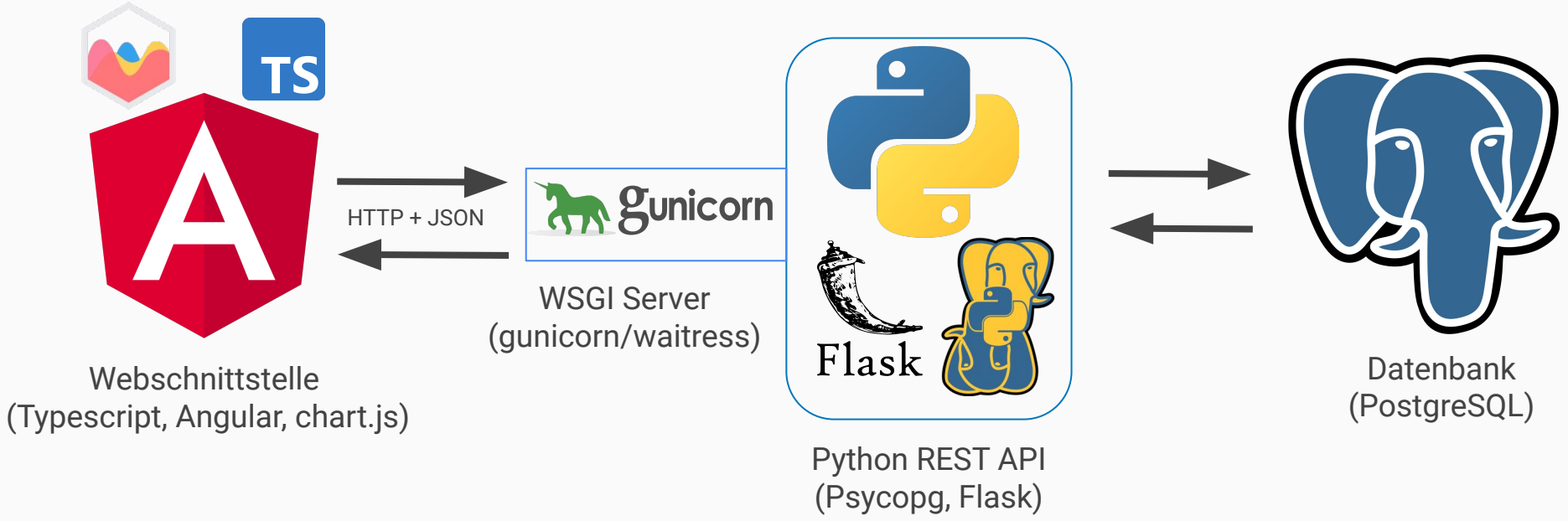
Wahlinformationssystem

Abschlusspräsentation

*Elitestudiengang Software Engineering - **Felix Rinderer, Tom Papke** - 7.2.2022*

Technologie-Stack



Webschnittstelle
(Typescript, Angular, chart.js)

WSGI Server
(gunicorn/waitress)

Python REST API
(Psycopg, Flask)

Datenbank
(PostgreSQL)

Demo

Einlesen der Daten

- **COPY** table **FROM STDIN**
- **SET** session_replication_role = ‘replica’ ➡ 10x schneller (danke Torben & Marius😉)
- insgesamt 18 Tables, 213 Columns

Berechnung der Sitzverteilung

- **iteratives** Divisorverfahren
  - zum Teil ausgelagert in PLPGSQL Funktion
- parallele Berechnung der Sitzverteilung für beide Wahlen mit 4 Fallunterscheidungen bzgl. des Wahljahres
- 29 CTEs und 450 LOC
- Laufzeit: ~85ms, (davon 5ms *planning time*), Kosten: 100.000
- ***trotzdem:*** Ergebnis wird materialisiert und beim Einfügen von Einzelstimmen über Trigger aktualisiert

Besonderheiten

- Anfragen liegen als Views (oder parametrisierten PLPGSQL functions) vor
- REST-Schnittstelle führt nur **SELECT** * **FROM** view **WHERE** … aus und konvertiert zu JSON
- laufende CI/CD (gitlab CI + heroku) mit automatisierten Tests:
  - Anzahl (Erst-/Zweit-/Ungültige) Stimmen
  - Sitzverteilung
  - gewählte Mitglieder des Bundestags
  - Überhangmandate

Leistungsfähigkeit

- Benchmarking über locust
- Maßnahmen zur Leistungssteigerung:
  - subqueries eliminieren
  - window functions
  - connection pooling
  - skalierter WSGI Server (#Worker = 2 * #Cores)

Leistungsfähigkeit

- n=100 und t=1s
  - ~180 Requests per Second (RPS)
  - durchschnittliche Antwortzeit: 14 ms
- n=100 und t=0.1s ~500 RPS
  - ~500 RPS
  - *keine Failures*
  - durchschnittliche Antwortzeit: 149 ms
- Ergebnisse ohne Materialisierung ähnlich

Ausprobieren unter

https://datenbanken-ws22-frontend.herokuapp.com/