Anonim szavazó rendszer

Modern webalkalmazások fejlesztése .NET környezetben

2024/2025 tavaszi

Készítette: Pataki Mátyás Balázs, DC88ON

Dátum: 2025.05.18

1. Fejlesztői adatok

Név: Pataki Mátyás Balázs

Neptun-kód: DC88ON

E-mail cím: pmatyi0402@gmail.com

Oktató: Dr. Cserép Máté

2. Feladatleírás

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).

3. Funkcionális elemzés

Fő felhasználói esetek (use cases):

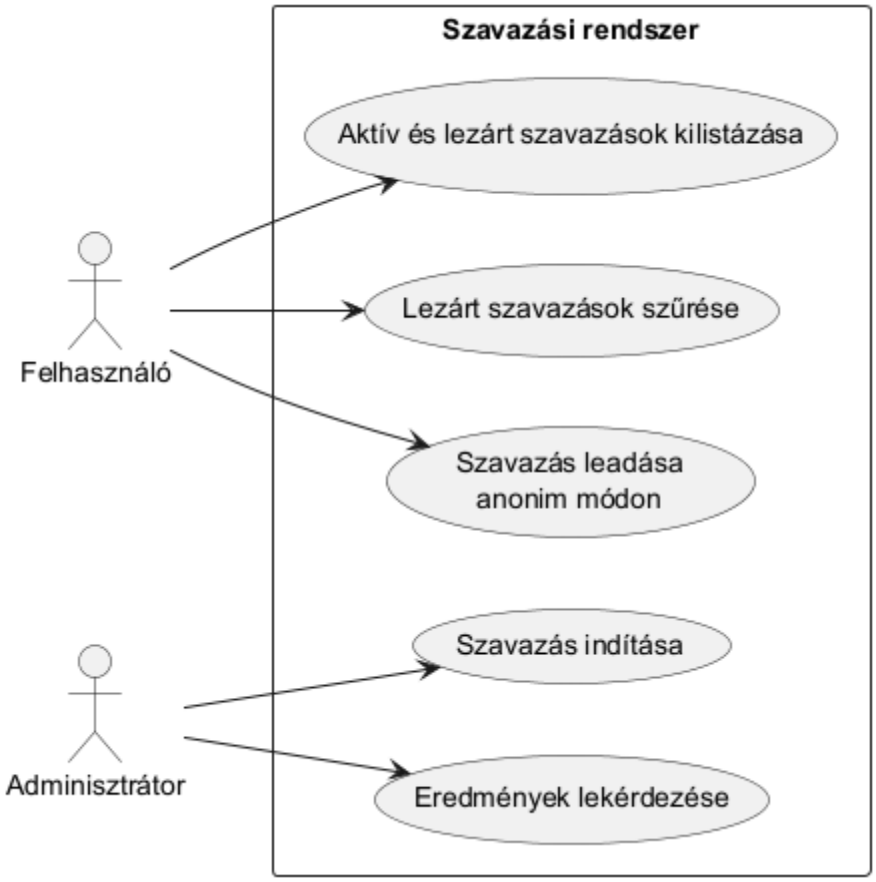
- Aktív és lezárt szavazások kilistázása

- Lezárt szavazások szűrése

- Szavazás leadása anonim módon

- Szavazás indítása (admin)

- Eredmények lekérdezése (admin)



Funkcionális követelmények listája

1. REST végpontok CRUD műveletekre

2. ASP.NET Identity alapú autentikáció/authorizáció

3. Entity Framework Core adatbázis-kezelés

4. Blazor admin felület

5. MVC architektúrának megfelelő WebAPI

6. ORM paradigmát követő adatbázis

Nem funkcionális követelmények:

- Biztonság (jelszó hasítás, jogosultságkezelés)

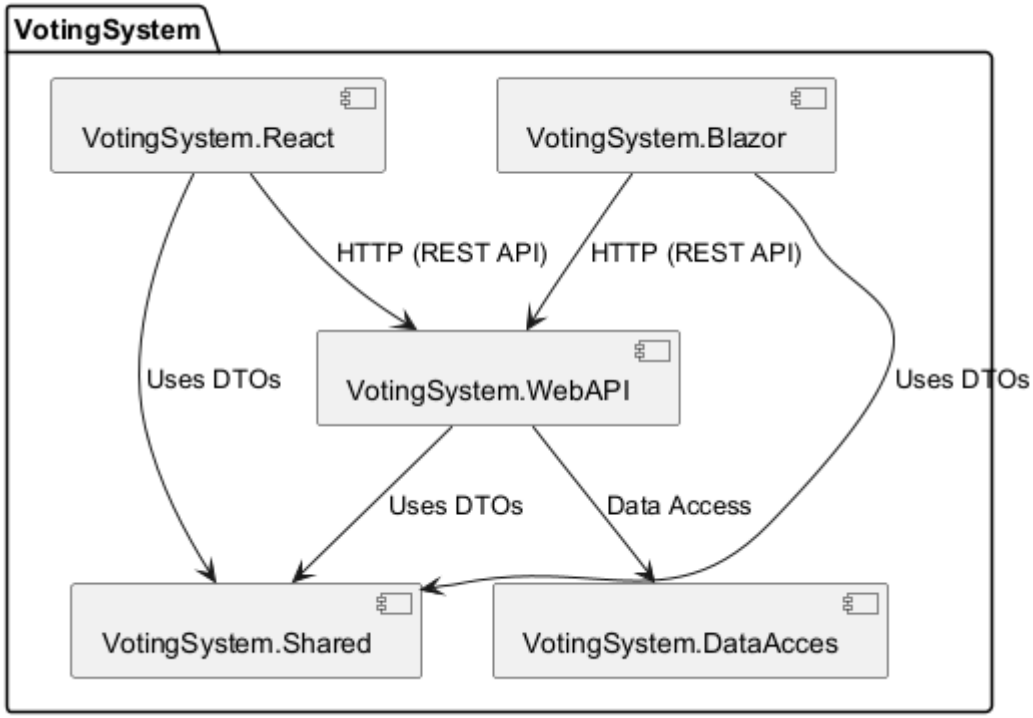
- Informatív felület

- Skálázhatóság, hibatűrés

4. Fejlesztői dokumentáció

4.1 Rendszer statikus terve

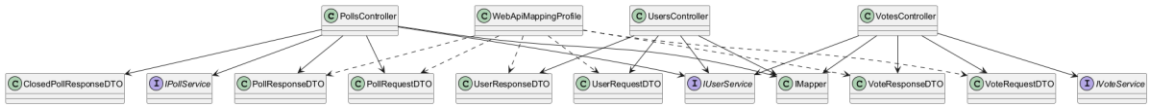
4.1.1 Komponens diagram



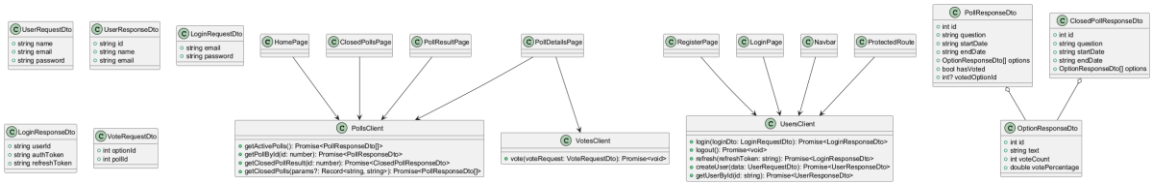
4.1.2 Osztálydiagrammok

-DataAccess:

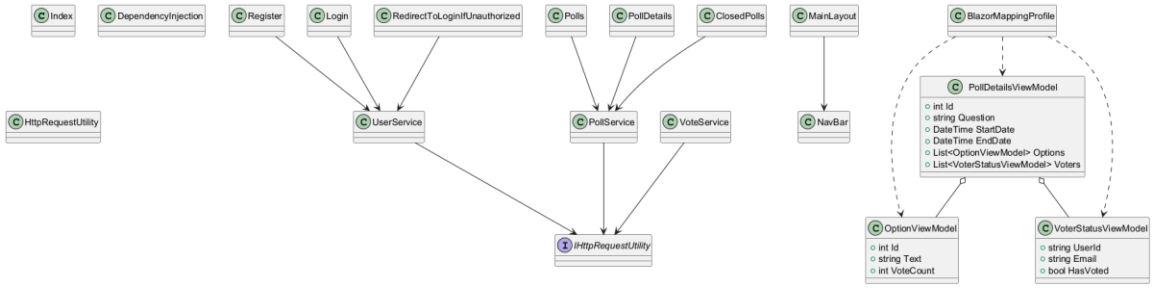
WebAPI:



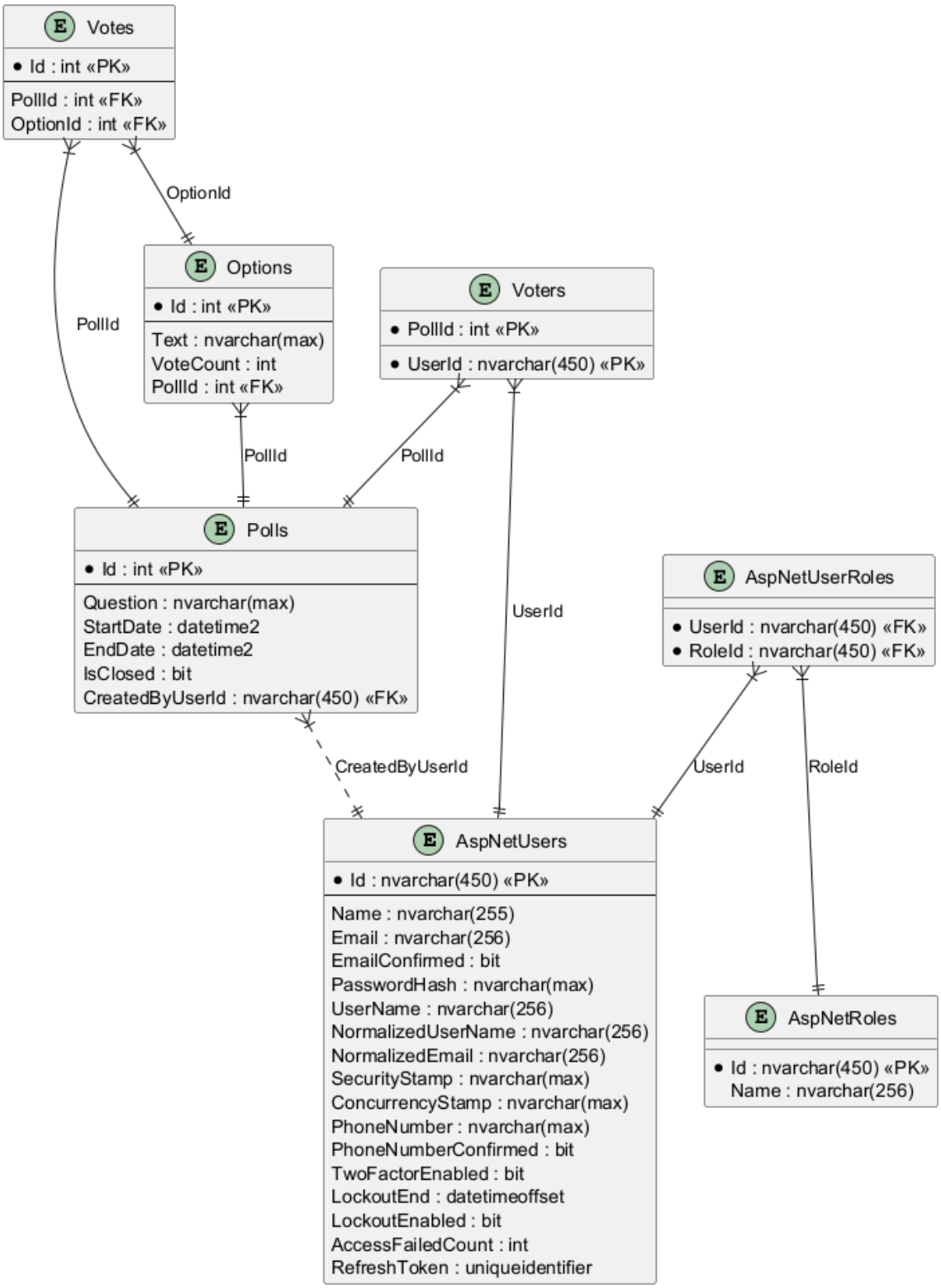
React (látogatói felület):



Blazor (adminisztrátori felület):



4.2 Adatbázis-felépítés



4.3 Webszolgáltatás felülete

HTTP metódus	URL	Jogosultság	Leírás
POST	/api/users/register	Anonim	Regisztrál egy új felhasználót. 201 – Sikeres regisztráció; 400 – Hibás adatok; 409 – Ütközés
POST	/api/users/login	Anonim	Hitelesítés (JWT + refresh token). 200 – Sikeres; 400 – Hibás kérés; 403 – Érvénytelen adatok
POST	/api/users/logout	Authenticated	Kijelentkezteti a felhasználót (refresh token érvénytelenítése). 204 – Sikeres; 401 – Nincs bejelentkezve
GET	/api/polls/{id}	Authenticated	Lekér egy szavazást ID alapján, tartalmazza a leadás státuszát; tulajdonosok látják a teljes szavazói listát. 200, 401, 404
GET	/api/polls/active	Authenticated	Aktív szavazások listája, minden elemhez HasVoted flag. 200, 401
GET	/api/polls/closed	Authenticated	Lezárt szavazások listája, szűrés szövegre és dátumokra. 200, 401
GET	/api/polls/closed/{id}	Authenticated	Részletes eredmény ID alapján; százalékos bontás opciókra. 200, 404
GET	/api/polls/mine	Authenticated	A bejelentkezett felhasználó által létrehozott szavazások listája. 200, 401
POST	/api/polls/create	Authenticated	Új szavazás létrehozása; opciók megadása. 200, 400, 401, 500

POST	/api/votes	Authenticated	Lead egy szavazatot PollId és OptionId alapján. 204, 400, 401

4.4 Tesztesetek leírása

4.4.1. Service-szintű tesztek

Cél: A PollService, UserService és VoteService üzleti logikájának, adatvalidációjának és exception-kezelésének ellenőrzése.

Megközelítés:

In-memory adatbázis (EF Core InMemory) használata az izolált, gyors futtatás érdekében.

Hibás bemenetek (pl. null/üres mezők, rossz dátumok, kevés opció) kivétel (ArgumentException/InvalidOperationException) dobását várjuk.

Sikeres esetben az adatbázis-módosítások ellenőrzése: új entitások mentése, rekordok száma.

Lekérdező metódusoknál (pl. GetActivePollsAsync, GetByIdAsync, GetClosedPollsAsync) visszaadott kollekciók, DTO-k tartalmának, méretének és szűrési feltételeknek való megfeleltetése.

Példa: CreatePollAsync_ThrowsArgumentException_WhenQuestionIsNullOrWhitespace – ha a kérdés nulla vagy üres, ArgumentException-t várunk.

Példa: GetActivePollsAsync_ReturnsOnlyCurrentlyActive – csak a jelenleg aktív (kezdő és záródátum között lévő) poll szerepel a listában.

4.4.2 Controller-szintű tesztek

Cél: A WebAPI-kontrollerek helyes HTTP-válaszainak és hívott service-metódusoknak az ellenőrzése, a ModelState és exception-kezelés tesztelése.

Megközelítés:

A kontroller konstruktorába injektált függőségek (IPollService, IUserService, IVoteService, IMapper) mockolása Moq-bal.

Különböző jogosultsági helyzetek (authenticated vs. anonymous) szimulálása a ControllerContext.HttpContext.User beállításával.

Hibás esetekben várt HTTP státuszok ellenőrzése (400 BadRequest, 404 NotFound, 409 Conflict).

Sikeres hívásnál OkObjectResult-ből visszakapott DTO mezőinek vizsgálata, illetve adatbázis-oldali műveletek mock verifikációja.

Példa: GetPollById_ReturnsOk_WithCorrectDto_ForOwner – a tulajdonos lekérdezésénél minden mező helyes értékkel tér vissza, beleértve a szavazói listát is.

Példa: CreatePoll_BadRequest_WhenServiceThrowsArgumentException – ha az service véletlenszerűen ArgumentException-t dob (pl. kevés opció), a controller 400-as hibát ad vissza.

4.4.3. Integrációs tesztek

Cél: A teljes HTTP-request → pipeline → EF Core InMemory DB → HTTP-válasz útvonal működésének ellenőrzése a tényleges alkalmazáshoston (ASP.NET Core TestServer/WebApplicationFactory).

Megközelítés:

WebApplicationFactory<Program> segítségével felépített tesztkörnyezet, ahol a production DB helyett InMemory adatbázist regisztrálunk.

A tesztadatbázis seedelése (alapvető Poll, User, Vote rekordok).

HTTP-kliens (HttpClient) használata valós API endpointok hívására: GET, POST, JSON body-val, Authorization header-rel.

JWT alapú autentikáció végigkövetése: /api/users/login, a visszakapott token beállítása a DefaultRequestHeaders.Authorization-ben.

Várható HTTP státuszok ellenőrzése, valamint JSON-beolvasás DTO-kba (ReadFromJsonAsync<T>()) és DTO-mezők validálása.

Példa: GetClosedPolls_ReturnsOnlyClosed_WhenAuthed – csak a zárt pollok listázása sikeres JWT tokennel.

Példa: CreatePoll_Unauthorized_Then_Succeeds – először 401-et kapunk, majd bejelentkezés után 200 OK, és a /api/polls/mine listában megjelenik az új poll.