



Vas Megyei Szakképzési Centrum
Nádasy Tamás Technikum és Kollégium

SZAKDOLGOZAT

Éttermi szoftver

Bécs Tamara

**Konzulens:
Balics Gábor**

2022

Nyilatkozat

Alulírott, Bécs Tamara kijelentem, hogy az Éttermi szoftver című szakdolgozat feladat kidolgozása a saját munkám, abban csak a megjelölt forrásokat, és a megjelölt mértékben használtam fel, az idézés szabályainak megfelelően, a hivatkozások pontos megjelölésével.

Eredményeim saját munkán, számításokon, kutatáson, valós méréseken alapulnak, és a legjobb tudásom szerint hitelesek.

Csepreg, 2022.

Bécs Tamara

Kivonat

Éttermi szoftver

Az éttermi szoftver számos vendéglátó egység számára nyújthat segítséget, abban, hogy könnyebben kezelni tudják a rendeléseket. A program célja az, hogy megkönnyítse a vendéglátó egységben dolgozók mindennapjait. Hiszen számos olyan funkcióval rendelkezik, ami fontos lehet egy rendelés leadása során.

De mit is értek ezek alatt?

Az alkalmazásban van lehetőség arra, hogy a felhasználó új ételeket vegyen fel ezzel is gyarapítva a választási lehetőségeket. Mint tudjuk számos ember küzd különböző ételallergiákkal vagy érzékenységekkel például tojás esetleg glutén vagy laktóz, de emellett még tömördek más allergén van, amit órákon keresztül sorolhatnánk. Tehát ezekből a nehezítő tényezőkből kiindulva van lehetőség arra is, hogy az adott ételknél ezeket feltüntessük, illetve a módosításukra és törlésükre van lehetőség.

Mindemellett arra is van mód, hogy kategóriákba rendezzük az ételeket. Ezeket az ételtípusokat tartalmazó listát is bármikor lehet módosítani, ha a felhasználó igényt tart rá.

Ezenfelül mint ahogy a fentiekből következik komplett ételeket is feltudunk venni a hozzájuk tartozó allergénekkal, illetve megjegyzésekkel és még más ehhez tartozó dolgokkal. Például van arra is lehetőség, hogy megadjuk, hogy adott étel jelenleg elérhető-e vagy sem.

Tehát mivel adottak a feltételek a felhasználó a meglévő adatok segítségével könnyen és kényelmesen tud rendeléseket is felvenni. Ezáltal egy jól átlátható programot fogunk kapni.

Abstract

Restaurant software

Restaurant software can help many catering units to make it easier to handle orders. The goal of the program is to help the everyday lives of those working in the hospitality unit. After all, it has a number of features that can be important when placing an order.

But what do I mean by that?

The app allows the user to add new dishes, increasing their choices. As we know, many people have different food allergies or intolerance, such as eggs may have gluten or lactose, but there are still quite a few other allergens that could be listed for hours. Based on these aggravating factors, it is also possible to indicate them in the given dishes, and it is possible to modify and delete them.

However, there is also a way to categorize the dishes. The list of these food types can be modified at any time if the user requires it.

In addition, as it can be seen from the above, we can include complete food types with their associated allergens and comments, and even related things. For example, it is also possible to specify whether a particular food is currently available or not.

Therefore, since the conditions are given the user can add specific data to orders. This will give us a transparent program.

Tartalomjegyzék

1.	Bevezetés.....	7
2.	Fejlesztői környezet bemutatása.....	8
2.1.	.NET keretrendszer	8
2.1.1.	A .NET keretrendszer bemutatása	8
2.1.2.	A .NET egy alkalmazásfejlesztési ökoszisztéma.....	8
2.1.3.	.NET nyelvek és -fordítók használata.....	9
2.2.	C# programozási nyelv története	10
2.2.1.	A C# felhasználási területei	10
2.2.2.	A C# kialakulása.....	10
2.3.	ASP.NET MVC – Model View Controller.....	12
2.4.	MSSQL -Adatbázis kezelő rendszer	14
2.5.	Code-First	14
2.5.1.	Mi az a Code First?.....	14
2.5.2.	Code-First munkafolyamat	15
2.6.	Mi az a migráció?	16
3.	Rendszer ismertető	17
3.1.	Az alkalmazás követelményei	17
3.2.	Az adatbázis	18
3.2.1.	Mit nevezünk adatbázisnak?.....	18
3.2.2.	Adatbázis a programban	19
3.3.	Koncepcionális (tervezési) szint	20
3.4.	Fizikai (implementációs) szint.....	22
4.	A rendszer megvalósítása	23
4.1.	Telepítés	23

4.1.1.	A rendszer követelményei	23
4.2.	Megjelenés	23
4.3.	Bejelentkező felület	24
4.4.	Menüpontok	25
4.4.1.	Főoldal	25
4.4.2.	Emberek	26
4.4.3.	Asztalok	27
4.4.4.	Ételek	28
4.4.5.	Rendelések	29
5.	Tesztelés	31
6.	Továbbfejlesztési lehetőségek	36
7.	Összegzés	37
8.	Ábrajegyzék	41
9.	Irodalomjegyzék	42

1. Bevezetés

Az éttermi szoftver elsősorban azért jött létre, hogy megkönnyítse a vendéglátásban dolgozó pincérek munkáját. A program segítségével, sokkal hatékonyabban és gyorsabban lehet kezelni a rendeléseket, hiszen a felhasználónak csak pár gombot kell megnyomnia és el is készül a vendég által leadott kérés. Ennek köszönhetően a rendeléssel töltött idő sokkal rövidebb lesz. Továbbá a program használatának köszönhetően elkerülhetők a félreértések egy - egy rondábban írt vagy olvashatatlan étel esetleg ital neve kapcsán. Mindezen felül kényelmesebb és kevésbé olyan megterhelő lesz a rendelésvétel a pincérek számára. A lerövidült folyamatnak köszönhetően, akár több rendelést is fel tudnak majd venni ugyanannyi idő alatt. Ez azt eredményezi, hogy a vendégek sokkal elégedettebbek lesznek, hiszen nem kell olyan sokat várni az elfogyasztani kívánt ételre, italra.

Azért választottam ezt a témát, mivel egy nyári munka keretein belül én is dolgoztam étteremben. Így volt szerencsém megismerni ennek a folyamatnak a részeit. Viszont az az étterem egy újonnan induló vendéglő volt. Ez sajnos számos dologban megnyilvánult, például abban is, ahogy a pincérek kezelték a rendeléseket. A szoftver hiánya miatt, minden rendelés háromszor, jobb esetben kétszer ment át a kezük alatt. Hiszen nem csak maguknak, de a konyhán dolgozó személyek számára is ismertté kellett tenni ezeket, ennek okán mindent többször leírtak. Ezáltal nagyon sok papír fogyott és rengeteg megrendelést tartalmazó papírt kellett megőrizni.

Ennek a folyamatnak a leegyszerűsítése miatt szerettem volna kialakítani egy olyan rendszert, amely könnyen átlátható, így nem igényel különösebb informatikai tudást vagy akár továbbképzést.

2. Fejlesztői környezet bemutatása

2.1. .NET keretrendszer



1. ábra: Microsoft .NET Framework logó

2.1.1. A .NET keretrendszer bemutatása

A .NET keretrendszert azaz a .NET Frameworkot a Microsoft fejlesztette ki, és rengeteg osztálykönyvtárt (előre megírt kódot) tartalmaz, aminek a segítségével a fejlesztők gyorsabban és hatékonyabban hozhatnak létre új alkalmazásokat.

A C#-ot és a .NET-et gyakran együtt emlegetik, mivel mindkettő a Microsoft terméke. De vannak más keretrendszerek is, amiket C#-pal használnak – ilyen például a Unity.

2.1.2. A .NET egy alkalmazásfejlesztési ökoszisztéma

Az ökoszisztéma kifejezés utal egy környezetre és a körülötte kialakult közösség kialakulására. Sokfélesége az egyik legmeghatározóbb dolog, ami az mellett szól, hogy miért érdemes megtanulni, alkalmazni és alkalmazásokat fejleszteni a .NET-ben.

2.1.3. .NET nyelvek és -fordítók használata

A szoftverfejlesztők .NET nyelveken általában C# esetleg F# nyelven írják meg a forráskódokat. Minden kódsor egy utasítást vagy parancsot fejez ki a számítógépnek, ennek köszönhetően tudja futtatni a számítógép az adott programokat.

Ahhoz, hogy a programozók futtatni tudják a kódokat először le kell fordítani őket. Itt jön képbe a .NET. A keretrendszer egy fordítóprogram, mely egy köztes nyelvre (IL) konvertálja a forráskódot. A .NET -fordító az IL -kódot egy .NET szerelvénybe menti. Ennek a folyamatnak köszönhetően ugyanaz a kód bárhol futtatható, Linux vagy Windows rendszeren, és 32 bites, illetve 64 bites számítógép-hardveren egyaránt. A .NET-futtatókörnyezet a lefordított .NET-szerelvény végrehajtási környezete. Ez azt jelenti, hogy a gazda operációs rendszeren futó alkalmazást a .NET-futtatókörnyezet hajtja végre és kezeli.

2.2. C# programozási nyelv története

2.2.1. A C# felhasználási területei

A C# (ejtsd: „szí sárp”) egy objektumorientált programnyelv, amit elsősorban asztali, mobil- és webes alapú alkalmazások fejlesztésére használnak Windowsra és más Microsoft által kibocsátott termékekre. A C#-pal szinte bármit fejleszthetünk Microsoftra a .NET keretrendszer használatával. A .NET keretrendszer rengeteg programnyelvet támogat, például a VB.NET-et, a C++-t és az F#-ot, de a C# a legnépszerűbb közülük.

2.2.2.A C# kialakulása



2. ábra: C# logó

A C# programozási nyelv béta verziójához 2000 júniusában jutott hozzá az informatikus közösség, azonban a hivatalos változat 2002 tavaszán látott napvilágot. Ez is azt bizonyítja, hogy egy elég fiatal nyelvről beszélünk.

A fent említett nyelvet elsősorban a Microsoft egy dolgozója Andres Hejlsberg fejlesztette ki a Sun Microsystems által készített Java konkurenciájaként.

Andres Hejlsbergéről azt kell tudni, hogy a Dán Mérnöki Egyetemen folytatta tanulmányait, azonban nem diplomázott le. Egyetemi éve alatt számos programot írt beleértve egy Pascal fordítót is a későbbi években számos nyelven volt ismert az újraírt változata. Mint például a Compas Pascal és a PolyPascal. Később a termék Borland licencet kapott és beépítették egy integrált fejlesztői környezetbe így megszületett a Turbo Pascal.

Anders és partnerei egy dán számítógépes boltot üzemeltettek. A PolyData nevű cég volt a Microsoft termékeknek a dán forgalmazója, ami szembeállította őket a Borlanddal. Hejlsberg és Philippe Kahn, a Borland alapítója, először 1986-ban találkozott. Így kezdődött Anders kapcsolata a Borlanddal, ahol a későbbiekben a cég vezető mérnöke lett, majd annak a csoportnak lett a vezetője, akik a Delphi programozási nyelvet fejlesztették.

Hejlsberg 1996-ban távozott a fent említett cégtől és csatlakozott a Microsofthoz,

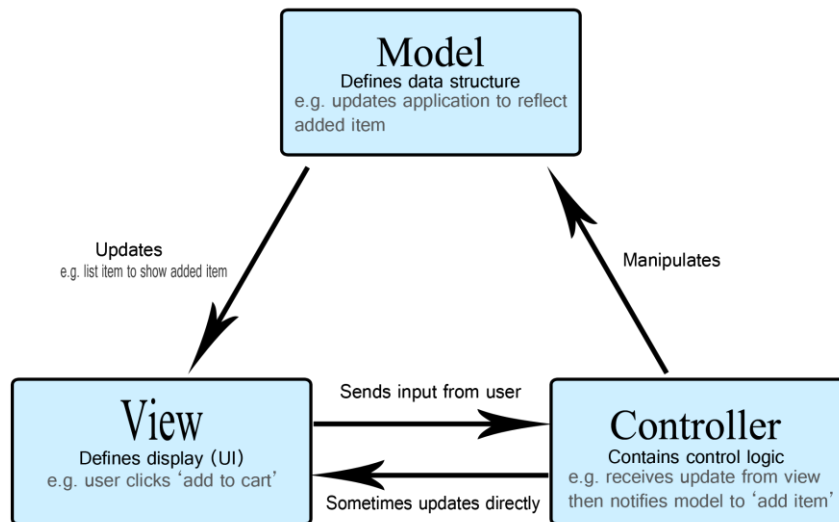
ahol megalkotta a Visual J++ -t és a Windows Foundation Classes-t. 2000-óta pedig a C#-ot fejlesztő csapat vezető mérnöke.

A fejlesztő azt állítja, hogy a C# közelebb áll a C++-hoz, azonban számos közös tulajdonsága van a Java-val is. A fejlesztések során megpróbálták a létező programozási nyelvek jó tulajdonságait, illetve további javításokkal és fejlesztésekkel ruházták fel a nyelvet.

A modern Java és a C++ nyomdokaiba lépve a C# is egy általános célú objektumorientált programozási nyelv, ami az egyszerűsége mellett a kiemelkedő hatékonyságot célozza meg.

Igen hatékony és rugalmas programozási nyelv, mellyel alkalmazások széles körét készíthetjük el. Maga a nyelv nem korlátozza a programozót a tevékenységében, így fantáziánk határtalanul szárnyalhat. Mindezt mi sem bizonyítja jobban, mint az, hogy a C#-ot használták már dinamikus webhelyek, fejlesztőeszközök, sőt még fordítóprogramok készítéséhez is.

2.3. ASP.NET MVC – Model View Controller



3. ábra: MVC

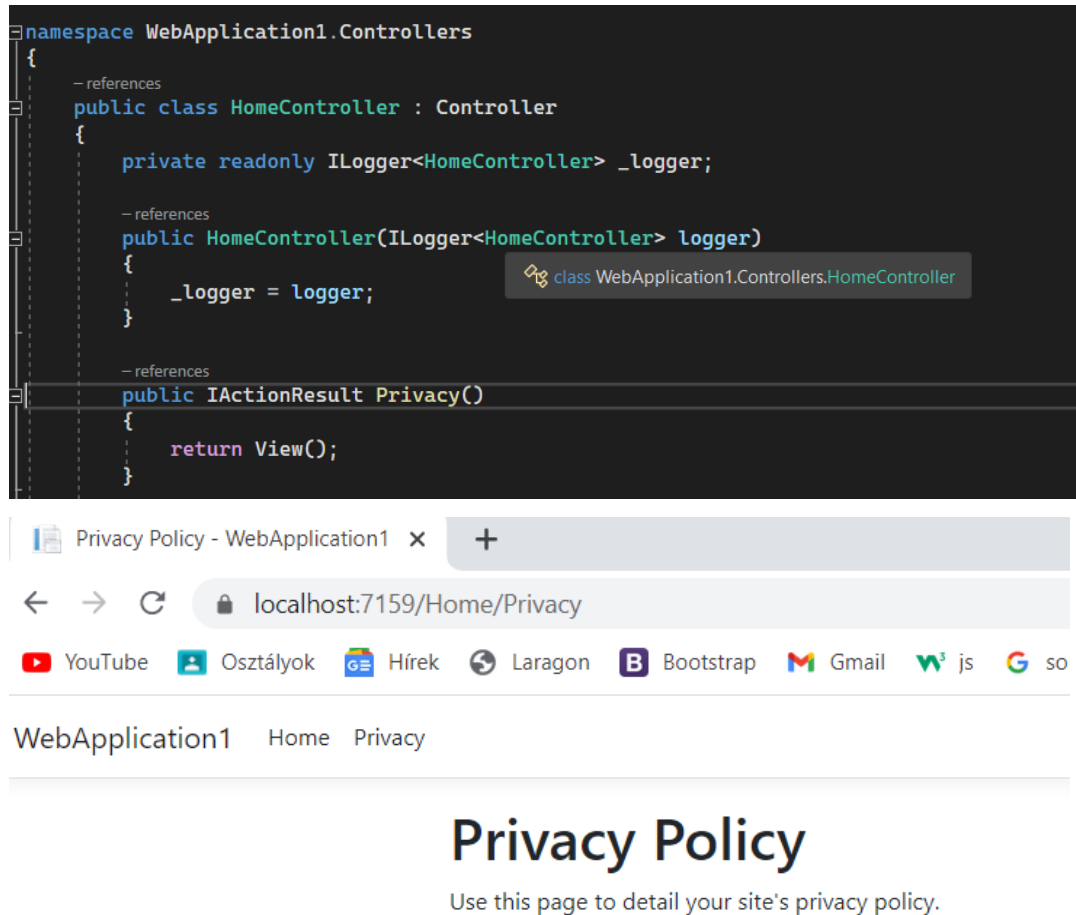
Az MVC a Model View Controller rövidítése, amely a felhasználói felületek megvalósításának architektúráis mintája. Eredetileg az 1970-es években fejlesztették ki az asztali alkalmazásokhoz, azonban ez az évek során széles körben elterjedt. Pont úgy, mint az MVC-hez hasonló keretrendszerek mint például a Ruby on Rails vagy az Express.

A keretrendszerről egy kicsit részletesebben; a „Model” az adatok, objektumok reprezentálásáért felelős, az alkalmazás állapotát és szabályait testesíti meg. A modellekbe olyan osztályok tartoznak, melyek függetlenek a felhasználói felülettől, emiatt könnyen használhatóak és „átvihetőek”. Ennek köszönhetően asztali, vagy akár mobilalkalmazásban is egyaránt használhatók. A „View” a nézetért és a HTML elemek vezérléséért felelős, amiket megjelenítünk a felhasználónak, egyszerűen a grafikus megjelenést hivatott szolgálni. A „Controller” felel a View és a Modell közötti kapcsolatért. Összességében ez a három pont felelős azért, hogy egy könnyen karbantartható alkalmazást kapjunk.

Viszont arról sem szabad megfeledkeznünk, hogy amikor egy kérés érkezik az alkalmazásba, egy controller kerül kiválasztásra a kérés kezelésére. A megfelelő controller

kiválasztása a route engine feladata. A kiválasztás folyamata kötött szabályokon alapul, amelyet az URL /-jel utáni része határoz meg. Ez azért fontos mert az osztály több metódusból is állhat.

Az asp.net MVC a vezérlő metódusait műveleteknek nevezzük, így pontosabb azt mondani, hogy a vezérlőben végrehajtott művelet felelős a kérés kezeléséért.



2.4. MSSQL -Adatbázis kezelő rendszer



4. ábra: MS SQL Server logó

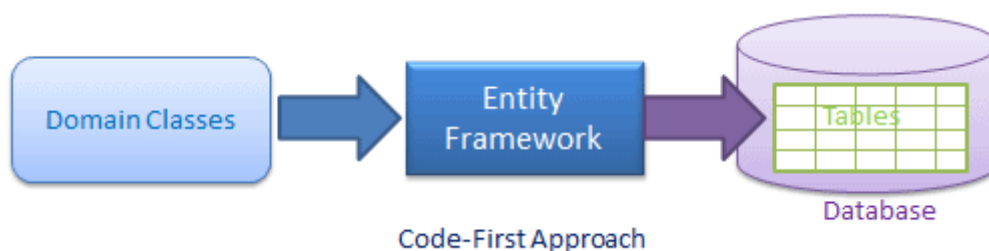
A Microsoft SQL Server egy relációs adatbázis-kezelő rendszer, melyet a Microsoft fejlesztett ki. Adatbázisszerverként ez egy szoftvertermék, amelynek elsődleges funkciója az adatok tárolása és visszakeresése más szoftveralkalmazások által kért módon – amelyek futhatnak ugyanazon a számítógépen vagy egy másik számítógépen a hálózaton (beleértve az internetet is).

A Microsoft a Microsoft SQL Server legalább egy tucat különböző kiadását forgalmazza, amelyek különböző közönségeknek és munkaterhelésnek felelnek meg, a kis egygépes alkalmazásoktól a nagy, internetre néző alkalmazásokig sok egyidejű felhasználóval.

2.5. Code-First

2.5.1. Mi az a Code First?¹

Az Entity Framework bevezette a Code-First megközelítést az Entity Framework 4.1-gyel. A Code-First főként a domain vezérelt tervezésben hasznos. Az alkalmazás domainére összpontosítunk, és először osztályokat hozunk létre a domain entitásai számára, ahelyett, hogy az adatbázis tervezésével kezdenénk és utána hoznánk létre az adatbázisnak megfelelő osztályokat. A következő ábra a Code-First megközelítést szemlélteti.



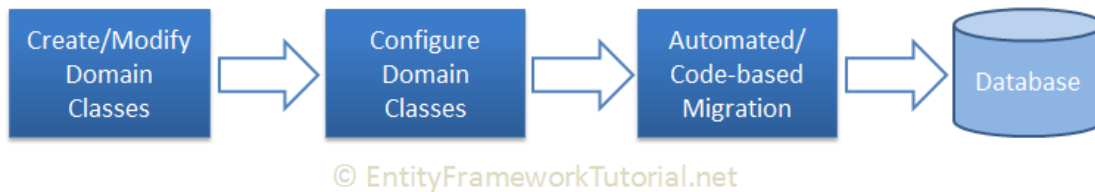
5. ábra: Code-First megközelítés

¹<https://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx>

Amint a fenti ábrán látható, az EF API a domain osztályai és konfigurációja alapján hozza létre az adatbázist. Ez azt jelenti, hogy először el kell kezdenie a kódolást C#-ban vagy VB.NET-ben, majd az EF létrehozza az adatbázist a kódoból.

2.5.2. Code-First munkafolyamat

A következő ábra a Code-First fejlesztési munkafolyamatot mutatja be.



6. ábra: Code-First fejlesztési munkafolyamat

A fejlesztési munkafolyamat a Code-First megközelítésben a következő lenne:

1. Tartományosztályok létrehozása vagy módosítása
2. Ezen tartományosztályok konfigurálása Fluent-API vagy adatfejlesztési attribútumok használatával
3. Adatbázisséma létrehozása vagy frissítése automatizált vagy kódalapú migrációval.

2.6. Mi az a migráció?²

Napjainkban számos értelemben értelmezhetjük a migrációt. Viszont jelen esetben a program szempontjából lesz fontos számunkra. Hiszen amikor új alkalmazást hozunk létre, az adatmodell gyakran változik, és minden alkalommal, amikor a modell változik, nem szinkronizálódik az adatbázissal. Ahhoz, hogy migrációkat tudjunk használni az első lépésben az Entity Framework-öt (későbbiekben EF) kell konfigurálnunk és ha még nem létezik akkor létre kell hoznunk egy adatbázist is. Ez után minden alkalommal, amikor módosítjuk az adatmodellt- új entitáosztályokat adunk hozzá. Ezek eltávolításokat vagy módosításokat hajtanak végre, vagy esetleg módosítják a DbContext osztályt-, esetleg törlik az adatbázist és az EF létrehoz egy újat, amely már megfelel a modellnek, és feltölti azt a tesztadatokkal.

Ez a módszer jól működik az adatbázis és az adatmodell szinkronban tartására mindaddig, amíg az alkalmazást nem helyezik „üzembe”. Amikor az alkalmazás élesben fut, általában olyan adatokat tárol, amiket a későbbiekben megszeretne tartani és nem szeretne mindent elveszíteni minden egyes apró változtatáskor. Például új oszlop hozzáadásakor.

Az Entity Framework Core Migrations szolgáltatás úgy oldja meg ezt a problémát, hogy lehetővé teszi az EF számára az adatbázisséma frissítését új adatbázis létrehozása helyett.

Példa a migrációk használatára (Package Manager Console ablakon keresztül):

- `PM> Add-Migration InitialMigration`
 - áttelepítés létrehozása
- `PM> Update-Database`
 - Adatbázis frissítés
- `PM> Add-Migration TestMigrationFromSeparateProject`
 - Új migráció hozzáadása
- `PM> Remove-Migration`
 - Migráció eltávolítása

² <https://docs.microsoft.com/en-us/aspnet/core/data/ef-mvc/migrations?view=aspnetcore-6.0&viewFallbackFrom=aspnetcore-6.0https%3A%2F%2Fdocs.microsoft.com%2Fen-us%2Faspnet%2Fmvc%2Foverview%2Fgetting-started%2Fgetting-started-with-ef-using-mvc%2Fmigrations-and-deployment-with-the-entity-framework-in-an-asp-net-mvc-applicationhttps%3A%2F%2Fwww.entityframeworktutorial.net%2Fefcore%2Fentity-framework-core-migration.aspx>

3. Rendszer ismertető

3.1. Az alkalmazás követelményei

A rendszer tervezése során a felhasználói felülettel szemben támasztott elvárások:

- A rendszer fő célja egy olyan munkakörnyezet kiépítése, amiben sokkal könnyebb kezelni és átlátni a rendeléseket. Ennek érdekében a fő elvárás a programmal szemben, hogy használata kézenfekvő legyen, és a pincérek a kezelését könnyen el tudják sajátítani, ne legyenek emiatt fennakadások.
- Emellett nem szabad megfeledkeznünk a színösszeállításról sem, ugyanis fontos, hogy a szövegek eltérjenek a háttértől és könnyen olvashatóak legyenek. Továbbá arra is kell figyelni, hogy huzamosabb használat alatt se bántsa a szemet.
- Figyelni kell a felület egységességére, átláthatóságára, hogy a felhasználó ne vesszen el a menüpontok között. Illetve, hogy a működése is logikus legyen.

A programmal, mint informatikai rendszerrel kapcsolatban a következő követelményeket fogalmazhatjuk meg:

- Szükségünk van egy bejelentkező felületre, ahonnan mindenki a saját jogosultsága szerint léphet tovább.
- A felhasználók jelszavát védeni kell, hogy illetéktelenül senki ne férjen hozzá másnak a felhasználói adataihoz.

3.2. Az adatbázis

3.2.1. Mit nevezünk adatbázisnak?

A mindennapi élet szinte minden területén, ha nem is tudunk róla kapcsolatban állunk nem egy adatbázissal. Például van TAJ vagy akár személyi igazolvány számunk, tanfolyami azonosítónk stb. Ezek mind-mind részei egy már működő adatbázisnak. Ugyanakkor vannak kézzelfoghatóbb példák is mint a telefonszámok vagy a könyvek, de akár a feljegyzések is ide tartozhatnak. Ezek is olyan információk, amelyek rendszerezett tárolás nélkül később már nem lesznek számunkra elérhetők.

Az információ és az adat látszólag ugyanazt jelenti és hasonló kontextusban is használják őket. A számítástechnikában az adat az észlelhető, érzékelhető és felfogható ismeretet jelenti, míg az információ az értelmezett adatot. Mindemellett az információ mindig valami újat közöl és ez által döntésre készíttet minket.

A fentiekből kiindulva ahhoz, hogy az adatból információ legyen valamilyen formában fel kell dolgozni azt. Ez azt jelenti, hogy az adat önmagában csak egy jelsorozat, amelyből a feldolgozás során válik információ.

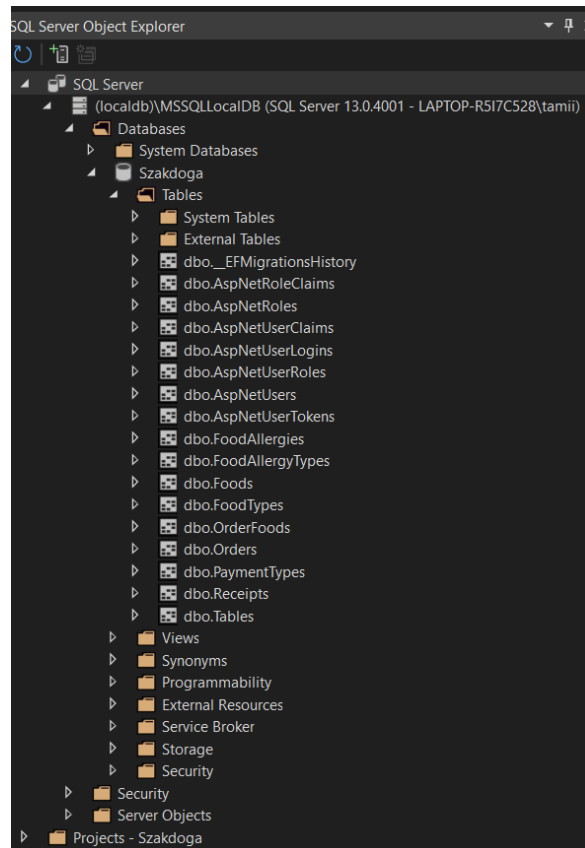
Az adatállomány olyan összefüggő adathalmaz, amelyben minden olyan adat megtalálható, amire egy bizonyos cél megvalósítása érdekében szükség lehet.

Adatbázison köznapi értelemben valamely rendezett, valamilyen szisztéma szerint tárolt adatokat értünk, melyek nem feltétlenül számítógépen kerülnek tárolásra. Az adathalmaz csak akkor válik adatbázissá, ha az valamilyen rend szerint épül fel, mely lehetővé teszi az adatok értelmes kezelését. Természetesen ugyanazon adathalmazból többféle rendszerezés alapján alakíthatunk ki adatbázist.

3.2.2. Adatbázis a programban

Mint ahogy fentebb már említettem az adatbázis nagyon fontos alkotóelem, mint a való életben mint pedig a programok szempontjából is. Hiszen a program az ebben lévő adatokkal dolgozik legyen szó akár egy könyvtáras, vagy jelen esetben egy éttermi szoftverről. Ugyanis a tervezés során az adatbázisra vonatkozó normálformák adnak útmutatást. Ezek segítségével sikeresen megtervezhető egy redundancia mentes, jól működő adatbázis, amelynek modellezése nagymértékben megkönnyítheti a program elkészítését, melynek szorosan az adatbázis struktúrája köré kell épülnie. A tesztelés során számos adattal dolgoztam melyek közül párat automatikusan visszatölt az adatbázis amikor azt újra feltöltjük a programba. Ilyenek például az ételkategóriák és az allergének, illetve az asztalok.

Az adatbázis az MS SQL Server 2016-os verziójával készítettem el.



7. ábra: Adatbázis táblák a Visual Studioban

3.3. Koncepcionális (tervezési) szint

A következő elvárások fogalmazódtak meg az adatok tárolásával kapcsolatban:

1. A legfontosabb, hogy biztonságosan tároljuk a felhasználók jelszavát, melyet PBKDF2 (Password-Based Key Derivation Function 2) használatával oldottam meg.

PBKDF2 rövid leírása³

A PBKDF2 egy pszeudovéletlen függvényt, például hash-alapú üzenet-hitelesítési kódot (HMAC) alkalmaz a bemeneti jelszóra vagy jelmondatra a sóértékkel együtt, és sokszor megismétli a folyamatot, hogy előállítson egy származtatott kulcsot, amelyet aztán kriptográfiai kulcsként használhat a későbbi műveleteknél. A hozzáadott számítási munka sokkal nehezebbé teszi a jelszavak feltörését, és kulcsnyújtásnak nevezik.

A PBKDF2 kulcslevezetési funkciónak öt bemeneti paramétere van:

```
DK = PBKDF2 (PRF, jelszó, só, c, dkLen)
```

8. ábra: PBKDF2 titkosítás

ahol:

- A *PRF* két paraméter pszeudovéletlen függvénye, kimeneti hossza *hLen* (pl. kulcsos HMAC)
- A *jelszó* az a fő jelszó, amelyből a származtatott kulcs generálódik
- A *só* bitek sorozata, amelyet kriptográfiai sóként ismernek
- *c* a kívánt iterációk száma
- A *dkLen* a származtatott kulcs kívánt bithossza
- A *DK* a generált származtatott kulcs

Példa (PBKDF2HmacSHA512-vel):

Eredeti jelszó: TitkosítottJelszó1

Titkosított jelszó: f8Sk3xmzqkKC++zRfSBFLw==

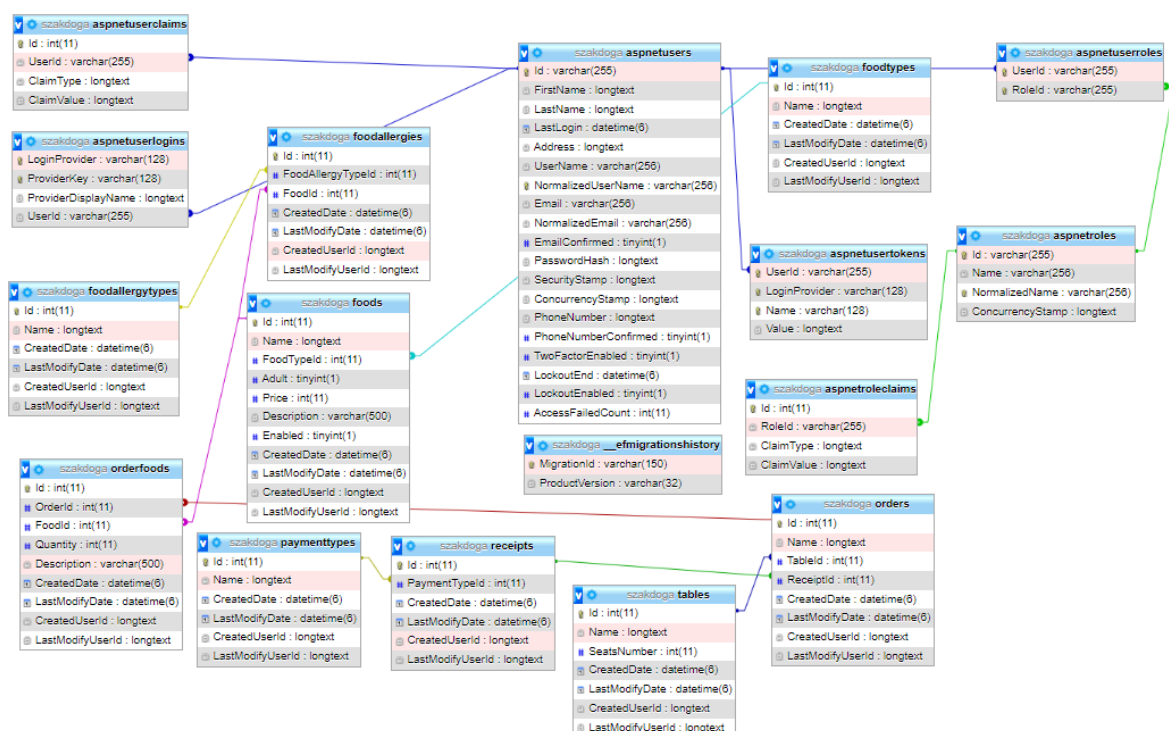
³Az eredeti cikk angol nyelvű: <https://en.wikipedia.org/wiki/PBKDF2>

2. Fontos megemlítenünk, hogy az alkalmazás, használatához elengedhetetlen egy felhasználónév és egy jelszó. Továbbá egy jogosultság is, amely segítségével eldönti a program, hogy melyik menüpontok és folyamatok lesznek elérhetőek a felhasználók számára.
3. A programban van lehetőség arra, hogy új felhasználókat regisztráljuk és új ételeket, kategóriákat, allergéneket és asztalokat is felvehetünk.
4. A fentemlítettekből az következik, hogy a felhasználók könnyedén feltudnak venni rendeléseket is.

3.4. Fizikai (implementációs) szint

A koncepcionális tervezés alatt meghatároztuk, hogy mely adatokra lesz szükségünk a program kivitelezése során. Viszont mindez nem lesz elég a teljes rendszer kiépítéséhez, hiszen felkell vennünk elsődleges kulcsokat melyek segítségével azonosítani tudjuk az adott elemeket és kapcsolatokat a táblák között. Ez a folyamat azért fontos, hogy az adatbázisunk ne legyen redundáns és hogy könnyen elérjük az adatainkat az adatbázisból.

Ahhoz, hogy ilyen adatbázist kapjunk először végig kell haladnunk a normálformákon, hogy a tervezési fázisban lévő adathalmazunkból egy jól működő adatbázist kapjunk.



9. ábra: Adatbázis – kapcsolat diagram a rendszerről

4. A rendszer megvalósítása

4.1. Telepítés

4.1.1. A rendszer követelményei

- ASP.NET Core RunTime 6 Hosting Bundle
 - [Download .NET 6.0 \(Linux, macOS, and Windows\) \(microsoft.com\)](https://www.microsoft.com/net/core/linux-macos)
- Internet Information Services (IIS)
- Windows 10/11 vagy Windows Server 2016 vagy újabb operációs rendszer
- Microsoft SQL Server 2016 local DB
- Böngésző

4.1.2. Telepítés folyamata

1. IIS telepítése
2. ASP.NET Core RunTime 6 Hosting Bundle telepítése
3. IIS site létrehozása
4. Alkalmazás állományainak másolása site mappába
5. SQL local DB telepítés

4.2. Megjelenés

A weboldal megtervezése során nagy hangsúlyt fektettem arra, hogy a felület egyszerű és könnyen átláthatón legyen. Ezzel is elősegítve a dolgozó munkáját, hiszen véleményem szerint egy egyszerű felületet sokkal könnyebb átlátni, ez által is növelve a tényleges munkával töltött időt. Hiszen, ha gyors a rendelés felvétel akkor az adott étel is hamarabb elkészülhet. Ez pedig azt eredményezi, hogy a vendégek sokkal jobb szájízzel és véleménnyel távoznak az étteremből és még inkább ajánlják azt barátaiknak és ismerőseiknek.

Mindemellett fontos megemlíteni, hogy a weboldal reszponzív tehát bármely felülethez alkalmazkodik. A táblázatok megjelenéséhez pedig a DataTables⁴ nevű templatet használtam, mely segítségével egy áttekinthető táblázatot kaptam. Melyben könnyedén

⁴DataTables: <https://datatables.net/>

tudunk keresni és kedvünk szerint állíthatunk be rendezést a címsorokra. Ezenfelül, ha a rekordok elérnek egy bizonyos szintet a kliens könnyen kezelheti azt, hogy hány adat jelenjen meg a táblázatban. Valamint, ha kinőtük a megadott mennyiséget, akkor lapozhatunk is az oldalak között.

4.3. Bejelentkező felület

A program elindítását követően egy olyan oldal jelenik meg előttünk, amely figyelmeztet minket, arra, hogy jelentkezünk be. A bejelentkezés a megadott felhasználónév és jelszó megadásával kezdődik. Amikor a felhasználó rákattint a „Bejelentkezés” gombra a háttérben a program megvizsgálja a megadott adatokat és lekéri azt, hogy az adott felhasználó milyen jogosultsággal rendelkezik. Jelen esetben két definiált jogosultságot különböztetünk meg egy 'user'-t és egy 'boss'-t. Ez a későbbiekben még fontos lesz számunkra, hiszen vannak bizonyos menüpontok és funkciók melyek csak a 'boss'-nak érhetőek el. Addig amíg a felhasználó nem jelentkezik be az alkalmazásba addig a fő menüpontok levannak tiltva ezzel is védve az oldal biztonságát.


Szakdoga Főoldal

Bejelentkezés

Üdv
Jelentkezz be!

10. ábra: Bejelentkezés előtti főoldal

Szakdoga Főoldal Bejelentkezés



Bejelentkezés

Felhasználónév
boss@localhost.com

Jelszó
.....

☐ Emlékezz rám

Bejelentkezés

11. ábra: Bejelentkezés

4.4. Menüpontok

Szakdoga Főoldal Emberek Rendelés Asztalok ▾ Ételek ▾ Hello user@localhost.com! Kijelentkezés

Asztalok ▾ Ételek ▾

Lista
Új asztal hozzáadás

Ételek ▾

Lista

Új étel

Új étel típus

Új allergén

12. ábra: Menüpontok

4.4.1. Főoldal

Miután a kliens bejelentkezett az oldalba, -jogosultságtól függetlenül-, egy olyan felületet lát, amely minden aktuális rendelést kilistáz. Először is a program megnézi azt, hogy milyen napot írunk és ennek függvényében megvizsgálja az adatbázisban található

25

összes rendelést, amelyet korábban felvettünk és amelyek egyeznek a mai dátummal azoknak kiírja a nevét és azt, hogy melyik asztalhoz tartoznak. Továbbá azt, hogy mennyit rendeltek az adott ételből és hogy mennyi lesz a fizetendő összeg.

Szakdoga Főoldal Emberek Rendelés Asztalok ▼ Ételek ▼ Hello boss@localhost.com! Kijelentkezés

Aktuális napi rendelések

Show 10 ▼ entries Search:

Asztal ▲	Étel név	Mennyiség	Teljes	Megjegyzés
Közepes asztal	Sparé ribs oldalas	2	5900	
Nagy asztal	Tökmag morzsás túrógombóc	10	9500	
Nagy asztal	Tökmag morzsás túrógombóc	10	9500	
Terasz	Szarvaspörkölt	2	6400	elvitel
Terasz3	Gesztenyepüre	2	1600	

Showing 1 to 5 of 5 entries Previous 1 Next

13. ábra: Aktuális napi rendelések

4.4.2. Emberek

Az 'Emberek' menüpontban a bejelentkezett felhasználó megtekintheti a programban regisztrált egyén adatait. Mint ahogy a bal felső sarokban is látható van lehetőség új dolgozó hozzáadására, viszont új dolgozót csak a boss jogosultsággal rendelkező személy vehet fel.

[Új dolgozó hozzáadása](#)

Felhasználó név	Vezetéknév	Keresztnév	E-mail
boss@localhost.com	Boss	System	boss@localhost.com
user@localhost.com	User	System	user@localhost.com

14. ábra: Emberek menüpont

Továbbá fontos megemlítenünk, hogy amikor új felhasználót veszünk fel a jelszónak tartalmaznia kell legalább egy nagy betűt (A-Z), kis betűt (a-z), számot (0-9) illetve egy speciális karaktert. Különben a regisztráció sikertelen lesz és hibaüzenetet

kapunk, miután a megadott feltételeknek eleget téve regisztrálhatunk.

Az alkalmazásba való legelső belépéskor két regisztrált felhasználóval léphetünk be az egyik a user mely a user@localhost.com felhasználónévre hallgat a jelszava pedig Admin123*, illetve a boss akinek boss@localhost.com a felhasználóneve és a jelszava változatlanul Admin123*.

- Passwords must have at least one non alphanumeric character.
- Passwords must have at least one lowercase ('a'-'z').
- Passwords must have at least one uppercase ('A'-'Z').

Vezetéknév
Minta

Keresztnév
Pál

Lakcím
1234 Valahol, Valamilyen utca 12.

Felhasználó név
Pál

E-mail
pal04@localhost.com

Jelszó

Jelszó megegyeszer

Register

15. ábra: Helytelen jelszó

4.4.3. Asztalok

Asztalok menüpontban van opció arra, hogy új asztalokat hozzunk létre, illetve módosítjuk a már meglévőket vagy ha a már meglévő asztalunk nem aktuális többé akkor van arra is lehetőség, hogy kitöröljük azt.

Asztalok

[Új felvétel](#)

Show entries
Search:

Asztal neve	Székek száma	
Kis asztal	4	Módosítás Részletek Törlés
Közepes asztal	8	Módosítás Részletek Törlés
Nagy asztal	12	Módosítás Részletek Törlés
Terasz	6	Módosítás Részletek Törlés
Terasz	2	Módosítás Részletek Törlés

Showing 1 to 5 of 5 entries

Previous
1
Next

16. ábra: Asztalok

4.4.4. Ételek

Ez a menüpont felel az étlapért, hiszen ennek a menüpontnak a segítségével tudunk új ételeket regisztrálni az adatbázisban. Továbbá, mint ahogy már fentebb említettem itt vehetünk fel új allergéneket és ételtípusokat is, melyeket szintén igény szerint módosíthatunk, törölhetünk. Azonban amikor a felhasználó felveszi az új ételt és megadja a hozzá tartozó allergéneket a táblázatban ezek nem jelennek meg, de ha belemegy a részletekbe, akkor a program minden olyan adatot kiír, amely hozzá tartozik az adott ételhez.

Mint ahogy minden étteremben itt is lehet szezonális ételeket is felvenni melyet a 'boss' könnyedén letilthat és engedélyezhet az 'Elérhető' mezővel, ha a mező bevan pipálva akkor az étel megjelenik a listában a 'user' számára is. Ellenben, ha egy meglévő ételt szeretnénk újra elérhetővé tenni, akkor azt csak akkor tudjuk megcsinálni, ha 'boss' jogosultsággal rendelkezünk. Hiszen az ő jogosultságával rendelkező személy azokat az ételeket is látja, amelyek jelen pillanatban levannak tiltva.

Étlap

[Új étel hozzáadása](#)Show entriesSearch:

Étel neve	Az étel típusa	Korhatáros	Az étel ára	Megjegyzés	Elérhető	
Gesztenyepüre	Desszert	Nem	800 Ft		Nem	Módosítás Részletek Törlés
Hosszúkává	Ital	Nem	350 Ft		Igen	Módosítás Részletek Törlés
Margagulyás	Leves	Nem	1150 Ft	magvas veknival	Igen	Módosítás Részletek Törlés
Sparé ribs oldalas	főétel	Nem	2950 Ft	sült burgonya	Igen	Módosítás Részletek Törlés
Staropramen	Ital	Igen	700 Ft	korsó	Igen	Módosítás Részletek Törlés
Szarvaspörkölt	főétel	Nem	3200 Ft	Dödölle	Igen	Módosítás Részletek Törlés
Tökmag morzsás túrógombóc	Desszert	Nem	950 Ft	Sós karamell	Igen	Módosítás Részletek Törlés

Showing 1 to 7 of 7 entries

Previous Next

17. ábra: Ételek menüpont, Lista

4.4.5. Rendelések

4.4.5.1. Mi található a menüpontban?

A Rendelések az a menüpont, ami összefogja az összes többi, hiszen ebben a részben tudunk új rendeléseket felvenni vagy a későbbiekben törölni amíg a Mentés gombbal nem mentjük el azokat az adatbázisba.

4.4.5.2. Hogy működik a rendelésfelvétel?

Szakdoga
Főoldal
Emberek
Rendelés
Asztalok ▾
Ételek ▾
Hello boss@localhost.com!
Kijelentkezés

Rendelések

Rendelések

Étel:
Rántott hús

Asztal:
Közepes asztal

Egységár:
1200

Mennyiség:
2

Kedvezmény:
0

Teljes:
2400,00

Megjegyzés:
sültkrumplival

Hozzáadás a listához

Felvett rendelések

Étel név	Asztal	Egységár	Mennyiség	Kedvezmény	Teljes	Megjegyzés	
Mákosguba	Virtuális	800	5	500	3500	Elvitelre	Eltávolítás

Mentés

18. ábra: Rendelésfelvétel

Mint ahogy az a képen is látszik van egy 'Étel' mező, amely lekéri az adatbázisból az összes regisztrált ételt, pont úgy, mint az asztalokat. Mindeközben egy másik szál megvizsgálja annak az ételnek az egységárát melyet szintén az adatbázisból kapunk meg, mellyel a későbbiekben gondtalanul kitudjuk számolni a teljes árat.

Ugyanis a kódban készítettem egy 'számológépet', amely az adott egységárat megszorozza a mennyiséggel és ezek után kivonja belőle a kedvezményt (a kedvezmény Forintban értetendő). A rendelés felvétel közben van mód megjegyzést is hozzáfűzni, ez akkor lehet fontos számunkra, ha a már felvett ételt a vendék esetleg más körettel kéri vagy esetleg elviszi azt. Így tudjuk jelezni a konyhán dolgozóknak az esetleges változást. Majd, ha minden adatot sikeresen felvettünk a 'Hozzáadás a listához' gombra kattintva elkészül a rendelésünk, melyet a Mentés gombbal elmenthetünk az adatbázisba továbbá a főoldalon található aktuális napi rendelések közé. Ez egy biztos módszer arra, ha a későbbiekben vissza szeretnénk keresni egy rendelést.

5. Tesztelés⁵

A tesztelés talán a program legfontosabb része, mivel ez alatt az idő alatt dől el, hogy a program az elvárásoknak megfelelően működik-e. Ez a folyamat nem a program megírása után zajlik, hanem vele egy időben, folyamatosan. A tesztelés célja, hogy a program fejlesztése során keletkezett hibákból minél több feltáruljon, és rájövünk, hogy ezeket mi okozza, ugyanis egy esetleges hibát később sokkal nehezebb kijavítani.

Az általam készített rendszer tesztelése során próbáltam a legtöbb hibát kiszűrni, emiatt igyekeztem minél több módszert kipróbálni viszont a legjobbnak mégis az bizonyult amikor egy laikusnak adtam a programomat így használat közben számos olyan dologra fény derült mely javításra szorult.

5.1. Unit teszt⁶

Unit tesztelés alatt azt értjük, amikor a szoftvernek csak egy kisebb komponensét vagy elemét teszteljük. Mivel minden egyes ilyen teszt működési területe igen behatárolt, az egyetlen járható út, ha írunk egy kódot a kód tesztelésére olyan keretrendszerek segítségével, mint a NUnit vagy a Microsoft Testing Framework. E keretrendszerek működési elvének részletes ismertetése nem témája ennek a cikknek, dióhéjban az unit tesztelés tehát csak annyi, hogy a fejlesztő különböző teszt metódusokat ír, amelyeket lefuttatva ellenőrizni tudja, hogy minden programegység megfelel-e a specifikációjának.

5.1.1. Előnyei

1. Egy rendszer átlagos tesztelése során be kell jelentkezni és egy sor előre meghatározott műveletet kell elvégezni annak érdekében, hogy a rendszer egy bizonyos funkcióját ellenőrizhessük. Ez egyáltalán nem hatékony és borzasztóan időigényes. A unit tesztelés tehát lehetőséget ad a fejlesztőnek arra, hogy célzott ellenőrzést végezzen a rendszer kérdéses területein.
2. Ha valami nem működik, a fejlesztő csapatnak nem kell átfésülni az egész rendszert a hiba megtalálásához, elég, ha lefuttatják a korábban megírt unit teszteket, leszűkítve ezzel a keresési területet.

⁵ Simon Levente Szakdolgozata

⁶ <https://szakembereknek.schonherzbazis.hu/cikkek/Miert-ertjuk-felre-a-unit-tesztelest>

3. Végül, de nem utolsósorban a kód meghatározott időközönkénti újra tervezésével való javítása elengedhetetlen a rendszer hosszú távú fenntarthatósága érdekében.

5.1.2. Mikor érdemes tesztet írni?

- Ha a metódus mögött meghúzódó logika annyira komplex, hogy szükséges egységenként is megvizsgálni, vajon tényleg működik-e.
- Ha a kód egy bizonyos funkciója meghibásodik és a javítása egy percnél hosszabb időt vesz igénybe.
- Ha a teszt megírása kevesebb időt vesz igénybe, mint a program lefuttatása, a bejelentkezés vagy a modell újra tervezése

5.1.3. Mikor nem érdemes tesztet írni?

- Ha bonyolult keretrendszereket kell készíteni vagy telepíteni azért, hogy az unit tesztek működjenek (pl. szimulált objektumok, függőségi befecskendezés (DI)).
- Ha a meghibásodott kódon lefuttatott unit teszt csak kis mértékben javít az egész program minőségén.
- Ha az egységtesztek karbantartása magasabb költségekkel jár, mint maga a szoftver karbantartása.

5.1.4. Összefoglalás

Összefoglalva tehát, az unit tesztek funkciója az, hogy segítse a fejlesztő csapatokat a költségek leszorításában, a tesztidő lerövidítésében, a regressziós tesztek lecsökkentésében és a karbantartás megkönnyítésében. A unit tesztelés fontos folyamat, ha azt akarjuk, hogy a program sikeres legyen. Azok a fejlesztők, akik túl sok unit tesztet tárolnak és kezelnek, sokkal inkább előidézik azokat a problémákat, amelyeket e tesztek megoldani voltak hivatottak.

5.2. Regressziós teszt⁷

A regressziós teszt a szoftver tesztelésének egy típusa, amely megerősíti, hogy a közelmúltbeli program- vagy kódmódosítás nem befolyásolta hátrányosan a meglévő szolgáltatásokat. A regressziós tesztelés nem más, mint a már végrehajtott tesztesetek teljes vagy részleges kiválasztása, amelyeket a meglévő funkciók megfelelő működésének biztosítása érdekében újra végrehajtanak.

⁷ <https://hu.csstricks.net/8222511-what-is-regression-testing-definition-test-cases-example#menu-3>

Ezt a tesztet annak biztosítására végzik, hogy az új kódváltozásoknak ne legyenek mellékhatásai a meglévő funkciókra. Biztosítja, hogy a régi kód továbbra is működjön, ha a legfrissebb kódváltozásokat elvégezték.

5.2.1. Mikor szükséges a regressziós teszt?

A regressziós tesztelés szükségessége főleg akkor merül fel, amikor szükség van a kód megváltoztatására, és meg kell vizsgálnunk, hogy a módosított kód befolyásolja-e a szoftveralkalmazás másik részét vagy sem. Ezenkívül regressziós tesztre van szükség, amikor új funkciót adnak a szoftveralkalmazáshoz, a hibák javításához, valamint a teljesítményproblémák javításához.

5.2.2. Hogyan kell elvégezni a regressziós tesztet?

A regressziós tesztelés elvégzéséhez először ki kell derítenünk a kódot a hibák azonosításához. Miután a hibákat azonosították, a javításhoz szükséges módosításokat hajtják végre, a regressziós tesztet úgy végezzük, hogy a tesztcsomagból kiválasztunk releváns teszteseteket, amelyek a kód módosított és érintett részeire egyaránt kiterjednek. A szoftverkarbantartás olyan tevékenység, amely magában foglalja a fejlesztéseket, a hibajavításokat, a meglévő szolgáltatások optimalizálását és törlését. Ezek a módosítások a rendszer hibás működését okozhatják. Ezért szükségessé válik a regressziós tesztelés.

5.2.3. Tesztesetek

1. Tesztesetek, amelyekben gyakran vannak hibák
2. A felhasználók számára jobban látható funkciók
3. Tesztesetek, amelyek igazolják a termék alapvető jellemzőit
4. A több és újabb változáson átesett funkciók tesztesetei
5. Minden integrációs teszteset
6. Minden komplex teszteset
7. Határérték-teszt esetek
8. Minta a sikeres tesztesetekből
9. Példa a kudarcteszt esetekre

5.3. Integrációs tesztelés

5.3.1. Az integrációs tesztelés stratégiái

A szoftverfejlesztés különböző stratégiákat határoz meg az integrációs teszt elvégzéséhez, nevezetesen:

- Big Bang megközelítés.
- Inkrementális megközelítés. Ez tovább oszlik a következőre
 - Felfelé irányuló megközelítés.
 - Alulról felfelé irányuló megközelítés.
 - Inkrementális megközelítés - felülről lefelé és alulról felfelé kombinálva.

5.3.1.1. Big Bang megközelítés

Minden egység és más alkatrész egyszerre integrálódik, és egyszerre tesztelünk.

Előnyei:

- Kényelmes kis rendszerekhez.
- Biztosak vagyunk benne, hogy az integrált egészet tesztelték.

Hátránya:

- A hibák megnehezítése nehéz.
- A nagyszámú interfész miatt, amelyet ebben a megközelítésben tesztelnünk kell, könnyen kimaradhat néhány link.
- Mivel csak akkor kezdhetünk integrációs tesztekkel, ha az összes modul kódolása megtörténik, több időt vesz igénybe a teszt megfelelő időpontban történő ütemezése.
- Az integrációs tesztelés ideje gyakran függ az elfogadási tesztől.
- Mivel az összes modult egyidejűleg tesztelik, a kritikus nagy kockázatú modulok nincsenek szigetelve, és nem tesztelték extra keményen.
- A modulokkal kapcsolatos perifériás eszközöket nem lehet külön-külön tesztelni.

5.3.1.2. Inkrementális megközelítés

Ebben a megközelítésben két vagy több, logikailag összefüggő modul összekapcsolásával tesztelünk. Ezután hozzáadjuk a többi kapcsolódó modult a teszthez. Ez létrehoz egy láncot, amelynek működését teszteljük. Ez a folyamat mindaddig folytatódik, amíg nem egyesítjük és teszteljük az összes modult.

A növekvő megközelítést két különböző módszerrel hajtják végre:

- Bottum fel.
- Felfelé.

6. Továbbfejlesztési lehetőségek

Mint tudjuk soha nem mondhatjuk egy programról, szoftverről azt, hogy teljesen készen van. Mindig van min változtatni és javítani, pont úgy, mint ahogy ezen is. Az elképzelésem szerint a következő módosításokkal lehetne még továbbfejleszteni a szakdolgozatomat:

- Számlázás: A program az eltárolt rendelések alapján kiállítani egy számlát, amit egy blokknyomtató segítségével kinyomtatna a vendék és az étterem számára.
- Fizetés: A későbbiekben van mód a fizetés kialakítására is.
- Rendelések rendezése: Az aktuális napi rendelések mellett a korábbi rendelések kilistázása egy másik menüpontban.
- Reszponzivitás lehetséges apróbb hibáinak javítása
- Házhozszállítás: A későbbiekben, ha a felhasználó igényt tart rá akkor kilehet alakítani egy olyan környezetet is az oldalon, ahol a vendégek saját maguknak tudnak ételek rendelni.
- Mobil alkalmazás fejlesztés

7. Összegzés

A szakdolgozatom segítségével bemutattam, hogy hogyan is lehet elkészíteni egy éttermi szoftver alkalmazást. Kezdve az elején a tervezéssel, amely sok időbe telt, de megérte, mivel így volt egy biztos alapom, aminek segítségével el tudtam kezdeni az alkalmazást.

Miután a tervezéssel elkészültem, elkezdtem ismerkedni az ASP.NET keretrendszerrel és az MVC-vel. Mivel a keretrendszer a .NET programozási nyelvet támogatja, így választanom kellett. A tanulmányaim alatt megismerkedtem a C# nevezetű programozási nyelvvel, és érdekesnek is találtam a használatát, ezért döntöttem úgy, hogy az alkalmazás ennek a nyelvnek a használatával fog elkészülni. A tervezés után elkészült a felhasználói felület, amiben így már könnyedén beépíthetők a szerver oldali funkciók.

A kódolás alatt, a legnagyobb problémát az adatbáziskezelő műveletek okozták, de még így is nagy segítséget nyújtott az Entity Framework, amelynek segítségével nem volt szükség SQL scriptek írására, hanem különböző funkciók meghívásával volt lehetőség adatokat felvitelére, módosítására stb.

Az elkészült alkalmazással van lehetőség a rendelések nyilvántartására és felvételére.

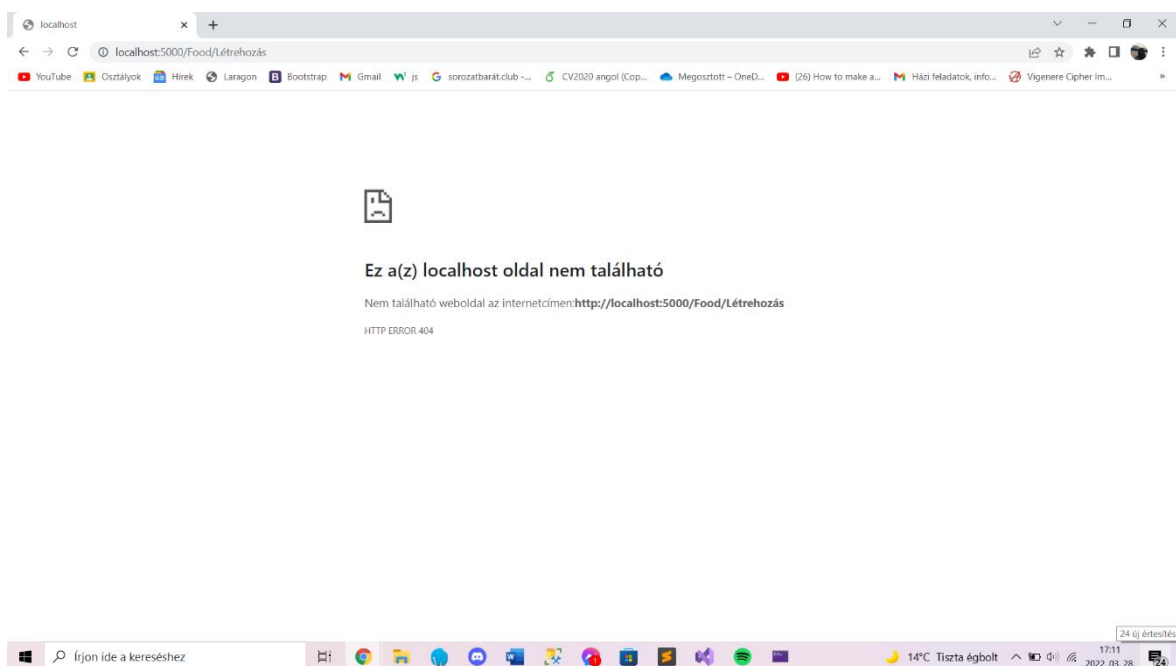
8. Tesztekhez végzett kód, teszteredmények

8.1. Új étel hozzáadása

Tesztelés során felmerült egy apróbb hiba, mely az új étel hozzáadásakor keletkezett, mivel elírtam egy szót. Ez miatt nem látta a program a megadott oldalt amire tovább szerettem volna lépni.

```
<div class="row">
  <div class="col-md-4">
    <form asp-action="Létrehozás">
      <div asp-validation-summary="ModelOnly" class="text-danger"></div>
      <div class="form-group">
        <label asp-for="Name" class="control-label"></label>
        <input asp-for="Name" class="form-control" />
      </div>
    </form>
  </div>
</div>
```

19. ábra: Hibás kód a kódban



20. ábra: Hibás kód az oldalon

Tesztekhez végzett kód, teszteredmények

```
<div class="col-md-4">  
  <form asp-action="Created">  
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>  
    <div class="form-group">  
      <label asp-for="Name" class="control-label"></label>  
      <input asp-for="Name" class="form-control" />  
    </div>  
  </form>  
</div>
```

21. ábra: Javított kód a kódban

Új étel felvétel

Étel neve

Az étel típusa
Válassz Kategoriót

☐ Korhatáros

Az étel ára
0

Megjegyzés

Allergének
Nothing selected

☐ Élérhető

Létrehozás

Vissza a listára

22. ábra: Javított kód az oldalon

8.2. Asztalok kezelése

Az elmulasztott await szó miatt a program nem várta meg az adatot.

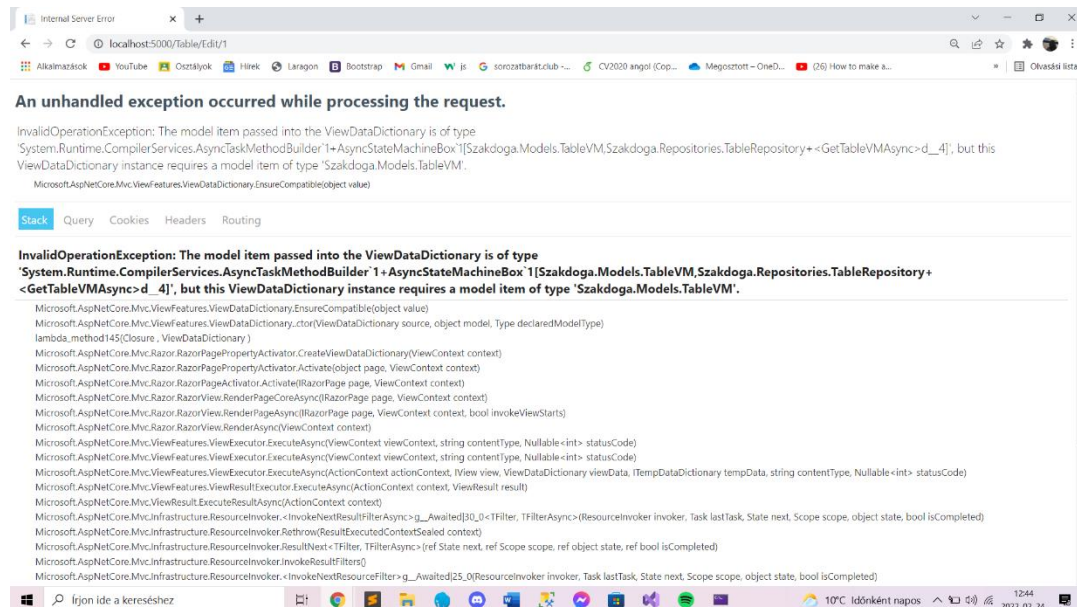
```
// GET: Tables/Delete/5
0 references
public async Task<IActionResult> Delete(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var table = tableRepository.GetAsync(id);
    if (table == null)
    {
        return NotFound();
    }

    return View(table);
}

// POST: Tables/Delete/5
[HttpPost, ActionName("Delete")]
```

23. ábra: Hibás kód a programban



24. ábra: Hibás kód az oldalon

```
var table = await tableRepository.GetAsync(id);
```

25. ábra: Javított kód

9. Ábrajegyzék

1. ábra: Microsoft .NET Framework logó	8
2. ábra: C# logó	10
3. ábra: MVC	12
4. ábra: MS SQL Server logó	14
5. ábra: Code-First megközelítés	14
6. ábra: Code-First fejlesztési munkafolyamat	15
7. ábra: Adatbázis táblák a Visual Studioban	19
8. ábra: PBKDF2 titkosítás	20
9. ábra: Adatbázis – kapcsolat diagram a rendszerről	22
10. ábra: Bejelentkezés előtti főoldal	24
11. ábra: Bejelentkezés	25
12. ábra: Menüpontok	25
13. ábra: Aktuális napi rendelések	26
14. ábra: Emberek menüpont	26
15. ábra: Helytelen jelszó	27
16. ábra: Asztalok	28
17. ábra: Ételek menüpont, Lista	29
18. ábra: Rendelésfelvétel	30
19. ábra: Hibás kód a kódban	38
20. ábra: Hibás kód az oldalon	38
21. ábra: Javított kód a kódban	39
22. ábra: Javított kód az oldalon	39
23. ábra: Hibás kód a programban	40
24. ábra: Hibás kód az oldalon	40
25. ábra: Javított kód	40

10. Irodalomjegyzék

- [1] *C# története, programozás:* <http://programozas.kereso.info/>(2022.02.22)
- [2] A C# programozás alapjai – C# programozás kezdőknek:
<https://codeberryschool.com/blog/hu/a-c-sharp-programozas-alapjai/> (2022.02.22)
- [3] Anders Hejlsberg: https://hu.wikipedia.org/wiki/Anders_Hejlsberg (2022.02.22)
- [4] Microsoft SQL Server: https://en.wikipedia.org/wiki/Microsoft_SQL_Server