Név: Schmidt Bálint Márk

Neptun-kód: [redacted]

Csoport: [redacted]

Gyakorlatvezető: [redacted]

Feladat: Anonim szavazó rendszer

Készítsünk egy online anonim szavazó rendszert. A kliens-szerver architektúrájú webalkalmazáson keresztül a rendszer felhasználóinak (vagy egy részüknek) kérhetjük ki a véleményét a kérdés és a válasz opciók megadásával. A szavazások minden esetben titkosak legyenek, amelyet a programnak garantálnia kell (külön tárolandó a szavazatukat gyakorló felhasználók és a leadott szavazat).

Általános követelmények

A feladat részeként egy REST architektúrát követő WebAPI webszolgáltatást kell megvalósítani ASP.NET keretrendszerben, C# programozási nyelven. A webszolgáltatáshoz egy látogatói és egy adminisztrációs kliens alkalmazást is fejleszteni kell.

• A WebAPI szolgáltatást MVC architektúrának megfelelően kell elkészíteni.

• A látogatói kliens alkalmazás szabadon választható technológiával, webes alkalmazásként, asztali alkalmazásként vagy mobil alkalmazásként is elkészíthető. Az alkalmazást a választott platformnak megfelelően, legalább kétrétegű architketúrában kell elkészíteni.

• Az adminisztrációs klienst a Blazor szoftverkönyvtár használatával, MV(VM) architektúrában kell elkészíteni legalább 1 támogatott platformra.

• A látogatói és az adminisztrációs alkalmazást együttesen is be kell tudni mutatni, a két alkalmazásnak funkcionalitásban egymást kiegészítve kell együtt működőképesnek lennie. A feladatkiírás alapfeladatból (2+1 pont) és opcionális választható részfeladatokból áll. A beadandó értékelése az elfogadott funkcionalitások összpontszámának egész része. Jeles (5+1) érdemjegy teljesítéséhez kötelező legalább egy *SignalR* alapú funkció megvalósítása.

Alapfeladat (2+1 pont)

Látogatói felület: készítsünk REST architektúrájú webalkalmazást és hozzá webes felületű, asztali grafikus vagy mobil kliens alkalmazást, amelyen keresztül a felhasználók az aktív kérdésekben szavazhatnak, valamint megtekinthetőek a korábbi szavazások eredményei.

• A felhasználók email cím és jelszó megadásával regisztrálhatnak, valamint jelentkezhetnek be. A portál további funkciói csak bejelentkezést követően érhetőek el.

• Bejelentkezést követő megjelenő felületen látható az aktív szavazások listája. Aktív az a szavazás, amely már elkezdődött, de még nem fejeződött be. A szavazásokat a befejező dátumuk szerint növekvő sorrendben kell listázni a kérdés szövegének, valamint a kezdő és befejező időpontnak a feltüntetésével. Vizuálisan legyen egyértelműen jelölve, hogy egy aktív szavazáson már szavazott-e a felhasználó vagy sem.

• Egy aktív szavazás kiválasztásával az alkalmazás jelenítse meg a kérdést és a válasz opciókat. Utóbbiak közül pontosan egyet kiválasztva lehet a szavazatot érvényesen leadni.

• A bejelentkezett felhasználók egy másik felületen kilistázhatják a már lezárult szavazásokat. Lezárultnak tekintendő az a szavazás, amelynek befejező időpontja elmúlt. A szavazások listáját lehessen szűrni a kérdés szövegének részlete vagy időintervallum alapján.

• Egy lezárult szavazás kiválasztásával a weboldal jelenítse meg annak eredményét, azaz válasz opciónkként a szavazatok száma és százalékos értéke.

Adminisztrációs felület: készítsünk Blazor keretrendszerre épülő kliens alkalmazást, amelyen keresztül új szavazásokat lehet kiírni a rendszerben. Bármely felhasználó írhat ki új szavazást.

• A felhasználók bejelentkezhetnek (email cím és jelszó megadásával) a programba. Sikeres bejelentkezést követően látja az általa kiírt korábbi szavazások listáját.

• Egy szavazást kiválasztva megjelenítésre kerül a feltett kérdés és a válasz opciók, valamint a szavazás kezdete és vége. Továbbá kerüljön megjelenítésre a szavazáshoz rendelt felhasználók listája, jelölve, hogy mely felhasználók szavaztak már és melyek nem.

• Legyen lehetőség új szavazás kiírására a kérdés, a dinamikus számú (legalább 2) válasz opció, továbbá a kezdő és a befejező időpont megadásával. A kezdő és a vég időpont létező kell legyen, továbbá mindkettőnek jövőbelinek kell lennie és a vég időpontnak legalább 15 perccel követnie kell a kezdőidőpontot.

• A szavazás a kiírása után már nem módosítható.

A feladat elkészítése során lehetőség volt választható részfeladatok megvalósítására is. A beadandóban én csak az alapfeladatot valósítottam meg.

Funkcionális elemzés: Anonim szavazó rendszer

Rendszer célja

A rendszer célja egy olyan webalapú, kliens–szerver architektúrájú anonim szavazó platform létrehozása, amely lehetővé teszi:

- Szavazások kiírását,
- Felhasználói vélemények begyűjtését titkosan,
- Az eredmények megjelenítését,
- Felhasználók regisztrációját, bejelentkezését.

A rendszer biztosítja, hogy a leadott szavazatok ne legyenek összeköthetőek a szavazó személyazonosságával.

Főbb rendszerkomponensek

1. WebAPI (Backend)

- ASP.NET alapú REST API, MVC architektúra
- Felhasználó- és jogosultságkezelés
- Szavazáskezelés (kiírás, lekérdezés, lezárás)
- Anonim szavazatkezelés (titkosított tárolás)

2. Látogatói kliens (React)

- Legalább kétrétegű architektúra
- Felhasználók regisztrálhatnak, bejelentkezhetnek
- Aktív szavazások listázása, szűrése, szavazás lehetősége
- Lezárt szavazások listázása, szűrése
- Szavazási eredmények megjelenítése

3. Adminisztrációs kliens

- Blazor technológiával készült alkalmazás
- Szavazások létrehozása, listázása, részleteinek megtekintése
- Szavazáskezelés (adatok validálása új szavazás létrehozásakor)
- Szavazási státuszok vizualizálása

Funkcionális követelmények

Felhasználók

- Regisztráció (név, felhasználó név, email, jelszó)
- Bejelentkezés (token alapú azonosítás)
- Jogosultság alapján hozzáférés korlátozása (pl. szavazás csak belépés után)

Szavazások

- Szavazások szűrése (időpont és szöveg alapján)
- Szavazás részleteinek megtekintése
- Aktív és lezárt szavazások elkülönítése
- Szavazat leadása (egy opció választható)

Eredmények

- Lezárt szavazásoknál: eredmények számszerű és százalékos megjelenítése

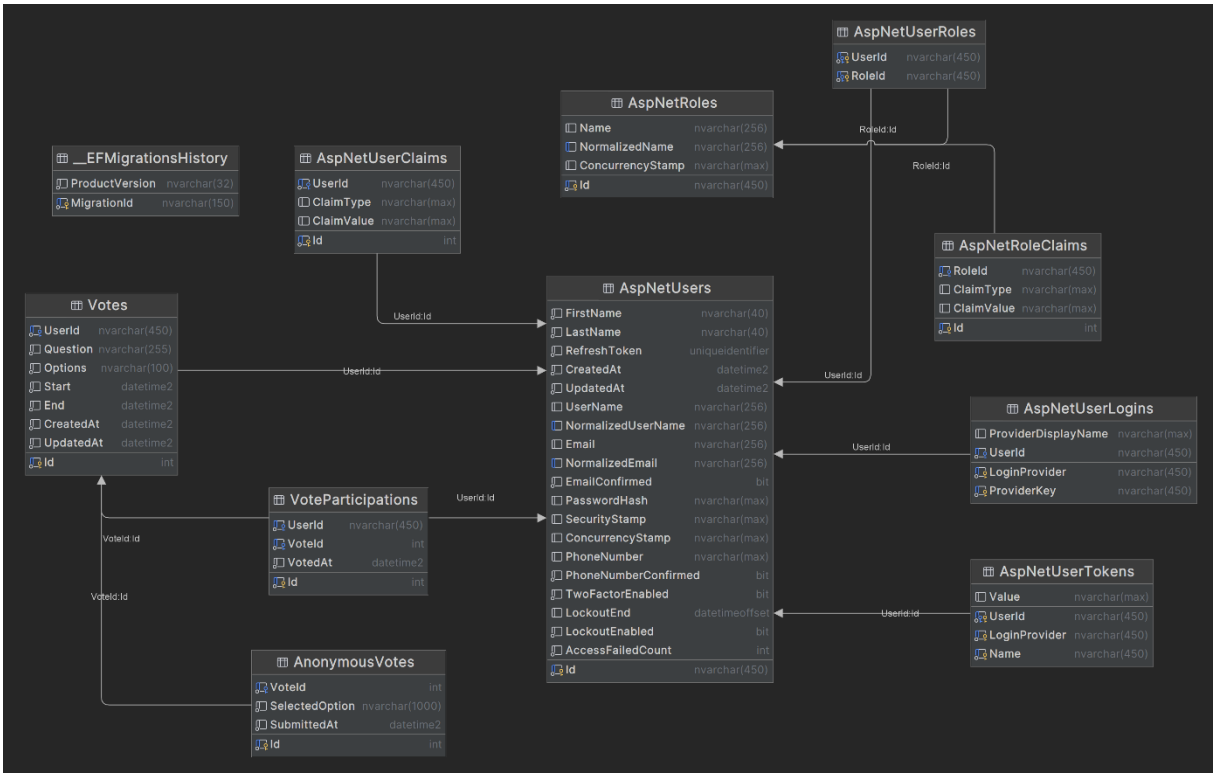
Adminisztratív funkciók

- Új szavazás létrehozása (kérdés, opciók, időintervallum)
- Szavazás státuszának megtekintése (alapadatok, ki szavazott, ki nem)

Nem-funkcionális követelmények

- Biztonság: token alapú autentikáció (JWT)
- Titkosság: szavazatok és felhasználói információk elkülönített tárolása
- Adatintegritás: időbeli ellenőrzések szavazás létrehozásakor
- Bővíthetőség: REST API + Blazor + frontend szeparáció

Az adatbázis felépítése



Az adatbázis a .NET Identity által biztosított felhasználókezelést egészíti ki szavazási funkciókat támogató táblákkal. Az adatmodell biztosítja az anonim, de ellenőrzötten egyedi szavazást.

Felhasználói és jogosultsági táblák (Identity)

- **AspNetUsers**
  A regisztrált felhasználók adatait tárolja. Ide tartozik:
  - Id (PK): felhasználó azonosító (nvarchar(450))
  - Email, jelszó, név, biztonsági jellemzők (pl. 2FA, lockout stb.)
  - RefreshToken: a bejelentkezéshez használt token
- **AspNetRoles**
  Rendszerbeli szerepkörök (pl. Admin, User).
- **AspNetUserRoles**
  Kapcsolótábla a felhasználók és szerepkörök között (sok-sok kapcsolat).
- **AspNetUserClaims**, **AspNetRoleClaims**
  Felhasználói és szerepköri jogosultságok egyedi igényekhez.
- **AspNetUserLogins**, **AspNetUserTokens**
  Külső bejelentkezések és tokenek kezelésére szolgálnak.

Szavazási funkciók táblái

- **Votes**
  Egy szavazási eseményt reprezentál:

  - Id (PK): szavazás azonosító
  - UserId (FK): kiíró felhasználó
  - Question: a szavazás kérdése
  - Options: válaszlehetőségek egy tömbben tárolva
  - Start, End: kezdési és befejezési időpont
  - CreatedAt, UpdatedAt: metaadatok

- **AnonymousVotes**
  A tényleges szavazatok táblája:
  - Id (PK)
  - VoteId (FK): melyik szavazásra vonatkozik
  - SelectedOption: a választott lehetőség
  - SubmittedAt: leadás ideje

  Nem tartalmaz felhasználói azonosítót, garantálja az anonimitást.

- **VoteParticipations**
  Ez a táblázat követi, hogy **ki szavazott már**:
  - Id (PK)
  - UserId (FK): a felhasználó
  - VoteId (FK): a szavazás
  - VotedAt: a szavazat leadásának időpontja

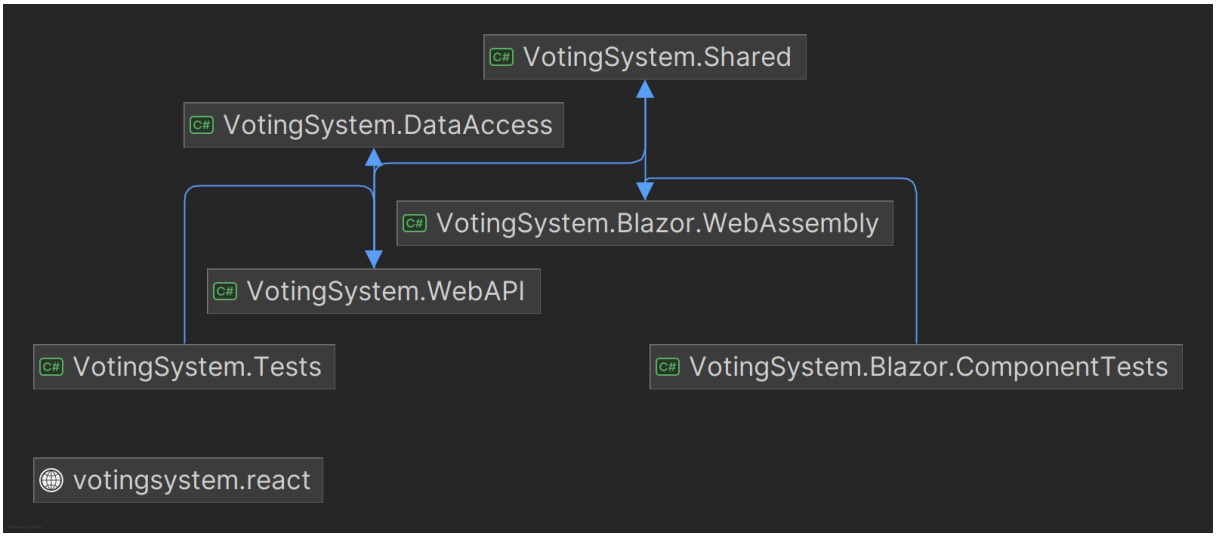
  Ez teszi lehetővé, hogy ellenőrizzük az egyszeri szavazást, de ne kapcsoljuk össze a szavazatot a felhasználóval.

Migrációs metaadat

- **__EFMigrationsHistory**
  Az Entity Framework automatikus adatbázis-migrációinak verziókövetése.

Rendszer statikus terve – Komponensdiagram leírása

A rendszer több különálló, de egymással együttműködő komponensből épül fel, amelyek különböző szerepköröket töltenek be a rendszer működésében. A következő ábra a rendszer fő komponenseit és azok kapcsolatait mutatja be:



Komponensek leírása:

1. VotingSystem.Shared

- **Típus:** Osztott projekt
- **Szerep:** Közös adattípusokat, DTO-kat (Data Transfer Object tartalmaz.
- **Felhasználói:**
  - VotingSystem.WebAPI
  - VotingSystem.Blazor.WebAssembly
  - VotingSystem.DataAccess
  - VotingSystem.Blazor.ComponentTests

2. VotingSystem.DataAccess

- **Típus:** Osztálykönyvtár
- **Szerep:** Az adatbázis-hozzáférést és az entitásmodelleket tartalmazza (Entity Framework Core használatával).
- **Felhasználói:**
  - VotingSystem.WebAPI

3. VotingSystem.WebAPI

- **Típus:** ASP.NET Web API
- **Szerep:** A backend üzleti logikát és REST API végpontokat biztosít a kliensek számára. Authentikációt, szavazáskezelést, jogosultságkezelést és adatbázisműveleteket végez.
- **Függőségei:**
  - VotingSystem.Shared
  - VotingSystem.DataAccess

- **Felhasználói:**
    - VotingSystem.Tests
    - VotingSystem.Blazor.WebAssembly (HTTP hívásokon keresztül)
    - votingsystem.react (HTTP hívásokon keresztül)

4. VotingSystem.Blazor.WebAssembly

- **Típus:** Blazor WebAssembly alkalmazás
- **Szerep:** Adminisztrációs felület, amely lehetővé teszi a szavazások kezelését, felhasználók adminisztrálását és az eredmények megjelenítését.
- **Függőségei:**
    - VotingSystem.Shared
- **Felhasználói:**
    - VotingSystem.Blazor.ComponentTests

5. votingsystem.react

- **Típus:** React alapú frontend (JavaScript)
- **Szerep:** A látogatók számára biztosított publikus felület, ahol a felhasználók regisztrálhatnak, bejelentkezhetnek, szavazhatnak, és megtekinthetik az eredményeket.
- **Kapcsolat:**
    - Közvetlen HTTP-kommunikáció a VotingSystem.WebAPI-val

6. VotingSystem.Tests

- **Típus:** xUnit alapú C# tesztprojekt
- **Szerep:** Egység- és integrációs teszteket tartalmaz a VotingSystem.WebAPI funkcióinak ellenőrzésére.
- **Függőségei:**
    - VotingSystem.WebAPI
    - VotingSystem.Shared
    - VotingSystem.DataAccess

7. VotingSystem.Blazor.ComponentTests

- **Típus:** Tesztprojekt (bUnit)
- **Szerep:** UI komponensek tesztelése a VotingSystem.Blazor.WebAssembly alkalmazásban.
- **Függőségei:**
    - VotingSystem.Shared
    - VotingSystem.Blazor.WebAssembly

DataAccess réteg – Adatbázismodell és entitáskapcsolatok

Áttekintés

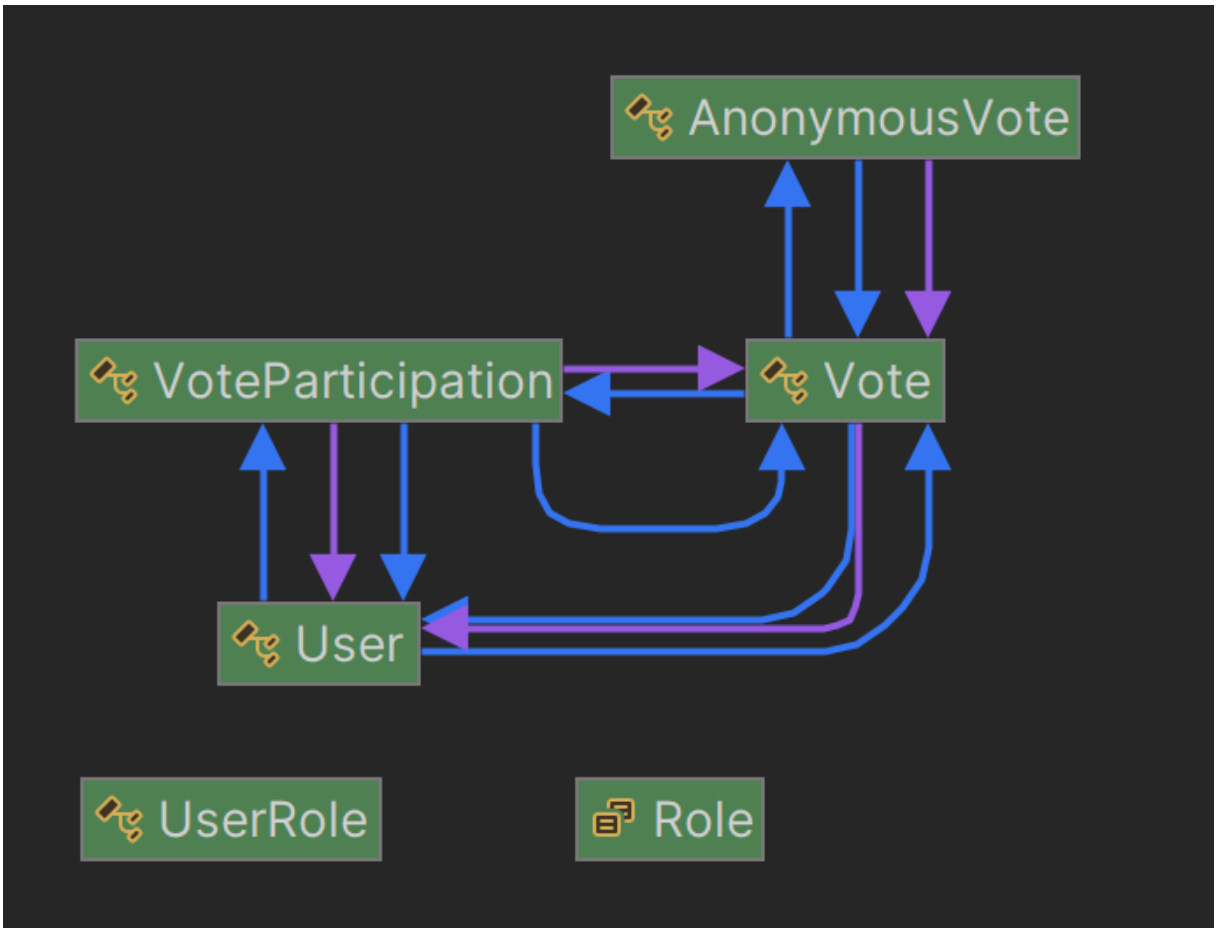
A VotingSystem.DataAccess projekt felelős az alkalmazás adatkezelési rétegéért, különösen az entitások, adatbáziskapcsolatok és a migrációk kezeléséért. Az Entity Framework Core ORM segítségével valósítja meg az objektum-relációs leképezést. Az alábbi diagram a főbb entitások közötti kapcsolatokat mutatja be:

Entitások és kapcsolataik

User

A rendszer regisztrált felhasználóit reprezentálja. Tartalmazza az alapvető információkat (pl. név, email, jelszó hash, státuszok).

- **Kapcsolatok:**
    - Vote (1:N): Egy felhasználó több szavazást is létrehozhat.
    - VoteParticipation (1:N): Egy felhasználó több szavazásban vehet részt.
    - UserRole (1:N): A felhasználó jogosultságait tárolja.



Vote

Egy konkrét szavazási eseményt reprezentál, amely egy kérdést és válaszlehetőségeket tartalmaz. Az Options mező stringként van tárolva, ;-vel elválasztott lehetőségekkel.

- **Kapcsolatok:**
    - User (N:1): A szavazás tulajdonosa.
    - VoteParticipation (1:N): A felhasználók részvételét tartalmazza.
    - AnonymousVote (1:N): A névtelen szavazatok kerülnek ide.

VoteParticipation

Kapcsolótábla, amely nyilvántartja, hogy melyik felhasználó melyik szavazásban vett részt.

- **Kapcsolatok:**
    - User (N:1): A szavazó felhasználó.
    - Vote (N:1): A szavazás, amiben részt vett.

AnonymousVote

Az anonim szavazásokat reprezentálja, ahol nem tárolunk felhasználói azonosítót, csak a választott opciót és a beküldés időpontját.

- **Kapcsolatok:**
    - Vote (N:1): A szavazás, amelyhez az anonim szavazat tartozik.

UserRole

Kapcsolótábla a User és Role entitások között. Lehetővé teszi, hogy egy felhasználónak több szerepköre is lehessen.

- **Kapcsolatok:**
    - User (N:1)
    - Role (N:1)

Role

A rendszerben elérhető szerepköröket tartalmazza (pl. Admin, Felhasználó).

- **Kapcsolatok:**
    - UserRole (1:N)

VotingSystem.Shared – DTO-k

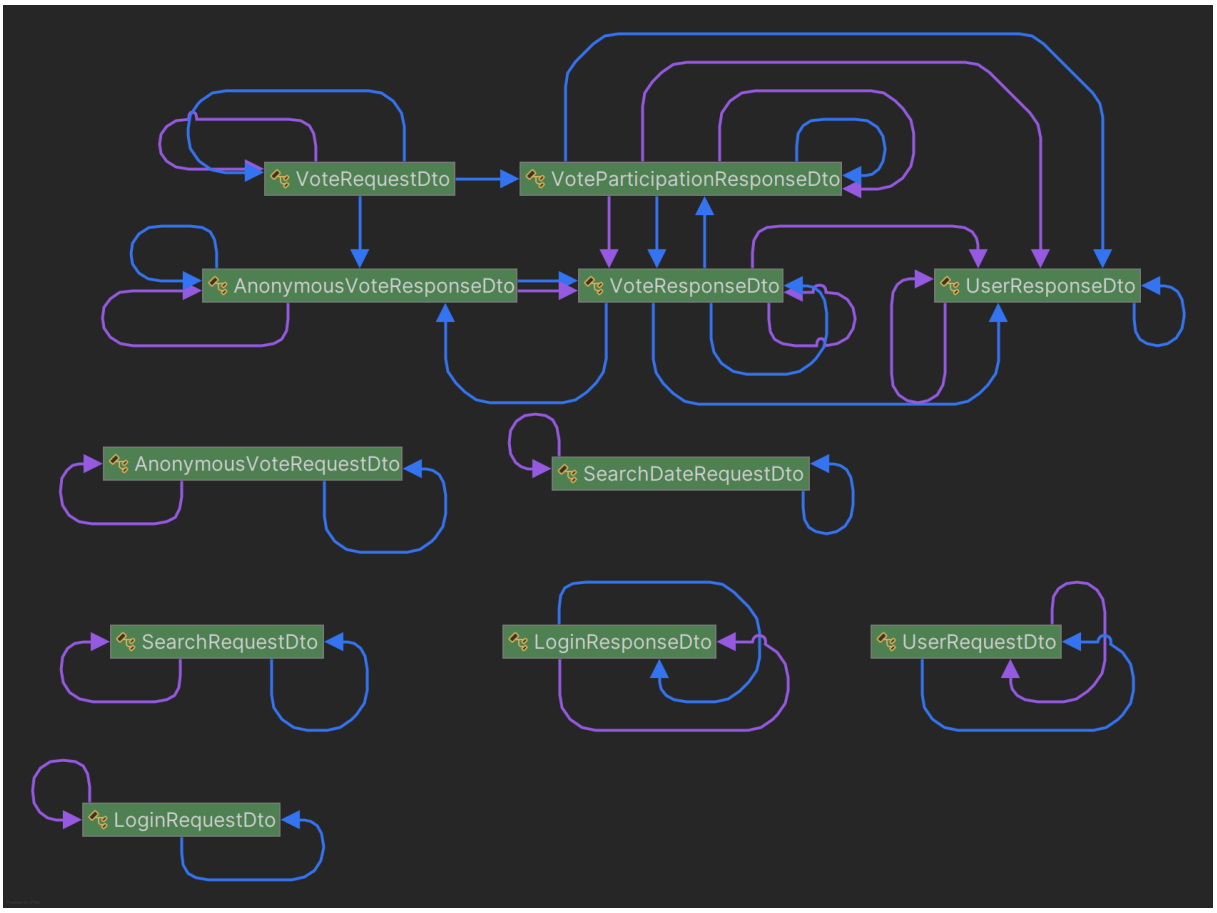
Áttekintés

A VotingSystem.Shared projekt tartalmazza az alkalmazás **Data Transfer Object (DTO)** osztályait, amelyek az adatforgalmat szabályozzák a különböző komponensek (WebAPI, WebAssembly, React) között. Ezek az osztályok biztosítják, hogy a backend és a frontend világosan és típusbiztosan kommunikáljon egymással.

A DTO-k általában az alábbi mintát követik:

- A RequestDto típusok a frontendtől a szerver felé haladnak.

- A ResponseDto típusok a backend válaszai.
- Több ResponseDto egymásba ágyazva dolgozik (pl. VoteResponseDto → UserResponseDto).



VoteRequestDto

A szavazás létrehozására szolgáló kérés objektum. Tartalmazza:

- Létrehozó felhasználó id-je
- Kérdés szövege
- Lehetőségek (string tömb)
- Kezdési dátum
- Befejezési dátum

VoteResponseDto

A szavazás lekérdezésének válasza:

- Azonosító
- Létrehozó felhasználó
- Részvétel adatai (→ VoteParticipationResponseDto)
- Anonim szavazatok (→ AnonymousVoteResponseDto)

VoteParticipationResponseDto

A felhasználók szavazásban való részvételének adatai:

- Szavazó felhasználó adatai (→ UserResponseDto)
- Választott opció

AnonymousVoteRequestDto

Kérés anonim szavazat leadására:

- Vote ID
- Választott opció

AnonymousVoteResponseDto

Válasz anonim szavazatra:

- Vote ID
- Beküldési időpont
- Választott opció

UserRequestDto

Kérés új felhasználó regisztrálására vagy módosítására:

- Név, e-mail, jelszó stb.

UserResponseDto

Felhasználói adatok:

- Név, e-mail
- Szerepkör(ök)
- Szavazások, amiket létrehozott vagy amelyekben részt vett

LoginRequestDto

Bejelentkezési kérés:

- Email
- Jelszó

LoginResponseDto

Bejelentkezés válasz:

- Felhasználó azonosító
- Token (ha van)
- Szerepkör(ök)

SearchRequestDto

Szavazások szűrésére szolgáló kérés:

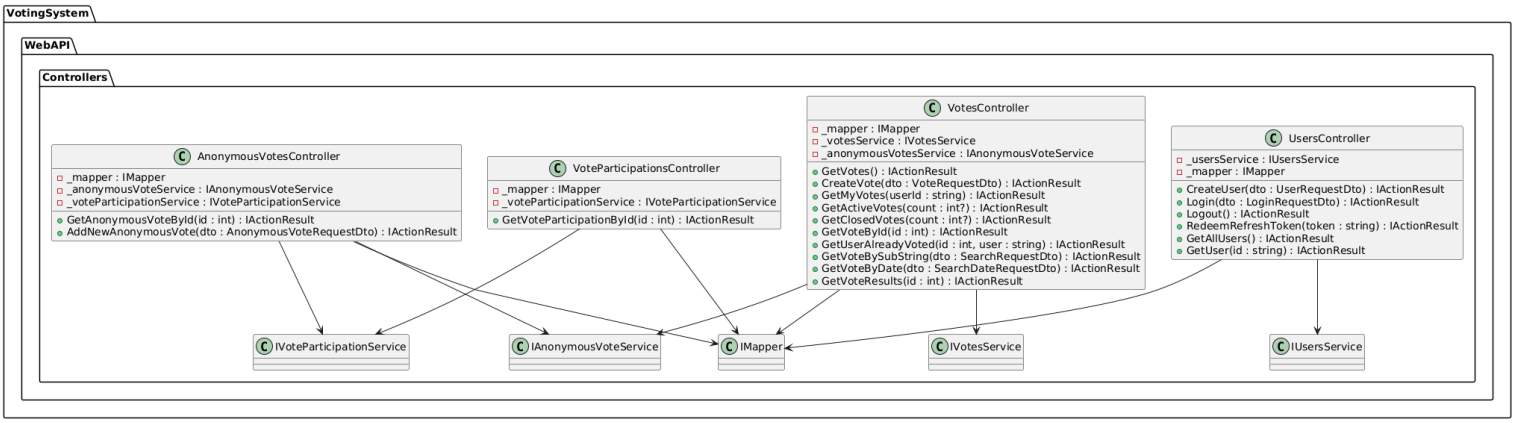
- Keresett szöveg (pl. szavazás kérdésben)
- Dátumtartomány

SearchDateRequestDto

Dátumos keresésre specializált DTO:

- FromDate
- ToDate

WebAPI Kontrollerek



UsersController

- POST /users: Új felhasználó létrehozása.
- POST /users/login: Bejelentkezés hitelesítéssel.
- POST /users/logout: Kijelentkezés (auth szükséges).
- POST /users/refresh: Token frissítése.
- GET /users: Összes felhasználó lekérése.
- GET /users/{id}: Felhasználó lekérése azonosító alapján.

VotesController

- GET /votes: Összes szavazás lekérése.
- GET /votes/active: Aktív szavazások lekérése.
- GET /votes/closed: Lezárt szavazások lekérése.
- GET /votes/{id}: Egy szavazás lekérése ID alapján.
- GET /votes/voted/{id}/{user}: Ellenőrzés, hogy a felhasználó szavazott-e.
- GET /votes/{id}/results: Szavazás eredményének lekérése.

- POST /votes: Szavazás létrehozása.
- POST /votes/my: Saját szavazások lekérése userId alapján.
- POST /votes/search: Szavazások keresése kulcsszó alapján.
- POST /votes/search-by-date: Szavazások keresése dátum alapján.

AnonymousVotesController

- GET /anonymousvotes/{id}: Anonim szavazat lekérése ID alapján.
- POST /anonymousvotes: Új anonim szavazat hozzáadása (azonosított felhasználóhoz kötve).

VoteParticipationsController

- GET /voteparticipations/{id}: Részvétel lekérése szavazáson ID alapján.

React (Front-end, látogatói felület)

Belépési pontok

- **main.tsx**
  Alkalmazás belépési pontja, itt történik a ReactDOM.createRoot() hívás és a BrowserRouter + UserContextProvider beágyazása.
- **index.css**
  Alap CSS stílusok.

API modul (/api)

- **client/**
  - http.ts: Axios konfiguráció (baseURL, interceptors).
  - users-client.ts: Felhasználói endpoint hívások (login, register).
  - votes-client.ts: Szavazás-specifikus API hívások.
- **models/**
  DTO-k az adatcsere szabványosításához, pl.:
  - LoginRequestDto.ts
  - UserResponseDto.ts
  - VoteResponseDto.ts
- **errors/**
  - HttpError.ts, ServerSideValidationError.ts: Hibakezelő típusok/API válaszmodellek.

Oldalak (/pages)

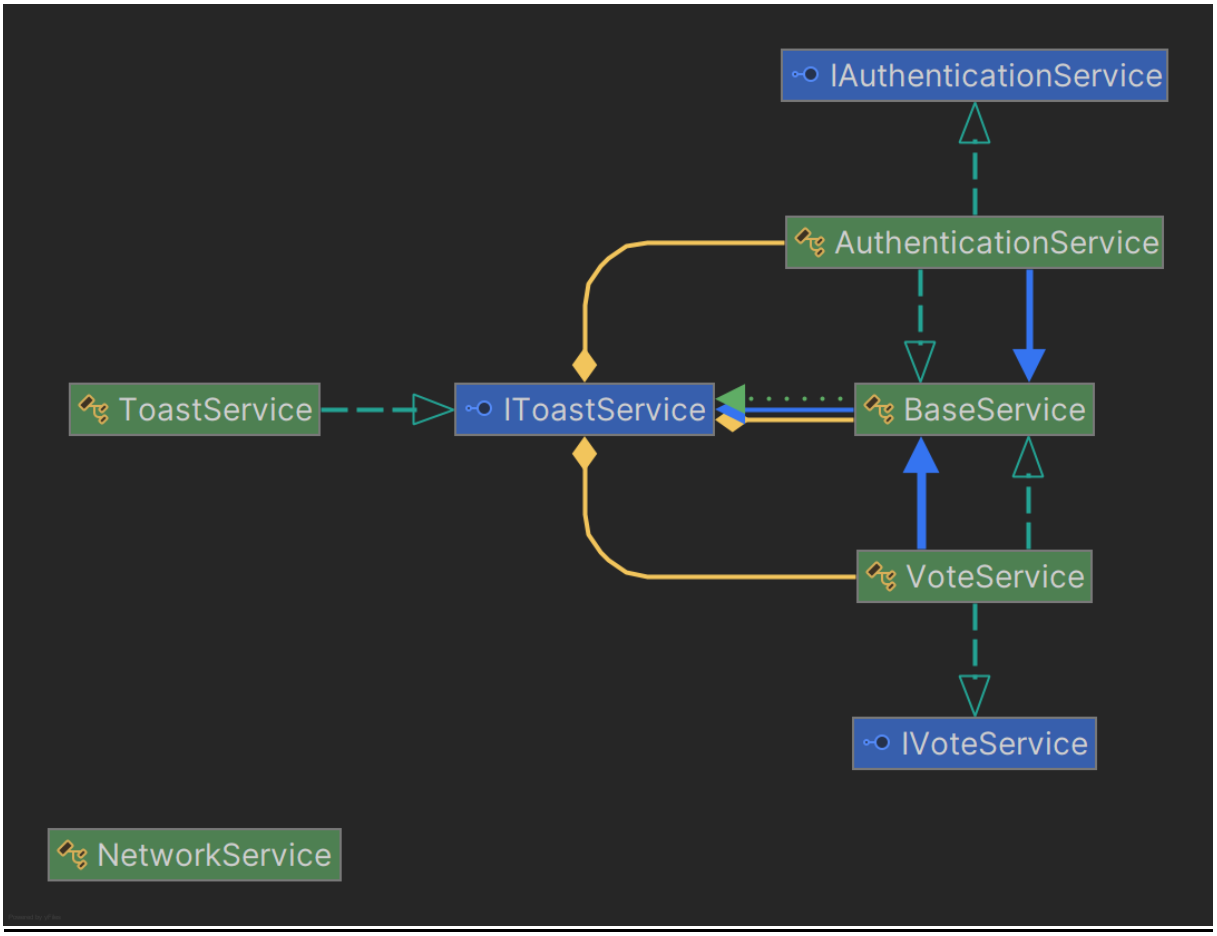
- **HomePage.tsx**
  Főoldal, listázza az aktív/inaktív szavazásokat.

- **RootLayout.tsx**
  Alap layout, tartalmazza a Header-t.
- **NotFoundPage.tsx**
  404-es oldal.
- **/pages/user/**
  - LoginPage.tsx, RegisterPage.tsx: Bejelentkezés / regisztráció formok.
  - LogoutPage.tsx: Felhasználó kijelentkeztetése.
- **/pages/votes/**
  - VotePageActive.tsx: Aktív szavazás, szavazás lehetőséggel.
  - VotePageClosed.tsx: Lezárt szavazás eredményekkel.
  - VoteSuccessPage.tsx: Sikeres szavazás visszajelzés.

Komponensek (/components)

- **header/**
  - Header.tsx: Navigációs fejléc.
  - HeaderLink.tsx: Egyedi linkelem stílus a navigációhoz.
- **votes/**
  - VoteCard.tsx: Szavazási kártya megjelenítése.
  - VotesGrid.tsx: Rács elrendezés több szavazás kártyához.
- **alerts/**
  - ErrorAlert.tsx, RouterSuccessAlert.tsx: Hibák és sikeres műveletek visszajelzése.
- **buttons/**
  - LinkButton.tsx, SubmitButton.tsx: Újrafelhasználható gomb komponensek.
- **Egyéb:**
  - LoadingIndicator.tsx: Betöltést jelző komponens.
  - FormError.tsx: Űrlaphibák megjelenítése.
  - Protected.tsx: Jogosultság-ellenőrzés komponensként.

Blazor.WebAssembly (Adminisztrációs felület)



Service-k

- IAuthenticationService és AuthenticationService
  Ez az interfész felelős az autentikációért. A felhasználók hitelesítése nélkül nem tudnak semmilyen funkciót igénybe venni.
- IToastService és ToastService
  Ezek a szolgáltatások a felhasználói értesítéseket (toast üzeneteket) kezelik, például megerősítések vagy hibaüzenetek megjelenítésére.
- NetworkService
  Ez a komponens biztosítja a hálózati kommunikációt, például a szavazatok küldését és az adatok lekérdezését.
- BaseService
  Ez az absztrakt osztály vagy alap szolgáltatás, amely a hálózattal és hibákkal kapcsolatos szolgáltatások összehangolásáért felel.
- IVoteService és VoteService
  Ez az interfész és szolgáltatás a szavazatok kezeléséért felelős, lehetővé téve új szavazat létrehozását vagy korábbi szavazások megtekintését.

Blazor.ComponentTests

MenuComponentTests

Ez az osztály a MenuComponent komponens viselkedését teszteli különböző autentikációs állapotok esetén. A teszteléshez a bUnit keretrendszert használja, valamint egy mockolt IAuthenticationService példányt és a FakeNavigationManager-t a navigáció szimulálására.

Konstruktor és Setup

A konstruktorban:

- Beállításra kerül egy IAuthenticationService mock, amely egy bejelentkezett felhasználót ("user2") szimulál.
- A mock regisztrálásra kerül az IServiceCollection-ben.
- A FakeNavigationManager segítségével tesztelhetővé válik a navigáció.

Tesztek

Navbar_WhenUserIsAuthenticated_ShouldShowFullMenu

Ellenőrzi, hogy egy bejelentkezett felhasználó esetén:

- A komponens megjeleníti a címet (Anonymous Voting Administration),
- A navigációs menüpontok elérhetők,
- A felhasználó üdvözlő szövege megjelenik (Welcome, user2!),
- A kijelentkezés gomb (Logout) látható.

LogoutButton_WhenClicked_ShouldCallLogoutAndRedirectToLoginPage

A teszt azt vizsgálja, hogy ha a kijelentkezés gombra kattint a felhasználó:

- Meghívásra kerül az IAuthenticationService.LogoutAsync metódus pontosan egyszer,
- A felhasználó átirányításra kerül a bejelentkezési oldalra (/login).

VoteAddTests

Ez az osztály a szavazás hozzáadásához és megjelenítéséhez kapcsolódó Blazor komponensek működését teszteli. A tesztkörnyezet a bUnit és több mock szolgáltatás segítségével kerül beállításra.

Konstruktor és Setup

A konstruktor:

- Két tesztszavazatot hoz létre és regisztrál a VoteService mockon keresztül,
- A következő szolgáltatásokat regisztrálja a DI konténerbe: IVoteService, IJSRuntime, valamint egy konfigurációs objektum (AppConfig).

A IVoteService mock több metódust szimulál, többek között: GetVoteByIdAsync, GetMyVotesAsync, GetAllUsersAsync, GetVotesAsync.

Tesztek

VoteList_ShouldShowCorrectNumberOfVotes

Ellenőrzi, hogy a VoteList komponensben helyesen jelenik-e meg a két tesztszavazat.

SelectedVote_WhenVoteClicked_ShouldShowVoteDetails

Egy adott szavazás (Id = 1) részletes megjelenítését teszteli a VotePage komponensben:

- A question mező validálása
- Ellenőrzi, hogy a kezdő és záró dátum megfelelően jelenik meg a DOM-ban.

VoteAdd_ShouldCallServiceAndUpdateUI

Ellenőrzi, hogy a VoteAdd komponens megfelelően meghívja a szavazás létrehozására szolgáló szolgáltatásokat:

- Egy új VoteViewModel példány kerül létrehozásra és megadott mezőkön keresztül kitöltésre,
- A felhasználói interakció szimulálása után a CreateVoteAsync és GetVotesAsync metódusok hívását verifikálja.

VotingSystem.Tests

Ez az osztály a VotesController API végpontjainak integrációs tesztjeit tartalmazza. A tesztek célja annak biztosítása, hogy a szavazásokkal kapcsolatos végpontok helyesen működjenek, megfelelően kezeljék a különböző bemeneti eseteket, és megfelelő HTTP státuszkódokat és válaszokat adjanak vissza.

Tesztelési környezet inicializálása

- In-Memory adatbázis a VotingSystemDbContext helyett.
- Szerepek (UserRole) és felhasználók (User) seed-elése a UserManager és RoleManager segítségével.
- Előre definiált szavazások, részvételi adatok, névtelen szavazatok beillesztése.

GET végpontok

GET /votes

Visszaadja az összes szavazást.

- Sikeres lekérdezés után:
    - 200 OK
    - Tartalom: List<VoteResponseDto>
    - Ellenőrzés: pontosan 4 szavazás.

GET /votes/active

Aktív szavazások lekérdezése (jelenlegi dátum alapján).

- 200 OK

- Ellenőrzés: pontosan 3 aktív szavazás.

GET /votes/closed

Lezárt szavazások lekérdezése.

- 200 OK
- Ellenőrzés: pontosan 1 lezárt szavazás.

GET /votes/{id}

Egy adott szavazás lekérdezése ID alapján.

- Ha létezik:
    - 200 OK
    - Ellenőrzés: ID és kérdés megegyezik.
- Ha nem létezik:
    - 404 Not Found
    - Válasz: ProblemDetails

GET /votes/{id}/results

Szavazás eredményének lekérdezése.

- Ha a szavazás létezik és lezárt:
    - 200 OK
    - Válasz: szavazatopciók és szavazatszám.
- Ha a szavazás még nyitott:
    - 400 Bad Request
    - Válasz: hibaüzenet.
- Ha nem létezik:
    - 404 Not Found
    - Válasz: ProblemDetails

GET /votes/voted/{voteId}/{userId}

Lekérdezi, hogy a megadott felhasználó részt vett-e az adott szavazáson.

- Ha a felhasználó nem létezik:
    - 200 OK
    - Válasz: false
- Ha a szavazás vagy felhasználó nem létezik:
    - 404 Not Found

- Válasz: ProblemDetails

POST végpontok – létrehozás, keresés

POST /votes

Új szavazás létrehozása.

- Ha érvénytelen az Options (pl. csak 1 opció):
  - 400 Bad Request
  - Válasz: ValidationProblemDetails
- Ha már létezik ugyanilyen kérdés:
  - 409 Conflict
  - Válasz: ProblemDetails
- Ha minden adat helyes:
  - 201 Created
  - Válasz: VoteResponseDto, ellenőrzés: kérdés és opciók egyeznek.

POST /votes/my

Lekérdezi a megadott felhasználó saját szavazásait.

- 200 OK
- Válasz: List<VoteResponseDto>
- Ellenőrzés: minden szavazásnál egyezik a User.Id.

POST /votes/search

Kérdés szövegrészlet alapján keres szavazásokat.

- Ha érvényes Sub:
  - 200 OK
  - Ellenőrzés: minden találat tartalmazza a Sub-ot.
- Ha Sub üres:
  - 400 Bad Request
  - Válasz: hibaüzenet: „Search term cannot be empty”

POST /votes/search-by-date

Keresés megadott dátumtartományon belül.

- 200 OK
- Válasz: List<VoteResponseDto>
- Ellenőrzés: minden szavazás kezdete ≥ Start, vége ≤ End.

Helper metódusok

- SeedVotes(): Előre meghatározott szavazások, részvételek, névtelen szavazatok beállítása.
- SeedUsers(): Három felhasználó létrehozása (johndoe, user2, teszt_elek).
- SeedRoles(): „Admin” szerepkör létrehozása.

Hitelesítés (Login metódus)

Minden teszt, amely hitelesítést igényel, a Login() metódussal lép be a user2@gmail.com felhasználóként.