

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Bui Quanganh Krisztián

Ingatlan hírdető portál megvalósítása

Konzulens

Benedek Zoltán

BUDAPEST, 2023

Tartalomjegyzék

[Összefoglaló 4](#_Toc135236089)

[Abstract 5](#_Toc135236090)

[1 Felhasznált technológiák 6](#_Toc135236091)

[1.1 .NET 6](#_Toc135236092)

[1.2 ASP.NET Core 7](#_Toc135236093)

[1.3 ASP.NET Identity 8](#_Toc135236094)

[1.4 Entity Framework Core 8](#_Toc135236095)

[1.5 TypeScript 8](#_Toc135236096)

[1.6 React 9](#_Toc135236097)

[1.7 Chakra UI 10](#_Toc135236098)

[2 Funkcionális követelmények 11](#_Toc135236099)

[2.1 Böngészés 11](#_Toc135236100)

[2.2 Regisztrálás és bejelentkezés 11](#_Toc135236101)

[2.3 Hirdetés létrehozása és kezelése 11](#_Toc135236102)

[2.4 Üzenetek küldése 11](#_Toc135236103)

[2.5 További követelmények 12](#_Toc135236104)

[3 Architektúrák 13](#_Toc135236105)

[3.1 Rendszer architektúra 13](#_Toc135236106)

[3.2 Szerveroldali alkalmazás 13](#_Toc135236107)

[3.3 Adatbázis felépítése 13](#_Toc135236108)

[3.3.1 AspNetUsers 14](#_Toc135236109)

[3.3.2 Ads 14](#_Toc135236110)

[3.3.3 Messages 14](#_Toc135236111)

[3.4 Kliensalkalmazás 14](#_Toc135236112)

[4 Irodalomjegyzék 15](#_Toc135236113)

[Függelék 16](#_Toc135236114)

Hallgatói nyilatkozat

Alulírott **Bui Quanganh Krisztián**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző, cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2023. 05. 04.

...…………………………………………….

Bui Quanganh Krisztián

Összefoglaló

A szakdolgozat, vagy diplomaterv elkészítése minden egyetemi hallgató életében egy fontos mérföldkő. Lehetőséget ad arra, hogy az egyetemi évei során megtanultakat kamatoztassa és eredményeit szélesebb közönség előtt bemutassa, s mérnöki rátermettségét bizonyítsa. Fontos azonban, hogy a dolgozat elkészítésének folyamata számos csapdát is rejt magában. Rossz időgazdálkodás, hiányos szövegszerkesztési ismeretek, illetve a dolgozat készítéséhez nélkülözhetetlen „műfaji” szabályok ismeretének hiánya könnyen oda vezethetnek, hogy egy egyébként jelentős időbefektetéssel készült kiemelkedő szoftver is csak gyengébb minősítést kapjon a gyenge minőségű dolgozat miatt.

E dokumentum – amellett, hogy egy általános szerkesztési keretet ad a dolgozatodnak – összefoglalja a szakdolgozat/diplomaterv írás írott és íratlan szabályait. Összeszedjük a Word kezelésének legfontosabb részeit (címsorok, ábrák, irodalomjegyzék stb.), a dolgozat felépítésének általános tartalmi és szerkezeti irányelveit. Bár mindenkire igazítható sablon természetesen nem létezik, megadjuk azokat az általános arányokat, oldalszámokat, amelyek betartásával jó eséllyel készíthetsz egy színvonalas dolgozatot. A részletes és pontokba szedett elvárás-lista nem csupán a dolgozat írásakor, de akár más dolgozatok értékelésekor is kiváló támpontként szolgálhat.

Az itt átadott ismeretek és szemléletmód nem csupán az aktuális feladatod leküzdésében segíthet, de hosszútávon is számos praktikus fogással bővítheti a szövegszerkesztési és dokumentumkészítési eszköztáradat.

Abstract

Ide jön a ½-1 oldalas angol nyelvű összefoglaló, amelynek szövege a Diplomaterv Portálra külön is feltöltésre kerül. Ez a magyar nyelvű összefoglaló angolra fordított változata.

# Felhasznált technológiák

## .NET

A .NET (korábban ismert nevén .NET Core) egy ingyenes és nyíltforráskódú szoftverkeretrendszer, amit 2016-ban mutatott be a Microsoft. A .NET Framework jelentősen átdolgozott cross-platform utódja, így futtatható Windows, Linus és MacOS rendszereken egyaránt.

A .NET alkalmazások magja a robosztus és felügyelt közös nyelvi futtatókörnyezet (Common Language Runtime - CLR). Többféle szolgáltatást nyújt, ilyen a memóriakezelés a szemétgyűjtő segítségével, kivételkezelés, típusellenőrzés és szálkezelés. A CLR biztosítja továbbá a különböző programozási nyelvek együttműködését, ezért egy alkalmazás könnyedén felhasználhatja a különböző .NET által támogatott nyelveken írt könyvtárakat.

A .NET környezet nyelvfüggetlen, C#, Visual Basic, illetve az F# mellett számos objektumorientált programozási nyelvet támogat, a különböző nyelveken készült osztályok egymásból örökölhetnek. Az alkalmazás fordításakor első sorban IL (Intermediate Language) kód készül, mely több architektúrára és operációs rendszerre fordítható, majd a platformtól függően natív kód áll elő, ami Just-in-time vagy Ahead-of-time módon fut le.

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás

Ábra : Kód fordítása .NET környezetben.

A keretrendszer moduláris felépítésű és csak alapvető osztályokat tartalmaz, így a környezet kis méretű marad, az alkalmazás pedig csak a használt könyvtárakat tartalmazza. További komponenseket a NuGet csomagkezelő rendszeren keresztül tudunk hozzáadni.

## ASP.NET Core

Az ASP.NET Core egy ingyenes, nyíltforráskódú keretrendszer, melyet modern web alapú alkalmazások készítésére fejlesztettek ki. A keretrendszer segítségével készíthetünk szerver oldalon renderelt multi-page alkalmazásokat (Razor pages), single-page alkalmazásokat (React, Angular) vagy REST (Representational State Transfer) API (Application Programming Interface) szolgáltatást, amelyt használhatunk tetszőleges kliensoldali technológia alkalmazásához.

Az ASP.NET támogatja a függőséginjektálást (Dependency injection) tervezési mintát. A minta lehetővé teszi azt, hogy egy osztály és azon függőségei között laza csatolás jöjjön létre azáltal, hogy az osztály nem függ az implementációtól. Az osztály a függőség szolgáltatásaihoz egy interfészen keresztül tud hozzájutni. Az esetleges függőségeket a Program.cs fájlban tudjuk rögzíteni, és a keretrendszer az osztály konstruktorán keresztül adja át a megfelelő függőségeket. Az injektált függőségeknek többféle életciklust tudunk beállítani: Transient esetén minden használatkor új objektum jön létre, ami többszálú alkalmazásban előny lehet; a Scoped objektum egy kérésen belül ugyanaz marad; a Singleton objektum esetén a kérések ugyanazt az egy objektumot használják.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

Ábra : A életciklusok típusainak működése

## ASP.NET Identity

Az ASP.NET Identity a Microsoft által biztosított felhasználóikezelő rendszer, amely biztonságos hitelesítési és authorizációs funkciók kiépítésére szolgál az ASP.NET alkalmazásokban. Az ASP.NET Core keretrendszer része, amelyet a regisztráció, a bejelentkezés, a jelszókezelés és a szerepek kezelésére terveztek.

Az ASP.NET Identity több hitelesítési mechanizmust is támogat, használhatunk külső bejelentkezés szolgáltatókat, például Facebook, Google vagy Microsoft felhasználókat, vagy eltárolhatjuk a bejelentkezési adatokat az Identity rendszer segítségével és használhatunk cookie-alapú és token-alapú hitelesítést JSON Web Token (JWT) segítségével.

A hitelesítés és authorizáció egyik legfontosabb része a biztonság. Mivel én ebben a területben nem vagyok elég jártas, a felhasználókezelő saját implementálása helyett egy előre megírt és jól dokumentált megoldás mellett döntöttem. Az ASP.NET Identity és az Entity Framework Core szorosan integrálódik egymással, ezért ennek a könyvtár használatát választottam.

## Entity Framework Core

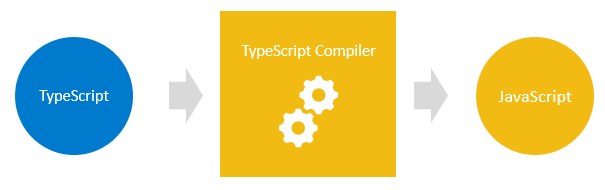
Az Entity Framework Core (EF Core) egy Microsoft által létrehozott objektum-relációs leképező könyvtár. A szoftver platformfüggetlen, használhatjuk Windows, MacOS és Linus operációs rendszeren, továbbá kompatibilis a legtöbb adatbáziskezelő rendszerrel, például a Microsoft SQL Server-hez, a MySQL-hez és a PostgreSQL-hez. Ennek használatának segítségével függetleníteni tudjuk alkalmazásainkat az adatbázismotortól.

Az EF Core egyik fő funkciója a code-first leképezési módszer. A fejlesztőnek van lehetősége arra, hogy elsődlegesen az alkalmazás objektummodelljét definiálja majd a keretrendszer segítségével hozza létre a kapcsolódó adatbázis sémát. A könyvtár egy migrációs rendszert is tartalmaz, ami az adatsémában való változtatások lebonyolítását segíti az adatok elvesztése nélkül.

Az EF Core és a LINQ (Language Integrated Query) rendszer együttműködésével könnyen tudunk olvasható adatbázis lekérdezéseket írni C# szintaxis használatával. A LINQ típusbiztonságot is biztosít, és olyan objektumokon is használhatjuk, amik implementálják az IEnumerable és az IQueryable interfészeket.

## TypeScript

A TypeScript egy magasszintű, objektum-orientált, böngészőkben használt, nyíltforráskódú programozási nyelv, ami a JavaScript kiterjesztése. A TypeScript fordító JavaScript kódot hoz létre, ezért minden JavaScript-et támogató környezetben futtatható, és minden JavaScript kód szintaktikailag érvényes TypeScript kód is. A nyelv funkciói jelentősen javítja a kód karbantarthatóságát és olvashatóságát.



Ábra : A fordító TypeScript kódból JavaScript kódot készít.

A TypeScript statikus típusosságot biztosít, így lehetővé teszi a típusellenőrzést fordítási időben. Ez jelentősen megkönnyíti a fejlesztést, mivel a legtöbb futási idejű hiba kiderül fordításkor, továbbá valódi IntelliSense támogatást is nyújt. A nyelv segítségével saját osztályokat és interfészeket is létrehozhatunk, melyeknek adattagokat és függvényeket adhatunk. Strukturális típusosság jellemzi, vagyis egy objektum strukturálisan kompatibilis egy másik objektummal, ha tartalmazza a publikusan elérhető tagváltozóit és függvényeit.

Alapértelmezetten a változók felvehetik a null és undefined értékeket, melyek további futási idejű hibához vezethetnek. A fordító --strictNullChecks flag beállítása esetén a fordító típushibát dob, ha egy típusos változónak az előbb említett értékek egyiket adjuk meg.

A TypeScript modulok használatát is biztosítja, ezek segítségével összefüggő osztályokat, függvényeket, változókat tudunk összefűzni egy logikai fájlba. Alapesetben egy modul tartalma csak exportálás után láthatóak kívülről, és minden használni kívánt elem importálás után használhatóak.

## React

A React egy ingyenes, nyíltforráskódú, JavaScript keretrendszer, melyet a Meta (korábban Facebook) fejlesztett ki. A könyvtár segítségével felhasználói felületet tudunk létrehozni. A felület leírását deklaratív módon tudjuk leírni, a keretrendszer automatikusan frissíti és újrarendereli a komponenseket a fájlok változtatásakor, ez jelentősen felgyorsíthatja a fejlesztést és megkönnyíti a hibakeresést. A keretrendszerhez mellékeltek típus fájlokat is, ezért TypeScript projektben is könnyedén használható.

A React segítségével komponens alapú felhasználói felületet tudunk létrehozni. A komponenseket kétféleképpen tudjuk leírni: osztálykomponensként vagy függvénykomponensként.

## Chakra UI

# Funkcionális követelmények

## Böngészés

Az egyik nélkülözhetetlen funkció az alkalmazásban a létrehozott hirdetésének böngészése. A felhasználó bejelentkezés nélkül csak ezt a funkciót éri el. A hirdetések egy listán keresztül tekinthetőek meg, melynek szűrési lehetőségeket kell biztosítania. A szűrési feltételnek megadható az ingatlan címe, ára, mérete és a szobák száma. A listanézetben a hirdetések száma korlátozott a könnyebb átláthatóság és navigálás érdekében.

Ha egy hirdetés felkeltette a felhasználó érdeklődését, a listában lévő hirdetésre kattintva meg kell jelennie az ingatlan részletes leírásának. Ennek a nézetnek egy külön oldalon kell megjelennie, a részletek között meg kell jelennie az ingatlan adatainak, egy róla készült fotó és a hirdető elérhetőségeinek. A bejelentkezett felhasználónak van lehetősége felvenni a kapcsolatot az alkalmazáson keresztül is.

## Regisztrálás és bejelentkezés

Ahhoz, hogy elérje az alkalmazás további funkcióit, a felhasználónak rendelkeznie kell egy saját fiókkal, amit a regisztrációs ablakon tud létrehozni. Regisztráció esetén meg kell adni egy érvényes e-mail címet, felhasználónevet, megfelelő erősségű jelszót és telefonszámot. A regisztráció és bejelentkezés után elérhetővé válik a hirdetés készítése és az üzenetek küldése.

## Hirdetés létrehozása és kezelése

A másik fontos funkció az alkalmazásban az ingatlan hirdetések létrehozása. A hirdetés létrehozásakor meg kell adni a hirdetés címét, a lakás címét, méretét, szobák számát, egy róla elkészült képet és az árát. A kliensnek és a szervernek validációt kell elvégeznie a kérés feldolgozása előtt. A képek eltárolása az adatbázisban történik, amit Base64 formájú szövegként tárolunk el. A hirdető eltávolíthatja a már nem aktuális hirdetést vagy módosíthatja a már elavult információkat.

## Üzenetek küldése

## További követelmények

A felhasználó felületnek letisztultnak, reszponzívnak és intuitívnak kell lennie. A kliensnek a felhasználó legtöbb műveletére reagálnia kell, a hosszabb műveletek esetén aszinkron működést kell biztosítani, hogy ne fagyjon le a felhasználói felület. Az esetleges hibákat a kliensnek jeleznie kell.

# Architektúrák

## Rendszer architektúra

## Szerveroldali alkalmazás

## Adatbázis felépítése

Az adatbázis kezeléséhez Microsoft SQL adatbázisszervert alkalmaztam. A tervezéskor Code first megközelítést alkalmaztam az Entity Framework Core segítségével, a táblázat szerkezete követi a Domain rétegben definiált modellek szerkezetét. Az adatbázis szerkezet az alábbi ábrán látható.



Ábra : Az adatbázis sémája

Az ASP.NET Core Identity több felhasználóval kapcsolatos táblázatot is generált, de azoknak a szerepe elhanyagolható, ezért nem szerepelnek. A fent lévő táblázatokat tekintsük át.

### AspNetUsers

A felhasználó fontos adatait tartalmazó táblázat. Az adatok között szerepel a felhasználó e-mail címe, teljes neve, felhasználóneve, jelszava és opcionálisan a telefonszáma. A további, fel nem sorolt attribútumok az ASP.NET Core Identity működéséhez szükséges. A jelszó hashelt formában érhető el.

### Ads

Ebben a táblázatban az ingatlanok hirdetésének adatait tárolom el. Szerepel benne a hirdetés címe, leírása, létrehozási ideje, tulajdonos idegen kulcsa, az ingatlan, címe, ára, mérete és a róla készült kép Base64 formátumban.

### Messages

A felhasználók közötti üzenetváltásokat tárolom ebben a táblázatban. A küldött üzenet tartalmát, idejét, olvasottságát, feladóját és címzettét tárolja el.

## Kliensalkalmazás

# Irodalomjegyzék

|  |  |
| --- | --- |
| [1] | K. Nahtkasztlija, „Az idegen szavak toldalékolása,” június 2009. [Online]. Available: http://www.pcguru.hu/blog/kredenc/az-idegen-szavak-toldalekolasa/5062. |
| [2] | P. Koopman, „How to Write an Abstract,” október 1997. [Online]. Available: https://users.ece.cmu.edu/~koopman/essays/abstract.html. [Hozzáférés dátuma: 20 október 2015]. |
| [3] | W3C, „HTML, The Web’s Core Language,” [Online]. Available: http://www.w3.org/html/. [Hozzáférés dátuma: 20 október 2015]. |

Függelék