Szoftvertechonlógia és technikák
II. Házi feladat

Rendszerterv

1. Alkalmazás leírása

1.1 Rendszer célja és működési környezete

A fejlesztendő szoftver egy asztali környezetben futó, webes működést szimuláló Szavazóplatform. A projekt elsődleges célja, hogy zárt közösségek (pl. iskolai osztályok, munkahelyi csoportok) számára biztosítson demokratikus, átlátható döntéshozatali lehetőséget szavazások formájában.

A rendszer architektúráját úgy alakítottuk ki, hogy az megfeleljen a hordozhatóság és az offline működés követelményeinek. A szoftver nem igényel telepített adatbázis-szervert (pl. SQL Server) vagy állandó internetkapcsolatot. Az adatokat strukturált, ember által is olvasható JSON állományokban tárolja a helyi fájlrendszerben. Ez a megoldás lehetővé teszi a rendszer egyszerű demonstrálását és a könnyű adatmozgatást.

1.2 Alkalmazott architektúra: MVVM

A rendszer tervezésekor a modern szoftverfejlesztésben, különösen a .NET/C# környezetben elterjedt MVVM (Model-View-ViewModel) tervezési mintát alkalmazzuk. Ez a megközelítés biztosítja a "Separation of Concerns" (a vonatkozások szétválasztása) elv érvényesülését, amely a rendszer modularitásának és tesztelhetőségének alapköve.

A rendszer logikailag négy, egymástól jól elkülönülő rétegre tagolódik:

- Model (Adatmodell): Ez a réteg reprezentálja a rendszer üzleti entitásait. Az itt található osztályok (pl. PollData, UserData, VotesData) tisztán adathordozó elemek (DTO - Data Transfer Objects), amelyek nem tartalmaznak üzleti logikát, kizárólag tulajdonságokat (properties). Például a PollData írja le egy szavazás szerkezetét (cím, leírás, határidő), míg az OptionData a válaszlehetőségeket.
- View (Nézet): A felhasználói felületért felelős réteg. A View csomagban található osztályok (pl. LoginView, ListPollsView, ModifyPollView) feladata kizárólag az adatok megjelenítése és a felhasználói interakciók (pl. gombnyomások, szövegbevitel) fogadása. A nézetek "buták", azaz nem végeznek adatfeldolgozást vagy döntéshozatalt; minden logikai kérést továbbítanak a hozzájuk rendelt ViewModel felé adatkötésen (Data Binding) keresztül.
- ViewModel (Nézetmodell): Ez a réteg a rendszer "motorja", amely összeköti a felhasználói felületet az üzleti logikával. A ViewModel-ek (pl. LoginViewModel, NewPollViewModel) felelősek a nézet állapotának kezeléséért (pl. töltésjelzők, hibaüzenetek megjelenítése), a felhasználói parancsok (Commands) fogadásáért és a

bemeneti adatok validálásáért. Minden ViewModel a közös BaseViewModel ősosztályból származik, amely biztosítja az alapvető funkcionalitást (pl. tulajdonságok változásának jelzése az UI felé).

- Services (Szolgáltatások): A háttérben futó üzleti logika és az adatperzisztencia helye. A Services réteg teljesen független a felhasználói felülettől. A PollServices és UserServices osztályok végzik a tényleges, komplex műveleteket (pl. szavazat érvényességének ellenőrzése, felhasználó beléptetése), és ők kommunikálnak az IDataService interfészen keresztül az adattároló réteggel.

1.3 Adatkezelés és Perzisztencia

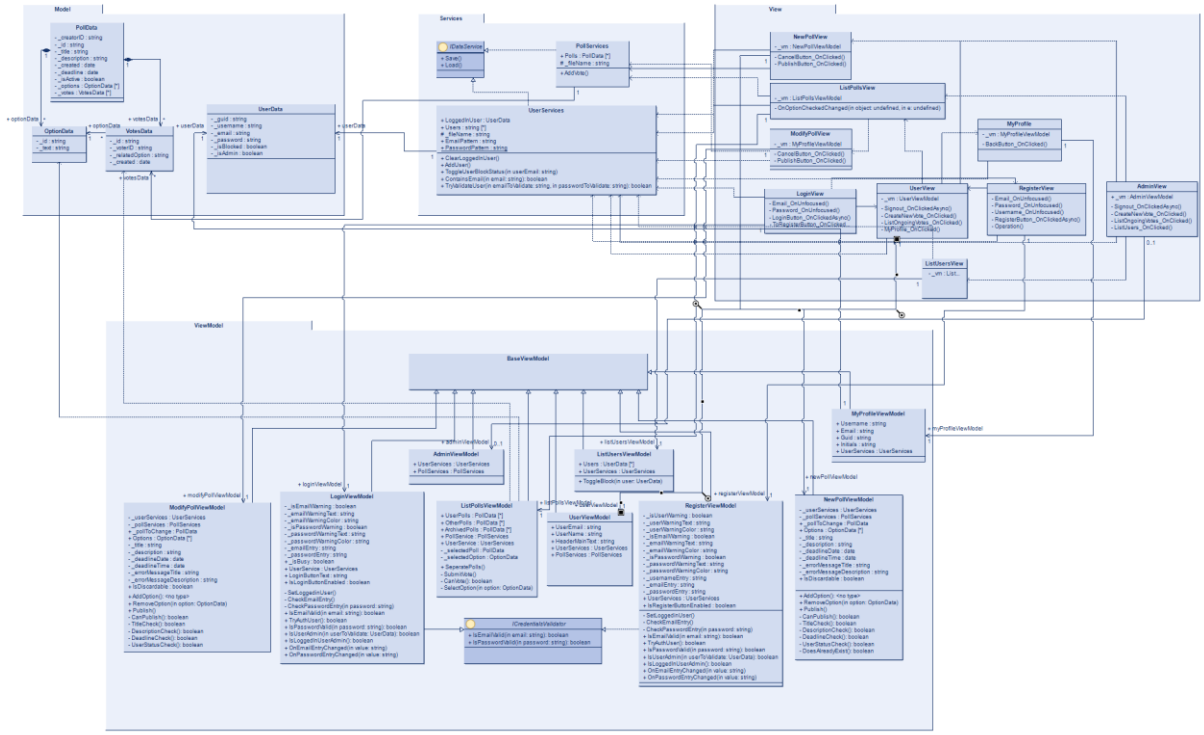
Az adatok tárolása JSON formátumban történik (users.json, polls.json). A rendszer induláskor az IDataService segítségével betölti az adatokat a memóriába, így a listázás és keresés gyorsan történik. A futás közbeni módosításokat (pl. új szavazás létrehozása, szavazat leadása) a szolgáltatások azonnal vagy időzítve mentik vissza a fájlokba, biztosítva az adatok megmaradását újraindítás után is.

1.4 Felhasználói folyamatok és jogosultságkezelés

A rendszer szigorú jogosultságkezelést valósít meg. Indításkor minden felhasználó a bejelentkezési képernyővel találkozik. A rendszer csak sikeres hitelesítés után enged hozzáférést a funkciókhoz.

- Adminisztrátor: Kiemelt jogkörrel rendelkezik. Jogosult a rendszer karbantartására, felhasználók letiltására (isBlocked flag kezelése) és a nem megfelelő szavazások moderálására. Számára az AdminView biztosít dedikált felületet.
- Felhasználó: A standard jogkör. Új szavazást hozhat létre, listázhatja a meglévőket és szavazhat. A rendszer biztosítja, hogy egy felhasználó egy szavazáson csak egyszer vehessen részt. Ezt a VotesData entitásban tárolt kapcsolatok ellenőrzésével valósítjuk meg: minden szavazatnál rögzítésre kerül a pollId és a userId párosa.

2. Osztálydiagram

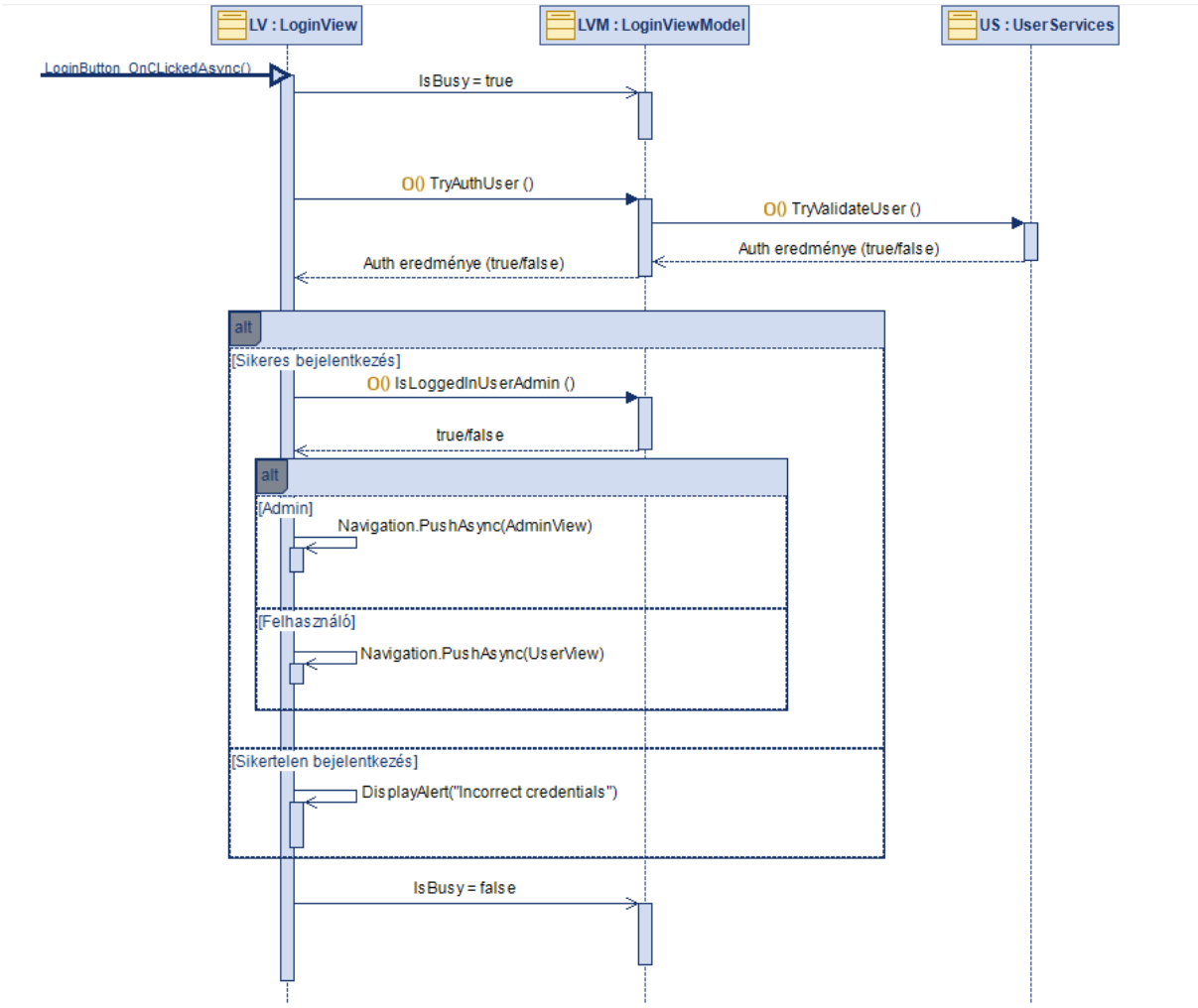


ábra 1: Osztálydiagram

A rendszer statikus felépítését az elkészült UML osztálydiagram szemlélteti. A diagram a modularitás elvét követve csomagokra (package) bontva mutatja a rendszer elemeit:

1. Model Csomag: Itt találhatók az egyszerű adatstruktúrák. Jól látható a kompozíció a PollData és OptionData között, ami azt jelzi, hogy egy szavazás elválaszthatatlan része a válaszopciók listája. A UserData tartalmazza a felhasználók azonosítóit és az isAdmin logikai változót, amely a jogosultságot szabályozza.
2. Services Csomag: A központi üzleti logikát tartalmazza. Az IDataService interfész definiálja az általános Save() és Load() műveleteket, amelyeket a konkrét implementáció valósít meg JSON alapokon. A PollServices osztály felelős a szavazásokkal kapcsolatos műveletekért (pl. AddVote(), CreatePoll()), míg a UserServices a felhasználók kezelését végzi (TryValidateUser(), CheckEmailEntry()).
3. ViewModel Csomag: Minden nézethez dedikált ViewModel tartozik (pl. ListPollsViewModel, RegisterViewModel), amelyek a közös BaseViewModel-ből származnak. Ez a struktúra biztosítja a kód újrahasznosíthatóságát (pl. az IsBusy tulajdonság mindenhol elérhető).
4. View Csomag: A konkrét képernyőket tartalmazza. A diagramon látható nyilak jelzik, hogy a View osztályok ismerik a hozzájuk tartozó ViewModel-t (pl. a LoginView használja a LoginViewModel-t), de a ViewModel-ek nem ismerik a View-kat, ami az MVVM minta alapvető szabálya.

3. Szekvenciadiagram



ábra 2: Szekvencia diagram

A mellékelt szekvenciadiagram a Bejelentkezési folyamatot (Login) ábrázolja, amely a rendszer belépési pontja és legfontosabb biztonsági kapuja. A folyamat lépései:

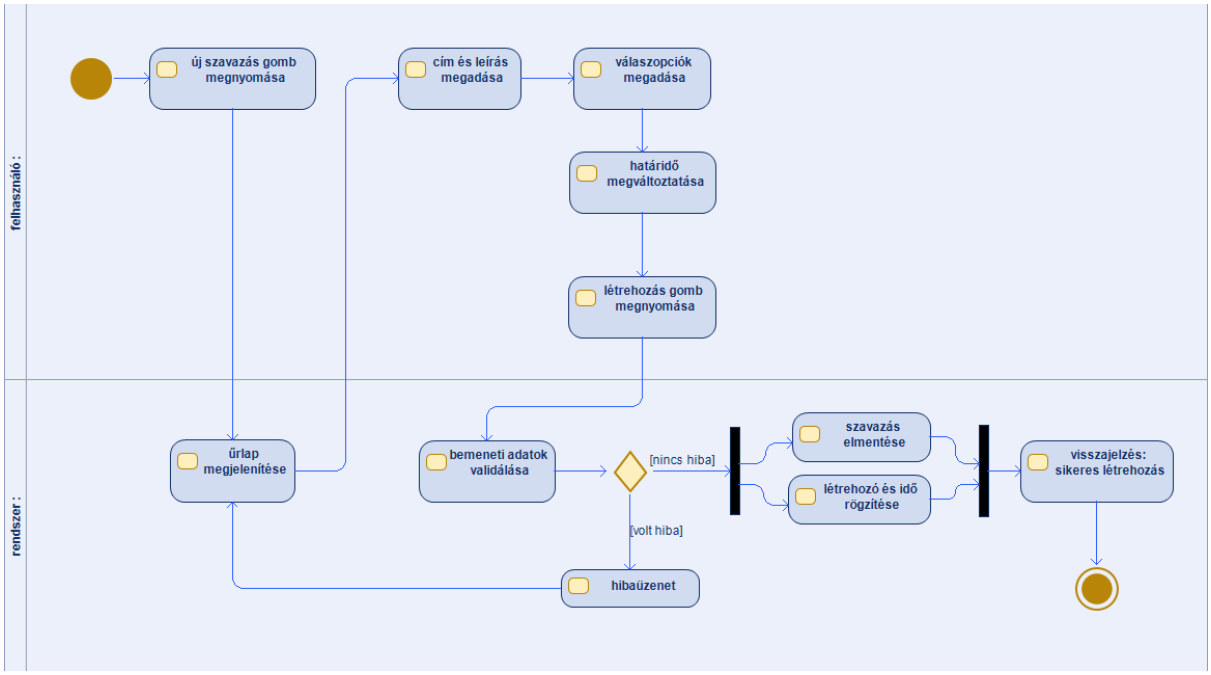
1. Felhasználói interakció: A felhasználó a LoginView felületén megnyomja a bejelentkezés gombot. Ez kiváltja a LoginButton_OnClickedAsync eseményt a "code-behind" fájlban.
2. Állapotváltozás: A vezérlés átkerül a LoginViewModel-hez (LVM). A ViewModel első lépésként IsBusy = true értékre állítja a státuszjelzőt, ami a felületen letiltja a gombokat és megjelenít egy töltésjelzőt (Spinner), jelezve a háttérmunka kezdetét.
3. Hitelesítési kérés: A ViewModel meghívja a saját TryAuthUser() metódusát.
4. Üzleti logika hívása: A kérés továbbítódik az alsóbb rétegbe, a UserServices-hez (US), ahol a TryValidateUser() függvény fut le. Ez a függvény ellenőrzi, hogy a megadott e-mail cím és jelszó páros létezik-e az adatbázisban.
5. Döntési pont (Alt blokk): A visszatérési érték alapján a folyamat elágazik:
    - Sikeres azonosítás (True): A rendszer lekérdezi a UserServices-től, hogy a bejelentkezett felhasználó adminisztrátor-e (IsLoggedInUserAdmin).

- Ha Igen: A navigációs szolgáltatás az AdminView-ra irányítja a felhasználót.
- Ha Nem: A navigációs szolgáltatás a UserView-ra irányítja a felhasználót.

- Sikertelen azonosítás (False): A rendszer egy felugró ablakban (DisplayAlert) hibaüzenetet jelenít meg: "Incorrect credentials".

6. Lezárás: A folyamat végén, függetlenül az eredménytől, az IsBusy jelző false értékre vált, visszaadva a vezérlést a felhasználónak.

4. Aktivitás diagram



ábra 3: Aktivitás diagram

A mellékelt aktivitás diagram az Új szavazás létrehozásának üzleti folyamatát (Business Process) mutatja be. A diagram "swimlane" (sávos) elrendezést alkalmaz, vizuálisan is elválasztva a Felhasználó és a Rendszer felelősségi köreit.

1. Kezdés és Adatbevitel (Felhasználó sáv): A folyamat a kezdőpontból indulva az "Új szavazás gomb megnyomásával" kezdődik. A rendszer megjeleníti az űrlapot. A felhasználó egymás utáni lépésekben megadja a szavazás címét, leírását, felviszi a válaszopciókat, és opcionálisan módosítja az alapértelmezett határidőt.
2. Kezdeményezés: Az adatok megadása után a felhasználó megnyomja a "Létrehozás" gombot. Ezen a ponton a vezérlés átkerül a Rendszer sávjába.
3. Validáció (Rendszer sáv): A rendszer első feladata a "Bemeneti adatok validálása". Ellenőrzi a specifikációban rögzített szabályokat:
   - A cím nem léehet üres.
   - Legalább két válaszopciónak lennie kell.
   - A határidőnek jövőbelinek kell lennie.

4. Elágazás (Decision Node):
    - [volt hiba]: Ha a validáció sikertelen, a rendszer hibaüzenetet generál, és visszavezeti a folyamatot az "űrlap megjelenítése" lépéshez, lehetővé téve a javítást.
    - [nincs hiba]: Ha az adatok helyesek, a folyamat párhuzamos ágra bomlik (Fork node). A rendszer egyidejűleg végzi el a "szavazás elmentését" az adatbázisba, valamint a metaadatok ("létrehozó és idő rögzítése") hozzáadását.
5. Befejezés: A párhuzamos folyamatok egyesülése (Join node) után a rendszer "visszajelzést" küld a sikeres létrehozásról (pl. felugró ablak vagy navigáció a listához), és a folyamat véget ér.

5. Modulok leírása

A rendszer a 'Separation of Concerns' elv alapján a következő logikai modulokból épül fel:

1. Hitelesítési és Felhasználókezelő Modul (Authentication Module)
    - Felelőssége: Ez a modul felel a belépési kapu védelméért. Kezeli a regisztrációt, a bejelentkezést, a jelszavak biztonságos ellenőrzését (ICredentialsValidator) és a jogosultsági szintek megállapítását. Biztosítja, hogy letiltott felhasználók ne férhessenek hozzá a rendszerhez.
    - Főbb osztályok: LoginViewModel, RegisterViewModel, UserServices, UserData.
    - Kapcsolatok: Szorosan együttműködik az Adatkezelő modullal a felhasználói adatok eléréséhez.
2. Szavazáskezelő Modul (Poll Management Module)
    - Felelőssége: A rendszer funkcionális magja. Kezeli a szavazások teljes életciklusát: létrehozás, szerkesztés (amíg nincs szavazat), törlés, lezárás. Felelős a szavazatok fogadásáért (AddVote), a duplikált szavazatok kiszűréséért és az eredmények valós idejű kalkulációjáért.
    - Főbb osztályok: ListPollsViewModel, NewPollViewModel, PollServices, PollData, VotesData.
    - Kapcsolatok: Felhasználja a Hitelesítési modul információit (ki a szavazó?), hogy a szavazatot a megfelelő felhasználóhoz kösse.
3. Adatperzisztencia Modul (Data Access Module)
    - Felelőssége: Az alacsony szintű fájlműveletek és az adatmodellek közötti híd. Ez a modul rejti el a fájlrendszerrel való kommunikáció technikai részleteit a felsőbb rétegek elől. Felelős a JSON szerializációért és deszerializációért.
    - Főbb osztályok: IDataService interfész és annak implementációja.
    - Kapcsolatok: Minden Service osztály ezen a modulon keresztül éri el az adatokat, így az üzleti logika független marad a fizikai tárolás módjától.