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
* Okręgi
* 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:
   1. Claimy:
      1. UserId
      2. ElectionId
      3. CurrentUserTypeId
      4. CurrentDistrictId
      5. 2faVeryfied (bool)
   2. Dla podanych metod trzeba dodać wrappery standardowych HttpGet gdzie endpoint: ../get-from-claim
   3. Role – pole Roles
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. Konwencja nazewnictwa ViewModel’i oraz ich tworzenie (Mapster)
  2. Przetrzymywanie w tokenie informacji na temat bieżące claimu uprawnień, czy wszystkich? – bierzący
  3. Panel zarządzania użytkownikiem / resetowanie hasła / zmiana lub dodanie uprawnień / 2fa?
  4. Dla danych historycznych frekwencji – wielokrotne powielanie okręgów / obwodów (problem z niestałością przypisanych numerów)
  5. Jednoczesne przeprowadzanie wielu wyborów? – dodatkowy claim z ElectionID w JWT?
  6. Tabela person i duplikaty tożsamości, pobranie danych do wypelnienia za pomocą istniejącego rekordu w Person?
  7. Obwody:
     1. Problem z ViewModelami – różne view modele / DTO / jeden wspólny view model ?
  8. 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
     4. Po stronie serwera / klienta ?
  9. Dodawanie składu komisji oraz głosujących dla obwodów w zależnośic od wyborów – zarządzanie okręgami oraz obwodami obejmuje tylko i wyłącznie te obwody / okręgi, które nie są powiązane z zakończonymi wyborami!!!  
     Powinny być te same dla wszystkich wyborów przeprowadzanych jednocześnie

1. Potencjalne modyfikacje w bazie danych:
   1. Kolumna userSecret w ElectionUsers
   2. Wydzielenie encji Headquarters dla zarządzania adresami i lokalizacjami
   3. Każdy obwód / okrąg jest unikalny per wybory przeprowadzane jednocześnie
   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

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 UserTypeViewModel

{

[Required]

public int IdUserType { get; set; }

[Required]

public string UserTypeName { get; set; }

}

1. Manualne odnowienie tokenu (POST) [Authorized]

Endpoint: /api/auth/renew-token

Parametry: UserInfoViewModel – potrzebne do zmiany uprawnień !

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]

Endpoint: /api/Constituencies

Parametry: {electionId} – identyfikator wyborów

Returns: List<ConstituencyCountiesViewModel> | NotFound

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

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

Endpoints: /api/Districts/filtered? voivodeshipId={voivodeshipId}&countyId={countyId}&provinceId={provinceid}

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

Returns: List<DistrictViewModel> | NotFound

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={districtId}

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

{

[Required]

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/{idElectionUser}

Parametry: { idElectionUser }

Returns: Ok | NotFound

* + - 1. Zarządzanie głosującymi

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}

Parametry: {pesel}

Returns: VoterPersonViewModel

public class VoterPersonViewModel

{

[Required]

public int IdVoter { get; set; }

[Required]

public PersonViewModel Person {get; set;}

}

1. Wypisywanie wyborcy (POST) [Authorized]

Endpoint: /api/Voters/discharge

Parametry: VoterPersonViewModel

Obraz zawierający tekst, zrzut ekranu, Jaskrawoniebieski, Czcionka

Opis wygenerowany automatycznie Returns: Ok | NotFound

public class PersonViewModel

{

public int IdPerson { get; set; }

[Required]

public string Name { get; set; }

[Required]

public string Surname { get; set; }

[Required]

public string PESEL { get; set; }

[Required]

public DateTime BirthDate { get; set; }

public string DateOfBirthString

}

* + - 1. Zarządzanie komitetami / kandydatami  
           
         Częściowa implementacja:

1. Pobierz spis kandydatów (GET)
2. Pobierz kandydata (GET)
3. Dodaj kandydata (POST)
4. Modyfikuj kandydata (PUT)
5. Usun kandydata
6. Pobierz komitety (GET) [Authorized]

Endpoint: /api/Parties

Parametry: brak  
  
Returns: List<PartyListViewModel> | NotFound

Wszystkie pola są

public class PartyListViewModel

{

public int IdParty { get; set; }

[Required] public string PartyName { get; set; }

public string? Abbreviation { get; set; }

public string? PartyAddress { get; set; }

[Required] public string PartyType { get; set; }

[Required] public bool IsCoalition { get; set; }

public int? ListCommiteeNumber { get; set; }

public string? Website { get; set; }

}

1. Pobierz komitet wyborczy (GET)

Endpoint /api/Parties/{id}

Parametry: {id}

Returns: PartyViewModel

public class PartyViewModel

{

public int IdParty { get; set; }

[Required] public string PartyName { get; set; }

public string? Abbreviation { get; set; }

public string? PartyAddress { get; set; }

[Required] public string PartyType { get; set; }

[Required] public bool IsCoalition { get; set; }

public int? ListCommiteeNumber { get; set; }

public string? Website { get; set; }

}

1. Dodaj komitet wyborczy (POST) [Authorized]

Endpoint: /api/Parties

Parametry: PartyViewModel

Returns: Created | Conflict

1. Modyfikuj komitet (PUT) [Authorized]

Endpoint /api/Parties/{id}

Parametry: {id}, PartyViewModel

Returns: Ok | Conflict | NotFound

1. Usun komitet (DELETE) [Authorized]

Endpoint /api/Parties/{id}

Parametry {id}

Returns: Ok | NotFound

* + - 1. Proces głosowania / wybory – uwaga potrzebny minimalny okres czasu w którym możemy zarządzać wyborami !

1. Pobierz wszystkie wybory (GET) [Authorized] ???

Endpoint /api/Elections

/api/Elections/current

/api/Elections/past

Parametry: brak

Returns: List<ElectionViewModel> | NotFound

public class ElectionViewModel

{

public int IdElection { get; set; }

[Required]

public DateTime ElectionStartDate { get; set; }

[Required]

public DateTime ElectionEndDate { get; set; }

public int ElectionTour { get; set; }

[Required]

public ElectionTypeViewModel ElectionType { get; set; }

//Brak pól do wprowadzenia w mocku

}

public class ElectionTypeViewModel

{

[Required]   
public int IdElectionType { get; set; }

[Required]

public string ElectionTypeName { get; set; }

}

1. Pobierz wybory (GET) [Authorized]

Endpoint /api/Elections/{id}

Parametry: {id}

Returns: ElectioViewModel | NotFound

1. Zarejestruj wybory (POST) [Authorized]

Endpoint: /api/Elections

Parametry: ElectionViewModel

Returns: Created | Conflict

1. Modyfikuj wybory (PUT) [Authorized]

Endpoint: /api/Elections/{id}

Parametry: {id}, ElectionViewModel

Returns: Ok | Conflict | NotFound

1. Usun wybory (DELETE) [Authorized]

Endpoint: /api/Elections/{id}

Parametry: {id}

Returns: Ok | NotFound

1. Pobierz typy wyborów (GET)

Endpoint: /api/ElectionTypes

Parametry: brak

Returns: List<ElectionTypeViewModel> | NotFound

1. Pobierz typ wyborów (GET)  
   Endpoint: /api/ElectionTypes/{id}

Parametry: brak

Returns: ElectionTypeViewModel

1. Sprawdź status głosowania (POST) [Authorized]

Endpoint /api/election-proccess/check

Parametry ElectionStatusCheckReq

Returns: Ok | Forbidden | NotFound

public class ElectionStatusCheckReq

{

[Required]   
public int IdDistrict { get; set; }

[Required]

public string PESEL { get; set; }

}

1. Zagłosuj (POST) [Authorized = Voter] ?? zaświadczenie - pdf

Endpoint: /api/election-proccess/vote

Parametry: VoteViewModel

Returns: Ok | Forbidden

public class VoteViewModel

{

[Required]   
public int IdElection { get; set; }

public int IdParty {get; set;}

[Required]

public int IdCandidate { get; set; }

[Required]

bool IsValid {get; set; }

[Required]

bool IsConfirmationNeeded {get; set;}

}

* + - 1. Zarządzanie lokalizacjami

Lokalizacje mockujemy w bazie danych – Tylko GET

1. Pobierz województwa
2. Pobierz województwo
3. Pobierz powiaty
4. Pobierz powiat
5. Pobierz gminy
6. Pobierz gminę
   * + 1. Statystyki – numer komisji czy to numer obwodu?
7. Pobierz wszystkie statystyki dla obwodów (GET)

Endpoint: /api/statistics/filtered?electionId={electionId}?voivodeshipId={voivodeshipId}&countyId={countyId}&provinceId={provinceid}&attendance={attendance}

Parametry: poszczególne id-ki służące do filtrowania wyników  
 attendances = (‘12’,’17’,’All’) - frekwencja

Returns: List<StatisticListViewModel> | NotFound

public class StatisticListViewModel

{

[Required]   
public int DistrictName { get; set; }

public int DistrictHeadquarters {get; set;}

[Required]

Public int VotersEligToVoteCount { get; set; }

[Required]

Public int ElectronicVotesCount {get; set; }

public double Percantage {get => ElectronicVotesCount \* 100 / VotersEligToVoteCount }

}

1. Pobierz statystykę dla obwodu [Authorized]  
    szczegółowe wyniki – ekran wyników wyborów z podziałem na obwody

Endpoint: /api/statistics/{id}  
 /api/statistic/current-district – DistrictId z claimów z JWT

Parametry: {id} / brak

Returns: StatisticDistrictViewModel | NotFound

public class StatisticListViewModel

{

[Required]   
public string DistrictName { get; set; }

[Required]

Public int VotersEligToVoteCount { get; set; }

public double AttendanceAt12 {get; set; }

public double AttendanceAt17 {get; set;}

}

* + - 1. Dane do filtrowania:   
           
         Uwaga w zależności od potrzeb w trakcie developmentu można dodawać warianty bazujące na FilterListWrapper

1. Pobierz dane do podstawowego filtru z wyborami (GET)

Endpoint: /api/get-filters/full? electionId={electionId}?voivodeshipId={voivodeshipId}&countyId={countyId}

Parametry: poszczególne idki – ważna kolejność

Returns: FilterListWrapperFull

public class FilterListWrapperFull

{

[Required]   
public List<ElectionViewModelShort> electionFilter { get;}

[Required]

public FilterListWrapper localisationFilters { get;}

}

1. Pobierz dane do podstawowego filtru tylko względem lokalizacji

Endpoint: /api/get-filters/full? voivodeshipId={voivodeshipId}&countyId={countyId}&provinceId={provinceid}

Parametry: poszczególne idki – ważna kolejność

Returns: FilterListWrapper

public class FilterListWrapper

{

[Required]   
public List<VoivodeshipViewModelShort> voivodeshipFilter { get; }

[Required]   
public List<CountyViewModel> countyFilter { get; }

[Required]

Public List<ProvinceViewModel> provinceFilter {get; }

}

1. Pobierz dane do filtru z obwodami

Endpoint: /api/get-filters/with-districts? voivodeshipId={voivodeshipId}&countyId={countyId}&provinceId={provinceid}

Parametry: poszczególne idki – ważna kolejność

Returns: FilterListWrapperWithDistrict

public class FilterListWrapperWithDistrict

{

[Required]   
public List <DistrictViewModelShort> DitrictFilter { get; }

[Required]   
public FilterListWrapper LocalisationFilters { get; }

}