* Our technology:
  + Postgres as DBMS
  + ported in parts to HyPer
  + Django as a web framework -> easy talking to postgres and nicer to develop with than JS.
* Election analysis
  + Core Features (Implemented all Queries)
    - Overview showing total composition (including comparison to last election!)
    - View showing all elected representatives
    - A list of all constituencies -> Click leads to:
    - Overview over Consitituency, including ALSO breakdown of first votes
    - Same for closest outcomes
    - and Overhang mandates
  + Special Features
    - Selection of Constituency by map. Makes visualizing the result much easier.
      * Example: anomalies, Linke in former DDR.
      * CDU performing worse in bigger population centres.
    - Maps are easy adaptable to show other data -> example: nice gimmick showing popularity of a candidate in a constituency.
  + Performance
    - Indexes and data aggregation where possible
    - Divisor selection done with binary search
  + Election Algorithm Implementation
    - (show code?) -> divided the algorithm into lots of subtasks, most implemented in SQL and represented by a view, others as postgers functions
* Voting
  + Anonymity through tokens. -> Wahlhelfer has to issue tokens,
  + Not influencing voters by neutral design
* HyPer
  + Porting Process – some features required work-arounds:
    - SERIAL For PK not recognized by hyper
    - When trying to run a SELECT query with too many results hyper crashed
    - GREATEST not implemented => replaced with select and CASE
    - RETURN NEXT needs to be rewritten in hypersciprt
    - TEMP tables not available in hyper
  + Performance
    - Aggregation speed was impressive
    - Not that much more performance without aggregation (from what we were able to test, comparable with postgres)