# Frontend

## General

* The header *must* contains logo, title and current information, as well as the call to action for voting.
* A navigation bar at the left side *must* list all available top level pages.
* Lower level pages *must* be accessible via dropdown menu. These menus must work even without JavaScript.
* The UI *must* work in all current Browsers with a browser market share of more than 0.5%
* The UI *should* be responsive. Markup for mobile and desktop sited must be the same. CSS and JavaScript assets may vary.
* The UI *must* presented in a neutral way, not favouring certain results, candidates, parties or opinions
* All listings of any kind *must* be alphabetically sortable, ascending and descending.
* The UI *should* be evaluated by heuristic criteria.

## Analysis & Information

* Results *must* be displayed as graphs.
* Additional tabular information *must* be available for display.
* When JavaScript is disabled a fall-back *must* exist. The tabular information being the default option.
* Concerning the types of graphs see the Lastenheft and Wahlanalysen documents.
* Map based results *should* use an SVG graphic for display and interaction.

## Voting

* The voting ballot *must* be a separate page. It should be able to show it as an inline frame on the information page.
* Authorization to vote *must* be done using the ID number and an identification token.
* A general explanation section *must* be on the ballot. It can be collapsed.
* For each term on the ballot an explanation *must* be displayable via tooltip.
* Voting *must* include Erststimme and Zweitstimme
* Selection of candidates / parties *must* mutually exclusive (radios).
* An additional radio for invalidation *must* be displayed when the ballot is valid.
* Invalidation of both individually *must* be possible
* An invalid voting *must* be clearly indicated
* When invalid, an explanation *must* be displayed explaining what an invalid ballot means

# Backend

## Voting

* Every citizen with the right to vote *must* not vote more than once per election, entering valid or invalid Erstimme and Zweitstimme
* Citizens *must* not vote in any other Wahlbezirk than the one they are registered in *x*or by Briefwahl.
* Voting *must* only work for parties and candidates that are nominated in that year / in that Wahlkreis.
* It *should* be possible for votes be inserted into the database via batch loading interface.

## Nominations

* Parties *must* not be nominated more than once but only once per year
* Parties *must* not hand in more than one Landesliste per federal state per year
* Candidates *must* not be listed on more than one Landesliste per year
* Candidates *must* not run for more than one Wahlkreis per year
* Parties *must* not support more than one single candidate per Wahlkreis per year

## Evaluation

* Evaluation of election results *must* follow the current system (Saint Lague)
* (Preliminary) Results *should* be updated in real time as soon as voting occurs
* Sending updated results to the clients *should* use WebSockets[[1]](#endnote-1)
* A defined interface *could* exists to change the seat distribution method (e.g. from Saint Lague to D’Hondt)

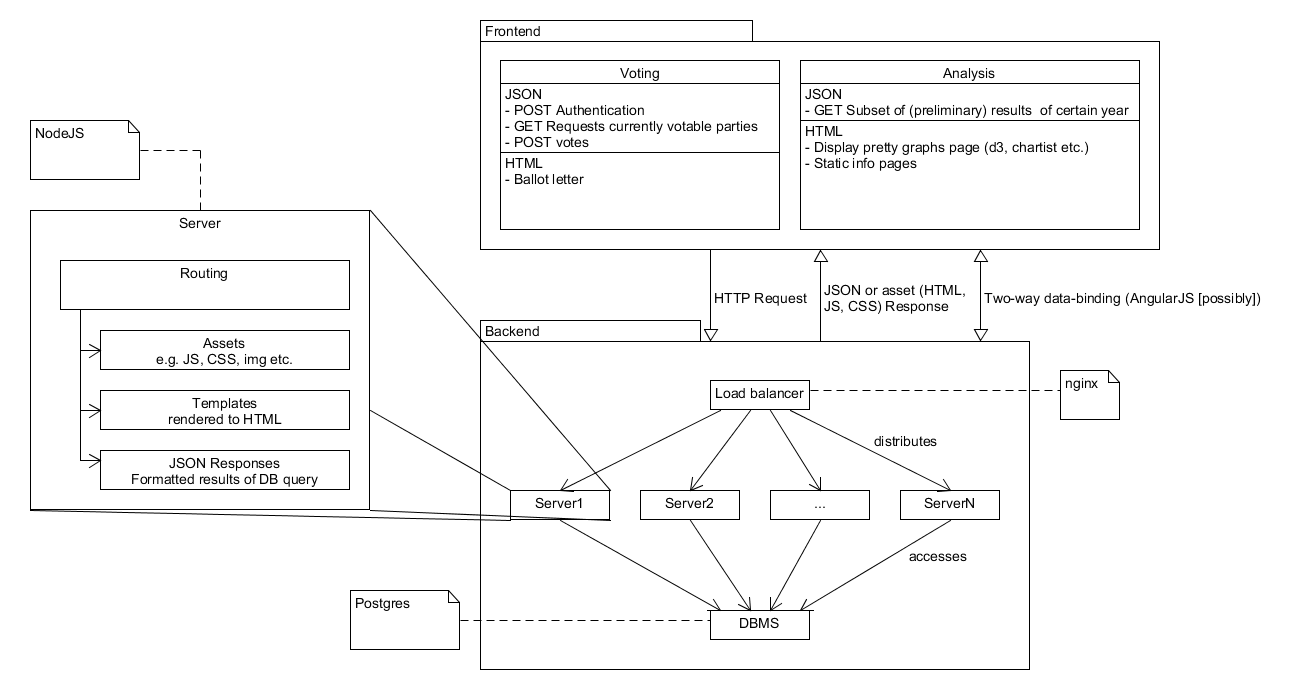
## Database

* The Database *will not* be redundant to prevent synchronization overhead.
* Constraints on number of votes and right to vote, Wahlbezirk, number of candidates, party membership, Landeslisten as listed above *must* be implemented on database level (check constraints).
* Error *should* be caught early, automatic recovery is preferred.
* Errors can be propagated, however error messages *must* be easy to understand and easy to recover from.
* Accumulation of results per Wahlkreis and federal state *must* be implemented with views.
* Results *must* be held in views for as long as the election is going on.
* Final results *must* be stored persistently once the election is over and official results have been calculated.
* Seat distribution according to the distribution system *should* be calculated in the database as a view.
* Identification tokens *must* be stored in hashed and salted format. Hashing must occur on the server. A slow hash function is preferred.
* Altering the evaluation system for seat distribution *should* only require the change of a single database query or backend function. Subqueries and sub-function not included.
* Geographic data *should* be stored in the database as well (outlines of state etc.)

## Server

* Servers should be redundant behind a load-balancing solution (e.g. nginx[[2]](#endnote-2))
* Load balancing will not be redundant.
* Serving assets[[3]](#endnote-3) and answering queries could be separated on dedicated servers.
* Voting *must* require authentification.
* Requesting results / accumulated data will not require authentification.
* Raw data *will not* be accessible
* Data that may offer insight into individual votes (e.g. low number of total votes, low entropy) *must* be withheld.
* A defined and **documented** API for accessing and altering data *must* exist.
* Error responses *must* have meaningful error messages and *must* be documented.
* Results that are unlikely to change (e.g. old election results) *should* be cached.

# General System Architecture



# Non-functional Requirements

* Privacy
  + Within the database there is no association between citizens and their votes. Within the database no such relation can be derived from other data.
  + Data aggregations that are accessible for user must be limited in a way that ensures no information can be inferred for the individual data subsets.
* Reliability and performance
  + The system must handle at least **100.000** voting transactions nearly simultaneous
  + The system must handle at least **200.000** analysis requests per minute
  + both must be handled at the same time
  + Response time for voting transactions of less than **1 second**
  + Response time for analysis requests of less than **3 second**
* Robustness
  + Data is stored in a way that prevents data loss due to hardware or software error
  + Backend systems have automatic failure recovery / restart capabilities.
* Security
  + A secure way of authenticating must be required for the user to cast his vote
  + All data must transported in a way that prevents unauthorized access.
  + Access to the database and the raw data must be restricted.

1. https://developer.mozilla.org/en-US/docs/Web/API/WebSocket [↑](#endnote-ref-1)
2. https://www.nginx.com [↑](#endnote-ref-2)
3. Assets include static HTML files, JavaScript and CSS [↑](#endnote-ref-3)