

Sudoku-Helper

Studienarbeit T3 3101

des Studienganges IN

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

Luca Rohmann und Noah Hauke

10.06.2022

Bearbeitungszeitraum

300 Stunden (pro Person)

Matrikelnummern, Kurs

3819525, 7750470, INF 19IN

Betreuer

Sebastian Trost

Erklärung

Wir erklären hiermit ehrenwörtlich, dass wir die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Aus den benutzten Quellen, direkt oder indirekt, übernommene Gedanken haben wir als solche kenntlich gemacht. Diese Arbeit wurde bisher in gleicher oder ähnlicher Form oder auszugsweise noch keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Esslingen am Neckar, 10.06.2022

Ort, Datum



Unterschrift

Göppingen, 10.06.2022

Ort, Datum



Unterschrift

Zusammenfassung

Bei Sudokus handelt es sich um eine sehr beliebte Rätselform. Das Ziel ist es eine Neun mal Neun Feld, welches zu Beginn einige Zahlen enthält, so mit den Zahlen eins bis neun aufzufüllen, dass am Ende in jeder Zeile, jeder Spalte und jedem Block jede Zahl genau einmal vorkommt.

Jedoch kann sich das Lösen dieses im ersten Moment simpel klingenden Rätsels als sehr schwierig erweisen, da es gar nicht so einfach ist die Zahlen entsprechend zu verteilen. Erschwerend kommt hinzu, dass ein richtiges Sudoku exakt eine Lösung hat, die es zu finden gilt.

Dies hat zur Folge, dass der Versuch diese Rätsel zu Lösen mitunter sehr frustrierend werden kann und der Wunsch nach einer Hilfe auftaucht. Klassische Sudoku Löser, von denen es online zahlreiche gibt zeigen dabei aber nur die Lösung an, anstatt Nutzer*innen bei der Hand zu nehmen und sie Schritt für Schritt durch den Lösungsvorgang zu führen. Dies hat u. a. den Hintergrund dass oft einfach ein Bruteforce Algorithmus eingesetzt wird, den Menschen natürlich nicht von Hand durchführen können, da es hierzu zu viele Möglichkeiten gibt.

An dieser Stelle setzt diese Studienarbeit an, das Ziel ist es hier einen Helfer zu entwickeln, welcher Schritt für Schritt zu der Lösung des Sudokus hinführt und dabei Algorithmen verwendet, welche mit Hilfe von Logik das Sudoku lösen.

Dabei sollen die einzelnen Algorithmen so visualisiert werden, dass die Hinweise die geliefert werden Schritt für Schritt deutlicher werden. Außerdem sollen die Algorithmen so erklärt werden, dass Interessenten in der Lage sind sie nachzuvollziehen und das Prinzip dahinter zu verstehen.

In diesem Bericht ist die Implementierung dieser Anwendung in Form einer Webseite und unter der Verwendung von Python mit dem Framework Django erläutert.

Abstract

Sudoku puzzles are a very popular form of puzzle. The goal is to fill up a nine by nine field, which contains some given numbers at the beginning, with the numbers one to nine in such a way that at the end in each row, each column and each block each number appears exactly once.

However, solving this puzzle, which sounds simple at first, can prove to be very difficult, since it is not so easy to distribute the numbers accordingly. To make matters worse, a correct Sudoku has exactly one solution to find.

As a result, trying to solve these puzzles can sometimes become very frustrating and the desire for help arises. Classic Sudoku solvers, of which there are many online, only show the solution instead of taking users by the hand and guiding them step by step through the solution process. This is due to the fact that often a brute force algorithm is used, which humans can't do by hand, because there are too many possibilities.

The goal of this work is to develop a helper, which leads step by step to the solution of the Sudoku and uses algorithms, which solve the Sudoku with the help of logic.

The algorithms should be visualized in such a way that the hints provided become clearer step by step. In addition, the algorithms should be explained in such a way that the users are able to follow them and understand the principle behind them.

In this report the implementation of this application is explained in the form of a web page and using Python with the framework Django.

Inhaltsverzeichnis

Erklärung	i
Inhaltsverzeichnis	iv
Abbildungsverzeichnis	vi
Tabellenverzeichnis	vii
Abkürzungsverzeichnis	vii
1 Aufgabenstellung	1
2 Theorie	1
2.1 Definitionen	1
2.1.1 Einheiten	1
2.1.2 Kandidaten	2
2.2 Echtes Sudoku	2
2.3 Algorithmen	2
2.3.1 Einfacher Lösungsalgorithmus Beispiel Block-Reihen-Check	3
2.3.2 Mittelschwerer Lösungsalgorithmus Beispiel Wolkenkratzer	4
2.3.3 Schwerer Lösungsalgorithmus Beispiel XY-Kette	5
2.4 Tests	6
2.5 Technologien	7
2.5.1 Backend	8
2.5.2 Frontend	10
2.5.3 Tests	13
3 Implementierung	14
3.1 Versionsverwaltung	14
3.2 Architekturdesign	14
3.2.1 Sudoku	15
3.2.2 Validierung	15
3.2.3 Algorithmen	15
3.2.4 Django Server und Zusätze	15
3.2.5 Frontend-Tests	16
3.2.6 Development Tools	16
3.2.7 Ordner- und Modulstruktur	16
3.3 Backend Implementierung	17
3.3.1 Reaktion auf Fehleingaben	17
3.3.2 Backtracking	18

3.3.3	Algorithmen.....	19
3.4	Datenfluss	23
3.4.1	Frontend zu Backend.....	23
3.4.2	Algorithmus Ergebnisse ans Frontend	24
3.5	Frontend.....	24
3.5.1	Arbeitsschritte	24
3.5.2	Allgemeines Design	25
3.5.3	Seiten.....	26
3.6	Tests.....	36
3.6.1	Backend.....	36
3.6.2	Frontend.....	37
3.6.3	Ergebnisse	38
4	Fazit	39
5	Literaturverzeichnis	40

Abbildungsverzeichnis

Abbildung 1: Leeres Sudoku.....	2
Abbildung 2: Sudoku mit gegebenen Zahlen	2
Abbildung 3: Beispiel Block-Reihen-Check.....	4
Abbildung 4: Beispiel Wolkenkratzer	4
Abbildung 5: Beispiel XY-Kette.....	5
Abbildung 6: Hinzufügen einer CSS-Klasse mit JavaScript (Erstellt mit: [15])	12
Abbildung 7: Hinzufügen einer CSS-Klasse mit JQuery (Erstellt mit: [15])	13
Abbildung 8: Verteiltes Versionsverwaltung.....	14
Abbildung 9: Beispiel Versionen	14
Abbildung 10: Ordner- und Modulstruktur der Anwendung.....	17
Abbildung 11: Backtracking Struktogramm.....	18
Abbildung 12: Block_Reihen_Check Struktogramm	19
Abbildung 13: Rückgabe Block-Reihen-Check (Erstellt mit: [15])	20
Abbildung 14: Rückgabe Wolkenkratzer (Erstellt mit: [15]).....	21
Abbildung 15: Wolkenkratzer Struktogramm	21
Abbildung 16: XY-Kette Struktogramm	22
Abbildung 17: vereinfachtes HTML für ein Sudoku-Eingabefeld (Erstellt mit: [15])...	23
Abbildung 18: vereinfachtes HTML für ein Eingabefeld, in dem die Kandidaten für ein Feld zwischengespeichert werden (Erstellt mit [15]).....	23
Abbildung 19: Mockup für die Homepage (Erstellt mit Figma).....	25
Abbildung 20: Implementierung der Homepage	25
Abbildung 21: Anordnung der Kandidaten innerhalb eines Sudoku Feldes	26
Abbildung 22: Homepage	27
Abbildung 23: Bestätigungsseite, dass eingetragenes Sudoku gültig ist	28
Abbildung 24: Seite zur Algorithmus Erläuterung, Schritt 1	29
Abbildung 25: Seite zur Algorithmus Erläuterung, Schritt 2	30
Abbildung 26: Seite zur Algorithmus Erläuterung, Schritt 3	30
Abbildung 27: Seite zur Algorithmus Erläuterung, Schritt 4	31
Abbildung 28: Fenster zur genaueren Erklärung der Algorithmen	32
Abbildung 29: Seite die den Stand des Sudokus nach der Anwendung eines Algorithmus anzeigt	33
Abbildung 30: Seite die anzeigt, dass das Sudoku vollständig gelöst ist	33
Abbildung 31: Seite, wenn das gegeben Sudoku nicht mit den Algorithmen gelöst werden kann	34
Abbildung 32: Diagramm für den Ablauf der Webseiten	35
Abbildung 33: Dev-Tools Startseite	36
Abbildung 34: Dev-Tools Sudoku Vorlagen	36

Tabellenverzeichnis

Tabelle 1: Arten von Software-Tests (vgl. [2]).....	6
Tabelle 2: Verwendete Testarten.....	7

Abkürzungsverzeichnis

ASP	<i>Active Server Pages</i>
CSS	<i>Cascading Style Sheets</i>
DOM	<i>Document Object Model</i>
HTML.....	<i>Hypertext Markup Language</i>
JSON	<i>JavaScript Object Notation</i>

1 Aufgabenstellung

Die Aufgabe die im Rahmen dieser Studienarbeit erfüllt werden soll, ist es einen so genannten „Sudoku Helper“ zu entwickeln. Gemeint ist dabei eine Anwendung, die Sudoku Spieler*innen bei der Lösung von schwierigen Sudokus unterstützt. Dabei soll das Sudoku allerdings nicht einfach automatisch gelöst werden, also es soll nicht direkt die Komplettlösung angezeigt werden. Stattdessen soll eine Schritt-für-Schritt Anleitung geboten werden. Dabei soll immer benannt werden, welche Lösungsstrategie angewandt wird und es soll erklärt werden, wie die Strategie funktioniert.

Der genaue Ablauf sieht dabei wie folgt aus:

- Öffnen der Webseite und eingeben der bekannten Werte
- Validierung, ob es sich um ein echtes Sudoku handelt
- Möglichkeit Kandidaten einblenden zu lassen
- Anwenden der Strategien
 - Hilfe 1: Benennen der anzuwendenden Strategie, sowie hervorheben der Bereiche auf die die Strategie angewendet wird
 - Hilfe 2: Markieren der konkreten Zellen und Kandidaten auf die die Strategie angewandt wird
 - Hilfe 3: Erklärung, warum bestimmte Kandidaten laut der Strategie gestrichen werden können
 - Hilfe 4: Kandidaten löschen bzw. Änderungen anwenden

Wichtig ist hierbei, dass das Programm in der Lage sein muss bei jedem lösbaaren Sudoku zu unterstützen, somit müssen laut Anforderungsdokument ca. 25 Lösungsstrategien implementiert werden.

Für die Implementierung werden zwei Möglichkeiten angeboten, so kann entweder eine Android App entwickelt werden, oder es kann eine Web-Anwendung basierend auf dem Django Framework entwickelt werden.

Dabei muss das Frontend, also die Visualisierung sowohl auf kleinen Bildschirmen wie denen von Smartphones, als auch auf großen Bildschirmen wie denen von Laptops eine benutzeroptimierte Darstellung liefern.

2 Theorie

2.1 Definitionen

Zuerst sollen einige Begriffe definiert werden, welche bei der Arbeit mit Sudokus auftreten.

2.1.1 Einheiten

Als Einheiten werden im Verlauf dieses Berichtes Mengen von Feldern beschrieben, in denen eine Zahl nur einmal auftauchen darf. Dabei gibt es drei Arten von Einheiten, Zeilen und Spalten, welche selbsterklärend sein sollten, sowie Blöcke, also die Verbünde aus 3x3 Feldern, welche in grafischen Darstellungen häufig durch dickere Linien voneinander abgegrenzt sind.

2.1.2 Kandidaten

Als Kandidaten werden die Werte bezeichnet, welche in ein Feld, welches noch keinen Wert hat, eingetragen werden können. Dabei besitzt jedes Feld eigene Kandidaten.

2.2 Echtes Sudoku

Das Standard-Sudoku besteht aus 81 Felder, die in 3x3 Blöcke unterteilt werden, diese Blöcke sind in jeweils 3x3 Felder unterteilt. Dadurch werden die 81 Felder auf neun Blöcke, neun Spalten und neun Reihen verteilt, wie in Abbildung 1 **Fehler! Verweisquelle konnte nicht gefunden werden.** zu erkennen ist. Zu Beginn eines Rätsels sind bereits einige Zahlen gegeben, siehe Abbildung 2. Diese Zahlen müssen so verteilt werden, dass für das ausgefüllte Sudoku nur eine einzige Lösung entsteht. Sobald es mehr als eine Lösung gibt, handelt es sich um kein echtes Sudoku mehr.

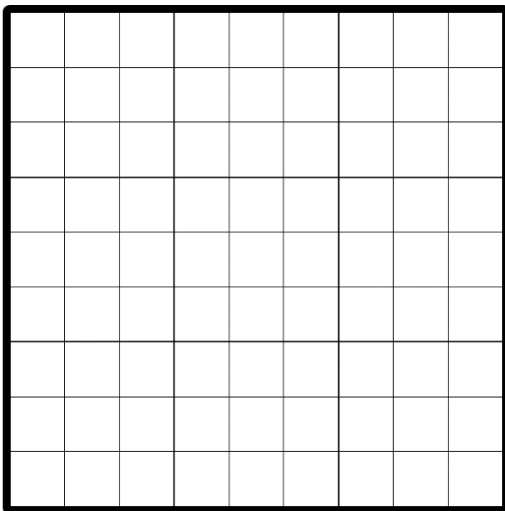


Abbildung 1: Leeres Sudoku

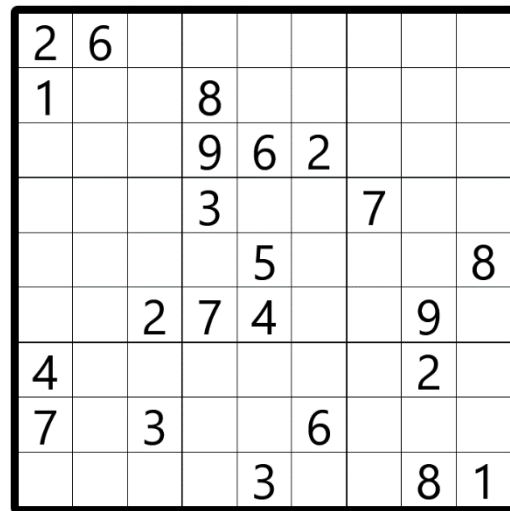


Abbildung 2: Sudoku mit gegebenen Zahlen

2.3 Algorithmen

In den Algorithmen werden die Lösungsstrategien realisiert. Während einfache Sudokus meist durch intuitives und konzentriertes Absuchen gelöst werden können. Einige der einfachen Lösungsstrategien werden hierbei intuitiv und unbewusst eingesetzt. Bei kniffligen Sudokus müssen Notizen in Form von Kandidaten gemacht und sehr komplexe Lösungsansätze verwendet werden. Diese Strategien verständlich zu erklären, ist Teil dieses Projektes. (vgl. [1])

Im Folgenden sind die implementierten Lösungsstrategien nach ihrer Komplexität kategorisiert:

Einfache Lösungsstrategien:

- Nackter Single
- Versteckter Single
- Nacktes Paar
- Verstecktes Paar
- Nackter Dreier
- Versteckter Dreier
- Nackter Vierer

- Versteckter Vierer
- Reihen Block Check
- Block Reihen Check

Mittelschwere Lösungsstrategien:

- X-Wing
- Steinbutt
- Drittes Auge
- Wolkenkratzer
- Schwertfisch
- Drache
- Verbotenes Viereck Typ1/Typ2/Typ4

Schwere Lösungsstrategien:

- XY-Wing
- XYZ-Wing
- X-Kette
- XY-Kette
- Schwertfisch mit Flosse
- W-Wing
- Einfacher Lösungsalgorithmus Beispiel Block-Reihen-Check

2.3.1 Einfacher Lösungsalgorithmus Beispiel Block-Reihen-Check

Einer der einfacheren Lösungsalgorithmen ist der Block-Reihen-Check, dieser Lösungsalgorithmus ist sowohl für einfache als auch für schwerere Sudokus relevant.

Die Regel besagt, befinden sich in einem Block alle Kandidaten eines Wertes in derselben Reihe, so kann aus den restlichen Feldern dieser Reihe Kandidaten dieses Wertes eliminiert werden.

In dem Beispiel, welches man in Abbildung 3 zu sehen ist, befinden sich in dem Block 1 alle Kandidaten des Wertes 8 in der Reihe 1, deshalb muss zwingend der Wert in den Feldern R1S1 oder R1S3 sein. Das hat zu Folge, dass aus den restlichen Feldern der Reihe 1 der Wert 8 aus den Kandidaten eliminiert werden kann. (vgl. [1])

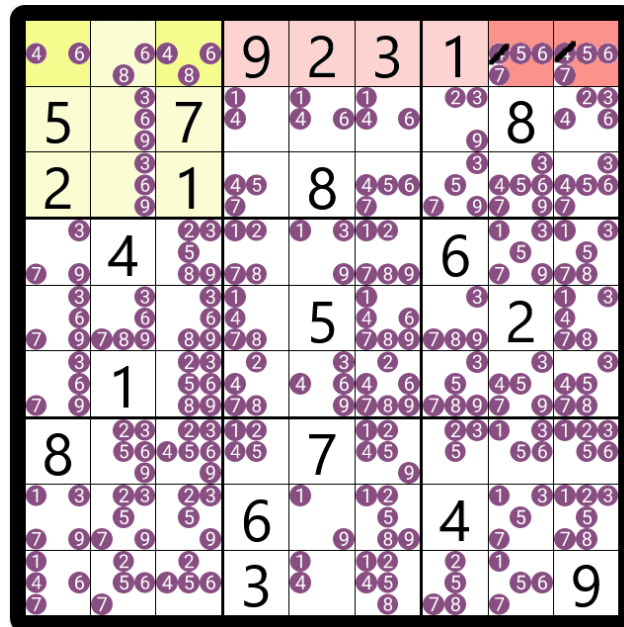


Abbildung 3: Beispiel Block-Reihen-Check

2.3.2 Mittelschwerer Lösungsalgorithmus Beispiel Wolkenkratzer

Der Wolkenkratzer besteht aus zwei Merkmalen. Merkmal eins ist, dass sich in zwei Spalten jeweils genau zweimal derselbe Kandidat steht. Das zweite Merkmal ist, dass ein Kandidat einer Spalte mit einem der Kandidaten der anderen Spalte auf derselben Höhe liegt. Nun kann aus dem gemeinsamen Einflussbereich der anderen beiden Kandidaten der Wert eliminiert werden.

In dem Beispiel, welches in Abbildung 4 zu sehen ist, kommt in den gelb markierten Spalten der Kandidat 2 nur zweimal vor. Diese beide Kandidatenpaare haben jeweils ein Feld auf derselben Höhe in Reihe 9. Nun kann aus dem gemeinsamen Einflussbereich der anderen Felder der Paare, hier schwarz umrandet, der Wert 8 aus den Kandidaten eliminiert werden. (vgl. [1])

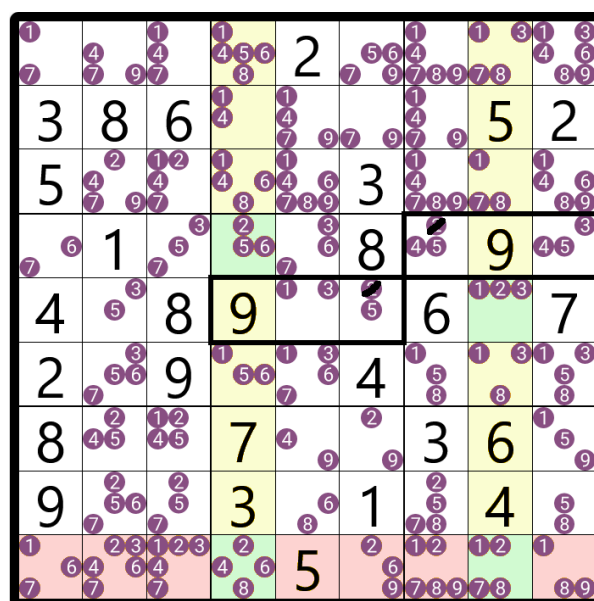


Abbildung 4: Beispiel Wolkenkratzer

2.3.3 Schwerer Lösungsalgorithmus Beispiel XY-Kette

Bei der XY-Kette werden zwei Ketten gebildet. Diese beide Ketten starten im selben Feld.

Eine Kette kann folgendermaßen gebildet werden. Alle Felder in einer Kette dürfen nur zwei Kandidaten enthalten. Es wird bei dem Startfeld von einem Kandidaten ausgegangen und wenn in einem Einflussbereich dieses Felds ein Feld mit zwei Kandidaten existiert, in dem einer der Kandidaten der andere Kandidat des Startfelds ist, wird dieser der Kette hinzugefügt. Diese Kette wird weiter gebildet, in dem man das Gleiche mit diesem Feld macht.

In diesem Lösungsalgorithmus beginnt eine Kette mit dem einen Kandidaten des Startfelds und die andere Kette mit dem anderen Kandidaten. Enden nun beide Enden der Kette mit dem gleichen Kandidaten, so kann aus dem gemeinsamen Einflussbereich dieser beiden Endfelder dieser Kandidat eliminiert werden.

In dem Beispiel, welches in Abbildung 5 zu sehen ist, ist das Startfeld grün markiert. Die erste Kette nach links beginnt mit dem Kandidat 6 im Startfeld, deshalb endet diese Kette im Feld R2S1 mit dem Kandidaten 1. Die zweite Kette nach rechts beginnt mit dem Kandidat 9 in dem Startfeld, über das Feld R1S9 mit dem Kandidaten 9 und dem Feld R1S4 mit dem Kandidaten 2, wird das Ende der Kette in dem Feld R1S6 mit dem Kandidaten 1 bestimmt. Nun kann in dem gemeinsamen Einflussbereich, hier schwarz umrandet, der Kandidat 1 aus den Feldern eliminiert werden. (vgl. [1])

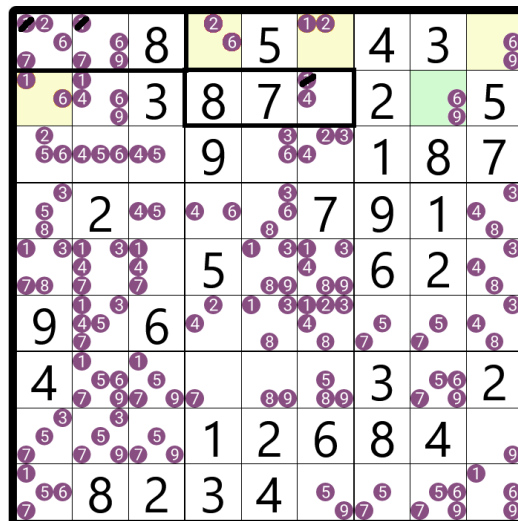


Abbildung 5: Beispiel XY-Kette

2.4 Tests

Für die Studienarbeit war das Ziel, ein Testkonzept so auszuarbeiten, dass sichergestellt werden kann, dass die Anwendung die an sie gestellten Anforderungen erfüllen kann.

Laut [2] gibt es verschiedene Arten von Software-Tests:

Unit-Tests	<ul style="list-style-type: none">• setzen auf einem sehr niedrigen Level an• testen einzelne Funktionen, Klassen oder Module der Software• einfach zu automatisieren
Integrations-Tests	<ul style="list-style-type: none">• Verifizieren, dass die einzelnen Komponenten gut zusammenarbeiten• Z. B. Zusammenarbeit aus Frontend und Backend
Funktions-Tests	<ul style="list-style-type: none">• Testen die Anforderungen an eine Anwendung• Unterschied zu Integrations Tests:<ul style="list-style-type: none">◦ Integrations-Tests testen nur ob eine Interaktion möglich ist◦ Funktions-Tests testen auch ob durch die Interaktion richtige Werte geliefert werden
End-to-End-Tests	<ul style="list-style-type: none">• Stellen das Verhalten von Benutzer*innen nach• Verifizieren, dass die Work-Flows entsprechend der Anforderungen funktionieren
Akzeptanztest	<ul style="list-style-type: none">• Formelle Tests, die testen, ob ein System die gestellten Anforderungen erreicht• Der Fokus wird auf das Nachstellen von Benutzer*innen-Verhalten gelegt• Es kann auch die Performance getestet werden
Performance-Tests	<ul style="list-style-type: none">• Testen wie sich das System unter Last verhält• Fokus liegt auf Werten wie Antwortzeiten, Zuverlässigkeit, usw.
Smoke-Tests	<ul style="list-style-type: none">• Sammlung an günstigen und schnell durchzuführenden Tests• Nützlich um zu bestimmen, ob sich die Durchführung aufwändigerer und teurerer Tests überhaupt lohnt
Exploratives Testen	<ul style="list-style-type: none">• Nicht automatisierbar, also manuell durchzuführen• Durchgehen der Anwendung und testen, wie sich das System verhält

Tabelle 1: Arten von Software-Tests (vgl. [2])

Um sicherzustellen, dass die Anwendung die gestellten Anforderungen erfüllen kann, genügt es nicht nur eine Testart umzusetzen. Außerdem müssen sowohl das Frontend als auch das Backend getestet werden. Die Entscheidung, welche Testarten umgesetzt werden sollen, fällt wie folgt aus:

Testart	Verwendet	Begründung
Unit-Tests	Ja	<ul style="list-style-type: none"> • Relativ einfach umzusetzen und zu automatisieren • Sehr nützlich, um die Funktionalität der Grundfunktionen sicherzustellen
Integrations-Tests und Funktions-Tests	Ja	<ul style="list-style-type: none"> • Hilft dabei zu verhindern, dass Änderungen die Zusammenarbeit zwischen Frontend und Backend zerstören • Viele Frontend Tests stellen entweder Integrations- oder Funktions-Tests dar, da sie auch auf Funktionen des Backends zugreifen • Zu viele Seiten um manuell zu testen
End-to-End-Tests	Nein	<ul style="list-style-type: none"> • Es werden immer nur Teile eines Benutzer Work-Flows abgebildet, aber nie ein kompletter • Testen der einzelnen Teile eines Work-Flows ist aussagekräftiger
Akzeptanztests	Ja	<ul style="list-style-type: none"> • Es ist die Anforderung gegeben, dass das Programm bei allen gültigen Sudokus eine Lösung finden muss
Performance-Tests	Nein	<ul style="list-style-type: none"> • Auf erwartete Performance wird in den Anforderungen nicht eingegangen
Smoke-Tests	Nein	<ul style="list-style-type: none"> • Durchführung von allen Tests ist mit keinen hohen Kosten verbunden • Implementierung lohnt sich somit nicht
Exploratives Testen	Teilweise	<ul style="list-style-type: none"> • Gut um Fehler aufzudecken, welche nicht durch die Tests abgedeckt sind • Tritt während der Entwicklung ohnehin auf • Aber: es wurden nie formal Testsessions abgehalten, stattdessen war diese Testform in Form von manuellem Testen der Anwendung ohnehin in dem Entwicklungsprozess enthalten

Tabelle 2: Verwendete Testarten

2.5 Technologien

In diesem Abschnitt wird auf die verwendeten Technologien eingegangen. Außerdem werden Alternativen aufgezeigt und es wird begründet, warum die Auswahl auf die verwendeten Technologien gefallen ist.

Im Rahmen der Aufgabenstellung sind zwei Möglichkeiten zur Umsetzung gegeben. So kann einerseits eine Smartphone App entwickelt werden, andererseits kann eine Web-Anwendung, welche auf einem bereitgestellten Server laufen soll, implementiert werden. Beide Möglichkeiten bieten ihre Vor- und Nachteile. So müssen z.B. bei einer Smartphone App nur wenige Bildschirmformate unterstützt werden, während eine Web-Anwendung alle Formate, von Smartphone bis Computerbildschirm unterstützen muss.

Da aber die bereits vorhandene Erfahrung bei der Entwicklung von Web-Anwendungen deutlich höher ist als bei der App-Entwicklung, wurde sich für die Umsetzung mithilfe einer Web-Anwendung entschieden.

Hierbei wird zwischen Frontend und Backend unterschieden. Laut [3] bezeichnet das Frontend den Teil des Programms, der auf dem Client läuft und die Benutzeroberfläche bereitstellt. Das Backend bezeichnet den Programmteil, der auf dem Server läuft. (vgl. [3])

2.5.1 Backend

Für das Backend muss es nötig sein, einen Web-Server zu hosten. Dieser Server soll verschiedene Seiten zur Verfügung stellen, wobei die Inhalte der Seiten über HTML geteilt werden sollen.

Hierfür gibt es eine Vielzahl an Frameworks und Tools, welche für unterschiedliche Programmiersprachen ausgelegt sind. Somit ist die Wahl der Programmiersprache also auch von der Wahl des Frameworks abhängig.

Außerdem sollen die Lösungsalgorithmen in dem Backend implementiert werden, das heißt, die Programmiersprache, welche für das Backend verwendet wird, sollte sich auch hierfür eignen.

Zuerst sollen einige Möglichkeiten für das Backend miteinander verglichen werden und die Entscheidung für eine Backend-Technologie soll begründet werden.

2.5.1.1 C# und ASP.NET

Gemäß [4] handelt es sich bei ASP.NET um ein Framework zur Entwicklung von professionellen Webseiten, sowie Webanwendungen. ASP.NET sei hierbei Teil des .NET-Frameworks, welches aus dem Hause Microsoft stammt. (vgl. [4])

Gemäß [5] heißt neueste Version des ASP.NET Frameworks ASP.NET Core. Im Gegensatz zu vorherigen Versionen soll ASP.NET Core plattformübergreifend funktionieren, anstatt nur auf Windows zu laufen, was bei vorherigen Versionen der Fall war. Außerdem handelt es sich bei ASP.NET Core um ein Open-Source Projekt. (vgl. [5])

ASP.NET bietet verschiedene Möglichkeiten, um Webseiten zu implementieren. Im Rahmen dieser Aufgabe wären dabei die Razor Pages am relevantesten.

Bei diesem Ansatz werden sogenannte Razor Templates mit dazugehörigen Modellen erstellt. In den Templates wird das statische HTML mit spezieller Syntax erweitert, so dass dynamisch HTML-Dokumente generiert werden können. Die zu den Templates zugehörigen Modelle, implementieren die Logik hinter den entsprechenden Webseiten. So können z.B. Funktionen definiert werden, welche bei auftretenden Get- oder Post-Anfragen ausgeführt werden. (vgl. [6])

Da ASP.NET zu der .NET-Plattform von Microsoft gehört, werden die für diese Plattform üblichen Programmiersprachen verwendet. Zu diesen gehört die objektorientierte Sprache C#. (vgl. [5])

2.5.1.2 Ruby und Ruby on Rails

Bei Ruby on Rails handelt es sich ebenfalls um ein quelloffenes Framework zur Entwicklung von Webanwendungen. (vgl. [7])

Das Framework verwendet eine sogenannte Model-View-Controller Architektur. Dabei werden Models verwendet, um mit relationalen Datenbanken zu interagieren. Sie bilden Klassen auf Datenbanktabellen und deren Attribute auf die passenden Spalten ab. Die View-Schicht ist für die Visualisierung bzw. für die Formulierung von Antworten auf Anfragen an den Webserver zuständig, wobei diverse Ausgabeformate unterstützt werden. Die Controller

fungieren als Schnittstelle zwischen der Model- und der View-Schicht. Sie verarbeiten ankommende Anfragen, holen dabei die entsprechenden Daten und leiten diese zur Visualisierung an die Views weiter. (vgl. [7])

Das Framework wird mit der objektorientierten Programmiersprache Ruby verwendet. Hierbei handelt es sich anders als bei C# um eine interpretierte Sprache. (vgl. [7])

2.5.1.3 Python und Django

Auch bei Django handelt es sich um ein quelloffenes Framework für die Entwicklung von Webanwendungen.

Ähnlich wie Ruby on Rails (sh. Abs. 2.5.1.2) setzt Django auf eine Architektur, welche sich nahe an der Model-View-Controller Architektur orientiert. Auch hier gibt es Models, welche die Datenbankbindung abstrahieren. Anders als bei Ruby on Rails übernehmen die Views in Django aber die komplette Bearbeitung einer Anfrage, also u.a. auch das Beschaffen der Daten, sowie das Formulieren der Antwort. Um Daten zu Visualisieren und Webseiten zu erstellen, werden HTML-Templates verwendet, mit denen sich dynamisch HTML-Code erstellen lässt. (vgl. [8])

Das Framework mit der objektorientierten Programmiersprache Python verwendet, wobei es sich wie bei Ruby um eine interpretierte Sprache handelt.

2.5.1.4 Vergleich und Entscheidung

Es zeigt sich somit, dass alle hier aufgezeigten Web-Frameworks nahezu die gleichen Funktionen bieten. Zumindest für den Umfang, der im Rahmen dieser Studienarbeit benötigt wird. So ist es mit allen Frameworks möglich, Web-Server zu erstellen und dynamisch HTML-Code für das Frontend zu erzeugen.

Dementsprechend hängt die Entscheidung über das zu verwendende Framework auch von der damit zusammenhängenden Programmiersprache ab. Da die Lösungsalgorithmen im Backend implementiert werden sollen, ist es vor allem wichtig, dass dies möglichst einfach möglich ist. Es wurden aber auch Faktoren berücksichtigt, die aus anderen Entwicklungsaspekten hervorgehen.

So handelt es sich bei C# um eine kompilierte Sprache, während Ruby und Python interpretierte Sprachen sind. Laut [9] haben kompilierte Sprachen den Vorteil, dass der Overhead, welcher durch das Kompilieren entsteht, nur einmal auftritt, danach kann das Programm beliebig ausgeführt werden. Bei interpretierten Sprachen hingegen tritt der Overhead bei jeder Ausführung auf. Somit haben kompilierte Sprachen in der Regel einen Effizienzvorteil. Dem steht gegenüber, dass es bei interpretierten Sprachen einfacher sein kann Änderungen an dem Code vorzunehmen, da der Code nicht immer neu kompiliert werden muss. (vgl. [9])

Auch aus eigener Erfahrung mit C# und ASP.NET zeigt sich, dass die Tatsache, dass der Code immer erst kompiliert werden muss, bevor er ausgeführt werden kann die

Entwicklung sehr verlangsamen kann. So muss der Code auch neu kompiliert werden, wenn nur kleine Änderungen an den HTML-Templates vorgenommen werden, was sehr häufig vorkommt. Deshalb werden im Rahmen dieses Projektes interpretierte Sprachen verwendet.

Aus den oben genannten Beispielen bleiben dann noch Python und Ruby übrig. Der eigene Kenntnisstand ist bei der Kombination aus Python und Django sehr viel höher als bei Ruby und Ruby on Rails, oder anderen Programmiersprachen und deren Web-Frameworks. Dies hat den Grund, dass Python und Django bereits für mehrere Projekte eingesetzt wurden. Dementsprechend haben wir uns für die Verwendung von Python und Django entschieden.

2.5.2 Frontend

Das Frontend bezeichnet den Programmteil, der auf dem Client läuft (vgl. Abs. 2.4). Hier werden also hauptsächlich die Ergebnisse, welche im Backend berechnet werden, dargestellt. Auch hier werden verschiedene Frameworks verwendet.

2.5.2.1 *Java Script Framework*

Es gibt eine große Anzahl von Frameworks, welche das Erstellen von Webseiten vereinfachen sollen.

2.5.2.2 *Dynamisches HTML*

Da im Frontend Daten aus dem Backend angezeigt werden sollen, muss hier HTML dynamisch erstellt werden können. Es müssen also z.B. Werte eingefügt werden können.

Django enthält bereits eine Möglichkeit Backend Daten in HTML einzufügen und somit HTML-Code dynamisch zu generieren. Das sind die sogenannten Templates. Hierbei handelt es sich um Dateien, welche sowohl statisches HTML als auch spezielle Syntax enthalten. Mithilfe dieser speziellen Syntax kann HTML in Abhängigkeit von Daten aus dem Backend erzeugt werden. So kann z.B. über Listen iteriert werden, es können die Werte von Variablen eingefügt werden, aber auch Verzweigungen sind möglich. Das Rendern der Templates findet dabei im Backend statt, es wird also vom Server übernommen. (vgl. [10])

2.5.2.3 CSS Libraries

Theoretisch würde es genügen, eine Webseite nur über HTML zu definieren, diese Webseite wäre visuell dann aber sehr wenig ansprechend und es ließen sich wenig visuelle Hilfen einbringen. Deshalb werden die Elemente der Webseite mithilfe von CSS und JavaScript visuell aufgewertet.

Um dabei zu helfen können CSS-Frameworks verwendet werden. Laut [11] bestehen diese Frameworks dafür hauptsächlich aus CSS-Stylesheets, also aus Stildefinitionen durch CSS. Durch diese Stylesheets werden viele Standard-Elemente (wie Knöpfe, Navigationsleisten, etc.) gestylt und gegebenenfalls auch mit entsprechenden Funktionalitäten versehen. Dafür können neben CSS auch andere Technologien wie z.B. JavaScript zum Einsatz kommen. Somit muss nur noch darauf geachtet werden, dass das HTML der Webseite die richtige Struktur hat und die Attribute, wie die IDs oder die Klassen, der einzelnen HTML-Elemente korrekt sind. (vgl. [11])

Es gibt verschiedene CSS Libraries, welche hier kurz vorgestellt werden sollen, außerdem soll die Entscheidung für eine Library begründet werden.

2.5.2.3.1 Bootstrap

Bei Bootstrap handelt es sich um ein Open-Source Framework, welches CSS und JavaScript basierte Vorlagen enthält. Außerdem legt Bootstrap einen hohen Fokus darauf, dass Webseiten auf allen Bildschirmgrößen funktionieren. So existieren viele Klassen und Systeme, wie z.B. das Grid-System, welche auf unterschiedlichen Bildschirmgrößen unterschiedliche Designs besitzen. (vgl. [11])

Ein Bootstrap Grid ordnet die Elemente bspw. erst ab einer bestimmten Bildschirmgröße horizontal an, bei kleineren Bildschirmen sind die Elemente vertikal angeordnet.

2.5.2.3.2 Tailwind CSS

Auch bei Tailwind CSS handelt es sich um ein CSS-Framework. Ähnlich wie Bootstrap legt auch Tailwind CSS einen hohen Fokus auf das Darstellen von Webseiten auf unterschiedlichen Bildschirmgrößen. (vgl. [11])

Im Gegensatz zu Bootstrap verfügt Tailwind CSS aber über ein höheres Maß an Möglichkeiten zur Individualisierung. So ist z.B. kein Standard-Farbmuster vorgegeben und es existieren auch keine Komponenten, welche Nutzer*innen Design Entscheidungen abnehmen. Stattdessen besteht es nur aus einer Anzahl an Widgets, aus denen die Seite aufgebaut werden kann. (vgl. [12])

2.5.2.3.3 Entscheidung

Von einer oberflächlichen Betrachtungsweise unterscheiden sich die hier aufgelisteten Frameworks also hauptsächlich dadurch, wie gut sie individualisierbar sind und welche Entscheidungen einem abgenommen werden. Zu den verfügbaren UI-Komponenten lässt sich noch keine Aussage treffen. Da die zu implementierende Webseite aber auch einen simplen Aufbau haben soll, ist ein genauer Vergleich auf dieser Ebene nicht nötig.

Stattdessen wird die Entscheidung, welches Framework verwendet werden soll von der eigenen Erfahrung, sowie von Hilfsmitteln, wie z.B. Tutorials abhängig gemacht.

Somit fällt die Entscheidung auf Bootstrap, da dieses Framework bereits verwendet wurde, es sind also bereits Erfahrungen vorhanden. Außerdem gibt es zu Bootstrap ein sehr umfangreiches Tutorial von den W3Schools ⁽¹⁾, welches auch Vorlagen und Beispiele enthält, wodurch die Entwicklung von Webseiten sehr vereinfacht wird.

2.5.2.4 Java Script Libraries

Laut [13] ist Java Script neben HTML und CSS eine der grundlegenden Technologien für das Frontend von Webseiten. Während mit HTML und CSS die Struktur, sowie das Design der Webseite definiert werden, sorgt JavaScript dafür dass die Webseite dynamisch ist. So können mit JavaScript beispielsweise Reaktionen auf einen Mausklick definiert werden. (vgl. [13])

Im Rahmen dieses Projektes wird JavaScript verwendet um das Frontend mit diversen Funktionen zu versehen. Dabei werden hauptsächlich die folgenden Funktionen mit JavaScript umgesetzt:

- Manipulation der DOM um Hilfen ein- und ausblenden zu können und um die Ergebnisse der Algorithmen zu visualisieren
- Simple Verifikation von Eingaben, sowie das markieren von Falscheingaben

Um diese Anforderungen umzusetzen, gibt es mehrere Möglichkeiten, welche an dieser Stelle beleuchtet werden sollen.

2.5.2.4.1 Standard Java Script

Die erste Möglichkeit besteht daraus, dass nur standardmäßiges Java Script verwendet wird. Denn hiermit lassen sich alle nötigen Anforderungen erfüllen.

2.5.2.4.2 JQuery

Laut [14] handelt es sich bei JQuery um eine freie JavaScript Bibliothek, welche von vielen Webseiten verwendet wird und außerdem die am meisten verwendete JavaScript Bibliothek ist. Dabei fügt JQuery aber keine neuen Funktionen zu JavaScript hinzu. Das heißt, alles was mit JQuery möglich ist, ist auch mit JavaScript möglich. Stattdessen stellt JQuery Hilfen bereit, mit denen sich die gleichen Funktionen mit weniger Code umsetzen lassen. Dadurch sollen die Programme übersichtlicher werden und auch der Entwicklungsaufwand soll verringert werden. (vgl. [14])

Soll z. B. zu einem Element eine CSS-Klasse hinzugefügt werden, ergibt sich mit reinem JavaScript folgender Code:

```
document.querySelector("img#img-logo").classList.add("class1");
```

Abbildung 6: Hinzufügen einer CSS-Klasse mit JavaScript (Erstellt mit: [15])

¹ Bootstrap Tutorial W3Schools: <https://www.w3schools.com/bootstrap5/index.php>

Bei Verwendung von JQuery ergibt sich folgender Code:

```
$("img#img-logo").addClass("class1");
```

Abbildung 7: Hinzufügen einer CSS-Klasse mit JQuery (Erstellt mit: [15])

Es zeigt sich also, dass der mit JQuery geschriebene Code kürzer und auch übersichtlicher ist, als der mit reinem JavaScript geschriebene Code.

Laut [16] hat die Verwendung von JQuery aber nicht nur Vorteile. So muss der Code für die Bibliothek auch erst geladen werden, wobei auch viel Code mitgeladen wird, der gar nicht benötigt wird. Möchte man also eine möglichst schlanke Anwendung entwickeln, ist die Verwendung von JQuery hier kontraproduktiv. Außerdem bietet die Verwendung von JQuery keinen Mehrwert in dem Sinne, dass neue Funktionalitäten ergänzt werden. Im Endeffekt ist die Frage, ob JQuery verwendet werden soll aber Geschmackssache, da die Geschwindigkeitseinbußen, welche durch die Verwendung von JQuery entstehen nur marginal sind, die Bibliothek aber oft auch keinen starken Mehrwert bietet. (vgl. [16])

2.5.2.4.3 Entscheidung

Wie bereits erwähnt, ist die Entscheidung ob reines JavaScript oder JQuery verwendet werden soll, eine Frage des Geschmacks, da die Verwendung von JQuery zwar nur minimale Geschwindigkeitseinbußen mit sich bringt, der Code aber auch nur ein wenig kürzer und übersichtlicher wird.

Da JQuery aber viele Funktionen bietet, welche sehr nützlich für die Manipulation der Webseite sind, also z. B. CSS-Klassen hinzufügen oder entfernen, wird für dieses Projekt JQuery verwendet. Dadurch soll der Code für das Frontend vor allem kürzer und übersichtlicher, sowie leichter verständlich werden.

2.5.3 Tests

Wie bereits erwähnt sollen sowohl das Frontend als auch das Backend getestet werden.

2.5.3.1 Testframework

Für alle Tests wird zuerst ein Testframework benötigt, mit dem Tests geschrieben und ausgeführt werden können. Dabei ist es hier nur nötig, dass Test-Cases erstellt und im Rahmen dieser Test-Cases Bedingungen überprüft werden können.

Die Auswahl begrenzt sich hier auf zwei Frameworks, da Python standardmäßig über das Modul „unittest“ verfügt, mit dem Test-Cases erstellt und ausgeführt werden können. So können Test-Cases mit zugehörigen Test-Funktionen erstellt werden, in denen mit Hilfe von „assert“-Funktionen Bedingungen überprüft werden können. (vgl. [17])

Außerdem liefert Django ein Testframework mit, welches auf dem Modul „unittest“ basiert und somit ähnliche Funktionalität bietet wie „unittest“, aber mehr auf das Testen von Django-Anwendungen zugeschnitten ist. (vgl. [17])

Da das Modul „unittest“ bereits alle Anforderungen, welche hier an ein Testframework gestellt werden erfüllt und hauptsächlich die grundlegende Logik und nicht die Django-

Anwendung getestet werden soll, wird das Python-eigene Modul „unittest“ als Testframework verwendet.

2.5.3.2 Frontend-Steuerung

Um Tests für das Frontend implementieren zu können, wird eine Möglichkeit benötigt, das Frontend automatisiert steuern zu können. Um die Code-Basis möglichst einheitlich zu halten, sollen die Frontend Tests gleich aufgebaut sein, wie die Tests für das Backend, also die Testdefinition soll ebenfalls in Python mit dem Modul „unittest“ erfolgen.

Hierfür ist die Python Bibliothek Selenium sehr gut geeignet. Mit ihr lässt sich ein Browser über Python fernsteuern. So lassen sich Webseiten aufrufen, sowie Aktionen auf diesen Webseiten tätigen, mit denen jegliches Verhalten von Nutzer*innen simuliert werden kann. So können z. B. Werte in Felder eingetragen oder Knöpfe gedrückt werden. (vgl. [18])

3 Implementierung

3.1 Versionsverwaltung

Die Versionsverwaltung wurde mit Github gemacht. Es ist ein Verteiltes Versionskontrollsystem. Nutzer*innen benutzen nicht nur die letzte Version des Projektes, sie arbeiten stattdessen auf der einer vollständigen Kopie des Repository. Auf diese Weise kann wenn die Daten auf dem Server beschädigt werden diese wieder von dem Anwenderrechner zurückkopiert werden oder auch andersrum. Jede Kopie ist ein Klon, ein vollständiges Backup der gesamten Projektdaten. Der Aufbau eines Verteilten Versionskontrollsystem ist in Abbildung 8 zu sehen und die ein Beispiel für die Versionen ist in Abbildung 9 zu sehen. (vgl. [19])

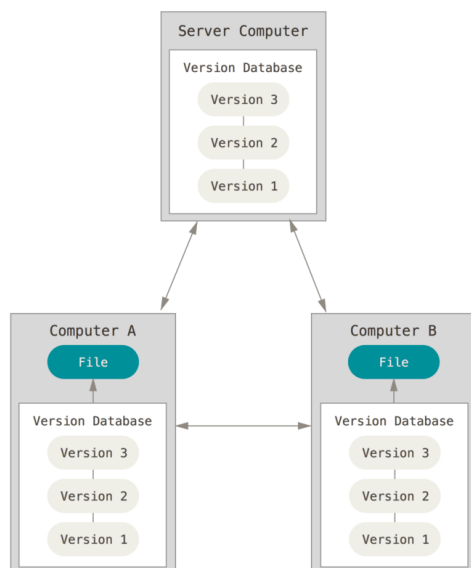


Abbildung 8: Verteiltes Versionsverwaltung



Abbildung 9: Beispiel Versionen

3.2 Architekturdesign

Das Architekturdesign des Programms ergibt sich hauptsächlich aus den Komponenten und Aufgaben, in die das Programm aufgeteilt werden kann. Diese sind:

- Sudoku und dessen Felder
- Validierung des Sudokus
- Algorithmen
- Django Server

Diese Teile werden in jeweils eigenen Python-Modulen implementiert, um die Implementierung möglichst übersichtlich zu halten und um die Komponenten möglichst unabhängig voneinander zu halten.

3.2.1 Sudoku

Das Sudoku stellt die Basis für alle weiteren Implementierungen dar. In diesem Modul werden die Funktionen und Klassen implementiert, die im Allgemeinen verwendet werden, um auf das Sudoku zuzugreifen.

Dazu gehört z. B. das Speichern und Bearbeiten der Werte und der Kandidaten von den Sudoku-Feldern. Es gehören aber auch komplexere Operationen dazu, so kann z. B. auf ganze Einheiten zugegriffen werden. Auch die simplen Validierungsformen, also dass alle Felder gültige Werte besitzen und alle Werte maximal einmal pro Einheit auftauchen, sind in diesem Modul implementiert.

Außerdem enthält dieses Modul Tests für kompliziertere Operationen. Somit soll sichergestellt werden, dass keine Fehler existieren, welche weiterführende Komponenten beeinträchtigen könnten.

3.2.2 Validierung

In diesem Modul ist der Backtracking-Algorithmus enthalten. Dieser validiert, dass das Sudoku exakt eine Lösung besitzt (sh. Abs. 3.3.2). Außerdem enthält dieses Modul Tests, mit denen sichergestellt werden soll, dass die Validierung ordnungsgemäß funktioniert.

3.2.3 Algorithmen

Die Implementierung der Algorithmen ist eine der zentralen Aufgaben dieses Projektes. Dementsprechend ist dieses Modul sehr umfangreich. Es enthält die Implementierung aller Algorithmen. Außerdem werden Funktionen bereitgestellt, mit denen das gezielte Ausführen einzelner Algorithmen, oder das Ausführen aller Algorithmen, bis sich einer auf das Sudoku anwenden lässt, stark vereinfacht wird. Dies dient dazu an anderen Stellen Aufwand zu sparen.

Außerdem enthält dieses Modul viele Hilfsfunktionen, welche für die Implementierung der Algorithmen notwendig sind, oder diese vereinfachen.

Dieses Modul enthält außerdem noch Tests für einige der Hilfsfunktionen, Tests für die einzelnen Algorithmen, sowie Tests mit denen ermittelt wird, inwieweit die implementierten Algorithmen ausreichen, um eine Reihe von gegebenen Sudokus zu lösen.

3.2.4 Django Server und Zusätze

Die Implementierung der tatsächlichen Web-Anwendung mit Django wird auf mehrere Module aufgeteilt. Das Modul „backend“ stellt das Django-Projekt dar, die Logik für die Web-Anwendung wird in das Modul „sudoku_helper“ ausgelagert. Hierbei handelt es sich um eine Django-App.

Der Unterschied zwischen einer Django-App und einem Django-Projekt besteht darin, dass die App eine Web-Anwendung mit einer Aufgabe abbilden soll. Das Django-Projekt hingegen ist eine Ansammlung von Konfigurationen und Django-Apps, und stellt die vollständige Webseite dar. Dadurch können Django-Apps einfach in verschiedenen Projekten wiederverwendet werden. (vgl. [20])

Außerdem befinden sich alle statischen Dateien, d. h. JavaScript Skripte, CSS-Dateien und Bilder in dem Ordner „static“ innerhalb des Moduls „sudoku_helper“.

Für die Django-Templates, d. h. die Dateien, in denen mit Hilfe statischen HTMLs, sowie zusätzlicher Syntax dynamisch HTML generiert wird (vgl. [10]), wird (der Übersicht halber) ein eigener Ordner angelegt, welcher sich auf gleicher Höhe wie die Module befindet. Innerhalb dieses Ordners werden die Templates mit weiteren Ordnern sortiert, dies ist hier aber nicht relevant.

3.2.5 Frontend-Tests

Die Frontend Tests stellen ein eigenes Modul dar, dies befindet sich aber eine Ebene über den bisher beschriebenen Modulen, da diese alle zur Implementierung der Web-Anwendung dienen, während die Frontend Tests diese lediglich testen sollen.

Da innerhalb der Frontend Tests die Visualisierung für jeden Algorithmus extra getestet wird, werden die Tests hierfür mit einem Ordner gruppiert.

3.2.6 Development Tools

Hierbei handelt es sich um einen Teil der Anwendung, welcher zwar nicht in den Anforderungen beschrieben wird, aber hinzugefügt wird, um die Entwicklung zu vereinfachen.

Hauptsächlich dient dieser Teil der Anwendung dazu, bei der Entwicklung des Frontends zu unterstützen und hier gezielte Tests zu ermöglichen.

Dementsprechend wird in den Dev-Tools (abgekürzt für „Development Tools“ also „Entwicklungswerkzeuge“) auch sehr wenig Logik implementiert. Stattdessen werden hauptsächlich Vorlagen für Sudokus angelegt. Es werden dabei zwei verschiedene Funktionen implementiert, zuerst gibt es vorausgefüllte Sudokus, welche verwendet werden können, um sich das Eintippen der Werte in die Webseite zu ersparen. Außerdem gibt es auch die Möglichkeit gezielt Algorithmen zu Testen. Das bedeutet es wird ein vorgefertigtes Sudoku verwendet und darauf wird lediglich der gewünschte Algorithmus angewandt. Dies ist vor allem zum Testen des Algorithmus-spezifischen Frontends (sh. Abs. 3.5.3.3) nützlich.

3.2.7 Ordner- und Modulstruktur

Hier ist das Ziel nur, eine Vorstellung von der Ordner- und Modulstruktur zu vermitteln, deshalb werden keine einzelnen Dateien angezeigt, und es werden nur die in den vorherigen Abschnitten beschriebenen Ordner und Module betrachtet. Dies dient auch der Übersichtlichkeit.

Es ergibt sich folgende Struktur:



Abbildung 10: Ordner- und Modulstruktur der Anwendung

3.3 Backend Implementierung

Das Backend stellt den größten Teil der Implementierung dar, denn hier ist neben der Logik für den Webserver auch die Logik für das Sudoku, sowie die Lösungsalgorithmen implementiert.

3.3.1 Reaktion auf Fehleingaben

Wie in Abs. 3.5 noch beschrieben wird, existieren zwar im Frontend schon Mechanismen, um Fehleingaben durch den Nutzer zu verhindern, jedoch könnten diese Mechanismen durch Nutzer*innen deaktiviert werden, z. B. in dem die Ausführung von JavaScript in dem Browser deaktiviert wird. Deshalb ist es nötig, dass die Eingaben auch im Backend noch einmal überprüft werden.

Hierfür wird zuerst überprüft, ob die eingetragenen Werte in dem gültigen Wertebereich liegen, falls nicht werden sie einfach ignoriert. Eine Zehn oder Buchstaben werden somit z. B. durch das Backend gelöscht.

Außerdem wird überprüft, dass keine Einheit einen Wert mehr als zweimal enthält. Werden durch diese Tests Fehler in der Nutzereingabe aufgedeckt, so wird dies entsprechend über das Frontend kommuniziert.

3.3.2 Backtracking

Eine Backtracking Algorithmus ist ein rekursiver Algorithmus, es werden alle möglichen Wege versucht, um das Problem zu lösen. Jeder mögliche Weg wird getestet bis eine Lösung gefunden wird, in dem Fall der Validierung eines echten Sudoku wird ein Zähler erhöht.

Sobald der Zähler 2 erreicht, ist es kein echtes Sudoku mehr, da der Algorithmus zwei Lösungen gefunden hat.

In Abbildung 11 zusehen ist ein vereinfachtes Struktogramm des implementierten Backtracking Algorithmus.

Hierbei kann man in der ersten If-Abfrage die Abbruchbedingung sehen, falls mehr als eine Lösung gefunden wurde.

Des Weiteren ist zusehen das durch das komplette Sudoku iteriert wird und jeder mögliche Wert für ein Feld ausprobiert wird.

In der innersten If-Abfrage, wird überprüft, ob das Sudoku gelöst wurde, wenn ja, dann wird der counter um 1 erhöht und als Rückgabe weitergegeben, wenn nein wird die Funktion rekursiv aufgerufen.

Der Backtracking Algorithmus beruht auf der Verwendung von rekursiven Funktionen. Eine rekursive Funktion ist eine Funktion, die sich selbst aufruft, bis eine bestimmte Bedingung erfüllt ist. (vgl. [21])

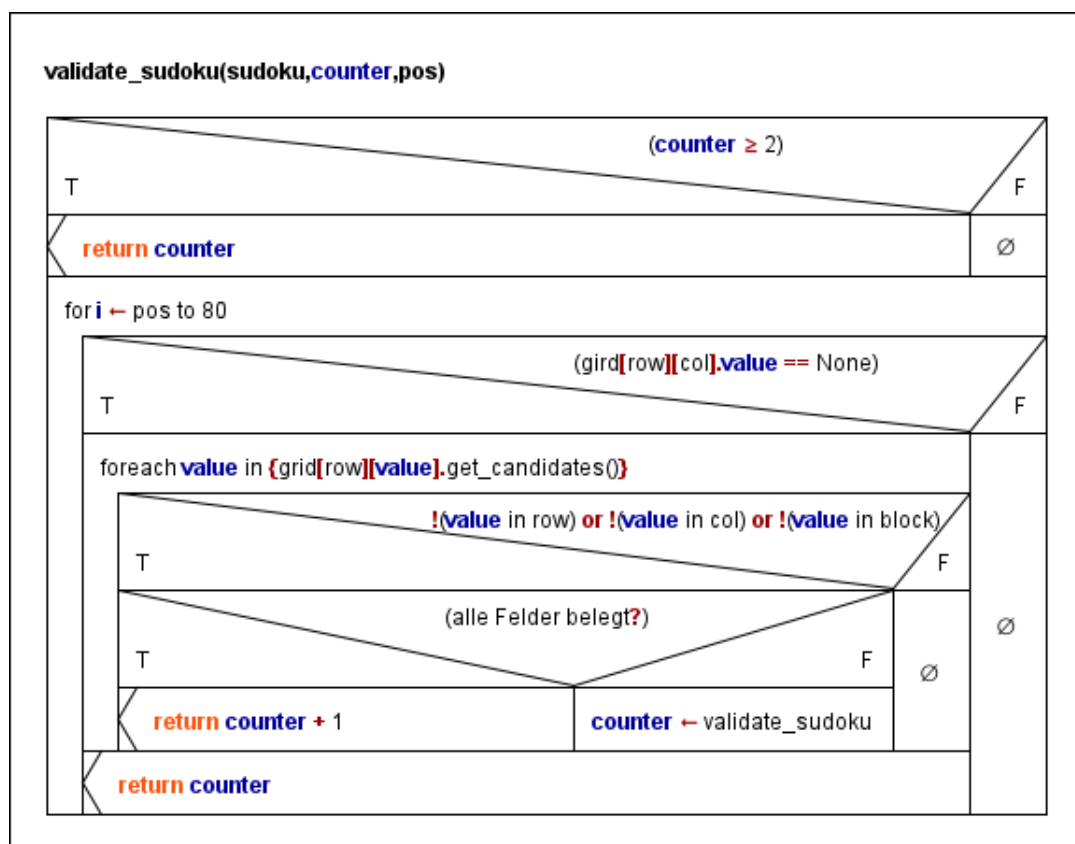


Abbildung 11: Backtracking Struktogramm

3.3.3 Algorithmen

Wie bereits in 2.3 beschrieben wurden insgesamt 25 Algorithmen implementiert. Die Implementation von drei dieser Algorithmen wird in diesem Kapitel erklärt.

3.3.3.1 Einfacher Lösungsalgorithmus Beispiel Block-Reihen-Check

Die Implementierung des Algorithmus wurde in Abbildung 12 vereinfacht beschrieben.

In der ersten for-Schleife wird durch die Blöcke iteriert, wobei die Konstante NINE_RANGE Werte von 0-8 umfasst. Daraufhin wird durch alle Werte iteriert, die die Konstante ALL_FIELD_VALUES umfasst die Werte eines Sudokus 1-9. Im nächsten Iterationsschritt wird durch alle Felder in dem Block *i* iteriert. Wenn nun ein Wert in dem Feld ist, wird geschaut in welcher Reihe das Feld liegt, insgesamt gibt es pro Block drei verschiedene Reihen. Nach dem alle Felder des Blocks geprüft wurden, wird mit Hilfe einer If-Abfrage geschaut ob, nur eine Reihe belegt wird, dies wird mit einem Exklusiv-Oder gemacht.

Damit ist das erste Merkmal dieses Algorithmus gegeben. Im nächsten Schritt hier in der If-Abfrage (check if candidates can be removed) zusammengefasst wird durch diese Reihe iteriert und geschaut, ob ein Feld außerhalb des Blocks existiert welches diesen Kandidaten beinhalten. Wenn das gegeben ist, kann dieser Algorithmus auf das gegebene Sudoku angewendet werden und liefert True und ein dem Algorithmus angepasstes Dictionary zurück. (vgl. [1])

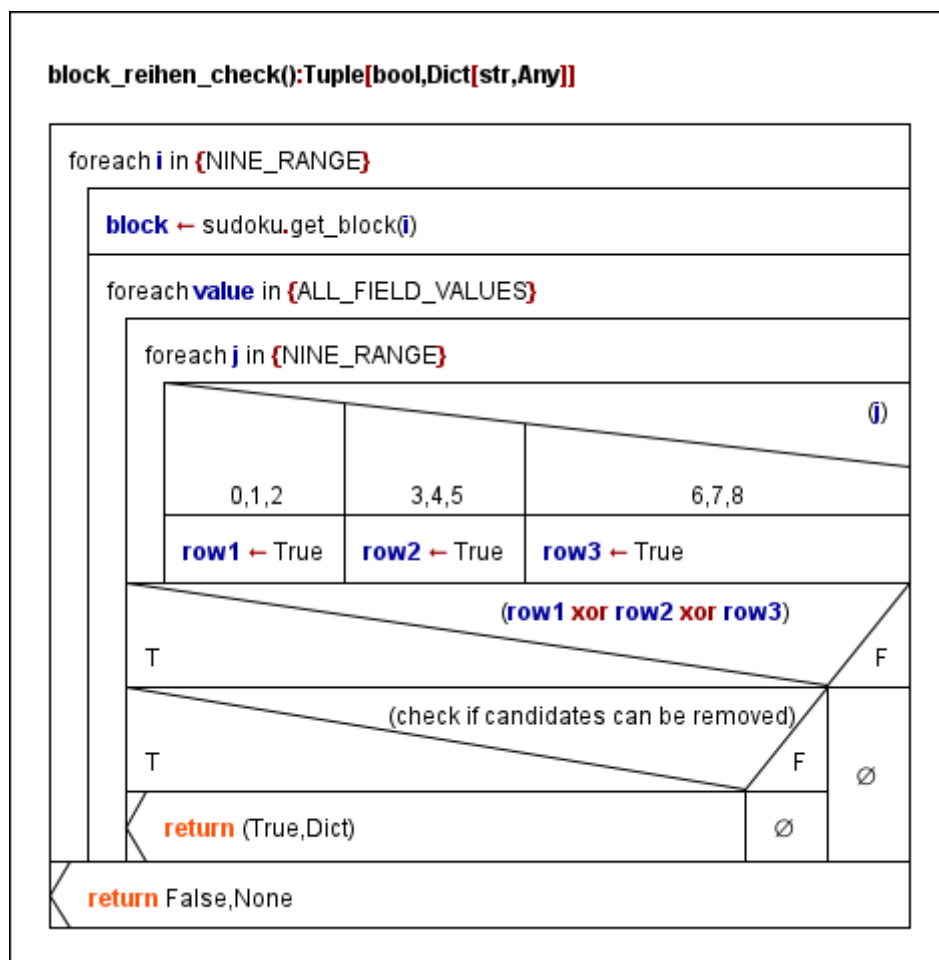


Abbildung 12: Block_Reihen_Check Struktogramm

Bei erfolgreich angewendetem Algorithmus wird ein Tuple aus True und einem Dictionary zurückgegeben. Das Dictionary ist für jeden Algorithmus anders und besteht aus key, Wert Paaren. In diesem Fall ist `algorithm` der Name des Algorithmus, `value` der Wert, der aus den Kandidaten gestrichen werden kann, `intersect_fields` die betroffenen Einheiten (Block und Reihe) und `removed_candidates` die Felder, aus denen der Kandidat eliminiert wurde.

```
return (True, {
    'algorithm': 'block_row_check',
    'value': value,
    'intersect_fields': intersect_fields,
    'removed_candidates': removed_candidates
})
```

Abbildung 13: Rückgabe Block-Reihen-Check (Erstellt mit: [15])

3.3.3.2 Mittelschwerer Lösungsalgorithmus Beispiel Wolkenkratzer

Die Implementierung des Algorithmus wurde in Abbildung 15 vereinfacht beschrieben.

Zuerst wird durch jede Reihe iteriert. In dem nächsten Iterationsschritt wird durch jeden möglichen Wert in einem Sudoku iteriert und danach durch die Felder in der Reihe.

Wenn ein Wert in den Kandidaten dieses Feld liegt, wird dieser zu der Liste fields hinzugefügt.

Wenn in dieser Reihe zwei oder mehr Felder gefunden wurden, wird durch fields iteriert.

Für jedes Feld in Fields wird überprüft, ob genau zwei Felder in der Spalten den Kandidaten besitzen, wenn ja wird diese zwei Felder dieser Spalte als Liste (hier: col) einer anderen Liste (hier: cols) hinzugefügt. Das heißt am Ende der Iteration durch die fields, gibt es eine Liste (cols), welche Listen von Spalten(col) beinhaltet in denen es genau zwei Felder mit diesem Kandidaten gibt. Im Anschluss wird durch jedes Feld dieser Spalten iteriert, wenn sie sich einen gemeinsamen Einflussbereich teilen und aus diesem Einflussbereich Kandidaten eliminiert werden können, so kann dieser Algorithmus angewendet werden und gibt ein Tuple aus True und einem Dictionary zurück. (vgl. [1])

Diese Dictionary ist in Abbildung 14 zusehen und beinhaltet andere Key, Wert Paare als der in 3.3.3.1 beschriebene Algorithmus. Neben den bereits erklärten Paaren gibt es `fields-staggerd`, welche die Felder beinhaltet, die für die Eliminierung der Kandidaten verantwortlich sind. Und `row_same_height` ist die Reihe, auf welcher die beiden Felder liegen, die auf derselben Reihe liegen.

```

return (True, {
    'algorithm': 'skyscraper',
    'value': value,
    'fields_staggered': [b, y],
    'row_same_height': i,
    'removed_candidates': removed_candidates
})

```

Abbildung 14: Rückgabe Wolkenkratzer (Erstellt mit: [15])

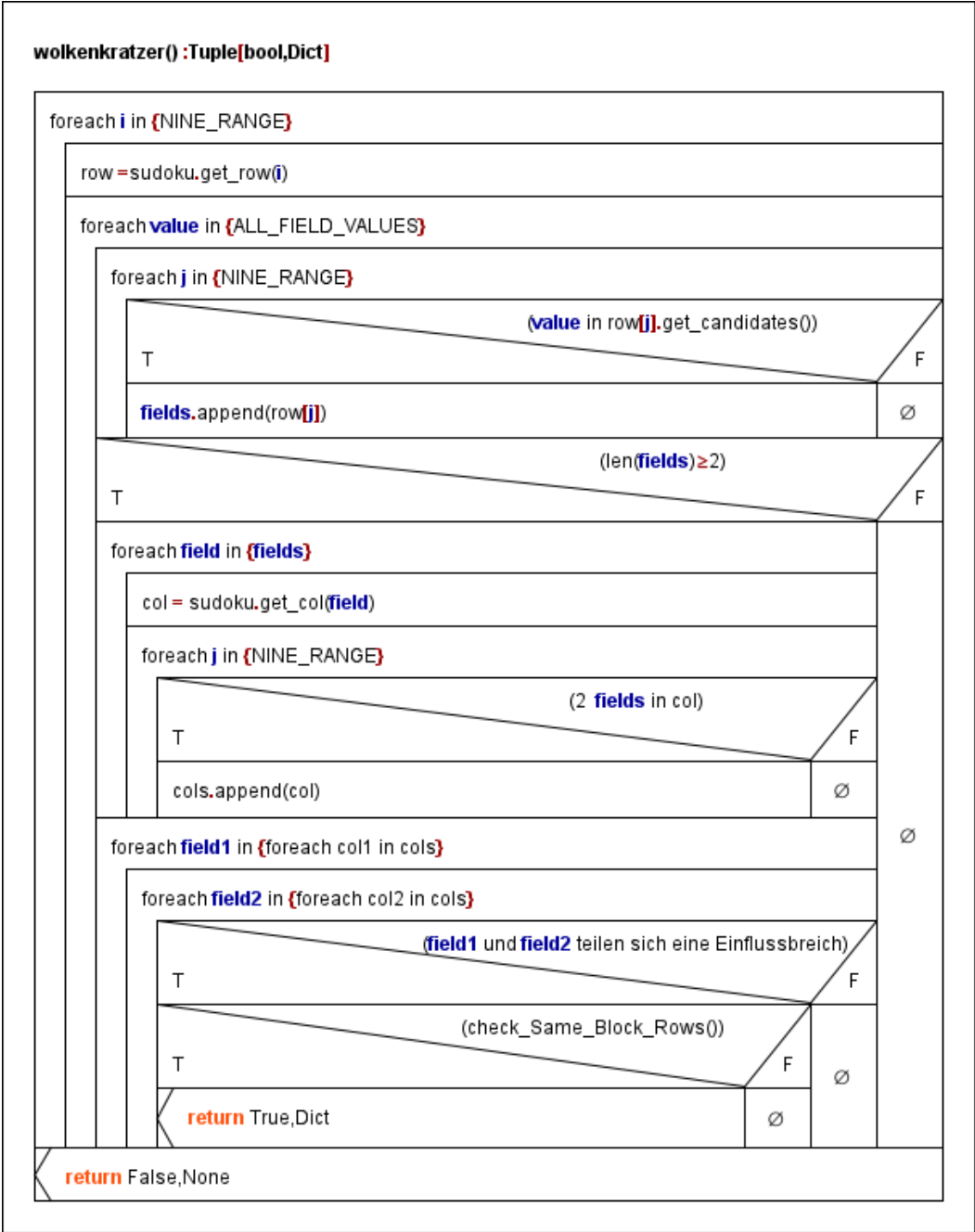


Abbildung 15: Wolkenkratzer Struktogramm

3.3.3.3 Schwerer Lösungsalgorithmus Beispiel XY-Kette

Die Implementierung des Algorithmus wurde in Abbildung 16 vereinfacht beschrieben.

Zuerst wird durch das gesamte Sudoku iteriert und Feldern mit zwei Kandidaten werden der Liste fields hinzugefügt.

Dann wird durch die Liste fields iteriert und ein geprüft, ob das Feld f der Anfang zweier Ketten ist. Da das Feld f mehr als zwei Ketten bilden kann, werden die jeweils ersten Felder der Kette in einer Liste gespeichert. Eine Liste fields1 für den ersten Kandidatwert von f und eine Liste fields2 für den zweiten Kandidatwert von f.

Nun werden alle Felder in fields1 und fields2 miteinander verglichen, das funktioniert mit Hilfe der Iterationen über fields1 und fields2.

Für jedes dieser Felderpaare werden die Ketten mit Hilfe eine Hilfsfunktion get_chain gesucht. Und man bekommt eine Liste an Tuple zurück, bestehend aus dem Kandidaten und dem dazugehörigen Feld.

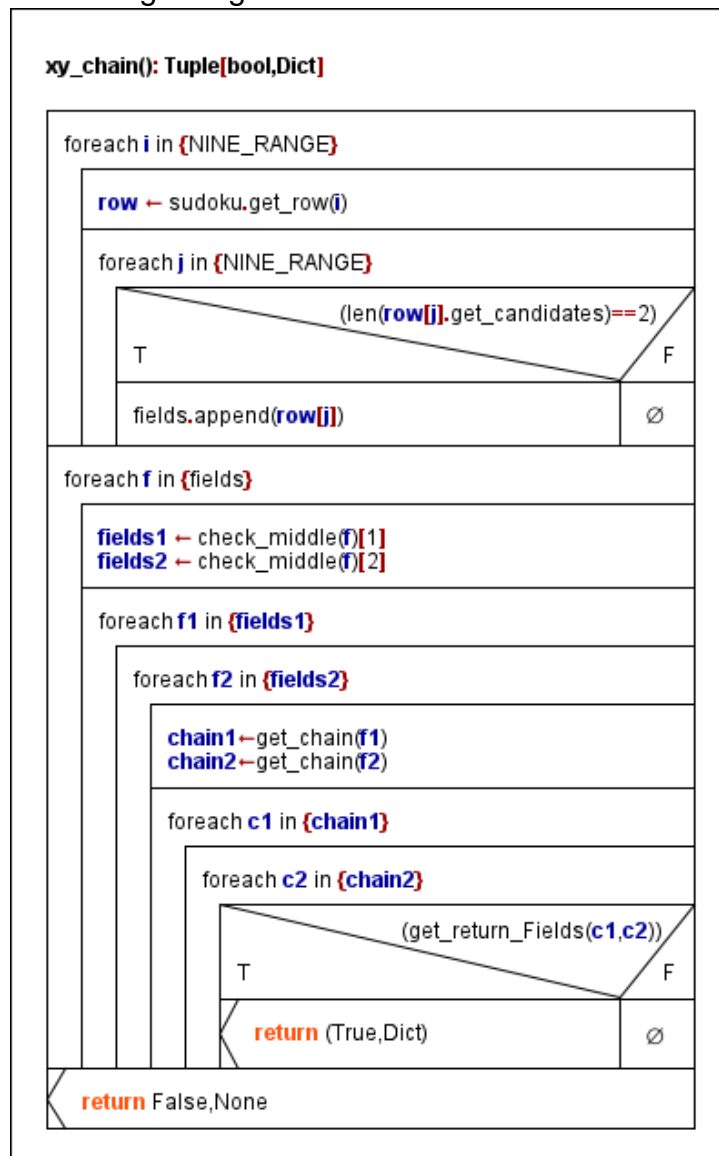


Abbildung 16: XY-Kette Struktogramm

Nun wird durch beide Ketten-Listen iteriert und verglichen, ob es in beiden Ketten ein Feld mit demselben Kandidaten gibt und diese einen gemeinsamen Einflussbereich haben, in dem es ein Feld gibt, das diesen Kandidaten besitzt. Dies wird in get_return_Fields gemacht. (vgl. [1])

3.4 Datenfluss

Da die Anwendung sowohl im Backend als auch im Frontend über Logik verfügt, müssen einige Daten zwischen Front- und Backend übertragen werden. Die wichtigsten Übertragungswege sollen an dieser Stelle erläutert werden.

3.4.1 Frontend zu Backend

Von dem Frontend ans Backend müssen die Werte des Sudokus übertragen werden. Dafür wird das Sudoku als ein HTML-Formular gestaltet.

HTML-Formularen beinhalten verschiedene Arten von Eingabemöglichkeiten, welche bei einer Bestätigung des Formulars an den Server übertragen werden. So sind z.B. Textfelder, oder auch Felder für die Eingabe von Zahlen möglich. Den Eingabefeldern kann dabei mit dem Attribut „name“ ein Schlüssel zugewiesen werden, mit dem der Inhalt des Feldes versehen wird. Dieser Schlüssel kann also serverseitig dazu verwendet werden, die Werte aus der Anfrage zu extrahieren. (vgl. [22])

Damit der Server die einkommenden Sudoku Werte möglichst einfach aus der Anfrage extrahieren kann, wird das Attribut „name“ von jedem Eingabefeld mit den x- und y-Koordinaten, die das Feld in dem Sudoku besitzt, versehen.

Für Seiten, auf denen die Sudoku Werte nicht durch den Nutzer eingegeben werden, sondern von dem Server geliefert werden, werden die entsprechenden Werte mithilfe der Django Templates dynamisch das HTML der Eingabefelder eingetragen und das

```
<input readonly id="2_1" name="2_1" type="number" value="7" maxlength="1"/>
```

Abbildung 17: vereinfachtes HTML für ein Sudoku-Eingabefeld (Erstellt mit: [15])

Schreiben in die Eingabefelder wird blockiert. Der HTML Code für ein solches Eingabefeld, welches an den Koordinaten y=2, x=1 liegt und den Wert 7 hat, wird vereinfacht in Abbildung 17 dargestellt.

Da der Server die Sudokus immer nur aus den eingehenden Anfragen konstruiert und nicht zwischenspeichert, und die Algorithmen hauptsächlich die Kandidaten verändern, müssen auch die Kandidaten über das Frontend übertragen werden. Dafür wird für jedes Sudoku Feld ein zweites Eingabefeld definiert, welches die Liste der Kandidaten als Wert enthält. Dieses Feld ist für das Frontend aber irrelevant und wird dem Nutzer auch nicht angezeigt, es dient lediglich zum Zwischenspeichern der Kandidaten. Besitzt das Feld an den Koordinaten y=3, x=4 die Kandidaten 7,8 und 9, ergibt sich der HTML Code, wie er in Abbildung 18 vereinfacht dargestellt ist.

Außer den Kandidaten, sowie den Sudoku Werten müssen innerhalb dieser Anwendung keine Daten über das Frontend an das Backend übertragen werden.

```
<input id="candidates_3_4" name="candidates_3_4" value="[7, 8, 9]" readonly>
```

Abbildung 18: vereinfachtes HTML für ein Eingabefeld, in dem die Kandidaten für ein Feld zwischengespeichert werden (Erstellt mit [15])

3.4.2 Algorithmus Ergebnisse ans Frontend

Die Rückgabe besteht aus einem Tupel aus einem Wahrheitswert und einem Dictionary. Der Wahrheitswert gibt Informationen ob der Algorithmus angewendet werden kann oder nicht.

Das Dictionary beinhaltet alle anderen Informationen wie Name des Algorithmus, Felder/ Einheiten die markiert werden sollen, Kandidaten die aus Feldern gelöscht werden können.

Das Dictionary ist ein Sammlung Daten welche in key:value Paaren gespeichert werden. Ein Dictionary ist außerdem auch geordnet. Es ist somit sehr flexibel einsetzbar was für die Rückgabe der Lösungsalgorithmen sehr wichtig ist, da diese sich zwischen den verschiedenen Lösungsalgorithmen unterscheidet.

Das Dictionary muss außerdem an das Frontend übertragen werden, da der Code für die Algorithmenvisualisierung, welcher im Frontend läuft, die Informationen aus dem Dictionary benötigt.

Dafür wird das Dictionary im Backend mit JSON serialisiert und mithilfe der Templates in das übertragene HTML eingefügt. Im Frontend wird das serialisierte Dictionary wiederum geparsed und steht somit zur weiteren Verarbeitung zur Verfügung.

3.5 Frontend

Das Frontend ist hier ein sehr wichtiger Teil der Anwendung, da die Aufgabe nicht nur daraus besteht das Sudoku mithilfe der Algorithmen im Backend zu lösen, stattdessen sollen die Algorithmen verständlich dargestellt und schrittweise konkretisiert werden. (sh. Abs. 0)

Im Gegensatz zum Backend wird im Frontend also weniger Logik implementiert, dafür wird ein verständliches und klares Design umgesetzt, außerdem werden die Erkenntnisse, welche durch die Logik im Backend gewonnen werden, visualisiert.

3.5.1 Arbeitsschritte

Um das Frontend zu entwickeln werden zuerst Mockups für einige Seiten erstellt. Hierfür wird die Anwendung Figma verwendet. Die Mockups sind noch keine finalen Designs, sie dienen aber dazu, die Elemente auf der Seite zu verteilen und eine grobe Auswahl an Farben zu treffen.

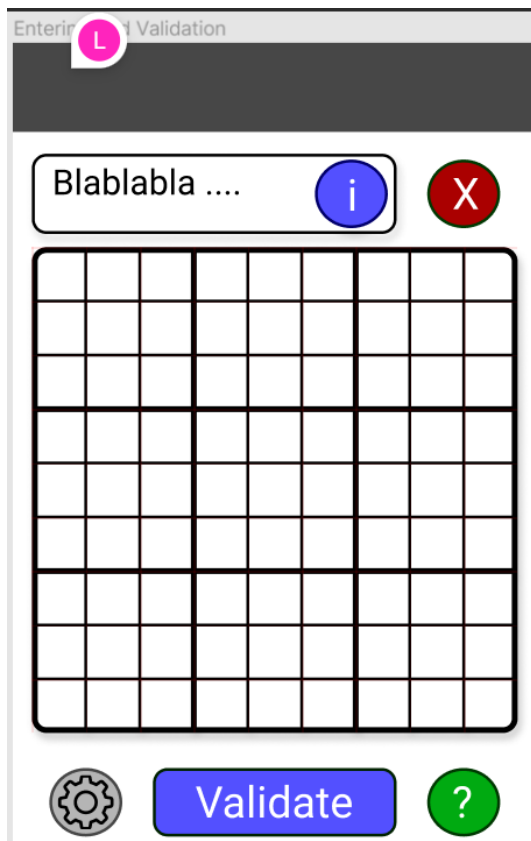


Abbildung 19: Mockup für die Homepage (Erstellt mit Figma)



Abbildung 20: Implementierung der Homepage

In den Abbildungen werden das Mockup sowie die tatsächliche Implementierung der Homepage gezeigt. Hier zeigt sich, dass die Aufteilung der Seite gleichgeblieben ist, auch wenn sich die Abstände der Elemente zueinander geändert haben. Auch die Farben sind ähnlich geblieben, so ist der Knopf für die Validierung beides mal. Hier wird deutlich, dass zwar die groben Designentscheidungen, wie die Struktur der Seite, sowie die ungefähre Farbwahl von dem Mockup in die tatsächliche Implementierung übernommen wurden. Details, wie die genaue Farbwahl, Abstände zwischen den Elementen, usw. wurden im Rahmen der Implementierung jedoch noch angepasst. Auch die Inhalte der Seite wurden an die tatsächlichen Anforderungen angepasst.

Da das Mockup nur für die grobe Struktur der Webseite verwendet wird, werden auch nicht für alle Seiten eigene Mockups erstellt, stattdessen basieren nur die ersten Seiten, welche implementiert werden auf Mockups, alle anderen Seiten verwenden dann das Design weiter, wobei bei Bedarf nur kleine Änderungen getroffen werden. So werden z. B. je nach Seite Elemente hinzugefügt oder weggenommen.

3.5.2 Allgemeines Design

Da alle vorhandenen Seiten ein sehr ähnliches Design aufweisen, sollen zuerst die grundlegenden Designentscheidungen erläutert werden.

3.5.2.1 Kurzinformation

Dieses Designelement dient dazu Nutzer*innen durch die Seite zu führen und dabei festzuhalten, was auf der jeweiligen Seite dargestellt wird, sowie darauf hinzuweisen, welche Knöpfe zu dem nächsten Schritt führen.

3.5.2.2 Sudoku und Kandidaten

Das Sudoku ist ein zentrales Element dieser Anwendung und wird dementsprechend auf allen Seiten verwendet. Deshalb gibt es einige Aufgaben, welche durch diese Komponente erfüllt werden müssen. Dazu zählen z. B. das Eingeben von Werten, das Darstellen von Kandidaten, das Darstellen von Änderungen, welche durch einen Algorithmus bewirkt werden.

Dies hat zur Folge, dass das Sudoku auf der Webseite immer möglichst groß dargestellt werden muss, aber gleichzeitig komplett auf den Bildschirm passen muss, da notwendiges Scrollen die Übersichtlichkeit einschränken würde.

Deshalb wurde das Sudoku so designt, dass es mit dem Bildschirm mitwächst und mitschrumpft. Damit soll erreicht werden, dass der verfügbare Platz möglichst gut ausgenutzt werden kann, ohne dass Teile abgeschnitten werden.

Auch das Anzeigen der Kandidaten ist wie bereits erwähnt eine wichtige Aufgabe des Sudokus im Frontend. Ein Problem hierbei ist, dass die Größe des Sudokus abhängig von der Bildschirmgröße ist und es somit auch sehr klein dargestellt werden kann. Deshalb muss eine Möglichkeit gefunden werden, die Kandidaten, welche noch mal kleiner sind als die Werte der Felder, möglichst klar darzustellen. Deshalb wurde die Entscheidung getroffen, dass die Kandidaten für jede Zahl immer an der gleichen Position in dem Sudoku Feld sein sollen. Die Aufteilung sieht wie folgt aus:

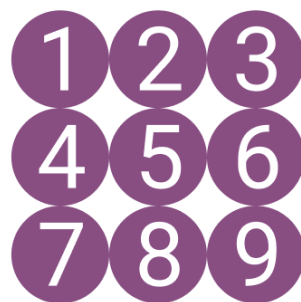


Abbildung 21: Anordnung der Kandidaten innerhalb eines Sudoku Feldes

Die Kandidaten werden mit weißen Ziffern auf farbigem Hintergrund dargestellt, da es auf diese Weise sehr einfach ist zu erkennen, ob sich z. B. an der Position von der 9 angezeigt wird, dass dieser Kandidat existiert, oder nicht. Dies ist vor allem auf kleinen Geräten vorteilhaft, da hier die Ziffern oft nicht mehr gut lesbar sind.

3.5.3 Seiten

Im Folgenden werden die einzelnen Webseiten, aus denen sich die Anwendung zusammensetzt, aufgezeigt und ihre Funktion wird erklärt.

3.5.3.1 Homepage

Wird nur die URL von dem Web-Server eingegeben, wird zuerst die Homepage angezeigt. Wie bereits in den Anforderungen beschrieben, dient diese Seite dazu, die bekannten Werte des Sudokus einzutragen (sh. Abs. 0). Die Seite ist in Abbildung 22 gezeigt.



Willkommen, bitte geben Sie die Nummern in das Sudoku ein und drücken Sie Validier...



		2		1				
							4	
		2						
						-1		
		10						

Validieren

Abbildung 22: Homepage

Wie in Abbildung 22 zu sehen, lassen sich die Nummern in das Sudoku eintragen, wobei diese aber schon auf Plausibilität geprüft und gegebenenfalls Fehler markiert werden. So ist es nicht möglich, dass eine Nummer zweimal in einer Spalte auftaucht, weswegen die beiden Zweier rot hinterlegt sind. Die beiden Nummern Null und Zehn sind ebenfalls nicht möglich, weswegen sie auch markiert werden. Der Knopf für das Validieren, ist hier deaktiviert, da das Sudoku erst eingereicht werden kann, wenn keine Fehler mehr detektiert werden. Somit soll Fehlern, welche durch eine falsche Bedienung der Seite entstehen vorgebeugt werden.

Um das Eingeben eines Sudokus möglichst komfortabel zu gestalten, wurde die Eingabe so strukturiert, dass es auf dem Computer möglich ist mit der Tabulator Taste zeilenweise über die Felder zu springen und sie so auszufüllen. Diese Seite ist auch die einzige, auf der es möglich ist, Werte in das Sudoku einzutragen, auf allen anderen Seiten wurde die Eingabe deaktiviert,

Wurden die bekannten Werte des Sudokus vollständig eingegeben kann der Knopf „Validieren“ gedrückt werden und das Sudoku wird an den Server übermittelt. Wird von dem Server ermittelt, dass es sich nicht um ein echtes Sudoku handelt, wird eine Fehlermeldung angezeigt und die eingetragenen Werte können bearbeitet werden.

3.5.3.2 Validierungs-Seite

Hat man auf der Homepage ein Sudoku eingetragen und dieses wurde vom Server validiert, wird im als nächstes die Bestätigung angezeigt, dass das eingetragene Sudoku valide ist.

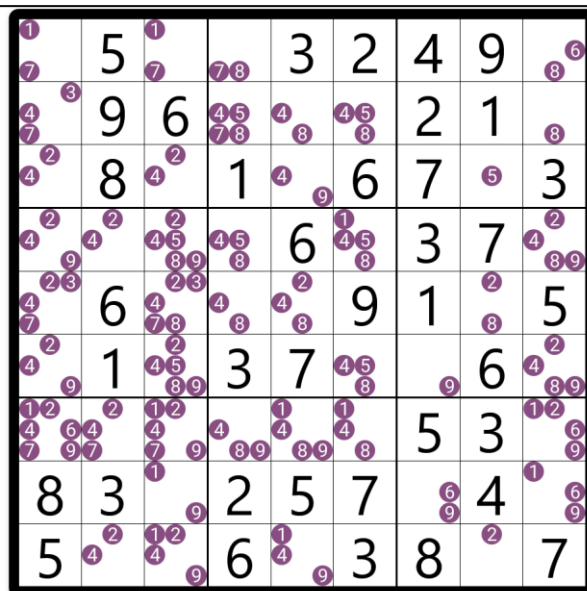
Die Seite sieht wie folgt aus:



☑ Das eingegbene Sudoku ist gültig!



Ihr Sudoku wurde verifiziert, jetzt können Sie beginnen es zu lösen! Drücken Sie Lösen um f...



Kandidaten an-/ausschalten

Lösen

Abbildung 23: Bestätigungsseite, dass eingetragenes Sudoku gültig ist

Neben der Erfolgsmeldung, dass es sich bei dem eingereichten Sudoku um ein Gültiges handelt, wird hier nur die Möglichkeit geboten, die Kandidaten anzeigen zu lassen und mit dem Lösen des Sudokus fortzufahren.

3.5.3.3 Algorithmen Visualisierung

Wurde ein Sudoku eingetragen, durch den Server validiert und dann auf der Validierungs-Seite (sh. Abs. 3.5.3.2) der Knopf „Lösen“ gedrückt, beginnt der Server die Lösungsalgorithmen auf das Sudoku anzuwenden. Wurde ein Algorithmus gefunden, der sich auf das Sudoku anwenden lässt, wird eine Seite angezeigt, auf der Nutzer*innen schrittweise zur Lösung des Algorithmus hingeführt werden.

In den Anforderungen wurde dafür bereits eine Reihe von Schritten vorgegeben. Im Rahmen der Implementierung wurden diese Schritte aber dezent angepasst, um die Implementierung zu vereinfachen.

Hier sollen die implementierten Schritte, sowie Abweichungen von Anforderungen am Beispiel des Algorithmus „offener Dreier“ erläutert werden.



Es gibt einen offenen Dreier in dem markierten Bereich



	7					8		
	2		8			9	5	
					9	6		2
			3		4	2	8	9
		3	9			1	6	4
4			6	1		5		
	4		2	9	6		1	5
							9	6
			5	3		4	2	8

Nächster Hinweis



Abbildung 24: Seite zur Algorithmus Erläuterung, Schritt 1

Wie in Abbildung 24 zu sehen, besteht der erste Schritt daraus, dass benannt wird, welcher Algorithmus auf das Sudoku angewandt werden kann (hier ein offener Dreier). Außerdem wird der Bereich markiert, auf den sich der Algorithmus anwenden lässt. In diesem Fall wird dafür die erste Zeile markiert, abhängig von dem darzustellenden Algorithmus, können aber auch mehrere Zeilen, Spalten und Blöcke markiert werden. Dadurch haben Nutzer*innen die Möglichkeit, bereits selber zu versuchen den Algorithmus anzuwenden.

Wird der Knopf „Nächster Hinweis“ betätigt, wird der nächste Schritt angezeigt.

Der zweite Schritt besteht (wie in den Anforderungen angegeben) darin, dass die Kandidaten eingeblendet werden. Dadurch sollen Nutzer*innen bereits die Möglichkeit haben, den Algorithmus ohne Hilfe durch den Computer anzuwenden und somit ihr eigenes Wissen zu erweitern. (sh. Abbildung 25)

Auch hier führt die Betätigung des Knopfes „Nächster Hinweis“ dazu, dass der nächste Lösungsschritt eingeblendet wird.

1		3
4		6
	8	

Step-by-Step Sudoku

Es gibt einen offenen Dreier in dem markierten Bereich



1	3	7	1	4	5	6	9	4	2	1	2	3	5	8	4	3	1	3
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9	1	2	3	5
1	3	6	9	2	4	5	6	9	4	4	4	5	6	9				

Nächster Hinweis ?

Abbildung 25: Seite zur Algorithmus Erläuterung, Schritt 2

1		3
4		6
	8	

Step-by-Step Sudoku

Mit dem offenen Dreier Algorithmus werden die hervorgehobenen Kan...



1 5 6 9	3 7	1 4 5 6 9	1 4	2 4 5 6	1 2 3 5	8	4	3 1	3
1 3 6	2	1 4 6	8	4 7	6 7	9	5	1 3	3
1 5 8	3 1 8	3 1 4 5 8	1 4 7	4 5 7	9	6	4 7	3 7	2
1 5 6 7	1 5 6 7	1 5 6 7	3	5 7	4	2	8	9	
2 5 7 8	5 8	3	9	2 5 7 8	2 5 7 8	1	6	4	
4		2	6	1	7 8	5		3	3
	3	8 9 7 8 9	2	9	6		7	7	
7 8	3 7 8	3 7 8	1 4 7	4 7 8	1	7	3	1	5
1 2 3 5 7 8	3 1 5 8	3 1 5 7 8	1 4 7	4 7 8	1		9	6	
1 6 9	1 6 9	6 9	5	3	7	4	2	8	

Nächster Hinweis ?

Abbildung 26: Seite zur Algorithmus Erläuterung, Schritt 3

In diesem Schritt (sh. Abbildung 26) werden die betroffenen Kandidaten entsprechend dem Effekt, den der Algorithmus auf sie hat eingefärbt. Hierbei gibt es mehrere Möglichkeiten:

- werden Kandidaten entfernt, werden sie grau hinterlegt
- werden Kandidaten als Wert in das Feld eingetragen, werden sie grün hinterlegt
- werden Kandidaten fest mit einem Feld verbunden werden sie hellblau hinterlegt (am Beispiel oben: Die hellblau markierten Kandidaten müssen in den entsprechenden Feldern sein, können also nicht in anderen Feldern in der Einheit auftauchen)

Dieser Schritt weicht von den Anforderungen ab. Hier wäre gefordert, dass erläutert wird, warum diese Effekte auf die Kandidaten auftreten (sh. Abs. 0). Da eine situationsbedingte Erläuterung aber kompliziert umzusetzen wäre, wird nur der Algorithmus allgemein erläutert und es wird visualisiert, welche Felder und Kandidaten genau betroffen sind.

Auch bei diesem Schritt führt ein Klick auf den Knopf „Nächster Hinweis“ dazu, dass der nächste Schritt angezeigt wird.

1		3
4	6	
	8	

Step-by-Step Sudoku

Mit dem offenen Dreier Algorithmus werden die hervorgehobenen Kandidaten als Kandidat...
i

Kandidaten Berechnen

?

Abbildung 27: Seite zur Algorithmus Erläuterung, Schritt 4

Wie in den Anforderungen beschrieben, werden in diesem Schritt die durch den Algorithmus ermittelten Änderungen auf das Sudoku angewandt. Das bedeutet die zu entfernenden Kandidaten werden entfernt und Kandidaten die als Wert in das Feld eingetragen werden sollen, werden eingetragen. Dabei werden die jeweiligen Felder

entsprechend den Änderungen an ihren Kandidaten hervorgehoben. Dabei gibt es die folgenden Möglichkeiten:

- Wert wird eingetragen: Grün
- Kandidaten werden für das Feld festgelegt: Orange
- Kandidaten werden entfernt: Grau
- Es werden Kandidaten festgelegt und andere entfernt: Braun

Dies ist der letzte Schritt für das Anzeigen der Effekte, die der verwendete Algorithmus auf das Sudoku hat. Dementsprechend steht auf dem unteren Knopf auch nicht mehr „Nächster Hinweis“, sondern „Kandidaten Berechnen“. Durch Betätigen dieses Knopfes kommt man auf die nächste Seite.

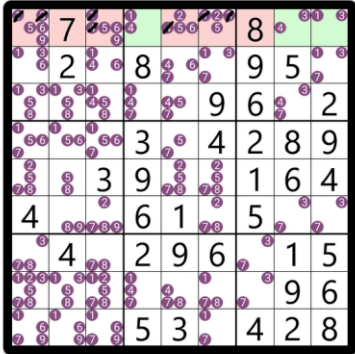
Die Visualisierung der Algorithmen geschieht dabei hauptsächlich über das Frontend mit JavaScript. Von dem Backend werden nur das Sudoku nach dem Anwenden des Algorithmus, sowie Informationen über die durch den Algorithmus getroffenen Änderungen übertragen. Diese werden dann im Frontend entsprechend verarbeitet.

Außerdem existiert auf dieser Seite noch ein grüner Knopf mit einem Fragezeichen. Bei Betätigung dieses Knopfes öffnet sich ein Fenster, in dem der verwendete Algorithmus allgemein erklärt wird. Diese allgemeine Erklärung ist ausführlicher als die kurze Erklärung, welche auf die aktuelle Situation bezogen ist. Dadurch soll der Algorithmus besser verständlich werden. Für den Algorithmus „offener Dreier“ sieht die Erklärung wie folgt aus:

Erklärung

Nackter Dreier

Wenn in drei Feldern einer Einheit nur drei verschiedene Kandidaten auftauchen (und keine anderen) müssen sie sich auf diese Felder verteilen. D. h. sie werden aus den übrigen Feldern der Einheit gelöscht.



Quelle: Rätselbüro Martin Simon, www.thinkgym.de/rätselarten/sudoku/lösungsstrategien-1/ (auch 2,3)

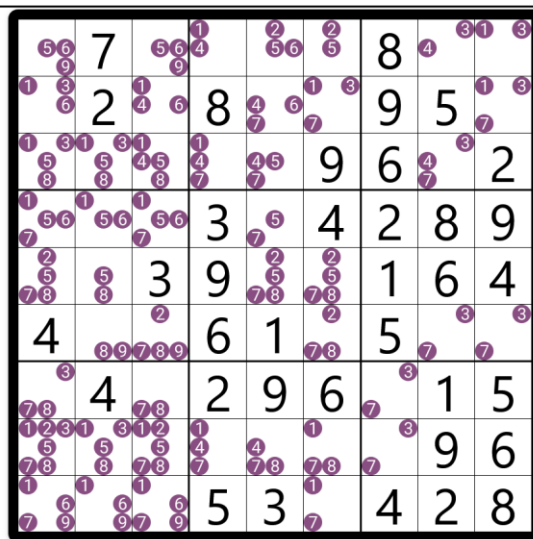
Abbildung 28: Fenster zur genaueren Erklärung der Algorithmen

3.5.3.4 Kandidaten anzeigen

Diese Seite wird nach der Seite für die Algorithmen angezeigt. Sie dient nur dazu, dass der aktuelle Stand, auf dem sich das Sudoku befindet, noch einmal klar und deutlich angezeigt wird. Dementsprechend ist diese Seite auch sehr simpel gehalten.



Nach den Änderungen durch den letzten Algorithmus sehen das Sudoku und die Kandidaten wie folg...



Kandidaten an-/ausschalten

Lösen

Abbildung 29: Seite die den Stand des Sudokus nach der Anwendung eines Algorithmus anzeigt

Hier werden nur die Werte, welche in den Feldern stehen angezeigt. Außerdem gibt es die Möglichkeit die Kandidaten an- und auszuschalten. Wird der Knopf „Lösen“ betätigt wird, wird der Server den nächsten anwendbaren Lösungsalgorithmus suchen und anwenden. Man kommt also wieder zu der Seite, auf der die Lösungsalgorithmen visualisiert werden (sh. Abs. 3.5.3.3)

3.5.3.5 Seite wenn vollständig gelöst

Wenn genug Algorithmen durchgeführt wurden, ist das Sudoku irgendwann fertig gelöst. Die Seite, die dies anzeigt, ist sehr simpel gehalten. Sie zeigt lediglich das Sudoku an, sowie eine Meldung dass es erfolgreich gelöst wurde. Außerdem wird ein Knopf angezeigt, mit dem ein neues Sudoku gestartet werden kann.

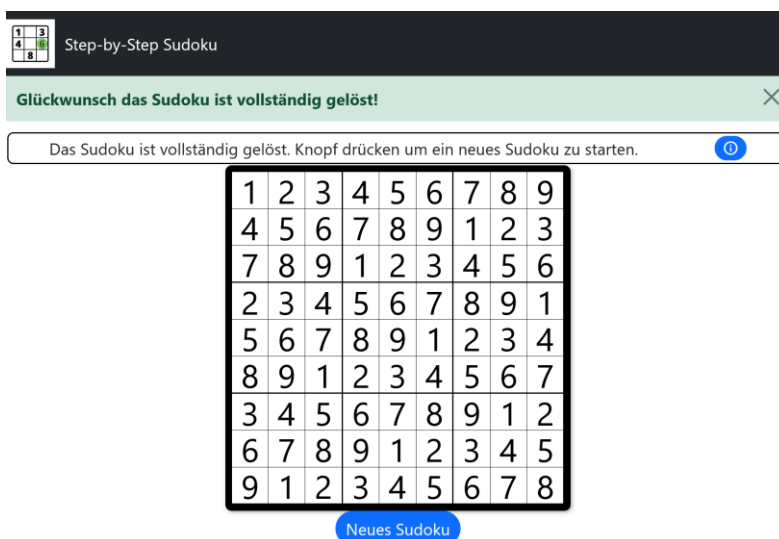


Abbildung 30: Seite die anzeigt, dass das Sudoku vollständig gelöst ist

3.5.3.6 Seite wenn keine Lösung

Wie bereits in Abs. 3.6.1.4 beschrieben, ist es nicht möglich zu garantieren, dass mit den implementierten Algorithmen alle Sudokus gelöst werden können. Deshalb muss es auch eine Seite geben, welche angezeigt wird, falls keiner der implementierten Algorithmen auf das gegebene Sudoku angewendet werden kann. In diesem Fall wird wie in Abbildung 31 gezeigt, der aktuelle Stand, auf dem sich das Sudoku befindet, angezeigt. Außerdem wird die Möglichkeit geboten, die durch das Backtracking ermittelte Lösung anzuzeigen.

Step-by-Step Sudoku

⚠ Das Sudoku kann mit den gegebenen Algorithmen nicht gelöst werden!

Kein Algorithmus konnte das gegebene Sudoku lösen. Sie können sich die Komplettlösung anzeigen l...

1	5	7	8	3	2	4	9	6
3	9	6	7	4	5	2	1	8
2	8	4	1	9	6	7	5	3
9	2	8	5	6	1	3	7	4
7	6	3	4	2	9	1	8	5
4	1	5	3	7	8	9	6	2
6	7	2	9	8	4	5	3	1
8	3	1	2	5	7	6	4	9
5	4	9	6	1	3	8	2	7

Lösung an-/ausschalten Neues Sudoku

Abbildung 31: Seite, wenn das gegeben Sudoku nicht mit den Algorithmen gelöst werden kann

3.5.3.7 Ablauf

Der Ablauf der Seiten sieht dann aus wie in Abbildung 32 dargestellt. Dieses Diagramm betrachtet hierbei einen Ablauf, wie er bei einer sinngemäßen Verwendung der Anwendung möglich ist. Es wäre z. B. auch möglich direkt von der Seite „Validierung“ zu der Seite „Gelöst“ zu kommen, wenn das eingetragene Sudoku bereits komplett gelöst ist. Dies stellt aber nicht den gewünschten Anwendungsbereich dieser Anwendung dar.

So werden zuerst auf der Homepage die Werte in das Sudoku eingetragen, dann wird das Sudoku validiert. Tritt bei der Validierung ein Fehler auf, müssen die Werte auf der Homepage korrigiert werden. Ist das Sudoku validiert, werden als nächstes die Algorithmen angewandt, wie bereits beschrieben teilt sich die Visualisierung hierbei in vier Schritte auf. Danach kommt die Kandidaten Anzeige, in der der aktuelle Stand des Sudokus noch mal klar gezeigt wird. Danach gibt es drei Möglichkeiten. Ist das Sudoku vollständig gelöst, oder lässt sich kein Algorithmus mehr anwenden, werden die entsprechenden Seiten angezeigt. Lässt sich ein Algorithmus anwenden, geht es wieder zur Algorithmen Visualisierung.

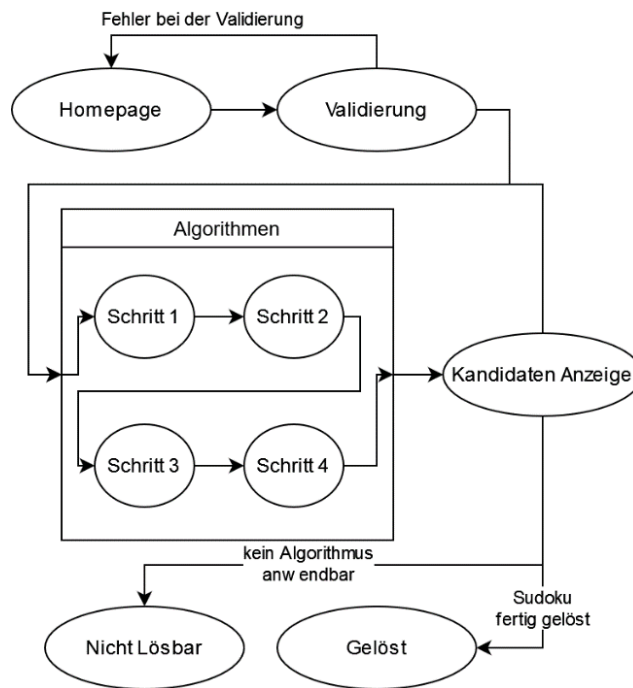
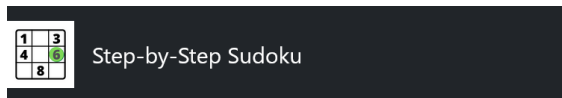


Abbildung 32: Diagramm für den Ablauf der Webseiten

3.5.3.8 Development Tools

Die beiden Webseiten, welche durch diese Anwendung bereitgestellt werden, sehen wie folgt aus. In Abbildung 33 wird die Startseite der der Dev-Tools gezeigt. Auf dieser Seite befinden sich lediglich eine Reihe von Knöpfen. Einen der zu den Sudoku Vorlagen weiterleitet und unter „Algorithmen testen“ eine Reihe an Knöpfen, mit denen sich die spezifischen Algorithmen testen lassen. In Abbildung 33 wird die Seite dargestellt, auf der Vorlagen für Sudokus angezeigt werden, welche einfach verwendet werden können. Dabei besteht jede Vorlage aus einem kurzen Titel, der Visualisierung der Vorlage, sowie einem Knopf, mit dem das entsprechende Sudoku ausgewählt werden kann.

Da dieser Teil weniger für Endnutzer*innen gedacht ist sondern eher als Unterstützung bei der Entwicklung, wurden z. B. die Algorithmen nicht extra schön beschriftet. Da es aber auch einfacher macht, gezielt Algorithmen auszuprobieren und anzuschauen, wird dieser Teil trotzdem in der Anwendung beibehalten und den Endnutzer*innen zur Verfügung gestellt.



Sudoku Vorlagen

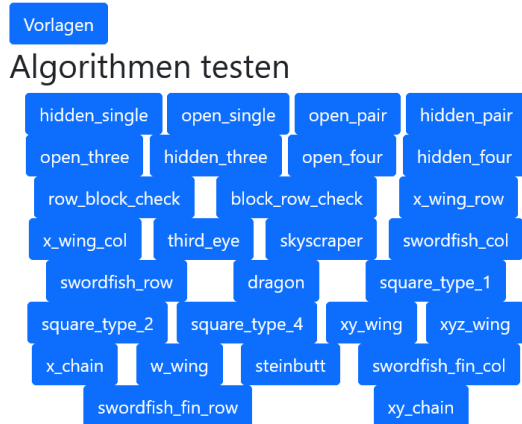
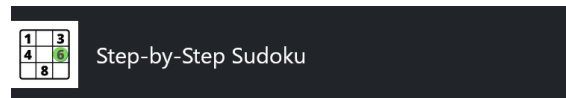


Abbildung 33: Dev-Tools Startseite



Valid

	5			3	2	4	9	
	9	6				2	1	
	8		1		6	7		3
				6		3	7	
	6				9	1		5
	1		3	7			6	
						5	3	
8	3		2	5	7		4	
5			6		3	8		7

Validieren

Abbildung 34: Dev-Tools Sudoku Vorlagen

3.6 Tests

3.6.1 Backend

Der Großteil der Programmlogik ist im Backend implementiert, deshalb ist es wichtig dieses zu testen, um Fehlern vorzubeugen. Die Backend-Tests bestehen zu einem Großteil aus Unittests. (sh. Abs. 2.4)

3.6.1.1 Grundfunktionen testen

Der Backend-Testaufbau ist in den folgenden Kapiteln genauer erklärt. Es werden die Komplexeren Funktionen wie die Lösungsalgorithmen und der Backtracking Algorithmus spezifisch mit Hilfe von Unittest getestet. Der gesamte Ablauf wird dann wie in 3.6.1.4 beschrieben getestet.

3.6.1.2 Backtracking testen

Bei dem Testen des Backtracking Algorithmus werden mehrere Sudokus verwendet, bei denen die Anzahl der Lösungen bekannt ist. Hierbei werden 5 verschiedene Sudokus getestet, diese Sudokus haben folgende Eigenschaften und sollten alle Fälle abdecken:

- Sudoku 1 hat nur eine Lösung und ist damit ein echtes Sudoku
- Sudoku 2 hat genau zwei Lösungen
- Sudoku 3 hat über 2000 Lösungen
- Sudoku 4 hat nur eine Lösung
- Sudoku 5 hat nur eine Lösung, aber es sind nur 17 Zahlen gegeben. Es gibt kein Sudoku, das weniger als 17 Zahlen hat und nur eine Lösung besitzt.

Dieser Test ist erfolgreich und man kann den Backtracking Algorithmus als vollständig getestet ansehen.

3.6.1.3 Algorithmen testen

Bei dem Testen der jeweiligen Algorithmen wurde für jeden Algorithmus ein Sudoku verwendet, auf das der Algorithmus anwendbar ist, diese Sudokus wurden aus [1] entnommen. Dies testet, ob im Allgemeinen ein Algorithmus eine Lösung findet, kann aber die vollständige Korrektheit eines Algorithmus nicht gewährleisten. Um die

Algorithmen so gut es geht zu testen wurde eine weitere Testinstanz entwickelt, die in 3.6.1.4 beschrieben wird.

3.6.1.4 Vollständigkeit testen

Eine Anforderung ist es, dass das Programm in der Lage sein muss, alle Sudokus zu lösen. Um dies zu testen, kann aber nicht jede Möglichkeit, die es für ein Sudoku gibt, getestet werden, da es dafür zu viele Möglichkeiten sind. Laut [23] gibt es ca. 6,7 Trilliarden Möglichkeiten für gültige Sudokus. Werden Sudokus, die durch einfache Methoden, wie z.B. Tauschen von Einheiten voneinander abgeleitet werden können nicht beachtet, gibt es immer noch 5,5 Milliarden verschiedene Möglichkeiten ein Sudoku regelkonform auszufüllen. (vgl. [23])

Da es somit nicht möglich ist jede Möglichkeit für ein Sudoku zu testen, wurde ein Test implementiert, welcher versucht, mithilfe der implementierten Algorithmen eine Reihe an vorgegebenen Sudokus zu lösen. Dieser Test bezieht sich dabei rein auf das Backend, d. h. es werden nur die Funktionen aufgerufen, ohne dabei auf die Visualisierung durch das Frontend einzugehen. Dabei wird das Sudoku nach jedem angewendeten Algorithmus erneut validiert. Dies wird benötigt, da es vorkommen kann, dass Algorithmen, wenn sie Fehler in der Implementierung besitzen, das Sudoku unlösbar machen können. Das Validieren dient also dazu, zu Unterscheiden, ob ein unlösbares Sudoku daran liegt, dass die Algorithmen nicht ausreichen oder daran, dass Algorithmen es zerstört haben.

Am Ende wird ausgegeben, wie viele der gegebenen Sudokus gelöst werden konnten.

Im Gegensatz zu den anderen Backend-Tests handelt es sich hierbei also nicht um einen Unit-Test, sondern um einen Anforderungstest. (sh. Abs. 2.4)

3.6.2 Frontend

Wie bereits erwähnt muss auch das Frontend getestet werden, um sicherzustellen, dass die Anwendung anforderungsgemäß funktioniert. Dafür werden nahezu alle Seiten getestet, außer den Seiten „Nicht Lösbar“ und „Kandidaten Anzeige“. (sh. Abs. 3.5.3)

Auf allen Seiten wird immer getestet, dass die Entsprechenden Modals (wie z. B. das Hilfsfenster (sh. Abs. 3.5.3.3)) ordnungsgemäß funktionieren.

Anders als die Backend-Tests stellen die Frontend-Tests Integrations- und Funktions-Tests anstatt Unittests dar. So werden keine JavaScript Funktionen Explizit getestet, sondern es wird mit dem Frontend interagiert und die entsprechende Reaktion wird ausgewertet. Dafür muss auch der Backend Server laufen.

3.6.2.1 Browser- und Gerätesupport

Da die Anwendung als Web-Anwendung implementiert wird, ist auch das Thema Browsersupport sowie der Support von unterschiedlichen Geräten mit unterschiedlichen Bildschirmgrößen von Bedeutung.

Die Frontend Tests werden alle mit dem Firefox-Browser in der Auflösung 360x800, also in der Größenordnung eines Handybildschirms im Hochformat, durchgeführt werden, kann die Funktionalität durch die Tests natürlich nur in diesem Format sichergestellt werden.

3.6.2.2 Homepage

Die Homepage dient zum Eintragen der Werte (sh. Abs. 3.5.3.1). Deshalb wird für den Test dieser Seite hauptsächlich getestet, ob das Eintragen der Werte, sowie das Markieren von falschen Eingaben, also falsche Werte oder Werte die mehrfach pro Einheit auftauchen, funktioniert wie geplant.

3.6.2.3 Validierungs-Seite

Für diesen Test werden erst automatisiert Sudokus in die Homepage eingetragen und dann wird überprüft, ob der Druck auf den Knopf „Validieren“ die richtige Reaktion liefert.

Für das valide Sudoku soll die Seite angezeigt werden, die aussagt, dass das Sudoku valide ist. Das invalide Sudoku soll weiterhin bearbeitet werden können und es soll eine Warnung angezeigt werden.

3.6.2.4 Algorithmen Visualisierung

Hier wird für jeden Algorithmus die entsprechende Visualisierung getestet. Dies dient dazu sicherzustellen, dass die Übertragung der Algorithmus Rückgaben vom Backend ans Frontend funktioniert und dass alle Schritte korrekt implementiert wurden.

Dafür wird mit Hilfe der Development-Tools-App (sh. Abs. 3.5.3.8) gezielt die Visualisierung für jeden Algorithmus aufgerufen und die Visualisierungs-Schritte werden nacheinander durchgegangen. Dabei wird immer überprüft, dass Felder entsprechend dem gewünschten Ergebnis markiert werden.

3.6.2.5 Gelöst

Um die Seite zu testen, die angezeigt wird, wenn ein Sudoku vollständig gelöst ist, wird ein bereits vollständig gelöstes Sudoku in die Homepage eingetragen. Dann muss nach der Validierungs-Seite, also an der Stelle, an der normalerweise die Algorithmen dargestellt werden, die Meldung kommen, dass das Sudoku vollständig gelöst ist. Dieses Verhalten wird überprüft.

3.6.3 Ergebnisse

Interessant ist neben der Information welche Tests vorhanden sind natürlich auch die Frage, ob sie alle durchlaufen, bzw. welche Rückmeldungen sie liefern.

3.6.3.1 Backend

Bei den Backend-Tests laufen die meisten Unit-Tests (sh. Abs. 3.6.1) fehlerfrei durch.

Der einzige Fehler der auftritt, ist dass der Algorithmus X-Kette bei einem Sudoku, bei dem er eigentlich eine Lösung finden sollte keine Lösung findet, also die Implementierung fehlerhaft ist. Dies sollte eigentlich noch behoben werden, da aber zahlreiche Reparaturversuche keine Besserung brachten oder den Algorithmus nur an anderer Stelle zerstörten und es auch an Zeit mangelte, wurde der Algorithmus X-Kette aus der Liste der Algorithmen, welche zum Lösen der Sudokus verwendet werden, entfernt.

Der Anforderungstest, der testet, inwieweit die gegebenen Algorithmen ausreichen, um verschiedene Sudokus zu lösen kommt zu folgendem Ergebnis:

- Schwer: 0/3 gelöst
- Mittel: 2/2 gelöst

- Variierende Schwierigkeit: 20/26 gelöst

Von 31 gegebenen Sudokus können also 22 gelöst werden.

3.6.3.2 Frontend

Von den Frontend Tests laufen alle Fehlerfrei durch. Das Frontend und die Interaktion zwischen Frontend und Backend laufen also im Rahmen der Testabdeckung so wie gewünscht.

Es gibt nur eine Ausnahme, bei der ein Fehler auftritt, dieser lässt sich aber mit den Informationen aus dem vorherigen Absatz sehr einfach erklären. Dadurch dass der Algorithmus X-Kette aus der Liste der zur Verfügung stehenden Algorithmen entfernt wurde, kann der Server nicht mehr auf diesen Algorithmus zugreifen. Dementsprechend scheitert der Frontend Test, welcher die Algorithmus-Visualisierung für den X-Kette Algorithmus testet daran, dass der zugehörige Algorithmus nicht vorhanden ist und somit logischerweise auch keine Visualisierung existiert, welche getestet werden kann.

4 Fazit

Die Aufgabe einen Sudoku-Helfer als Web-Anwendung zu entwickeln hat sich als sehr vielschichtig herausgestellt. Wie auch dieser Bericht zeigt wird nämlich sowohl eine komplexe Logik im Backend benötigt, als auch eine vernünftige Visualisierung im Frontend, damit es immer verständlich ist, welche Algorithmen warum angewendet werden können.

Im Backend hat sich die Implementierung der Algorithmen als deutlich schwerer als gedacht herausgestellt, da es insbesondere bei den komplexeren Algorithmen nicht einfach möglich ist, genügend Testfälle zu finden, da dafür immer erst Sudokus generiert werden müssten. Dementsprechend ist die Testabdeckung der Unittests für die Algorithmen nicht sehr ausführlich, was zur Folge hat, dass es viele Bugs in den Algorithmen geben kann, welche erst nach und nach auftauchen.

Aber auch die Frontend Visualisierung hat sich als sehr anspruchsvoll herausgestellt, da allein das Darstellen von Sudokus mit Kandidaten nicht einfach ist und einige Kenntnisse über HTML und CSS erfordert.

Aus den Testergebnissen erkennt man, dass das die implementierten Funktionalitäten zu einem großen Teil funktionieren. Lediglich ein Algorithmus ist bekanntermaßen fehlerhaft. Bei den anderen Algorithmen haben sich keine Fehler aufgetan. Jedoch wurde die Anforderung, dass alle Sudokus mit dem Programm lösbar sein sollen nicht erreicht. Es können von gegebenen 31 Sudokus nur 22 gelöst werden.

Dafür wurde das Ziel erreicht, die Algorithmus-Ergebnisse Schritt für Schritt zu visualisieren und somit den Weg zur Lösung des Sudokus aufzuzeigen.

5 Literaturverzeichnis

- [1] M. Simon, „thinkgym.de,“ thinkgym.de, 2019. [Online]. Available: <https://www.thinkgym.de/rätselarten/sudoku/lösungsstrategien-1/>. [Zugriff am 08.06.2022].
- [2] S. Pittet, „The different types of software testing,“ o. J.. [Online]. Available: <https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing>. [Zugriff am 06. Juni 2022].
- [3] Wikipedia, „Front-End und Back-End,“ 15. Februar 2022. [Online]. Available: https://de.wikipedia.org/wiki/Front-End_und_Back-End. [Zugriff am 12. April 2022].
- [4] B. Jung, „Homepage Webhilfe - ASP.NET - Einführung,“ o. J.. [Online]. Available: <https://www.homepage-webhilfe.de/ASP-NET/>. [Zugriff am 23. Mai 2022].
- [5] IONOS SE, „Was ist ASP.NET?,“ 26. August 2021. [Online]. Available: <https://www.ionos.de/digitalguide/websites/web-entwicklung/was-ist-aspnet/>. [Zugriff am 23. Mai 2022].
- [6] Microsoft, „Einführung in Razor Pages in ASP.NET Core,“ 21. Mai 2022. [Online]. Available: <https://docs.microsoft.com/de-de/aspnet/core/razor-pages/?view=aspnetcore-6.0&tabs=visual-studio>. [Zugriff am 23. Mai 2022].
- [7] IONOS SE, „Ruby on Rails: Das MVC-Framework für komplexe Webanwendungen,“ 16. März 2020. [Online]. Available: <https://www.ionos.de/digitalguide/websites/web-entwicklung/ruby-on-rails-mit-wenig-code-zur-eigenen-web-app/>. [Zugriff am 23. Mai 2022].
- [8] Mozilla Foundation, „MDN Webdocs - Django Introduction,“ o. J.. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>. [Zugriff am 24. Mai 2022].
- [9] J.-D. Kranz, „Interpretierte vs. kompilierte Programmiersprachen,“ 27. Februar 2018. [Online]. Available: <https://it-talents.de/it-wissen/interpretierte-vs-kompilierte-programmiersprachen/>. [Zugriff am 25. Mai 2022].
- [10] Django Software Foundation, „Documentation - Templates,“ o. J.. [Online]. Available: <https://docs.djangoproject.com/en/4.0/topics/templates/>. [Zugriff am 27. Mai 2022].
- [11] S. Bose, „Top 5 CSS Frameworks for Developers and Designers,“ 22. Juni 2021. [Online]. Available: <https://www.browserstack.com/guide/top-css-frameworks>. [Zugriff am 27. Mai 2022].
- [12] Stackshare, „Bootstrap vs. Tailwind CSS,“ o. J.. [Online]. Available: <https://stackshare.io/stackups/bootstrap-vs-tailwind-css>. [Zugriff am 27. Mai 2022].

- [13] Brain4ce Education Solutions Pvt. Ltd., „JavaScript vs jQuery: Key Differences You Need to Know,“ 14 April 2022. [Online]. Available: <https://www.edureka.co/blog/javascript-vs-jquery/>. [Zugriff am 30 Mai 2022].
- [14] B. M. GmbH, „jQuery: eine praktische Bibliothek für die Arbeit mit JavaScript,“ o. J.. [Online]. Available: <https://bmu-verlag.de/jquery-eine-praktische-bibliothek-fuer-die-arbeit-mit-javascript/>. [Zugriff am 30 Mai 2022].
- [15] carbon_app, „carbon,“ o. J.. [Online]. Available: <https://carbon.now.sh>. [Zugriff am 30 Mai 2022].
- [16] D. Potschien, „Framework jQuery: die Vorteile und Nachteile,“ 26 Mai 2017. [Online]. Available: <https://www.drweb.de/framework-jquery-vorteile-nachteile-83398/>. [Zugriff am 30 Mai 2022].
- [17] A. Shaw, „Getting Started with Testing in Python,“ o. J.. [Online]. Available: <https://realpython.com/python-testing/>. [Zugriff am 08 Juli 2022].
- [18] F. Schürmeyer, „Selenium WebDriver in Python - die Webbrowser API,“ 26 Dezember 2020. [Online]. Available: <https://hellocoding.de/blog/coding-language/python/selenium-browser-automatisierung>. [Zugriff am 10 Juni 2022].
- [19] S. C. & B. Straub, „Getting Started,“ in *ProGit*, CA, US, Apress, Berkeley, CA, 2014, pp. 1-14.
- [20] Django Software Foundation, „Writing your first Django App, Part 1,“ o. J.. [Online]. Available: <https://docs.djangoproject.com/en/4.0/intro/tutorial01/>. [Zugriff am 09 Juni 2022].
- [21] „101.computing.net,“ 14 10 2017. [Online]. Available: <https://www.101computing.net/backtracking-algorithm-sudoku-solver/>. [Zugriff am 10 06 2022].
- [22] tutorialspoint, „HTML Forms,“ o. J.. [Online]. Available: https://www.tutorialspoint.com/html/html_forms.htm. [Zugriff am 08 Juni 2022].
- [23] Wikipedia, „Sudoku,“ 22 Mai 2022. [Online]. Available: <https://de.wikipedia.org/wiki/Sudoku>. [Zugriff am 08 Juli 2022].
- [24] W3Schools, „HTML Forms,“ o. J.. [Online]. Available: https://www.w3schools.com/html/html_forms.asp. [Zugriff am 08 Juni 2022].