Contents

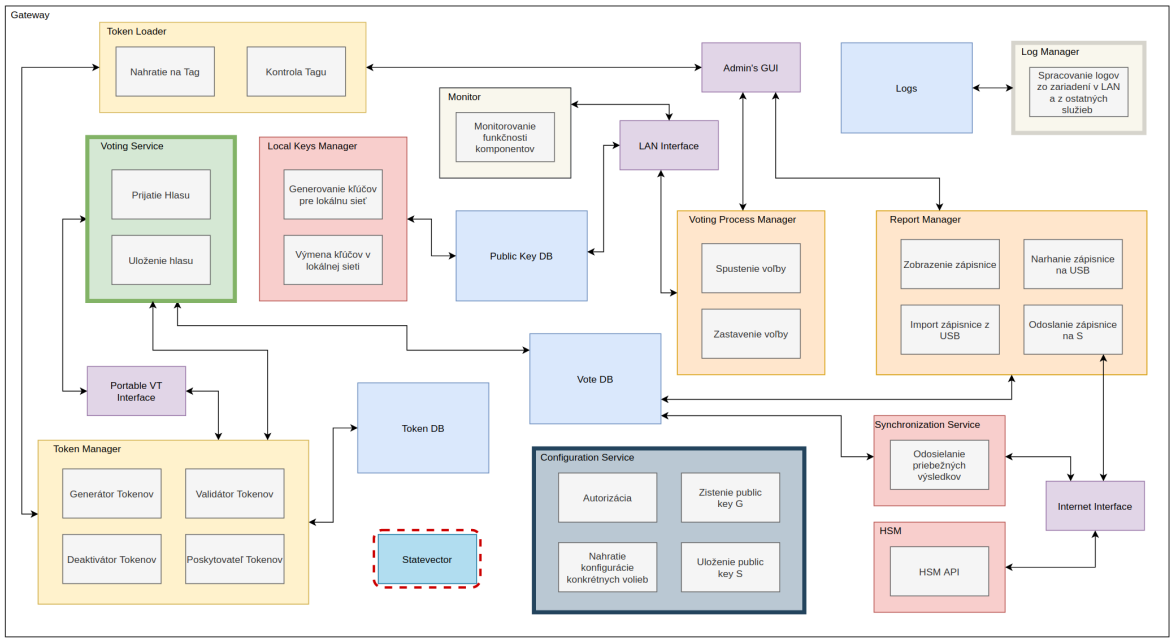
1. Gateway

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

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

get_state_election_state_election_get

Code samples

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

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

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

set_state_election_state_election_post

Code samples

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

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

```

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

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

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

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

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

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

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

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

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

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

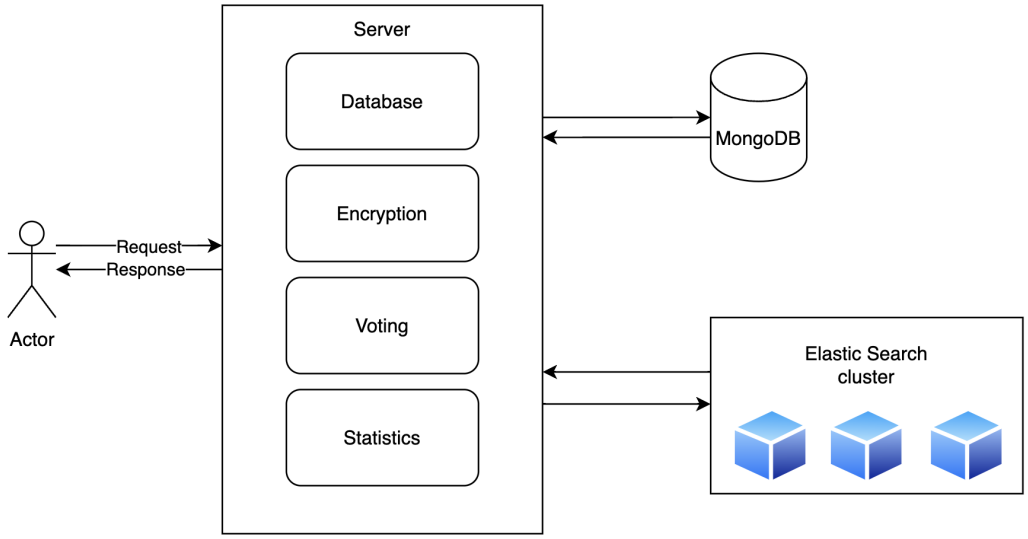
Properties

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.

```
1 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
```