UNI MEET

Közösségépítő felület egyetemisták számára

Projekt labor BSC

Készítette: Kucska Boldizsár, Retki Levente, Földesi Attila

# BEVEZETÉS

## Mi is az az UNI MEET?

A "UNI MEET" egy innovatív közösségi platform, amelynek elsődleges célja az egyetemi hallgatók közötti valós, mély és támogató kapcsolatok kialakításának elősegítése. A felsőoktatásban töltött évek a személyes és szakmai fejlődés kulcsfontosságú időszakát jelentik, azonban a modern egyetemi környezetben a hallgatók egyre növekvő mértékben szembesülnek a magány, a szociális szorongás és a közösségi elszigeteltség kihívásaival. Ezen problémák nem csupán a hallgatók mentális és érzelmi jólétét veszélyeztetik, hanem negatívan befolyásolhatják tanulmányi eredményeiket és későbbi karrierkilátásaikat is.

A platform erre a komplex problémára kínál egy célzott, digitális megoldást, mert egy olyan biztonságos és inspiráló online teret hoz létre, ahol az azonos érdeklődésű, hasonló célokkal és értékrenddel rendelkező hallgatók könnyedén egymásra találhatnak. A rendszer lehetővé teszi a felhasználók számára, hogy érdeklődési körök, szakmai célok, vagy akár szabadidős tevékenységek mentén csoportokat hozzanak létre, eseményeket szervezzenek, és új baráti, illetve szakmai kapcsolatokat építsenek ki.

A projekt célcsoportját azok az egyetemi hallgatók alkotják, akik nyitottak a szociális életre, de nehézségekbe ütköznek a kapcsolatépítés terén. Ide tartoznak a beilleszkedési nehézségekkel küzdő elsőévesek, az elmagányosodott felsőbb évesek, valamint a nemzetközi hallgatók, akik új kulturális és szociális környezetben keresik a helyüket.

A "UNI MEET" nem csupán egy újabb közösségi média alkalmazás; egy olyan célirányos eszköz, amely a felszínes online interakciók helyett a mély, személyes kapcsolatok kialakítását helyezi a középpontba. Hisszük, hogy a platform hozzájárul a hallgatói lemorzsolódás csökkentéséhez, a mentális egészség javításához, és egy olyan támogató, dinamikus egyetemi közösség létrejöttéhez, amely a diploma megszerzése után is értékes erőforrást jelent a végzett hallgatók számára. A jelen dokumentum részletesen bemutatja a projekt hátterét, célkitűzéseit, a célközönség igényeit, valamint a platform által kínált innovatív megoldásokat.

## A Probléma Mélysége: A Modern Egyetemi Lét Szociális Kihívásai

Az egyetemi évek a közhiedelemmel ellentétben nem mindenki számára jelentenek felhőtlen szórakozást és pezsgő társasági életet. A felsőoktatási intézményekbe való bekerüléssel a fiatal felnőttek egy teljesen új, gyakran ijesztő életszakaszba lépnek. A megnövekedett tanulmányi terhek, az otthontól való elszakadás és az új szociális környezethez való alkalmazkodás kényszere komoly pszichés terhet róhat a hallgatókra. A statisztikák és kutatások egyértelműen alátámasztják, hogy a magány és a szociális szorongás globális problémát jelent az egyetemi populáció körében.

### Az Elsőévesek Beilleszkedési Nehézségei és a Szociális Szorongás

Az elsőéves hallgatók, a "gólyák", különösen sérülékeny csoportot alkotnak. A középiskolából egy jóval nagyobb és anonimabb közegbe kerülve gyakran érzik magukat elveszettnek és magányosnak. A korábbi, jól bevált baráti körök felbomlanak, és az új kapcsolatok kialakítása komoly kihívást jelenthet. A szociális szorongás, azaz az attól való félelem, hogy mások negatívan ítélik meg őket, gátolhatja a kezdeményezőkészséget, és tovább mélyítheti az elszigeteltséget. Kutatások szerint a hallgatók jelentős része számol be arról, hogy az első félév során intenzív magányt és honvágyat él meg, ami negatívan hat a tanulmányi teljesítményükre és az általános jóllétükre.

### A Felsőbb Éves Hallgatók Elmagányosodása és a Közösségi Hiány

A probléma nem korlátozódik az elsőévesekre. A felsőbb éves hallgatók körében is gyakori a közösségi kapcsolatok hiánya. A kezdeti, gólyatáborokban és első hetekben kialakult felszínes ismeretségek gyakran nem mélyülnek el, a tanulmányi terhek növekedésével pedig egyre kevesebb idő és energia jut a szociális életre. A szakmai gyakorlatok, a diákmunka és az egyéni kutatási projektek tovább aprózhatják az időbeosztást, és a hallgatók könnyen elszigetelődhetnek egymástól. Ez a jelenség különösen azokat érinti, akik nem a kollégiumban, hanem albérletben, esetleg egyedül élnek.

### A Nemzetközi Hallgatók Speciális Igényei

A nemzetközi hallgatók helyzete még összetettebb. Nekik nem csupán az új oktatási rendszerhez, hanem egy idegen kultúrához, nyelvhez és társadalmi normákhoz is alkalmazkodniuk kell. A nyelvi akadályok és a kulturális különbségek megnehezíthetik a helyi hallgatókkal való kapcsolatfelvételt, ami mély magányhoz és elszigeteltséghez vezethet. Számukra egy olyan platform, amely segít hasonló hátterű vagy érdeklődésű hallgatókat találni, létfontosságú lehet a sikeres integrációhoz és a pozitív egyetemi élményhez.

# Célok, Küldetés és Innovatív Megközelítés

Az "UNI MEET" platform létrehozásának alapvető motivációja, hogy választ adjon a modern egyetemi életben tapasztalható, fentebb részletezett szociális kihívásokra. Célunk egy olyan digitális ökoszisztéma megteremtése, amely proaktívan támogatja a hallgatókat a közösségi hálójuk kiépítésében és megerősítésében.

## A Platform Víziója és Missziója

* **Vízió:** Egy olyan egyetemi világ megteremtése, ahol minden hallgató – származásától, szakától és személyiségétől függetlenül – egy támogató és inspiráló közösség aktív tagjának érezheti magát.
* **Misszió:** Az "UNI MEET" küldetése, hogy egy intelligens, felhasználóbarát és biztonságos digitális platformot biztosítson, amely összeköti az azonos érdeklődésű hallgatókat, elősegíti a valós idejű interakciókat, és katalizátorként működik a tartós baráti és szakmai kapcsolatok kialakulásában.

## Az "UNI MEET" Célkitűzései

A platform fejlesztése során az alábbi konkrét, mérhető célokat tűztük ki:

* **A kapcsolatépítés megkönnyítése:** Egy olyan intelligens algoritmus fejlesztése, amely a felhasználók által megadott érdeklődési körök, szakmai célok, tanulmányok és akár személyiségjegyek alapján releváns kapcsolatokat és csoportokat ajánl.
* **A közösségi aktivitás ösztönzése:** Egy integrált eseménynaptár és csoportkezelő funkció létrehozása, amely lehetővé teszi a felhasználók számára, hogy egyszerűen szervezzenek és csatlakozzanak különböző programokhoz, legyen szó egy közös tanulásról, egy sporteseményről, egy kulturális programról vagy egy szakmai workshopról.
* **Biztonságos és inkluzív környezet biztosítása:** Szigorú közösségi irányelvek és hatékony moderációs eszközök bevezetése a zaklatás, a diszkrimináció és a visszaélések megelőzése érdekében. Minden felhasználó számára biztosítani kell a lehetőséget, hogy biztonságban és elfogadva érezze magát.
* **A szakmai és társasági élet integrálása:** Lehetőséget teremteni a hallgatók számára, hogy ne csak baráti, hanem szakmai kapcsolatokat is építsenek. A platformon helyet kapnának szakmai csoportok, mentorprogramok és vállalati partnerek által szervezett események is.
* **A hallgatói jólét javítása:** A platform hosszú távú célja, hogy a közösségi kapcsolatok erősítésén keresztül hozzájáruljon a hallgatók mentális egészségének javulásához, a szorongás és a depresszió tüneteinek enyhítéséhez, és végső soron a lemorzsolódási arány csökkentéséhez.

## Innovatív Megközelítés a Közösségépítésben

Az "UNI MEET" több ponton is eltér a hagyományos közösségi média platformoktól:

* **Célirányosság:** Míg a globális platformok a lehető legszélesebb közönséget célozzák, az "UNI MEET" kifejezetten az egyetemi mikroközösségekre fókuszál.
* **Érdeklődés-alapú szerveződés:** A platform alapját nem a már meglévő ismerősök hálózata, hanem az érdeklődési körök és közös célok mentén történő szerveződés adja.
* **Offline interakciók ösztönzése:** Az alkalmazás elsődleges célja nem az, hogy a felhasználókat a képernyő előtt tartsa, hanem hogy hidat képezzen a digitális és a fizikai világ között, és valós találkozók létrejöttét ösztönözze.
* **Gondozott tartalom és közösség:** A platform aktívan támogatja a pozitív és konstruktív interakciókat, és szűri a káros, irreleváns tartalmakat.

# 4. Célközönség: Kiknek Készül az "UNI MEET"?

A platform sikerének kulcsa a célközönség igényeinek mély megértése és precíz kiszolgálása. Az "UNI MEET" az egyetemi hallgatók széles spektrumát célozza meg, de különös figyelmet fordít azokra a csoportokra, amelyek a leginkább ki vannak téve a szociális elszigeteltség veszélyének.

# 5. A versenytársak

A digitális közösségi platformok piaca telített, azonban az egyetemi hallgatók specifikus igényeit célzó, jól megtervezett alkalmazások számára jelentős piaci rés kínálkozik. A siker kulcsa a differenciálódás és az egyedi értékajánlat megfogalmazása.

## Meglévő Platformok és Alkalmazások Elemzése

* **Facebook:** A legnagyobb közösségi hálózat, amelyen szinte minden egyetemistának van profilja. Az egyetemi csoportok (évfolyam-, szakcsoportok) fontos információs csatornaként működnek, de a valódi közösségépítésre korlátozottan alkalmasak. A hírfolyam algoritmusa, a reklámok és az álhírek elvonják a figyelmet a lényegről, a csoportok pedig gyakran átláthatatlanok és személytelenek.
* **Instagram & TikTok:** Ezek a platformok elsősorban a vizuális tartalommegosztásra és a szórakoztatásra fókuszálnak. Bár lehetővé teszik a kapcsolatfelvételt, nem támogatják a célzott, érdeklődés-alapú közösségszervezést. A hangsúly az önreprezentáción és a felszínes interakciókon van.
* **Meetup:** Egy népszerű eseményszervező platform, amely érdeklődési körök mentén szervez közösségeket. Bár a koncepciója hasonló az "UNI MEET" eseményszervező funkciójához, a Meetup nem kifejezetten egyetemi hallgatókra specializálódott, és hiányoznak belőle az egyetemi élethez kapcsolódó specifikus funkciók (pl. tanulócsoportok, szakmai mentorálás).
* **Egyetemi Intranetes Rendszerek (pl. Neptun, CooSpace):** Ezek a platformok az adminisztrációra és az oktatásra fókuszálnak. Bár tartalmazhatnak fórumokat, a felhasználói felületük gyakran elavult, és nem ösztönzik a hallgatók közötti informális interakciókat.

# A Projekt Jelentősége és Várható Hatásai

Az "UNI MEET" platform létrehozása túlmutat egy újabb mobilalkalmazás fejlesztésén. A projekt egy valós és egyre súlyosbodó társadalmi problémára kíván megoldást nyújtani, és hosszú távon pozitív hatást gyakorolhat a hallgatói populáció egészére, valamint a felsőoktatási intézmények közösségi életére.

## Hatás a Hallgatói Jólétre és Mentális Egészségre

A támogató közösségi háló a mentális egészség egyik legfontosabb védőfaktora. Azáltal, hogy az "UNI MEET" segít a hallgatóknak barátokat találni és egy közösséghez tartozni, közvetlenül hozzájárul a magány, a szorongás és a depresszió csökkentéséhez. Egy erős szociális háló puffert képez a stresszel szemben, növeli a rezilienciát, és javítja az általános élettel való elégedettséget.

## Hozzájárulás a Szakmai Fejlődéshez és a Hálózatépítéshez

A platform nem csupán a baráti, hanem a szakmai kapcsolatok kialakítását is támogatja. A közös projekteken dolgozó hallgatók, a tanulócsoportok tagjai és a szakmai érdeklődésű közösségek mind hozzájárulnak a résztvevők szakmai fejlődéséhez. Az itt kialakított kapcsolatok a későbbiekben értékes szakmai hálózat alapját képezhetik, segítve a pályakezdést és a karrierépítést.

## Az Egyetemi Közösség Erősítése

Az "UNI MEET" platform az egyetemek számára is értékes eszközzé válhat. Segítségével az intézmények hatékonyabban kommunikálhatnak a hallgatókkal, támogathatják a hallgatói szervezeteket, és egy élénkebb, integráltabb kampusz-életet hozhatnak létre. Egy összetartóbb hallgatói közösség növeli az intézmény vonzerejét, javítja a hallgatói elégedettséget és csökkenti a lemorzsolódási arányt.

## Hosszú Távú Perspektívák

Hosszú távon az "UNI MEET" egy olyan dinamikus, önszerveződő közösségi hálót hozhat létre, amely az egyetemi évek alatt és azt követően is támogatja a tagjait. A platformon létrejövő kapcsolatok, együttműködések és barátságok a hallgatók személyes és szakmai életének meghatározó részévé válhatnak, hozzájárulva ahhoz, hogy az egyetemi évek valóban életük egyik legértékesebb és legmeghatározóbb időszakává váljanak. A projekt sikeres bevezetése után a modell más egyetemekre és városokra is kiterjeszthető, létrehozva egy országos, sőt akár nemzetközi hálózatot a közösségi élményre nyitott egyetemisták számára.

# TERVEZÉS

## A rendszer tervezésének menete

A „UNI MEET" alkalmazás célja, hogy egyetemi hallgatók számára olyan digitális közösségi platformot biztosítson, amely elősegíti a valós, mély emberi kapcsolatok kialakítását, és támogatja a hallgatók integrációját az egyetemi életbe. A rendszer tervezése ennek a célkitűzésnek megfelelően egy modern, biztonságos és skálázható webes architektúra mentén történt. A fejlesztés alapját a Microsoft [ASP.NET](https://asp.net/) Core Web API technológia képezi, amely lehetővé teszi a korszerű, rétegezett alkalmazásstruktúra megvalósítását.

## Tervezési alapelvek és architekturális megközelítés

A fejlesztés során a réteges architektúra (layered architecture) elvét alkalmaztuk, amely biztosítja a felelősségek egyértelmű szétválasztását és a kód könnyű karbantarthatóságát. A rendszert három fő logikai réteg köré építettük, amelyek világosan elkülönítik az adatkezelést, az üzleti logikát és a megjelenítést.

## A megoldás két fő komponense:

### UniMeet (Prezentációs réteg)

Ez a fő [ASP.NET](https://asp.net/) Core projekt, amely az alkalmazás belépési pontja és a HTTP-alapú kommunikációért felel. Feladatai közé tartozik:

* HTTP kérések kezelése és az API végpontok definiálása
* A vezérlők (Controllers) működtetése
* A middleware-ek (pl. autentikáció, hibakezelés, naplózás) beállítása és a függőségek konfigurálása
* A webkiszolgáló hosztolása

A réteg nem tartalmaz közvetlen adatbázis-hivatkozásokat; minden adatlekérés és módosítás az AppDbContext-en keresztül történik, amelyet Dependency Injection (DI) segítségével juttatunk be az egyes vezérlőkbe.

### UniMeet.DataContext (Adatelérési réteg)

Ez egy különálló osztálykönyvtár, amely kizárólag az adatbázis-kezeléssel kapcsolatos műveleteket tartalmazza. A réteg az Entity Framework Core ORM-re épül, amely lehetővé teszi, hogy az adatbázis rekordjai C# objektumként kezelhetők legyenek.

A rétegben található:

* Az Entity Framework Core DbContext (AppDbContext), amely a fő kapcsolatot biztosítja az adatbázissal
* A modell osztályok (User, Post, Comment stb.)
* Az adatbázis-kapcsolatok és relációk konfigurációja (pl. egy felhasználóhoz több bejegyzés tartozhat)

A DbContext egyben Repository és Unit of Work szerepet is betölt, vagyis központi pontként kezeli az összes adatbázis-tranzakciót.

### Üzleti logikai réteg (tervezett bővítés)

A jövőbeni fejlesztés célja egy külön Service Layer bevezetése, amely az üzleti szabályokat és feldolgozási logikát választja el a prezentációs rétegtől. Ezáltal a vezérlők kizárólag az adatáramlásért, nem pedig az üzleti döntésekért felelnének. Ez a megközelítés még inkább növeli a tesztelhetőséget és újrafelhasználhatóságot.

A logikai rétegek szétválasztása biztosítja, hogy a prezentációs réteg független maradjon az adattárolási mechanizmusoktól, így a rendszer rugalmasan bővíthető és könnyen tesztelhető.

## Kulcsfontosságú technológiák és tervezési minták

### RESTful API

Az alkalmazás funkcionalitása REST architektúrán alapul, amely az interneten elterjedt szabványos kommunikációs modellt alkalmazza. Az erőforrásokhoz (felhasználók, bejegyzések, kommentek) HTTP metódusokon keresztül férünk hozzá (GET, POST, PUT, DELETE). A vezérlőkben (ApiController, Route, [HttpPost] attribútumok) definiált végpontok egységes, jól dokumentált felületet biztosítanak a kliensalkalmazások számára. Ez a megközelítés biztosítja az interoperabilitást, azaz hogy bármilyen kliens (web, mobil, desktop) képes legyen a szerverrel kommunikálni.

### Dependency Injection (DI)

Az [ASP.NET](https://asp.net/) Core beépített függőséginjektálási mechanizmusát használja a rendszer, amely a modern szoftverfejlesztés egyik alapvető tervezési mintája. A DI lényege, hogy az objektumok nem példányosítják saját függőségeiket, hanem azokat kívülről kapják meg, jellemzően konstruktor paramétereken keresztül.

### A DI működése az UNI MEET alkalmazásban

A rendszerben az alkalmazás indulásakor regisztráljuk a szolgáltatásokat a DI konténerbe. Itt határozzuk meg, hogy mely osztályok legyenek elérhetők az alkalmazás különböző részei számára, valamint azt is, hogy ezek milyen életciklussal rendelkezzenek. Az adatbázis kontextus (AppDbContext), a jelszókezelő segédosztály (PasswordHelper), valamint a jövőbeli szolgáltatások (például e-mail küldés, fájlkezelés) mind itt kerülnek beállításra.

A vezérlőkben ezután egyszerűen hivatkozhatunk ezekre a szolgáltatásokra konstruktor paramétereken keresztül, anélkül hogy közvetlenül példányosítanánk őket. Az [ASP.NET](https://asp.net/) Core automatikusan feloldja a függőségeket és beinjektálja a szükséges objektumokat.

### Szolgáltatás életciklusok

Az [ASP.NET](https://asp.net/) Core három különböző életciklust támogat a szolgáltatások számára, amelyek meghatározzák, hogy mikor és hogyan jönnek létre az objektumpéldányok:

#### Transient:

Minden alkalommal új példány jön létre, amikor a szolgáltatást kérik. Ideális állapot nélküli, könnyű szolgáltatásokhoz, mint például validációs osztályok vagy segédeszközök, amelyek nem tárolnak állapotot a műveletek között.

#### Scoped:

HTTP kérésenkénti egy példány jön létre. Az adatbázis kontextust (AppDbContext) általában scoped életciklussal regisztráljuk, így biztosítva, hogy egy kérés során ugyanazt a kontextust használjuk a tranzakciók konzisztenciája érdekében, de kérések között ne alakuljanak ki memóriaszivárgások. Ez az életciklus garantálja, hogy a kérés összes műveletét ugyanaz a kontextus példány szolgálja ki.

#### Singleton:

Az alkalmazás teljes életciklusa alatt egyetlen példány létezik, amely az első használatkor jön létre és az alkalmazás leállásáig él. Konfigurációs szolgáltatásokhoz, cache-ekhez, vagy olyan osztályokhoz használatos, amelyek költségesek a létrehozásban és biztonságosan megoszthatók az összes kérés között.

### A DI előnyei a projektben

**Tesztelhetőség:**

Mivel a vezérlők nem függenek közvetlenül a konkrét implementációktól, hanem interfészeken vagy absztrakt osztályokon keresztül hivatkoznak rájuk, könnyen helyettesíthetők mock objektumokkal unit tesztek során. Például egy teszt során az AppDbContext helyett egy in-memory adatbázist használhatunk, amely gyorsabb és nem igényel valódi adatbázis-kapcsolatot. Ez lehetővé teszi az automatizált tesztek hatékony futtatását.

**Kód összefonódásának csökkenése (Loose Coupling):**

Az egyes komponensek csak interfészeken vagy absztrakciós rétegeken keresztül kommunikálnak, így a változások lokalizáltak maradnak. Ha például az adatbázist SQL Serverről PostgreSQL-re vagy MySQL-re szeretnénk váltani, elegendő az adatelérési réteg konfigurációját módosítani, a vezérlők és az üzleti logika érintetlen marad. Ez jelentősen csökkenti a refaktorálás költségeit és kockázatait.

**Újrafelhasználhatóság:**

A szolgáltatások több vezérlőben és más komponensekben is felhasználhatók anélkül, hogy újra kellene őket implementálni vagy duplikálni kellene a kódot. Például a PasswordHelper osztály minden olyan helyen használható, ahol jelszókezelésre van szükség, legyen az regisztráció, bejelentkezés vagy jelszó-visszaállítás.

**Karbantarthatóság:**

Az implementációk egyszerűen cserélhetők anélkül, hogy a felhasználó kódját módosítani kellene. Ha például új naplózási megoldást szeretnénk bevezetni (pl. áttérés konzol alapú naplózásról Serilog vagy NLog használatára), elegendő az új implementációt regisztrálni a DI konténerben, és az összes helyen automatikusan az új megoldás lesz használva.

**Központi konfiguráció:**

A szolgáltatások regisztrációja egy helyen történik, ami átláthatóságot és könnyű menedzsmentöt biztosít. Egyértelműen látható, hogy az alkalmazás milyen függőségekkel rendelkezik.

### Entity Framework Core (EF Core)

Az adatelérés az EF Core segítségével történik, amely objektum-relációs leképezést (ORM) biztosít. A LINQ-alapú lekérdezéseknek köszönhetően az adatbázisműveletek C# objektumokon és LINQ lekérdezéseken keresztül valósulnak meg, a háttérben SQL Server adatbázis használatával. A fejlesztő SQL helyett magasabb absztrakciós szinten dolgozhat, a keretrendszer pedig automatikusan generálja a megfelelő SQL-parancsokat. Ez gyorsabb fejlesztést és kisebb hibalehetőséget biztosít, továbbá könnyebbé teszi az adatbázis-változtatások nyomon követését migrációk segítségével.

### Data Transfer Object-ek (DTO-k)

Az API adatkommunikációja DTO-kon (pl. RegisterUserDto, LoginUserDto) keresztül történik. A DTO-k célja, hogy elválasszák a belső adatmodellt az API által szolgáltatott struktúrától.

**Ez a megközelítés:**

* Növeli az adatbiztonságot (nem engedélyez túlposztolást, illetve a kliens nem küldhet több adatot, mint ami engedélyezett)
* Elválasztja a belső adatmodellt a külső API-szerkezettől
* Csökkenti az adatforgalmat a kliens és a szerver között kisebb méretű adatcsomagok biztosításával
* Függetleníti az API-t az adatbázis szerkezetének változásától

### Repository-szerű adatkezelés

Bár formálisan nincs külön Repository interfész, az EF Core DbContext gyakorlatilag betölti ezt a szerepet, repository-szerűen centralizálva az adatkezelést. Így az adatműveletek központilag kezeltek, biztosítva az egységes logikát és konzisztenciát. Ez az elv a Single Responsibility Principle betartását is segíti: a vezérlők csak a folyamatirányításért felelősek, az adatkezelés kizárólag a DbContext feladata.

## API tervezése és biztonság

Az API vezérlői tartalmazzák az alkalmazás fő üzleti logikáját. Például a UsersController a regisztráció és bejelentkezés funkciókat valósítja meg:

* Ellenőrzi az űrlapadatok helyességét
* Megvizsgálja, hogy a felhasználó már létezik-e
* Érvényesíti az e-mail cím doménjét (csak egyetemi e-mail-címeket fogad el, ezáltal garantálva, hogy valódi hallgatók használják a szolgáltatást)
* Jelszót hashel és sóz a PasswordHelper osztály segítségével
* Végül menti az új entitást az adatbázisba

### Biztonsági és adatvédelmi megfontolások

Mivel az alkalmazás felhasználói adatokat (pl. e-mail-cím, jelszó, érdeklődési kör) kezel, a biztonság kiemelt szempont volt a tervezés során.

A következő megoldásokat vezettük be:

**Jelszókezelés:**

A jelszavakat soha nem kerülnek tárolásra nyers formában, kizárólag biztonságos, hash-elt és sózott formában a PasswordHelper osztály segítségével. Így még adatbázis-szivárgás esetén sem visszafejthetők az eredeti jelszavak. Ezzel az alkalmazás megfelel az alapvető információbiztonsági követelményeknek.

**Adatvalidálás:**

A bejövő adatok ellenőrzése minden vezérlőben megtörténik, megelőzve a hibás vagy rosszindulatú bevitelt.

**Swagger/OpenAPI integráció:**

A dokumentálhatóság érdekében a projekt a Swashbuckle.AspNetCore (Swagger/OpenAPI) csomagot is tartalmazza, amely automatikusan generálja az API végpontok leírását és tesztfelületét. Ez nagyban segíti a frontend fejlesztőket és az integrációs munkát, valamint átláthatóságot és auditálhatóságot is biztosít.

**A fenti megoldások összességében megfelelnek a GDPR alapelveinek:** adatminimalizálás, célhoz kötött adatkezelés és biztonságos tárolás.

## Tesztelhetőség és karbantarthatóság

A rendszer komponensei laza csatolással kommunikálnak egymással, ezért az egyes rétegek külön-külön is tesztelhetők. Például a UsersController működése validálható anélkül, hogy valódi adatbázisra lenne szükség, mivel az AppDbContext helyettesíthető egy mock implementációval.

**A tervezésnél a SOLID elveket követtük:**

* Single Responsibility Principle:
  + minden osztály egy jól meghatározott feladatért felel
* Open/Closed Principle:
  + új funkciók hozzáadhatók anélkül, hogy a meglévő kódot módosítani kellene
* Dependency Inversion Principle:
  + a magas szintű modulok nem függnek a konkrét implementációktól, csak absztrakcióktól

Ez a szemlélet biztosítja, hogy a rendszer hosszú távon is fejleszthető és karbantartható maradjon, ami kulcsfontosságú egy közösségi alkalmazás esetében, ahol a funkciók gyorsan bővülnek (pl. chat, eseménynaptár, üzenetküldés).

## Jövőbeli bővítési lehetőségek

A jelenlegi architektúra előkészíti a terepet több lehetséges fejlesztési irányhoz:

Frontend integráció: Az API-t egy Angular vagy React alapú webes klienssel, illetve mobilalkalmazással (pl. Flutter, React Native) lehet összekötni.

Szerveroldali autentikáció (JWT): A következő verzióban tervezett token-alapú hitelesítés még biztonságosabb hozzáférés-kezelést biztosít, lehetővé téve a stateless autentikációt és a horizontális skálázhatóságot.

Push értesítések és valós idejű kommunikáció: A SignalR technológia integrálásával a felhasználók valós idejű üzenetküldést és azonnali értesítéseket valósíthatnak meg.

További mikroszolgáltatások bevezetése: A későbbiekben a rendszer könnyen felosztható különálló szolgáltatásokra (UserService, EventService, NotificationService stb.), amelyek önállóan skálázhatók és fejleszthetők.

Caching mechanizmusok: Redis vagy MemoryCache használatával javítható a rendszer teljesítménye gyakran lekérdezett adatok esetén.

API verziókezelés: Az API evolúciójának támogatása érdekében verziókezelési stratégia bevezetése (pl. URL-based vagy header-based versioning).