Dokumentacji API - E-Wybory

ZPI 2024

Członkowie:  
Jan Jankowski  
Krzysztof Saar  
Michał Starba  
Krzysztof Wróblewski

1. Informacje:
   1. Adnotacja [Required] – wartość danej własności klasy nie może być NULL
   2. Adnotacja [Authorize] – użytkownik mus być zalogowany aby uzyskać dostęp do endpointu API, jego autentykacja zostaje zapewniona tokenem JWT z odpowiedmin claim’em do uprawnień
   3. Każdy endpoint zwraca obiekt HttpResponse, tylko content się rózni w zależności od przetworzenia a serwerze
   4. Parametry:
      1. {parametr} – parametr w ścieżce np. do zapytać GET
      2. NazwaViewModel – głównie BODY żądania http
   5. Odpowiedzi nieujęte w specyfikacji:
      1. BadRequest – w zależności od okna czasowego w którym można modyfikować dane o wyborach, może zostać zwrócony komunikat w rodzaju „Cannot modify election structure due to time window closure”
      2. BadRequest – błąd składni zapytania / walidacja modelu
      3. Conflict – kolejność dodawania rekordów do bazy:
      4. InternalServerError – na serwerze doszło do błędu przetwarzania np. nieobsłużony wyjątek
   6. Kolejność dodawania encji:

* Lokalizacje (np. gminy, powiaty)
* Typy wyborów
* Wybory
* Obwody //problem z diagramem i ograniczeniami kluczy głównych
* Okręgi najpierw Obwody potem okregi
* Grupy uprawnień
* Typy użytkowników
* Użytkownicy / Ludzie
* Komitety (partie)
* Kandydaci / Ludzie

1. Struktura w projekcie:
   1. ViewModel – E-wybory.Client\ViewModels
   2. Pozostałe - E-wybory.Application
2. Struktura tokenu JWT:
3. Do wyjaśnienia:
   1. Jakie role będą w aplikacji – trzeba wyspecyfikować do endpointów

Obraz zawierający tekst, zrzut ekranu, Jaskrawoniebieski, Czcionka

Opis wygenerowany automatycznie

W domyśle, każdy zalogowny użytkownik otrzymuje Uprawnienie – Wyborca

* 1. Dla tabeli ElectionUsers – potrzeba kolumny userSecret – długość jak w passwordhash
  2. Konwencja nazewnictwa ViewModel’i oraz ich tworzenie (Mapster)
  3. GUID vs id z bazy – id z bazy
  4. Usuwanie użytkowników, czy ich wyrejestrowywanie?
  5. Przetrzymywanie w tokenie informacji na temat bieżące claimu uprawnień, czy wszystkich? – bierzący
  6. Panel zarządzania użytkownikiem / resetowanie hasła / zmiana lub dodanie uprawnień / 2fa?
  7. Dla danych historycznych frekwencji – wielokrotne powielanie okręgów / obwodów (problem z niestałością przypisanych numerów)
  8. Jednoczesne przeprowadzanie wielu wyborów? – dodatkowy claim z ElectionID w JWT?
  9. Tabela person i duplikaty tożsamości, pobranie danych do wypelnienia za pomocą istniejącego rekordu w Person?
  10. Tworzenie: Required na Id-kach?
  11. Okręgi:
      1. Brak pola headquoaters - czy chodzi o główny District
  12. Obwody:
      1. Problem z ViewModelami – różne view modele / DTO / jeden wspólny view model ?
  13. Filtrowanie:
      1. Stopniowe zawężanie wyników w trakcie wypelniania
      2. Czy po dopiero po wypełnieniu wszystkich, niektórych pól?
      3. Podejście hybrydowe np. województwa i coś się pokazuje, i dalej lecimy co zmiane lokalizacji

1. Potencjalne modyfikacje w bazie danych:
   1. Kolumna userSecret w ElectionUsers
   2. Wydzielenie encji Headquarters dla zarządzania adresami i lokalizacjami
   3. Powiązania dla danych historycznych (District -> Voters -> Election) tyle razy ILE TRZBEA
   4. Numeracja obwodów / okręgów a dane historyczne
   5. Powiązania obowiązkowe na linii Voters -> District
   6. Członkowie komisji a dodawanie do obwodu
2. Zakresy operacji dla grup uprawnień (UserTypesGroups)
   1. Administratorzy
   2. Komisja wyborcza
   3. Praocwnicy PKW
   4. Urzędnicy wyborczy
   5. Operatorzy lokali wyborczych
   6. Wyborcy

Dokumentacja:

* + - 1. Zarządzanie użytkownikami

1. Logowanie (POST)

Endpoint: /api/auth/login

Parametry wejściowe UserLoginViewModel :

public class LoginViewModel

{

[Requierd]  
public string Email { get; set; }

[Requierd]  
public string Password { get; set; }

}

Return: JWT token (string, nullable) | Unauthorized

1. Rejestracja (POST)

Endpoint: /api/auth/register

Parametry: RegisterViewModel:

public class RegisterViewModel

{

[Required]

public string FirstName { get; set; }

[Required]

public string LastName { get; set; }

[Required]

public string Pesel { get; set;

[Required]

public DateTime DateOfBirth { get; set; }

[Required]

public string Email { get; set; }

[Required]

public string PhoneNumber { get; set; }

[Required]

public string Password { get; set; }

[Required]

public string ConfirmPassword { get; set; }

}

Returns: Created | Conflict

1. Wylogowanie (POST) [Authorized]  
   Endpoint: /api/auth/logout   
     
   Parametry: Brak  
     
   Returns: Ok | NotFound
2. Pobranie informacji o zalogowanym użytkowniku (GET) [Authorized]  
   Endpoint: /api/users/userinfo

Parametry : Brak

Returns UserInfoViewModel | NotFound

public class UserInfoViewModel

{

[Requierd]  
public string Name { get; set; }

[Requierd]  
public string Surname { get; set; }

[Required]

Public string Username { get; set; }

[Requierd]  
public CurrentUserTypeName { get; set; }

[Requiered]

public List<UserTypeViewModel> AvailableUserTypes { get; set; }

}

public class UserInfoViewModel

{

[Required]

public int IdUserType { get; set; }

[Required]

public string UserTypeName { get; set; }

}

1. Manualne odnowienie tokenu (POST) [Authorized]

Endpoint: /api/auth/renew-token

Parametry: UserViewModel

Returns: JWT token (string, nullable) | Unauthorized(Token previously expired)

1. Dodaj autentykacje 2FA (POST) [Authorized]

Endpoint: /api/auth/add-2fa

Parametry: brak

Returns: 2FAResponse

public class 2FAResponse

{

[Requierd]  
public string QRCodeUri { get; }

[Requierd]  
public string Secret { get; }

}

1. Potwierdz autentykacje 2FA (POST)

Endpoint: /api/auth/verify-2fa

Parametry: 2FAVerifyRequest

public class 2FAVerifyRequest

{

[Requierd]  
public string Code { get; set; }

}

Response: JWT Token (claim: 2faVerified: true) | Unauthorized(Bad 2fa code)

* + - 1. Zarządzanie okręgami

1. Pobierz okręgi wyborcze (GET) [Authorized]

Endpoints: /api/Constituencies/current – obecne

/api/Constituencies/filtered?electionId={electionId} – w zaleznosci od wyborów //?

Parametry: {electionId} – identyfikator wyborów

Returns: List<ConstituencyCountiesViewModel>

public class ConstituencyCountiesViewModel

{

public int IdConstituancy { get; set; }

[Required]

public string ConstituancyName {get; set; }

[Required]

public List<CountyViewModel> CountiesRange {get; set;}

}

public class CountyViewModel

{

public int IdCounty { get; set; }

[Required]

public string CountyName

}

1. Pobierz okrąg wyborczy (GET) [Authorized]

Endpoint: /api/Constituencies/{constituencyId}

Parametry: {constituencyId}

Returns: ConstituencyCountiesViewModel | NotFound

1. Dodaj okrąg wyborczy (POST) [Authorized]

Endpoint /api/Constituencies

Parametry: ConstituencyCountiesViewModel

Returns: Created | Conflict

1. Modyfikuj okrąg wyborczy (PUT) [Authorized]

Endpoint: /api/Constituencies/{constituencyId}

Parametry: {constituencyId}, ConstituencyCountiesViewModel

Returns: Ok | Conlict | NotFound

1. Usun okrąg wyborczy (DELETE) [Authorized]

Endpoint /api/Constituencies/{constituencyId}

Parametry: {constituencyId}

Returns: Ok | NotFound

* + - 1. Zarządzanie obwodami //czy filtrowanie jak dla kandydatów?

1. Pobierz obecne obwody wyborcze (GET) [Authorized]

Endpoints: /api/Districts/current – obecne

/api/Districts/filtered?electionId={electionId} – w zaleznosci od wyborów //?

Parametry: {electionId} – identyfikator wyborów

Returns: List<DistrictViewModel >

public class DistrictViewModel //Full? //Short??

{

public int IdDistrict { get; set; }

[Required]

public string DistrictName {get; set; }

[Required]

public bool DisabledFacilities {get; set;}

[Required]   
public string DistrictHeadquarters { get; set; }  
  
public LocalisationFilter LocalisationFilter {get; set;}

}

public class LocalisationFilter

{

public VoivodshipViewModel VoivodshipViewModel {get; set;} = !null

public CountyViewModel CountyViewModel {get; set;} = !null

public ProvinceViewModel ProvinceViewModel {get; set;} = !null

}

public class VoivodshipViewModel

{  
 public int IdVoivodeship {get; set;}  
 [Required]  
 punlic int VoivodeshipName {get; set;}

}

public class CountyViewModel

{  
 public int IdCounty {get; set;}  
 [Required]  
 punlic int CountyName {get; set;}

}

public class ProvinceViewModel

{  
 public int IdProvince {get; set;}  
 [Required]  
 punlic int ProvinceName {get; set;}

}

1. Pobierz obwód wyborczy (GET) [Authorized]

Endpoints: /api/Districts/{id}

Parametry: {id}

Returns: DistrictViewModel | NotFound

1. Dodaj obwód wyborczy (POST) [Authorized]

Endpoints: /api/Districts

Parametry: DistrictViewModel (można przekazać tylko z ProvinceViewModel)

Returns: Created | Conflict

1. Modyfikuj obwód wyborczy (PUT) [Authorized]
2. Endpoints: /api/Districts/{id}

Parametry: {id}, DistrictViewModel (można przekazać tylko z ProvinceViewModel)

Returns: Ok | Conflict | NotFound

1. Usuń obwód wyborczy (DELETE) [Authorized]

Endpoints: /api/Districts/{id}

Parametry: {id}

Returns: Ok | NotFound

* + - 1. Zarządzanie składem komisji

1. Pobierz spis komisji (GET) [Authorized]

Endpoint: /api/Commissioners/filtered?

voivodeshipId={voivodeshipId}&countyId={countyId}&provinceId={provinceid}&districtId={disttrictId}

Parametry – poszczególne id-ki służące do filtrowania, możliwe pominięcie od końca dla przyjętej konwencji

Returns: List<CommissionerListViewModel> | NotFound

public class CommissionerListViewModel

{

[Required]

public int IdElectionUser { get; set; }

[Required]

public string FirstName { get; set; }

[Required]

public string LastName { get; set; }

[Required]

public string Pesel { get; set; }

[Required]

public string Email { get; set; }

[Required]

public string PhoneNumber { get; set; }

[Required]

public UserTypeViewModel CommisionMemberLevel {get; set;}

}

1. Pobierz członka komisji (GET) [Authorized]

Endpoint: /api/Commissioners/{idElectionUser}

Parametry: {idElectionUser}

Returns: CommisionerViewModel | NotFound

public class CommissionerViewModel

{

public int IdElectionUser { get; set; }

[Required]

public string Pesel { get; set; }

[Required]

public UserTypeViewModel CommisionMemberLevel {get; set;}

public LocalisationFilter LocalisationFilter {get; set;} = !null

[Required]

public DistrictViewModelShort CommissionMemberDistrict

}

public class DistrictViewModelShort

{

public int IdDistrict { get; set; }

[Required]

public string DistrictName {get; set; }

}

1. Dodaj członka komisji do obwodu (POST) [Authorized]

Endpoint: /api/Commissioners

Parametry: CommissionerViewModel

Returns: Created | Conflict

1. Modyfikuj członka komisji w obwodzie (PUT) [Authorized]

Endpoint: /api/Commissioners/{idElectionUser}

Parametry: {idElectionUser}, CommissionerViewModel

Returns: Ok | Conflict | NotFound

1. Usuń członka komisji z obwodu (DELETE) //Obwód i tak zostaje zmieniają się tylko uprawnienia

Endpoint: /api/Commissioners/{pesel}

Parametry: {pesel}

Returns: Ok | NotFound

* + - 1. Zarządzanie głosującymi – problem z obowiązkowością powiązania z Obwodami w chwili zakładania konta dla ElectionUser – Voter

1. Pobierz spis wyborców (GET) [Authorized] – VoterListViewModel

Endpoint: /api/Voters/filtered?

voivodeshipId={voivodeshipId}&countyId={countyId}&provinceId={provinceid}&districtId={districtId}

Parametry – poszczególne id-ki służące do filtrowania, możliwe pominięcie od końca dla przyjętej konwencji

Returns: List< VoterListViewModel > | NotFound

public class VoterListViewModel

{

[Required]

public int IdVoter { get; set; }

[Required]

public string FirstName { get; set; }

[Required]

public string LastName { get; set; }

[Required]

public string Pesel { get; set; }

[Required]

public string Email { get; set; }

[Required]

public string PhoneNumber { get; set; }

}

1. Pobierz wyborcę (GET) [Authorized]   
   Endpoint /api/Voters/{idVoter}

Paremtry: {idVoter}

Returns: VoterViewModel | NotFound

public class VoterViewModel

{

public int IdVoter { get; set; }

[Required]

public string Pesel { get; set; }

[Required]

public LocalisationFilter LocalisationFilter {get; set;} = !null

[Required]

public DistrictViewModelShort VoterDistrict {get; set;}

}

1. Dodaj wyborcę do obwodu (POST) [Authorized]

Endpoint: /api/Voters

Parametry: VoterViewModel

Returns: Created | Conflict

1. Modyfikuj wyborcę w obwodzie (PUT) [Authorized]

Endpoint: /api/Voters/{idVoter}

Parametry: {idVoter}, VoterViewModel

Returns: Ok | Conflict | NotFound

1. Usuń wyborcę z obwodu (DELETE) [Authorized]  
   Endpoint: /api/Voters/{idVoter}

Parametry: {idVoter}

Returns: Ok | NotFound

1. Pobranie wyborcy do wypisania (GET) [Authorized]

Endpoint: /api/Voters/discharge/{pesel}

1. Wypisywanie wyborcy (POST) [Authorized]

Endpoint: /api/Voters/discharge

Obraz zawierający tekst, zrzut ekranu, Jaskrawoniebieski, Czcionka

Opis wygenerowany automatycznie

* + - 1. Zarządzanie komitetami / kandydatami
      2. Proces głosowania / wybory
      3. Zarządzanie lokalizacjami
      4. Statystyki
      5. Dane do filtrowania:

1. Pobierz dane do podstawowego filtru z wyborami (GET)
2. Pobierz dane do podstawowego filtru tylko względem lokalizacji

1. Oddaj głos (POST)
2. Pobierz informacje o spisie kandydatow (GET)
3. Dodaj wybory (POST)
4. Pobierz obecne wybory (GET)
5. Modyfikuj wybory (PUT)
6. Edytuj okrąg wyborczy (PUT)
7. Pobierz obecne obwody wyborcze (GET)
8. Pobierz okrąg wyborczy (GET)
9. Edytuj (PUT)
10. Usun (DELETE)