



Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

PROJEKTBERICHT PARALLELRECHNEREVALUATION

MatchFinder

Wintersemester 2020/2021

vorgelegt von

Felix Wolf

11. April 2021

Fachbereich Informatik

Studiengang: Bachelor Software-System-Entwicklung

Matrikelnummer: 7199604

Inhaltsverzeichnis

1	Einleitung	1
1.1	Zielsetzung	1
1.2	Ergebnis	1
1.3	Verwandte Arbeiten	2
1.4	Reflektion	3
2	Ungarische Methode	4
2.1	Vorgehensweise	4
2.2	Beispiel	5
3	Dokumentation - Fachliches	7
3.1	Startseite	7
3.2	Verteilung auswerten	7
3.2.1	Verteilung aus einer Datei auswerten	7
3.2.2	Verteilung aus Datenbankdaten auswerten	8
3.3	Resultate einer Auswertung	9
3.4	Verteilung erstellen*	9
3.5	Verteilung teilen*	10
3.6	Daten anlegen*	10
3.6.1	Erstellung mittels Formular	11
3.6.2	Erstellung mittels Datei	12
3.7	Angelegte Daten*	12
3.8	Vorschau	13
3.9	Präferenzvergabe	14
4	Dokumentation - Technisches	15
4.1	Struktur	15
4.2	Layout	16
4.3	Python-Dateien	16
4.3.1	__init__.py	16
4.3.2	database_helper.py	16
4.3.3	config.py	17
4.3.4	matchCalculator.py	17
4.3.5	password_helper.py	18
4.4	Endpunkte	18
4.5	HTML-Templates	18
4.6	Technologien	19
4.6.1	Flask	19
4.6.2	Jinja2	19
4.6.3	Vue.js	19

4.6.4	SQLAlchemy	20
4.6.5	WTForms	20
5	Dokumentation - Installationsanleitung	22
	Quellenverzeichnis	24

1 Einleitung

Dieser Projektbericht bildet zusammen mit der Dokumentation auf GitHub [1] den theoretischen Teil des Moduls *Projekt: Parallelrechnerevaluation*, belegt im Wintersemester 2020/2021 am Arbeitsbereich Wissenschaftliches Rechnen an der Universität Hamburg. Die Dokumentation des Projekts befindet sich zusätzlich zu GitHub auch in diesem Bericht in den Kapiteln 3 - 5. Der praktische Teil ist das System *MatchFinder*. Die Web-Applikation ist auf den Servern des Arbeitsbereichs gehostet [2], der Quellcode kann auf GitHub eingesehen werden [3].

1.1 Zielsetzung

Ziel des Projekts war die Entwicklung einer Software, die Betreuern von Universitätsveranstaltungen und weiteren Mitarbeitern dabei hilft, Teilnehmer einer Veranstaltung zu gegebenen Gruppen bzw. Themen zuzuordnen. Da die Gruppen, zu denen Teilnehmer zugeordnet werden, auch als Themen verstanden werden können, sind diese Begriffe miteinander austauschbar. Im Folgenden wird der Begriff Gruppen stellvertretend für sowohl Gruppen als auch Themen verwendet.

Eine Zuordnung sollte nicht zufällig geschehen, sondern die Präferenzen der Teilnehmer berücksichtigen. Die Teilnehmer sollten die Möglichkeit haben, die zur Verfügung stehenden Gruppen in eine Reihenfolge zu bringen und ihnen somit Präferenzen zuzuordnen. Dies sollte dazu führen, dass möglichst viele Teilnehmer in die von ihnen bevorzugte Gruppe sortiert werden.

Die Daten mit Teilnehmern und Gruppen sollten mittels Dateiupload dem System zur Verfügung gestellt, Ergebnisse einer Verteilung persistent gespeichert werden. Nachdem ein Teilnehmer seine Präferenzen vergeben hat, soll er unter bestimmten Voraussetzungen diese Präferenzen anpassen können. Durch eine Veto-Funktion sollten einzelne Themen ausgeschlossen werden können, die Teilnahme an der Präferenzvergabe sollte per Link oder QR-Code geschehen.

Um möglichst vielen Personen diesen Dienst zur Verfügung zu stellen, bot sich die Umsetzung als Webapplikation an, welche öffentlich auf der Homepage des Arbeitsbereichs Wissenschaftlichen Rechnens [4] verlinkt wird. Da der Arbeitsbereich seine Webseiten selber hostet, besteht ein gewisser Technologiestack aus u.a. Apache2 und einer PostgreSQL Datenbank, in den sich die Webapp möglichst leicht einbinden lassen sollte. Ein weiterer, wichtiger Aspekt war die Sicherheit. So sollte überprüft werden, inwiefern die Webapplikationen vor Brute-Force-Angriffen oder Betrugsversuchen geschützt werden kann.

Eine ausführliche Spezifikation und damit eine Auflistung aller durch die Betreuung gewünschten Funktionen, befindet sich auf GitHub [5]. Nicht alle dort notierten Funktionen konnten umgesetzt werden. Bei vielen handelte es sich auch um zusätzliche Funktionalität, welche als Bonus und nicht als Pflichtinhalt zu verstehen ist.

1.2 Ergebnis

Das Ergebnis der Projektarbeit ist die Webapplikation *MatchFinder*, die nach den in Abschnitt 1.1 der Zielsetzung beschriebenen Anforderungen modelliert wurde. Die Software ist in Python

geschrieben und benutzt im Kern die Technologie Flask [6], mit der sich einfach und effizient Webapplikationen entwickeln lassen. Kapitel 3 behandelt die Handhabung und den Funktionsumfang des Systems, Kapitel 4 die technischen Details der Umsetzung.

MatchFinder realisiert das Berechnen von fairen Verteilungen mithilfe der Ungarischen Methode [7]. Das ist ein Algorithmus, welcher Minimierungsprobleme in Matrizenform löst. Folglich werden die Präferenzen der Teilnehmer in eine sogenannte Kostenmatrix transformiert. Bevor die Zuordnung stattfindet, hat jeder Teilnehmer die Gelegenheit, die verfügbaren Gruppen mit Präferenzen zu versehen. Das System interpretiert diese Präferenzen als „Kosten“ einer Zuordnung von einem Teilnehmer und einer Gruppe. Eine $n \times m$ -Matrix beinhaltet alle Präferenzen der n Teilnehmer zu den m Gruppen.

Es wird nun nach der Lösung mit minimalen Kosten gesucht. Da dies im Zweifel auch mehr als eine Lösung sein kann, gibt das System alle Verteilungen mit minimalen Kosten nacheinander aus. Der Munkres Algorithmus, wie ihn das System verwendet, liefert für eine Eingabematrix nur eine mögliche Ausgabe, auch wenn es weitere Ergebnisse mit den gleichen minimalen Gesamtkosten gibt. Um dies zu umgehen, wird die Eingabematrix einmal vollständig rotiert (die oberste Zeile wird zur untersten, alle anderen Zeilen rutschen nach oben, $n - 1$ mal). Jedes Zwischenergebnis dieser Rotation wird als Eingabematrix verwendet. Im Anschluss werden doppelte Ergebnisse gelöscht, sodass schlussendlich $0 \leq k \leq n$ Lösungen existieren, die alle die gleichen Gesamtkosten haben, diese jedoch unterschiedlich verteilt sind. Um eine maximale Anzahl der Ergebnisse zu garantieren, müssten alle Permutationen der Matrix als Eingabe verwendet werden. Dies ist jedoch mit erhöhtem Rechenaufwand verbunden, welcher nicht durch die zusätzlichen Ergebnisse im Vergleich zur Rotation gerechtfertigt wird.

Das System unterscheidet grundsätzlich zwei Arten von Benutzern: Jene, die authentifiziert sind und alle anderen. Die Authentifizierung findet dabei durch eine Passwortabfrage statt, ein vollständiges Benutzer-Passwort-System übersteigt die Anforderungen. Authentifizierten Benutzern stehen mehr Unterseiten mit weiteren Informationen und Funktionen zur Verfügung, wie zum Beispiel das Einsehen, Anlegen und Löschen von Daten.

An Verteilungen kann per Link und QR-Code teilgenommen werden, die Erstellung der Verteilungen beinhaltet das Konfigurieren einer Reihe von Parametern, die die verschiedenen Szenarien und Anforderungen umsetzen. So kann beispielsweise angegeben werden, ob Teilnehmer sich gegenüber dem System mit ihrer Matrikelnummer authentifizieren müssen. Dies wird anschließend dazu benutzt, sichergestellt zu werden, dass nur Veranstaltungsteilnehmer an der Verteilung teilnehmen.

Das System bietet an mehreren Stellen die Funktion an, Daten als Dateiupload zu übermitteln. Ergebnisse von Verteilungsauswertungen werden tabellarisch dargestellt und können in verschiedenen Formaten heruntergeladen werden.

Auch der Sicherheitsaspekt wurde beachtet: Besonders sensible Seiteninhalte werden vor wiederholten Anfragen geschützt, indem die Anfragen zeitlich limitiert sind. Ein Brute-Force-Angriff ist somit theoretisch noch möglich, dauert aber erheblich länger im Vergleich zu einer ungeschützten Webseite. Des Weiteren wird beim ersten Betreten der Seite die IP-Adresse des Benutzers mit einer Blacklist verglichen. Ist eine Übereinstimmung gefunden, wird der Seitenzugriff verwehrt.

1.3 Verwandte Arbeiten

Kurz vor der Implementierung der Präferenzvergabe kam durch die Projektbetreuung die Frage auf, ob bereits fertige Lösungen für eine Präferenzvergabe in das Projekt integriert werden können, um hier Implementationsaufwand und Zeit zu sparen. Tatsächlich gibt es bereits Webapplikationen, die Teilnehmern die Möglichkeit geben, z.B. gegenüber Terminen die Verfügbarkeit

zu signalisieren (Doodle [8]) oder Präferenzen abzugeben (Bitpoll [9]).

Jedoch bietet keiner dieser Dienste eine offene, kostenfreie API an, also eine Schnittstelle für dritte Programme, um mit den Services zu interagieren. Mit einer API könnte das MatchFindersystem automatisch die Events erstellen und einen entsprechenden Link für die Teilnehmer bereitstellen. Auf Anfrage teilte mir die Server-AG des Fachbereichs Informatik der Universität Hamburg mit, dass zu diesem Zeitpunkt (Dezember 2020) keine öffentliche API für das Uni-interne Bitpoll existiert. Die Server-AG betreibt die Server, auf welchen Bitpoll und andere der auf Mafiasi [10] bereitgestellten Diensten gehostet sind. Nach Einschätzung der AG sollte der Entwicklungsaufwand für eine API überschaubar sein. Voraussetzung hierfür sei ein sicherer Umgang mit Django Frameworks [11]. Seit zwei Jahren existiert im GitHub-Repository von Bitpoll ein Issue, der nach einer API-Lösung fragt [12]. Es scheint also ein Interesse an einer API über diese Projekt hinaus zu geben.

Da ich vor diesem Projekt noch keine Berührungspunkte mit Django und dementsprechend auch nicht mit Django Frameworks hatte, habe ich mich gegen die Entwicklung einer Api entschieden. Der schlussendliche Entwicklungsaufwand war für mich sehr schlecht abschätzbar und die Vermutung lag nahe, dass der Zeitaufwand der Implementation einer eigenen Präferenzvergabe kleiner oder gleichgroß sein würde, die Präferenzvergabe dabei jedoch präziser auf den Anwendungsbereich zugeschnitten wäre.

1.4 Reflektion

Rückblickend würde ich das Projekt als erfolgreich einschätzen. Alle Basisanforderungen sowie einige Zusatzfunktionen konnten umgesetzt werden, an einigen Stellen bietet das System Komfortfunktionen, die über eine reine Umsetzung der Anforderungen hinausgehen. Als Beispiel sei hier einmal das Ausklappen der Tabellen auf der `Daten anlegen`-Seite genannt oder die Tatsache, dass die Seite IP-Adressen filtert und an einigen Stellen wiederholte Seitenzugriffe beschränkt.

Dennoch ist das Potential für Erweiterungen und Verbesserungen nicht erschöpft. Eine Funktion, von der das System profitieren würde, ist das Bearbeiten von Datensätzen. Zu diesem Zeitpunkt besteht nur die Möglichkeit, Daten anzulegen oder zu löschen. Bei einem Tippfehler muss ein Datensatz gelöscht und neu hinzugefügt werden, wodurch alle mit diesem Datensatz in Verbindung stehenden Verteilungen auch gelöscht werden. Will ein Teilnehmer seine Präferenzen bearbeiten, werden ihm diese nicht erneut präsentiert. Der Benutzer kriegt bei der initialen Präferenzvergabe und bei der Bearbeitung die gleiche leere Oberfläche.

Ebenso stellt die Begrenzung auf maximal zehn Präferenzen eine Hürde für größer skalierte Anwendungen. Die Begrenzung ist der Implementation der Verteilungslogik geschuldet, jeder Wert muss *hart* im System stehen. Hier Bedarf es einer Erweiterung auf mehr Werte oder gleich eines ganzen Refactorings. Für den Anwendungsbereich der Universität sollten zehn Prioritätsstufen jedoch ausreichen, die Plattform STiNE bietet ebenfalls nicht mehr an.

Während der Projektlaufzeit konnte ich früh schnelle Fortschritte erzielen und war über das gesamte Semester hinweg vor dem Zeitplan. Gegen Ende hat die Betreuung die Entwicklung maßgeblich unterstützt, indem sie das System mehrfach auf mögliche Schwachstellen und Sonderfälle. Die so gefundenen Fehler konnten anschließend behoben werden.

Zusammenfassend bleibt die Umsetzung der Aufgabenstellung positiv in Erinnerung, ich konnte mir zuvor unbekannte Konzepte und Technologien kennenlernen und einsetzen. Das Projekt ist damit eine Bereicherung für mein Studium und ein Arbeitsergebnis, auf das ich stolz bin. Ich möchte mich herzlich bei meinen Betreuern Anna Fuchs und Janneck Squar bedanken, die mich durch den direkten Kontakt, der Unterstützung bei Fragen und Problemen und der Freiheiten, die mir bei der Umsetzung gegeben wurden, unterstützt haben.

2 Ungarische Methode

MatchFinder verwendet die ungarische Methode für die Auswertung der Kostenmatrizen. Auch wenn es sich bei dem Verfahren um einen erprobten Algorithmus handelt, sollen die Funktionsweise und seine Eigenschaften hier nochmal erläutert werden.

Die Ungarische Methode (*Hungarian Method*, auch Kuhn-Munkres-Algorithmus genannt), ist ein Algorithmus für das Lösen von gewichteten Zuordnungsproblemen auf bipartiten Graphen [7]. Der Algorithmus wurde 1955 von Harold Kuhn entwickelt und in *The Hungarian method for the assignment problem* veröffentlicht [13]. Die zu lösenden Probleme gehören zur Klasse der Linearen Optimierung, der Algorithmus besitzt unter Verwendung geeigneter Datentypen eine Komplexität von $O(n^3)$. Die Vorgehensweise des Algorithmus ist nicht in jedem Lehrbuch einheitlich und seit der Veröffentlichung 1955 hat sich der Algorithmus durch die Werke diverser Wissenschaftler (Edmunds & Karp, Ford & Fulkerson [14]) gewandelt. Im Folgenden beziehe ich mich daher auf die Aufarbeitung der Bevilacqua Research Corporation mit dem Titel *Munkres' Assignment Algorithm* [15]. Diese Version des Algorithmus bietet sich an, weil die Python-Library [16], die von MatchFinder benutzt wird, auf dieser Version aufbaut [17].

2.1 Vorgehensweise

Der Algorithmus besteht aus sechs Schritten, die teilweise wiederholt auftreten. Der Ausgangspunkt ist eine rechteckige $n \times m$ Matrix, mit den Einträgen $w_{i,j}$ mit $1 \leq i \leq n$ und $1 \leq j \leq m$. Der Eintrag $w_{1,2}$ enthält die Kosten von Objekt1 für Objekt2. Vor Beginn des Algorithmus wird die Matrix ggf. so gedreht, dass es mindestens ebenso viele Spalten wie Zeilen gibt (bei einer nicht-quadratischen Matrix gibt es folglich immer mehr Spalten als Zeilen). Darüber hinaus sei $k = \min(n, m)$.

1. Für jede Zeile der Matrix, finde das kleinste Element und subtrahiere es von jedem Eintrag in seiner Zeile.
2. Finde eine 0 in der neuen Matrix. Wenn es keine mit einem Stern versehene Null in der Zeile und Spalte dieser Null gibt, versehe sie mit einem Stern. Wiederhole für jede Null der Matrix.
3. Markiere jede Spalte in der sich eine Stern-Null befindet. Wenn k Spalten markiert wurden, sind die bis hierhin gefundenen Stern-Nullen die Lösung des Problems, gehe zu Schritt Ende. Wenn nicht, muss Schritt 4 ausgeführt werden.
4. Finde eine Null, die sich nicht in einer markierten Spalte befindet und markiere die Null. Wenn es in der Zeile dieser Null keine Stern-Null gibt, gehe zu Schritt 5. Sonst, markiere diese Zeile und entferne die Markierung der Spalte mit der Stern-Null. Wiederhole bis es keine Nullen in markierten Spalten mehr gibt. Speichere den kleinsten Wert, der sich nicht in einer markierten Spalte oder Zeilen befindet und gehe zu Schritt 6.
5. Erstelle eine abwechselnde Serie von markierten und Stern-Nullen wie folgt: Sei z_0 die markierte Null aus Schritt 4 und z_1 die Stern-Null in der Spalte von z_0 (wenn es eine gibt). Sei z_2 die markierte Null in der Zeile von z_1 (gibt es immer). Wiederhole bis die Serie mit einer markierten Null endet, in deren Spalte es keine Stern-Null gibt. Entferne alle Sterne der Serie und

mache aus markierten Nullen Stern-Nullen (entferne dabei die Markierungen) und entferne alle Markierungen von Spalten und Zeilen der Matrix. Gehe zurück zu Schritt 3.

6. Addiere den Wert aus Schritt 4 zu jedem Element aus jeder markierten Zeile (das keine markierte oder Stern-Null ist) und subtrahieren ihn von jedem Element jeder nicht-markierten Spalte. Gehe zurück zu Schritt 4.

Ende Wenn in Schritt 3 k Spalten markiert wurden, kann aus dieser Matrix die Lösung der Ausgangsmatrix abgeleitet werden. Die Kosten, die addiert die minimalen Gesamtkosten ergeben, befinden sich in der Ausgangsmatrix an den Stellen, in denen sich in Schritt 3 die Stern-Nullen befinden. Dies ist im Beispiel in Abbildung 2.1 mit den letzten beiden Matrizen dargestellt.

2.2 Beispiel

In Abbildung 2.1 wird beispielhaft die Ungarische Methode angewendet, um in einer Matrix eine minimale Zuordnung zu finden. Die Jobs p, q, r sollen den Arbeitern a, b, c zugeteilt werden. Dabei führt Arbeiter i den Job j unter den Kosten von $w_{i,j}$ aus. Daraus ergibt sich die Kostenmatrix 2.1.

$$C(i, j) = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix} \quad (2.1)$$

Aufgabe: Finde eine Zuordnung von Arbeiten zu Jobs, bei der jeder Arbeiter genau einen Job ausführt und es keine andere Zuordnung gibt, die geringere Gesamtkosten hat.

Aufgabe: Arbeiter = { a, b, c } sollen Jobs = { p, q, r } zugeteilt werden

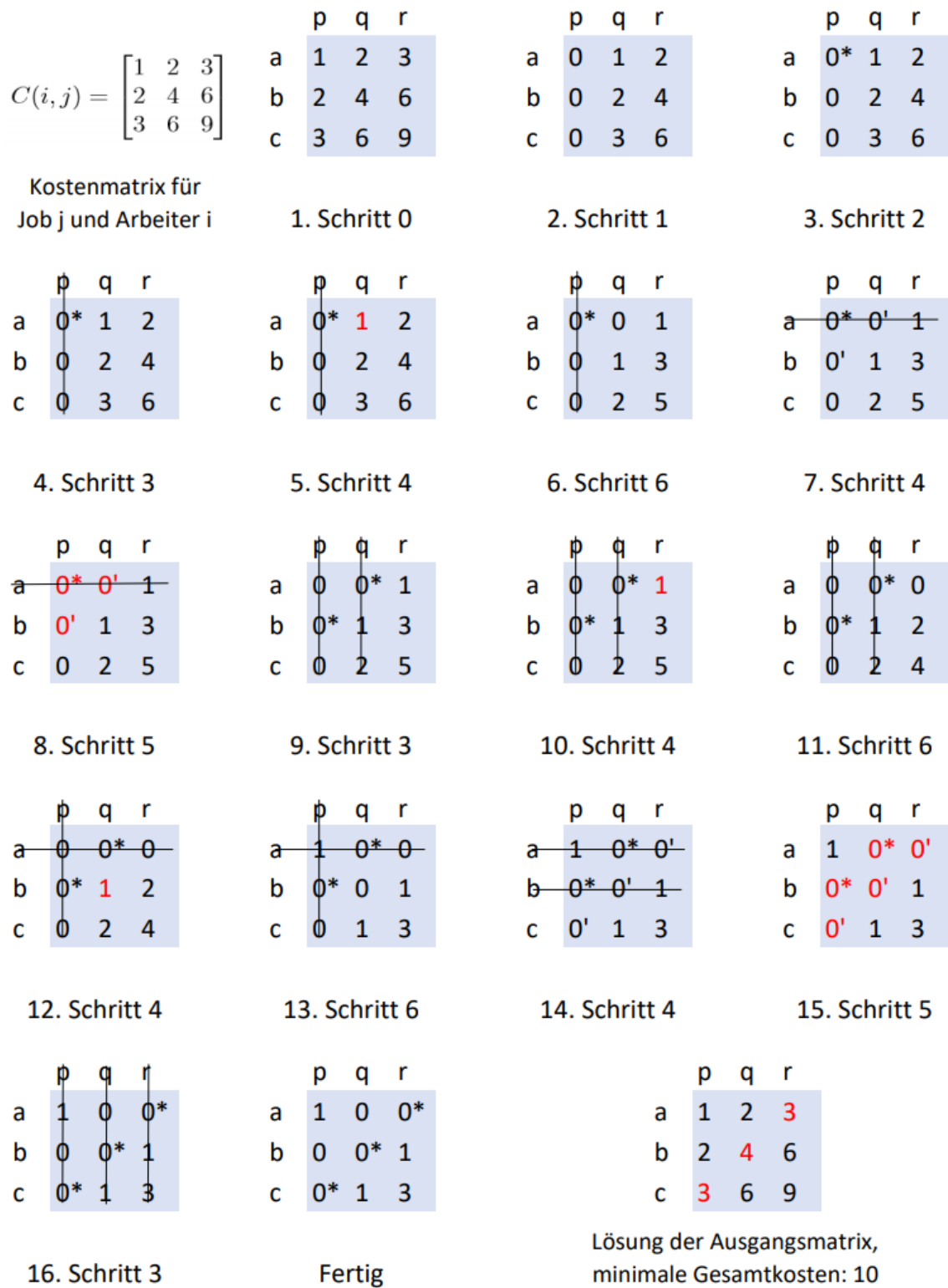


Abbildung 2.1: Anwendung der ungarischen Methode an einem Beispiel. Eigene Darstellung, orientiert an [15].

3 Dokumentation - Fachliches

Seitenunterpunkte, die nur authentifizierten Benutzern zur Verfügung stehen, sind mit einem * gekennzeichnet.

3.1 Startseite

Auf der Indexseite (/) findet der Nutzer allgemeine Informationen über den Dienst und die angebotenen Funktionen. Sie ist sowohl über den Menüpunkt Home als auch über den Schriftzug MatchFinder in der oberen Leiste zu erreichen.

3.2 Verteilung auswerten

Über den Menüpunkt Verteilung auswerten (/evaluate) wird die Funktion angeboten, Verteilungen (aus Dateien oder aus Datenbankdaten) auszuwerten. Die volle Funktionalität steht nur einem authentifizierten Benutzer zu Verfügung. Nicht authentifizierte Benutzer haben nur Zugriff auf die Funktion, eine Verteilung aus einer Datei auszuwerten, welche im folgenden Abschnitt näher beschrieben ist.

3.2.1 Verteilung aus einer Datei auswerten

Eine Verteilung kann ausgewertet werden, wenn die Präferenzen als CSV-Datei vorliegen. Voraussetzung hierfür ist, dass die Datei folgendem Format folgt:

1. die durch die Datei beschriebene Matrix ist rechteckig (alle Zeilen haben gleiche Länge)
2. in der ersten Zeile sind die Namen der Gruppen / Themen, wobei Zelle 0,0 frei bleibt bzw. nur einen Platzhalter beinhaltet
3. in der ersten Spalte befinden sich die Namen der Teilnehmer. Sie dürfen Leerzeichen enthalten, aber keine Kommata
4. die Präferenzen sind als Bezeichnung angegeben, nicht als Zahl (Erstwahl statt 1, Zweitwahl statt 2 etc.)
5. die maximal auszugebende Präferenz ist die Zehntwahl, darüber hinaus werden keine weiteren Präferenzen angegeben

Beispielformat: Eine formatgerechte Datei sieht demnach so aus:

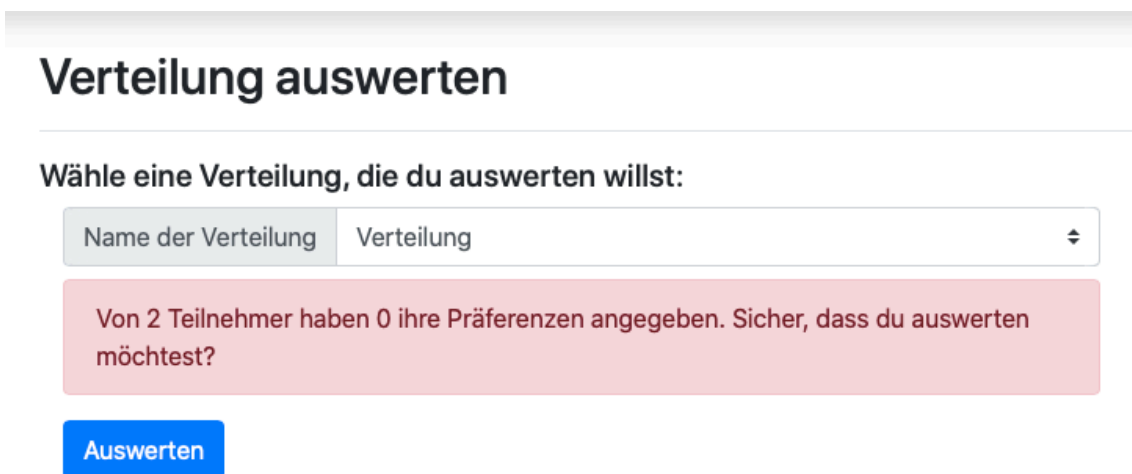
```
PLATZHALTER,Thema1,Thema2,Thema3,Thema4
Teilnehmer1,Erstwahl,Zweitwahl,Drittwahl,Viertwahl
Teilnehmer2,Viertwahl,Drittwahl,Zweitwahl,Erstwahl
Teilnehmer3,Erstwahl,Zweitwahl,,
Teilnehmer4,,Zweitwahl,Drittwahl,Erstwahl
```

Für eine optimale Auswertung sollten so viele Angaben wie möglich gemacht werden. Soll eine Verteilung ausgewertet werden, kann dies mit dem Auswählen des `Auswerten`-Knopfes veranlasst werden. Der Benutzer wird dann auf eine neue Seite weitergeleitet. Diese ist im Abschnitt 3.3 näher beschrieben.

3.2.2 Verteilung aus Datenbankdaten auswerten

Sind noch keine Daten in der Datenbank angelegt, steht dem authentifizierten Benutzer die gleiche Funktionalität wie in Abschnitt 3.2.1 beschrieben zur Verfügung. Er erhält außerdem noch einen Verweis auf die `Verteilung anlegen`-Funktion, über welche er eine Verteilung anlegen kann.

Existiert jedoch eine Verteilung in der Datenbank, kann diese ausgewählt werden, um sie auszuwerten. Unter dem Auswahlelement wird u. U. darauf hingewiesen, dass eine Auswertung noch nicht ausgewertet werden sollte. Beispiel:



The screenshot shows a web interface titled "Verteilung auswerten". Below the title, there is a prompt: "Wähle eine Verteilung, die du auswerten willst:". This is followed by a dropdown menu with "Name der Verteilung" and "Verteilung" as options. Below the dropdown, a red warning box contains the text: "Von 2 Teilnehmer haben 0 ihre Präferenzen angegeben. Sicher, dass du auswerten möchtest?". At the bottom of the interface is a blue button labeled "Auswerten".

Abbildung 3.1: Eine Warnung bei der Auswertung einer Verteilung

Eine Warnung kann aus verschiedenen Gründen angezeigt werden:

1. kein Teilnehmer hat an der Verteilung teilgenommen
2. nicht alle Teilnehmer haben an der Verteilung teilgenommen (wie in Abbildung 3.1, die Schwelle liegt bei 80%)
3. Es haben mehr Teilnehmer an der Verteilung teilgenommen, als es Plätze gibt (Anzahl Plätze = Anzahl Themen * erlaubte Teilnehmer pro Thema). Dies kann nur bei offenen Verteilungen vorkommen, bei denen die Anzahl der Teilnehmer im Voraus nicht bekannt ist.

Während 1.-2. lediglich Warnungen sind, führt 3. dazu, dass die Verteilung nicht ausgewertet werden kann. Die Verteilung muss erneut erstellt und Präferenzen erneut vergeben werden. Ist eine Verteilung auswertbar, kann dies mit dem `Auswerten`-Knopfes veranlasst werden. Der Benutzer wird auf eine neue Seite weitergeleitet. Diese ist in Abschnitt 3.3 näher beschrieben.

The screenshot shows the MatchFinder application interface. On the left is a sidebar with navigation links: Home, Verteilung auswerten, Verteilung erstellen, Daten anlegen, Angelegte Daten, Doku, and Github. The main area is titled 'Resultate' and contains the following text: 'Es gibt insgesamt 1 mögliche Verteilung.' and 'Die Varianten sind nach ihrem Median und der absoluten Abweichung vom Median sortiert.' Below this is a section 'Variante 1' with a table:

Studi	Thema / Gruppe	# Wahl
Marie Musterfrau	Thema5	1
Max Mustermann	Thema6	1

Below the table is an 'Export als...' button with a dropdown menu showing 'CSV Datei' and 'WikiDocs-Format'.

Abbildung 3.2: Die Auswertungsergebnisse einer Verteilung

3.3 Resultate einer Auswertung

In Abbildung 3.2 ist beispielhaft dargestellt, wie die Resultate einer Auswertung präsentiert werden. Je nach Datenbasis gibt es 1 bis n mögliche Verteilungen von Teilnehmern zu Themen, welche alle die gleichen optimalen (d.h. minimalen) Gesamtkosten haben. Gibt es mehr als eins solcher Resultate, werden diese untereinander angezeigt, wobei nach dem Median und der absoluten Abweichung des Medians sortiert wird. Das beste Resultat ist immer das Oberste. Hier ein Beispiel zur Erklärung:

Verteilung1 habe die Kosten $[2,2,2]$, Verteilung2 die Kosten $[1,2,3]$. Damit haben beide Verteilungen einen Median von 2. Jedoch ist die absolute Abweichung vom Median bei Verteilung1 0, während sie bei Verteilung2 2 ist. In diesem Anwendungsfall ist eine geringere Abweichung besser, also wird Verteilung1 gegenüber Verteilung2 bevorzugt und über ihr angezeigt. Pro Verteilungsvariante wird nun angezeigt, welcher Student welches Thema bekommen hat und die wievielte Wahl es jeweils war.

Export

Die Seite bietet dem Benutzer Exportfunktionen in zwei Formaten an:

1. **CSV-Datei:** Hier wird die Tabelle ins CSV-Format umgewandelt und ausgegeben, nützlich für Programme wie Microsoft Excel o.ä.
2. **WikiDocs-Format:** Speziell implementiert für den Arbeitsbereich Wissenschaftliches Rechnen an der UHH, die auf ihrer Webseite WikiDocs verwenden. Bei diesem Format handelt es sich um eine verändertes Markdown.

3.4 Verteilung erstellen*

Sind zuvor Teilnehmer und Gruppen angelegt worden, kann auf dieser Unterseite eine Verteilung erstellt werden. Wenn nicht, beinhaltet die Seite einen Verweis auf die Daten-anlegen-Funktion (siehe Abschnitt 3.6), über welche dann Teilnehmer und Gruppen / Themen erstellt werden können.

Verteilungen zu erstellen bedeutet, zu einer gegebenen Liste von Themen eine Liste an Teilnehmern (vorausgesetzt, die Verteilung ist geschützt) zuzuordnen und den Teilnehmern die Möglichkeit zu geben, ihre Präferenzen zu den Themen abzugeben. Eine Verteilung wird dabei beschrieben durch eine Reihe von Eigenschaften:

- **Name:** Jede Verteilung braucht einen Namen, um sie später identifizieren zu können, wenn die Verteilung ausgewertet werden soll
- **Gruppenliste:** Die Liste der Gruppen / Themen, zu denen Präferenzen angegeben werden sollen
- **# / Gruppe:** Diese Eigenschaft gibt an, wie viele Teilnehmer auf eine Gruppe kommen dürfen. Beispiel: Ist $\# / \text{Gruppe} = 1$, so darf nur ein Teilnehmer auf jede Gruppe verteilt werden, eine 1:1 Beziehung. Ist $\# / \text{Gruppe} = n$ mit $n > 1$, handelt es sich um eine 1:n Beziehung, wobei nur dann jede Gruppe voll ist, wenn es genau $\text{längeDerGruppenliste} * \# / \text{Gruppe}$ Teilnehmer gibt.
- **Mindeststimmen:** Gibt an, wie viele Präferenzen jeder Teilnehmer mindestens vergeben muss. Dabei ist der Wert nach unten bis 1 und nach oben bis 10 bzw. Anzahl der Themen beschränkt (je nach dem, was zuerst eintritt).
- **Veto erlaubt:** Wenn der Haken gesetzt ist, darf ein Teilnehmer eine der angebotenen Themen / Gruppen vollkommen ausschließen.
- **geschützt:** Ist diese Option ausgewählt, muss sich jeder Teilnehmer mit seiner Matrikelnummer gegenüber dem System authentifizieren, bevor er seine Präferenzen angeben darf. Ist die Option nicht ausgewählt fallen die beiden nächsten Optionen weg.
- **Teilnehmerliste:** Soll die Verteilung geschützt sein muss im Vorhinein feststehen, welche Teilnehmer teilnehmen werden. Über diese Option wird diese Teilnehmerliste ausgewählt.
- **editierbar:** Diese Option gibt an, ob Teilnehmer nach dem ersten Bestätigen ihrer Präferenzen ihre Angabe erneut anpassen dürfen. Diese Funktion gilt für alle Teilnehmer und unbegrenzt oft.

Mit dem Auswählen des Erstellen-Knopfes wird die Verteilung erstellt und der Benutzer wird auf eine Seite weitergeleitet, auf welcher die Verteilung geteilt werden kann.

3.5 Verteilung teilen*

Auf dieser Seite kann eine erstellte Verteilung geteilt werden. Zu diesem Zweck wird der Link angezeigt, welcher benötigt wird, um zu der Verteilung zu navigieren. Darunter ist ein QR-Code, welcher von den Teilnehmern gescannt werden kann, er führt ebenfalls zur Präferenzvergabe.

3.6 Daten anlegen*

Um eine Verteilung aus Datenbankdaten erstellen zu können, müssen diese im Voraus angelegt werden. Diese Funktionalität bietet der Unterpunkt Daten anlegen (/upload).

Die App arbeitet mit zwei Arten von Daten: Teilnehmer und Gruppen / Themen. Beide lassen sich über einen Datei-Upload oder per Formular erstellen.

3.6.1 Erstellung mittels Formular

Liegen die Daten nicht als Dateien vor, ist die Erstellung mittels Formular eine effiziente und angenehme Art, die Daten zu erstellen. Hierzu muss vor der eigentlichen Erstellung zunächst festgelegt werden, wie viele Teilnehmer oder Gruppen / Themen angelegt werden sollen (siehe Abbildung 3.3). Es muss mindestens ein Teilnehmer oder eine Gruppe angelegt werden. Nach oben gibt es keine Beschränkung.

Abbildung 3.3: Anzahl der zu erstellenden Daten festlegen

An Verteilungen können immer nur ganze Gruppen von Teilnehmern teilnehmen. Es gibt **nicht** die Möglichkeit, nach der Erstellung einer Gruppen- oder Teilnehmerliste die dazugehörigen Gruppen oder Teilnehmer zu bearbeiten, d.h. die Daten zu verändern oder Einträge zu löschen / hinzuzufügen. Diese Listen sind erst zu erstellen, wenn alle Einträge feststehen.

Bei der Erstellung von Teilnehmern oder Gruppen / Themen gibt es Informationen, die ausgefüllt werden müssen, während andere optional sind. Der Name der Teilnehmer- bzw. Gruppenliste ist immer erforderlich, die Notwendigkeit der weiteren Feldern ist den Tabellen der Abbildung 3.4 zu entnehmen.

Feld	muss angegeben werden
Vorname	Ja
Nachname	Nein
Matrikelnummer	Ja

(a) Felder des Teilnehmer-Datentyps

Feld	muss angegeben werden
Name	Ja
Uhrzeit	Nein
Betreuer	Nein

(b) Felder des Thema/Gruppe-Datentyps

Abbildung 3.4: Feldspezifikationen von Teilnehmer 3.4a und Thema / Gruppe 3.4b

Sind die benötigten Felder nicht ausgefüllt, lässt sich die Verteilung nicht erstellen.

Wenn die Erstellung erfolgreich war, wird der Benutzer zur vorherigen Seite zurückgeleitet und es erscheint ein kleines Banner, welches Auskunft über die Anzahl der erstellen Einträge gibt.

Erstellte Daten können unter *Angelegte Daten* (Abschnitt 3.7) betrachtet und bearbeitet werden.

3.6.2 Erstellung mittels Datei

Alternativ zum Formular können Daten auch per Datei erstellt werden. Hier sind besonders die Formatvorgaben zu beachten.

Für beide Datentypen gilt: die Dateien müssen auf .csv oder .txt enden.

Teilnehmer

Während die Themen / Gruppen im standardmäßigen CSV-Format angegeben werden, ist dies bei den Teilnehmern anders. Hier werden die Spalten mit Tabs getrennt. Dies liegt daran, dass es möglich sein soll, Teilnehmerlisten aus STiNE zu exportieren und in MatchFinder zu importieren. Also muss Matchfinder dem Format von STiNE folgen. Da STiNE die Listen je nach Kontext in zwei Darstellungen exportiert, unterstützt MatchFinder auch beide.

Hier beide Varianten:

Variante 1:

Id <TAB> Matrikelnummer <TAB> Nachname <TAB> Vorname <TAB>

Variante 2:

Id <TAB> Matrikelnummer <TAB> Nachname <TAB> Vorname <TAB> Zeitpunkt <TAB>

Beispieldateiinhalte Variante 1:

```
1 1234567 Mustermann Max Übungen Gr. 03 (Do. 10-12 Uhr)
2 7654321 Musterfrau Marie Übungen Gr. 05 (Do. 14-16 Uhr)
```

Der Inhalt der Zeitpunkt-Spalte aus Variante zwei wird verworfen.

Themen / Gruppen

Themen / Gruppen werden als CSV-Dateien hochgeladen, dabei ist das Format:

Name, Zeit, Betreuer

Beispiel:

```
Thema1,Heute,Frau Schmidt
Thema2,Morgen,Herr Schultz
Thema3,,Ich
Thema4,Nächste Woche,
Thema5,,
```

3.7 Angelegte Daten*

Die Seite Angelegte Daten (/edit) bietet die Funktion, die angelegten Daten zu betrachten und zu bearbeiten. Die Seite ist in vier Abschnitte aufgeteilt: Jeweils eine Tabelle für Teilnehmer, Gruppen / Themen und Verteilungen und eine Legende. Die Legende zeigt nur diejenigen Symbole an, die für die existierenden Daten relevant sind. Gibt es keine Daten, wird die Legende ausgeblendet.

MatchFinder Ausloggen

- Home
- ✓ Verteilung auswerten
- + Verteilung erstellen
- ↑ Daten anlegen
- ☰ Angelegte Daten
- 📄 Doku
- 🔗 Github

Angelegte Daten

Teilnehmer-Listen

#	Name	Aktion
1	TestTeilnehmer	Löschen

#	Matr.Nr	Nachname	Vorname
1	1		Max
2	2		Lisa

Gruppen / Themen-Listen

#	Name	Aktion
1	Termine	Löschen

#	Name	Betreuer	Zeit
1	Heute		
2	Morgen		

Verteilungen

#	Name	# Teilnehmer	# Themen	👤	Infos	Aktion	Löschen
1	Test-Verteilung	2	2	1 / 2	Infos	Aktion	Löschen

Abbildung 3.5: Unterseite 'Angelegte Daten' mit Beispielinhalt

Die Tabellen für Teilnehmer und Gruppen / Themen zeigen auf oberster Ebene die Listen von Teilnehmer bzw. Gruppen Themen und es wird eine Löschfunktion angeboten. Listen lassen sich nur als Ganzes löschen und die einzelnen Einheiten können nicht bearbeitet werden. Bevor ein Eintrag endgültig gelöscht wird, wird der Benutzer um Bestätigung gebeten, um ein versehentliches Löschen zu vermeiden. Durch einen Klick auf eine Liste wird die Tabelle ausgeklappt und der Listeninhalt wird angezeigt.

Die Tabelle der Verteilungen kann nicht ausgeklappt werden. Über den Infos-Knopf wird ein Tooltip angezeigt, in dem sich weitere Details und Parametereinstellungen (z.B. ob sie geschützt ist) befinden. Über den Aktion-Knopf kann die Verteilung ausgewertet, geteilt oder in der Vorschaufunktion (siehe Abschnitt 3.8) betrachtet werden. Die Teilen-Funktion leitet den Benutzer zur Teilen-Seite (Abschnitt 3.5) weiter, die Löschfunktion wird ebenfalls angeboten.

Werden Themen- oder Teilnehmerlisten gelöscht, die Teil von bestehenden Verteilungen sind, werden diese Verteilungen ebenfalls gelöscht.

3.8 Vorschau

Über die Vorschau-Funktion (`/preview`) können die Präferenzen der Teilnehmer einer Verteilung eingesehen werden. Dies ist besonders nützlich, um herauszufinden, welche Teilnehmer noch nicht ihre Präferenzen angegeben haben. Zusätzlich kann die Vorschau-Funktion dazu dienen, die zugrundeliegenden Daten von Verteilungsergebnissen einzusehen.

Abbildung 3.6 zeigt eine Beispielverteilung mit den Teilnehmer Max und Lisa und den Gruppen Heute und Morgen. Während Max seine Präferenzen schon verteilt hat, hat Lisa noch keine angegeben. Eine Verteilung kann in der Vorschau über den entsprechenden Knopf unter `Aktion` auf der `Angelegte Daten`-Seite betrachtet werden.

The screenshot shows the MatchFinder application interface. On the left is a sidebar with navigation items: Home, Verteilung auswerten, Verteilung erstellen, Daten anlegen, Angelegte Daten, Doku, and Github. The main content area is titled 'Vorschau' and contains the text 'Vorschau der Teilnehmer der Verteilung und die von ihnen abgegebenen Präferenzen.' Below this is a table with the following data:

Name des Teilnehmers	Heute	Morgen
Max (**1)	1	2
Lisa (**2)		

Abbildung 3.6: Vorschau-Ansicht einer Beispielverteilung

3.9 Präferenzvergabe

Bei der Präferenzvergabe priorisiert ein Teilnehmer die ihm/ihr angebotenen Gruppen oder Themen. Die Priorität wird von Erstwahl bis Zehntwahl angegeben, u. U. kann ein Thema mittels Veto ausgeschlossen werden.

Bei den Verteilungen wird zwischen offenen und geschützten Verteilungen unterschieden. Ist die Verteilung geschützt wird der Benutzer aufgefordert, sich mittels seiner Matrikelnummer zu authentifizieren, bevor die Präferenzen vergeben werden können. Ist die Verteilung offen, hat er stattdessen die Möglichkeit, seinen Vor- und Nachnamen einzutragen, wobei nur der Vorname Pflicht ist.

Wird bei der Präferenzvergabe eine der Antwortmöglichkeiten (z.B. Erstwahl) für eine der Themen angegeben, verschwindet diese Priorität aus den Antwortmöglichkeiten der anderen Themen. Themen, die mit „Keine Präferenz“ markiert sind, werden vom System mit passenden, zufälligen Gewichten versehen, um eine faire und sichere Verteilung zu gewährleisten.

4 Dokumentation - Technisches

4.1 Struktur

Die allgemeine Struktur des Repositories und folglich auch der Flask-App lässt sich wie folgt darstellen:

```
MatchFinder (root)
├── matchFinder
│   ├── Python-Dateien ...
│   ├── forms
│   │   └── ...
│   ├── models
│   │   └── ...
│   ├── static
│   │   └── ...
│   └── templates
│       └── ...
├── documentation
│   └── ...
├── README.md
├── setup.sh
├── install.sh
├── requirements.txt
├── list_of_blocked_ips.txt
└── ...
```

Die Dateien auf root-Ebene haben folgende Aufgaben:

- **install.sh** erstellt und aktiviert ein virtuelles Environment, bevor alle nötigen Dependencies, die von der App benötigt werden, installiert werden.
- **setup.sh** startet die Flask-App im Develop-Modus.
- **list_of_blocked_ips.txt** ist eine Liste von IP-Adressen, die in Vergangenheit verantwortlich für Spam und unerwünschte Aufrufe waren. Bei erstmaligem Aufrufen der Webseite eines Nutzers wird geprüft, ob sich die IP-Adresse des Benutzers auf dieser Liste befindet. Ist dies der Fall, wird der Zugriff verwehrt.
- **README.md** beinhaltet die Übersicht der Dokumentation.
- **requirements.txt** listet alle benötigten (transitiven) Abhängigkeiten des Systems auf. Die Datei wird mit Pip Freeze generiert und kann dazu genutzt werden, ähnlich wie install.sh alle Abhängigkeiten gebündelt zu installieren.

Der Unterordner `matchFinder` beinhaltet die Flask App.

- **Root Level:** Auf Root Level von `matchFinder` befinden sich alle Endpunkte der App. Mehr dazu im Abschnitt Python-Dateien 4.3.

- **Forms:** In `forms` sind die Formularvorlagen von `WTForms` für einige Formularabfragen abgelegt.
- **Models:** In `models` befinden sich die ORM-Klassen für `SQLAlchemy`, also die Vorlagen aller Datenbanktabellen.
- **Static:** Im `static` Ordner sind statische HTML-Ressourcen, wie das Favicon, das CSS für das Layout und die `Vuejs`-Library gespeichert.
- **Templates:** Der `templates`-Ordner beinhaltet alle HTML-Templates der App, die mit `Jinja` ausgebaut werden.
- **Documentation:** In `documentation` liegen die Markdown-Dateien der Dokumentation und die verwendeten Abbildungen.

4.2 Layout

Die Webapp bietet ein simples Layout mit einer Seitenleiste für die Navigation, einem Button zur Authentifikation in der Topbar und eine Überschrift auf jeder Seite.

Bei Darstellung auf kleinen Displays (z.B. auf mobilen Geräten) wechselt die App in ein mobile-freundliches Layout. Hier ist die Seitenleiste nicht dauerhaft sichtbar, sondern wird ausgeklappt. Für die visuelle Aufbereitung von HTML-Elementen über `CSS` etc. wird `Bootstrap` [18] verwendet.

4.3 Python-Dateien

Die Python-Dateien im `matchFinder`-Order beinhalten die gesamte Logik der App. Einzelne Dateien werden nun genauer beschrieben.

4.3.1 `__init__.py`

Hier wird die `Flask`-App initialisiert und die Blueprints (definierte Endpunkte in anderen Dateien) registriert. Auch die Datenbank wird hier initialisiert.

Hier ist auch die Logik der Blacklist definiert: Die App blockiert alle IP-Adressen, die auf der Blacklist stehen. Beim ersten Aufrufen der App wird die IP-Adresse des Aufrufers einmalig mit diesen gesperrten IPs verglichen. Der Status (Zutritt erlaubt oder verweigert) wird in einem Cookie festgehalten. So muss der Status innerhalb einer Session nicht zweimal geprüft werden.

Zusätzlich beinhaltet die Datei einen Endpunkt für den Index (`/`) und einen Endpunkt für alle Anfragen, die auf keinen gültigen Endpunkt verweisen. Index wird auf `/home` weitergeleitet, ungültige Anfragen geben einen 404-Fehler zurück.

4.3.2 `database_helper.py`

`Database_helper` ist die Schnittstelle zwischen App und Datenbank. Datenbankzugriffe geschehen nur in dieser Datei. Aus diesen Gründen bietet die Datei alle benötigten Datenbankoperationen für die andere Klassen und Blueprints an.

4.3.3 config.py

Dies ist eine Konfigurationsdatei, ohne die die App nicht startet. In ihr wird der Pfad zur Datenbank, der Secret-Key [19] und Eigenschaften der Dateien, die das System für den Upload erlaubt, definiert.

4.3.4 matchCalculator.py

Das Herz der Matchberechnung. Hier wird mithilfe des Munkres-Algorithmus [7, 16] ein faires Match berechnet. Die Daten zu einem Match kommen dabei entweder aus einer Datei oder aus bestehenden Datenbankdaten.

Das Datenmodell ist dabei eine rechteckige Matrix mit Aufbau

Name	Thema1	Thema2	Thema3
Teilnehmer1	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$
Teilnehmer2	$w_{2,1}$	$w_{2,2}$	$w_{2,3}$
Teilnehmer3	$w_{3,1}$	$w_{3,2}$	$w_{3,3}$

Abbildung 4.1: Datenmodell Matrix

Dabei ist $w_{x,y}$ das Gewicht (die *Präferenz*) von Teilnehmer x zu Thema y . Wird eine Verteilung als Datei hochgeladen, dann als *comma seperated values (csv)* Datei, in dem die Präferenzen als Text angegeben sind:

```
Gewicht 1 = Erstwahl  
Gewicht 2 = Zweitwahl  
...  
Gewicht 5 = Fünfwahl  
...
```

Das höchste unterstützte Gewicht ist zehn (*Zehntwahl*). Wenn eine Option ganz ausgeschlossen werden soll, kann diese mit *Veto* markiert werden. Optionen ohne Präferenz werden automatisch durch das System aufgefüllt.

Eine formatgerechte Datei sieht demnach so aus

```
Name,Thema1,Thema2,Thema3,Thema4  
Teilnehmer1,Erstwahl,Zweitwahl,Drittwahl,Viertwahl  
Teilnehmer2,Viertwahl,Drittwahl,Zweitwahl,Erstwahl  
Teilnehmer3,Erstwahl,Zweitwahl,,Viertwahl  
Teilnehmer4,Viertwahl,Zweitwahl,Drittwahl,Erstwahl
```

Begrenzung auf nur zehn Präferenzen

Das System unterstützt momentan bis zu 10 Präferenzen. Die bedeutet, dass die Felder nach der Zehntwahl entweder dynamisch hochgezählt (Bei Präferenzvergabe über das System) oder mit dem statischen Gewicht von 100 (bei Berechnung aus Datei) belegt werden.

Dieser Umstand ist eine bewusste Entscheidung. Unter anderem sprechen diese Gründe für eine Begrenzung:

1. In den seltensten Fällen möchte ein Teilnehmer mehr als 10 Präferenzen angeben. Die Wahrscheinlichkeit, nicht eine der ersten 10 Wahlen zu bekommen ist bei moderat großen Themen und Teilnehmern gering.

- Um den Teilnehmern die Vergabe einer Erstwahl anstatt einer 1 zu ermöglichen, müssen diese Antwortmöglichkeiten hardgecodet werden und können nicht (anders als Zahlen) dynamisch generiert werden. Hier stellt sich die Frage, ob sich eine Implementation von beispielsweise 100 Antwortmöglichkeiten überhaupt lohnt, wenn in den meisten Fällen nur die ersten 10 Vergeben werden.

4.3.5 password_helper.py

Die Datei `password_helper.py` ist zuständig für das Generieren und Überprüfen von Passwörtern. MatchFinder benutzt kein vollständiges User-Password-System, hierfür existiert schlichtweg keine Notwendigkeit. Vielmehr können sich User über zuvor definierte geheime Schlüssel gegenüber dem System authentifizieren. Einmal authentifiziert stehen dem User eine Vielzahl von Funktionen zur Verfügung, die vor dem Zugriff eines unauthorisierten Benutzers verborgen und blockiert sind.

Geschützte Seitenendpunkte sind visuell in der Seitenleiste nicht sichtbar und die einzelnen Endpunkte überprüfen bei jedem Zugriff den Status des Nutzers. Ist dieser nicht berechtigt wird der User auf die Hauptseite zurückgeleitet:

```
@bp.before_request
def check_status():
    if session.get('is_authenticated') != True:
        return redirect(url_for('home.index'))
```

Die auf diese Weise geschützten Endpunkte sind: `/share`, `/edit`, `/upload`, `/create`, `/preview`. Mithilfe der `Session` können Informationen über die Session des Nutzers gespeichert werden. In diesem Fall wird sie für die Authentifizierung mittels Passwort genutzt. In ihr wird auch gespeichert, ob der Seitenbesucher mit einer vertrauenswürdigen IP-Adresse die Seite ansteuert (siehe Abschnitt 4.3.1).

Diese Informationen werden in einem Cookie gespeichert. Dieser Cookie ist der einzige, der auf der Webseite zum Einsatz kommt. Es gibt **kein** Tracking oder ähnliches.

4.4 Endpunkte

Pro Endpunkt der Flask-App gibt es eine Datei, die einen sog. Blueprint definiert. Die Endpunkte der App sind `/auth`, `/create`, `/edit`, `/evaluate`, `/home`, `/preference`, `/preview`, `/share` und `/upload`.

Jeder dieser Blueprints definiert bestimmte Haupt- und Untereendpunkte, generiert Inhalte und veranlasst Redirects bzw. das Rendern von HTML-Dateien.

Einige dieser Endpunkte sind nur für authentifizierte Benutzer erreichbar. Geschützte Endpunkte sind: `/create`, `/edit`, `/share`, `/preview` und `/upload`.

4.5 HTML-Templates

Die HTML Templates befinden sich im Verzeichnis `MatchFinder/matchFinder/templates`. Die Datei `layout.html` ist das Base-Template. Die anderen Templates erben mithilfe des Befehls

```
{% extends "layout.html" %}
```

vom Base-Template und können die in `layout.html` definierten Blöcke befüllen. So muss z.B. die Seitenleiste und die Topbar nur einmal im Base-Template definiert werden und steht durch Vererbung in allen anderen Templates zur Verfügung.

Das Base-Template lädt auch Bootstrap [18] für das Styling, das Favicon und bindet bei Bedarf Vue.js ein.

4.6 Technologien

4.6.1 Flask

Flask [6] ist die Hauptkomponente des Backends der App. Mit Flask wird das Routing konfiguriert und die allgemeine Struktur der App definiert. Flask bietet im Vergleich zu Alternativen wie z.B. Django [11] nur die absolute Basisfunktionalität und wird dann sukzessive durch Erweiterungen ausgebaut. Im Folgenden soll kurz etwas zu den verwendeten Erweiterungen gesagt werden:

Flask-Limiter

Mit Flask-Limiter [20] kann die Anzahl der Zugriffe global oder auf spezifische Endpunkte begrenzt werden. Dies wird von MatchFinder genutzt, um einzelne Endpunkte vor Brute-Force-Angriffen zu schützen und die gesamte App gegen Angriffe robuster zu machen. Der Endpunkt für die allgemeine Authentifikation und die der Matrikelnummereingabe haben ein Limit von 5 Anfragen pro Minute.

Flask-SQLAlchemy

Flask-SQLAlchemy [21] ist die Schnittstelle zwischen Flask und SQLAlchemy, ein Object-Relation-Model (ORM) für Python [22]. Für weitere Informationen siehe Abschnitt 4.6.4.

Flask-WTF

Ähnlich wie Flask-SQLAlchemy ist Flask-WTF eine Schnittstelle zwischen Flask und einer bestehenden Python-Erweiterung, hier WTForms [23]. Für mehr Informationen siehe Abschnitt 4.6.5.

4.6.2 Jinja2

Jinja2 [24] ist die Template-Sprache, die standardmäßig bei Flask dabei ist. Mit Jinja können in HTML Basisoperationen wie if-Abfragen, Schleifen u. w. verwirklicht werden. Viel wichtiger ist jedoch die Möglichkeit, Daten, die dem Template mitgegeben werden, zu rendern. Wird Flask zu einem Template mit

```
render_template('template.html', parameter="Dies ist ein Parameter")
```

auch ein Parameter übergeben, kann dieser mit Jinja über `{{parameter}}` als Klartext gerendert und damit sein Inhalt angezeigt werden. Auch können ganze Objekte oder Arrays ins Template geladen und mit Jinja verarbeitet werden (z.B. ein HTML-Block für jeden Arrayeintrag).

4.6.3 Vue.js

Vue.js [25] ist ein progressives JavaScript Framework für dynamisches Rendern von HTML-Elementen. In Matchfinder gibt es mehrere Stellen, in denen Vue.js benutzt wird. Allgemein wird Vue in der App benutzt, um situationsbedingt Seitenelemente anzuzeigen oder zu verstecken.

Des Weiteren berechnet Vue einzelne Schwellwerte, von denen die Darstellung von Elementen abhängt. Vue wird in den Bereichen Verteilung auswerten, Verteilung Erstellen, Daten anlegen und Präferenzvergabe (Abschnitt 3.2, 3.4, 3.6 und 3.9) benutzt, um Seitenelemente bei bestimmten Bedingungen ein- und auszublenden.

4.6.4 SQLAlchemy

SQLAlchemy [22] bietet die Möglichkeit, ein Datenbankschema als ORM, also Object-Relation-Model anzulegen. Dies bedeutet, dass einzelne Tabellen als Klassen aus der Objektorientierung erstellt werden, welche die Tabellenspalten als Exemplarvariablen beinhalten. In diesen Exemplarvariablen werden dann Eigenschaften wie Typ, Referenz zu anderen Tabellen und Cascade-Bedingungen beschrieben.

Beispielsweise ist die Teilnehmer-Tabelle wie folgt definiert:

```
from flask_sqlalchemy import *

class Teilnehmer(db.Model):
    __tablename__ = "teilnehmer"
    id = db.Column(db.Integer, primary_key=True)
    list_id = db.Column(db.Integer, db.ForeignKey("teilnehmer_lists.id"),
                       nullable=False)
    praeferenzen = db.relationship("Praeferenz", cascade="all,delete",
                                   backref="teilnehmer", lazy=True)
    first_name = db.Column(db.String(80), nullable=False)
    last_name = db.Column(db.String(80))
    matr_nr = db.Column(db.Integer)
```

4.6.5 WTForms

Mit WTForms [23] können Formulare einfach und effizient über Klassen erstellt werden. Wie bei SQLAlchemy wird durch eine Klasse und ihre Variablen definiert, welche Arten von Feldern ein Formular hat. Besonders interessant ist hier die Möglichkeit zu überprüfen, ob ein Formular alle nötigen Informationen beinhaltet um als 'gültig' zu gelten. Mit WTForms können also Formulare und Formularinformationen ohne großen Aufwand in Objekte umgewandelt werden.

WTForms wird für die Erstellung der Teilnehmer und Themen per Formular benutzt. Hier ist im Voraus nicht bekannt, wie viele Teilnehmer und Themen erstellt werden sollen, also wie viele Formularfelder erzeugt werden müssen. WTForms erzeugt dynamisch die richtige Anzahl von TeilnehmerEntryForm (siehe Codebeispiel) und validiert diese später.

Die Klasse für das Formular der Teilnehmer ist wie folgt definiert:

```
from flask_wtf import FlaskForm
from wtforms import StringField, IntegerField, FieldList, FormField
from wtforms.validators import DataRequired

class TeilnehmerEntryForm(FlaskForm):
    first_name = StringField('Vorname', validators=[DataRequired()])
    last_name = StringField('Nachname')
    matr_nr = IntegerField('Matrikelnummer')
```

```
class TeilnehmerForm(FlaskForm):
    teilnehmer_name = StringField('Name', validators=[DataRequired()])
    teilnehmer = FieldList(FormField(TeilnehmerEntryForm),
        validators=[DataRequired()])
```

An Stellen, an denen die Komplexität der Formulare überschaubarer ist, konnte auf WTForms verzichtet werden. Dort werden die Formularfelder manuell definiert.

5 Dokumentation - Installationsanleitung

In diesem Kapitel ist beschrieben, welche Abhängigkeiten installiert sein müssen, um die Flask-App zu starten und sie weiterzuentwickeln. Darüber hinaus können weitere Informationen zur Funktionsweise von Flask-Applikationen und deren Entwicklung einem Installationsguide [26] und einem Tutorial [27] der Herausgeber von Flask entnommen werden.

Es wird empfohlen, alle benötigten Bibliotheken (Dependencies, Abhängigkeiten) in ein *Virtual Environment* zu installieren, um Konflikte mit bereits installierten Bibliotheken zu vermeiden (dies ist erneut in Abschnitt 4.1 unter *install.sh* erläutert).

Environment erstellen

Das virtual Environment erstellen:

```
python3 -m venv venv
```

Environment aktivieren

Das erstellte Environment aktivieren („betritt“die Umgebung):

```
. venv/bin/activate
```

Dependencies installieren

Benötigte Dependencies innerhalb es aktivierten Environment installieren (pip statt pip3):

```
pip install flask
pip install flask_sqlalchemy
pip install munkres
pip install pandas
pip install flask_limiter
pip install Flask-WTF
pip install qrcode
pip install Pillow
pip install bcrypt
```

Vor dem ersten Start

Vor dem ersten Start der Anwendung müssen ein paar Vorkehrungen getroffen werden, damit die App richtig startet.

1. Um die Cookies für die Authentifizierung zu verwenden, muss der App ein Secret-Key zugeteilt werden. Dieser wird in `MatchFinder/matchFinder/config.py` unter `SECRET_KEY` definiert und sollte möglichst sicher sein (siehe [28] für eine beispielhafte Generierung).

2. Vor dem ersten Start müssen auch die Datenbanktabellen und gleichzeitig auch die Passwörter für die Authentifizierung erstellt werden. Die zu erstellenden Passwörter werden als Plain-Text in eine Datei mit dem Namen `passwords.txt` geschrieben (Datei auf Root-Ebene erstellen). Anschließend müssen folgende Zeilen einkommentiert werden:

```
# __init__.py

with create_app().app_context():
    # ...
    from . import database_helper #<- this line
    database_helper.init_db()     #<- this line

# database_helper.py

def init_db():

reset_db() # <-- DANGER # this line

rows = password_model.Password.query.count()
if rows == 0:
    password_model.Password.query.delete()
    from . import password_helper
    password_helper.create_passwords()
```

Die ersten beiden Zeilen rufen `init_db()` in `database_helper.py` auf. Die Funktion `reset_db()` setzt die Datenbank zurück, indem alle Tabellen verworfen und neu angelegt werden. Anschließend speichern die unteren Zeilen die Passwörter aus `passwords.txt` als Hash in die Datenbank. **Wichtig:** Nach dem ersten erfolgreichen Start **müssen** die mit `# <- this line` kommentierten Zeilen wieder auskommentiert werden, um bei einem späteren App-Neustart nicht versehentlich die Datenbank zurückzusetzen.

Die App ausführen

Im Environment kann mit den folgenden Befehlen die App gestartet werden:

```
export FLASK_APP=matchFinder
export FLASK_ENV=development # <-- nur für development-Zwecke
flask run
```

Alternativ kann mit

```
flask run --host=0.0.0.0
```

die Flask App gestartet und allen Geräten im gleichen Netz zur Verfügung gestellt werden. Die App kann dann über `<IP DES HOSTS>:5000` erreicht werden.

Quellenverzeichnis

- [1] Felix Wolf. *MatchFinder Dokumentation auf GitHub*. Abgerufen am 02.02.2021. URL: <https://github.com/felix-wolf/MatchFinder/blob/master/README.md>.
- [2] Felix Wolf. *MatchFinder*. Abgerufen am 11.04.2021. URL: <https://matchfinder.wr.informatik.uni-hamburg.de/home/>.
- [3] Felix Wolf. *MatchFinder GitHub-Repository*. Abgerufen am 03.02.2021. URL: <https://github.com/felix-wolf/MatchFinder>.
- [4] Universität Hamburg. *Wissenschaftliches Rechnen*. Abgerufen am 02.02.2021. URL: <https://wr.informatik.uni-hamburg.de>.
- [5] Felix Wolf. *MatchFinder Spezifikation auf GitHub*. Abgerufen am 02.02.2021. URL: <https://github.com/felix-wolf/MatchFinder/blob/master/documentation/spezifikation.md>.
- [6] Pallets. *Welcome to Flask*. Abgerufen am 01.02.2021. URL: <https://flask.palletsprojects.com/en/1.1.x/>.
- [7] Wikipedia. *Hungarian algorithm - Wikipedia*. Abgerufen am 01.02.2021. URL: https://en.wikipedia.org/wiki/Hungarian_algorithm.
- [8] Doodle AG. *Doodle*. Abgerufen am 03.02.2021. URL: <https://doodle.com/de/>.
- [9] Fachbereich Informatik Universität Hamburg. *Bitpoll*. Abgerufen am 03.02.2021. URL: <https://bitpoll.mafiasi.de>.
- [10] Universität Hamburg. *Dashboard - Mafiasi*. Abgerufen am 11.04.2021. URL: <https://mafiasi.de/dashboard/>.
- [11] Django Software Foundation. *The Web framework for perfectionists with deadlines | Django*. Abgerufen am 01.02.2021. URL: <https://www.djangoproject.com>.
- [12] GitHub. *Api-Issue auf Github*. Abgerufen am 03.02.2021. URL: <https://github.com/fsinfuhh/Bitpoll/issues/41>.
- [13] Harold W Kuhn. "The Hungarian method for the assignment problem". In: *Naval research logistics quarterly* 2.1-2 (1955), S. 83–97.
- [14] Jack Edmonds und Richard M. Karp. "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems". In: *J. ACM* 19.2 (Apr. 1972), S. 248–264. ISSN: 0004-5411. DOI: 10.1145/321694.321699. URL: <https://doi.org/10.1145/321694.321699>.
- [15] Bevilacqua Research Corporation. *Munkres' Assignment Algorithm*. Abgerufen am 25.02.2021. URL: <https://brc2.com/the-algorithm-workshop/>.
- [16] PyPI. *munkres - PyPI*. Abgerufen am 01.02.2021. URL: <https://pypi.org/project/munkres/>.
- [17] Brian M. Clapper. *Munkres Implementation for Python - Documentation*. Abgerufen am 25.02.2021. URL: <https://software.clapper.org/munkres/>.
- [18] Bootstrap. *Bootstrap - The most popular HTML, CSS, and JS library in the world*. Abgerufen am 01.02.2021. URL: <https://getbootstrap.com>.

- [19] Flask. *API - Flask Documentation (1.1x)*. Abgerufen am 01.02.2021. URL: https://flask.palletsprojects.com/en/1.1.x/api/#flask.Flask.secret_key.
- [20] Ali-Akber Saifee. *Flask-Limiter*. Abgerufen am 01.02.2021. URL: <https://flask-limiter.readthedocs.io/en/stable/>.
- [21] Pallets. *Flask-SQLAlchemy*. Abgerufen am 01.02.2021. URL: <https://flask-sqlalchemy.palletsprojects.com/en/2.x/>.
- [22] SQLAlchemy. *SQLAlchemy*. Abgerufen am 01.02.2021. URL: <https://www.sqlalchemy.org>.
- [23] WTForms. *WTForms*. Abgerufen am 01.02.2021. URL: <https://wtforms.readthedocs.io/en/2.3.x/>.
- [24] Pallets. *Jinja*. Abgerufen am 01.02.2021. URL: <https://jinja.palletsprojects.com/en/2.11.x/>.
- [25] Evan You. *Vue.js*. Abgerufen am 01.02.2021. URL: <https://vuejs.org>.
- [26] Pallets. *Installation - Flask Documentation (1.1.x)*. Abgerufen am 02.02.2021. URL: <https://flask.palletsprojects.com/en/1.1.x/installation/#install-create-env>.
- [27] Pallets. *Tutorial - Flask Documentation (1.1.x)*. Abgerufen am 02.02.2021. URL: <https://flask.palletsprojects.com/en/1.1.x/tutorial/>.
- [28] Stack Exchange Inc. *Stackoverflow - python - Where do i get a SECRET_KEY for flask?* Abgerufen am 04.03.2021. URL: <https://stackoverflow.com/questions/34902378/where-do-i-get-a-secret-key-for-flask>.