Eötvös Loránd Tudományegyetem

Informatikai Kar

Informatikatudományi Intézet

Programozáselmélet és Szoftvertechnológia Tanszék

Autós Portál

Szerző:

Vass-Horváth Balázs

Programtervező informatikus BSc.

Témavezető:

Dr. Gludovátz Attila

egyetemi adjunktus

Szombathely, 2024

Tartalomjegyzék

Tartalom

1. Fejlesztői dokumentáció

1.1 Felhasznált technológiák, programozási nyelvek

1.1.1 C# - ASP.NET keretrendszer

A Microsoft által fejlesztett .NET keretrendszer napjaink egyik legelterjedtebb, legismertebb, leggyakrabban használt keretrendszere. Népszerűsége betudható többek között annak, hogy mára már platformtól függetlenül is fejleszthetünk benne alkalmazásokat, valamint a nyelv fejlesztői hihetetlenül részletes, mintakódokkal teli dokumentációt biztosít a weboldalán. A C# programozási nyelv a .NET keretrendszer része. Fontos kiemelni, hogy teljes mértékben objektum orientált nyelvről beszélünk, melynek alapjai a C++, és Java nyelvek. Mára már elérhető a nyelv 11.0-s verziója is, valamint a keretrendszer 7.0-s verziója. A hozzá tartozó legismertebb, leggyakrabban használt fejlesztői környezetként a Microsoft által fejlesztett Visual Studiot szokták emlegetni. Fontos kiemelni, hogy rengeteg kiegészítő, úgynevezett „Nuget Package" könyvtár érhető el hozzá, mely lényegesen leegyszerűsítheti a munkánkat. A legismertebb csomagok közé tartozik például az adatbáziskezelést leegyszerűsítő „Entity Framework", vagy a „Newtonsoft JSON" elnevezésű package, mely elősegíti a JSON formátumú adatok kezelését. Webes alkalmazások készítéséhez az ASP.NET-et használhatjuk, amely napjainkra rengeteg féle alkalmazás létrehozását támogatja (Pl.: Web API, Razor Web, MVC Web App), de akár olyan népszerű Frontend keretrendszerekkel is társíthatjuk, mint például a React JS, vagy az Angular.

1.1.2 MySql 8.0

A MySql az Oracle Corporation által fejlesztett SQL alapú relációs adatbázis szerver. A MySql mára az egyik legelterjedtebb adatbázis kiszolgáló olyan okok miatt, mint például a nyílt forráskód, a megbízhatóság, a stabilitás, valamint az, hogy platform független, tehát elérhető többek között Windows, Linux, MacOS X, FreeBSD, és szinte minden további operációs rendszeren. Az adatbázis kezeléséhez szükségünk van továbbá egy adatbázis kezelő szoftverre, mint például a MySql Workbench. MySql adatbázis kezelő szoftverek is szinte minden operációs rendszerre léteznek, még Androidra, vagy IOS-re is. Alternatívájaként érdemes megemlíteni az MSSQL-t, amely hasonló hatékonysággal képes ellátni az adatbázis feladatokat.

1.1.3 HTML

A HTML, avagy Hyper Text Markup Language egy weboldalak készítésére szolgáló leíró nyelv. A HTML jelenleg legfrissebb verziója a HTML 5. A HTML önmagában nem alkalmas komplex weboldalak létrehozására azon tulajdonsága miatt, hogy csak a weboldal megjelenéséért felel. A weboldalak formázásáért a CSS felelős, míg a kliens oldalon futó funkciókat jellemzően JavaScript segítségével oldhatjuk meg. Weboldalak esetén a szerveroldali funkciók kiszolgálására napjainkban már rengeteg nyelv biztosít lehetőséget, de a legismertebbek talán a PHP, a C# (ASP.NET keretrendszer), Java (Spring keretrendszer), Python, Node JS.

1.1.4 JavaScript, jQuery

A JavaScript napjaink egyik legelterjedtebb weboldalakon használt script nyelve. Fontos tulajdonsága, hogy kliens oldalon fut le a kód, melyet a böngésző futtat, azaz a felhasználó által is kiolvasható. Ezen tulajdonságának betudható, hogy komplex alkalmazás fejlesztésére önmagában nem alkalmas, mivel nem futtatható szerver oldalon. Az elnevezése kisség megtévesztő lehet, mivel bár a szintaktikája kissé hasonló a Java-hoz, de semmilyen kapcsolatban nincs a két nyelv. A jQuery egy JavaScript könyvtár, melynek népszerűsége betudható annak, hogy gyorsan, egyszerűen használható, és rengeteg hasznos funkciót, eseménykezelést tartalmaz, melyeket így már JavaScriptben nem kell nekünk elkészíteni. Rengeteg publikus, más programozók által készített jQuery kód érhető el az interneten, melyekkel például bemeneti mezőket, és egyéb, a felhasználó számára látható elemeket tehetünk látványosabbá, használhatóbbá. A jQuery használatához script-ként be kell importálnunk azt az oldalunkra, viszont ez alapból elérhető az ASP.NET projektekben, így ezzel már nem kell foglalkoznunk. Le is tölthetjük, valamint hivatkozhatunk online elérhető verziójára is. Az online verzió mellett szól az, hogy naprakész, viszont hátránya lehet a rendelkezésreállás. További előnye lehet, hogy igény esetén Ajax segítsége mellett egyszerűen használhatjuk API-val.

1.1.5 CSS, Bootstrap

A CSS, vagy teljes nevén Cascading Style Sheets egy stílusleíró nyelv, mely főleg weboldalak esetén használatos. Célja, hogy a weboldalunk tartalmát és kinézetét elválassza egymástól. A CSS segítségével több módon is lehetőségünk van az oldalunk kinézetének csoportosítására. Hivatkozhatunk egy tag azonosítójára (id), nevére (name), vagy akár típusára is. Továbbá lehetőség van egy adott elem közvetlen, beágyazott stílus módosítására is, ám ez nem ajánlott, ugyanis sokkal olvashatatlanabb kódot eredményez, és nem utolsó sorban nem tudjuk több

helyen is használni. A .NET keretrendszer által létrehozott projektekhez automatikusan társul a Bootstrap aktuális verziója. A Bootstrap egy nyílt forráskódú, ingyenes CSS keretrendszer. Alternatíva híján napjainkban a legelterjedtebb front-end keretrendszer. Első körben weboldalak kinézeteinek kialakításához használatos. Napról-napra növekszik a Bootstrap alapú sablonok száma, melyeket web designerek készítenek. Található ezek között ingyenes, valamint fizetős verzió is, de lehetőségünk nyílik saját sablon létrehozására is. Továbbá jelentős, hogy könnyen személyre szabható, és átlátható, erre azonban nem minden esetben van szükség, mivel alapból található benne a legtöbb webfejlesztés során használatos elemre (Például gombok, formok, beviteli mezők) megfelelő kinézet. Ezen kinézeteket leggyakrabban úgy használhatjuk fel, hogy az adott elemre alkalmazzuk a számunkra szükséges CSS osztályt. A Bootstrap használata mellett a legnagyobb érv a reszponzivitás. Reszponzív oldalról akkor beszélhetünk, amikor az azt megjelenítő kijelző méretétől, felbontásától, arányaitól függetlenül az alkalmazás úgy jelenik meg, hogy minden eleme megfelelően látszódjon, valamint olvasható és használható maradjon. Az előzőleg említett tulajdonság az egyik legfontosabb elvárás napjainkban a weboldalaknál, mivel a felhasználók egyre változatosabb eszközöket használnak változatos kijelző specifikációkkal kezdve a nagyfelbontású monitoroktól, az átlagos/nagyfelbontású mobiltelefonokon, okostelevíziókon, táblagépeken keresztül az alacsony felbontású okosórákig.

1.1.6 GitHub

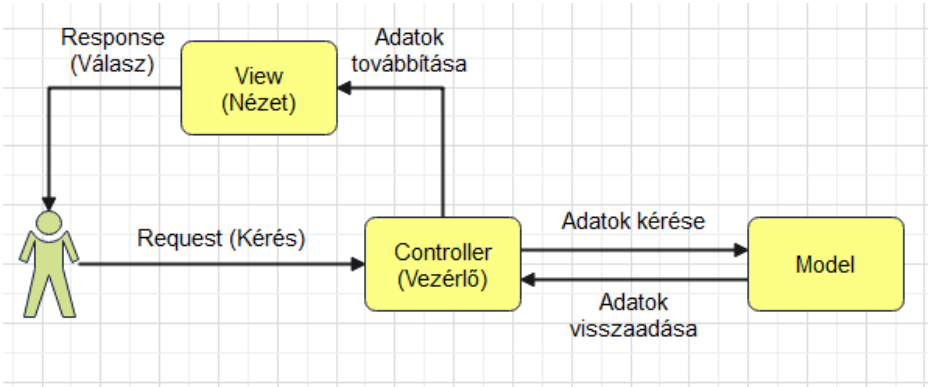
A GitHub a Microsoft egy leányvállalataként a verziókezelés, verziókövetés lehetőségét biztosítja. A diplomamunkám elkészítése közben számomra fontos volt, hogy szükség esetén az otthoni számítógépemen kívül máshonnan is elérhessem, fejleszthessem a kódomat. Ezen felül lehetőséget biztosít számunkra, hogy egy esetleges hibás fejlesztést követően visszaállítsuk a kódot egy régebbi verziójára, és nem utolsó sorban egy esetleges adatvesztés következtében is vissza tudom nyerni a diplomamunkámat. Segítségével nyomon követhetem, hogy miként fejlődik a program, mikor milyen fejlesztéseket, újításokat építek bele. Csoportos munkák esetén további létfontosságú funkciók is megnyílnak előttünk. Lehetőséget biztosít arra, hogy az általunk létrehozott projektet munkatársaink is elérhessék, megtekinthessék, módosíthassák jogosultságuktól függően. Ebben az esetben gyakran külön branch-et, úgynevezett munkaasztalt kapnak a fejlesztők, és a saját módosításaikat oda töltik fel, míg a fő munkaasztalt csak arra külön kijelölt személyek érik el, ők egyesítik a fejlesztők munkáit. Ez a lehetőség nagyvállalati környezetben elengedhetetlen. Bár legfőképpen forráskódjainkat tároljuk itt, de lehetőség van például dokumentáció, Wikipédia, integrációs könyvtárak (LIB-

ek), és még rengeteg virtuális adat tárolására, nyomon követésére. A GitHub továbbá lehetőséget nyújt számunkra kisebb weboldalak tárolására, valamint kódjaink publikálására, és még megannyi hasznos funkcióra, melyeket szinte a végtelenségekig lehetne sorolni. Alternatívaként rengeteg lehetőséget felsorolhatnék, de talán a legismertebb az azonos alapokra épülő GitLab, valamint rengeteg cég mára már saját Git alapú verziókövető rendszert üzemeltet, és használ.

1.1.7 Model-View Controller (MVC)

A Model-View-Controller, vagy rövidítve MVC napjaink egyik legnépszerűbb, és leghasználgatóbb (jellemzően webes) program tervezési mintája. Célja, hogy az alkalmazásunkat három fő komponensre bontjuk, melyek a modell, a nézet (view), és a vezérlő (controller). Legfőbb előnyei többek között a párhuzamos fejlesztési lehetőség, vagy éppen a modularitás. Az egyes rétegek feladatai:

- Modell: Jellemzően az alkalmazás által használt adatok strukturált (osztályokba szervezett) tárolásáért, eléréséért felel.
  Esetünkben az alkalmazás tartalmaz Adatbázis modelleket (DBModels), melyek az adott modellhez tartozó adatbázis-tábla szerkezetét írja le, és ennek segítségével Entity Frameworkön keresztül könnyen összekapcsolhatjuk az alkalmazásunkat és az adatbázisunkat.
  Ezen túl az alkalmazás tartalmaz Request, és Response modelleket, melyek a Nézet- és vezérlő rétegek közötti rendszerezett adatcserét biztosítja.
- Nézet: A nézet réteg felel az alkalmazásunk megjelenéséért, valamint a felhasználóval történő „kapcsolattartásért”. A nézeten jelenítjük meg az adatokat a felhasználó számára, valamint ezen keresztül küld kéréseket a felhasználó a Vezérlő rétegnek (Pl.: egy gomb megnyomásának hatására adatokat küldünk a szervernek).
- Vezérlő: Mint az a nevéből is adódik, az alkalmazásunk vezérléséért felel. Ez a (webes alkalmazások esetén szerveroldali) réteg felel azért, hogy feldolgozza a nézet rétegtől kapott (felhasználó által kiváltott) kéréseket, valamint a kérésekre valamilyen válasszal reagáljon. Ez a válasz webes alkalmazás esetén lehet egy elérési út (Nézet) visszaadása, vagy adatok visszaadása a megjelenítő számára, esetleg valamilyen hibakód.



1. ábra MVC felépítés

1.2 Felhasznált szoftverek, fejlesztői környezetek

1.2.1 Visual Studio 2022

A Visual Studio a Microsoft több programozási nyelvet támogató, valamint az évek során folyamatosan fejlődő fejlesztői környezete. Lehetőséget biztosít többek között C#, C++, F#, Visual Basic, Python és még néhány további nyelven történő alkalmazás fejlesztésre. Ezeken túl több Leíró (Pl.: XML, HTML), és Script (Pl.: JS) nyelvet, kliens oldali keretrendszert (Angular, React), valamint JSON formátumot is képes felismerni. A fejlesztői környezet első verzióját 1997-ben mutatták be, viszont az évek során rengeteg verzió jelent meg (jellemzően 3-4 évente), napjainkban a 2022-es a legfrissebb. A .NET keretrendszerben történő fejlesztés először a 2002-es verzióban jelent meg.

1.2.2 MySql Server 8.0

A MySql Server 8.0 verziója az egyik lehető legjobb választás, ha valamilyen komplex alkalmazáshoz szeretnénk adatbázis kiszolgálót. Legnagyobb előnye, hogy platformfüggetlenség mellett ingyenesen elérhető szinte minden elterjedt operációs rendszerre. Az adataink tárolása mellett lehetőséget biztosít kényszerek használatára is, mely segítségével tovább csökkenthetjük annak esélyét, hogy hibás adatok kerüljenek be az adatbázisunkba. További lehetőségünk van adatbázis programozásra is, mint például triggerek, és tárolt eljárások készítése.

1.2.3 MySql Workbench

A MySql Workbench az Oracle által fejlesztett grafikus MySql adatbáziskezelő szoftver. Előnye, hogy ingyenes, és ezen túl Windows, Linux, valamint OS X rendszerekre is elérhető. Továbbá célszerű választás lehet azért is, mivel az Oracle cég fejleszti, akik a MySql fejlesztői is, ezáltal feltehető, hogy folyamatosan naprakész verziót használhatunk. Az alkalmazás telepítését a MySql Server 8.0 telepítője is felajánlja, de külön is letölthetjük a mysql hivatalos oldaláról. Segítségével nem csak az adatbázisainkat kezelhetjük, de a MySql szerverünket is, továbbá rendelkezik például teljesítmény elemzéssel, és még sok más beépített funkcióval. Hátrányaként említeném, hogy a felhasználói felülete számomra elmarad a legtöbb fizetős adatbáziskezelő rendszerével szemben.

1.2.4 GitHub Desktop

A GitHub Desktop a GitHub saját fejlesztésű szoftvere. Célja, hogy a GitHub-on kezelt munkáinkat könnyebben, átláthatóbban elérhessük. Segítségével megkönnyíthetjük az életünket, ugyanis rengeteg GitHub CLI-ben kiadott parancsot válthatunk ki csupán néhány kattintással. Az alkalmazás elérhető Windows, valamint MacOS rendszerekre is. Lehetőséget biztosít a verziókövetés nyújtotta előnyök szinte minden lehetőségének kihasználására, mint például a projectünk klónozása, módosítások feltöltése, vagy épp korábbi verziók visszaállítása. Bár mára már a Visual Studio is biztosítja a verziókövetést valamilyen szinten, funkcionalitása elmarad a GitHub Desktop apptól. Itt kiemelném, hogy a GitHub rendelkezik mobilos alkalmazással is, mely elérhető Android, és IOS eszközökre is.

1.2.5 Microsoft Windows 10

A fejlesztéshez használt eszközeimen Microsoft Windows 10 operációs rendszert használok, mely napjaink egyik legelterjedtebb operációs rendszere. Folyamatos biztonsági, és funkcionális frissítéseknek köszönhetően garantálja a felhasználói élményt, és a naprakész biztonságot. Mára elérhető a Windows 11 operációs rendszer is, ám mivel egy viszonylag friss rendszerről beszélünk, ezért még előfordulhatnak kompatibilitási, vagy stabilitási problémák.

1.2.6 IIS Web Szerver

Az alkalmazás futtatásához IIS Web Szervert használtam, mely a minden újabb Windows operációs rendszeren alapból elérhető, csupán aktiválni kell. Segítségével minden ASP.NET alatt készített alkalmazást futtatni tudunk, és mint minden más fejlett web szerver, ez is lehetőséget ad számunkra minden fontos biztonsági funkció használatára. Napjainkban a legnépszerűbb web szerver kiszolgálók közé tartozik, ezért a folyamatos frissítések garantálják a stabilitást és a naprakész biztonságot. Tovább könnyíti a dolgunkat, hogy az aktiválását követően a windows frissítésekkel együtt a web szerverünk is automatikusan kapja a frissítéseket. A fejlesztés folyamán is találkoztam már a szerver frissítésével, és semmilyen problémát nem okozott a futtatott alkalmazások üzemelésében, ezért ettől sem kell tartanunk.

1.2.7 Google Chrome

A Google Chrome egy ingyenesen elérhető webböngésző, mely többek között elérhető Windows, Linux, MacOS, Android, és IOS rendszerekre is. A felhasználók előszeretettel használják, mert felhasználóbarát kezelőfelülettel rendelkezik, és végtelenül egyszerűen használható. További pozitívum, hogy számos bővítmény érhető el hozzá, hátrányként viszont érdemes megemlíteni a konkurens böngészőkhöz képest valamivel magasabb memóriahasználatot.

1.3 Alkalmazásterv

1.3.1 Témaválasztás, alkalmazás funkcionális leírása

Nagyon sok időt töltöttem azzal, hogy a diplomamunkámnak szánt alkalmazás témáját kitaláljam. Rengeteg ötletem támadt, ám végül egy olyan témájú app mellett döntöttem, amely az informatika után talán a legközelebb áll a szívemhez: az autózás/autóipar. Már kisgyerek koromban is rengeteget foglalkoztam a témával, ám akkor még csak játékként tekintettem rá: Autós kártyák, Hot Wheels, később lego autók, autóversenyzős számítógépes játékok, és persze a kedvenc mesefilmem is a verdák című animációs film volt. Nem is volt kérdés, amint lehetett, 17éves koromban megszereztem a jogosítványt, azóta pedig csak még tovább nőtt az autózás iránti rajongásom, szinte nem telik el úgy nap, hogy ne ülnék kormány mögé. Mivel a választott témakörben szintén rengeteg lehetőség rejlik, főleg napjainkban, amikor az ipar, és az autózás is hatalmas változásokon megy keresztül, ismét elkezdett pörögni az agyam: Milyen célt szolgáljon az alkalmazásom? – Önvezető autó szimuláció? Autóversenyzős játék?

Végül egy ezeknél sokkal célszerűbb ötletem támadt: Egy gépjármű nyilvántartó alkalmazás. Mint azt már írtam, az autózás, és ezáltal az autóipar hatalmas nehézségeken, és változásokon megy át napjainkban. Éppen, hogy kezdett lecsengeni a COVID-19 nevű világjárvány, egy újabb nehézséget kell áthidalni: Kitört az Orosz-Ukrán háború, mely hatalmas inflációt eredményezett világszinten. Tapasztalható, hogy jelentősen megváltoztak az emberek autózási- és autóvásárlási szokásai is ebben a válságközeli helyzetben: A legtöbben sokkal költséghatékonyabban próbálnak közlekedni, és a használtautó piac egyre jobban maga alá szorítja az újautó piacot.

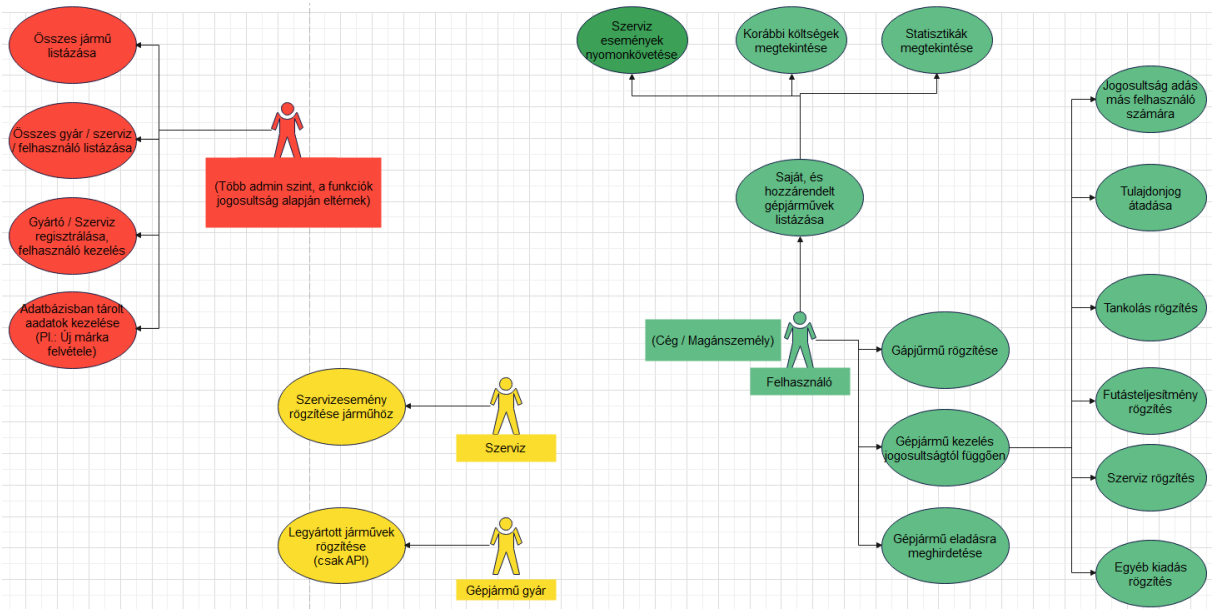
Az általam kigondolt alkalmazás célja, hogy elősegítse a költséghatékony gépjármű üzemeltetést. A felhasználók számára lehetőséget biztosít, hogy rögzítse a gépjárművének költségeit, mint például a tankolás, a szervizek, vagy bármilyen további, a felhasználó által megadott költség. Ezeken túl lehetőséget biztosít, hogy nyomon kövessük a gépjármű használati szokásainkat úgy, hogy rögzíteni tudjuk az aktuális futásteljesítményét, illetve a tankolásokat.

Mindazonáltal gépjárműveket nem csak magánszemélyek használnak, sőt a járműhasználat csupán töredékét teszik ki magánszemélyek. Az alkalmazás éppen ezért sokkal inkább a vállalkozásokat célozná meg: Lízing cégek, Személy- és áruszállítással foglalkozó cégek, futárszolgálatok, vagy bármilyen cégautó(ka)t tartó vállalkozások. Lehetőségünk van korlátlan számú gépjárművet rögzíteni a rendszerbe, melyeknek teljes életciklusát lekövetné az

alkalmazás a vásárlástól egészen az eladásig. Miután a céges autókat jellemzően nem csak egy személy vezeti, ezért az alkalmazás lehetőséget biztosít arra, hogy egy járművet a tulajdonoson kívül több felhasználóhoz is kirendeljen, akik jogosultságuktól függően (Tulajdonos, Üzembentartó, Sofőr) érhetik el az alkalmazás bizonyos funkcióit a járműre nézve.

Az alkalmazás főbb tervezett funkciói:

- Autógyártó cégek számára: API-n keresztül lehetőséget biztosít a legyártott gépjárművek rögzítésére a rendszerbe.
- Autószervizek számára: A náluk szervizelt gépjárművekre rögzíteni tudják az elvégzett szervizek adatait (szerviz időpont, elvégzett munka, költségek, egyéb megjegyzések/észrevételek)
- Cégek/Vállalkozások számára: A cég gépjárműveinek kezelése, nyilvántartása; Jármű hozzárendelése felhasználó(k)hoz; Új jármű(vek) rögzítése; Jármű(vek) eladásra meghirdetése, használt járművek keresése, eladásra kínált jármű rögzített adatainak megtekintése; Sikeres eladás esetén az új tulajdonosnak átadhatja a járművet, aki így tovább tudja vezetni a jármű eseményeit.
- Magánszemélyek számára: A cégekhez hasonló funkciók korlátozott elérése (Pl.: Limitált számban rögzíthet csak járműveket, vagy épp limitált számban rendelheti felhasználóhoz azt), a hozzá rendelt járművek kezelése jogosultságtól függően (Tulajdonos; Üzembentartó/Másodlagos tulajdonos; Sofőr).
- Admin jogosultság: Természetesen az alkalmazás rendelkezik jogosultságkezeléssel, ugyanis bizonyos esetekben előfordulhat, hogy adminisztrátori közbenjárásra van szükség. Ilyen például a kereskedők / szervizek regisztrációjának megerősítése, vagy a gyártók regisztrálása. Ezen túl még rengeteg felhasználó-támogató funkcióval rendelkezik, mint például a járművek tulajdonjogának átírása, valamint felhasználó támogató szerepet is betöltenek.



2. ábra Követelmény specifikáció

1.3.2 Tervezett erőforrásigény

Az alkalmazás üzemeltetéséhez szükséges erőforrás-igény megbecsülése talán az egyik legnehezebb feladat. Fejlesztés során ügyeltem az optimális erőforrás-felhasználásra, ezzel is csökkentve a szükséges hardware erőforrást. Mivel webes alkalmazásról beszélünk, ezért az első, és legfontosabb kitétel a stabil, nagy sávszélességű dedikált internetkapcsolat. Nem elhanyagolható továbbá a folyamatos rendelkezésreállás se, ezért javasolt szünetmentes tápegység használata is. Az alkalmazás zár körű (akár céges környezetben) történő használatához elegendő csupán egy átlagosnak nevezhető szervergép (Windows Server operációs rendszerrel számolva):

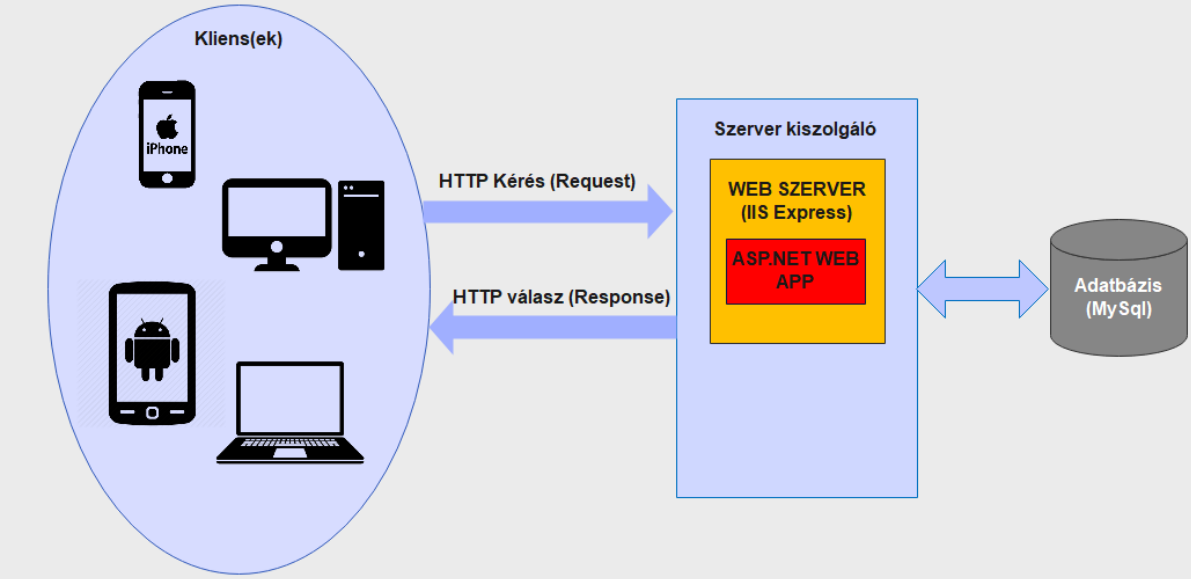
- RAM: 8GB belső memória
- CPU: min. 1.8GHz 4mag
- Tárhely: 25GB belső tárhely

Mindazonáltal amennyiben az alkalmazást rendeltetésének megfelelően, globális környezetben szeretnénk használni, abban az esetben egy komplett szerver farmra lehet szükségünk.

1.4 Alkalmazás felépítése, működése

1.4.1 Felépítés

Az alkalmazást három fő rétegre bonthatjuk. Ezek az Adatbázis, a Backend (szerveroldal), és a Frontend (kliensoldal). A felhasználók ezek körül csupán a frontendet, azaz a kliensoldali réteget látják, és érik el közvetlenül.



3. ábra Alkalmazás felépítése

A rétegek feladatai:

1.4.1.1 Adatbázis réteg

Az adatbázis feladata, hogy az alkalmazás által használt adatok strukturált, biztonságos tárolását biztosítsa. A biztonság garantálása érdekében a felhasználó közvetlenül semmilyen módon nem férhetnek hozzá ehhez a réteghez. A kapcsolatot egy köztes réteg, a backend biztosítja. Természetesen az adatbázis a szerveroldali rétegek közé tartozik.

1.4.1.2 Backend réteg

A Backend (azaz szerveroldali réteg) fő feladata, hogy a kliens, és az adatbázis közötti biztonságos összeköttetést garantálja. Segítségével ellenőrizni tudjuk, hogy az adatbázisunkba csak megfelelő adatok kerülhessenek, valamint hogy a felhasználók csak a jogosultságaiknak megfelelő tartalmakat érhessék el.

1.4.1.3 Frontend réteg

A frontend, azaz (kliensoldali) felhasználói réteg feladata, hogy a felhasználók számára is értelmezhető, átlátható, és kezelhető módon megjelenítse a szükséges adatokat, valamint lehetőséget biztosítson, hogy a felhasználó kényelmesen kéréseket küldjön a szerver felé (Pl.: adat beszúrás), valamint a szervertől kapott választ megjelenítse a felhasználónak.

1.4.2 Frontend

Az alkalmazás megjelenítéséért a kliens oldal felel. A tartalomért HTML, a megjelenésért CSS, azon belül is Bootrstrap keretrendszer, míg a kliens oldali funkciók működéséért JavaScript, valamint a legismertebb JS könyvtár, a JQuery felel. Az ASP.NET keretrendszer lehetőséget biztosít számunkra az MVC tervezési minta használatára, melyet jellemzően Razor Web-es technológiával szokás társítani. Az alkalmazás felépítésének köszönhetően lehetőségünk van egy projecten belül fejleszteni a Kliens- és Szerver oldalt is úgy, hogy mégis elkülönítjük a rétegeket, azaz biztosítjuk a párhuzamos fejlesztési lehetőséget, és az átláthatóságot. Szimplán néhány kattintással létrehozhatunk egy új „Nézetet” (oldalt) az alkalmazásunkhoz. Fontos kiemelni, hogy a nézeteinkbe csupán az adott oldal „body” részének tartalmát kell megírnunk, ugyanis az alkalmazás minden lapja egy úgynevezet Layout-ra (sablonba) kerül bele, mely tartalmazza az alkalmazás vázát (Pl.: Navbar, Boorstrap, és JQuery könyvtárak importálása). Amennyiben egy nézethez kliensoldali funkciókat (scripteket) szeretnénk hozzáadni, abban az esetben a kódot érdemes a nézeten belül „Scripts” section-be rakni, melyet az alkalmazásunk a sablonban, a body végén fog „behúzni”. Hasonló a helyzet a formázás esetében is, itt „CSS” section-be írjuk a kódjainkat, melyet az alkalmazásunk a head rész megfelelő helyére fogja behúzni.

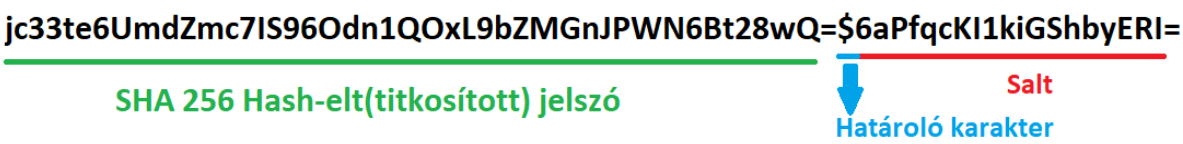
1.4.3 Backend

Az alkalmazás szerveroldali rétegét egy C# nyelven. ASP.NET keretrendszer alatt készült alkalmazás szolgálja ki. A szerveroldal legfőbb komponensei a controllerek, avagy vezérlők. Minden adatbázis művelet keresztül megy a vezérlőnkön, ugyanis a kliensek a controller felé intéznek kéréseket, melyre a szerverünk választ ad. Annak érdekében, hogy ne sértsük meg a

szoftverfejlesztési elveket (Pl.: SOLID) érdemes a lehető legjobban elkülöníteni a komponenseket, épp ezért az alkalmazásban minden fő tevékenységi körre külön controller van (Pl.: Auth, Admin, User, Api). Az adatbázissal történő kommunikáció Entity Frameworkon keresztül történik, melynek segítségével nem kell folyamatosan adatbázis lekéréseket írnunk, hanem elég egy táblastruktúrát leíró modellt készítenünk, majd DBSet-eken keresztül már el is érhetjük az adatbázisunkat.

1.4.4 Adatbázis, tárolt adatok, adatvédelem

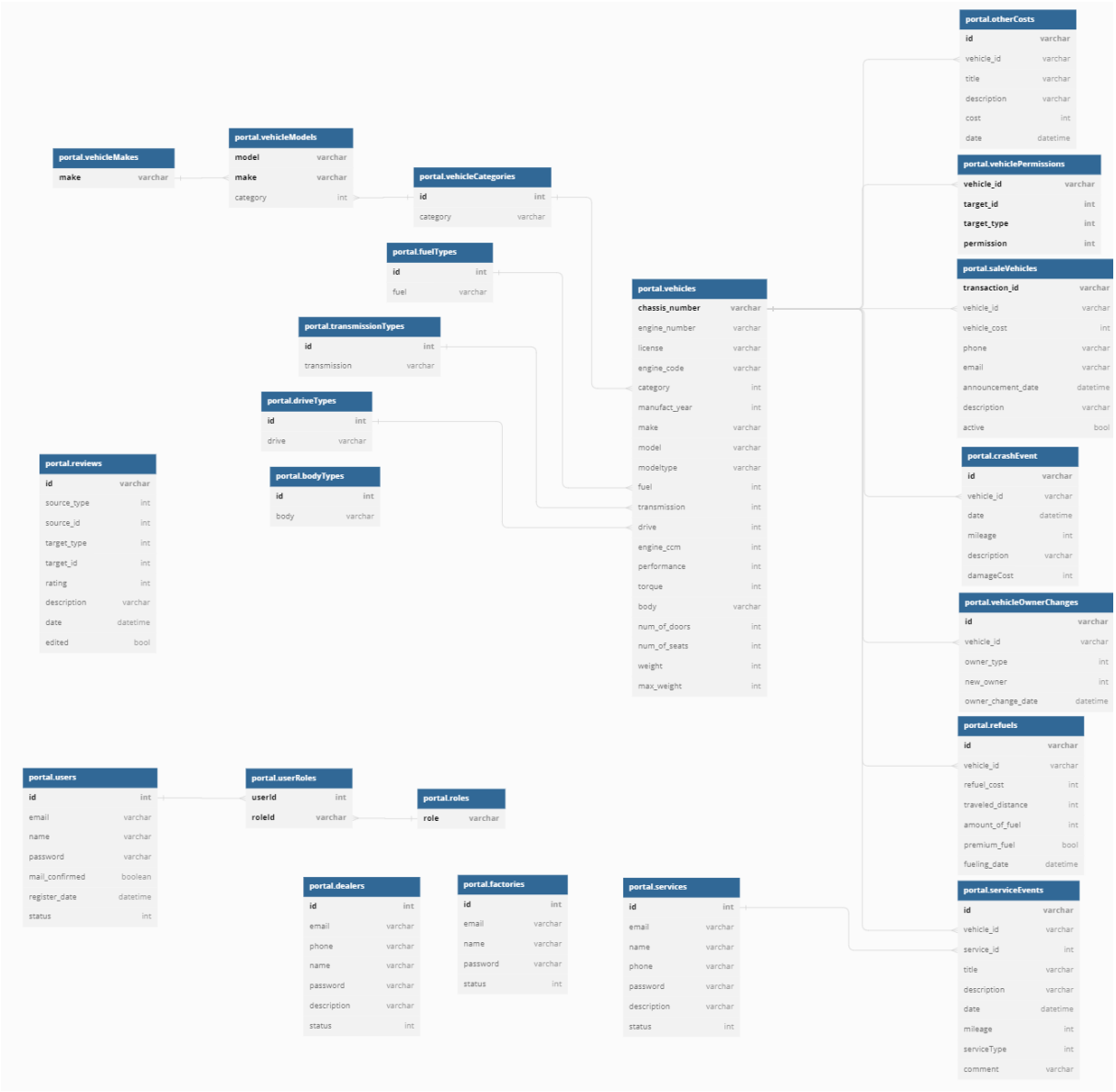
Az adatbázis struktúra kialakításakor fontos szempont volt az optimális felépítés, valamint a megfelelő tábla-kapcsolatok kialakítása is annak érdekében, hogy elkerüljük az adatintegritási problémákat, bár az alkalmazás kliens, és szerver oldali adatellenőrzést is végez az adatmanipulációs műveletek előtt. Mivel az alkalmazás rendeltetésszerű használata során hatalmas mennyiségű adat kerülhet eltárolásra, ezáltal fontos volt, hogy azok eltárolása optimális, átgondolt legyen. Egy hibásan megtervezett, vagy elkészített adatbázis esetén nem csak a végtelenségekig lassíthatjuk az alkalmazásunkat, de akár egy idő után az egész appot használhatatlanná teheti. Az alkalmazás minden felhasználó típusának a jelszavát saltozva, SHA-256 titkosítással tárolja. Az SHA-256 titkosítás során a bevitt jelszóból egy 256 bit hosszúságú karakter sorozat keletkezik. A hashelési algoritmusnak köszönhetően így egy biztonságos jelszó feltörése bruteforce megoldással napjainkban még egy szuperszámítógép számára se valósítható meg belátható időn belül, természetesen visszafejtésre (napjainkban) szintén nincs lehetőség. A saltozás (sózás) célja, hogy esetleges két azonos karakter sorozat esetén is eltérő legyen a hash-elt érték, melyet úgy érünk el, hogy a hashelés előtt egy véletlenszerűen generált (esetünkben 10-15 közötti véletlen hosszúságú) karaktersorozatot (ez lesz az ún. salt) fűzünk a titkosítani kívánt karaktersorozathoz. Természetesen a saltot szintén el kell tárolnunk, hogy később (esetünkben bejelentkezéskor) a bevitt jelszóhoz újra hozzáfűzhessük, és ezáltal a generált hash megegyezzen az tárolttal. A saltot esetünkben a jelszó mögött tároljuk el ’$’ határolókarakterrel elválasztva. Az alábbi képen egy minta látható az eltárolt jelszóról:



4. ábra Jelszó tárolás

1.4.4.1 *Adatbázis terv*

Az alkalmazás adatait tároló MySql adatbázis pillanatnyi felépítése az alábbi ábrán látható:



5. ábra Adatbázis terv

1.5 Beüzemelési javaslat

Az általam készített alkalmazás egy webes alapú rendszer, ezáltal többféleképpen üzemeltethetjük (Pl.: helyi számítógépen, szerver gépen, felhőben), melyeknél a beüzemelés módja eltérő lehet. Én a windows operációsrendszer alatti beüzemelésre fogok kitérni. A beüzemeléshez szükség lehet valamennyi rendszergazdai ismeretre, tapasztalatra.

1.5.1 MySql szerver telepítése, beüzemelése

Töltsük le a MySql Server lehetőleg legújabb (jelenleg 8.0) támogatott verzióját az oracle hivatalos oldaláról, majd haladjunk végig a telepítőn. A telepítés során szükséges lesz, hogy megadjuk majd az úgynevezett ’root’ (admin) felhasználónk jelszavát, mindenképpen jegyezzük meg az itt megadott jelszót, ugyanis ha elfelejtjük, akkor később ezt meg kell változtatnunk MYSQL SHELL-ben, mely szintén némi MYSQL-kezelési ismeretet kíván.

Jelentkezzünk be az adatbázis szerverünkre, az általunk használt adatbázis-kezelő alkalmazással (Pl.: MySql Workbench), a felhasználó nevünk ’root’, míg a jelszó az általunk megadott érték lesz. Hozzunk létre egy új MySql felhasználót az alkalmazásunk számára.

Ezt a „CREATE USER `felhasznalo`@`host` IDENTIFIED BY `password`;” paranccsal tehetjük meg. Természetesen a ’felhasznalo’ helyére az általunk kívánt felhasználó nevet írjuk (célszerű olyant választani, mely utal a felhasználására, Pl.: portal), a ’host’ helyére az alkalmazást futtató kiszolgáló IP címét (Pl.: localhost), mellyel lekorlátozzuk, hogy csak az adott címről lehessen belépni a felhasználóba, míg a ’password’ helyére az újonnan létrehozott felhasználónk jelszava fog kerülni. Jelszó választáskor célszerű megfelelő hosszúságú, kis-és nagybetűket, számokat, valamint speciális karaktereket egyaránt tartalmazó (akár véletlenszerűen generált) jelszót választani.

Az adatbázisunk létrehozását a mellékelt adatbázis forrásfájl tartalmazza, ezáltal elegendő, ha a fájl tartalmát egy az egyben lefuttatjuk. Ekkor létrejön az adatbázisunk ’portal’ néven.

Most adjunk jogosultságot a létrehozott felhasználónknak az adatbázisunkhoz való hozzáférésre. A jogosultság megadásakor célszerű, hogy a felhasználónk az adatbázisunkhoz korlátlanul hozzáférjen (ellenkező esetben az alkalmazás nem fog megfelelően működni). A jogosultság megadásához futtassuk le a „GRANT ALL PRIVILEGES ON portal.* TO 'username'@'host';” természetesen a ’username’, és ’host’ értékeket itt is cseréljük ki a létrehozott felhasználónk adataira. A parancs lefuttatásának a hatására a felhasználónk teljes hozzáférést (adatok beszúrása, törlése, módosítása; tábla létrehozása, törlése, módosítása stb.) kap a portal adatbázisunk minden táblájához.

Ezzel az adatbázisunk beüzemelésével végeztünk is.

1.5.2 Program összekötése az adatbázissal

Következő lépésben kössük össze az alkalmazásunkat az adatbázissal. Mivel az alkalmazás rendelkezik saját „config” fájllal, ezért különösen egyszerű dolgunk van. Keressük meg az alkalmazásunk mappáját, majd nyissuk meg az „appsettings.json” fájlt. A fájl szerkesztésére akár még hagyományos jegyzettömböt is használgatunk, mindazonáltal célszerű valamilyen olyan szerkesztőt használni, amely átláthatóan tagolja a szöveges állományt, és még akár systax highligting is rendelkezésre áll. Ilyen alkalmazások lehetnek például a notepad++, vagy a visual studio code, de akár a használt fejlesztői környezetben, a visual studio 2022-ben is módosíthatjuk.

A konfigurációs állományban keressük ki a „ConnectionString”, azon belül pedig az „SQL” részt, majd állítsuk be az adatbázisunk elérhetőségét. Mivel a konfigurációs fájl egy JSON állomány, ezért fontos figyelni annak szintaktikájára is.

```
"ConnectionStrings": {
  "SQL": "Server=127.0.0.1;Database=portal;Uid=vallgyak;Pwd=vallgyak;"
},
```

6. ábra Alkalmazás-Adatbázis összekötés

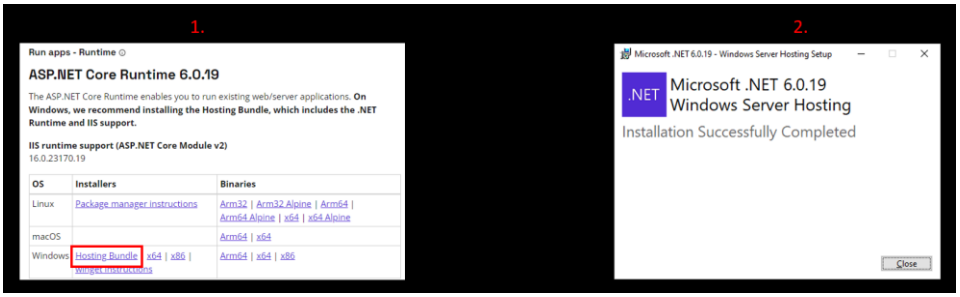
A kapcsolat kialakításának sikerességét valamint a tábláink meglétét úgy ellenőrizhetjük, hogy az alkalmazás elindítását követően az „IP/Dev/CheckDbConnection” elérési útra navigálunk. Ekkor JSON formátumban látni fogjuk az adatbázisunk státuszát (Csatlakozási próbálkozás) a CanConnect alatt, majd az összes táblánkat felsorolva állapot jelzéssel. Természetesen a true jelenti a sikert, ellenben ha false értékel látunk, akkor valamilyen hiba van, ebben az esetben ellenőrizzük az adatbázisunkat, és a kapcsolatot.


```
CanConnect => True



tables => {
  "BodyType": true,
  "DriveType": true,
  "FuelType": true,
  "Role": true,
  "TransmissionType": true,
  "User": true,
  "UserRole": true,
  "Vehicle": true,
  "VehicleCategory": true,
  "VehicleMake": true,
  "VehicleModel": true,
  "VehiclePermission": true,
  "OtherCost": false,
  "CrashEvent": false,
  "VehicleOwnerChange": false,
  "Refuel": false,
  "ServiceEvent": false,
  "Service": false,
  "Factory": true,
  "Dealer": false,
  "Review": false
}
```


7. ábra Adatbázis kapcsolat ellenőrzése

1.5.3 ASP.NET Core Runtime letöltése, telepítése

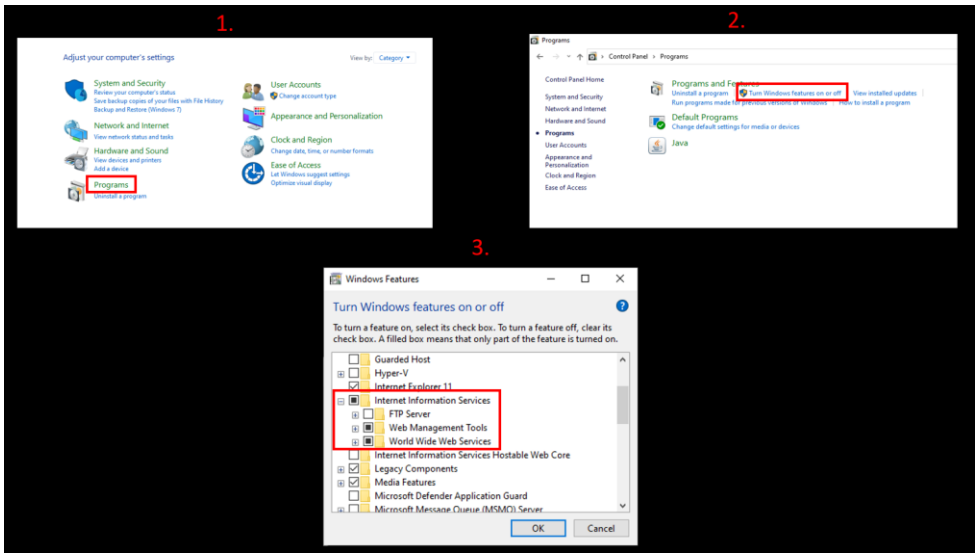
Az alkalmazás futtatásához szükségünk van az ASP.NET Core Runtime (lehetőség szerint) legújabb verziójára, melyet a dotnet hivatalos oldaláról tölthetünk le (https://dotnet.microsoft.com/en-us/download/dotnet/6.0). Javasolt a Hosing Bundle verzió letöltése windows alatt, ugyanis ez minden szükséges dolgot feltelepít. Amennyiben Linux, vagy macOS rendszeren szeretnénk futtatni az alkalmazást, akkor sem kell aggódnunk, ugyanis azokra is elérhetőek ugyan itt a szükséges telepítők. Sikeres telepítés után érdemes újraindítani a gépünket annak érdekében, hogy minden módosítást biztosan elvégezzen, és életbe léptessen a rendszerünk.



8. ábra .NET keretrendszer letöltése, telepítése

1.5.4 IIS Express webszerver beüzemelése

Az IIS Express webszerver alapból elérhető minden Windows10 operációs rendszert futtató számítógépen, mindazonáltal használatához engedélyeznünk kell ezt a szolgáltatást. Ennek legegyszerűbb módja, hogy belépünk a vezérlőpultba, kiválasztjuk a Programok menüpontot, azon belül pedig a Programok és szolgáltatások alatti Windows-szolgáltatások be- és kikapcsolása opció lehetőséget válasszuk. Fontos megjegyezni, hogy rendszergazdai jogosultságra van szükségünk ahhoz, hogy a szolgáltatást engedélyezzük. Ekkor előjön egy ablak. Itt keressük meg az Internet Information Services, illetve az Internet Information Services üzemeltető webmagja lehetőségeket, és a mellettük található négyzet bepipálásával engedélyezzük a szolgáltatásokat. Amennyiben még korábban nem használtuk ezeket, abban az esetben a rendszerünk automatikusan frissítéseket keres hozzá, és letölt minden hozzá szükséges eszközt, amely kis időbe telhet.



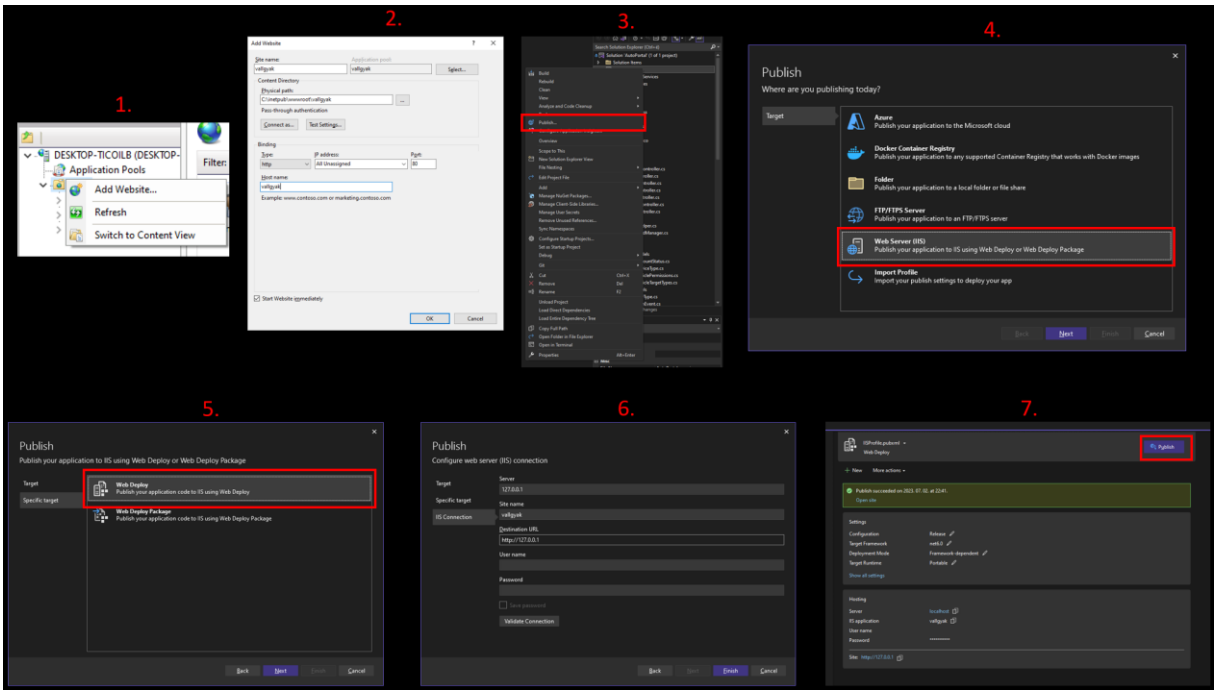
9. ábra IIS Express Webszerver beüzemelése

1.5.5 Webhely létrehozása, project exportálása (Deploy)

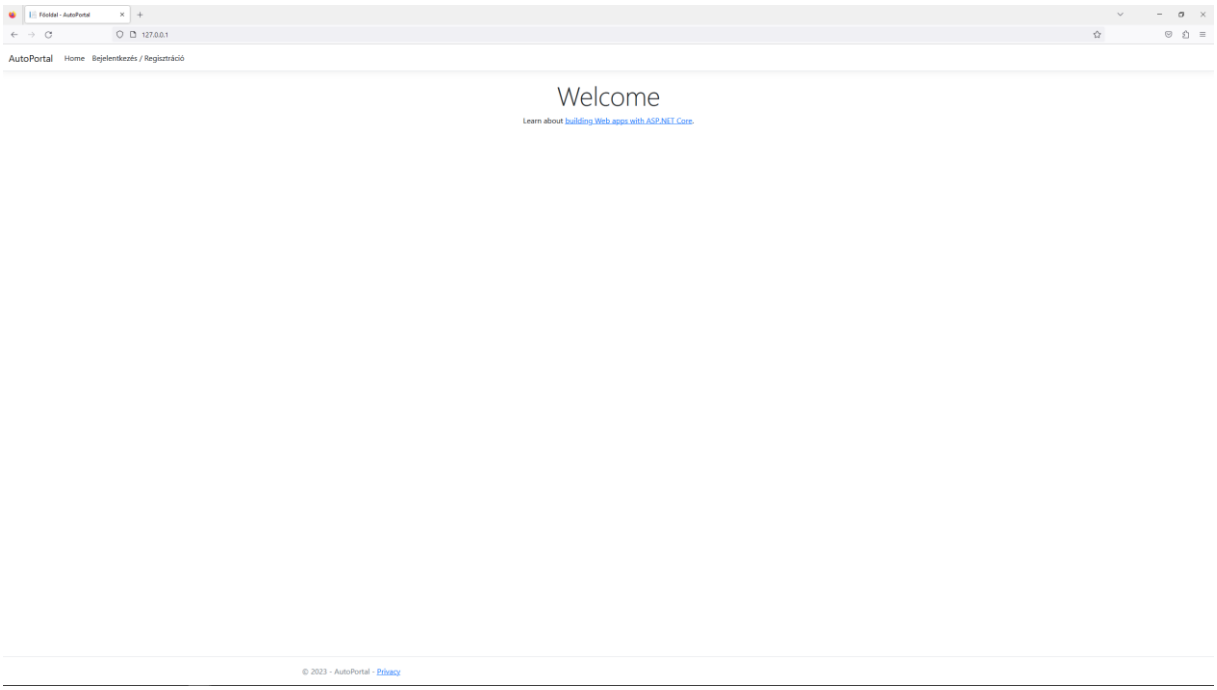
Miután aktiváltuk a webszerverünket, indítsuk újra a számítógépünket. Ezt követően elérhetővé válik számunkra egy új alkalmazás, mely „Internet Information Services (IIS) kezelője” névre hallgat. Nyissuk is meg ezt az alkalmazást, amely elérhetővé teszi a szerverünk kezelőfelületét. Baloldalon megjelenik a számítógépünk neve, ezt nyissuk le, hogy megjelenjen a helyek címszóra hallgató opció. Erre kattintsunk jobb egérgombbal, és válasszuk a webhely hozzáadása lehetőséget. A hely neve alatt tetszőleges nevet adhatunk meg, viszont javasolt olyan címet választani, hogy tudjuk, miről van szó. A fizikai elérési úthoz azt a helyet kell kiválasztanunk, ahová az oldalunkat „Telepítjük”. Továbbá itt állíthatjuk be, hogy http, vagy

https protokollt szeretnénk használni, valamint azt, hogy mely porton szeretnénk futtatni az oldalunkat. Éles környezetben javasolt a https protokoll használata, ugyanis az jóval biztonságosabb, továbbá mondhatni kötelező az SSL tanúsítvány használata, melyet alapból kapunk a webszerver mellé, így ezt csak ki kell választanunk. Állomásnévhez az oldalunk domain nevét állítsuk be, majd miután mindennel végeztünk kattintsunk az OK gombra. Miután ezekkel végeztünk kattintsunk a baloldali listában a számítógépünket jelző elemre, és a jobboldali menüben kattintsunk az indításra. Miután elindult a webszerverünk a helyek között válasszuk ki a korábban létrehozott webhelyünket, és szintén a jobboldali menüben indítsuk el azt is.

Következő lépésként indítsuk el rendszergazdaként a Visual Studio 2022-t, és nyissuk meg a projektünket. A jobb oldali solution explorer-ben kattintsunk jobb egérgombbal a projectünkre, majd válasszuk a publish opciót. Ekkor megjelenik számunkra egy új ablak, ahol megjelennek a lehetséges exportálási opciók. Esetünkben válasszuk ki a Web Server (IIS) opciót, majd kattintsunk a Next (tovább) gombra. A következő lépésben válasszuk ki a sima Web Deploy opciót, és szintén kattintsunk a tovább gombra. A most megjelenő ablakban kell megadnunk a szerverünk adatait. A Server mezőbe adjuk meg a szerverünk címét (esetünkben localhost), az oldalunk nevét (ennek meg kell egyeznie a webszerveren létrehozott oldal névvel) a Site Name mezőbe, ez esetünkben a vallgyak. A Destination URL, azaz cél elérésiút mezőbe pedig adjuk meg az oldalunk majdani elérési útját (protokoll://IPCÍM). Amennyiben nem helyi, hanem külső szerverre történik az exportálás szükséges felhasználónév, és jelszó megadása is. Miután minden adatot helyesen megadtunk, a Validate Connection gomb megnyomásával ellenőrizhetjük a csatlakozást, majd kattintsunk a Finish gombra. A Publish gombra kattintva sikeres fordítás után fel is kerül az oldalunk a szerverre, melyet az alapértelmezett böngészőben meg is nyit nekünk a Visual Studio.



10. ábra Alkalmazás beüzemelése



11. ábra Beüzemelt alkalmazás

1.5.6 Microsoft Azure felhőszolgáltatás, és lehtőségei

Az alkalmazás üzemeltetésénél mindenképpen kiemelném a Microsoft által nyújtott Azure felhőszolgáltatást, mely rengeteg lehetőséget biztosít számunkra. Segítségével egy platform alatt elérünk minden olyan szolgáltatást, mely az alkalmazásunk üzemeltetéséhez szükséges. Ilyenek például a web szerver, az adatbázis szerver, vagy éppen a domain név vásárlás. Azonban egy web alkalmazás élet környezetben történő üzemeltetése nem merül ki ebben a néhány lépésben, ugyanis nem szabad megfeledkeznünk például a rendelkezésreállásról, és terhelésmegosztásról sem, ugyanis például application gateway szolgáltatást is igénybe tudunk venni. Ezen szolgáltatás segítségével az alkalmazásunk nem csupán egy, hanem több szerverről szolgálja ki a felhasználókat, így csökkentve a magas terhelés esetén a válaszidőt, valamint csökkentjük annak esélyét, hogy az alkalmazásunk teljes mértékben elérhetetlenné váljon (egy szerver kiesése esetén a többi szerverünk a meghibásodottól függetlenül ki tudja szolgálni a kéréseket, és át tudják vállalni annak szerepét). Természetesen az adatbázisunk kiesése esetén az alkalmazásunk szintén használhatatlanná válik, ezért érdemes arról is folyamatosan mentést készíteni, vagy még biztonságosabb megoldásként két- vagy több párhuzamosan működő adatbázist üzemeltetni, melyeken a tárolt adatok megegyeznek (mindegyiket folyamatosan frissítjük), viszont egyik kiesése esetén az alkalmazásunk azonnal, kiesés nélkül át tud állni egy másikra. Mindazonáltal a Microsoft által nyújtott felhőszolgáltatásoknál a garantált magasszintű rendelkezésreállásnak köszönhetően (akár közel 100%-os) nem kell túlgondolnunk a meghibásodási, és rendelkezésreállási incidensek lehetőségét.

Egy népszerű webes alkalmazást szinte folyamatos támadások érnek (Például DOS, DDOS, POD), melyeket a Microsoft tűzfalai szintén mondhatni 100%-os hatékonysággal tudnak kivédeni megfelelő beállításokkal, ezért ezektől szintén nem kell tartaunk. A folyamatos frissítéseknek köszönhetően mindig naprakész szerver hátteret kapunk.

Mint minden szolgáltatásnál, a minőséget sajnos itt is meg kell fizetni, épp ezért az előnyök után most hátrányként kiemelném az Azure által nyújtott szolgáltatások talán legnagyobb hátrányát, a magas árazást. Bár folyamatosan (heti-havi) rendszerességgel fejlődik, és változik a szolgáltatás, jelenleg nagyon kevés szolgáltatás érhető el ingyen, azok is csak korlátozottan.