BOSCH
Technik fürs Leben

DHBW
Duale Hochschule
Baden-Württemberg

Sudoku Solver

Studienarbeit

Bachelor of Science

des Studiengangs Informatik

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

Ruben Hartenstein, Annika Harter

10.06.2022

Bearbeitungszeitraum	22.10.2022 - 10.06.2022
Matrikelnummer, Kurs	2746235, 4810277, TINF19ITA
Ausbildungsfirma	Robert Bosch GmbH, Stuttgart
Betreuer	Sebastian Trost
Gutachter	Sebastian Trost

Duale Hochschule Baden-Württemberg, Stuttgart

Ausbildungsbereich Technik, Fachrichtung IT-Automotive

Bericht über die Ausbildung in der betrieblichen Ausbildungsstätte im 5. und 6. Studienhalbjahr.

Name des Studierenden:	Ruben Hartenstein, Annika Harter		
Studienjahrgang:	2019		
Einsatz in Abteilung:			
Standort:	Stuttgart		
Thema:	Sudoku Solver		
Betreuer:	Sebastian Trost		
vom:	22.10.2022	bis:	10.06.2022

Stellungnahme des Betreuers:

Dieser Bericht wurde geprüft und ist sachlich und fachlich richtig.

Ort	Datum	Abteilung, Unterschrift

Selbstständigkeitserklärung des Studenten

Gemäß §5(2) der „Studien- und Prüfungsordnung DHBW Technik“ vom 29. September 2015: Ich habe die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Ort	Datum	Unterschrift

Sperrvermerk

Die vorliegende Studienarbeit mit dem Titel

Sudoku Solver

enthält unternehmensinterne bzw. vertrauliche Informationen der Robert Bosch GmbH, ist deshalb mit einem Sperrvermerk versehen und wird ausschließlich zu Prüfungszwecken am Studiengang Informatik der Dualen Hochschule Baden-Württemberg Stuttgart vorgelegt.

Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungsprozesses und des Evaluationsverfahrens zugänglich gemacht werden, sofern keine anders lautende Genehmigung der Ausbildungsstätte (Robert Bosch GmbH) vorliegt.

Stuttgart, 10.06.2022

Ruben Hartenstein, Annika Harter

Selbstständigkeitserklärung

Ich versichere hiermit, dass ich meine Studienarbeit mit dem Thema: *Sudoku Solver* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Stuttgart, 10.06.2022

Ruben Hartenstein, Annika Harter

Abstract

TODO: deutscher Abstract....

Inhaltsverzeichnis

Abkürzungsverzeichnis

BSD	Berkley Software Distribution
BSP	Board Support Package
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
HTTPS	Hypertext Transport Protocol Secure
MVC	Model View Controller

Abbildungsverzeichnis

Tabellenverzeichnis

Formelgrößenverzeichnis

a	rad	Bedeutung von a
b	rad	Bedeutung von b
λ	rad	Bedeutung von lambda
ϕ	rad	Bedeutung von phi

Formelverzeichnis

Listings

1 Einleitung

In vielen Zeitungen ist eine Vielzahl von Logikrätseln gedruckt. Darunter eines der häufigsten Logikrätsel: das Sudoku. In der üblichen Version ist es das Ziel, ein 9×9-Gitter mit den Ziffern 1 bis 9 zu füllen. Dabei kann das Gitter in drei unterschiedliche Einheiten aufgeteilt werden. Diese Einheiten sind die Spalten, Zeile und Blöcke des Gitters. Ein Block ist eine 3×3-Unterquadrat des Gitters. In jeder Einheit darf jede Ziffer genau einmal vorkommen. Ausgangspunkt ist ein Gitter, in dem bereits mehrere Ziffern vorgegeben sind. Sudokus gibt es in unterschiedlichen Schwierigkeitsgraden.

1.1 Motivation

Sudokus gibt es mittlerweile in vielen verschiedenen Varianten mit modifizierten Regeln. In dieser Arbeit und für den Sudoku Solver wird jedoch nur das klassische Sudoku, wie es in der Einleitung beschrieben wurde, betrachtet. Es gibt verschiedene Schwierigkeitsstufen von Sudokus. Meistens werden Sudokus mit vielen vorgegebenen Zahlen als einfach eingestuft und Sudokus mit wenigen vorgegebenen Zahlen als schwierig. Ein weiterer Punkt, der Einfluss auf die Schwierigkeitsstufe eines Sudokus hat, ist die Anordnung der vorgegebenen Ziffern.

Wer schon einmal versucht hat ein schweres Sudoku zu lösen, ist vielleicht auch bereits an seine Grenzen gekommen. Gerade bei schwierige Sudokus mit wenigen vorgebenden Zahlen kann es schwierig sein selbst eine passende Lösungsstrategie für das Sudoku zu finden. Oft sind die Lösungsstrategie schwierig zu verstehen und anzuwenden und es lassen sich damit nur Kandidaten eliminieren. Bei einfachen Sudokus werden diese Lösungsstrategien meist unbewusst und intuitiv eingesetzt. In dieser Studienarbeit soll ein Web Frontend entwickelt werden, auf dem ein Bearbeiter des Sudokus unterstützt wird, eine Lösung zu finden.

Das Ziel dieser Arbeit ist es diese schwierige Lösungsstrategie sichtbar zu machen und über mehrere Stufen von Hilfestellungen den Bearbeiter des Sudokus zu zeigen, wel-

che Lösungsstrategie im nächsten Schritt anzuwenden sind und an welcher Stelle des Sudokugitters.

1.2 Forschungsstand

Im Internet findet man immer wieder Websites mit dem Angebot Sudokus zu lösen. Hierbei können die angegebenen Zahlen eingegeben werden und das Sudoku wird korrekt vervollständigt, falls es eine eindeutige Lösung gibt. Daraus kann der Anwender jedoch keinen Nutzen ziehen. Des Weiteren gibt es Websites mit Lösungsstrategien, die bei schwierigeren Sudokus angewendet werden müssen.

Das Beherrschen der Lösungsstrategien besitzt einen großen Stellenwert. Einfache Sudokus lassen sich meist noch intuitiv und durch konzentriertes Absuchen lösen. Die beiden dafür ausreichenden Anfängertechniken heißen „Nackter Single“ und unter „Versteckter Single“ und werden im Verlauf dieser Arbeit noch erläutert. Sie werden mehrheitlich intuitiv und unbewusst eingesetzt, wenn für ein bestimmtes Feld nur eine Zahl möglich ist oder wenn eine bestimmte Zahl nur in ein einziges Feld passt. Bei kniffligeren Sudokus müssen gewissenhaft Notizen gemacht werden. Dafür wird die Methode der „Kandidaten“ verwendet. Damit lassen sich die Zusammenhänge am besten beschreiben. Manche Situationen in einem anspruchsvollen Sudoku erfordern sehr komplexe Lösungsansätze, die nicht immer so einfach zu verstehen sind.

1.3 Forschungsfrage

Für diese Studienarbeit ist die Zielsetzung ein Sudoku Solver zu entwickeln. Ziel ist es nicht ein Sudoku automatisch zu lösen, sondern dem Benutzer Schritt für Schritt den Weg zur Lösung aufzuzeigen. Auf dieser Basis wird zunächst geprüft, ob das Sudoku eindeutig lösbar ist. Ist dies der Fall, muss der Nutzer die Möglichkeit haben sich Anweisungen geben zu lassen, wie man weiter vorgehen könnte und welche Lösungsstrategie der Benutzer anwenden muss, damit man das Sudoku lösen kann. Das heißt, dass die Zahlen auf Basis der bereits vorhandenen Zahlen und Kandidaten mit ein wenig überlegen ermittelt und erklärt werden kann, anstatt einfach eine korrekte Zahl zu zeigen, damit der Benutzer das Sudoku lösen kann. Die Lösungsstrategien sollen mittels des aktuellen Rätels erläutert und von der Software angewandt werden können. Diese Anweisungen, im folgenden auch Hilfestellungen genannt, sind in vier Varianten unterteilt.

Für die erste Hilfestellung soll das Programm dem Benutzer die anzuwendende Strategie nennen und den Bereich hervorheben, auf dem die Strategie angewendet werden soll. Wenn diese Hilfestellung nicht ausreichend ist, dann markiert das Programm die konkreten Zellen und Kandidaten, die von der Strategie betroffen sind. Mit der dritten Hilfestellung soll eine Erläuterung gezeigt werden, warum bestimmte Kandidaten laut Strategie gestrichen werden können. Mit der letzten Hilfestellungen löscht das Programm den betroffenen oder mehrere betroffene Kandidaten.

Um ein lösbares Sudoku beenden zu können, müssen etwa 25 Lösungsstrategien implementiert werden.

Das Programm soll auf einem Ubuntu Server mit Apache und Django laufen. Hier gibt es des Weiteren noch die Anforderung, dass das Frontend responsiv sein soll und immer eine benutzeroptimierte Darstellungen liefert.

2 Rahmenbedingungen

Für das Projekt werden verschiedene Rahmenbedingungen festgelegt. Dazu gehört die Programmiersprache, indem der Sudoku Solver programmiert werden soll, ein Webserver auf dem das Programm später laufen wird und ein Framework, dass als Gerüst für das Programm auf dem Webserver fungiert. Wichtig ist, dass das Framework auf die Programmiersprache angepasst ist, um Probleme zu vermeiden. Im Zuge der Recherchen wurden verschiedenen Webserver in Betracht gezogen und besonders die Webserver Apache HTTP Server und Nginx Server aufgrund ihrer Beliebtheit miteinander verglichen. Auch bei den Frameworks wurden zwei besonders in Betracht gezogen und evaluiert, welches besser geeignet ist für die Studienarbeit.

2.1 Programmiersprachen

2.1.1 Python

Für die Implementierung der Strategien und generell im Backend wurde sich für die Programmiersprache Python entschieden. Python ist eine höhere, universelle Sprache und ist als eine sehr vielseitige Programmiersprache anerkannt. Mit Python kann ein objektorientierter, aspektorientierter oder funktionaler Programmierstil umgesetzt werden und die Typisierung funktioniert dynamisch. Des weiteren wird Python oft bei der Datenanalyse und bei der Verarbeitung großer Datenmenge verwendet. Zusätzlich spielt die Laufzeit zunächst nur eine untergeordnete Rolle und ist nicht sicherheitsrelevant, wodurch bei der Nutzung von Python keine größeren Probleme entstehen sollten. Ein weiterer Vorteil von Python ist die große Standardbibliothek, die falls nötig durch Module erweitert werden kann. Noch ein Vorteil ist die hohe Kompatibilität mit mehreren Frameworks. Im Laufe des Kapitels werden noch zwei Frameworks vorgestellt, die für das Projekt in Betracht gezogen worden.

Für das Backend ist eine Objektorientierte Umsetzung vorgesehen. die mit Python umgesetzt werden kann. Zudem ist Python einfach zu lesen und klar strukturiert. Auch für die

Umsetzung der Lösungsstrategien bietet sich Python durch Module für mathematische Berechnungen und Datenwissenschaft an.

In diesem Projekt wird mit der Version 3.10 von Python gearbeitet. [Ern20] [Sar20]

2.1.2 JavaScript

Im Frontend wurde sich für JavaScript mit jQuery entschieden. JavaScript ist eine der meist verwendeten Programmiersprachen der Welt. Als JavaScript erfunden wurde, ging man davon aus, dass es für kurze Codeschnipsel verwendet wird eingebettet in eine Webseite. Es ist eine Skriptsprache und wurde für die Benutzerinteraktion entwickelt. Mithilfe von JavaScript lassen sich Inhalte verändern oder neu generieren. Für das Projekt ist die Interaktion mit dem Benutzer von entscheidender Bedeutung, da er ständig neue Zahlen oder Kandidaten eingeben kann.

Diese Anforderungen lassen sich mit JavaScript umsetzten. Daher wurde sich beim Frontenddevolpment für JavaScript mit der Erweiterung von jQuery entschieden.

jQuery

Neben JavaScript wird die freie Programmbibliothek jQuery für das Frontend verwendet. jQuery bietet die Funktionalität der Document Object Model Navigation und Manipulation sowie ein erweitertes Event-System. Mit jQuery wird die JavaScript Programmierung vereinfacht und viele Methoden können in einer einzelnen Zeile von Code umgesetzt werden. Ein weiterer Vorteil ist, dass auch jQuery einfach erweitert werden kann und sehr populär ist. https://www.w3schools.com/jquery/jquery_intro.asp

2.2 Webserver

Ein Webserver ist in der Regel ein Server, der zur Verbreitung von Webinhalten im Inter- oder Intranet dient. Die jeweiligen Informationen und Dokumentationen können demnach weltweit oder firmenintern erreicht werden. Damit eine Website jederzeit erreichbar sein kann, muss der Webserver permanent online sein. Der Rechner, auf dem der Webserver läuft, wird als Host bezeichnet. Der Webserver ist für die zuverlässige Übertragung von statischen, wie beispielsweise von unveränderlichen Hypertext Markup Language (HTML)-Dateien, aber auch von dynamischen Dateien verantwortlich. Für dynamische Dateien muss

der Webserver vor der Antwort Programmcode ausführen. Dieser Programmcode wird in diesem Fall in der Programmiersprache Python geschrieben. Mittels einer Cascading Style Sheets (CSS) Datei können die Inhalte der dynamischen Dateien weitestgehend von den Darstellungsvorgaben getrennt werden. In der HTML Datei wird folglich nur die inhaltliche Gliederung definiert und in der CSS Datei die Darstellung, wie etwa Farben oder Layout. Für die Übermittlung wird das Übertragungsprotokoll Hypertext Transport Protocol (HTTP) oder die verschlüsselte Variante Hypertext Transport Protocol Secure (HTTPS) verwendet. Ein Webserver ist in der Lage, die Inhalte auf viele verschiedene Rechner gleichzeitig zu übermitteln. Wie viele Nutzeranfragen (Requests) ein Server bearbeiten kann, hängt von der Hardware und der Auslastung des Hosts ab.

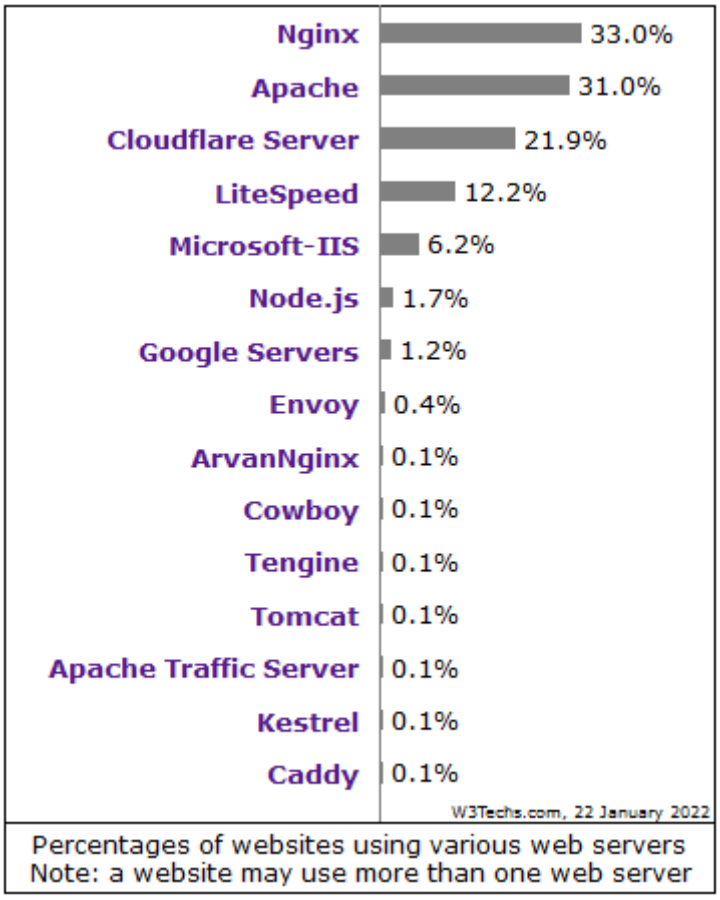

Abbildung 2.1: Benutzungsstatistiken von Webservern [W3t]

Am verbreiteten sind unter anderem die Webserver Apache HTTP Server und nginx, wie in die Statistik 2.1 dargestellt. Beide Programme sind freie Software und wurden daher für das Projekt in Betracht gezogen. Im Folgenden werden diese beide Programme verglichen.

2.2.1 Apache HTTP Server

Der Apache Webserver wurde erstmals 1195 veröffentlicht und hat sich schnell zu einem der beliebtesten Webserver entwickelt. Er unterstützt neben Unix und Linux noch eine Vielzahl an weiteren Betriebssystemen.

Der Apache Server ist modular aufgebaut, wodurch benötigte Funktionen, die der Server nicht nativ bereitstellt, durch Module importiert werden können. Das Erstellen dynamischer Webseiten wird mittels serverseitiger Skriptsprachen bewerkstelligt. Über ein Modul kann der entsprechende Interpreter in den Server integriert werden.

Beim Apache-Webserver wird ein Ansatz verfolgt, bei dem jede Clientanfrage von einem separaten Prozess oder Thread bearbeitet wird. Dadurch werden Prozesse, die Schreib- oder Leseoperationen erfordern, nacheinander abgearbeitet und es kann passieren, dass ein Request in der Warteschlange verweilen muss, bis der vorherige Request durchgeführt werden konnte. Damit man dieses Problem umgehen kann, gibt es die Möglichkeit, mehrere Single-Threading-Prozesse gleichzeitig zu starten. Diese Strategie ist jedoch mit einem hohen Ressourcenaufwand verbunden. Um dies zu vermeiden, kommen Multi-Threading-Mechanismen zum Einsatz. Für die parallele Abfrage von Clientanfragen gibt es verschiedene Multi-Processing-Module, die integriert werden können.

2.2.2 Nginx Webserver

Der Marktanteil von Nginx ist in den letzten Jahren kontinuierlich gestiegen, weshalb auch dieser Webserver für die Studienarbeit in Betracht gezogen wurde. Nginx wurde erstmals 2004 veröffentlicht und ist wie der Apache Server auch mit diversen Betriebssystemen kompatibel.

Wie Apache ist auch Nginx modular aufgebaut und verschiedene Funktionen können über Module bereitgestellt werden. Zu diesen Modulen gehört jedoch nicht die Option, Interpreter für eine Programmiersprache entsprechend in den Webserver zu integrieren. Es wird also ein weiterer Anwendungsserver benötigt. Für kleine Webprojekte ist das ein Mehraufwand, der nicht immer eingegangen werden muss.

Dieser Server zeichnet sich durch eine hohe Performance aus. Dabei können eine möglichst große Anzahl an Clients gleichzeitig bedient werden und der Ressourcenverbrauch trotzdem gering gehalten. Durch die ereignisorientierte Architektur können Client-Anfragen asynchron bearbeitet werden, wodurch Arbeitsspeicher und Zeit gespart werden kann. Die Nebenläufigkeit ist realisiert, ohne dass für jede neue Verbindung ein zusätzlicher

Prozess oder Thread benötigt wird. Die Stärke dieser Architektur zeigt sich bei großen Webprojekten.

2.2.3 Fazit

Die folgende Tabelle 2.1 zeigt die Unterschiede und Gemeinsamkeiten der beiden Webserver auf. Der ausschlaggebende Grund warum sich für einen Apache Server entschieden wurde, ist das Merkmal der dynamischen Webinhalte.

Merkmal	Apache	NGINX
Funktion	Webserver, Proxy-Server	Webserver, Proxy-Server, E-Mail-Proxy, Load-Balancer
Betriebssystem	Alle unixoiden Plattformen, Windows	FreeBSD, Linux, Solaris, IBM AIX, HP-UX, macOS, Windows
Lizenz	Apache License v2.0	BSD-Lizenz
Entwickler	Apache Software Foundation	Nginx, Inc.
Statische Webinhalte	Ja	Ja
Dynamische Webinhalte	Ja	Nein
Software-Architektur	Prozess-/threadbasiert	Eventgesteuert

Tabelle 2.1: Vergleich der Webserver Apache HTTP Server und Nginx Webserver

Der Apache Webserver bietet eine breite Möglichkeit an Modulen, um die Software zu erweitern. Der ausschlaggebende Grund, warum sich in diesem Projekt für den Apache HTTP Server entschieden wurde, ist die Möglichkeit Interpreter für Programmiersprachen über ein Modul direkt in den Webserver zu integrieren. Zudem muss für das Projekt nur ein Client bedient werden. Dadurch wird für das Anzeigen von dynamischen Inhalten kein separaten Anwendungsserver benötigt. Damit bietet der Apache HTTP Server eine bequemere Lösung für kleine Websites, deren Inhalt dynamisch erzeugt wird.

2.3 Framework

Ein Framework ist ein Programmiergerüst. Es wird insbesondere bei komponentenbasierten und bei der objektorientierten Softwareentwicklung verwendet. Ein Framework ist kein fertiges Programm, sondern stellt den Rahmen, in der eine Anwendung erstellt werden soll, zur Verfügung. Frameworks lassen sich in Typen gliedern, wobei es nicht immer eine

strikte Trennung gibt. Zu diesen Typen gehören beispielsweise Application Frameworks oder Webframeworks.

Die beiden Frameworks, Django und Flask, die für das Projekt in Betracht gezogen wurden, gehören zu den beliebtesten Webframeworks für Python und sind für die Entwicklung von dynamischen Webanwendungen ausgelegt.

2.3.1 Django

Django ist ein kostenloses Open Source Webframework und kann plattformübergreifend genutzt werden. Es folgt dem MVC Entwurfsmuster, das auch in der Studienarbeit genutzt wird. Der Fokus von Django liegt in einer schnellen Entwicklung mit weniger Code. Mit dem Framework kommen viele bereits eingebaute Pakete, die genutzt werden können. Dazu gehört unter anderem die integrierte Anbindungen an verschiedenen Datenbanksysteme. Django hat verschiedene Mechanismen integriert, um die Sicherheit der Website zu garantieren. So können SQL Injektionen oder Cross-site Scripting ausgeschlossen werden.

Ein weiterer Vorteil von Django ist die große und aktive Community, wodurch viel Fragen schnell beantwortet werden können und es gibt es eine ausführliche Dokumentation zu dem Framework.

2.3.2 Flask

Webseiten die Flask als Framework nutzten sind meistens einseitige Anwendungen. Flask ist im Gegensatz zu Django modular und eher minimalistisch aufgebaut. Daher muss in der Entwicklung eher mit externen Bibliotheken gearbeitet werden. Man nennt Flask deshalb auch ein Mikroframework. Für die Sicherheit der Website gibt es die Flask-Security Bibliothek, die dieselben Mechanismen wie Django beinhaltet.

Aus diesen Gründen ist Flask eher für kleinere Projekte geeignet, die benutzerdefinierte Komponenten erfordern oder für das Prototyping. Die Anzahl der Webseiten, die Flask als Framework nutzen, steigt stetig weiter. Der Vorteil, warum Flask immer beliebter wird ist die Kontrolle, die man über das Projekt bekommt, indem die Komponenten selbst bestimmt.

2.3.3 Fazit

Beide Frameworks wären für das Projekt geeignet. Die Django Community ist größer und aktiver als die von Flask. Daher gibt es mehr Informationen über Django. Jedoch wird Flask mittlerweile in mehreren Projekten genutzt. In der Tabelle 2.2 sind die wichtigsten Eigenschaften nochmals gegenübergestellt.

Merkmal	Django	Flask
Use cases	mittelgroße bis große Projekte	kleine bis mittelgroße Projekte
Community	große, aktive Community	kleinere Community
Packages	built-in Packages	modular
Performance	gut	gut

Tabelle 2.2: Vergleich der Frameworks Django und Flask

Aufgrund der Leichtigkeit wurde sich bei den Frameworks für Flask entschieden. Da es bei der Performance keine großen Unterschiede zwischen Django und Flask gibt, ist es als Merkmal für die Entscheidung eines Frameworks nicht relevant. Für den Sudoku Solver wird nur eine einseitige Anwendung ohne Datenbankanbindung benötigt, wodurch Flask das geeignetere Framework ist. Auch der modulare Aufbau spricht für Flask, da nur wenige zusätzliche Pakete genutzt werden.

Das Entwurfsmuster MVC wird im Framework Flask übernommen.

2.4 Entwurfsmuster

Wie im Abschnitt der Frameworks schon erwähnt, wird in diesem Projekt eine bestimmte Softwarearchitektur verwendet. Damit wird einem Projekt die grundlegenden Komponenten und das Zusammenspiel dieser visualisiert. Damit ist es einfacher nicht-funktionale Eigenschaften wie die Modifizierbarkeit, Wartbarkeit oder Sicherheit darzustellen und zu verstehen.

Das Framework Django arbeitet immer mit dem MVC Architekturmuster. Da aber Flask als Framework genutzt wird, muss diese Architektur selbst implementiert werden. Im Folgenden wird das Architekturmuster mit seinen drei Komponenten kurz beschrieben *(und wie es in der Studienarbeit umgesetzt wurde)*.

2.4.1 Model View Controller

Das Ziel dieser Entwurfsarchitektur ist es, spätere Änderungen oder Erweiterungen zu vereinfachen und die Wiederverwendbarkeit der einzelnen Komponenten zu steigern. In der Abbildung 2.2 ist die Architektur visualisiert. Durchgezogene Linien stehen hierbei für eine direkte Assoziation und gestichelte für eine indirekte Assoziation.

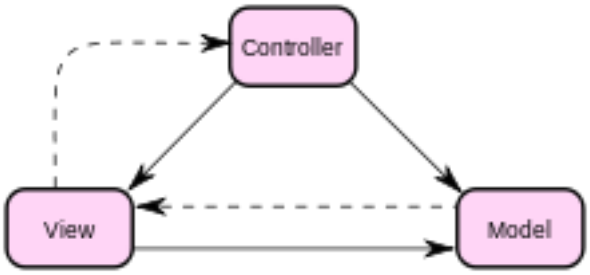

Abbildung 2.2: MVC Konzept

Model

Im Model sind die Daten enthalten, die dargestellt werden sollen. In Falle des Sudoku Solvers ist hier das Sudoku Board gespeichert.

View

In der View oder der Präsentation werden die Daten aus dem Model dargestellt. Außerdem wird in der View die Benutzerinteraktion geregelt. Die View kennt die Daten aus dem Model, soll diese aber nicht bearbeiten oder verändern und ist völlig unabhängig von dem Controller.

Controller

Im Controller werden das Model und die View verwaltet. Die View unterrichtet den Controller von Benutzerinteraktionen. Im Controller wird diese dann ausgewertet und gegebenenfalls das Model und die View angepasst. In dieser Komponente sind die Lösungsstrategien für den Sudoku Solver implementiert.

3 Sudoku

Nachdem in der Einleitung schon kurz auf das Spielfeld eines Sudokus und die Regeln eingegangen wurde, werden beides in diesem Kapitel noch einmal ausführlicher behandelt. Es wird auch auf Elemente des Spielfelds eingegangen, die relevant sind für die Strategien und wie das Spielfeld in dem Programm aufgebaut ist.

3.1 Spielfeld Aufteilung

Das Spielfeld auf dem ein Sudoku gespielt wird besteht aus einem 9x9 Gitter und damit aus 81 einzelnen Zellen. Diese Gitter lässt sich nochmal aufteilen mit drei verschiedenen **Units**. Jede **Unit** besteht immer aus 9 Zellen. Damit kann ein Sudoku also nochmal in 27 verschiedene Zellen aufgeteilt werden. Wie diese Units

3.1.1 Zeile

Eine Zeile in einem Sudokuspielfeld sind Zellen die horizontal zueinander ausgerichtet sind. In der Abbildung 3.1 wurde die oberste Zeile beispielhaft markiert.

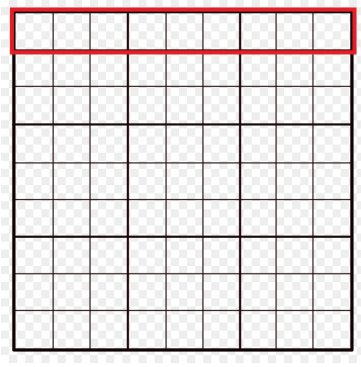

Abbildung 3.1: Darstellung des Sudokugitters mit der Markierung einer Zeile

3.1.2 Spalte

Analog zu einer Reihe sind Spalten Zellen die vertikal zueinander ausgerichtet sind. In der Abbildung 3.2 wurde beispielhaft eine Spalte markiert.

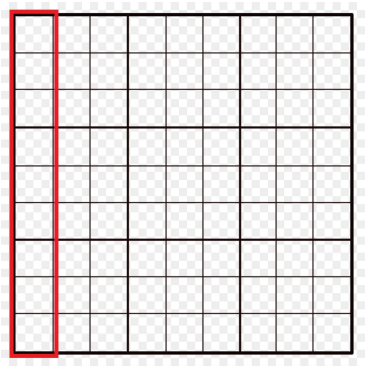

Abbildung 3.2: Darstellung des Sudokugitters mit der Markierung einer Spalte

3.1.3 Box/Block

Boxen bestehen aus 3x3 Zellen. Boxen können aber nicht einfach so an jeder beliebigen Stelle eines Sudokus erstellt werden. Der rechte untere Wert einer Box muss sowohl in der x-Achse durch drei teilbar sein, wie in der y-Achse. In der Abbildung 3.3 wurde eine Box markiert. Die markierten Zellen können kein Teil einer weiteren Box mehr sein. Wie in der Abbildung 3.3 zu erkennen ist sind die Boxen auch im Frontend durch dickere Linien voneinander abgegrenzt.

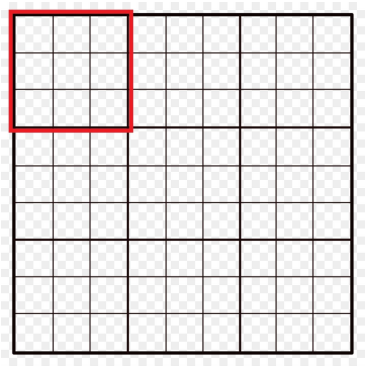

Abbildung 3.3: Darstellung des Sudokugitters mit der Aufteilung in 9 Boxen

3.2 Regeln

Für die Regeln ergeben sich einfache Bedingungen wenn man die Aufteilung des Spielfeldes kennt. In jeder Zeile dürfen die Zahlen eins bis neun nur einmal vorkommen. Genau dieselbe Bedingung gibt es für eine Spalte und für einen Block.

Am Ende kommen die Zahlen eins bis neun also jeweils neunmal auf dem gesamten Sudokugitter vor und in jedem Block, jeder Zeile und jeder Zeile genau einmal.

Damit man das Sudokugitter ausfüllen kann sind zu Spielbeginn einige Ziffern vorgegeben. Mithilfe dieser Ziffern ist es möglich mit den gerade erklärten Regeln das ganze Sudokugitter zu füllen. [Sud22]

3.3 Strategie

Um ein Sudoku zu lösen können verschiedene Strategien umgesetzt werden. Das Ziel einer Strategie ist es hierbei eine passende Zahl in das Sudoku einzufügen oder mögliche Kandidaten zu eliminieren. Es gibt zwei Strategien um direkt Zahlen einfügen zu können, nämlich den Hidden Single und den Naked Single. Beide Strategien können oft intuitiv und ohne eine große Erklärung anwenden. Weitere leichte Strategien wie ein Hidden Double oder ein Naked Double können mithilfe der Kandidaten-Notation gefunden und angewendet werden.

Insgesamt gibt es über 20 Strategien die bei schweren Sudokus angewendet werden können. Zum Teil sind die Strategien für das menschliche Auge schwer zu entdecken und auch schwer zu erläutern. Wichtig bei diesen schweren Strategien ist eine gewissenhafte Ausführung der Kandidaten-Notation.[Mar]

3.3.1 Lösungshilfen: Kandidaten-Notation

Kandidaten sind Zahlen in einer Zelle des Sudokugitters, die die Möglichkeit besitzen in diese Zelle eingetragen zu werden. Das Problem liegt aber dabei, dass mehrere Zahlen in diese Zelle können, weil es noch nicht genügend andere feste Zahlen gibt, die bereits eingetragen wurden. Wenn man dabei jede Zelle einzeln betrachtet geben die Kandidaten keinen weiteren Nutzen. Jedoch, wenn die Kandidaten mehrerer Zellen in Bezug zueinander angeschaut werden, lassen sich Zusammenhänge erkennen. Aufgrund dieser Erkenntnisse wurden weitere Strategien entwickelt.

Sobald eine Zahl mittels des Hidden Single oder des Naked Single eingetragen wird, müssen die Kandidaten aktualisiert werden. Es besteht die Möglichkeit, dass man einige Kandidaten die denselben Wert haben wie die neu eingetragene Zahl und in Zellen stehen, die eine Verbindung über eine Reihe, Spalte oder einen Block haben zu der Zelle in der die Zahl eingetragen wurde, herausgestrichen werden können.

4 Die Mathematik hinter Sudoku

In diesem Kapitel sollen einige Mathematische Fragen über Sudokus geklärt werden. Dazu gehören Fragen zur eindeutigen Lösbarkeit und wie diese bewiesen werden kann oder wie viele mögliche fertige Sudokus es denn überhaupt gibt.

4.1 Abzählfragen

Ein trivialer Ansatz, um die Frage zu beantworten wie viele vollständig ausgefüllten 9×9 Standard-Sudokus es gibt, könnte sein zeilenweise von links nach rechts oder von oben nach unten. Damit würden sich pro Zeile, Spalte oder Block 9! Möglichkeiten ergeben. Daraus ergeben sich

$$(9!)9 = 1{,}1 \cdot 10^{50} \tag{4.1}$$

Möglichkeiten ein Sudokugitter auszufüllen. Von diesen ausgefüllten Gittern entsprechen aber nicht alle der Konvention, dass neben den Zeilen auch die Reihen und Boxen jeweils nur einmal eine Ziffer von 1-9 beinhalten dürfen.

Felgenhauer und Jarvis hatten dieselbe Herangehensweise. Als nächstes haben sie sich die immer drei Reihen nebeneinander angeschaut, oder eben drei Zeilen untereinander. Im Grunde haben die Möglichkeiten von drei linear verbundenen Blöcken die möglichen Lösungen errechnet. Dabei sind gibt es

$$2 \cdot (3!)6 + 18 \cdot 3 \cdot (3!)^6 = 56 \cdot (3!)^6 = 2612736 \tag{4.2}$$

Erweiterungen einer bereits ausgefüllten Reihe.

Damit gibt es

$$9! \cdot 2612736 = 948109639680 \tag{4.3}$$

Möglichkeiten die oberen drei Reihen eines Sudokugitters auszufüllen.

Aus diesen 2612736 Möglichkeiten wollen Felgenhauer und Jarvis dann versuchen die verbleibenden Blöcke des Sudokugitters mittels der Brute Force Methode auszufüllen. Da es mit den 2612736 Möglichkeiten immer noch ein Riesen Aufwand wäre, haben sich Felgenhauer und Jarvis drei Reduktionen überlegt um die Anzahl an Möglichkeiten die ersten drei Reihen auszufüllen zu reduzieren.

Auf diese 2612736 Möglichkeiten haben Jarvis und Felgenhauer drei mögliche Reduktionen angewendet: Die Lexikographische Reduktion, die Reduktion der Permutationen und die Reduktion der Spalten. Durch alle diese Reduktionen werden Sudokus mit Lösungen die sich durch Spiegeln, Drehen oder durch symmetrische Eigenschaften doppeln lassen, herausgefiltert. Diese Reduktionen sind auf den ersten Blick nicht einfach zu verstehen.

Nach den Reduktionen bleiben 44 Möglichkeiten die Zeilen zwei und drei auszufüllen. Danach muss herausgefunden werden wie viele Konfigurationen eines Sudokus es gibt, diese drei Zeilen zu vervollständigen.

Felgenhauer und Jarvis haben in ihrer Veröffentlichung von 2006 die These aufgestellt, dass es 6.670.903.752.021.072.936.960 also ca 6,7 Trilliaden verschiedene, vollständig ausgefüllte Standard-Sudokus Rästel gibt. Nach der Reduktion von symmetrischen Lösungen bleiben nur noch 5.5 Milliarden Lösungen. [FJ06]

4.2 Komplexität

Bei Sudokus, die in den Medien veröffentlicht werden, und immer eindeutig lösbar sein müssen, kann durch logische Schlussfolgerung das Sudokurätsel gelöst werden.Durch die eindeutige Lösbarkeit ist es nicht notwendig Fallunterscheidungen zu treffen. Wenn ein Sudoku diese Eigenschaft nicht hast

Mathematisch gesehen ist ein Sudoku- Problem n-ter Ordnung. n ist eine natürlich Zahl. Auf einen $N \cdot N$ Gitter, wobei $N = n^2$ gilt, die Zahlen 1 bis N in das Gitter einzufügen, sodass in jeder Zeile, Spalte und in jedem Block der Größe $n \cdot n$ jede der Zahlen 1 bis N genau einmal vorkommen. N^2 Felder des Gitters können dabei vorbelegt sein. Ein $9 \cdot 9$ Sudoku ist damit ein Problem dritter Ordnung.

4.3 Eindeutige Lösbarkeit

Für ein Sudokugitter ohne vorgegebene Ziffern gibt es 5.472.730.538 (5,5 Milliarden) richtige Lösungen. Auch wenn nur eine oder zwei Ziffern vorgegeben werden gibt es immer noch eine sehr große Anzahl an Lösungen für dieses Sudoku.

Sudokus die in irgendeiner Form veröffentlicht werden, sind normalerweise mit der Vorgabe einer eindeutigen Lösungen erstellt. Sobald ein Sudoku nur eine korrekte Vervollständigung hat, ist es eindeutig lösbar. Daraus lässt sich folgern, dass in eindeutigen Sudokus in jede freie Zelle nur eine einzige Ziffer eingetragen werden kann, ohne die Regeln zu brechen. Sobald mehr als eine Ziffer in dem Rätsel gesucht wird, kann es zu einer Mehrdeutigkeit kommen.

Unter den vorgegebenen Zahlen eines Sudoku Rätsels müssen daher immer mindestens acht unterschiedliche Zahlen von 1-9 vorkommen. Dieses Kriterium ist gegeben durch den Fakt, dass bei nur sieben vorgegebenen unterschiedlichen Ziffern die beiden übrigen in der zugehörigen Lösung vertauscht werden können. [HM07]

4.3.1 Anzahl vorgegebene Ziffern für ein eindeutiges Sudoku

Es gibt die Vermutung, dass die minimale Anzahl an vorgegebenen Ziffern 17 ist. Mittels der Brute-Force Methode wurden eindeutige Sudokus mit nur 17 vorgegebenen Zahlen gesucht. Daran wurde 2011 von einem Forschungsteam um Gary McGuire geforscht. Es gibt jedoch noch keinen mathematischen Beweis für die Vermutung. Diese Vermutung basiert also hauptsächlich auf dem Generieren und Ausprobieren von vielen unterschiedlichen Sudokurätseln mit nur 17 Ziffern und einer eindeutigen Lösung. [Zei12]

4.4 Algorithmische Lösungsmethode: Backtracking

Backtracking ist eine Problemlösungsstrategie und mit der Rekursion verwandt. Es werden alle möglichen Lösungen ausprobiert und in jedem Schritt nach einer Abbruchbedingung überprüft. Die Bedingungen an das Problem, dass mit Backtracking gelöst werden sollen sind die Folgenden:

1. Das Problem ist in endlich vielen Teilschritten lösbar
2. Jeder der Teilschritte besitzt Abbruchbedingungen

3. Jeder der Schritte hat eine endliche Anzahl Lösungsmöglichkeiten

Für jedes Element des Problems, in diesem Fall für jedes freie Kästchen, werden alle möglichen Zustände ausprobiert. Die Zustände sind im Fall des Sudokus die Zahlen eins bis neun. Wenn ein Zustand zulässig ist, also in der Zeile, Reihe oder Kästchen die Zahl nicht bereits eingetragen ist, so wird rekursiv überprüft, ob es für den aktuellen Zustand eine Lösung gibt. Wenn es diese nicht gibt, wird der vorherige Schritt Rückgängig gemacht und eine neue Lösung gesucht. Damit basiert Backtracking auf dem Trial and Error Prinzip und versucht eine erreichte Teillösung in eine Gesamtlösung zu transferieren.

Bei z möglichen Verzweigungen jeder Teillösung und einem möglichen Verweigungsbaum mit der Tiefe von N hat das Backtracking sofern $z > 1$ ist im schlechtesten Fall mit $O(z^N)$ eine exponentielle Laufzeit.

$$1 + z + z^2 + z^3 + ... + z^N \tag{4.4}$$

Wenn mittels dem Backtracking keine Lösung für das Sudoku gefunden wurde, gibt es keine Lösung. Wenn jedoch eine Lösung gefunden wurde so ist noch nicht bewiesen, dass diese Lösung eindeutig ist.

Da es für das Lösen eines Sudokus keinen effizienten Algorithmus gibt und bereits vor der Anwendung der Lösungsstrategie überprüft werden soll, ob es eine eindeutige Lösbarkeit gibt, ist dieser Ansatz am Sinnvollsten. [Log14, S. 209 ff.]

4.4.1 Backtracking mit Brute-Force-Methode

Backtracking mit der Brute Force Methode funktioniert wie oben beschrieben auf dem Prinzip des Backtraking mit dem Ausprobieren aller möglichen Fällen. Mit dem ersten freien Feld probiert man mit der Eins beginnend, ob man zu einer Lösung kommt. Als Abbruchbedingung ist das singuläre Auftreten der neu eingefügten Zahl in Reihe, Zeile und Kasten implementiert.

Bei der Brute-Force Methode wird in jeder freien Zelle von der eins an beginnend die Zahlen nacheinander eingetragen und überprüft. Das Problem dabei ist, dass sobald in einer freien Zelle eine eins getragen wird und die Abbruchbedingung nicht ausgelöst wird, in der nächsten freien Zelle wieder von der eins an beginnend die Zahlen eingesetzt wird.

Die Laufzeit des Algorithmus hängt von der Anzahl der freien Zellen ab und damit auch von der Anzahl der vorgegeben Zahlen. Wenn viele Zahlen vorgegeben sind dann ist die Tiefe N des Verzweigungsbaum und damit auch die Laufzeit geringer.

```
def solve(self, numbers=(1, SIZE + 1), board=None):
  """
  Solves the SudokuBoard recursively via backtracking
  :return: False if not solvable, True if solved
  """
  square = self.get_empty_square(board)

  if not square:
    return True

  for i in range(*numbers):
    if self.is_valid(i, square, board=board):
      board[square[0]][square[1]][0] = i

      if self.solve(numbers=numbers, board=board):
      return True

      board[square[0]][square[1]][0] = 0

  return False
```

Abbruchbedingung

Die Abbruchbedingung für das Backtracking ist mit einer eigenen Funktion *is_valid* umgesetzt.

```
def is_valid(self, number, position, board=None):
  ...
  # Check row
  for j in range(SIZE):
    if board[position[0]][j][0] == number and j != position[1]:
    return False
  ...
  # Check Box
  box_x = (position[1] // 3) * 3
  box_y = (position[0] // 3) * 3

  for i in range(box_y, box_y + 3):
```

```
        for j in range(box_x, box_x + 3):
            if board[i][j][0] == number and (i, j) != position:
                return False

    return True
```

Wie in dem Code implementiert, wird in dem Board, nachdem eine Zahl in eine freie Zelle gesetzt wurde, überprüft ob diese Zahl bereits in der Reihe, Zeile oder Block der nun besetzten Position auf dem Board vorhanden ist. Wenn die Zahl gültig ist gibt die Funktion True zurück und in einem anderen Fall False. [Kno17]

4.4.2 Beweis Eindeutigkeit

Der Algorithmus des Backtracking bricht nach dem Finden einer vollständigen Lösung für ein Sudoku ab. Damit ist Bestätigt, dass eine Lösung für das Sudoku Rätsel gibt, jedoch nicht, dass diese Lösung eindeutig ist. Um die Eindeutigkeit eines Sudokus zu beweisen müsste nach dem Finden einer Lösung das Backtracking bis zum Ende weiter durchlaufen ohne ein weitere Lösung zu finden.

Um das ganze effizienter zu gestalten und die Zeit der Überprüfung des Sudokus zu verringer ist der Beweis der Eindeutigkeit, zwar auch durch Backtracking realisiert, aber nicht wie gerade beschrieben.

Um die Eindeutigkeit eines Sudoku Rätsels zu beweisen wird nach dem Finden einer Lösung das Backtracking nochmals von hinten begonnen. Damit ist gemeint, dass bei der ersten Lösungssuche die in den freien Zellen die Zahlen von eins bis neun eingetragen werden und nach dem Abbruchkriterium überprüft. Beim Suchen nach einer zweiten Lösungen beziehungsweise bei dem Beweis der Eindeutigkeit wird das Backtracking nochmals ausgeführt, und trägt die Zahlen von neun bis eins in die freien Zellen eingetragen.

```
def is_uniquely_solvable(self):
    temp_board = deepcopy(self.board)
    self.solve(numbers=(9, 0, -1), board=temp_board)
    return self.solved == temp_board
```

Nach dem Ausführen der beiden Backtracking Algorithmen werden die beiden gefundenen Lösungen miteinander verglich. Wenn das Sudoku eindeutig lösbar ist, dann stimmen beide gefundenen Lösungen miteinander überein. Anderen falls ist das Sudoku nicht eindeutig lösbar.

5 Frontend

6 Backend

6.1 Serverseite

6.2 Board

6.3 Implementierung Lösungsstrategien

7 Testing/Validierung

Literatur

[Ern20] P. Ernesti J. und Kaiser. *Python 3, Das umfassende Handbuch*. 5., aktualisierte Auflage. Bonn: Rheinwerk Computing, 2020. ISBN: 978-3-8362-7926-0.

[FJ06] Bertram Felgenhauer und Frazer Jarvis. „Mathematics of Sudoku I“. In: *Mathematical Spectrum* 39 (Jan. 2006).

[HM07] Agnes Herzberg und Ram Murty. „Sudoku squares and chromatic polynomials“. In: *Internationale Mathematische Nachrichten* 54 (Juni 2007).

[Kno17] Simon Knott. *Backtracking*. 2017. URL: https://simonknott.de/articles/backtracking.

[Log14] D. Logofătu. *Grundlegende Algorithmen mit Java: Lern- und Arbeitsbuch für Informatiker und Mathematiker*. Springer Fachmedien Wiesbaden, 2014. ISBN: 9783834819727. URL: https://books.google.de/books?id=GBcyngEACAAJ.

[Mar] Simon Martin. *Sudoku*. URL: https://www.thinkgym.de/r\%C3\%A4tselarten/s^^doku/.

[Sar20] Dejan Sarka. *A Python Data Analyst's Toolkit, Learn Python and Python-based Libraries with Applications in Data Analysis and Statistics*. 6., aktualisierte Auflage. Berkeley, CA: Apress, 2020. ISBN: 978-1-4842-7172-8.

[Sud22] Sudopedia. 2022. URL: http://www.sudopedia.org/.

[W3t] *Usage statistics of web servers*. URL: https://w3techs.com/technologies/overview/web_server.

[Zei12] Frankfurter Allgemeine Zeitung. *Mathematik: Der heilige gral der sudokus*. 2012. URL: https://www.faz.net/aktuell/wissen/physik-mehr/mathematik-der-heilige-gral-der-sudokus-11682905.html.

Anhang