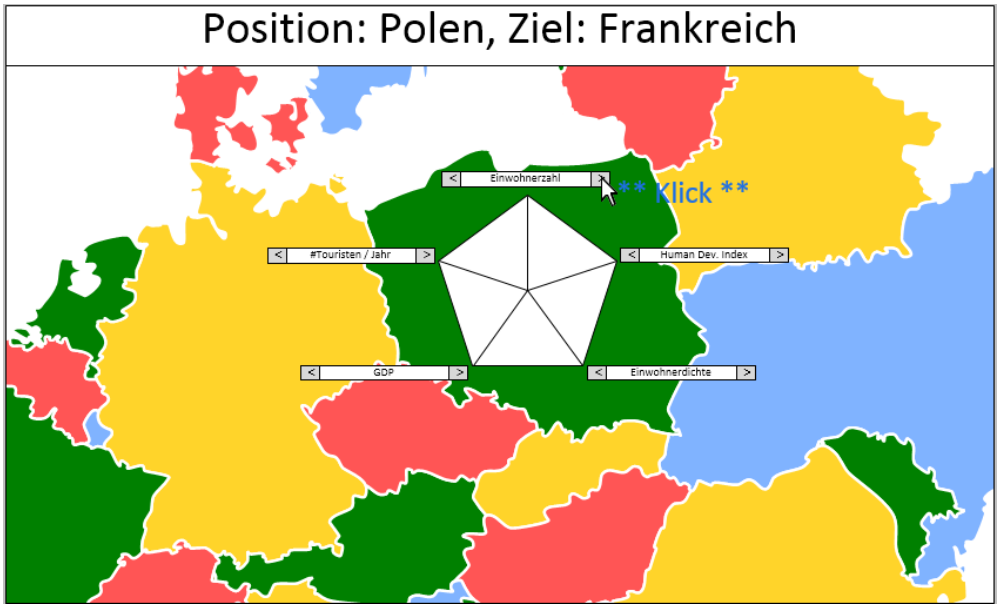

Abbildung 2.3: Um in das nächste Land zu springen muss der Spieler sich entscheiden, welche Eigenschaft er verwenden möchte und ob diese erhöht oder verringert werden soll. Der Spieler entscheidet sich, die Einwohnerzahl zu erhöhen.

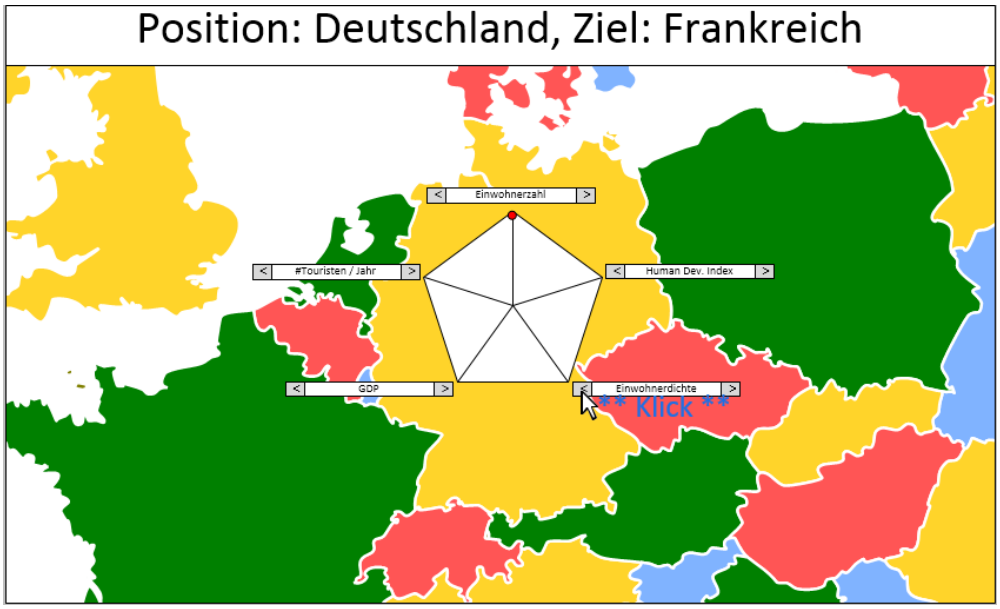

Abbildung 2.4: Von allen Nachbarländern Polens hat Deutschland die nächst höhere Einwohnerzahl. Deshalb springt der Spieler in dieses Land. Im hier eingezeichneten Radarplot erscheint nun ein Punkt an der Achse für die Einwohnerzahl, weil dieses Datum mit dem Hop freigespielt wurde. Dass der Punkt ganz oben ist, signalisiert, dass Deutschland die höchste Einwohnerzahl aller Länder hat. Ein Punkt am Ursprung wiederum würde den niedrigsten Wert repräsentieren. Anschließend entscheidet sich der Spieler, die Einwohnerdichte zu reduzieren.

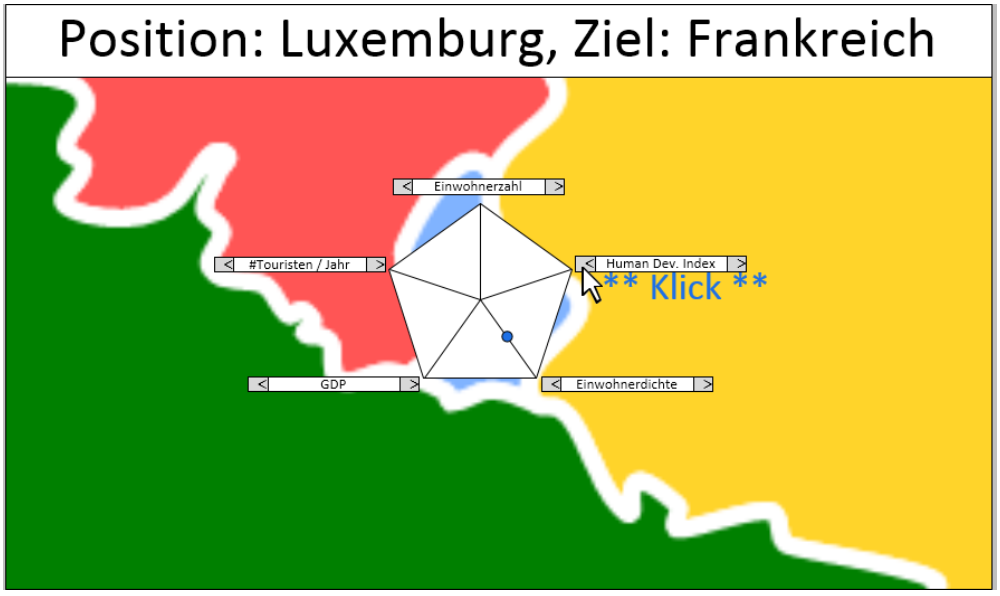

Abbildung 2.5: Dies führt nach Luxemburg, da zwar auch andere Nachbarländer eine geringere Einwohnerdichte als Deutschland haben, die Abweichung bei Luxemburg aber die niedrigste ist. Nun soll der Human Development Index verringert werden.

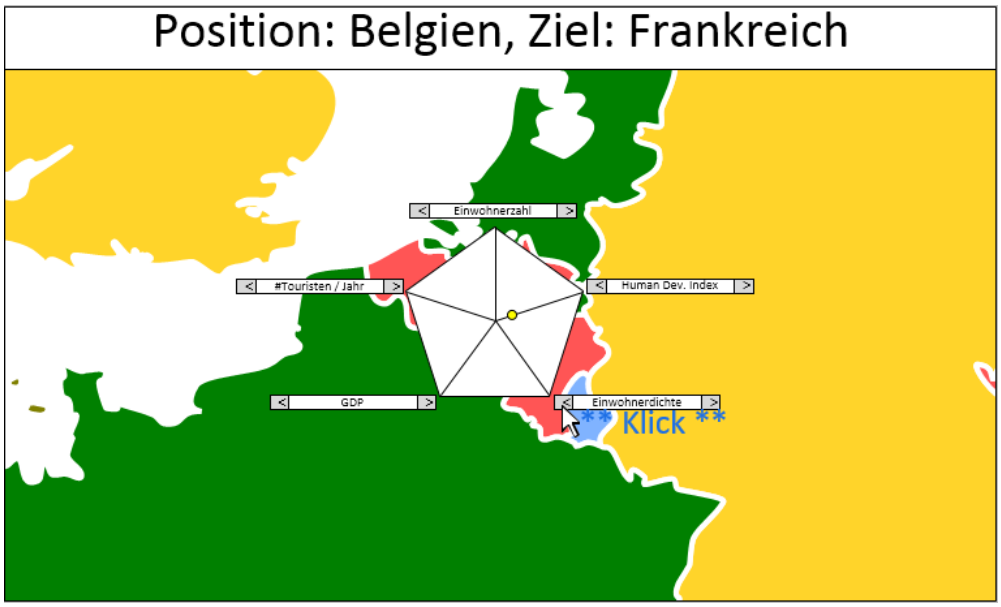

Abbildung 2.6: Dies hat den Spieler noch immer nicht nach Frankreich geführt, sondern nach Belgien. Er entscheidet sich erneut, die Einwohnerdichte zu verringern.

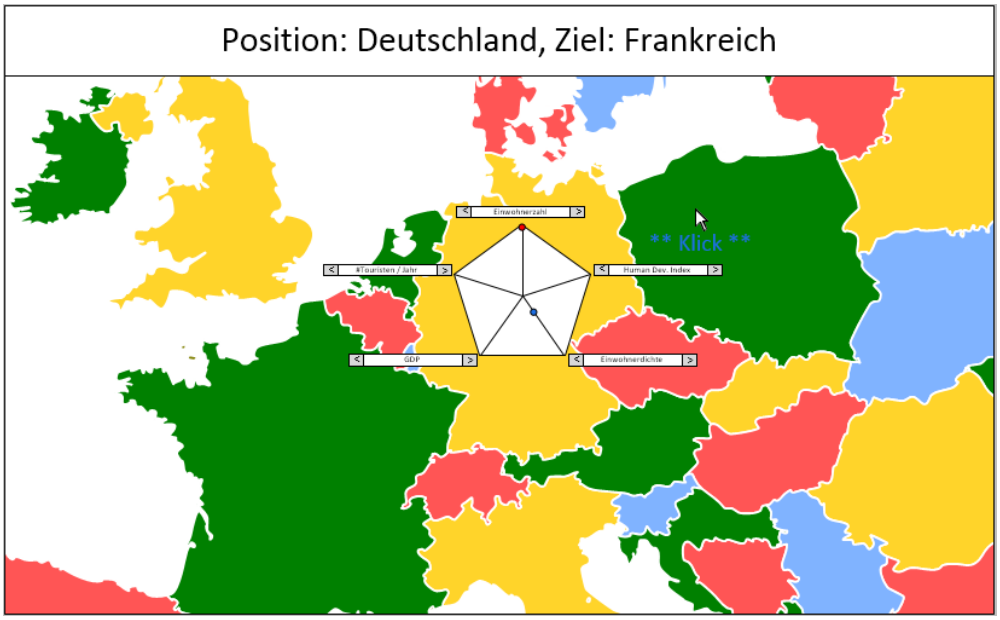

Abbildung 2.7: Er landet wieder in Deutschland. Hier können nun bereits zwei Datenpunkte auf dem Radarplot betrachtet werden, da diese beiden Eigenschaften bereits einmal in das Land oder aus dem Land hinaus geführt haben. Jetzt entscheidet der Spieler sich, nicht sofort weiter zu reisen, sondern möchte vorher erneut Polen ansehen.

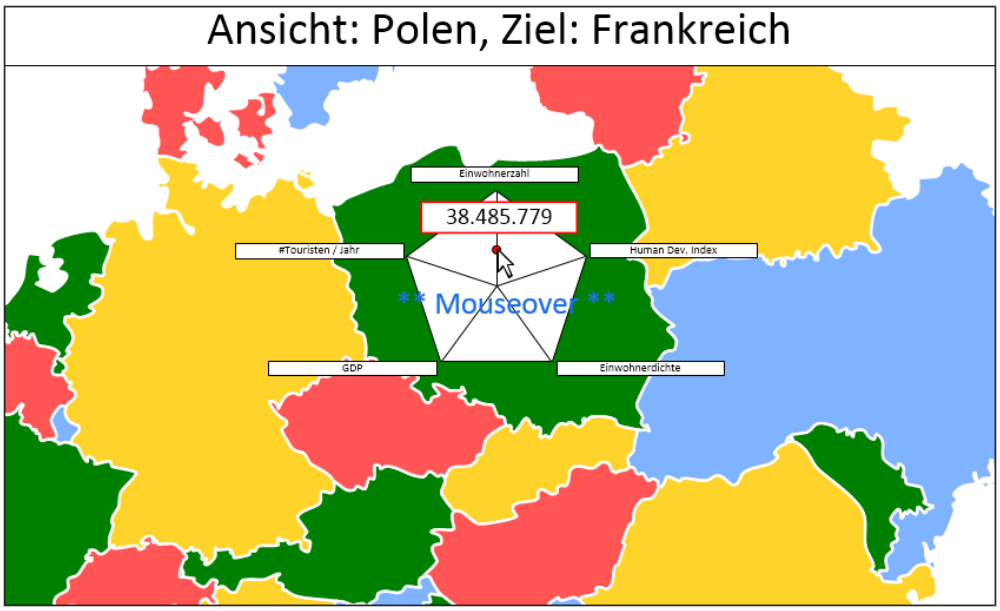

Abbildung 2.8: In Polen kann nun der Radarplot mit allen freigespielten Daten betrachtet werden. Dies ist in diesem Fall nur die Einwohnerzahl, da der Spieler bisher nur einmal in Polen war, und die Einwohnerzahl genutzt hat, um in das nächste Land zu reisen. Wenn er mit dem Mauszeiger über den Datenpunkt fährt, wird der konkrete Wert angezeigt.

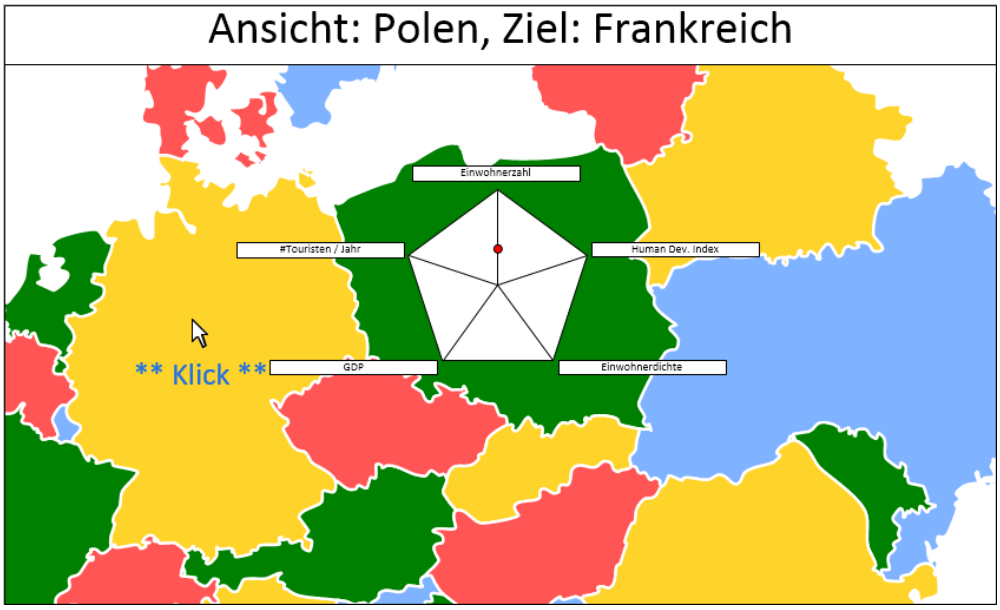

Abbildung 2.9: Der Spieler klickt nun auf seinen aktuellen Standort Deutschland um mit dem Spiel fortzufahren.

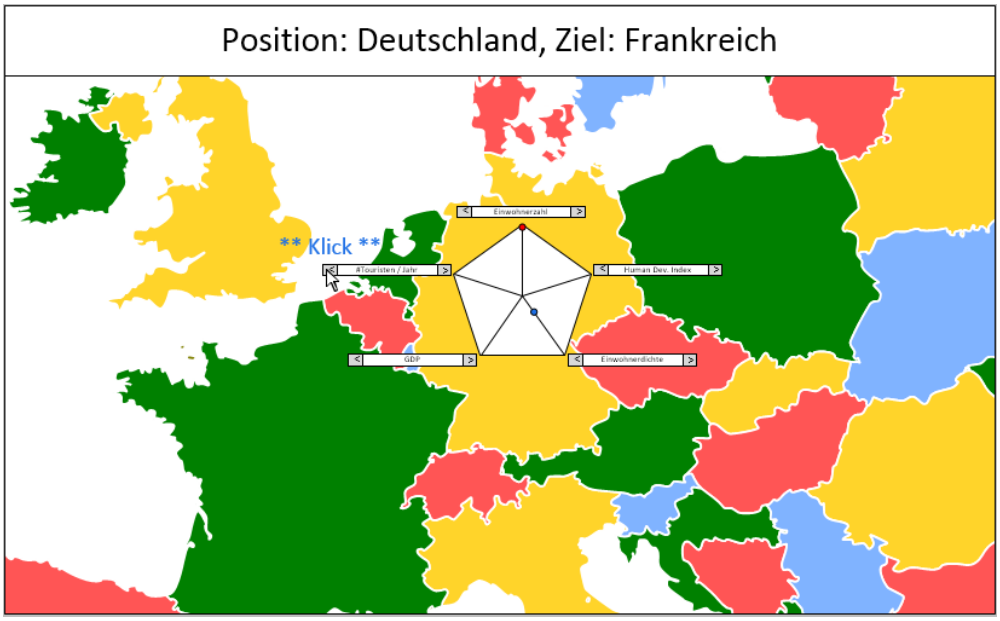

Abbildung 2.10: Zurück in Deutschland entscheidet der Spieler sich, die Anzahl der Touristen pro Jahr zu reduzieren.

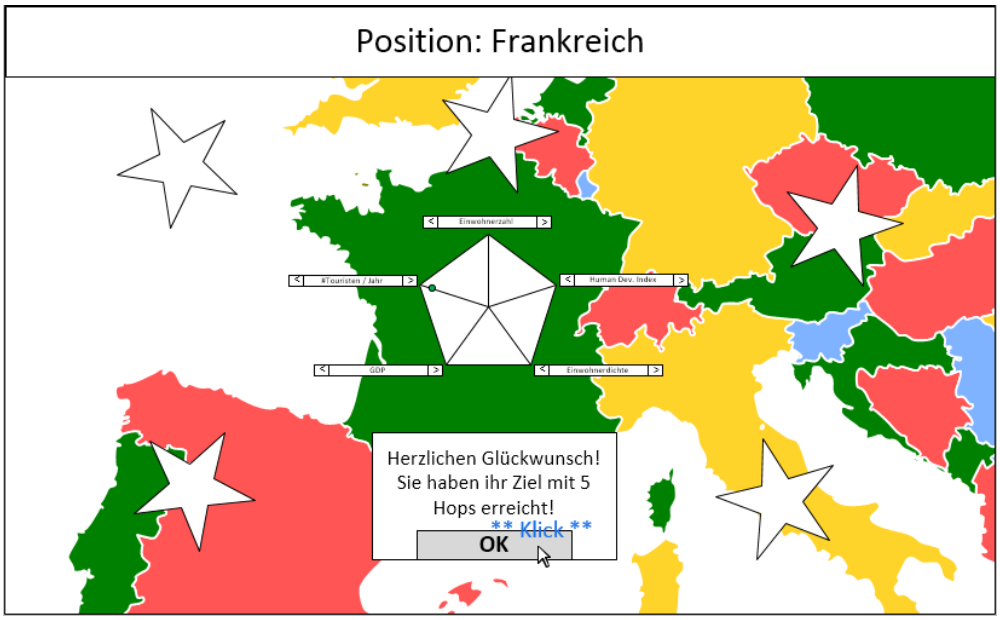

Abbildung 2.11: Der Spieler erreicht Frankreich und hat somit sein Ziel mit 5 Schritten erreicht.

2.2 Navigation Map

Das oben dargestellte Storyboard zeigt nur einen konkreten Durchlauf durch das Spiel, bei dem auch einige Optionen fehlen. Abbildung 2.12 zeigt deshalb alle Möglichkeiten durch das Spiel zu navigieren.

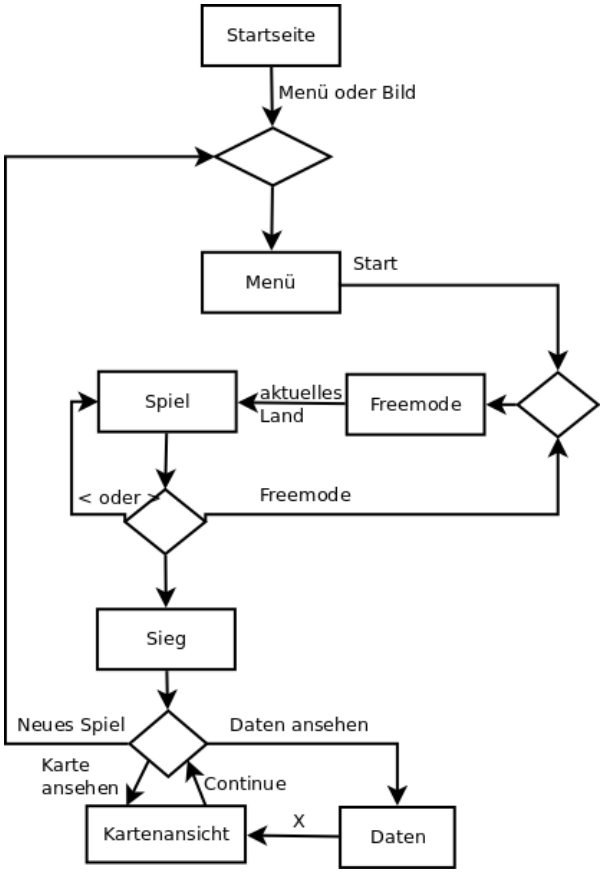

Abbildung 2.12: Navigationsmöglichkeiten in Statehopper

- Der Spieler beginnt auf der Startseite. Von dort aus gelangt er durch einen einfachen Klick auf das Menü. Hier kann der Name angegeben und die Eigenschaften ausgewählt werden. Außerdem besteht die Möglichkeit, das Tutorial zu aktivieren bzw. zu deaktivieren.

→ Anschließend landet der Spieler auf der eigentlichen Spielfläche im „Free Mode“. Das heißt, er kann sich frei bewegen und beliebige Länder ansehen.

→ Durch einen Doppelklick auf ein Land der Karte oder durch Klick auf eines der in der Menüleiste verlinkten Länder gelangt er in den richtigen Spielmodus.

↻ Nun kann der Spieler regulär spielen, wodurch er immer wieder in den Spielmodus kommt.

⇆ Alternativ kann er zwischen Spielmodus und Free Mode hin und her wechseln.

→ Wenn das Spiel gewonnen wurde erscheint ein Dialogfenster.

→ Von dort aus kann der Spieler entweder die Visualisierung der Daten als Balkendiagramme auswählen oder zurück auf die Karte gelangen.

⇆ Auch zwischen diesen beiden Punkten kann beliebig oft hin und her gewechselt werden, indem der Siegesdialog wieder aufgerufen wird.

→ Letztlich kann aus dem Siegesdialog auch ein neues Spiel begonnen werden.

3 Prototyp

3.1 Systemarchitektur

Auf höherer Ebene stand die Systemarchitektur relativ früh im Entwicklungsprozess fest, da durch die Entscheidung, ein Browser-Spiel mit einem *Grails* Server zu entwickeln, die grobe Struktur vorgegeben war 3.1. Das gesamte Projekt unterliegt also

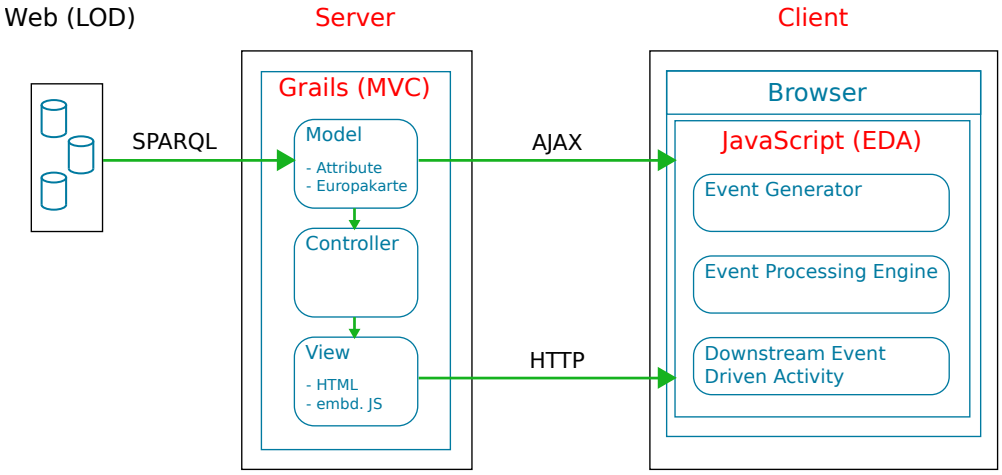

Abbildung 3.1: In Statehopper verwendete Architekturmuster

aufgrund der Art der Anwendung einer Client-Server Architektur. Das serverseitig eingesetzte *Grails-Framework* nutzt als Grundgerüst eine *Model-View-Controller* Architektur. Das *Model* beschreibt eine Liste von Länder-Objekten, die Daten bestimmter Attribute enthalten. Die Werte dieser Attribute können von Datenbanken mit *SPARQL-Endpoints* zur Verfügung gestellt werden. Außerdem ist eine Europakarte Teil des *Models*, die statisch als Datei auf dem Server vorliegt. Beim Starten des Servers wird lediglich eine Iteration des *MVC* ausgeführt, in welcher der *Controller* ein *HTML-Dokument* mit eingebettetem *JavaScript* erzeugt. Diese *View* kann über *HTTP* von *Clients* angefragt werden, sodass die in *JavaScript* umgesetzte Spiellogik im Browser ausgeführt wird. Auf der Seite des *Clients* liegt nun eine *Event-Driven-Architecture* vor, wobei der Browser die Funktion des *Event Generator* übernimmt indem er u. a. *Events* bei abgeschlossenem Laden des Dokuments oder Mausklicks erzeugt. *JQuery* übernimmt die Aufgabe des *Event Processing*, das die gewünschten *Events* identifiziert, die beispielsweise durch *onClick listener* spezifiziert wurden. *Downstream Event*

Driven Activities sind dann die Beschaffung der Attribute und der Karte über *AJAX* und ein Hop in ein anderes Land.

3.2 Entwurfsentscheidungen

Vor und während der Entwicklung mussten noch einige grundlegende Entscheidungen getroffen werden:

Karte Als Grundlage für das Spiel dient eine Karte. Die ganze Welt mit allen Ländern erschien als zu groß und damit zu komplex und langwierig für den Spielablauf. Die Weltkarte mit Kontinenten als einzelne Felder wiederum bietet zu wenig Möglichkeiten. Deshalb fiel die Wahl auf einen Kontinent und hier auf Europa, da dies derjenige ist, über den das meiste Wissen vorlag.

Beschränkungen der Karte Nach der Wahl des Kontinents mussten noch einige Länder ausgenommen werden. Dies sind zum einen alle Länder, die nur teilweise in Europa liegen, da hier die Daten schwierig zu beurteilen sind. Zum anderen aber auch alle sehr kleinen Länder, insbesondere Stadtstaaten. Bei diesen weichen die Daten oft sehr extrem in eine Richtung ab und sind daher nicht brauchbar. Außerdem haben diese Länder meist nur ein oder zwei Nachbarländer und sind daher schwierig zu erreichen bzw. zu verlassen.

Hop-Auswahl Grundsätzlich wird für den nächsten Hop das Nachbarland gewählt, dessen gewählte Eigenschaft in die gewählte Richtung minimal zum Ausgangsland abweicht. Hierbei können allerdings zwei Ausnahmen eintreten die behandelt werden müssen:

1. Mehrere Länder haben wegen gleicher Werte eine minimale Abweichung in diese Richtung. Dann wird zufällig eines dieser Länder ausgewählt.
2. Kein Land weicht in die gewählte Richtung ab. Dann erscheint eine Fehlermeldung (dies wird als Hop gezählt).

Auswahl Start und Ziel Eine weitere Frage war es, wie das Start- und das Zielland ausgewählt werden sollten, damit es nur zu spielbaren Kombinationen kommt. Hierfür gibt es mehrere Einschränkungen:

1. Start- und Zielland dürfen nicht benachbart sein, da das Spiel dann zu trivial wäre.
2. Es muss möglich sein, das Spiel zu beenden. Durch die unvorhersehbare Vielfalt der Attribute könnte es passieren, dass das Zielland nie zu erreichen ist. Um daher keine unmöglichen Kombinationen zu erstellen, wird nur das Startland ausgewürfelt und ein Weg zum Zielland simuliert. Der Pfad der Simulation hat eine zufällige Länge von 10 bis 15 Hops. Wenn Dieser nicht gefunden wird, geschieht die Auswahl des Startlandes erneut.

Zeitmodell Das Zeitmodell der Anwendung musste im Grunde genommen nicht wirklich festgelegt werden. Aus dem Aufbau des Spiels ergibt sich direkt, dass ein

Eventbasiertes Zeitmodell vorliegt, da keine feste zeitliche Abfolge bestimmt wird, sondern nur eine grobe Reihenfolge, die dann bei bestimmten Events (Klicks des Spielers) durchlaufen wird.

Raummodell Auch das Raummodell wurde nicht bewusst ausgewählt. Aus dem Umstand, dass die Spielkarte als SVG vorliegt ergibt sich direkt, dass auch das SVG-Ortsmodell genutzt werden muss. Das bedeutet, dass ein Koordinatensystem vorliegt, dessen Ursprung in der oberen linken Ecke des Bildschirms liegt. Die Koordinaten werden als Fließkommazahlen abgebildet, was die Grundlage dafür ist, das Objekte stufenlos skalierbar sind. Außerdem liegt eine Hierarchie von Objekten vor, welche relativ zu einander positioniert werden.

Fehlerbehandlung Im Menü wird überprüft, ob ein Name und mindestens fünf verschiedene Attribute ausgewählt wurden. Falls die nicht getan wurde, werden die entsprechenden Bereiche hervorgehoben. Als Methode des *Error Avoiding Design* wurde implementiert, dass der Start-Knopf auch nur dann gedrückt werden kann. Außerdem wird man auf das Menü zurückgeleitet, falls man per Browserkonsole den Knopf aktiviert, obwohl man keine fünf Attribute aktiviert hatte.

3.3 Werkzeuge

Für die Entwicklung des Prototypen spielten einige Werkzeuge eine Wichtige Rolle. Neben den Programmiersprachen Java, JavaScript und Groovy, sowie den Formaten HTML und SVG kamen die folgenden Frameworks und Bibliotheken zum Einsatz.

3.3.1 Grails

Grails ist ein Framework zur Webentwicklung, das die Model-View-Controller Architektur nutzt. Als Programmiersprache in dem Grails Projekt kann Groovy oder Java benutzt werden [gra15].

Grails wurde hauptsächlich genutzt um auf einfache und schnelle Art einen Webserver zu erstellen, der das Hauptspiel mit Daten und Medien-Assets versorgt. Das Hauptspiel ist in JavaScript erstellt und kann per AJAX auf Controller Methoden des Grails Programm zugreifen, die wiederum z.B. die Daten für die Länder aus dem Web abfragen.

3.3.2 jQuery

jQuery ist eine JavaScript Bibliothek, die als Schnittstelle für den einfachen Zugriff auf das DOM eines HTML-Dokuments und dessen Manipulation genutzt werden kann [jqu15a].

Die Wahl auf jQuery ist einfach begründet: Die komplexe Interaktion mit den einzelnen Ländern und der Karte wr̈e mit purem JavaScript ein sehr hoher nicht gerechtfertigter Mehraufwand gewesen.

3.3.3 jQueryUI

jQueryUI ist ebenfalls eine JavaScript Bibliothek, die jQuery als Basis nutzt. Wie der Name suggeriert, können leicht Elemente der Benutzerinteraktion erstellt werden [jqu15b].

Alle Dialoge sind mit jQueryUI erstellt. Dazu zählen Start- und Endbildschirm, die Meldung, dass ein Hop nicht möglich ist und das Fenster der Balkendiagramme.

3.3.4 Bootstrap

Bootstrap ist ein Framework zur Erstellung von Oberflächen von Webanwendungen. Es nutzt dafür HTML, CSS und JavaScript. Der besondere Fokus von Bootstrap liegt im Responsive Web Design. Es ist möglich, sehr schnell Webseiten zu erstellen, die auch von mobilen Geräten problemlos angezeigt werden können und in dem Fall auch andere Layouts bieten können [boo15a].

Es wurde genutzt, um die Navigationsleiten oberhalb und unterhalb der Karte auf einfach Art zu implementieren.

3.3.5 Bootstrap-Switch

Boostrap-Switch ist als Komponente eine Erweiterung für Bootstrap um Checkboxen und Radiobuttons visuell an das Bootstrapdesign anzupassen [boo15b].

Es wurde für alle Optionen im Menü und den Schaltern im Spiel genutzt.

3.3.6 ChartJS

Chart.js ist eine JavaScript-Bibliothek zum einfachen erstellen verschiedener Diagramme. Es werden unter anderem Balken- und Tortendiagramme, aber auch die Abbildung von Funktionen in Koordinatensystemen unterstützt. Die Grundlage für alle Diagramme bildet das HTML5-canvas-Element, welches in allen modernen Browsern angezeigt werden kann [cha15].

ChartJS wurde in diesem Prototypen für die Balkendiagramme verwendet, welche der Spieler sich nach dem Sieg ansehen kann.

3.3.7 SVG Pan & Zoom

SVG Pan & Zoom ist eine JavaScript Bibliothek, die das vergrößern und verkleinern, sowie verschieben von SVG Bilder ermöglicht. Die geschieht, indem das Bild in ein neues SVG Bild als SVG Group eingefügt wird und mittels Matrixtransformationen von SVG verändert wird [svg15b].

Der Spieler soll sich in der Karte frei bewegen, um die gewohnte Interaktion mit Karten im Web zu gewährleisten. Mit SVG Pan & Zoom ist dieses Verhalten umgesetzt worden.

Bezeichnung	Quelle
Europa Karte	Wikimedia Commons
Logo Grails	Grails
Logo Groovy	Blogspot
Logo JavaScript	alex-arriaga.com
Logo jQuery	Blogspot
Logo jQueryUI	gregfranko.com
Logo Bootstrap	W3Schools
Logo ChartJS	jimsider.com
Logo SVG	balasubramanyamlanka.com
Logo SVG Pan & Zoom	GitHub
Logo Snap SVG	Snap SVG
Logo Semantic Web	W3C

Tabelle 3.1: Im Prototypen genutzte Medienassets

3.3.8 Snap.svg

SVG Snap ist eine JavaScript Bibliothek, die einfachen Zugriff auf den DOM von SVG Bildern anbietet, wie jQuery für den HTML-DOM. Es können nicht nur SVG Elemente im HTML-DOM erstellt werden, sondern auch bestehende Bilder handlich verändert werden [svg15a].

Snap wurde in Statehopper dafür genutzt, den Radarplot als Datenanzeige der Länder zu erstellen, die Länder farblich hervorzuheben oder auf einfache Art eine Bounding-Box eines Landes zu berechnen.

3.4 Genutzte Medien

Für den Prototypen wurden in erster Linie Bilder als Medienassets verwendet. Diese können Tabelle 3.1 entnommen werden. Darüber hinaus wurden einige Daten verwendet, welche in Form von Text auch ein Medium darstellen. Die Daten stammen zum einen aus dem von der CIA herausgegebenen „The World Factbook“ und zum anderen von DBpedia. Tabelle 3.2 listet alle Eigenschaften, die diesen beiden Quellen entnommen wurden. Von DBpedia wurden außerdem einige Texte für die Beschreibung der Eigenschaften im Menü entnommen.

Sämtliche Daten wurden mit Hilfe von SPARQL abgerufen. DBpedia stellt hierfür einen eigenen Endpunkt zur Verfügung. Um auf die Daten des Factbooks zuzugreifen wurde ein Endpunkt der Universität Mannheim verwendet. Grundsätzlich sind auch andere Datenquellen und andere Arten der Anbindung problemlos realisierbar. Die Begrenzung auf SPARQL erfolgte auf Grund der Entscheidung, ausschließlich Linked Open Data zu verwenden, was in der Aufgabenstellung auch durchaus gewünscht war. Die Wahl der Datenquellen fiel auf DBpedia, da dies das bekannteste und Linked Open Data Projekt ist und auf das Factbook, da hier, auf Grund des Herausgebers, qualitativ hochwertige Daten zu erwarten waren. Nach mehr Datenquellen wurde nicht recher-

The World Factbook	DBpedia
Anzahl der Flughäfen	BIP
Fläche gesamt	Gini Koeffizient
Landfläche	Human Develeopment Index
Wasserfläche	
Höchster Punkt	
Anzahl der Internetnutzer	
Bevölkerung	
Gesamtlänge aller Straßen	
Anzahl der Telefonanschlüsse	
Number of Mobile phones	

Tabelle 3.2: Beispielhafte Kategorien von Eigenschaften mit deren Datenquellen

chiert, da die vorliegenden Eigenschaften für eine sinnvolle Erstellung eines Prototypen völlig ausreichen.

3.5 Entwicklungsprozess

Die Entwicklung des Projekts wurde von der Konzeption über die eigentliche Implementierung bis zur Dokumentation unter den drei Mitgliedern aufgeteilt. Es gab mindestens einen Nachmittag in der Woche, an dem zusammen in einem Raum gearbeitet wurde. Hierbei würde insbesondere bei Problemen häufiger die Technik des Pair Programmings genutzt.

Die Reihenfolge der Implementierung erfolgte nach keinem strikten „top down“ oder „bottom up“-Ansatz, da in einigen Teilen von beiden Seiten aus parallel gearbeitet werden konnte und das Ziel von Beginn an ein vertikaler und kein horizontaler Prototyp war. Angefangen wurde mit dem Aufsetzen des Grails-Projekts und der zeitgleichen Entwicklung des Radarplots. Daraufhin folgten parallel die Einbindung der Karte und die Datenanbindung. Danach wurde die eigentliche Spiellogik implementiert und währenddessen die Benutzeroberfläche weiterentwickelt. Als letztes entstanden die Daten- und die Routenvisualisierung am Ende, sowie die Startseite.

Ein wichtiges Werkzeug bei der Entwicklung war die Entwicklungsumgebung IntelliJ IDEA, insbesondere, da diese nativ eine umfangreiche Unterstützung für die Arbeit mit Grails liefert [int15].

3.6 Systemanforderungen

Der Prototyp ist in der folgenden Umgebung getestet:

Betriebssystem Windows/Linux

Application Server Apache Tomcat 8

Browser Firefox Version 39.0, Chrome Version 43.0

4 Fazit

Dieses Kapitel fasst kurz zusammen, was mit dem zuvor beschriebenen Prototypen erreicht wurde und inwiefern er die gesetzten Anforderungen erfüllt. Außerdem wird ein Ausblick gegeben, welche Erweiterungen für das Spiel noch denkbar wären.

4.1 Zusammenfassung

Im Rahmen des Moduls „Theoretical Foundations and Applications of Interactive Multimedia“ wurde ein funktionierender vertikaler Prototyp des Spiels Statehopper entwickelt.

4.1.1 Anforderungen

Im Großen und Ganzen erfüllt der Prototyp alle Anforderungen. Die grundlegenden Anforderungen waren es, eine interaktive, multimediale Anwendung zu entwickeln, die Linked Data visualisiert. Dies ist geschehen. Der Prototyp ist auf mehrere arten graphisch interaktiv. Es kommen diskrete Medien (Texte und Bilder) und das kontinuierliche Medium Animation zum Einsatz. Die Anwendung visualisiert auf zwei unterschiedliche Arten (Radarplot und Balkendiagramme) Linked Open Data aus zwei verschiedenen Quellen.

Aber auch die im Rahmen der Konzeption entstandenen expliziten Anforderungen werden weitestgehend erfüllt. Der Spielablauf ähnelt dem vorher entworfenen Skript sehr stark. Einzig ein Highscore wurde nicht implementiert, da sich dieser als verhältnismäßig wenig sinnvoll herausstellte. Dafür wurden aber einige zusätzliche Funktionen, wie die Visualisierung des zurückgelegten Weges und das Tutorial, implementiert.

4.1.2 Probleme und Einschränkungen

Während der Entwicklung kam es zu einigen Problemen, die kleinere Einschränkungen mit sich zogen. Insbesondere mussten einzelne Länder aus dem Spiel ausgeschlossen werden, da die Datenquellen nur sehr wenige Daten zu diesen bereithielten.

4.2 Ausblick

Der entwickelte Prototyp bietet an verschiedenen Stellen Raum für Erweiterungen.

4.2.1 Verschiedene Level

Eine Idee ist es, verschiedene Level zur Auswahl zu stellen, welche auf unterschiedlichen Karten Spielen. Beispielsweise eine Karte der USA bietet vermutlich eine gute Grundlage und auch ausreichend viel Datenmaterial.

4.2.2 Weitere Attribute

Generell könnten zusätzliche Attribute eingebunden werden. Hierzu kann auf weitere Datenquellen zugegriffen werden. Unter Umständen müsste dann allerdings von der Abfrage via SPARQL abgewichen und auf andere Techniken zurückgegriffen werden. Bei einer größeren Anzahl von Attributen könnten diese in mehrere thematische Kategorien eingeteilt werden, welche man den Benutzer dann auch komplett an- bzw. abwählen lassen könnte.

4.2.3 Schwierigkeitsgrade

Die Einführung verschiedener Schwierigkeitsgrade könnte über Zeitlimits erreicht werden. Beispielsweise könnte der Benutzer in der einfachen Version beliebig viel Zeit pro Hop haben (wie aktuell vorgesehen), in einer mittelschweren Version nur 60 Sekunden und im schwierigsten Grad nur 30 Sekunden. Bei einer Zeitüberschreitung könnte dann ein „Straf-Hop“ hinzugefügt werden, oder der Spieler in ein anderes Land versetzt werden, welches weiter von seinem Zielort entfernt ist.

4.2.4 Achievements

Eine Möglichkeit, den Spielanreiz zu erhöhen und vor allem aufrecht zu erhalten wären Erfolge, die der Spieler sammeln kann. Denkbar wären z.B.

- Alle Länder bereist
- Alle Eigenschaften verwendet
- Alle Daten zu Land xy freigespielt

Für diese Erfolge könnte der Spieler dann wiederum Punkte bekommen mit denen dann eventuell neue Features (wie die oben beschriebenen Level oder Schwierigkeitsgrade) freigespielt werden könnten.

4.2.5 Highscore

Eine andere Methode, den Spielanreiz zu steigern wären Highscores, in denen lokal, oder sogar global die Spielergebnisse gespeichert und vergleichen werden können. Problematisch ist hierbei die Metrik, da die Anzahl der Hops bei verschiedenen Paarungen von Start- und Zielland nur wenig aussagekräftig ist.

Literaturverzeichnis

[boo15a] Bootstrap. http://getbootstrap.com/, 2015.

[boo15b] Bootstrap Switch. http://www.bootstrap-switch.org/, 2015.

[cha15] ChartJS. http://www.chartjs.org/, 2015.

[gra15] GRAILS. https://grails.org/, 2015.

[int15] IntelliJ IDEA. https://www.jetbrains.com/idea/, 2015.

[jqu15a] jQuery. https://jquery.com/, 2015.

[jqu15b] jQueryUI. https://jqueryui.com/, 2015.

[svg15a] Snap. http://snapsvg.io/, 2015.

[svg15b] SVG Pan & Zoom. https://github.com/ariutta/svg-pan-zoom, 2015.