Wahlinformationssystem + Stimmabgabe zur Bundestagswahl 2021 (2017)

Inken Grüner, Adnan Karamehic

Dokumentation

Inhalt

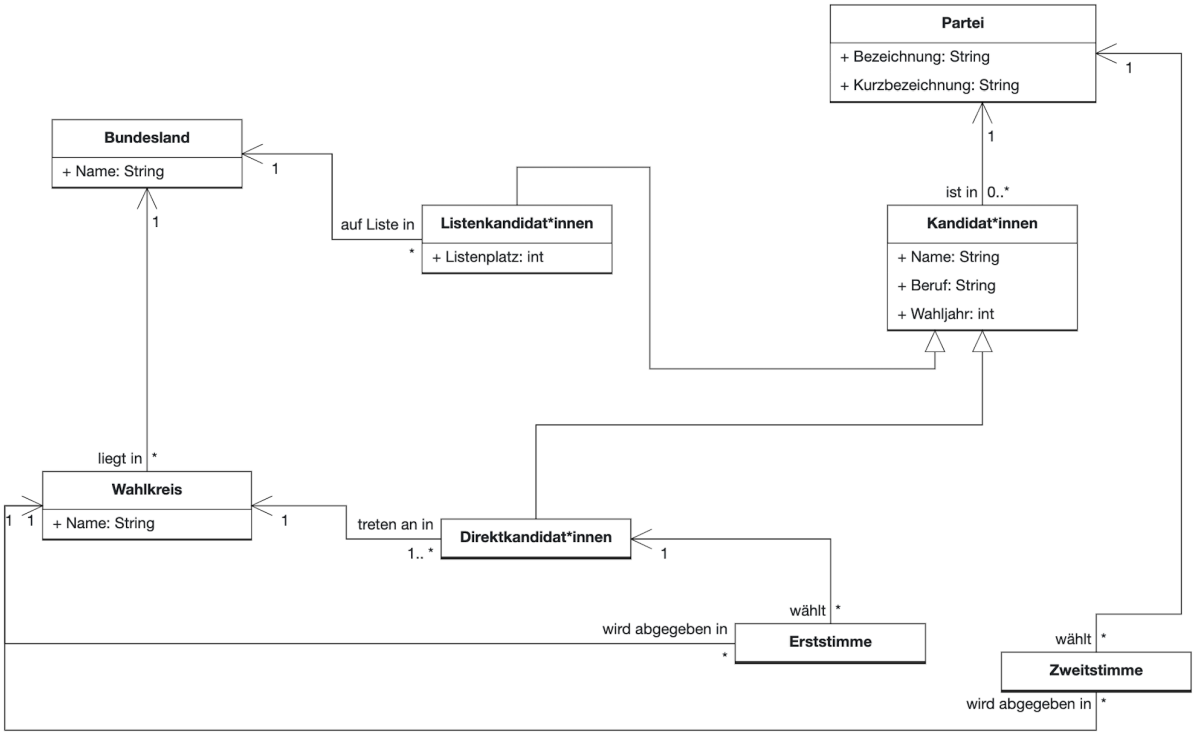
1. Aufgabenstellung
2. Modellierung
3. Lastenheft
4. Pflichtenheft
5. Erklärung Sitzverteilungsalgorithmus
6. Erklärung Wahlkonzept
7. Anleitung zum Aufsetzen und zur Verwendung des Systems
8. Benchmarks
9. Präsentation

1. Aufgabenstellung

- Erstellung eines Portals zur Information über laufende/abgeschlossene deutsche Bundestagswahlen
- Erstellung einer Komponente zur Stimmabgabe, inklusive Benutzer-Schnittstelle, über die ein:e Wähler:in eine Stimme abgeben kann, einem Sicherheitskonzept, um Wahlbetrug zu verhindern und Datenschutz sicherzustellen, und die Verarbeitung der abgegebenen Stimmen

2. Modellierung

a. UML-Diagramm des Wahlinformationssystems



b. Weitere Tabellen mit Aggregationsinformationen + Relevanz für die Stimmabgabe:

- Wahlkreisaggregation: {[WahlKreis: integer, WahlJahr: integer, UngueltigeErst: integer, UnGueltigeZweit: integer, AnzahlWahlBerechtigte: integer, AnzahlWaehlende: integer]}

- Wahlkreisstimmenaggregation: {[WahlJahr: integer, Partei: integer, WahlKreis: integer, AnzahlZweitStimmen: integer, AnzahlErstStimmen: integer]}
- Wahlkreisprozenterst: {[WahlJahr: integer, WahlKreis: integer, ParteiKurz: varchar(60), ProzentErstStimmen decimal(10, 8)]}
- Wahlkreisprozentzweit: {[WahlJahr: integer, WahlKreis: integer, ParteiKurz: varchar(60), ProzentZweitStimmen decimal(10, 8)]}
- Bundeslandaggregation: {[BundesLand: integer, WahlJahr: integer, UnGueltigeErst: integer, UnGueltigeZweit: integer, AnzahlWahlBerechtigte: integer, AnzahlWaehlende: integer, Bevoelkerung: integer]}
- Bundeslandstimmenaggregation: {[BundesLand: integer, WahlJahr: integer, Partei: integer, AnzahlErstStimmen: integer, AnzahlZweitStimmen: integer, DirektMandate: integer]}
- Bundeslandprozenterst: {[BundesLand: integer, WahlJahr: integer, ParteiKurz varchar(60), ProzentErstStimmen decimal(10, 8)]}
- Bundeslandprozentzweit: {[BundesLand: integer, WahlJahr: integer, ParteiKurz varchar(60), ProzentZweitStimmen decimal(10, 8)]}
- Deutschlandaggregation: {[WahlJahr: integer, UnGueltigeErst: integer, UnGueltigeZweit: integer, AnzahlWahlBerechtigte: integer, AnzahlWaehlende: integer, Bevoelkerung: integer]}
- Deutschlandstimmenaggregation: {[WahlJahr: integer, Partei: integer KEY, AnzahlErstStimmen: integer, AnzahlZweitStimmen: integer, DirektMandate: integer]}
- Admintokens: {[token char(64), WahlKreis: integer]}
- Tokenrange: {[WahlKreis: integer, TokenRangeMin: big integer, TokenRangeMax: big integer]}
- Tokens: {[token char(64), WahlKreis: integer]}

3. Lastenheft

Benutzerschnittstelle:

- Analysen:
    - Webseite, auf der Ergebnisse grafisch dargestellt sind
    - Verschiedene Reiter zur Auswahl von allgemeinen Informationen, Ergebnissen in ganz Deutschland, Bundesländern und Wahlkreisen
    - Darstellung der Ergebnisse als Diagramme und Tabellen
    - Wo möglich Button mit “Vergleich zum Vorjahr”

- Stimmabgabe:
    - Darstellung des Stimmzettels soll stark analoger Version ähneln, mit Erststimme links (Name, Beruf, Partei), Zweitstimme rechts (Partei: Kurz- und Langbezeichnung, erste 5 Listenkandidat:innen)

Funktionale Anforderungen:

- Das System soll verschiedene authorization levels haben für Wähler und Wahladministratoren
- Das System soll die Möglichkeit zur Verfügung stellen, sich Daten zur Wahl für ganz Deutschland, pro Bundesland und pro Wahlkreis anzuschauen.
- Das System soll die Möglichkeit zur Verfügung stellen, die Wahljahre 2021 und 2017 zu vergleichen bezüglich Gesamtergebnisse pro Wahlkreis, Bundesland und für ganz Deutschland
- Das Datenbanksystem soll so gestaltet sein, dass es auch als Backend in einem Wahlautomaten verwendet werden kann.
- Folgende Daten sollen verfügbar sein: Sitzverteilung, Mitglieder des Bundestags, Wahlkreisübersicht, Bundeslandübersicht, Überhangsmandate, Kandidatenübersicht, Erst- und Zweitstimmenverteilung, Parteien
- Batch-Loading von Stimmen
- Neuberechnung von Aggregationen
- Dokumentation

Nicht-Funktionale Anforderungen:

- Das Wahlgeheimnis muss gewahrt werden. (Datenschutz)
- Die Datenbank soll änderbar und erweiterbar sein.
- Das System soll performant und effizient sein, dafür ist auch Redundanz akzeptabel, die Ergebnisse sollen in unter 1s angezeigt werden (Ausnahme: Anfragen auf Einzelstimmen)
- Das System soll einfach und intuitiv verwendbar sein, so sollen Nutzer beispielsweise mit maximal zwei Klicks alle Informationen erreichen können
- Sub-menus und Überschriften sollen aussagekräftig und selbsterklärend sein
- Die Ergebnisse und Analysen müssen korrekt sein
- Das System muss sicher sein und darf nicht manipulierbar sein.

Abnahmekriterien:

- Es ist nicht möglich, eine Stimmabgabe zu einem/einer Wahlberechtigten zurückzuverfolgen.
- Bei der zufälligen Auswahl von 2 Wahlkreisen, Kandidat:innen, Bundesländern, Parteien werden korrekte Ergebnisse zurückgeliefert.
- Eine Dokumentation ist vorhanden.
- Nach Abgabe einer Stimme für eine zufällige Partei und ein:e zufällige Kandidat:in, sind die Informationen korrekt geupdatet.

4. Pflichtenheft

1. Zielsetzungskriterien

Muss:

- Das System muss die Möglichkeit zur Verfügung stellen, die Wahljahre 2021 und 2017 zu vergleichen
- Das System muss die Möglichkeit zur Verfügung stellen, sich Daten zur Wahl für ganz Deutschland, pro Bundesland und pro Wahlkreis anzuschauen.
- Das Datenbanksystem muss so gestaltet sein, dass es auch als Backend in einem Wahlautomaten verwendet werden kann.
- Das Wahlsystem muss in der Lage sein die Wahlmöglichkeiten, die eine analoge Wahl hat, ebenfalls anzubieten
- Folgende Daten müssen verfügbar sein: Sitzverteilung, Kandidatenübersicht, Parteien, Mitglieder des Bundestags, Überhangsmandate
- Neuberechnung von Aggregationen
- Das Wahlgeheimnis muss gewahrt werden. (Datenschutz)
- Die Ergebnisse und Analysen müssen korrekt sein
- Das System muss sicher sein und darf nicht manipulierbar sein, so können nicht autorisierte Nutzer beispielsweise nur vordefinierte Analysen betrachten

Soll:

- Das System soll verschiedene authorization levels für Wähler und Wahlleiter haben
- Dokumentation
- Die Datenbank soll änderbar und erweiterbar sein.
- Das System soll performant und effizient sein, dafür ist auch Redundanz akzeptabel.
- Das System soll einfach und intuitiv verwendbar sein, so sollen Nutzer beispielsweise mit maximal zwei Klicks alle Informationen erreichen können
- Sub-menus und Überschriften sollen aussagekräftig und selbsterklärend sein

Kann:

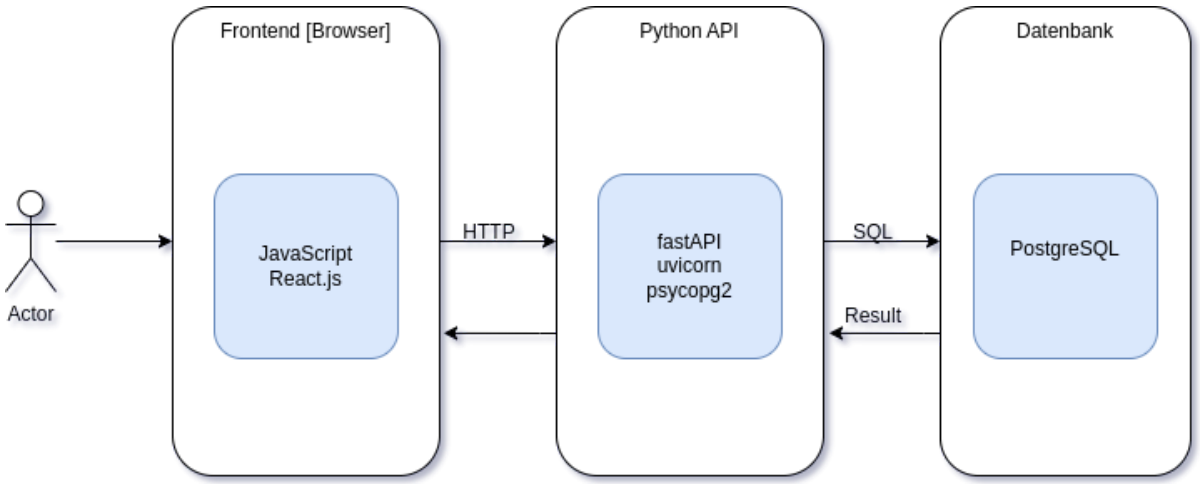
- Analysen pro Alter, Geschlecht, Bildung und anderen Faktoren
- Unterscheidung zwischen Direkt- und Briefwahl
- Mögliche Koalitionen anzeigen
- Graphische Darstellung (Landkarte…)

Nicht:

- Das System soll keine detaillierten Beschreibungen der Parteien und Kandidaten enthalten, sondern sich auf die wesentlichsten Punkte beschränken
- Wahlkreise sind die tiefste Analyseebene, für einzelne Wahlbezirke werden keine Analysen gemacht

2. Technische Umsetzung

- Das Datenbanksystem PostgreSQL wird verwendet, um die Daten zu speichern und abzufragen. Es ist über eine API mit dem Backend verbunden.
- Das Backend wird mit Python und dem Framework FastAPI implementiert. Es stellt eine API bereit, die es ermöglicht, auf die Daten in der Datenbank zuzugreifen und sie zu verarbeiten. Es stellt auch weitere Dienste, wie z.B. Authentifizierung bereit.
- Das Frontend wird mit React implementiert. Es stellt die Benutzeroberfläche dar und ermöglicht es dem Benutzer, auf die Daten in der Datenbank zuzugreifen und sie anzuzeigen. Es kommuniziert mit dem Backend über die bereitgestellte API.



3. GUI-Mockups

- Stimmabgabe: https://www.figma.com/proto/OctH4O7qQ4AmhdyXjVa10Y/Stimmabgabe?node-id=15%3A377&starting-point-node-id=15%3A377
- Analysen: https://www.figma.com/proto/YlsEJhVQRiwj2r7LmTTkfI/Bundestagswahl?node-id=3401%3A1155&starting-point-node-id=3401%3A1155

4. Glossar

- Wahlkreis: Ein Wahlkreis ist ein Gebiet, in dem die Bürger oder Bürgerinnen einen Abgeordneten oder eine Abgeordnete wählen. Der Abgeordnete oder die Abgeordnete vertritt das Gebiet in einem Parlament oder zum Beispiel in einem Stadtrat.
- Wahlbezirk: Ein Wahlbezirk ist eine organisatorische Einheit bei politischen Wahlen. Kein Wahlbezirk soll mehr als 2.500 Einwohner umfassen. Die Zahl der Wahlberechtigten eines Wahlbezirks darf aber nicht so gering sein, dass erkennbar wird, wie einzelne Wahlberechtigte gewählt haben (Wahlgeheimnis).
- Überhangmandat: Direktmandat, das eine Partei über die ihr nach dem Verhältniswahlrecht zustehenden Parlamentssitze hinaus gewinnt
- Ausgleichsmandat: Dient durch die Vergabe zusätzlicher Sitze dem Ausgleich von Überhangmandaten, so dass am Ende die Sitzverteilung nach dem Verhältnis der Zweitstimmen gewahrt bleibt.
- PostgreSQL: freies, objektrelationales Datenbankmanagementsystem
- FastAPI: modernes, hochleistungsfähiges Web-Framework zur Erstellung von APIs mit Python 3.7+
- uvicorn: Uvicorn ist eine ASGI-Webserver-Implementierung für Python. Es ist eine minimale Low-Level-Server-/Anwendungsschnittstelle für asynchrone Frameworks.
- psycopg2: PostgreSQL-Datenbankadapter für die Programmiersprache Python.

5. Erklärung Sitzverteilungsalgorithmus

Vorbereitung:

- Bestimme die Parteien, die in den Bundestag gelangen (über 5%-Hürde/ mehr als 3 Direktmandate/ als Partei nationaler Minderheiten)

Schritt 1:

- Verteile Sitzkontingente (insgesamt 598) nach Bevölkerungsanteil auf Bundesländer (mithilfe Divisorverfahren nach Sainte-Laguë):
    - Divisor = Bevölkerung (72,5 Mio) / Sitze gesamt (598) }
    - Sitze Bundesland = Bevölkerung Bundesland / Divisor
- Bei einem Rest von mehr oder weniger als 0,5 wird auf- oder abgerundet; bei einem Rest von genau 0,5 entscheidet das Los.
- Wenn bei der Summe der Sitz weniger als 598 herauskommt, muss der Divisor verringert werden, kommt mehr raus, muss er erhöht werden

Schritt 2:

- In jedem Bundesland wird die Mindestsitzzahl für jede Partei bestimmt.

- Dies geschieht wieder nach dem zuvor verwendeten Divisorverfahren.
- Grundsätzlich entspricht die Mindestsitzzahl dem Anteil der Zweitstimmen der Partei.
- Hat eine Partei mehr Direktmandate als Zweitstimmen prozentual, bekommt sie die Direktmandate als Sitze (Überhangmandate) → die Sitzzahl des Bundeslands erhöht sich.

Schritt 3:

- Voraussetzungen: für alle Parteien müssen die Mindestsitzansprüche (Summe BL) ohne und mit Überhang berechnet werden und es muss gespeichert werden, ob es in einem Bundesland einen Überhang gibt.
- Kleinster Parteien Divisor für Mindestsitzanspruch ohne Überhang:
    - Beispiel für eine Partei:
        - Bestimme Zweitstimmenanteil
        - Bestimme Mindestsitzanspruch ohne Überhang (summe aus Mindestsitzanspruch pro Bundesland, ohne dass anzahl Direktmandate berücksichtigt wurde)
        - Divisor = Zweitstimmenanteil/(Mindestsitzanspruch-0.5)
    - Wiederhole diesen Prozess für jede Partei und bestimme das Minimum
- Viertkleinster Parteien Divisor für Mindestsitzanspruch mit Überhang:
    - Beispiel für eine Partei:
        - Bestimme, ob bei einer Partei Überhang droht (dafür ist nicht ausschlaggebend, ob es bei der Summe einen ÜH gibt, sondern ob es in einem BL passiert)
        - Wenn ja: berechne für diese Partei Divisoren, in dem Zweitstimmen durch tatsächlichen Mindestsitzansprüche minus 0.5/ 1.5/ 2.5/ 3.5 geteilt werden (mit ÜH)
    - Wiederhole diesen Prozess für jede Partei, für die Überhang droht.
    - Bestimme aus allen errechneten Werten den viertkleinsten.
- Bestimme aus den beiden zuvor ermittelten Werten das Minimum: das ist die Obergrenze für die neue Divisorspanne.
- Teile alle Zweitstimmen durch diese Obergrenze und runde das Ergebnis
- Division durch Sitze + 0.5, das ist die Untergrenze.
- Wähle den Divisor aus der Spanne zwischen Ober- und Untergrenze.

Schritt 4:

- Erklärung für eine Partei QWE:
    - Berechne den Anfangsdivisor durch Zweitstimmenanteil_D/Sitzanzahl _ Bundestag
    - Für jedes Bundesland:
        - Teile die Zweitstimmen, die QWE in dem BL erhalten hat durch den Anfangsdivisor und runde das Ergebnis
        - Summiere die gerundeten Ergebnisse und vergleiche die Summe mit dem Ergebnis für QWE aus Schritt
        - Ist die Summe höher als die Anzahl der Sitze, die QWE zusteht, muss der Divisor heraufgesetzt werden.
    - Erhöhe den Divisor so lange, bis die Summe den tatsächlich erlaubten Sitzen entspricht.
- Wiederhole diesen Prozess für jede Partei

6. Erklärung Wahlkonzept

- Wähler sind Wahllokalen zugeordnet (wie momentan) -> man kann nicht in zwei verschiedenen Wahllokalen wählen
- Wähler betritt Wahllokal
- Muss sich bei Helfern anmelden (wie es auch momentan gelöst ist) -> stellt sicher, dass man wahlberechtigt ist und nicht ein zweites Mal das Wahllokal betreten kann
- Tokenverfahren:
    - Es wird für jeden Wähler vor Ort ein individueller Token generiert, die Tokengenerierung geschieht zufällig, so dass man Wähler:innen Token nicht zuordnen kann -> Sicherung Wahlgeheimnis
    - Dafür hat jeder Wahlkreis 10_000_000_000 Tokens zu Verfügung
    - Der generierte Token wird verschlüsselt (-> kann von niemandem außer Wähler vor Ort verwendet werden) einer dynamischen Tokentabelle hinzugefügt, die als weitere Spalte den zugehörigen Wahlkreis hat
    - Sollte der sehr unwahrscheinliche Fall (weit unter 1 Promille) auftreten, dass der Token schon existiert, wird dies als Fehlermeldung ausgegeben und ein neuer Token muss generiert werden
    - Tokentabelle ist mit HashIndex versehen, damit in O(1) überprüft werden kann…
        - …ob der Token existiert
        - …zu welchem Wahlkreis der Token gehört
- Wähler betritt Wahlkabine, kann sich mit dem Token authentisieren (Token besteht nur aus Zahlen, Whitelisting der Eingabe, um SQL-Injection zu vermeiden) und seine Stimme abgeben
- In dem Moment, in dem der Wähler seinen Token eingibt, wird dieser aus der dynamischen Tokentabelle gelöscht -> er kann mit seinem Token nicht zwei mal wählen
- Da Token mit Wahlkreis verbunden ist, werden abhängig vom Token die richtigen Kandidat:innen/ Listen angezeigt
- Selbst wenn 10_000_000 Tokens gleichzeitig aktiv wären, betrüge die Wahrscheinlichkeit, dass man zufällig einen anderen Token richtig errät, nur: 10_000_000/(10_000_000_000*299) = 0.0003%

7. Anleitung zum Aufsetzen und zur Verwendung des Systems

Die Query-API muss eingeschaltet sein und im Hintergrund laufen, damit die Analyse- und Voting-Websites funktionieren. PostgreSQL muss installiert sein.
Die API liefert JSON-Daten an das Frontend, das diese dann anzeigt.

Ausführen des Setups:

- Führen Sie das Skript `run_setup.sh`
    - Erstellung und Füllung der Tabellen in der Postgres-Datenbank
    - Ausführung des Algorithmus zur Berechnung der Sitzverteilung
    - Erstellung der Token-Tabellen

Eingabe der generierten Erststimmen/Zweitstimmen:

- Um die Stimmen zu generieren und die Tabellen mit den Erststimmen/Zweitstimmen zu füllen, führen Sie folgendes Python-Skript aus: `python3 stimmzettelgenerator.py`

Verwendung der Abfrage-API (vorläufig):

- `python3 -m uvicorn query_api:app --reload`

a. Analysen:

- Im Projektverzeichnis können Sie ausführen:
  - `npm install` um alle Abhängigkeiten zu installieren
  - `npm start` um die Webseite zu starten
  - Öffnen Sie http://localhost:3000 um die Webseite aufzurufen
  - Die Seite wird neu geladen, wenn Sie Änderungen vornehmen

b. Stimmabgabe:

- Im Projektverzeichnis können Sie ausführen:
  - `npm install` um alle Abhängigkeiten zu installieren
  - `npm start` um die Website zu starten
  - Öffnen Sie http://localhost:3000 um die Webseite aufzurufen
  - Die Seite wird neu geladen, wenn Sie Änderungen vornehmen

8. Benchmarks

Hardware:

- 1: MacBook Pro 2021, M1 Pro, 32GB RAM
- 2: MacBook Air 2022, M2 8GB RAM

Betriebssystem:

- macOS Ventura 13.0.1

Software:

- python 3.9.6
- Libraries:
  - requests

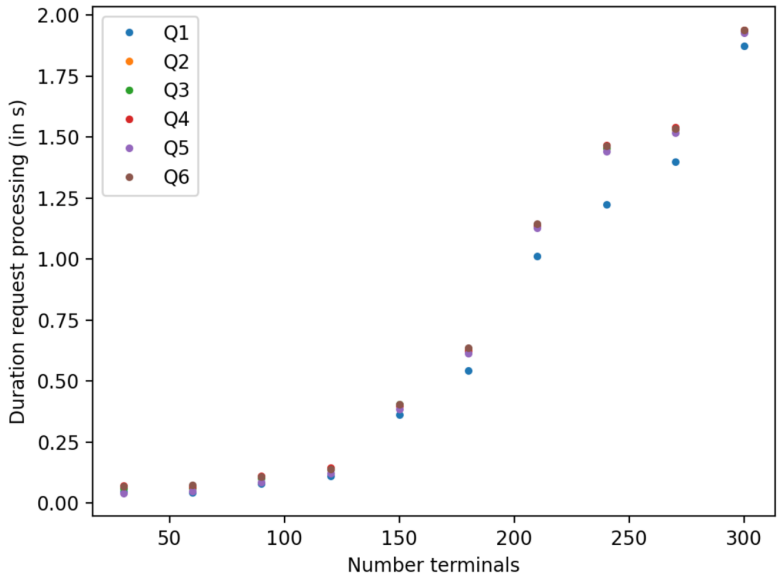
    - numpy
    - threading
    - timeit, time
    - statistics
    - random, re

1:

Jeder Thread führt 500 Anfragen aus, t: durchschnittliche Wartezeit, n: Anzahl Terminals

- t = 1, n = 10
    - Q1: 0.01988 s
    - Q2: 0.03280 s
    - Q3: 0.03246 s
    - Q4: 0.04074 s
    - Q5: 0.01110 s
    - Q6: 0.04153 s
- t = 4, n = 20
    - Q1: 0.032418
    - Q2:0.040356 s
    - Q3: 0.037987 s
    - Q4: 0.048809 s
    - Q5: 0.034193 s
    - Q6: 0.055246 s
- t = 4, n = 200
    - Q1: 0.08066 s
    - Q2: 0.08334 s
    - Q3: 0.08294 s
    - Q4: 0.10768 s
    - Q5: 0.06913 s
    - Q6: 0.08873 s
- t = 0.8, n = 100
    - Q1: 0.35390 s
    - Q2: 0.37053 s
    - Q3: 0.36213 s
    - Q4: 0.36485 s
    - Q5: 0.34843 s
    - Q6: 0.37525 s

2:



- Jeder Thread führt 150 Anfragen aus
- t = 2s

9. Präsentation

- https://tome.app/greenfashion/revolutionizing-seating-arrangements-with-the-sitzverteilungsalgorithmus-cldejmi3j0wfg5l3xsnirjnvk