Záródolgozat

Készítették:
Csitári Ádám János – Kristály Adrián

Konzulensek:

Kerényi Róbert Nándor, Németh Bence

Miskolc

2024.

Miskolci SZC Kandó Kálmán Informatikai Technikum

Miskolci Szakképzési Centrum

SZOFTVERFEJLESZTŐ- ÉS TESZTELŐ SZAK

EventWave

Eseménykezelő közösségioldal

Csitári Ádám János – Kristály Adrián

2022-2024

Tartalomjegyzék

Eventwave eseménytár weboldal

1. Miért választottuk ezt a projektet?

1.1. Szükség

Egy olyan online platformra van szükség, ahol az emberek egyszerűen létrehozhatnak és megtalálhatnak eseményeket a különböző témákban és kategóriákban. Rengeteg ember manapság sajnos nehezen mozdul ki otthonról viszont a mi oldalunknak köszönhetően egy lépéssel előrébb kerülhet az átlagember ahhoz, hogy egy közösség tagja lehessen, sőt, hogy közösséget építhessen.

1.2. Közösségépítés

Az események összehozzák az embereket, lehetőséget teremtve a közösségek építésére és a kapcsolatok kialakítására a közös érdeklődési körök mentén. A közös érdeklődéssel rendelkező emberek így könnyedén találkozhatnak.

1.3. Tájékozódás

A felhasználók szeretnék könnyen és gyorsan megtalálni az érdeklődésüknek megfelelő eseményeket a különböző városokban vagy online.

1.4. Lehetőség

Rengetek embernek csak egy platformra van szüksége ahhoz, hogy ennek az oldalnak a segítségével indítsa el esemény szervezői karrierjét.

1.5. Rugalmas használat

A weboldal kialakítása intuitív és felhasználóbarát legyen, hogy mindenki könnyen megtalálja az általa keresett információkat, illetve könnyedén tudjon eseményeket létrehozni és kezelni.

Ezen szempontok alapján úgy gondoljuk, hogy az Eventwaves weboldal nagy segítséget nyújthat az embereknek az események felfedezésében és részvételében, valamint elősegítheti az online és offline közösségek kialakulását és erősödését.

2. Technológiák

2.1. Adatbázis

Mint adatbázis kezelő, felváltva használtunk XAMPP-ot illet Wamp-ot is egyaránt, programozási nyelv mit adatbázisban használtunk az a MySql.

Az XAMPP és a WAMP olyan szoftvercsomagok, amelyek teljesen konfigurált fejlesztői környezetet biztosítanak a PHP, a MySQL és az Apache webkiszolgáló számára. Ezek a csomagok könnyen telepíthetőek és konfigurálhatóak, így ideálisak lokális fejlesztéshez és teszteléshez.

2.1.1. XAMPP

A XAMPP egy szabad és nyílt forrású platformfüggetlen webszerver-szoftvercsomag, amelynek legfőbb alkotóelemei az Apache webszerver, a MariaDB adatbázis-kezelő, valamint a PHP és a Perl programozási nyelvek értelmezői.

2.1.2. WampServer(WAMP)

A WampServer(WAMP) a Microsoft Windows operációs rendszerhez készült megoldáscsomagra utal, amelyet Romain Bourdon hozott létre, és amely az Apache webszerverből, az SSL-támogatáshoz szükséges OpenSSL-ből, a MySQL adatbázisból és a PHP programozási nyelvből áll.

A XAMPP és a WAMP tartalmazza az Apache HTTP szervert, a MySQL adatbázis-kezelő rendszert és a PHP-t, valamint más alkalmazásokat és szolgáltatásokat is, mint például a phpMyAdmin adatbázis-kezelő felület.

2.1.3. PHP

A PHP egy széles körben használt, skriptnyelv alapú programozási nyelv, amely különösen alkalmas webes alkalmazások fejlesztésére. A MySQL pedig egy ingyenesen elérhető, nyílt forráskódú relációs adatbázis-kezelő rendszer, amely hatékonyan tárolja és kezeli az adatokat.

A XAMPP és a WAMP segítségével könnyedén létrehozhatunk egy lokális fejlesztői környezetet, ahol gyorsan és hatékonyan fejleszthetünk és tesztelhetünk webalkalmazásokat, miközben kihasználjuk a PHP és MySQL által nyújtott lehetőségeket.

2.2. Backend

- Backendbe az adatok megfelelő áramlását, illetve hibakezelés, adatvédelem egyaránt megjelenik.

2.2.1. A fejlesztői környezet

A fejlesztői környezet backend fejlesztéséhez használtunk egy olyan integrált fejlesztői környezetet (IDE egy olyan szoftveralkalmazás, amely átfogó létesítményeket biztosít a számítógépes programozók számára a szoftverfejlesztéshez. Az IDE általában legalább egy forráskódszerkesztőből, automatizálási eszközökből és hibakeresőből áll.), mint például a Visual Studio, amely lehetővé teszi a könnyű kódírást, hibakeresést és projektmenedzsmentet. A fejlesztői környezet lehetővé teszi a szoftverfejlesztőknek a kódírás és a tesztelés folyamatának optimalizálását.

2.2.2. C#

A C# egy erős, típusos programozási nyelv a .NET platformhoz, amelyet gyakran használnak a backend fejlesztéséhez. A C# támogatja az objektumorientált programozást, a modern nyelvi elemeket és erőteljes eszközöket kínál a fejlesztőknek.

2.2.3. Web API

A Web API-k (Application Programming Interface) lehetővé teszik a backend szolgáltatások elérését HTTP protokollon keresztül. A Web API egy alkalmazás programozási interfésze a weben. A Böngésző API kiterjesztheti a webböngésző funkcionalitását. A Szerver API kiterjesztheti a webszerver funkcionalitását. A RESTful elveket követve a Web API-k segítségével könnyen kommunikálhatunk a frontend alkalmazásokkal, például AJAX kérések vagy egyéb HTTP kérések révén.

2.2.4. Entity Framework

Az Entity Framework egy ORM (Object-Relational Mapping) keretrendszer a .NET-hez, amely lehetővé teszi az adatbázis-entitások és az objektumok közötti átjárást és az adatbázis-műveletek kezelését C# objektumokként.

Az Entity Framework egy modern objektum-relációs leképző, amely lehetővé teszi egy tiszta, hordozható és magas szintű adatelérési réteg építését .NET környezetben többféle adatbázisra, beleértve az SQL Database-t (helyi és Azure), SQLite-ot, MySQL-t, PostgreSQL-t és Azure Cosmos DB-t is. Támogatja a LINQ lekérdezéseket, az állapotkövetést, a frissítéseket és a séma migrációkat. "https://learn.microsoft.com/en-us/ef/"

2.2.5. Az Object-Relational Mapping (ORM)

Egy olyan technika, amely lehetővé teszi az objektumok és az adatbázisok közötti átjárást, anélkül, hogy a fejlesztőnek közvetlenül SQL-lel kellene dolgoznia. Az ORM segítségével a fejlesztők egyszerűbb és absztrakt módon dolgozhatnak az adatokkal.

2.2.6. A Cross-Origin Resource Sharing (CORS)

Egy HTTP fejlécalapú mechanizmus, amely lehetővé teszi a szerver számára, hogy jelezze az eredet (domain, séma vagy port) más forrásokat is, mint a sajátját, amelyekről a böngészőnek engedélyeznie kell az erőforrások betöltését.

A CORS továbbá egy mechanizmusra támaszkodik, amelynek segítségével a böngésző "preflight" (egy előzetes kérés, amelyet a böngésző küld a szervernek a tényleges keresés előtt a CORS mechanizmus során.) kérést küld a kereszt eredetű erőforrást hostoló szervernek, annak ellenőrzése érdekében, hogy a szerver engedélyezi-e a tényleges kérést. Ebben a preflight kérésben a böngésző olyan fejléceket küld, amelyek jelzik a tényleges kérésben használt HTTP módot és fejléceket. https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

2.2.7. A CRUD műveletek (Create, Read, Update, Delete)

Az alapvető adatműveletek, amelyek lehetővé teszik az adatok kezelését az alkalmazásban. Amikor API-kat építünk, azt szeretnénk, hogy a modellek négy alapvető funkcionalitást nyújtsanak. A modellnek képesnek kell lennie arra, hogy létrehozzon (Create), olvasson (Read), frissítsen (Update) és töröljön (Delete) erőforrásokat. Egy modellnek képesnek kell lennie legalább ezekre a négy műveletre ahhoz, hogy teljes legyen. Ha egy művelet nem írható le egyik ezek közül a négy művelet közül, akkor potenciálisan egy új modell lehet. A biztonsági szempontok fontosak a backend fejlesztésekor, például az autentikáció, az autorizáció és a JWT tokenek használata révén. https://www.codecademy.com/article/what-is-crud

2.2.8. JWT token

A JSON Web Token (JWT) egy kompakt, önálló és biztonságos módon átvihető token formátum, amelyet általában az autentikáció és az autorizáció során használnak. A JWT tokenek lehetővé teszik a biztonságos hitelesítést és az azonosítást a felhasználók számára a backend alkalmazásokban.

A JWT hitelesítés egy token-alapú állapotmentes hitelesítési mechanizmus. Ez általában ügyféloldali állapotmentes munkamenetként használatos, ami azt jelenti, hogy a szervernek nem kell teljesen támaszkodnia egy adattárolóra vagy adatbázisra a munkamenetinformációk mentéséhez.

A JWT-k titkosíthatók, de általában kódoltak és aláírtak. Az aláírt JWT célja nem a adatok elrejtése, hanem az adatok hitelességének biztosítása. És ezért nagyon ajánlott az HTTPS használata aláírt JWT-kkal.

2.2.8.1. A JWT három része

Fejléc (Header): A fejléc két részből áll: az algoritmus, amelyet a token aláírásához használnak, például SHA256 vagy RSA, valamint a token típusa, amely ebben az esetben JWT.

Adattartalom (Payload): Az adattartalom a claim-eket tartalmazza. A claim-ek entitásspecifikusak, például átadhatunk felhasználói azonosítót, felhasználónevet vagy bármit, amit szeretnénk.

Aláírás (Signature): Az aláírás az encoded header, az encoded payload és egy, csak a szerver által ismert titkos kulcs kombinációjából képződik. Azt szolgálja, hogy ellenőrizze a token integritását és biztosítsa, hogy ne lehessen manipulálni.

https://www.miniorange.com/blog/what-is-jwt-json-web-token-how-does-jwt-authentication-work/

https://medium.com/@ritikkhndelwal/jwt-authentication-in-the-frontend-enhancing-security-in-web-applications-ece418206b4f

2.3. Frontend

2.3.1. React

A frontend fejlesztéséhez használtuk a React keretrendszert. A React egy modern, deklaratív és hatékony JavaScript könyvtár, melyet a felhasználói felületek dinamikus és interaktív módon történő készítésére használnak.

A Virtual DOM segítségével hatékonyan kezeli a felület változásait, minimalizálva ezzel a tényleges DOM manipulációkat, ami jelentős teljesítménybeli előnyökkel jár.

A komponens alapú architektúra lehetővé teszi a kód újrafelhasználhatóságát és a komplex alkalmazások könnyű kezelhetőségét. A széles körben elérhető közösségi támogatás és a gazdag ökoszisztéma (pl. Redux, React Router) segítségével a React ideális választás a nagyobb méretű és skálázható frontend projektekhez.

2.3.2. node.js

A node.js egy JavaScript futtatókörnyezet, mely lehetővé teszi a JavaScript kódok futtatását a szerveroldalon, így lehetővé téve az aszinkron I/O műveletek hatékony kezelését. Ezáltal a frontend fejlesztőknek lehetőségük van teljes stack alkalmazások fejlesztésére, melyben a kliens és a szerver oldali kód is JavaScriptben íródik.

A node.js segítségével könnyen készíthetünk RESTful API-kat vagy valós idejű alkalmazásokat. A technológia dinamikus közössége és bőséges moduláris csomagrendszere (npm) lehetővé teszi a fejlesztők számára, hogy gyorsan és hatékonyan építsenek komplex alkalmazásokat. Azáltal, hogy egy JavaScript alapú környezetet biztosít mind a kliens, mind a szerveroldali fejlesztéshez, a node.js jelentős mértékben elősegíti a fejlesztési folyamat egyszerűségét és gyorsaságát.

2.3.3. Visual Studio Code

A kód szerkesztésére és a projekt menedzselésére a Visual Studio Code-ot használtuk. A Visual Studio Code egy ingyenesen elérhető forráskód szerkesztő és fejlesztői környezet, amelyet a Microsoft fejleszt. A Visual Studio Code egy rendkívül népszerű és teljes körű fejlesztői eszköz, amely széles körben használható különféle programozási nyelvekhez és technológiákhoz.

A könnyen bővíthető funkciók és a gazdag kiegészítőkészlet lehetővé teszi a fejlesztők számára, hogy testreszabják és optimalizálják az IDE-t(Integrated Development Environment) az egyedi igényeiknek megfelelően.

A Visual Studio Code integrált támogatást nyújt a git verziókezeléshez és más fejlesztési eszközökkel, például a Dockerrel vagy a szerveroldali fejlesztéshez használt node.js-el történő együttműködéshez. Emellett könnyen kezelhető felhasználói felülete és gyors teljesítmény teszi az egyik legvonzóbb választássá a fejlesztők számára.

2.3.4. JWT Token

Az autentikáció és az autorizáció megvalósításához használtuk a JWT (JSON Web Token) tokent. A JWT egy nyílt szabvány, amelyet az információk biztonságos megosztására használnak két fél között – egy kliens és egy szerver között.

2.4. WPF

A Windows Presentation Foundation (WPF) egy grafikus felhasználói felületet (GUI) készítő keretrendszer, amelyet a Microsoft fejlesztett. A WPF lehetővé teszi a fejlesztők számára, hogy korszerű, testreszabható és esztétikus felhasználói felületeket hozzanak létre Windows alkalmazásokhoz. A WPF alapvetően XAML (eXtensible Application Markup Language) segítségével épül fel, amely lehetővé teszi a felhasználói felületek deklaratív leírását.

2.4.1. XAML

A WPF felhasználói felületek leírására használt nyelv, amely XML alapú. A XAML segítségével lehetőség van a felületek könnyű tervezésére és testre szabására.

2.4.1.1. Adatkötés:

A WPF erőteljes adatkötési mechanizmust biztosít, amely lehetővé teszi az adatok dinamikus megjelenítését és frissítését a felhasználói felületen.

2.4.1.2. Stílusok és sablonok:

A WPF lehetővé teszi a stílusok és sablonok alkalmazását a felhasználói felület egyes elemeire, így egyszerűen testreszabhatók a megjelenés és viselkedés szempontjából.

2.4.1.3. Animáció és átmenetek:

A WPF lehetőséget biztosít a dinamikus animációk és átmenetek létrehozására, amelyek fokozzák a felhasználói élményt.

2.4.1.4. Eseménykezelés:

A WPF lehetőséget biztosít a különböző események kezelésére és az ezekhez kapcsolódó műveletek végrehajtására.

2.4.1.5. Képfeldolgozás:

A WPF lehetőséget biztosít a különböző képek és grafikák kezelésére, manipulálására és megjelenítésére.

3. Munkafolyamat

3.1. Adatbázis

A projekt alapköve számunkra az adatbázis volt itt. Az üzleti logika alapjait, az adatáramlás rendszerét itt fektettük le az adatbázisnál, itt alapoztuk meg hogyan is fog működni az oldal.

A projekt adatbázisának tervezése során törekedtünk arra, hogy optimalizáljuk a tárolást és minimalizáljuk a redundanciát. Ennek érdekében két fő elvet alkalmaztunk:

3.1.1. 3NM (Third Normal Form):

A harmadik normálforma alkalmazása lehetővé teszi számunkra, hogy a tábláinkat úgy tervezzük meg, hogy minimalizáljuk a redundanciát és ezáltal optimalizáljuk a tárolást. Ez azt jelenti, hogy a tábláinkban minden nemkulcs attribútum csak a kulcsoktól függ, és nincsenek tranzitív funkcionális függőségek. Így biztosítjuk, hogy az adatok egyszerűen és hatékonyan tárolhatók legyenek, miközben minimalizáljuk az anomáliákat és a felesleges adatmennyiséget.

3.1.2. Cascade törlés:

A Cascade törlés lehetőséget biztosít számunkra arra, hogy konzisztens állapotban tartsuk az adatbázist. Amikor egy rekordot törölünk az altáblából, automatikusan töröljük a hozzá tartozó rekordokat a fő táblából is. Ez segít elkerülni az adatbázisban a felesleges vagy elavult adatokat, és biztosítja az adatbázis integritását.

Fontos azonban megjegyezni, hogy ez a művelet veszélyes lehet, hiszen sokszor nem törlik az adatokat, hanem csak jelzik, hogy törölt rekordként kezeljék. Ezért

körültekintően kell eljárni a cascade törlés alkalmazásakor, hogy ne veszítsünk el fontos információkat az adatbázisban.

Az adatbázis tervezése során ezeket az elveket alkalmazva biztosítottuk, hogy az adatok hatékonyan tárolódjanak és könnyen kezelhetőek legyenek, miközben megőrizzük az adatbázis integritását és konzisztenciáját.

3.1.3. Adatbázis adatbázismodell-diagramja (utalás)

3.2. Backend

3.2.1. Alapelvek

A backend a rendszer olyan része, amely a felhasználói felülettől függetlenül működik. Ez azt jelenti, hogy nem kell minden egyes változtatást mind a frontend, mind a backend részen megváltoztatni, ami időt és erőforrásokat takarít meg. A kommunikációhoz használhatunk különböző módszereket, például az egyszerű HTTP-t vagy a biztonságosabb HTTPS-t. Az alkalmazást könnyen futtathatjuk Dockerrel vagy IIS szolgáltatás alatt, ami rugalmasságot biztosít a környezet kialakításában.

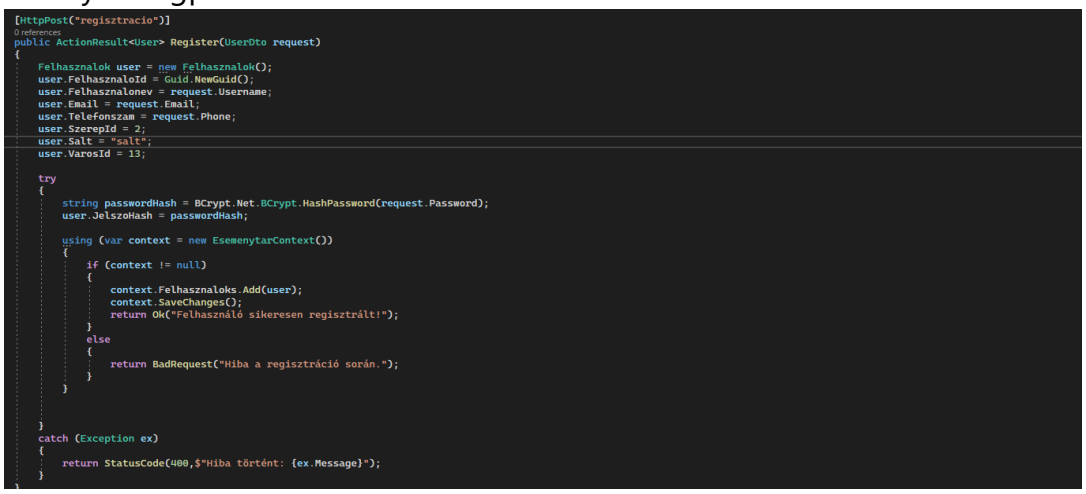
3.2.2. Adatkezelés

Az adatbázisban tárolt adatokhoz az Entity Framework-öt használjuk, ami lehetővé teszi az objektum-orientált adatkezelést. Fontos megjegyezni, hogy az Entity Framework nem engedi minden adatbázis-módosítást, különösen a kulcsok (elsődleges és idegen kulcsok) módosítását, amire figyelnünk kell.

3.2.3. Kapcsolatok és API-k

A biztonságos kommunikáció érdekében a különböző platformok között CORS-t kell beállítanunk. Az adatok kezelését a CRUD műveletekkel végzünk (Create, Read, Update, Delete), amelyeket az API-k segítségével érünk el.

Bizonyos végpontokhoz tesztek is lettek írva.

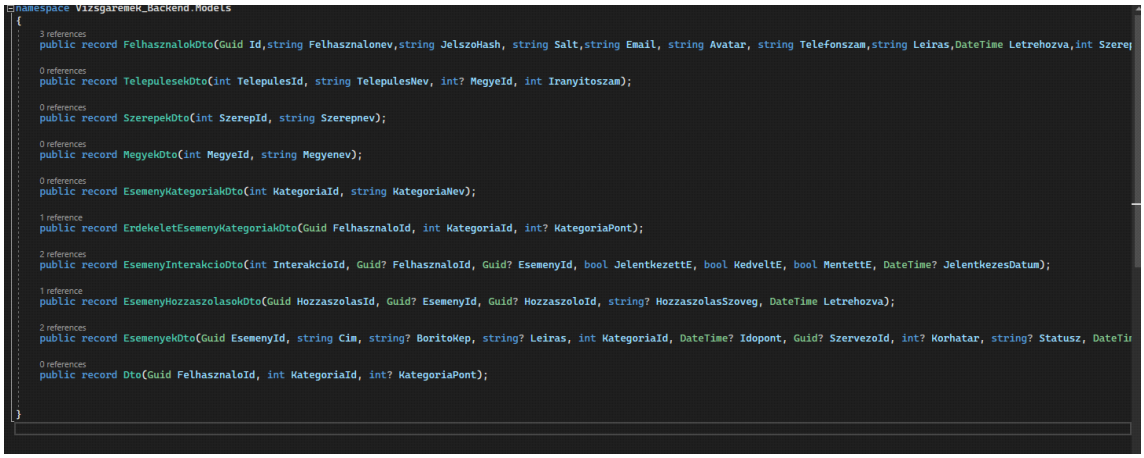


```
[HttpPost("regisztracio")]
0 references
public ActionResult<User> Register(UserDto request)
{
    Felhasznalok user = new Felhasznalok();
    user.FelhasznaloId = Guid.NewGuid();
    user.Felhasznalonev = request.Username;
    user.Email = request.Email;
    user.Telefonszam = request.Phone;
    user.SzerepId = 2;
    user.Salt = "salt";
    user.VarosId = 13;

    try
    {
        string passwordHash = BCrypt.Net.BCrypt.HashPassword(request.Password);
        user.JelszoHash = passwordHash;

        using (var context = new EsemenytarContext())
        {
            if (context != null)
            {
                context.Felhasznaloks.Add(user);
                context.SaveChanges();
                return Ok("Felhasználó sikeresen regisztrált!");
            }
            else
            {
                return BadRequest("Hiba a regisztráció során.");
            }
        }


    }
    catch (Exception ex)
    {
        return StatusCode(400,$"Hiba történt: {ex.Message}");
    }
}
```

(itt a weboldalhoz való bejelentkezés valósul meg, alapvetően vannak alap értékek, mik módosulnak aszerint, amit a felhasználó megad.)

3.2.4. Adatátviteli Objektumok (DTO-k)

Az adatátviteli objektumok segítségével csak a szükséges adatokat küldjük el. Például, ha egy adott táblából nem minden információt szeretnénk elküldeni, használhatjuk a DTO-kat, ezáltal minimalizálva az adatok mennyiségét és növelve a kommunikáció hatékonyságát és biztonságát.



```
namespace Vizsgaremek_Backend.Models
{
    3 references
    public record FelhasznalokDto(Guid Id,string Felhasznalonev,string JelszoHash, string Salt,string Email, string Avatar, string Telefonszam,string Leiras,DateTime Letrehozva,int Szere

    0 references
    public record TelepulesekDto(int TelepulesId, string TelepulesNev, int? MegyeId, int Iranyitoszam);

    0 references
    public record SzerepekDto(int SzerepId, string Szerepnev);

    0 references
    public record MegyekDto(int MegyeId, string Megyenev);

    0 references
    public record EsemenyKategoriakDto(int KategoriaId, string KategoriaNev);

    1 reference
    public record ErdekeletEsemenyKategoriakDto(Guid FelhasznaloId, int KategoriaId, int? KategoriaPont);

    2 references
    public record EsemenyInterakcioDto(int InterakcioId, Guid? FelhasznaloId, Guid? EsemenyId, bool JelentkezettE, bool KedveltE, bool MentettE, DateTime? JelentkezesDatum);

    1 reference
    public record EsemenyHozzaszolasokDto(Guid HozzaszolasId, Guid? EsemenyId, Guid? HozzaszoloId, string? HozzaszolasSzoveg, DateTime Letrehozva);

    2 references
    public record EsemenyekDto(Guid EsemenyId, string Cim, string? BoritoKep, string? Leiras, int KategoriaId, DateTime? Idopont, Guid? SzervezoId, int? Korhatar, string? Statusz, DateTi

    0 references
    public record Dto(Guid FelhasznaloId, int KategoriaId, int? KategoriaPont);


}
```

3.2.5. Felhasználói Azonosítás és Jogosultságok

Az autentikációt és az autorizációt a JWT (JSON Web Token) és BCrypt technológiákkal végzünk. A JWT token-alapú autentikációs megoldás, amely lehetővé teszi a felhasználók azonosítását és az információk biztonságos átvitelét.

```
1 reference
private string CreateToken(User user)
{
    using (var context = new EsemenytarContext())
    {
        List<Claim> claims = new List<Claim>()
    {
        new Claim(ClaimTypes.NameIdentifier, user.Id.ToString()),
        new Claim(ClaimTypes.Name, user.Username),
        new Claim(ClaimTypes.Email, user.Email),
        new Claim(ClaimTypes.DateOfBirth, user.Letrehozva.ToString()),
        new Claim(ClaimTypes.PostalCode, user.Varos_Id.ToString()),
        new Claim(ClaimTypes.MobilePhone, user.Telefonszam.ToString()),
        new Claim(ClaimTypes.Role, context.Szerepeks.FirstOrDefault(x=>x.SzerepId == user.Szerep_Id).Szerepnev)

    };

        var kulcs = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration.GetSection("AppSettings:Token").Value!));

        var cr = new SigningCredentials(kulcs, SecurityAlgorithms.HmacSha512Signature);

        var token = new JwtSecurityToken(
            claims: claims,
            expires: DateTime.Now.AddDays(1),
            audience: _configuration.GetValue<string>("Authentication:Schemes:Bearer:ValidAudiences:0"),
            issuer: _configuration.GetSection("Authentication:Schemes:Bearer:ValidIssuer").Value,
            signingCredentials: cr);

        var jwt = new JwtSecurityTokenHandler().WriteToken(token);

        return jwt;
    }
```

(a JWT tokenbe, itt menti el minden olyan adatot mik szükségesen a felhasználók megfelelő kezeléséhez)

A BCrypt jelszó-hashelési algoritmus biztosítja a jelszavak biztonságos tárolását és ellenőrzését, megvédve azokat a támadásoktól, például a brute force-tól és a rainbow támadásoktól. Ezáltal biztosítva, hogy csak azonosított felhasználók végezhessenek műveleteket, míg az adminisztrátoroknak különleges jogosultságok vannak a rendszerben.

3.3. Frontend

A weboldal fejlesztése a Visual Studio Code fejlesztői környezetben történik, mivel ez egy könnyen használható és sokoldalú kódszerkesztő, amely támogatja a modern webfejlesztési technológiákat. Bővítményei (extensions) rendkívül hasznosnak bizonyultak a munkafolyamat során. Csak az alább említett reacthoz 3 különböző, bővítményt használtunk. Az auto save funkció pedig rendkívül gyorsítja a tesztelést.

3.3.1. Node.js

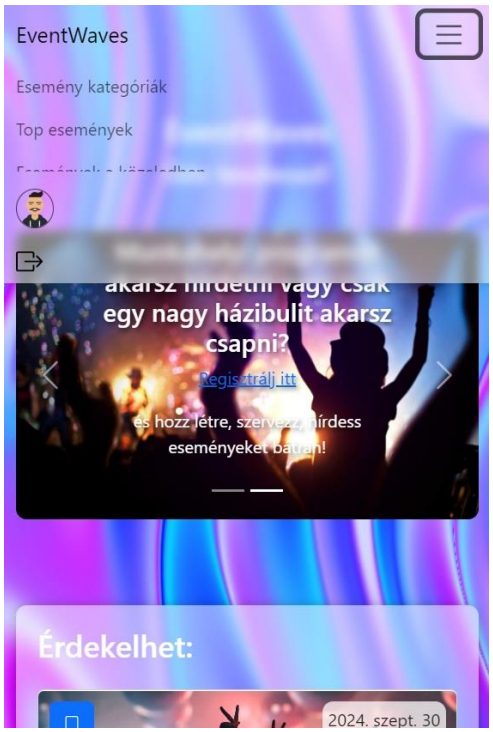
3.3.1.1. Telepített Node.js Komponensek

A projekt működéséhez elengedhetetlenül szükséges Node.js komponensek telepítése a parancssorban történik. Ezek a komponensek biztosítják a fejlesztő számára a szükséges környezetet és eszközöket a frontend kód írásához és futtatásához.

3.3.1.2. Node verzió

A projekt szempontjából fontos, hogy a megfelelő Node.js verziót használjuk, hogy biztosítsuk a kompatibilitást és a stabilitást. Az akkor még legfrissebb verziót installáltuk, amely a v20.10.0-as kiadás.

3.3.2. Bootstrap



A Bootstrap keretrendszert választottuk a weboldal megjelenítésének és a reszponzív design kialakításának céljából. A Bootstrap egy jól ismert és elismert CSS keretrendszer, amely lehetővé teszi a gyors és egyszerű reszponzív weboldalak készítését.

3.3.2.1. Megjelenítés

A Bootstrap segítségével könnyen létrehozhatunk vonzó és esztétikus felhasználói felületeket, anélkül, hogy túl sok időt kellene fordítanunk a CSS stílusok írására.

3.3.2.2. Reszponizivitás

A Bootstrap használatára egy erősen ösztönző tulajdonság, hogy a Bootstrap alapú weboldalak reszponzív jellege garantálja, hogy azok megfelelően jelenjenek meg és működjenek különböző eszközökön és képernyőméreteken, így javítva a felhasználói élményt, valamint használata a munkafolyamatot is gyorsítja. Majdnem minden komponensben használtunk bootstrappet, pontosítva react-bootstrap elemeket.

3.3.3. React

A React keretrendszert választottuk a weboldal frontendjének fejlesztéséhez, mivel ez egy hatékony és korszerű eszköz a dinamikus felhasználói felületek készítéséhez és kezeléséhez.

A React segítségével könnyen létrehozhatunk egy új projektet a create-react-app paranccsal, amely előre beállítja a projekt struktúráját és inicializálja a szükséges fájlokat.

A React alkalmazás fejlesztése során a szokásos műveleteket végezzük, beleértve a projekt buildelését, a helyi fejlesztői szerver indítását és a változtatások figyelését és újrafordítását.

3.3.3.1. Előnyök és Hátrányok

A React keretrendszernek számos előnye van, mint például a könnyű tanulhatóság és az újrafelhasználható komponensek, ugyanakkor néhány hátrány is lehet, például a komplexitás és a kezdeti beállítások bonyolultsága.

3.3.3.2. Telepített Komponensek

A projekt során telepítettünk különböző kiegészítő modulokat, például az Axios-t, amely segítségével könnyen kommunikálhatunk a szerverrel aszinkron HTTP kérésekkel. Különböző megjelenítéshez szükséges komponenseket is töltöttünk le, pl. a „react-bootstrap", „Autosuggest", „mdb-react-ui-kit", stb.

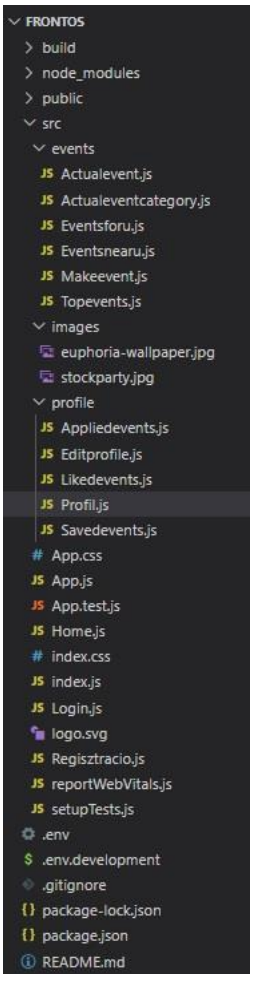
```
import React from 'react'
import { Col, Button, Row, Container, Card, Form} from 'react-bootstrap';
import { BrowserRouter as Router, Link, Route, Routes } from 'react-router-dom';
import Regisztracio from './Regisztracio';
import axios from 'axios';
import { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { jwtDecode } from 'jwt-decode';
import {
  MDBFooter,
  MDBContainer,
  MDBCol,
  MDBRow
} from 'mdb-react-ui-kit';
```

Nagyon fontos volt még a „react-router-dom" amely lehetővé teszi dinamikus útvonalak megvalósítását egy webalkalmazásban. Lehetővé teszi az oldalak megjelenítését és a felhasználók navigálását közöttük. A React Router Domot arra használják, hogy egyoldalas alkalmazásokat hozzanak létre, azaz olyan alkalmazásokat, amelyeknek sok oldala vagy komponense van, de az oldal soha nem frissül, hanem a tartalom dinamikusan lekérdezésre kerül az URL alapján.

3.3.4. Tesztelés

A projekt fejlesztése során rendszeres teszteléseket végeztünk, beleértve a komponensek és funkciók, integrációs tesztjeit és felhasználói felület tesztjeit is.

3.3.5. Komponensek és felépítéseik



Mi nem a szokványos react fájl és komponens architektúra alapján építettük fel a projektet mivel így egyszerűbbnek tűnt adatátadás és elhatárolás szempontjából, de sajnos ennek a hátránya, hogy lassabb az oldal, mint amilyen lehetne, valamint rengeteg dolgot egyszerűbben meg lehetett volna oldani retrospektív szemmel nézve.

Jelenleg a könyvtár így néz ki:

Egy oldal egy komponens, ami mindent egybe gyúr, amit az adott oldalon meg akarunk jeleníteni, nem iskolapélda, de működik és néhány szempontból gyorsította néhány szempontból viszont lassította a munkafolyamatot, ennek ellenére összeségében hasznosnak bizonyult ez a felépítés.

3.4. WPF

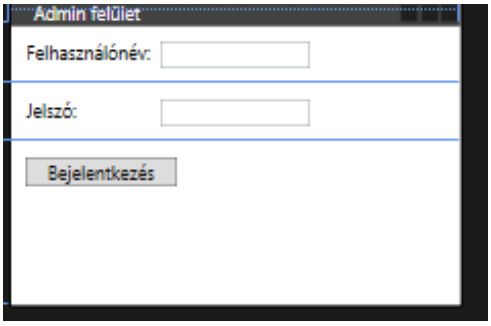
Az alkalmazás célja lehetővé tenni az események és felhasználók kezelését, valamint biztosítani egy biztonságos bejelentkezési mechanizmust az adminisztrátori jogosultságokhoz.

3.4.1. Funkciók

3.4.1.1. Bejelentkezés

A felhasználók bejelentkezhetnek az alkalmazásba a felhasználónevével és jelszavával.

A bejelentkezési adatokat ellenőrzi az alkalmazás az adatbázisban tárolt adatok alapján.



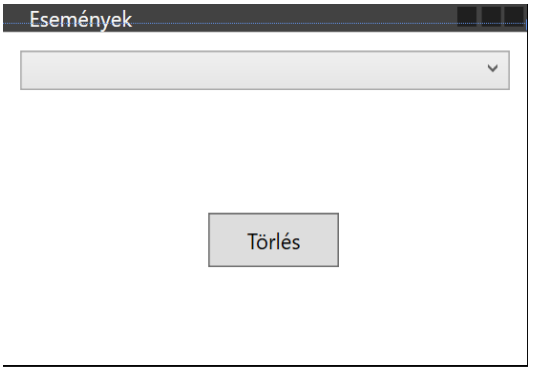
Sikeres bejelentkezés esetén az adminisztrátori felületre irányítja a felhasználót.

3.4.1.2. Adminisztrátori Felület (AdminWindow)

Az adminisztrátori felületen az adminisztrátor további tevékenységeket végezhet el. Lehetőség van az események és felhasználók kezelésére, valamint a kijelentkezésre.

3.4.1.3. Eseménykezelés (EventWindow)

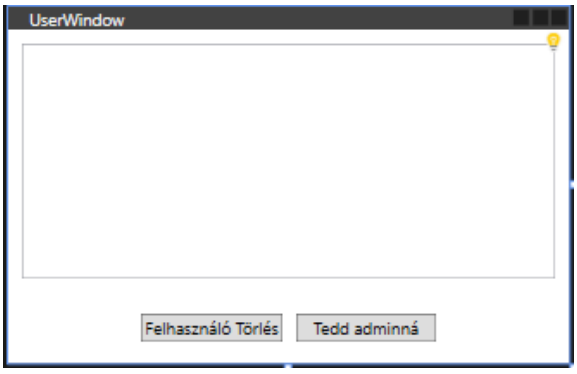
Az eseményeket lehet megtekinteni és törölni az EventWindow ablakon. Az események betöltődnek az adatbázisból és megjelennek egy ComboBox-ban. A felhasználók az egyes eseményeket kiválaszthatják, majd törölhetik őket az alkalmazásból.



3.4.1.4. Felhasználókezelés (UserWindow)

A felhasználókat lehet megtekinteni és törölni a UserWindow ablakon.

Lehetőség van egy felhasználót adminisztrátorrá tenni.



3.4.2. Adatbázis és Adatkezelés

Az alkalmazás MySQL adatbázist használ az adatok tárolására.

A kapcsolódáshoz és az adatlekérdezésekhez szükséges SQL műveleteket a megfelelő C# osztályok végzik.

A felhasználói jelszavakat biztonságosan tárolják a BCrypt hashelés segítségével.

3.4.3. Hibakezelés és Biztonság

Az alkalmazás különböző hibákat kezel, beleértve az adatbázis kapcsolat hibáit és a felhasználói input validálását.

A felhasználói jelszavakat biztonságosan tárolják hashelt formában, hogy megakadályozzák a jelszavak illetéktelen hozzáférését.

3.4.4. Továbbfejlesztési lehetőségek

Az alkalmazás további funkciókkal bővíthető, például részletes esemény- és felhasználói adatlapokkal.

Az UI/UX javítása érdekében újabb dizájn elemek bevezetése lehetséges.

A kód optimalizálása és struktúrájának további tisztázása a karbantarthatóság és kiterjeszthetőség érdekében.

3.4.5. Megjegyzések

Az alkalmazás jelenlegi verziója a legfontosabb alapfunkciókat valósítja meg.

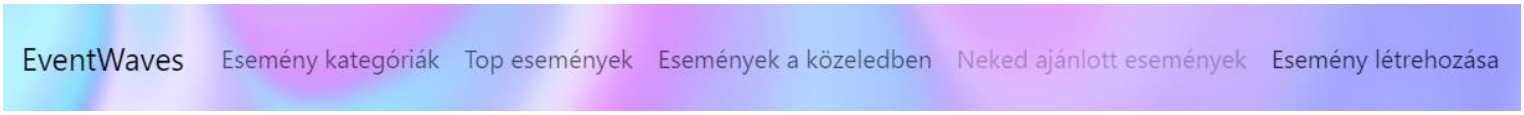
Az alkalmazás főbb komponensei (MainWindow, AdminWindow, EventWindow, UserWindow) külön osztályokba vannak szervezve a jobb karbantarthatóság és olvashatóság érdekében.

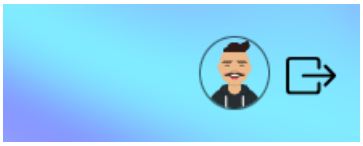
4. Webalkalmazás rövid bemutatása és használata

4.1. Főoldal

Amint a felhasználó rámegy az oldalra a főoldallal találkozik, azaz a Home komponenssel, amiben az oldal neve látszódik, és ha be van jelentkezve egy felhasználó akkor egy üdvözlő felirat fogadja a felhasználónevével viszontlátva. Itt még van egy canvas amiben reklám szöveg található. A canvas alatt négy véletlenszerűen kiválasztott esemény kártyája jelenik meg, alatta pedig a footer.

Az app.js -be beépített navbarral is találkozik a felhasználó, amelynek tartalma változik attól függően, hogy a felhasználó bevan-e jelentkezve vagy nincs. A navbaron elérhetőek az alábbi menüpontok.





4.2. Profil menü

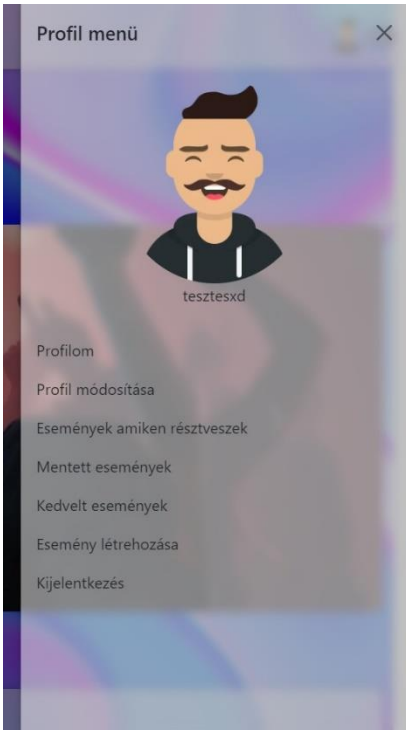
Ha a felhasználó a navbaron profilképére kattint megjelenik az alábbi sidebar amin a felhaszálóval kapcsolatos komponensek érhetőek el.

„Profilomra" kattintva a felhasználó megtekintheti a felhasználóiadatait.

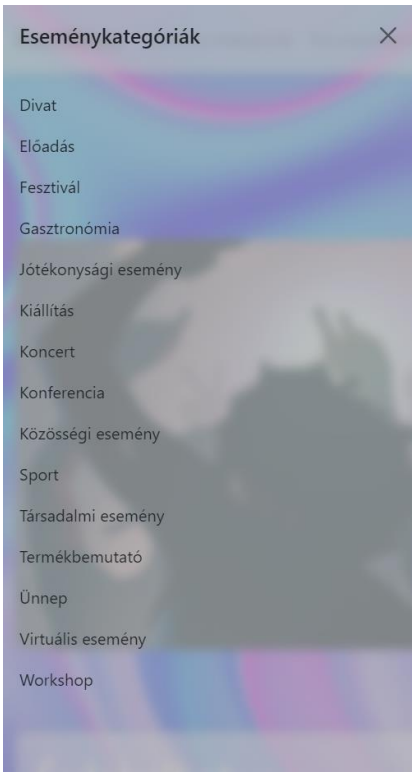
Profil módosítására kattintva a felhasználó egy form-al találkozik, amelyben az eredeti adatai találhatóak placeholderként amelyeket módosíthat.

Mentett, kedvelt és „események amiken résztveszek" olyan oldalakra visz amely kilistázza azon események kártyáját melyekkel az adott interakcióval interaktált.

Kijelentkezésre kattintva pedig kitörli a felhasználó tokenjét ezzel kijelentkeztetve a felhasználót.



4.3. Eseményekkel kapcsolatos komponensek



Ha felhasználó a navbaron az eseménykategóriákra kattint akkor egy sidebar menüben megjelennek az esemény kategóriák. Nyilvánvalóan, ha rákattintunk egy eseménykategóriára, olyan eseményeknek láthatjuk a kártyáit, amelyek a kiválasztott kategóriával rendelkeznek.

A „top események” az aktuálisan 40 legfrissebb eseményt listázza ki.

Az „események a közeledben” pedig figyeli a felhasználó települését és abból a megyéből ajánl eseményeket, amelyben a felhasználó tartózkodik.

A navbaron és profil menüben is megtalálható az esemény létrehozása, amely egy form ami bekéri egy esemény jellemző adatait és a feltöltés gombbal létrehozhat a felhasználó egy eseményt.

6. Kommunikációs eszközök

A csapat hatékony működéséhez elengedhetetlenek voltak az olyan kommunikációs eszközök, amelyek lehetővé teszik a gyors és strukturált információ-, és fájláramlást, verziókezelést, valamint a közös munkát és annak vázolását. Az alábbiakban öt ilyen eszközt használtunk fel projektmunkánk során.

6.1. Trello

A Trello egy digitális kanban tábla, amely lehetővé teszi a feladatok és projekt részleteinek vizuális szervezését és nyomon követését. A felhasználók egyszerűen hozzáadhatnak kártyákat különböző listákhoz, és mozgathatják azokat az előrehaladásuknak megfelelően. Használata átláthatóvá teszi a feladatok elosztását és követését, valamint elősegíti a csapatmunkát és a hatékony projektmenedzsmentet.

6.2. Github

A Github egy online verziókezelő platform, amely lehetővé teszi a fejlesztők számára a projektforrások verzióinak nyomon követését és kezelését. A Git alapú rendszer lehetővé teszi a kódbázisok közös munkáját, a változtatások kezelését és azok visszavonását. A pull requestek és ágak segítségével a fejlesztők könnyen együtt dolgozhatnak, és az alkalmazás fejlesztése strukturált és nyomon követhető marad.

6.3. Messenger

Az üzenetküldő alkalmazások, mint például a Messenger, azonnali kommunikációt biztosítanak a csapattagok között. Ez lehetővé teszi az egyszerű kérdések gyors megválaszolását, az ötletek megbeszélését és azonnali visszajelzést a projektekkel kapcsolatban. Azonnali és informális kommunikációs csatornaként szolgált, ami felgyorsította a döntéshozatalokat és a problémamegoldást.

6.4. Discord

A Discord egy olyan kommunikációs platform, amely lehetővé teszi a csapatok számára a szöveges, hangos és video alapú kommunikációt. Különböző szerverek létrehozásával és csatornák létrehozásával a csapatok könnyen szervezhetik és kategorizálhatják a kommunikációt a projektjeikhez kapcsolódóan. A Discord rugalmas platformként szolgált a csapatnak, ahol egyszerre lehetőség van informális beszélgetésekre és hivatalos megbeszélésekre és közös munkára.

6.5. Google Drive

A Google Drive egy felhőalapú dokumentumtároló és megosztó platform, amely lehetővé teszi a felhasználók számára a dokumentumok, prezentációk, táblázatok és program fájlok tárolását és közös munkáját. A csapatok könnyen megoszthatják és egyidejűleg szerkeszthetik a fájlokat, valamint könnyen hozzáférhetnek hozzájuk bármilyen eszközről. A Google Drive hatékony eszköz a dokumentumok megosztásához és együttműködéshez, amely elősegíti a csapatok hatékonyabb munkáját és a projektmenedzsmentet.

Ezek az eszközök számos előnyt hoztak a csapatunk számára, segítve a hatékonyabb kommunikációban, együttműködésben és projektmenedzsmentben. A megfelelő eszközök kiválasztása és alkalmazása kulcsfontosságú a sikeres csapatmunka és projektvezetés szempontjából.

7. Összefoglalás

7.1. Célkitűzések és teljesítések, jövőbeli tervek:

Összességében a projekt a fő célja megvalósult, jöjjön létre az oldal és egy olyan eseménykezelő oldal, ami platformot ad az embereknek a közösség építésre, szervezésre, hirdetésre, és arra, hogy találjanak maguknak programokat.

A projekt elején terveztünk barátlistát és chatelést is, de akkor csak egy újabb messengert csináltunk volna, szóval elvetettük ezt a lehetőséget mivel a fő fókuszunk az események. Viszont hasonló funkciók alapjait „elrejtettük" a projektben, mint fejlesztési lehetőséget, az adatbázisban már megalapoztuk ezt a fejleszthetőséget és backendben is.

Az interakciók területén is még fejleszthető a projekt. A jelenlegi like, mentés és jelentkezés interakció mellett, kommenteléshez is szeretnénk lehetőséget nyújtani a felhasználóknak az események alá, akár profilok alá is. Ezzel is az online közösség építést kicsit közelebb hozni a felhasználókhoz a mi platformunkon. Ehhez is megalapoztunk az adatbázisban és a backendben.

Terveztünk még egy kisebb, alapszintű algoritmust is, amely figyeli, hogy a felhasználó milyen kategóriájú események iránt érdeklődik. Ez egy erős fejlesztési lehetőség, amely az elsők között van. Az elképzelés, hogy a felhasználó miután beregisztrál az oldalra az első bejelentkezésnél már ajánlunk esemény kategóriákat és a felhasználó által kiválasztottak kapnak például 15 pontot, további interakciókért minthogy, megnézi az eseményt, likeol, jelentkezik stb. plussz x pontot kapnak az esemény kategóriák így tudjuk megállapítani a felhasználó preferenciáját és ez alapján ajánlani neki eseményeket.

A konklúzió az, hogy a projektben rengeteg továbbfejlesztési potenciál van.

7.2. Munkamegosztás

A projektben a „product owner/scrum master" szerepét, a projekt kreatív részének kitalálását megvalósítását és irányítását, üzleti logikáját, a frontend és adatbázis készítését, valamint a dokumentáció „főszerkesztését" Csitári Ádám János végezte.

A Backendet, a wpf admin asztali alkalmazást és annak telepítőjét, valamint a dokumentáció tartalmi részének több mint felének előállítását, Kristály Adrián végezte.

8. Köszönetnyilvánítás

Elsőkörben szeretnénk megköszönni a szüleinknek, hogy kitartóan támogattak ebben a rövid két évben és hogy hittek bennünk!

A következő szekcióban pedig szeretnénk megköszönni azoknak a tanároknak, akik valamilyen formában hozzájárultak ahhoz, hogy megszülessen a projektmunka:

Itt elsőként szeretnénk megköszönni Kerényi Róbert Nándor osztályfőnökünknek a kitartó támogatást, lehetőségeket, és szellemi vezetést, amit osztályfőnökként végzett, és a rengeteg szakmai tudást, számunkra a bevezetést az informatika világába, segítséget, közös munkát, amit szakmai tanárként nyújtott! Kitartó munkájának hála, hogy betekintést nyerhettünk a szakmai világba és fel vagyunk vértezve a szükséges ismeretekkel, amelyek hozzájárulnak egy versenyképes szakmai pozíció eléréséhez. Köszönjük összességében az elmúlt két év munkáját, vezetését és a végig töretlenül velünk töltött időt.

Szeretnénk megköszönni Németh Bencének is! Hiába, papíron a mi csoportunkat nem tanította, így is rengeteg hasznos tudással, tippel, eszközzel és segítséggel járult hozzá a projektmunkához és a szakmai tudásunkhoz és hogy általa is kaptunk betekintést és lehetőséget az iskola utáni szakmai életben.

Sike Lászlónak szeretnénk megköszönni az előző tanévben letett backendes és asztali szilárd alapokat! Ő fektette le precízen és odaadóan azokat az alapvető ismereteket, amik nélkül nem tartanánk ott, ahol tartunk. Talán az egyik legodaadóbb ember volt egész két évünk alatt.

Köszönjük Kiss Rolandnak az idei évi munkáját! Általa bővíthettük a backendes és asztali tudásunkat, rengeteg hasznos tippet és tudást kaptunk és a segítséget a projektmunkához.

Szeretnénk megköszönni Négyesi Gábornak is a kétéves, időt nem kímélő munkáját! Ha szükséges volt szabadidejéből nem kímélve segített munkánk sikerességében. Általa tanultuk meg először a sikeres projektmunkához való alapokat, a csapatmunkához szükséges és elengedhetetlen eszközök használatát és frontendhez szükséges rengeteg tudást.

Farkas Zoltánnak úgyszintén szeretnénk megköszönni az éves munkáját és segítőkészségét! A rengeteg gyakorlati tudást, tananyagot, tanítást, gyakorlást köszönhetően tudtunk a backendhez szükséges ismereteinket és gyakorlati tudásunkat fejleszteni.

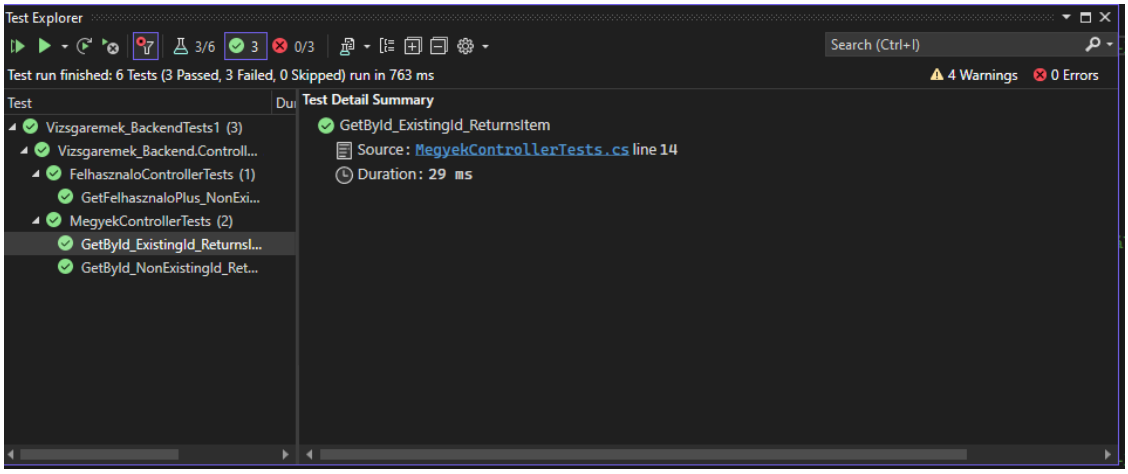
Köszönjük azoknak a tanáraiknak is, akiket név szerint nem soroltunk, de két évünk alatt jelen voltak és bővítették tudásunkat, valamint az vizsgaremek készítéséhez szükséges időszak alatt hagytak dolgozni, sőt, segítették munkánk sikerességét, vagy olyan tudással járultak hozzá életünkhöz, amit később tudunk kamatoztatni!

Utolsósorban pedig szeretnénk megköszönni osztálytársainknak az elmúlt két évet, ők voltak, akik könnyítették és rendkívül jó élménnyé tették a mindennapi munkát, nem beszélve a rengeteg közös munkáról, segítségről, kritikákról, amiket hozzánk és a projekthez tettek. Köszönjük nekik is.

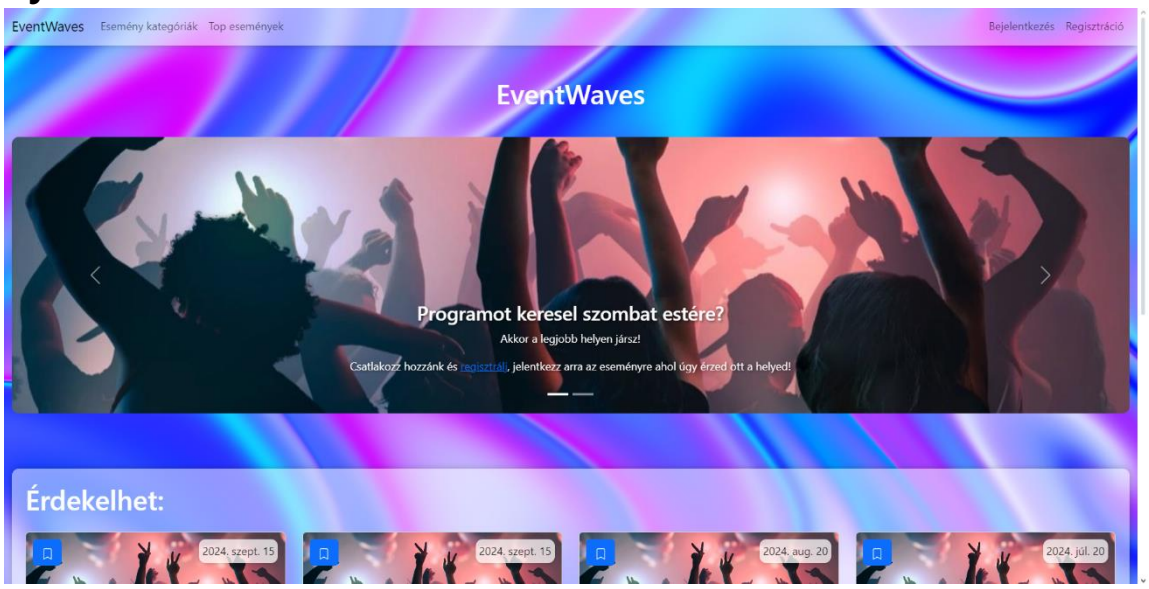
9. Mellékletek

Tesztelés (Unit test):

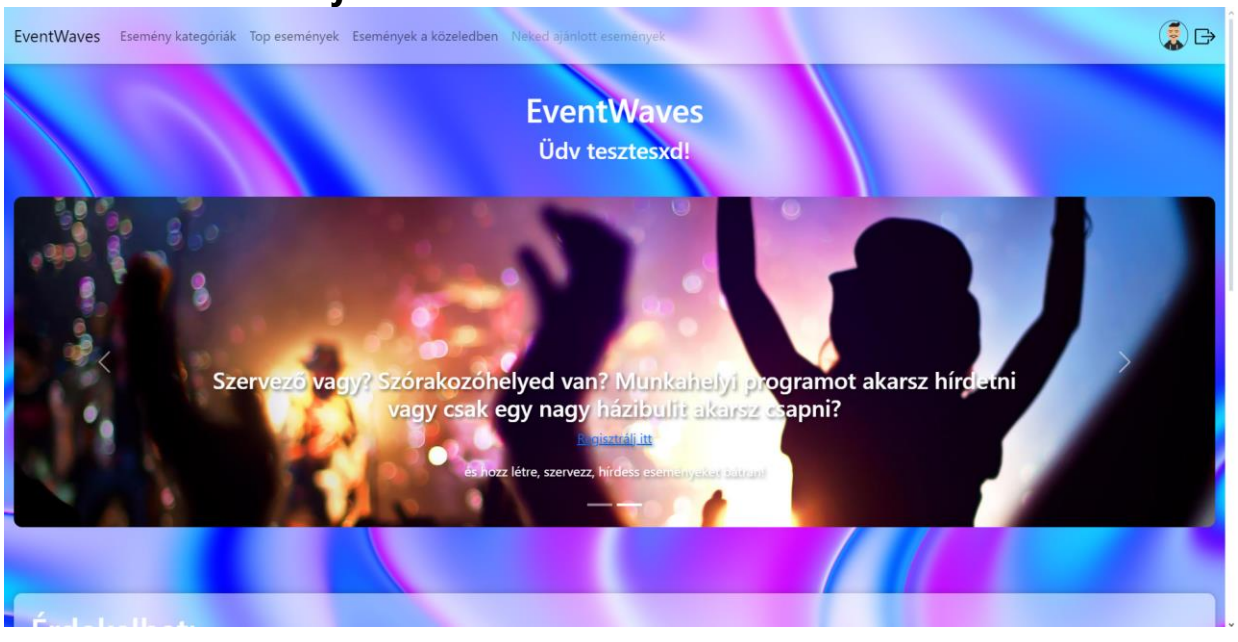
MegyekController és FelhasznalokController:



Home – nem bejelentkezve:



Home – bejelentkezve:



Az adatbázis adatbázismodell-diagramja: