

Contents

1. Gateway	1
1.1 Architektúra	2
1.2 Mikroslužby a ich smerovanie	2
1.3 State vector	2
1.3.1 Konfiguračný súbor	2
1.3.2 Popis API	3
1.4 Synchronization service	11
1.4.1 Štruktúra posielaných hlasov	11
1.4.2 Popis API	12
Token manager	15
1.4.3 Komunikácia s frontendom	15
1.4.4 Popis API	15
1.5 Token writer	25
1.5.1 Linuxový wrapper pre SL600-NFC zapisovačku	25
1.6.1 Registrácia volebného terminálu	26
1.6.2 Komunikácia medzi volebným terminálom	26
1.6.3 Popis API	26
1.7 Voting service	40
1.7.1 Popis API	40
2. Server	46
2.1 Architektúra	47
2.2 Inštalácia	47
2.2.1 Závislosti	47
2.2.2 Spustenie	47
2.2.3 Ako si nainštalovať skúšobné dáta a pripraviť Elastic Search cluster	47
2.2.4 Problém s Elastic search pamäťou	48
2.2.5 Testovanie vnútri dockeru	48
2.3 Databáza	48
2.3.1 Popis API	50
2.4 Generovanie hlasov	53
2.5 Hlasovanie	53
2.5.1 Validácia	53
2.5.2 Popis API	53
2.6 Výsledky a štatistiky	56
2.6.1 Dostupné koncové body:	56
2.6.2 Počítanie percent a parlamentných kresiel	56
2.6.3 Popis API	57

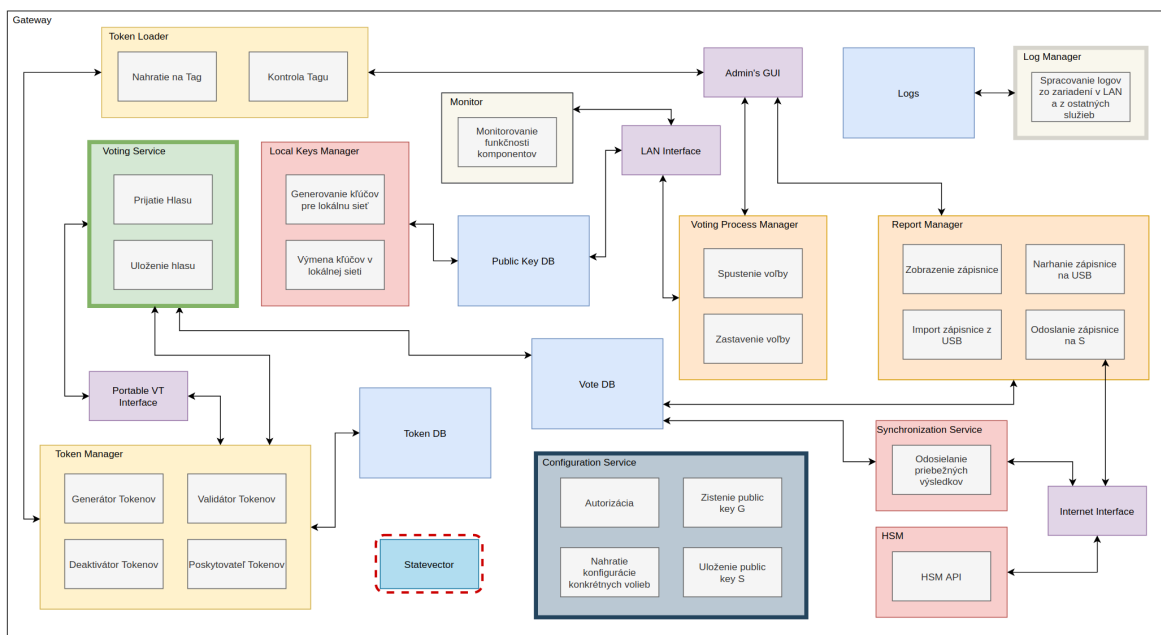
1. Gateway

Gateway je zariadenie nachádzajúce sa vo volebnej 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.

Gateway sa má nachádzať na chránenom mieste a prístupovať 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 súborov 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/
<i>config.json</i>	/statevector/config/config.json
<i>datamodels.yaml</i>	/statevector/config/datamodels.yaml

1.3 State vector

Služba zodpovedná za udržiavanie aktuálneho stavu gateway-u.

Udržiava tieto stavy: - **state_election** - stav voľieb - **state_write** - stav zapisovačky - **state_register_terminals** - stav registrácie terminálov - **office_id** - id volebnej miestnosti - **pin** - pin kód k GUI aplikácii na gateway-i - **server_key** - verejný kľúč servera - **server_address** - adresa servera

1.3.1 Konfiguračný súbor

Konfiguračný súbor obsahuje celú konfiguráciu voľieb 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
2 headers = {
    'Accept': 'application/json'
4 }

6 r = requests.get('/gateway/statevector/', headers = headers)
8 print(r.json())
```

GET /

Hello

Sample testing endpoint

Example responses

200 Response

```
{
2  "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

```
1 import requests
headers = {
3   '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

```
1 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

```
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

```
1 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

```

1 import requests
  headers = {
3     'Accept': 'application/json'
  }
5
  r = requests.get('/gateway/statevector/state_write', headers =
    headers)
7
  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

```

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

```

```

7 r = requests.post('/gateway/statevector/state_write', headers =
  headers)
9 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

```

1 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

```

1 import requests
  headers = {
3   'Accept': 'application/json'
  }
5
  r = requests.get('/gateway/statevector/state_register_terminals',
    headers = headers)
7
  print(r.json())

```

GET /state_register_terminals

State Register Terminals

Get terminals registration state string 0 or 1

Example responses

200 Response

```
1 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

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

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

9 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

```
1 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

```
1 import requests
  headers = {
3     'Accept': 'application/json'
  }
5
r = requests.get('/gateway/statevector/office_id', headers = headers)
7
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

```
1 import requests
  headers = {
3     'Accept': 'application/json'
  }
5
r = requests.get('/gateway/statevector/pin', headers = headers)
7
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

```
1 import requests
  headers = {
3     'Accept': 'application/json'
  }
5
r = requests.get('/gateway/statevector/server_key', headers = headers)
7
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

```
1 import requests
  headers = {
3     'Accept': 'application/json'
  }
5
  r = requests.get('/gateway/statevector/server_address', headers =
      headers)
7
  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

```
1 {
3     "detail": [
4         {
5             "loc": [
6                 "string"
7             ],
8             "msg": "string",
9             "type": "string"
10        }
11    ]
12 }
```

HTTPValidationError

Properties

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

1.3.2.12.2 ValidationError

```

1 {
2   "loc": [
3     "string"
4   ],
5   "msg": "string",
6   "type": "string"
7 }

```

ValidationError

Properties

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

1.4 Synchronization service

Služba je zodpovedná za synchronizáciu hlasov medzi gateway-om a serverom. Služba je implementovaná ako REST API v knižnici Fast API.

Služba pracuje z hlasmi v lokálnej Mongo databáze, ktoré boli vložené pomocou Voting service. Hlasy sa synchronizujú po dávkach (prednastavená hodnota je 10) a po zaširovaní sa posielajú pomocou HTTP požiadavky na endpoint servera, ktorý ich zvaliduje. Synchronizácia prebehne úspešne iba ak sú všetky hlasy v poriadku prijaté. Úspešne synchronizované hlasy označí ako {"synchronized": true}.

Synchronizácia prebieha na pozadí v intervale každú minútu (implementované pomocou Fast API Utils Repeated Tasks). Dá sa však spustiť aj manuálne pomocou endpointu `POST /api/synchronize`, ktorý je popísaný nižšie.

1.4.1 Štruktúra posielaných hlasov

Hlasy sú posielané v HTTP požiadavke, ktorú tvorí json s id volebnej miestnosti a hlasmi, ktoré sú zašifrované ako pole zašifrovaných hlasov pomocou knižnice *rsaelectie*, funkcie `encrypt_vote`.

```

1 {
2   "polling_place_id": 0,
3   "votes": [
4     {
5       "encrypted_message": "string",
6       "encrypted_object": "string"
7     }
8   ]
9 }

```

1.4.2 Popis API

root__get

Code samples

```
1 import requests
headers = {
3     'Accept': 'application/json'
}
5
r = requests.get('/gateway/synchronization-service-api/', headers =
    headers)
7
print(r.json())
```

GET /

Root

Simple hello message.

Example responses

200 Response

```
{
2     "status": "string",
    "message": "string"
4 }
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

synchronize__synchronize__post

Code samples

```
import requests
2 headers = {
    'Accept': 'application/json'
4 }

6 r = requests.post('/gateway/synchronization-service-api/synchronize',
    headers = headers)
8 print(r.json())
```

POST /synchronize

Synchronize

Try to send local votes to server and updates local status. If server response is different than 200, response has status 500 with error from server.

Example responses

200 Response

```
2 {
  "status": "string",
  "message": "string"
4 }
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

statistics_statistics_post

Code samples

```
import requests
2 headers = {
  'Accept': 'application/json'
4 }

6 r = requests.post('/gateway/synchronization-service-api/statistics',
  headers = headers)
8 print(r.json())
```

POST /statistics

Statistics

Provide statistics of votes in gateway database. Count of synchronized and unsynchronized votes.

Example responses

200 Response

```
2 {
  "status": "string",
  "last_synchronization": null,
4  "last_success_synchronization": null,
  "statistics": {
6    "all_count": 0,
    "synchronized_count": 0,
8    "unsynchronized_count": 0
  }
10 }
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

seed_seed_post

Code samples

```
import requests
2 headers = {
  'Accept': 'application/json'
4 }

6 r = requests.post('/gateway/synchronization-service-api/seed',
  headers = headers)

8 print(r.json())
```

POST /seed

Seed

Insert 10 unsynced dummy votes into gateway local gatabase.

Example responses

200 Response

```
{
2  "status": "string"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

test_encrypt_test_encrypt_get

Code samples

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

```

5 r = requests.get('/gateway/synchronization-service-api/test-encrypt',
7   headers = headers)
print(r.json())

```

GET /test-encrypt

Test Encrypt

Get a batch of encrypted votes.

Example responses

200 Response

```

{
2  "polling_place_id": 0,
   "votes": []
4 }

```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

Token manager

Služba je zodpovedná za generovanie, overovanie a deaktivovanie tokenov nahrávaných na NFC tagy. Služba rovnako ovláda a interaguje s Token writer-om , ktorý sa stará o samostné nahratie tokenu na NFC tag.

Token je generovaný pomocou uuid bez znakov -, napríklad 858c0eb798a8475dbcf67e29ddb4966e.

Deaktivovaný token je označený ako {"active": false}.

Aktivovaný a zapísaný token je označený ako {"active": true} a {"written": true}.

Token je považovaný ako platný iba ak je aktívny ({"active": true}).

1.4.3 Komunikácia s frontendom

Token manager komunikuje s frontendovou aplikáciou pomocou websocketov. Používateľa informuje o stave zapisovačky o úspešnom alebo neúspešnom zapísaní tokenu alebo o možnosti nahrávania ďalšieho tokenu. Vo websockete sa posiela udalosť `writer_status`, ktorý nadobúda hodnoty `off`, `idle`, `success`, `error`.

1.4.4 Popis API

`root__get`

Code samples

```

import requests
2 headers = {
    'Accept': 'application/json'
4 }

6 r = requests.get('/gateway/token-manager-api/', headers = headers)
8 print(r.json())

```

GET /

Root

Simple hello message.

Example responses

200 Response

```

{
2  "status": "string",
   "message": "string"
4 }

```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

activate_state_tokens_writer_activate_post

Code samples

```

import requests
2 headers = {
    'Accept': 'application/json'
4 }

6 r =
    requests.post('/gateway/token-manager-api/tokens/writer/activate',
        headers = headers)
8 print(r.json())

```

POST /tokens/writer/activate

Activate State

Activate NFC writer machine. After turning on, machine's LED will turn on and be able to write data to NFC tokens.

Example responses

200 Response

```
{
2  "status": "string",
  "message": "string"
4 }
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

deactivate_state_tokens_writer_deactivate_post

Code samples

```
import requests
2 headers = {
  'Accept': 'application/json'
4 }

6 r =
  requests.post('/gateway/token-manager-api/tokens/writer/deactivate',
    headers = headers)
8 print(r.json())
```

POST /tokens/writer/deactivate

Deactivate State

Deactivate NFC writer machine. Led on machine will turn off.

Example responses

200 Response

```
{
2  "status": "string",
  "message": "string"
4 }
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

delete_unwritten_tokens_writer_delete_post

Code samples

```
import requests
2 headers = {
    'Content-Type': 'application/json',
4    'Accept': 'application/json'
}
6
r = requests.post('/gateway/token-manager-api/tokens/writer/delete',
    headers = headers)
8
print(r.json())
```

POST /tokens/writer/delete

Delete Unwritten

Delete unwritten NFC tokens from database.

Body parameter

```
{
2  "event": "string"
}
```

Parameters

Name	In	Type	Required	Description
body	body	Body_delete_unwritten_tokens_writer_delete_post	true	none

Example responses

200 Response

```
1 {
2  "status": "string",
3  "message": "string"
}
```

Responses

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

Response Schema

This operation does not require authentication

update_written_tokens_writer_update_post

Code samples

```

import requests
2 headers = {
    'Content-Type': 'application/json',
4    'Accept': 'application/json'
}
6
r = requests.post('/gateway/token-manager-api/tokens/writer/update',
    headers = headers)
8
print(r.json())

```

POST /tokens/writer/update

Update Written

Update NFC token state from unwritten to written.

Body parameter

```

{
2  "token": "string"
}

```

Parameters

Name	In	Type	Required	Description
body	body	Body_update_written_tokens_writer_update_post	true	none

Example responses

200 Response

```

1 {
3  "status": "string",
  "message": "string"
}

```

Responses

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

Response Schema

This operation does not require authentication

create_token_tokens_create_post

Code samples

```

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

```

```

4 }
6 r = requests.post('/gateway/token-manager-api/tokens/create', headers
    = headers)
8 print(r.json())

```

POST /tokens/create

Create Token

Generates new token and returns it.

Example responses

200 Response

```

{
2  "status": "string",
4  "token": "string"
}

```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

validate__token__tokens__validate__post

Code samples

```

import requests
2 headers = {
    'Content-Type': 'application/json',
4    'Accept': 'application/json'
}
6
r = requests.post('/gateway/token-manager-api/tokens/validate',
    headers = headers)
8
print(r.json())

```

POST /tokens/validate

Validate Token

Validate if provided token is valid. If token is invalid returns empty response with status 403 else status 200.

Body parameter

```
{
```

```
2  "token": "string"
}
```

Parameters

Name	In	Type	Required	Description
body	body	Body_validate_token_tokens_validate_post	true	none

Example responses

200 Response

```
1 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

deactivate_token_tokens_deactivate_post

Code samples

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

7 r = requests.post('/gateway/token-manager-api/tokens/deactivate',
    headers = headers)
9 print(r.json())
```

POST /tokens/deactivate

Deactivate Token

Deactivate provided token. Change active status to false. If token is invalid returns empty response with status 403 else status 200.

Body parameter

```
{
2  "token": "string"
}
```

Parameters

Name	In	Type	Required	Description
body	body	Body_deactivate_token_tokens_deactivate_post	true	none

Example responses

200 Response

```
1 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

delete_token_tokens_delete_delete

Code samples

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

7 r = requests.delete('/gateway/token-manager-api/tokens/delete',
    headers = headers)

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

DELETE /tokens/delete

Delete Token

Delete provided token. If token is invalid returns empty response with status 403 else status 200.

Body parameter

```
{
2  "token": "string"
}
```

Parameters

Name	In	Type	Required	Description
body	body	Body_delete_token_tokens_delete_delete	true	none

Example responses

200 Response

```
1 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

Schemas

1.4.4.10.1 Body_deactivate_token_tokens_deactivate_post

```
1 {  
3   "token": "string"  
}
```

Body_deactivate_token_tokens_deactivate_post

Properties

Name	Type	Required	Restrictions	Description
token	string	true	none	none

1.4.4.10.2 Body_delete_token_tokens_delete_delete

```
1 {  
3   "token": "string"  
}
```

Body_delete_token_tokens_delete_delete

Properties

Name	Type	Required	Restrictions	Description
token	string	true	none	none

1.4.4.10.3 Body_delete_unwritten_tokens_writer_delete_post

```
1 {  
3   "event": "string"  
}
```

Body_delete_unwritten_tokens_writer_delete_post

Properties

Name	Type	Required	Restrictions	Description
event	string	true	none	none

1.4.4.10.4 Body_update_written_tokens_writer_update_post

```

1 {
  "token": "string"
3 }
```

Body_update_written_tokens_writer_update_post

Properties

Name	Type	Required	Restrictions	Description
token	string	true	none	none

1.4.4.10.5 Body_validate_token_tokens_validate_post

```

1 {
  "token": "string"
3 }
```

Body_validate_token_tokens_validate_post

Properties

Name	Type	Required	Restrictions	Description
token	string	true	none	none

1.4.4.11 HTTPValidationError

```

1 {
  "detail": [
3   {
    "loc": [
5     "string"
    ],
7     "msg": "string",
    "type": "string"
9   }
  ]
11 }
```

HTTPValidationError

Properties

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

1.4.4.11.1 ValidationError

```
1 {  
  "loc": [  
3    "string"  
  ],  
5  "msg": "string",  
  "type": "string"  
7 }
```

ValidationError

Properties

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

1.5 Token writer

Služba zodpovedná za zapisovanie tokenu na Mifare 1k tag.

Zapisovanie funguje nasledovne: - zapisovačka čaká, pokým sa Mifare tag nachádza v dosahu zapisovačky - prečíta zo štvrtého bloku zapísané 128-bitové číslo - pošle požiadavku na token manager na deaktiváciu prečítaného čísla - pošle požiadavku na token manager na vygenerovanie nového 128-bitového čísla - zapíše hodnotu na tag - verifikuje zapísanú hodnotu jej prečítaním z tagu - po úspešnej verifikácii pošle požiadavku na token manager o aktiváciu tokenu

Pre zamedzenie zapísania dvoch tokenov na jedno priloženie je vytvorený cooldown 30 sekúnd, ktorý v tejto dobe neumožní zapísať na rovnaký Mifare tag znova ďalší token.

Služba používa linuxový wrapper popísaný nižšie.

1.5.1 Linuxový wrapper pre SL600-NFC zapisovačku

Implementované funkcionality: - Vypnutie LED svetla - Zapnutie LED svetla - Čítanie z Mifare 1k tagu (štvrtý blok) - Zápis na Mifare 1k tagu (štvrtý blok) - Validácia zápisu

Ukážkový kód v knižnici zapíše náhodné 128-bitové číslo do štvrtého bloku Mifare 1k tagu, potom ho prečíta z neho a tým zvaliduje zápis. Ak všetko prebehne úspešne, program skončí.

Pre vývoj bez použitia dockera je potrebné nainštalovať závislosť pyusb:

```
1 pip3 install pyusb
```

Program musí bežať so sudo oprávneniami

Funguje iba na Linuxe (vo WSL 2 nekomunikuje s čítačkou). Testované na Ubuntu 20.04 a Ubuntu 22.04. ## 1.6 Voting process manager

Hlavná služba na gateway-i zodpovedná za spustenie a zastavenie volieb, registráciu volebných terminálov, poskytuje informáciu o stave pripojených terminálov a udalosti o spustení a zastavení volieb. Rovnako zabezpečuje generovanie zápisnice a odoslanie zápisnice na server.

1.6.1 Registrácia volebného terminálu

Pri spustení volebného terminálu sa terminál dopytuje na endpoint `/register-vt` kedy sa pri spustenej registrácii vymení verejný kľúč gateway-a, aby mohla prebiehať šifrovaná komunikácia medzi volebným terminálom a gateway-om. Ak registrácia nie je spustená vráti sa status 400.

1.6.2 Komunikácia medzi volebným terminálom

Táto služba komunikuje so všetkými registrovanými volebnými terminálmi pomocou websocketov. Vo websockete sa posíla udalosť `actual_state`, ktorý obsahuje aktuálny stav volieb volebným terminálom. Rovnako aj volebné terminály notifikujú gateway o ich aktuálnom stave udalosťou `vt_status`.

1.6.3 Popis API

root__get

Code samples

```
import requests
2 headers = {
    'Accept': 'application/json'
4 }

6 r = requests.get('/gateway/voting-process-manager-api/', headers =
    headers)

8 print(r.json())
```

GET /

Root

Simple hello message.

Example responses

200 Response

```
{
2  "status": "string",
  "message": "string"
4 }
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

election_config_election_config_get

Code samples

```

import requests
2 headers = {
    'Accept': 'application/json'
4 }

6 r =
    requests.get('/gateway/voting-process-manager-api/election-config',
        headers = headers)

8 print(r.json())

```

GET /election-config

Election Config

Returns necessary config fields for gateway from config.

Example responses

200 Response

```

{
2  "status": "string",
   "texts": {}
4 }

```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

terminals_status_terminals_status_get

Code samples

```

import requests
2 headers = {
    'Accept': 'application/json',
4    'Authorization': 'Bearer {access-token}'
}

6 r =
    requests.get('/gateway/voting-process-manager-api/terminals-status',
        headers = headers)

8 print(r.json())

```

GET /terminals-status

Terminals Status

Returns necessary status information of connected voting terminals.

Example responses

200 Response

```
{
2  "status": "string",
   "registration_status": false,
4  "terminals": [
6  ]
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

To perform this operation, you must be authenticated by means of one of the following methods:
OAuth2PasswordBearer

login_for_access_token_token_post

Code samples

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

7 r = requests.post('/gateway/voting-process-manager-api/token',
   headers = headers)
9 print(r.json())
```

POST /token

Login For Access Token

Log in user using username and password.

Body parameter

```
grant_type: string
2 username: string
  password: string
4 scope: ""
  client_id: string
6 client_secret: string
```

Parameters

Name	In	Type	Required	Description
body	body	Body_login_for_access_token_token_post	true	none

Example responses

200 Response

```

2 {
  "access_token": "string",
  "token_type": "string"
4 }
```

Responses

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

This operation does not require authentication

current_user_current_user__get

Code samples

```

import requests
2 headers = {
  'Accept': 'application/json',
4  'Authorization': 'Bearer {access-token}'
}
6
r = requests.get('/gateway/voting-process-manager-api/current-user/',
  headers = headers)
8
print(r.json())
```

GET /current-user/

Current User

Example responses

200 Response

```

2 {
  "username": "string",
  "disabled": true
4 }
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	User

To perform this operation, you must be authenticated by means of one of the following methods:
OAuth2PasswordBearer

start_voting_process_start_post

Code samples

```
import requests
2 headers = {
    'Accept': 'application/json',
4    'Authorization': 'Bearer {access-token}'
}
6
r = requests.post('/gateway/voting-process-manager-api/start',
    headers = headers)
8
print(r.json())
```

POST /start

Start Voting Process

Starts elections and notify all voting terminals.

Example responses

200 Response

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

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

To perform this operation, you must be authenticated by means of one of the following methods:
OAuth2PasswordBearer

end_voting_process_end_post

Code samples

```
1 import requests
headers = {
3    'Accept': 'application/json',
    'Authorization': 'Bearer {access-token}'
```

```

5 }

7 r = requests.post('/gateway/voting-process-manager-api/end', headers
  = headers)

9 print(r.json())

```

POST /end

End Voting Process

Stops elections and notify all voting terminals.

Example responses

200 Response

```

2 {
  "status": "string",
}

```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

To perform this operation, you must be authenticated by means of one of the following methods:
OAuth2PasswordBearer

register__vt__register__vt__post

Code samples

```

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

7 r = requests.post('/gateway/voting-process-manager-api/register-vt',
  headers = headers)

9 print(r.json())

```

POST /register-vt

Register Vt

Register a voting terminal. Returns status 400 if registration is disabled else return status 200 with id and public key.

Body parameter

```
{
```

```
2  "public_key": "string"
}
```

Parameters

Name	In	Type	Required	Description
body	body	Body_register_vt_register_vt_post	true	none

Example responses

200 Response

```
1 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

gateway_events_gateway_elections_events_get

Code samples

```
1 import requests
headers = {
3     'Accept': 'application/json',
    'Authorization': 'Bearer {access-token}'
5 }

7 r =
    requests.get('/gateway/voting-process-manager-api/gateway-elections-events',
        headers = headers)
9 print(r.json())
```

GET /gateway-elections-events

Gateway Events

Get all elections events of start and end of elections.

Example responses

200 Response

```
{
2  "status": "success",
  "events": []
4 }
```


Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

To perform this operation, you must be authenticated by means of one of the following methods:
OAuth2PasswordBearer

get_first_start_gateway_elections_events_first_start_get

Code samples

```
import requests
2 headers = {
    'Accept': 'application/json',
4    'Authorization': 'Bearer {access-token}'
}
6
r =
    requests.get('/gateway/voting-process-manager-api/gateway-elections-events/first-start',
    headers = headers)
8
print(r.json())
```

GET /gateway-elections-events/first-start

Get First Start

Get first start of elections.

Example responses

200 Response

```
{
2   "status": "string",
   "first_start": {}
4 }
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

To perform this operation, you must be authenticated by means of one of the following methods:
OAuth2PasswordBearer

get_last_end_gateway_elections_events_last_end_get

Code samples

```

import requests
2 headers = {
    'Accept': 'application/json',
4    'Authorization': 'Bearer {access-token}'
}
6
r =
    requests.get('/gateway/voting-process-manager-api/gateway-elections-events/last-end',
    headers = headers)
8
print(r.json())

```

GET /gateway-elections-events/last-end

Get Last End

Get last end of elections.

Example responses

200 Response

```

{
2  "status": "string",
   "last_end": {}
4 }

```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

To perform this operation, you must be authenticated by means of one of the following methods:
OAuth2PasswordBearer

generate_commission_paper_commission_paper_generate_post

Code samples

```

import requests
2 headers = {
    'Content-Type': 'application/json',
4    'Accept': 'application/json'
}
6
r =
    requests.post('/gateway/voting-process-manager-api/commission-paper/generate',
    headers = headers)
8
print(r.json())

```

POST /commission-paper/generate

Generate Commission Paper

Generate commission paper in pdf format encoded in base64 and store it into database.

Body parameter

```
{
  "polling_place_id": 0,
  "participated_members": [
    {
      "name": "Erik Malina",
      "agree": true
    },
    {
      "name": "Ferko Jablko",
      "agree": false
    },
    {
      "name": "Adam Jahoda",
      "agree": true
    }
  ],
  "president": {
    "name": "Samo Kiwi",
    "agree": true
  }
}
```

Parameters

Name	In	Type	Required	Description
body	body	CommissionPaper	true	none

Example responses

200 Response

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

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_commission_paper_commission_paper_get

Code samples

```
import requests
2 headers = {
    'Accept': 'application/json'
4 }

6 r =
    requests.get('/gateway/voting-process-manager-api/commission-paper',
        headers = headers)

8 print(r.json())
```

GET /commission-paper

Get Commission Paper

Get commission paper from database encoded in base64.

Example responses

200 Response

```
{
2  "status": "string",
   "data": "string"
4 }
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

send_commission_paper_commission_paper_send_post

Code samples

```
import requests
2 headers = {
    'Accept': 'application/json'
4 }

6 r =
    requests.post('/gateway/voting-process-manager-api/commission-paper/send',
        headers = headers)

8 print(r.json())
```

POST /commission-paper/send

Send Commission Paper

Send commission paper to server.

Example responses

200 Response

```
2 {  
  "status": "string",  
  "message": "string"  
4 }
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

Schemas

1.6.3.15.1 Body_login_for_access_token_token_post

```
2 {  
  "grant_type": "string",  
  "username": "string",  
4  "password": "string",  
  "scope": "",  
6  "client_id": "string",  
  "client_secret": "string"  
8 }
```

Body_login_for_access_token_token_post

Properties

Name	Type	Required	Restrictions	Description
grant_type	string	false	none	none
username	string	true	none	none
password	string	true	none	none
scope	string	false	none	none
client_id	string	false	none	none
client_secret	string	false	none	none

1.6.3.15.2 Body_register_vt_register_vt_post

```
2 {  
  "public_key": "string"  
}
```

Body_register_vt_register_vt_post

Properties

Name	Type	Required	Restrictions	Description
public_key	string	true	none	none

1.6.3.15.3 CommissionPaper

```
1 {
2   "polling_place_id": 0,
3   "participated_members": [
4     {
5       "name": "Erik Malina",
6       "agree": true
7     },
8     {
9       "name": "Ferko Jablko",
10      "agree": false
11     },
12     {
13       "name": "Adam Jahoda",
14       "agree": true
15     }
16   ],
17   "president": {
18     "name": "Samo Kiwi",
19     "agree": true
20   }
21 }
```

CommissionPaper

Properties

Name	Type	Required	Restrictions	Description
polling_place_id	integer	true	none	none
participated_members	[Member]	false	none	none
president	Member	true	none	none

1.6.3.15.4 HTTPValidationError

```
1 {
2   "detail": [
3     {
4       "loc": [
5         "string"
6       ],
7       "msg": "string",
8       "type": "string"
9     }
10  ]
11 }
```

```
}
```

HTTPValidationError

Properties

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

1.6.3.15.5 Member

```
1 {  
2   "name": "string",  
3   "agree": true  
4 }
```

Member

Properties

Name	Type	Required	Restrictions	Description
name	string	true	none	none
agree	boolean	true	none	none

1.6.3.15.6 Token

```
1 {  
2   "access_token": "string",  
3   "token_type": "string"  
4 }
```

Token

Properties

Name	Type	Required	Restrictions	Description
access_token	string	true	none	none
token_type	string	true	none	none

1.6.3.15.7 User

```
1 {  
2   "username": "string",  
3   "disabled": true  
4 }
```

User

Properties

Name	Type	Required	Restrictions	Description
username	string	true	none	none
disabled	boolean	false	none	none

1.6.3.15.8 ValidationError

```

2 {
4   "loc": [
6     "string",
7     "msg": "string",
8     "type": "string"
9   ]
10 }

```

ValidationError

Properties

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

1.7 Voting service

Služba zodpovedná za overovanie prichádzajúceho tokenu a za prijímanie hlasu z volebného terminálu.

1.7.1 Popis API

hello__get

Code samples

```

1 import requests
2 headers = {
3     'Accept': 'application/json'
4 }
5
6 r = requests.get('/gateway/voting-service-api/', headers = headers)
7
8 print(r.json())

```

GET /

Hello

Sample testing endpoint

Example responses

200 Response

```

null

```


Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

Response Schema

This operation does not require authentication

vote_api_vote_post

Code samples

```

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

7 r = requests.post('/gateway/voting-service-api/api/vote', headers =
    headers)

9 print(r.json())

```

POST /api/vote

Vote

Receives vote with valid token, validates the token, stores the vote and invalidates the token.

Returns: 200: Vote was successfully stored 403: Token is invalid 409: The election is not running at the moment 422: Invalid request body

Body parameter

```

{
2   "voting_terminal_id": "string",
  "payload": {
4     "encrypted_message": "string",
    "encrypted_object": "string"
6   }
}

```

Parameters

Name	In	Type	Required	Description
body	body	Body_vote_api_vote_post	true	none

Example responses

200 Response

```

1 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

token_validity_api_token_validity_post

Code samples

```

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

7 r = requests.post('/gateway/voting-service-api/api/token-validity',
    headers = headers)

9 print(r.json())

```

POST /api/token-validity

Token Validity

Checks if the provided token is valid.

Body parameter

```

{
2  "voting_terminal_id": "string",
  "payload": {
4    "encrypted_message": "string",
    "encrypted_object": "string"
6  }
}

```

Parameters

Name	In	Type	Required	Description
body	body	Body_token_validity_api_token_validity_post	true	none

Example responses

200 Response

```
1 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

Schemas

1.7.1.4.1 Body_token_validity_api_token_validity_post

```

1 {
2   "voting_terminal_id": "string",
3   "payload": {
4     "encrypted_message": "string",
5     "encrypted_object": "string"
6   }
7 }
```

Body_token_validity_api_token_validity_post

Properties

Name	Type	Required	Restrictions	Description
voting_terminal_id	string	true	none	none
payload	VoteEncrypted	true	none	Attributes—— - encrypted_message: str AES encrypted mes- sage.encrypted_object: str RSA encrypted AES key and other data.

1.7.1.4.2 Body_vote_api_vote_post

```

1 {
2   "voting_terminal_id": "string",
3   "payload": {
4     "encrypted_message": "string",
5     "encrypted_object": "string"
6   }
7 }
```

Body_vote_api_vote_post

Properties

Name	Type	Required	Restrictions	Description
voting_terminal_id	string	true	none	none
payload	VoteEncrypted	true	none	Attributes—— - encrypted_message: str AES encrypted mes- sage.encrypted_object: str RSA encrypted AES key and other data.

1.7.1.4.3 HTTPValidationError

```

1 {
2   "detail": [
3     {
4       "loc": [
5         "string"
6       ],
7       "msg": "string",
8       "type": "string"
9     }
10  ]
11 }
```

HTTPValidationError

Properties

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

1.7.1.4.4 ValidationError

```

1 {
2   "loc": [
3     "string"
4   ],
5   "msg": "string",
6   "type": "string"
7 }
```

ValidationError

Properties

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

Name	Type	Required	Restrictions	Description
msg	string	true	none	none
type	string	true	none	none

1.7.1.4.5 VoteEncrypted

```

1 {
  "encrypted_message": "string",
3  "encrypted_object": "string"
}
```

VoteEncrypted

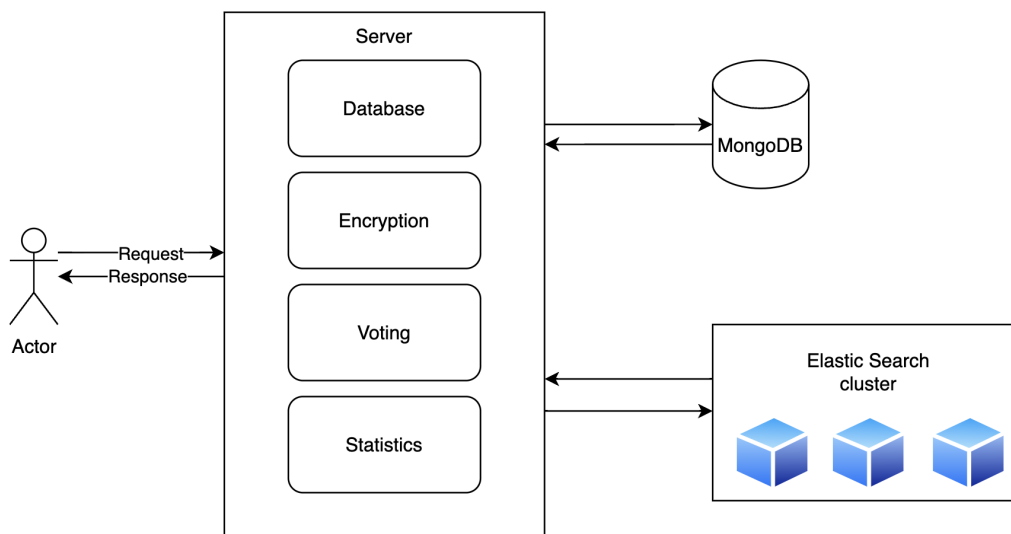
Properties

Name	Type	Required	Restrictions	Description
encrypted_message	string	true	none	none
encrypted_object	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 špecifikuje koľko hlasov bolo spracovaných. Uložené hlasy sa priebežne indexujú do technológie 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 premennú `vm.max_map_count` v kerneli dockeru na najmenej 262144.

V závislosti od operačného systému použite jeden z nasledovných príkazov:

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

4 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:

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

Dostupné príznaky: - `-p` - prepred 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:

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

2.3 Databáza

Server používa na ukladanie dát dokumentovú databázu MongoDB. Aj keď je do MongoDB vkladat dáta s rôznymi atribútmi, používame modely jednotlivých 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:

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

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

```
1 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 zindexovaný do Elastic Searchu.

Štruktúra politickej strany:

```
class Party(BaseModel):
2     id: int = Field(..., alias="_id")
    party_number: int
4     name: str
    official_abbr: str
6     abbr: str
    image: str
8     image_bytes: str
    color: str
10    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 kandidátov, ktorý sú reprezentovaný vlastným modelom.

Štruktúra volebnej miestnosti:

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

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

Štruktúra kandidáta:

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

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

Štruktúra kľúčového páru:

```

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

```

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

2.3.1 Popis API

schema_database_schema_get

Code samples

```

1 import requests
headers = {
3     'Accept': 'application/json'
}
5
r = requests.get('/database/schema', headers = headers)
7
print(r.json())

```

GET /database/schema

Schema

Get all collections from database

Example responses

200 Response

```

{
2     "collections": []
}

```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Collections

This operation does not require authentication

import_data_database_import_data_post

Code samples

```

1 import requests
headers = {
3     '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

```

2 {
  "status": "string",
  "message": "string"
4 }

```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Message

This operation does not require authentication

seed_data_database_seed_data_post

Code samples

```

import requests
2 headers = {
  'Accept': 'application/json'
4 }

6 r = requests.post('/database/seed-data', params={
  'number_of_votes': '0'
8 }, headers = headers)
10 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

```
{
```

```

2  "status": "string",
   "message": "string"
4 }

```

Responses

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

This operation does not require authentication

seed__votes__database__seed__votes__post

Code samples

```

import requests
2 headers = {
  'Accept': 'application/json'
4 }

6 r = requests.post('/database/seed-votes', params={
  'number_of_votes': '0'
8 }, headers = headers)
10 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

```

{
2  "status": "string",
   "message": "string"
4 }

```

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 nasledovná:

```
class VoteEncrypted(BaseModel):
2     encrypted_message: str
    encrypted_object: str
4
class VotesEncrypted(BaseModel):
6     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 duplicitná kombinácia tokenu a *id* volebnej miestnosti
- v prichádzajúcom 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

vote_elections_vote_post

Code samples

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

```

5 }
7 r = requests.post('/elections/vote', headers = headers)
9 print(r.json())

```

POST /elections/vote

Vote

Process candidate's vote

Body parameter

```

{
2  "polling_place_id": 0,
  "votes": [
4    {
      "encrypted_message":
        "36AMNvcpAWdHAXKCSWexgyjxrt7xeWwh0f+oUMBqip/C051...",
6      "encrypted_object":
        "1b5B/LAg2/38mot9jYzRpa906YwrXDilpspPrGrnTKKYUXS8..."
      }
8  ]
}

```

Parameters

Name	In	Type	Required	Description
body	body	VotesEncrypted	true	none

Example responses

200 Response

```

1 {
  "status": "string",
3  "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

get_voting_data_elections_voting_data_get

Code samples

```

import requests
2 headers = {
    'Accept': 'application/json'
4 }

6 r = requests.get('/elections/voting-data', headers = headers)
8 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

```

{
2  "polling_places": [],
   "parties": [],
4  "key_pairs": [],
   "texts": {
6    "elections_name_short": {
       "sk": "string",
8     "en": "string"
    },
10   "elections_name_long": {
       "sk": "string",
12    "en": "string"
    },
14   "election_date": {
       "sk": "string",
16    "en": "string"
    }
18  }
}

```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	VotingData

This operation does not require authentication

`get_zapisnica_elections_zapisnica_get`

Code samples

```

1 import requests
  headers = {
3   'Accept': 'application/json'

```

```

}
5 r = requests.get('/elections/zapisnica', headers = headers)
7 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 rátaajú 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`.

```

2 def calcualte_winning_parties_and_seats(transformed_data):
    """
    Find parties having more than 5% (threshold) and count all votes
    for these parties.
4    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

```


Algoritmus výpočtu: 1. Prepočítať počet 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 rozdane všetky kreslá, tak sa doplnia postupne stranám v poradí podľa zostatku po celočíselnom delení republikovým číslom.

2.6.3 Popis API

setup_elastic_votes_index_elastic_setup_elastic_vote_index_post

Code samples

```
import requests
2 headers = {
    'Accept': 'application/json'
4 }

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

8 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

```
{
2  "status": "string",
   "message": "string"
4 }
```

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

synchronize_votes_ES_elastic_synchronize_votes_es_post

Code samples

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

```

4 }
6 r = requests.post('/elastic/synchronize-votes-es', headers = headers)
8 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

```

{
2  "status": "string",
   "message": "string"
4 }

```

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

get_parties_results_elastic_get_parties_results_post

Code samples

```

import requests
2 headers = {
   'Content-Type': 'application/json',
4   'Accept': 'application/json'
}
6
r = requests.post('/elastic/get-parties-results', headers = headers)
8
print(r.json())

```

POST /elastic/get-parties-results

Get Parties Results

Body parameter

```
1 {  
2   "party": "SME RODINA"  
}
```

Parameters

Name	In	Type	Required	Description
body	body	StatisticsPerPartyRequest	true	none

Example responses

200 Response

```
1 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

get_parties_with_candidates_results_elastic_get_party_candidate_results_post

Code samples

```
1 import requests  
headers = {  
3   'Content-Type': 'application/json',  
   'Accept': 'application/json'  
5 }  
  
7 r = requests.post('/elastic/get-party-candidate-results', headers =  
   headers)  
  
9 print(r.json())
```

POST /elastic/get-party-candidate-results

Get Parties With Candidates Results

Body parameter

```
1 {  
2   "party": "SME RODINA"  
}
```

Parameters

Name	In	Type	Required	Description
body	body	StatisticsPerPartyRequest	true	none

Example responses

200 Response

```
1 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

get_candidates_results_elastic_get_candidates_results_post

Code samples

```
1 import requests
  headers = {
3     'Accept': 'application/json'
  }
5
  r = requests.post('/elastic/get-candidates-results', headers =
    headers)
7
  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

get_resilts_by_locality_mongo_elastic_get_results_by_locality_mongo_get

Code samples

```
1 import requests
  headers = {
3     'Accept': 'application/json'
  }
5
  r = requests.get('/elastic/get-results-by-locality-mongo', headers =
      headers)
7
  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

get_results_by_locality_elastic_get_results_by_locality_post

Code samples

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

7  r = requests.post('/elastic/get-results-by-locality', headers =
      headers)
9  print(r.json())
```

POST /elastic/get-results-by-locality

Get Results By Locality

Body parameter

```
2 {
  "filter_by": "region_name",
  "filter_value": "Presovsky kraj",
4 }
```

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

get_elections_status_elastic_elections_status_get

Code samples

```
1 import requests
  headers = {
3   'Accept': 'application/json'
  }
5
  r = requests.get('/elastic/elections-status', headers = headers)
7
  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