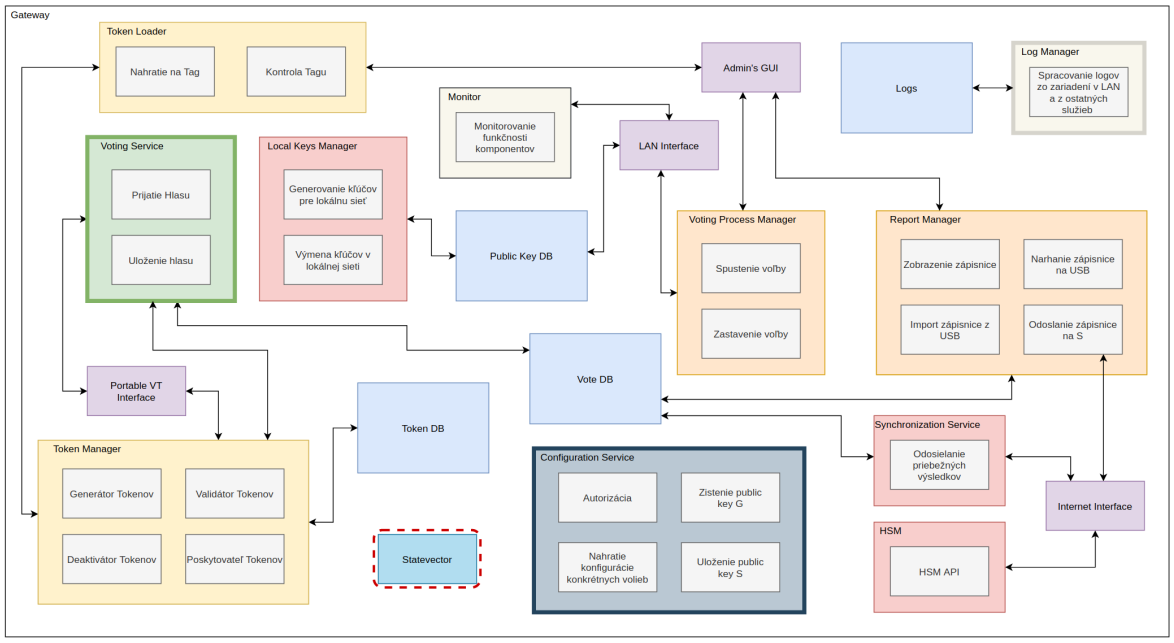
1. Gateway

Gataway je zariadenie nachádzajúce sa vo velebnej miestnosti. V miestnosti sa nachádza vždy len jeden gateway. Zabezpečuje komunikáciu medzi volebnými terminálmi a serverom. Gateway obsahuje lokálnu databázu pre hlasy aj tokeny, takže dokáže fungovať aj bez pripojenia k internetu a vie urobiť synchronizáciu na inom mieste, kde je internet dostupný.

Gataway sa má nachádzať na chránenom mieste a pristupovať k nemu smú iba členovia volebnej komisie napríklad pri spustení alebo zastavení volieb alebo nahrávaní tokenov na NFC tagy.

1.1 Architektúra

popis architektury



1.2 Mikroslužby a ich smerovanie

V nasledujúcej tabuľke uvádzame zoznam mikroslužieb a statických súborv na gateway-i a ich smerovanie.

Service	Path
Voting service	`/voting-service-api/`
Synchronization service	`/synchronization-service-api/`
Voting process manager	`/voting-process-manager-api/`
Token manager	`/token-manager-api/`
State vector	`/statevector/`
config.json	`/statevector/config/config.json`
datamodels.yaml	`/statevector/config/datamodels.yaml`

1.3 State vector

Služba zodpovedná za udržiavanie aktuálneho stavu gateway-u.

Udržuje tieto stavy: - `state_election` - stav volieb - `state_write` - stav zapisovačky - `state_register_terminals` - stav registrácie terminálov - `office_id` - id volebnej miestnosti - `pin` - pin kód k GUI aplikácii na gataway-i - `server_key` - verejný kľúč servera - `server_address` - adresa servera

1.3.1 Konfiguračný súbor

Konfiguračný súbor obsahuje celú konfiguráciu volieb pre konkrétnu volebnú miestnosť. Je dostupný ako statický súbor na adrese `/statevector/config/config.json` pomocou Nginx.

1.3.2 Popis API

1.3.2.1 hello___get

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('/gateway/statevector/', headers = headers)

print(r.json())
```

`GET /`

Hello

Sample testing endpoint

Example responses

200 Response

```
{
  "message": "string"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

1.3.2.2 get_state_election_state_election_get

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
}
```

```
5
r = requests.get('/gateway/statevector/state_election', headers =
    headers)
7
print(r.json())
```

`GET /state_election`

Get State Election

Get election state string 0 or 1

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

1.3.2.3 set_state_election_state_election_post

Code samples

```
1 import requests
  headers = {
3   'Content-Type': 'application/json',
    'Accept': 'application/json'
5 }

7 r = requests.post('/gateway/statevector/state_election', headers =
    headers)

9 print(r.json())
```

`POST /state_election`

Set State Election

Set election state string 0 or 1

Body parameter

```
"string"
```

Parameters

Name	In	Type	Required	Description
body	body	string	true	none

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline
422	Unprocessable Entity	Validation Error	HTTPValidationError

Response Schema

This operation does not require authentication

1.3.2.4 get__state__write__state__write__get

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('/gateway/statevector/state_write', headers =
    headers)

print(r.json())
```

`GET /state_write`

Get State Write

Get write state string 0 or 1

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

1.3.2.5 set__state__write__state__write__post

Code samples

```
import requests
headers = {
  'Content-Type': 'application/json',
  'Accept': 'application/json'
}

r = requests.post('/gateway/statevector/state_write', headers =
   headers)

print(r.json())
```

`POST /state_write`

Set State Write

Set write state string 0 or 1

Body parameter

```
"string"
```

Parameters

Name	In	Type	Required	Description
body	body	string	true	none

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline
422	Unprocessable Entity	Validation Error	HTTPValidationError

Response Schema

This operation does not require authentication

1.3.2.6 state__register__terminals__state__register__terminals__get

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
}

```

```
r = requests.get('/gateway/statevector/state_register_terminals',
    headers = headers)

print(r.json())
```

`GET /state_register_terminals`

State Register Terminals

Get terminals registration state string 0 or 1

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

1.3.2.7 set_state_register_terminals_state_register_terminals_post

Code samples

```
import requests
headers = {
  'Content-Type': 'application/json',
  'Accept': 'application/json'
}

r = requests.post('/gateway/statevector/state_register_terminals',
    headers = headers)

print(r.json())
```

`POST /state_register_terminals`

Set State Register Terminals

Set register terminals state string 0 or 1

Body parameter

```
"string"
```

Parameters

Name	In	Type	Required	Description
body	body	string	true	none

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline
422	Unprocessable Entity	Validation Error	HTTPValidationError

Response Schema

This operation does not require authentication

1.3.2.8 get__office__id__office__id__get

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('/gateway/statevector/office_id', headers = headers)

print(r.json())
```

`GET /office_id`

Get Office Id

Get office id

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

1.3.2.9 get__pin__pin__get

Code samples

```
import requests
```

```
headers = {
  'Accept': 'application/json'
}

r = requests.get('/gateway/statevector/pin', headers = headers)

print(r.json())
```

`GET /pin`

Get Pin

Get pin

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

1.3.2.10 get_server_key_server_key_get

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('/gateway/statevector/server_key', headers = headers)

print(r.json())
```

`GET /server_key`

Get Server Key

Get server key

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

1.3.2.11 get_server_address_server_address_get

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('/gateway/statevector/server_address', headers =
   headers)

print(r.json())
```

`GET /server_address`

Get Server Address

Get server address

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

1.3.2.12 Schemas

1.3.2.12.1 HTTPValidationError

```
{
  "detail": [
    {
      "loc": [
        "string"
      ],
      "msg": "string",
```

```
      "type": "string"
    }
  ]
}
```

HTTPValidationError

Properties

Name	Type	Required	Restrictions	Description
detail	[ValidationError]	false	none	none

1.3.2.12.2 ValidationError

```
{
  "loc": [
    "string"
  ],
  "msg": "string",
  "type": "string"
}
```

ValidationError

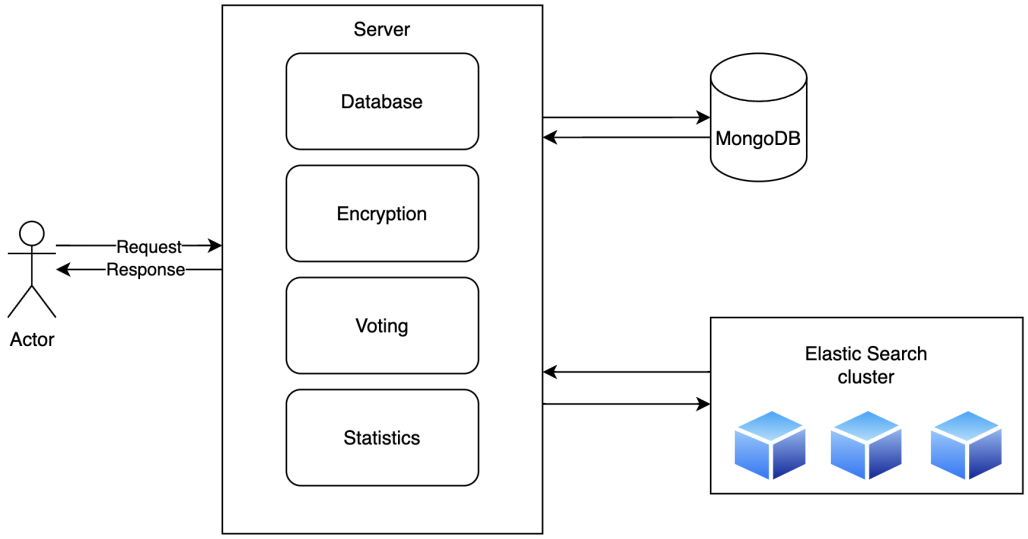
Properties

Name	Type	Required	Restrictions	Description
loc	[string]	true	none	none
msg	string	true	none	none
type	string	true	none	none

2. Server

Server je centrálna jednotka na spracovanie hlasov z volebných miestností. Server po prijatí požiadavky na uloženie hlasov zabezpečí ich validáciu, následné spracovanie a uloženie. Po úspešnom vykonaní vráti odpoveď v ktorej špeifikuje koľko hlascov bolo spracovaným. Uložené hlasy sa priebežne indexujú do technlógie Elastic Search, z ktorej sú následne získavené pri volaní koncových bodov na získanie výsledkov a štatistík.

2.1 Architektúra



2.2 Inštalácia

2.2.1 Závislosti

Pre spustenie docker kontajnerov je potrebné mať nainštalované technológie Docker, Docker compose. Pre účely vývoja ďalej odporúčame mať nainštalovaný jazyk Python, nástroj na testovanie koncových bodov ako Postman alebo Insomnia a nástroj na manipuláciu s MongoDB ako napríklad MongoDB Compass.

Knižnice pythonu su definované v textovom súbore requirements.txt, ktoré si nainštalujete príkazom: `pip install -r requirements.txt`

2.2.2 Spustenie

Lokálne samostatné spúšťanie jednotlivých častí potrebných pre chod serveru neodporúčame, z dôvodu radu problémov ktoré môžu vzniknúť. Najjednoduchším spôsobom je spustenie pomocou orchestrátora docker compose.

Prejdite do koreňového adresára servera a spustite nasledujúci príkaz.

```
docker compose up -d --build
```

Po vybudovaní by mali bežať všetky služby servera (MongoDB, FastAPI server a Elastic Search Cluster)

Zobrazenie všetkých dostupných koncových bodov servera navštívte adresu `http://localhost:8222/docs`

2.2.3 Ako si naimportovať skúšobné dáta a pripraviť Elastic Search cluster

V API docs špecifikácii spustite volania na jednotlivé koncové body v nasledovnom poradí: 1. `/database/import-data` 2. `/database/seed-votes` (s počtom hlasov, ktoré sa majú vygenerovať) 3. `/elastic/setup-elastic-vote-index` (Elastic uzly musia byť pred týmto volaním funkčné,

ak nie sú, skontrolujte prosím sekciu týkajúcu sa problému s malou pamäťou dockera.) 4. `/elastic/synchronize-votes-es` (Synchronize votes in batches)

2.2.4 Problém s Elastic search pamäťou

V prípade chybovej hlášky spomínajúcej prekročenie limitu pamäte, je potrebné nastaviť premmennú vm.max_map_count v kerneli dockeru na najmenej 262144.

V závislosti od operačného systému použite jeden z nasledovných príkazov:

```
docker-machine ssh
sudo sysctl -w vm.max_map_count=262144

wsl -d docker-desktop
sysctl -w vm.max_map_count=262144
```

Na apple zariadeniach je možné toto nastavenie zmeniť priamo v nastaveniach Docker Desktop App v sekcii: Settings -> Resources -> Advanced -> Memory. 8Gb pamäte by malo postačovať.

2.2.5 Testovanie vnútri dockeru

Jednotkové testovanie vykonávané v dockeri spustíte nasledovným príkazom v priečinku zdrojových kódov servera:

```
docker-compose -p test-server -f docker-compose.test.yml up --build
    --exit-code-from server --renew-anon-volumes
```

Dostupné príznaky: - -p - preped prefix to container names - -f - docker-compose yml file - –build - build images if changed sources - –exit-code-from - get overall exit code from specified container - –force-recreate - recreate all containers - –renew-anon-volumes - delete anonym volumens

Pre zastavenie kontajnerov použite príkaz:

```
docker-compose -f docker-compose.test.yml down
```

2.3 Databáza

Server používa na ukladanie dát dokumentovú databázu MongoDB. Aj keď je do MongoDB vkladať dáta s rôznymi atribútmi, používame modely jednoitlivých dátových entít, ktoré špecifikujú štruktúru objektu a definujú typy jeho atribútov. Pracujeme s nasledujúcimi kolekciami: - votes - parties - candidates - polling_places - key_pairs

Štruktúra uloženého hlasu:

```
class Vote(BaseModel):
    token: str
    party_id: Optional[int] = None
    election_id: str
    candidate_ids: List[int] = []
```

Ďalej sa počas spracovania hlasov dynamicky pridajú dva atribúty a to:

```
polling_place_id: int
synchronized: bool
```

Atribút polling_place_id slúži na spojenie hlasu s miestnosťou, v ktorej bol zvolený a atribút synchronized, ktorý indikuje, či bol daný hlas už zandexovaný do Elastic Searchu.

Štruktúra politickej strany:

```
class Party(BaseModel):
    id: int = Field(..., alias="_id")
    party_number: int
    name: str
    official_abbr: str
    abbr: str
    image: str
    image_bytes: str
    color: str
    candidates: List[Candidate] = []
```

Dátová štruktúra politickej strany obsahuje základné údaje ako názov, skratka a číslo a doplnkové údaje ako farba a logo, ktoré sa používajú v štatistickej aplikácii. Ďalej strana obsahuje zoznam kadidátov, ktorý sú reprezentovaný vlastným modelom.

Štruktúra volebnej miestnosti:

```
class PollingPlace(BaseModel):
    id: int = Field(..., alias="_id")
    region_code: int
    region_name: str
    administrative_area_code: int
    administrative_area_name: str
    county_code: int
    county_name: str
    municipality_code: int
    municipality_name: str
    polling_place_number: int
    registered_voters_count: int
```

Dátová štruktúra volebnej miestnosti obsahuje ifnormácie o územných celkoch, v ktorých sa daná miestnosť nachádza. Tieto údaje budú následne použité na prepočítavanie výslekov pre rôzne lokality (obce, okresy a kraje).

Štruktúra kandidáta:

```
class Candidate(BaseModel):
    id: int = Field(..., alias="_id")
    party_number: int
    order: int
    first_name: str
    last_name: str
    degrees_before: str
    age: int
    occupation: str
    residence: str
```

Dátová štruktúra kandidáta obsahuje základné údaje o kandidátovy, ktoré sú použité na zobrazovanie výsledkov a obsahuje taktiež prepojenie na politickú stranu, ktorej je súčasťou.

Štruktúra kľúčového páru:

```
class KeyPair(BaseModel):
    id: int = Field(..., alias="_id")
    polling_place_id: int
    private_key_pem: str
    public_key_pem: str
    g_private_key_pem: str
    g_public_key_pem: str
```

Kľúčový pár je špecifický pre každú volebnú meistnosť a jeho privátnym kľúčom je dešifrovaná iba kominikácia, ktorá prichádza z tejto volebnej miestnosti. Tento krok zvyšuje bezpečnosť komunikácie.

2.3.1 Popis API

2.3.1.1 schema_database_schema_get

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('/database/schema', headers = headers)

print(r.json())
```

`GET /database/schema`

Schema

Get all collections from database

Example responses

200 Response

```
{
  "collections": []
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Collections

This operation does not require authentication

2.3.1.2 import_data_database_import_data_post

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
}
```

```
5
r = requests.post('/database/import-data', headers = headers)
7
print(r.json())
```

`POST /database/import-data`

Import Data

Example responses

200 Response

```
{
  "status": "string",
  "message": "string"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Message

This operation does not require authentication

2.3.1.3 seed_data_database_seed_data_post

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.post('/database/seed-data', params={
  'number_of_votes': '0'
}, headers = headers)

print(r.json())
```

`POST /database/seed-data`

Seed Data

Parameters

Name	In	Type	Required	Description
number_of_votes	query	integer	true	none

Example responses

200 Response

```
{
```

```
  "status": "string",
  "message": "string"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Message
422	Unprocessable Entity	Validation Error	HTTPValidationError

This operation does not require authentication

2.3.1.4 seed_votes_database_seed_votes_post

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.post('/database/seed-votes', params={
  'number_of_votes': '0'
}, headers = headers)

print(r.json())
```

`POST /database/seed-votes`

Seed Votes

Parameters

Name	In	Type	Required	Description
number_of_votes	query	integer	true	none

Example responses

200 Response

```
{
  "status": "string",
  "message": "string"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Message
422	Unprocessable Entity	Validation Error	HTTPValidationError

This operation does not require authentication

2.4 Generovanie hlasov

Pre účely vývoja a testovania odporúčame generovať hlasy vo väčšom počte. Celý postup generovania spolu s naindexovaním prijatých hlasov dosiahnete vykonaním volaní v nasledujúcom poradí: 1. `/database/import-data` 2. `/database/seed-votes` (s počtom hlasov, ktoré sa majú vygenerovať) 3. `/elastic/setup-elastic-vote-index` (Elastic uzly musia byť pred týmto volaním funkčné, ak nie sú, skontrolujte prosím sekciu týkajúcu sa problému s malou pamäťou dockera.) 4. `/elastic/synchronize-votes-es`

V prípade potreby dogenerovania ďalších hlasov stačí vykonať kroky 2 a 4.

Ak potrebujete vymazať existujúce hlasy len z Elastic Searchu stačí spustiť krok č. 3.

V prípade potreby vymazania hlasov z MongoDB vykonajte príkazy 1 a 3.

2.5 Hlasovanie

Základná myšlienka hlasovania spočíva vo validácii prichádzajúceho zoznamu hlasov z gateway-u, ktorá musí prejsť niekoľkými krokmi. Samotný zoznam prichádzajúcich hlasov je zašifrovaný pomocou vlastnej knižnice *electiersa*, ktorého štruktúra je následovná:

```
class VoteEncrypted(BaseModel):
    encrypted_message: str
    encrypted_object: str

class VotesEncrypted(BaseModel):
    polling_place_id: int
    votes: List[VoteEncrypted] = []
```

Ak je validácia úspešná, spomínaný zoznam prichádzajúcich hlasov sa uloží do kolekcie *votes* a informuje používateľa. V opačnom prípade, server vráti špecifickú hlášku, vďaka ktorej používateľ bude vedieť, v akom kroku bola validácia neúspešná.

2.5.1 Validácia

- *id* volebnej miestnosti sa musí nachádzať v kolekcii *key_pair*
- počet kandidátov nesmie byť väčší ako 5
- každý kandidát sa v prichádzajúcom hlase môže vyskytovať iba raz
- nezadaná politická strana nesmie obsahovať žiadneho kandidáta
- kandidát musí patriť do správne politickej strany
- v kolekcii *votes* sa nesmie nachádzať duplitcitná kombinácia tokenu a *id* volebnej miestnosti
- v príchádzajúcom zozname hlasov sa nesmie nachádzať duplicitný token
- *id* volieb musí byť totožné s tým, ktoré sa nachádza v konfiguračnom súbore *config.py*

2.5.2 Popis API

2.5.2.1 vote_elections_vote_post

Code samples

```
import requests
headers = {
  'Content-Type': 'application/json',
  'Accept': 'application/json'
```

```
}

r = requests.post('/elections/vote', headers = headers)

print(r.json())
```

`POST /elections/vote`

Vote

Process candidate's vote

Body parameter

```
{
  "polling_place_id": 0,
  "votes": [
    {
      "encrypted_message":
         "36AMNvcpAWdHAXKCSWexgyjxrt7xeWwhOf+oUMBqip/C051EZWlN4N4x3hVPwwIQh/l78suUNY
      "encrypted_object":
         "1b5B/LAg2/38mot9jYzRpa9O6YwrXDilpspPrGrnTKKYUXVfQ9JhW5JIGoP6FuQBXM2XzlcXkb
    }
  ]
}
```

Parameters

Name	In	Type	Required	Description
body	body	VotesEncrypted	true	none

Example responses

200 Response

```
{
  "status": "string",
  "message": "string"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Message
400	Bad Request	Bad Request	Message
422	Unprocessable Entity	Validation Error	HTTPValidationError

This operation does not require authentication

2.5.2.2 get_voting_data_elections_voting_data_get

Code samples

```python
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('/elections/voting-data', headers = headers)

print(r.json())
```

`GET /elections/voting-data`

Get Voting Data

Downlaod voting data json using command curl http://localhost:8222/elections/voting-data > config.json

Example responses

200 Response

```
{
  "polling_places": [],
  "parties": [],
  "key_pairs": [],
  "texts": {
    "elections_name_short": {
      "sk": "string",
      "en": "string"
    },
    "elections_name_long": {
      "sk": "string",
      "en": "string"
    },
    "election_date": {
      "sk": "string",
      "en": "string"
    }
  }
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	VotingData

This operation does not require authentication

2.5.2.3 get_zapisnica_elections_zapisnica_get

Code samples

```python
import requests
headers = {
  'Accept': 'application/json'
```

```
}

r = requests.get('/elections/zapisnica', headers = headers)

print(r.json())
```

`GET /elections/zapisnica`

Get Zapisnica

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

2.6 Výsledky a štatistiky

Výsledky volieb sa rátajú na serveri pomocou dát získaných z Elastic Searchu a funkcie `calcualte_winning_parties_and_seats`.

Na zobrazenie výsledkov ponúkame viaceré koncové body ktoré výsledky vrátia s inou agregáciou alebo vráti len ich časť aby odpoveď nebola príliš veľká.

2.6.1 Dostupné koncové body:

- `/elastic/get-parties-results`
  - získanie výsledkov politických strán bez kandidátov
- `/elastic/get-party-candidate-results`
  - získanie výsledkov všetkých strán a kandidátov
- `/elastic/get-candidates-results`
  - získanie výsledkov všetkých kandidátov
- `/elastic/get-results-by-locality`
  - získanie výsledkov všetkých strán a kandidátov pre určitú lokalitu

2.6.2 Počítanie percent a parlamentných kresiel

Výpočet získaných kresiel sa vykonáva vo funkcii `calcualte_winning_parties_and_seats`.

```
def calcualte_winning_parties_and_seats(transformed_data):
    """
    Find parties having more than 5% (threshold) and count all votes
        for these parties.
    In case parties have less then threshold value, take all parties
    Calculate relative vote percentage from this set of parties and
        calculate result seats for each party
```

```
6       """
```

Algoritmus výpočtu: 1. Prepočítať poečt získaných hlasov pre všetky strany a získať tie ktoré majú nad 5%. 2. Počet republikové číslo (počet hlasov, potrebných pre získanie jedného mandátu, ráta s pomocou čísla 151) 3. Pomocou republikového čísla určiť na koľko kresiel má strana nárok a uchovať si počet po celočíselnom delení. 4. Ak neboli rozdané všetky kreslá, tak sa doplnia postupne stranám v poradí podľa zostatku po celočíselonom delení republikovým číslom.

2.6.3 Popis API

2.6.3.1 setup_elastic_votes_index_elastic_setup_elastic_vote_index_post

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.post('/elastic/setup-elastic-vote-index', headers =
   headers)

print(r.json())
```

`POST /elastic/setup-elastic-vote-index`

Setup Elastic Votes Index

Setup elastic search. Drop index if previously used. Create new index and variables mapping.

Example responses

200 Response

```
{
  "status": "string",
  "message": "string"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Message
400	Bad Request	Bad Request	Message
500	Internal Server Error	Internal Server Error	Message

This operation does not require authentication

2.6.3.2 synchronize_votes_ES_elastic_synchronize_votes_es_post

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
```

```
}

r = requests.post('/elastic/synchronize-votes-es', headers = headers)

print(r.json())
```

`POST /elastic/synchronize-votes-es`

Synchronize Votes Es

Batch synchronization of votes from Mongo DB to Elastic search 3 Node cluster. Shuld be called in specific intervals during election period.

Parameters

Name	In	Type	Required	Description
number	query	any	false	none

Example responses

200 Response

```
{
  "status": "string",
  "message": "string"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Message
400	Bad Request	Bad Request	Message
422	Unprocessable Entity	Validation Error	HTTPValidationError
500	Internal Server Error	Internal Server Error	Message

This operation does not require authentication

2.6.3.3 get__parties__results__elastic__get__parties__results__post

Code samples

```
import requests
headers = {
  'Content-Type': 'application/json',
  'Accept': 'application/json'
}

r = requests.post('/elastic/get-parties-results', headers = headers)

print(r.json())
```

`POST /elastic/get-parties-results`

Get Parties Results

Body parameter

```
{
  "party": "SME RODINA"
}
```

Parameters

Name	In	Type	Required	Description
body	body	StatisticsPerPartyRequest	true	none

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline
400	Bad Request	Bad Request	Message
422	Unprocessable Entity	Validation Error	HTTPValidationError
500	Internal Server Error	Internal Server Error	Message

Response Schema

This operation does not require authentication

2.6.3.4 get__parties__with__candidates__results__elastic__get__party__candidate__results__post

Code samples

```
import requests
headers = {
  'Content-Type': 'application/json',
  'Accept': 'application/json'
}

r = requests.post('/elastic/get-party-candidate-results', headers =
   headers)

print(r.json())
```

`POST /elastic/get-party-candidate-results`

Get Parties With Candidates Results

Body parameter

```
{
  "party": "SME RODINA"
}
```

Parameters

Name	In	Type	Required	Description
body	body	StatisticsPerPartyRequest	true	none

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline
400	Bad Request	Bad Request	Message
422	Unprocessable Entity	Validation Error	HTTPValidationError
500	Internal Server Error	Internal Server Error	Message

Response Schema

This operation does not require authentication

2.6.3.5 get_candidates_results_elastic_get_candidates_results_post

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.post('/elastic/get-candidates-results', headers =
   headers)

print(r.json())
```

`POST /elastic/get-candidates-results`

Get Candidates Results

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline
400	Bad Request	Bad Request	Message
500	Internal Server Error	Internal Server Error	Message

Response Schema

This operation does not require authentication

2.6.3.6 get__resilts__by__locality__mongo__elastic__get__results__by__locality__mongo__get

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('/elastic/get-results-by-locality-mongo', headers =
    headers)

print(r.json())
```

```
GET /elastic/get-results-by-locality-mongo
```

Get Resilts By Locality Mongo

Used to provide benchmark for ES vs Mongo aggregation queries

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

2.6.3.7 get__results__by__locality__elastic__get__results__by__locality__post

Code samples

```
import requests
headers = {
  'Content-Type': 'application/json',
  'Accept': 'application/json'
}

r = requests.post('/elastic/get-results-by-locality', headers =
    headers)

print(r.json())
```

```
POST /elastic/get-results-by-locality
```

Get Results By Locality

Body parameter

```
{
  "filter_by": "region_name",
  "filter_value": "Presovsky kraj",
}
```

Parameters

Name	In	Type	Required	Description
body	body	StatisticsPerLocalityRequest	true	none

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline
400	Bad Request	Bad Request	Message
422	Unprocessable Entity	Validation Error	HTTPValidationError
500	Internal Server Error	Internal Server Error	Message

Response Schema

This operation does not require authentication

2.6.3.8 get__elections__status__elastic__elections__status__get

Code samples

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('/elastic/elections-status', headers = headers)

print(r.json())
```

```
GET /elastic/elections-status
```

Get Elections Status

Example responses

200 Response

```
null
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline
400	Bad Request	Bad Request	Message
500	Internal Server Error	Internal Server Error	Message

Response Schema

This operation does not require authentication