NEUMANN JÁNOS
INFORMATIKAI KAR

SZAKDOLGOZAT

OE-NIK
2021

Hallgató neve: **Juhász János**
Hallgató törzskönyvi száma: **T/006445/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai Rendszerek Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Juhász János**
Törzskönyvi száma: T/006445/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

Webáruház teljes körű fejlesztése
Fully development of a web store

Intézményi konzulens: Rusznák Attila
Külső konzulens:

Beadási határidő: 2021. december 15.

A záróvizsga tárgyai: Számítógép architektúrák
Big Data és üzleti intelligencia specializáció

A feladat

A hallgató feladata egy webáruház teljes körű fejlesztése egy megrendelő felkérésére, egy információs rendszer tervezése, amelynek célja a rendelések nyomon követése és segítségnyújtás a dolgozóknak a teljesítésében. A feladat végrehajtása során elemezze az ügyféli igényeket és dolgozzon ki egy olyan követelményspecifikációt a megrendelővel közösen, ami ellátja az egyes funkciókat, üzleti folyamatokat. Tervezze meg, fejlessze ki és tesztelje le az azokat támogató szoftver és adatbázis sémákat, amelyek kielégítik az ügyféli igényt.

A dolgozatnak tartalmaznia kell:

- a feladat részletes ismertetését,
- szoftverkörnyezet bemutatását,
- a követelmény specifikációt,
- a rendszertervet, amely tartalmazza a funkciókat és az adatmodellt,
- az elért eredményeket: a kész webáruházat és ennek működését,
- a továbbfejlesztési lehetőségeket..

Ph.

Fleiner R.
.....
Dr. Fleiner Rita
intézetigazgató

A szakdolgozat elévülésének határideje: **2023. december 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

Ruscsák AAik
.....
intézményi konzulens

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2021.12.15.

.....

hallgató aláírása

KONZULTÁCIÓS NAPLÓ

Hallgató neve:	Neptun Kód:	Tagozat:
Juhász János	[redacted]	Nappali

Mérnök informatikus BSC 2018

Telefon: [redacted]

Levelezési cím (pl.: lakcím): [redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul: Webáruház teljes körű fejlesztése

Szakdolgozat / Diplomamunka[2] címe angolul: Fully development of a web store

Intézményi konzulens:
Rusznák Attila

Külső konzulens:

Kérjük, hogy az adatokat nyomtatott nagy betűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	**2021.02.12**	**Szakdolgozat felépítésének áttekintése**	Rusznák AAik
2.	**2021.03.19**	**Formai követelmények egyeztetése**	Rusznák AAik
3.	**2021.04.30**	**Szakmai kérdések megbeszélése, tisztázása**	Rusznák AAik
4.	**2021.05.7**	**Leadás előtti utolsó simítások**	Rusznák AAik

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplomamunka 2 / Diplomamunka 3 / Diplomamunka 4 [3] tantárgy követelményét teljesítette, beszámolóra / védésre [4]bocsátható

Rusznák AAik
.....
Intézményi konzulens

Budapest, 2021.05.14.

[1] Megfelelő aláhúzandó!

[2] Megfelelő aláhúzandó!

[3] Megfelelő aláhúzandó!

[4] Megfelelő aláhúzandó!

KONZULTÁCIÓS NAPLÓ

Hallgató neve:	Neptun Kód:	Tagozat:
Juhász János	[redacted]	Nappali

Mérnök informatikus BSC 2018

Telefon: [redacted]

Levelezési cím (pl.: lakcím): [redacted]

Szakdolgozat / Diplomamunka[5] címe magyarul: Webáruház teljes körű fejlesztése

Szakdolgozat / Diplomamunka[6] címe angolul: Fully development of a web store

Intézményi konzulens:	Külső konzulens:
Rusznák Attila	

Kérjük, hogy az adatokat nyomtatott nagy betűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	**2021.09.24**	**Projekt megvalósításának megbeszélése**	Rusznák AAik
2.	**2021.10.15**	**Felmerülő nehézségek tisztázása**	Rusznák AAik
3.	**2021.10.29**	**A projekt dokumentáció követelményei**	Rusznák AAik
4.	**2021.11.26**	**Leadással kapcsolatos kérdések egyeztetése**	Rusznák AAik

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplomamunka 2 / Diplomamunka 3 / Diplomamunka 4 [7] tantárgy követelményét teljesítette, beszámolóra / védésre [8]bocsátható

Rusznák AAik
.....

Intézményi konzulens

Budapest, 2021.12.15.

[5] Megfelelő aláhúzandó!

[6] Megfelelő aláhúzandó!

[7] Megfelelő aláhúzandó!

[8] Megfelelő aláhúzandó!

TARTALOM

1. BEVEZETÉS:

A bevezetés során röviden összefoglalom, miről is fog majd szólni a szakdolgozatom, milyen nyelvi elemeket használok fel, megírása során milyen fontosabb állomásokra fogok kitérni. Az ötleteimet a feladatom elkészítéséhez milyen irodalmakból, fórumokból merítettem.
Egyes emberek felhasználóként böngésznek honlapokat, és vannak, akik egészen más célból, maguk is szeretnének ilyen oldalt, oldalakat készíteni. Van, hogy csak saját célra például vlogolásra, és vannak, akik haszonszerzés céljából, például megrendelésre szeretnének webshop-ot készíteni és azt üzemeltetni.

Napjainkban a számítástechnika életünk egyik elengedhetetlen eszköze. A weboldal fejlesztésére két lehetőségünk van: az egyik, hogy forráskódot készítünk. Ez programtervezői készségeket és szaktudást igényel, a másik, hogy olyan megoldást alkalmazunk, amiben már kész weboldalakat szaktudás nélkül vagy csak minimális tudással tudunk szerkeszteni, előre használható sablonok segítségével. Ennek hátránya, hogy korlátozott lehetőségeink vannak és ezek a szolgáltatások általában fizetősek. Szakdolgozatom megírása során próbáltam odafigyelni arra, hogy fokozatosan építsem azt fel, lépésről lépésre, kezdve a programnyelv kiválasztásától egészen a kinézet megtervezéséig és a továbbfejlesztési lehetőségek tárgyalásáig. A dolgozat első felében a technológiai és tervezési lehetőségekről olvashatunk, a második felében pedig a tényleges kivitelezésről és azok fázisairól példákkal. Tekintsük át a fejlesztés fontosabb lépéseit:

Alap ötlet születése: Már régóta érdekelt, hogyan is működhet egy weboldal vagy egy webshop „belülről”. Milyen kódok állhatnak a sikeres működésének hátterében, ezért választottam szakdolgozatom témájának. Mivel még életem során nem csináltam sohasem ilyet, ezért alaposan utánajártam és olvastam a témának a siker érdekében.

Első lépésként tanulmányoztam, hogy jelenleg melyek azok a webfejlesztő nyelvek, amiben egy webshop-ot meg lehet írni. A stackoverflow 2018-as kutatása alapján a leginkább használt programozási nyelvek: JavaScript, HTML, CSS, SQL, Java, Bash/Shell, Python, C#, PHP, C++. A programozási nyelv kiválasztásánál első szempont volt az, hogy az egyetemen már megismert, könnyen elsajátítható programozási nyelvet alkalmazzak, így a választásom a C#-re esett. [1]

Utánajártam, hogy C#-on belül milyen webfejlesztésre, programozásra szolgáló tool-ok vannak, itt is fontos szempont volt az, hogy a nyelv ismert legyen korábbi tanulmányokból. Alkalmazásomat nem szeretném futtatni cross-platformokon pl.: mobilon (Xamarin forms). Lényeges volt, hogy a választásom során mely nyelvekben érzem magamat komfortosan. A választásom a(z) ASP.NET -re esett. [2]

A kiválasztásnál használhatom a Core-verziót is, ami egy nyílt forráskódú .net keretrendszer, ez támogatja a jelentősebb operációs rendszereket is pl.: Windows, OS X, Linux. Alkalmazásomnál nem szeretnék több operációs rendszert is támogatni, így döntésem a tanult MVC-re esett, az ismert jól működő Razor motor-ral. [2]

A webshop elkészítéséhez nem csak egy weboldal megírása fog kelleni, hanem szükség lesz adatbázis programozási nyelvekre is. Ezekben az adatbázisokban fogjuk majd tárolni a megrendeléseket, termékeket, kiszállítás alatti termékeket. Regisztrált felhasználókat és adataikat stb. Utánanéztem az interneten, milyen nyelvek jöhetnek itt szóba: OracleSQL, HBase, MongoDB, MSSQL. Fontos volt számomra, hogy melyiket tanultuk, és melyik adatbázis nyelvben érzem magamat a leginkább otthonosan. Így a választásaim az MSSQL és az OracleSQL. Végleges választásom az MSSQL-re esett, mert legkönnyebben ez kapcsolódik a C#-hoz. Könnyű syntax-a, támogatottsága miatt. [3] [4]

Fontos ismérvnek tartom a fejlesztőkörnyezet és számítógépek bemutatását, mind azt a fejlesztőkörnyezetet, amelyen a projekt készül, és azt is, amelyen a megvalósult projekt lesz. A projektet saját számítógépemen készítettem. (Virtuális és „rendes" operációs rendszer használatával.) Virtuális gép használatához a jól bevált WMware környezetet használom. A projektet képek és leírás, prezentáció alapján szeretném bemutatni, mivel éles szerveren a futtatása költséges lenne. Saját számítógépemen a méltán híres és biztos működésű Windows 10 Professional alapú operációs rendszer fut.

A weblapom egy kitalált cég munkáját fogja segíteni. Fel kell rajta tudunk venni a rendeléseket, be kell tudnunk jelentkezni a meglévő felhasználói profilokba. Tudnunk kell rajta új profilt létrehozni. Legfontosabb: Képes termék oldallal rendelkező termékek között tudjunk válogatni, böngészni és szűrni rájuk megfelelő feltételek alapján.

Utánanéztem, hogy tényleges megvalósítás esetén mit kellene csinálni, esetleg milyen könyvek és irodalom jöhetnek szóba, amiben már készítettek webshop-ot ASP.NET MVC-ben. Rátaláltam egy írásra/könyvre, amiben az író részletezi, ő hogyan is képzelt el egy ilyen weboldalt. Úgy gondolom, a szakdolgozatom megírásához ez egy remek támpontot fog adni. [5]

A weblapon több termék kép is található lesz. Ezeket képszerkesztő, vágó programokkal fogom készíteni. (pl.: Photoshop, Gyazo, Paint) Ezek a programok a késztermék kinézetében fognak szerepet játszani és a felhasználói élmény fokozásában. Interaktív elemeket is tartalmaz majd a weboldal.

A weblap esetleges továbbfejlesztési lehetőségeiről is gondolkodni kell: Itt fontos, hogy a rendszerünket milyen újabb technológiával lehetne még jobbá és felhasználó baráttá tenni. Új technológiák bemutatása, felkészülés a továbbfejlesztésére, esetleges későbbi hibák javítására. Microsoft új eszközei, a kód felkészítése erre:

Olyan weboldal bemutatása, ami ügyfélorientált és felhasználóbarát. Lehetőségek mérlegelése egy jobb weboldal megvalósításához. Fejlesztési lehetőségek, optimalizálhatóság áttekintése kliens és szerver oldalon. A cég munkáját segítő vállalatirányítási rendszer áttekintése. A programkód tesztelési lehetőségeinek kivitelezése. A projektet be is szeretném mutatni. Ennek a működéséről és a végleges kinézetről, végleges funkcionalitásokról a dolgozat II. félében írok, továbbá ppt segítségével meg fogom mutatni élesben.

2. PROGRAMNYELVEK BEMUTATÁSA:

2.1. C#:

Programnyelv kiválasztásánál nagyon fontos szempont volt, hogy olyan webfejlesztésre szánt nyelvet válasszak, amit már egyetemi tanulmányaink során, magas szinten elsajátítottunk. Választásom a bevezetésben felsorolt nyelvek közül a C#-ra esett. Ez az objektum orientált programozási nyelv (OOP) széles körben elismert, így sok információt találhatunk róla az interneten könyvekben, egyetemi jegyzetekben.

A programnyelv kiválasztása után megkezdődhet a tervezés: Itt külön választom a frontend, azaz webshop megjelenítésére szánt részeket és a backend, a webshop működtetését végző részeket. Most foglalkozzunk a backend-el.

A feladatot egészben egy kódként lekódolni szinte lehetetlen, ez spagetti-átláthatatlan kódot eredményezne, amelyen még a készítője is nehezen igazodik el, ehhez a kódot elemi részekre és megfelelő logikai egységekre kell tagolni, hogy minden csak a saját, neki szánt feladatát láthassa el. Tehát legyen egy réteg, ami kezeli az adatbázist, ellenőrzi az adatokat, kezeli azokat stb.

Layering alkalmazása: Feladata: Elkülöníteni az egybe tartozó részegységeket. Így alábbi részegységek lehetnek:

-Repository: Kezeli az adatbázist, megvalósítja a CRUD-metódusokat (Create, Read, Update, Delete, GetAll), ezekkel a metódusokkal valósítja meg a C# az MSSQL adatbázis kezelést, folyamatosan menti azt, ha változás áll fenn. A metódusok feladatai:

- Create: Elkészít egy specifikus adatbázis elemet.
- Read: Kiolvas egy specifikus sort az adatbázisból pl.: ID, e-mail, típus stb. alapján.
- Update: Create és Delete meghívása. Régi törlése, helyére új kerül.
- Delete: Kitöröl egy specifikus sort.
- GetAll: Minden elem listázása egy táblából IQuarieble-be.

-Model: Projekt elemeit alkotó alapvető részegységek „gyűjtőhelye”. Konfigurációs részek ide kerülnek statikus osztályként. Projekt alap elemei is itt lehetnek, mint például az adatbázis elemeit alkotó részegységek. Feladat megoldása során ezt a réteget a Repository (alsó) layer fogja megvalósítani és esetlegesen ASP.NET-es környezetben AutoMapper segítségével az MVC Model is használhatja.

-Logic: Itt kerülnek olyan elemek a kódba, amelyek elvégzik, hogy az adatbázisba ne kerülhessenek olyan elemek, amik nem oda valók. Tehát: Ha egy számérték helyére a felhasználó betűt ír, akkor az ne juthasson le az adatbázisig. Feladata ezeknek a hibáknak az elhárítása, és ha ezt nem tudja, jelzi ezt a felsőbb rétegek felé. Felsőbb projekt olyan metódusait is megvalósítja, amik már működését tekintve inkább hozzá tartozik. Feladata

a felhasználókezelés és az adatbázis olyan új elemeinek előállítása, amelyek már beszúrásra kerülhetnek az adatbázisba. Kezeli a Repository CRUD metódusait és csak olyan esetekben, hogy az adatbázisba csak valós elemek kerülhessenek. Továbbá az alsóbb rétegekből keletkező exception-ok (hibák, kivételek) kezelése.

Az egyes rétegek DLL (Dynamic Link Library)- ben tárolódnak. Ez lehetővé teszi a rétegek könnyű egymáshoz való csatolását. [12]

Minden egyes ilyen réteg (layer) megvalósít egy interface-et. Az interface-ek tulajdonsága az, hogy olyan elemeket tartalmaznak, amiket egy osztálynak kötelezően meg kell valósítania.
Így Repository megvalósít egy IRepository, Logic egy ILogic interface-t:
Tartalmuk lehet: (T, K bármely típus: Termék, Megrendelés, stb.)

IRepository:

- Create(T t)
- T Read(K k)
- Update(K k)

ILogic: Többet is megvalósíthat pl.: IManage felhasználó kezeléshez és IManageTables a projekt egyéb elemeit tartalmazza: Interfacektől függetlenül pár példa ezekre:

- ManageOrders ();
- ManageQuestions ();
- ModifyProduct (); stb.

Model: Nem tartalmaz interface-t, mivel ő csak alap osztályokat határoz meg.
Dependenci Injection: Egy olyan technológia a számítógép programozásban, amelynek lényege, hogy egy objektum egy másik objektum függőségét elégíti ki. A függőséget felhasználó objektum, szolgáltatást nyújt, az injekció a függőséget adja át a kliensnek. Kliens állapotának része a szolgáltatás. Ennek a fejlesztési mintának az alapkövetelménye, a szolgáltatás kliensnek való átadása helyett, hogy a kliens objektum a szolgáltató objektumot hozza létre. [6] [7]
Így a különböző rétegeket az interfacek felhasználásával kapcsoljuk össze, ezek biztosítják a kapcsolatot. A frontend és backend rétegek kapcsolatait megtekinthetjük az 1. ábrán.
MSSQL Repository réteghez való kapcsolása:
A C#-os legalsó Repository réteghez hozzá kell kapcsolni az adatbázist, ehhez kis utánanézésre volt szükségem, amit gyorsan meg is találtam a Microsoft hivatalos honlapján. Egy úgy nevezett ADO.NET hivatkozást kell használjunk, ezzel tudjuk majd kezelni az adatbázist a C#-os fejlesztő környezettel. [8]
Felhasznált nyelvi elemek: Tanulmányaink során a C# programozást és a nyelvi elemeket és ezek megfelelő felhasználását a sok beadandó, zárthelyi és féléves feladatokon keresztül el tudtuk sajátítani, de az interneten is sok segédanyag megtalálható. [9]

2.2. ASP.NET: Active Server Pages.Net

Ez egy olyan fejlesztési platform, aminek a segítségével a .net környezetben tudunk gyors web/más alkalmazásokat készíteni, akár cross platformra. Ezt a programozási rendszert már tanulmányaink során is használtuk webfejlesztésre, és érdekel is a használata, így célszerű volt a fejlesztésre ezt alkalmazni. Webáruházam nem készül cross platformra alkalmazás szinten (IOS, Android - Xamarin forms), így csak Windows-os (asztali gépes) környezetre készítem, ezen is szeretném bemutatni. A webshop frontend megjelenése viszont támogatja a telefonos, tabletes nézet megjelenítését is (responsive). [2]

A programnyelv bemutatása során a backend részekre összpontosítottam, most térjünk át a frontendre és a frontend rétegeire: Ezek a rétegek a Logic feletti részek. Az MVC-ről későbbiekben lesz szó. Tekintsük át a logikai felépítést:

-API Controller: Szerver-kliens modellen alapul, a kliensek kéréseket intéznek, a szerver felé majd az válaszol nekik. Egységes interfacet biztosít, amin keresztül elérhető a rétegzett architektúra. HttpGet és HttpPost metódusokból áll. [2]

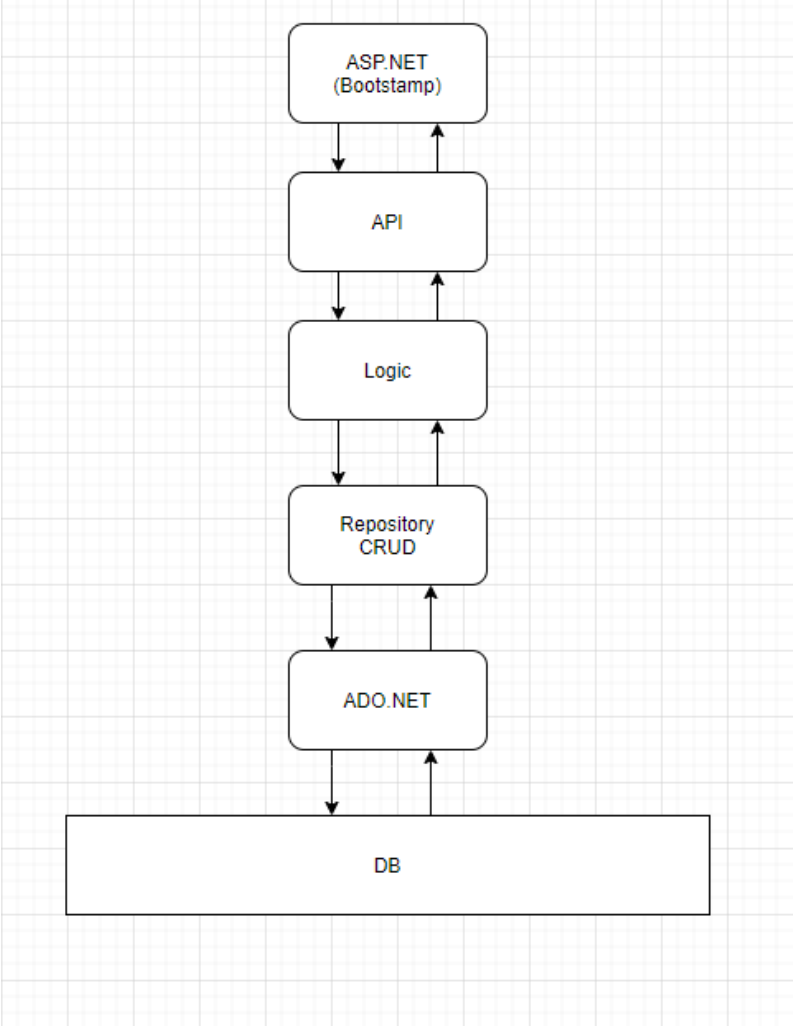
- HttpGet: Adatokat ad vissza a lentebbi rétegekből. Például visszaadja az összes olyan terméket, ami megfelel a szűrésnek.
- HttpPost: Adatokat küld a lentebbi rétegekbe. Például új felhasználót létrehoz, bevisz egy rendelést.

-Views: Azaz a kinézet: Webshopom tényleges kinézete itt dől el, ez egy weblap kinézet tervezés, ami HTML-ben íródik. Itt határozom meg azt, hogy melyik elem hol van, mik jelennek meg és hogyan a weblapon, az oldalnavigáció itt valósul meg. A tényleges tartalmat pedig az API segítségével lentebbi rétegekből szedjük. Csak az adatok megjelenítésével foglalkozunk.

Bootstrap: A weblap kinézetét HTML, CSS, Bootstrap, és JavaScript nyelvekkel fogom elkészíteni. Ezek a manapság használt legismertebb frontend fejlesztésre szolgáló nyelvek. [2] [10]

- HTML (HyperText Markup Language): Weblapok készítéséhez kifejlesztett leíró nyelv, ami mára már internetes szabvánnyá vált. A HTML 5 változatát fogom használni, mivel ez a jelenlegi legfrissebb. [11]
- CSS (Cascading Style Sheets): Stílusokat leíró nyelv, amely segítségével HTML alapú oldalak megjelenítését, stílusát írjuk le. [11]
- Bootstrap: Előre beállított sémák segítségével tudunk HTML oldalakat CSS és JavaScript beállításokkal ellátni. Célja a mobil alkalmazások tervezése.
- JavaScript: Weboldalakon elterjedt prototípus alapú objektum orientált scriptnyelv. Weboldalak frontend viselkedését tudjuk vele szabályozni.

Projekt rétegzett felépítését megtekinthetjük az 1. ábrán.



1. ábra: Layering

Az ábra magyarázata: Láthatjuk a projekt elvi felépítését, a rétegek és azok kapcsolatát, a backend és frontend elkülönülését. Láthatók az alsóbb rétegek adatbázis és Repository elhelyezkedése. A rétegek közti kommunikációt a nyilak jelképezik. Jól láthatóan a rétegek elkülönülnek egymástól, rétegsértés nem következik be. Ez azt jelentené, hogy egy réteg a mellette lévő réteg kihagyásával kommunikál egy másikkal. A kommunikáció alapértelmezetten fentről lefelé irányul, de lehetőség van fordított kommunikációra is események segítségével különleges esetekben. Alsóbb rétegekben keletkező kivételeket a felsőbb rétegek kezelik (Logic). Ezzel célunk, hogy a felhasználó egy esetlegesen képződő hibából ne érzékeljen semmit, az oldal futása hibátlan legyen.
Lehetségesen fellépő hibák: Nem található a kért adatbázis elem, ennek oka lehet, hogy rossz id-re hivatkozunk, és null hibába futunk, ami akár a kód azonnali megakadását is jelentheti, de kivételkezelés alkalmazásával az oldal futásában például egy hibaüzenetet írunk ki.

2.3. ASP.NET, .Framework, .Core, MVC, fejlesztési minták:

ASP.NET webfejlesztés során lehetőségünk van használni a .Core és .Framework változatot. Tekintsük át a különbségeket:

- .Core: Nyílt forráskód, folyamatosan újul, Cross-platform támogatása, elhivatottság kell hozzá, nem félni az újtól.
- .Framework: Stabil környezet, meglévő alkalmazás fejlesztésekor, gyorsan elsajátítható programnyelv. Válasszuk, ha már van meglévő tapasztalatunk benne. Nem való folyamatos frissítésekre és változtatásokra. Felhasználóknak szánt alkalmazások: például WPF (ablakos alkalmazás) támogatás.

ASP.NET MVC View Engine: Eredetileg WebForms-ra épült, később új motorok felhasználásával kiegészítették. (NDJango, Brail) Leggyakoribb és legstabilabb verzió a Razor View Engine. [2]

A projekt elkészítéséhez tanulmányoztam, milyen fejlesztési minták vannak, amiket felhasználhatok. Kezdjük a legfontosabbakkal, ezek befolyásolják legjobban a késztermék működését:

-MVC: Model-View-Controller: Összetett, sok adatot felhasználó és eltároló alkalmazásokhoz használják, ahol gyakori, hogy el kell választani az adatokhoz tartozó Model és a felhasználókhoz szánt nézetet (View). Ezt a folyamatot vezérli a Controller.

Részegységei röviden leírva: Model: Felel az adatok kezeléséért. (Adatbázis kapcsolatokat kezel, tárol) View: Megjeleníti a felhasználónak az adatokat. Ugyanahhoz a modellhez más-más nézet is kapcsolódhat. Controller: Felhasználói műveleteket dolgoz fel, és válaszol ezekre.

Működése: A felhasználó például megnyom egy gombot – a vezérlő átveszi az eseményt – kapcsolat a modellel, kezeli azt – a nézet megjeleníti a végeredményt – újabb esemény várása a felhasználói felülettől.

MVC fejlesztési minta hátránya a nehezen áttekinthető kód, ezt csak hozzáértők tudják elemezni és dolgozni vele. Nagyon elterjedt a webfejlesztők között, egész Framework is épült rá. [12]

-Factory method: A gyártó objektum egy olyan objektum az objektum orientált programozásban, ami más objektumokat képes létrehozni paramétere(i) függvényében.

Programtervezési minta megjelenése a feladatban: AutoMapper használata, összetett bemeneti objektumot alakítunk át megfelelő eljárások segítségével más egyszerűbb objektummá. [12] [13] [7]

API Controller is az MVC-s projektben található meg a controllerek között. Innen kezeljük az AutoMapper és a ViewModel bizonyos részeit is. A Get és Post metódusain kívül áll még Helper metódusokból is, amelyek a mapper kezeléséhez kellenek.

3. ADATBÁZISOK ÉS FEJLESZTŐKÖRNYEZETEK

3.1. MSSQL:

Előzőekben foglalkoztunk a frontend részekkel, most térjünk át a backend legalsó rétegére, az adatbázisra. Választott adatbázis rendszerem a C#-hoz legkönnyebben kapcsolható MSSQL. Ez egy jól bevált és általam tanult adatbázis nyelv. Ezt a réteget kezelik a CRUD metódusok.

Néhány fontosabb fogalom:

-Adatbázis: Azonos minőségű strukturált adatok összessége, amelyet tárolásra, lekérdezésre alkalmas szoftver kezel.

-Adatmodell: Adatbázis szerkezeti és működési leírásának a neve.

-Strukturált adatbázis: Vannak azonos és változó attribútumszámú adatbázisok. A változó attribútumszámú adatbázisokban bizonyos információk jóval tömörebben és egyszerűbben fejezhetők ki, mint rögzített attribútumszám esetében. [8]

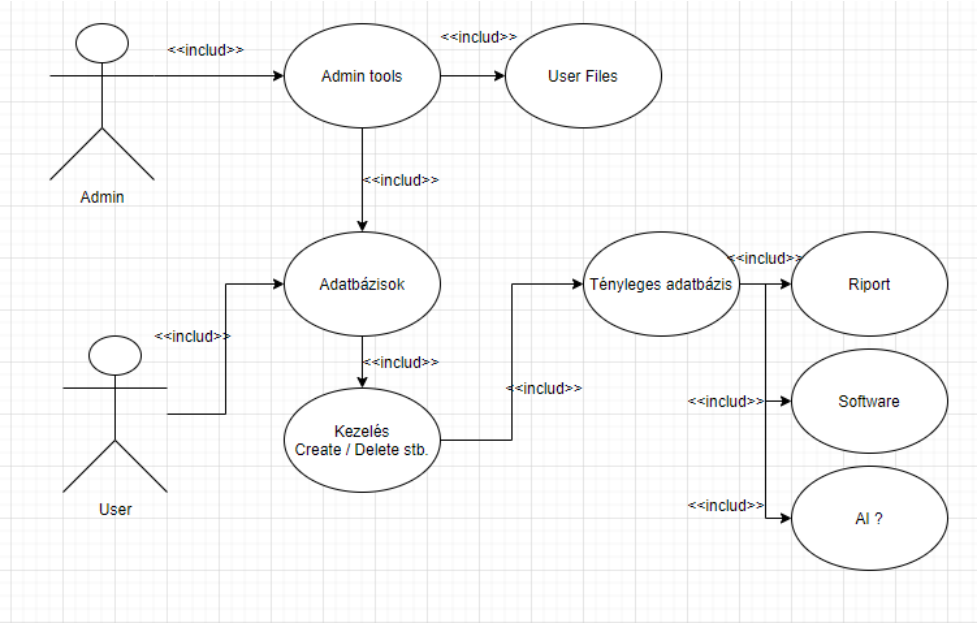
-Relációs adatbázis: Azt nevezzük relációs adatbázisnak, amely relációs adatmodell sajátosságai alapján hoznak létre adatokat. Meghatározunk ún. relációk egy véges halmazát. Ezeket az adatbázisokat relációsadatbázis-kezelőkkel tudunk létre hozni, szerkeszteni vagy törölni.

-Elsődleges kulcs: Meg kell kötelezően jelölni. Ez a kulcs minden rekordban egyedi.

-Idegen kulcs: Tábla elsődleges kulcsára mutató kulcsok. Egy elsődleges kulcsra több idegenkulcs is mutathat. [14]

Adatbázis nyelvek közül egy olyat kellett választanom, amiben már a félévek során a használatában tapasztalatot szereztünk. Az MSSQL-ben lehetőség van a változó adatkezelésre. Fejlesztőkörnyezetét (Microsoft SQL Server Management Studio) könnyű kezelni és sok ismertető olvasható róla az interneten.

Use case diagramm: Adatbázis egy lehetséges felhasználási módja, akár további felhasználás céljából. Adatbázis rendszer megtervezése előtt fontosnak tartottam részletezni, hogy az adatbázisomat mire is tudom felhasználni: Átgondolás után, Ábra 2 -n bemutatott use case mutatkozott helyesnek: Különféle felhasználói jogkörök vannak (Admin és User) Admin tudja kezelni a felhasználók adatait, tudja szerkeszteni az adatbázist, tud kimutatást, szoftveres eszközt esetleges további felhasználásra gondolva mesterséges intelligenciát felhasználni az adatok elemzéshez. Normál felhasználó csak kezelni tudja, esetlegesen kimutatást, szoftvert tud alkalmazni, AI-t kezelni, de nem tud hozzáférni a többi felhasználó személyes adataihoz. (Ábrán szereplő nyilak a kommunikációt jelképezik.) [15]



2. ábra: Use Case diagramm

Ezek után megterveztem, milyen főbb tábláim lesznek a feladat elkészítése folyamán, megterveztem az adatmodellt. A megoldás során ez még kiegészülhet újabb táblákkal is. Feladatkörönként elkülönítve:

- Profiles: Minden létező felhasználó adatait leíró tábla.

- AvaliableProducts: Minden elérhető termék.
- Products: Termék osztályok.
- Main Products: Fő termék osztályok.

- Feeds: Felhasználóktól termékekhez fűzött értékelések (1-5).
- Questions: Felhasználóktól termékekhez fűzött kérdések.
- Comments: Felhasználóktól termékekhez fűzött kommentek.

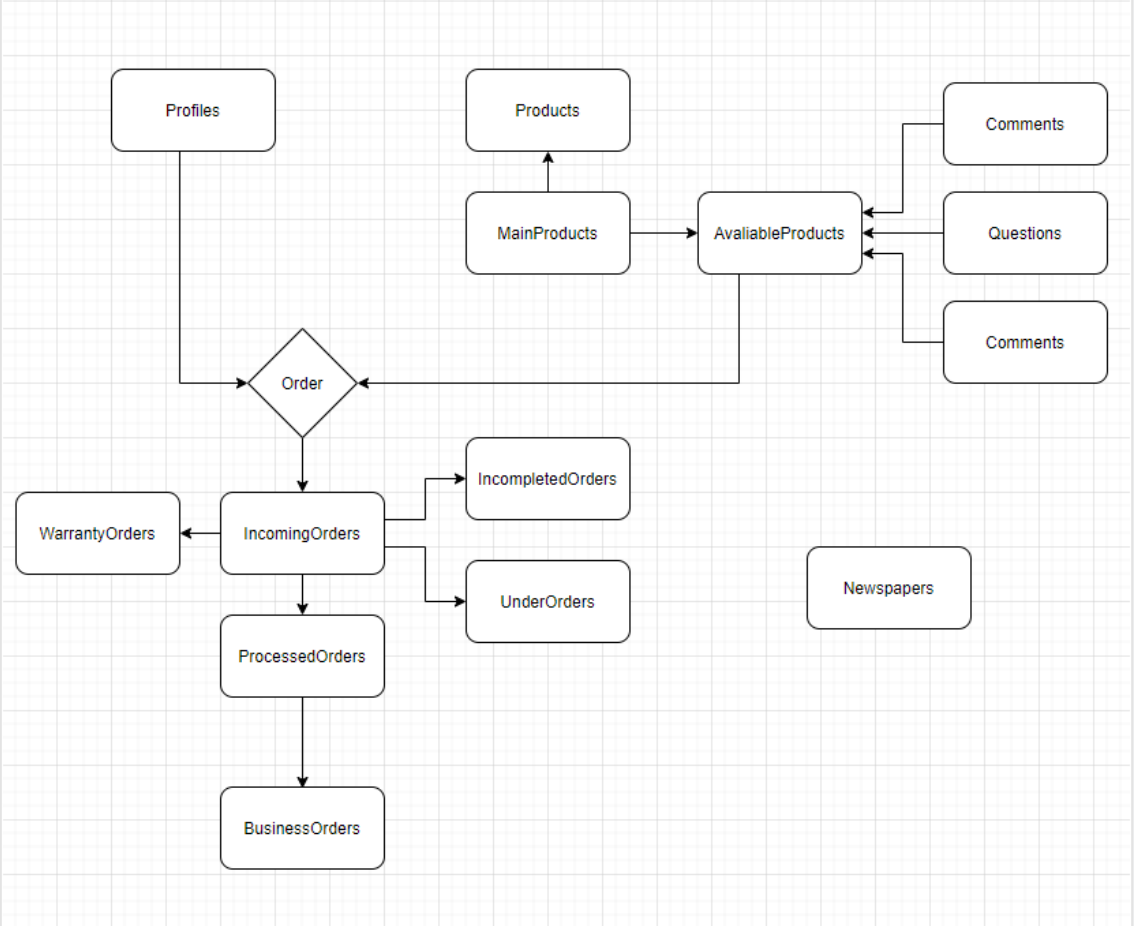
- Newspapers: Főoldalon megjelenő hírek.

- IncomingOrders: Beérkező megrendelések.
- IncompletedOrders: Nem teljesíthető beérkező megrendelések.
- UnderOrders: Beszállítás alatt álló termékek.
- WarrantyOrders: Garanciális megrendelések.
- BusinessOrders: Céges bevételek és kiadások.

- PromotionCodes: Megrendeléshez fűzött promóciós kódok.
- Egyéb fő táblákat segítő táblák és kapcsolótáblák.

A táblák leírásánál fontosnak tartottam, hogy csoportok és funkcionalitások szerint szét legyenek szedve. Tehát ami egy csoport pl.: Orders-ekkel foglalkozó táblák, azok egymás alá, illetve elkülönülve vannak a többitől. Így eljárva a többi részegységgel is.

A táblák egymáshoz való viszonyát, elsődleges és idegen kulcsainak kapcsolatát szemléltetem a 3. ábrán:



3. ábra: Táblák és kapcsolatok

A 3. ábra magyarázata: Adatbázis rendszer képes kezelni egy felhasználóhoz tartozó elérhető termékből képződő megrendelést. Termékek megfelelő rendszerezéséből (Raktáron van-e?, marketing, fő- és altermék osztályokból, megjegyzések, kommentek, értékelések alapján) tud a felhasználó kiválasztani egy terméket, majd abból egy megrendelést csinálni, amelyek eltérhetnek szállítás, garanciális szempontból. Ebből képződnek a céges megrendelések, bevételek, kiadások, amikből akár céges működés bemutatására kimutatások is készülhetnek. Táblák egymáshoz való kapcsolódását elsődleges és idegen kulcsok biztosítják. Order egy kapcsoló tábla, kapcsolatot biztosít a Profilok, Product-ok és Orders-ek között azok kulcsainak felhasználásával.

A 1. táblázatban jelenítettem meg a táblákat és mezőiket. A tábla nevek és a mezőinek nevei is angol elnevezésűek, mivel az angol egy nemzetközileg is sokak által ismert nyelv, sok ember számára érthető. A programozók leggyakrabban az alkalmazásokban ezt a nyelvet választják. Manapság már sok cég megköveteli, hogy a kódot angolul írjuk és adjuk be feléjük.

A táblák létrehozásakor egy közös pont, hogy minden tábla egy adott sorának elsődleges azonosításához egy ID mezőt használok, így azonosítva egyedileg a tartalmat. Próbáltam a megtervezéskor ügyelni arra, hogy az adott tábla a funkcionalitásának eleget tegyen, de ne használjon túl sok mezőt sem. A táblák egymás között logikai kapcsolatokat is megvalósítanak. A feladat végleges megvalósításakor ezek a mezők változhatnak.

Tábla neve	**Mezők**
Profiles	ID, FirstName, LastName, Username, Email, Pw, Registered
AvaliableProducts	ID, Name, IndexImg, PrimImg, SecImg, MainProductID, ProductID, FeedsID, QuestionsID, CommentsID, Price, Description, Spec Description, GaranteeTime, Stock
Products	ID, MainProductID, Name
Main Products	ID, Name
Feeds	ID, Rate, Profile, ProductID
Questions	ID, Text, Profile, ProductID, Text
Comments	ID, Profile, ProductID, Text
Newspapers	ID, Title, Subtitle, Description, ImgPath
IncomingOrders	ID, ProfileID, ProductID, Country, BilligAddress, BilligEmail, BilligTel, CourierState, ShippingAddress, ShippingEmail, ShippingTel, ShippingMode, Payment, Text, PromotionCode, Amount
IncompletedOrders	ID, OrderID, EstimetedTime
UnderOrders	ID, OrderID
WarrantyOrders	ID, OrderID, BeginDate, Time
BusinessOrders	ID, OrderID
PromotionCodes	ID, Price

1. táblázat

A Microsoft SQL Server Managment Studio az adatbázist localoson fogja létrehozni és ezt a számítógépemen fogja tárolni. A weboldal végleges üzembe helyezéséhez egy éles adatbázis szerverre van szükség (nem localhost), hogy az adatbázisokat ne csak az én gépemről lehessen elérni.

3.2. Fejlesztőkörnyezetek bemutatása:

A projekt tervezési fázisába fontosnak tartom részletezni a fejlesztő környezeteket és ezek választásának szempontjait, mind hardveriális mind szoftveriális téren, kliens és szerver oldalon egyaránt.

Elsősorban kezdjünk a C# programnyelv fejlesztőkörnyezetével. Itt kettő lehetőség merült fel:

- Visual Studio Code: Egy olyan ingyenes kódszerkesztő, amit a Microsoft kínál a C# programozáshoz Windows-os és Linux-os rendszerekhez egyaránt.
- Visual Studio 2019: Hátránya, hogy fizetős, de sok programozási nyelvhez, köztük webfejlesztéshez is egyaránt való, régóta támogatott fejlesztő környezet.

A kiválasztásnál fontos szempontot játszott, hogy az adott fejlesztőkörnyezet már ismert és tanult legyen, tartalmazzon intelligens kódkiegészítést is (IntelliSense). Így a fejlesztő környezetnek a Visual Studio 2019-et választottam.

A megfelelő működés eléréséhez egy olyan kódot kell, hogy alkossunk, amihez a felhasználó a bemenetre akármilyen adatot is ad meg, a kód helyesen és hiba nélkül kell, hogy fusson. Például: Szám érték helyére betűket stb. Ha ilyen kivételeket a kódban nem kezelünk, akkor az működésképtelen lesz és hibákat fog eredményezni. Ezért a kódot tesztelni kell a szélsőséges esetekre is, hogy az megfelelően tud-e működni. A választott fejlesztő környezetem erre kínált lehetőségei az alábbiak:

- NUnit: Egy nyílt forráskódú teszt Framework .Net-es rendszerekre, amit a fejlesztőkörnyezet alapértelmezetten támogat.
- Moq: Olyan objektumokat hoz létre, amelyek utánozzák teszt adatokkal a valódi objektumokat és azok viselkedéseit. Ezen objektumok létrehozásával tudjuk tesztelni az eredeti programunk viselkedését szélsőséges esetekben.

Utánanéztem, hogy a választott adatbázis nyelvhez (MSSQL) milyen adatbázis fejlesztő környezetek vannak. Itt egy lehetséges megoldást találtam:

- SQL Server Management Studio 2019: Lehetővé teszi a felhasználó számára, hogy a kiszolgáltatón található objektumokat böngéssze, kijelölje és dolgozhasson azokkal. Az SQL szerver összes összetevőjének a konfigurálására és kezelésére szolgál.

A projektet lehetőségem volt a saját operációs rendszeremen készíteni, de itt már elég sok alkalmazás fel volt telepítve, amelyek lehet zavarhatták volna a projekt megírását. Így egy Windows 10 alapú virtuális gépet hoztam létre a projekt fejlesztéséhez, mivel a legtöbb fejlesztő program is ezt támogatja. Az operációs rendszer frissen telepített, ezért itt már nem lehet akadályozó tényező a projekt megvalósításához. A gépet lehetőségem volt a VMware Workstation Pro-n és az Oracle VM VirtualBox segítségével futtatni.

A VMware egy újabb, jobban beállítható és tanult alkalmazás volt, így célszerűbb volt ezt a kezelőt választani.

Verziókövető rendszer: „Fejlesztés alatt álló dokumentumok, tervek, forráskódok és egyéb olyan adatok verzióinak kezelésére, amelyeken több ember dolgozik egyidejűleg. Az egyes változtatásokat verziószámokkal vagy verzióbetűkkel követik nyomon.” [7]

A projektet szerettem volna későbbi felhasználáshoz felhőbe is elmenteni, ehhez létrehoztam egy privát verziókövető rendszert (Repository-t).

> GIT: „Egy nyílt forráskódú elosztott szoftverforráskód-kezelő rendszer, ami a sebességre helyezi a hangsúlyt.” [7]
>
> Branches: Fejlesztési ágak, lehet többféle is fejlesztéstől függően: Hotfix, Release, Master, ezeket össze is lehet ’kötni’.
>
> Sourcetree: Egy felhasználói interfész (GUI), leegyszerűsített GIT, hogy a felhasználó csak a kódolásra tudjon fókuszálni. Megkönnyíti a kapcsolattartást a verziókövető rendszer és a felhasználó között.
>
> Commit: Leírja, hogy mit tartalmaz egy adott változás. Üzenettel, számozással.

A feladathoz csak egy Mester branche-t hoztam létre a Sourcetree felhasználásával a népszerű és ingyenes bitbucket.org -on. Ide kerülnek a commitok és a projekt.

A virtuális gép futtatásához kell egy elég erős számítógép is, ami ezeket a rendszereket képes elfuttatni. Saját számítógépemben egy i7-9700K, 32gb RAM, és GTX1080 8GB van. A virtuális gépek memória és processzor igényesek, ezért megfelelő és gyors alapot nyújt saját gépem a fejlesztéshez és a weboldal végső teszteléséhez. Operációs rendszerek közül a legnépszerűbbek erre a számítógépre:

- Windows: Legtöbb alkalmazás támogatja, könnyen kezelhető magas támogatottságú, de sajnos fizetős rendszer.
- Linux: Nyílt forráskódú, biztonságos, alacsony méretű, támogatott, de sajnos vannak gyakran használt alkalmazások, amelyek nem támogatják (Office).
- Ubuntu: Megbízhatóság, jó megjelenés, de sajnos alkalmazások többsége nem támogatja.

Saját számítógépemen a már jól ismert Windows 10 kigyakorolt és biztos működésű operációs rendszer található. A legtöbb program ezen képes a legjobban működni és ez a dokumentáláshoz elengedhetetlen. Így célszerű ezt az operációs rendszert használni a virtuális gép futtatásához és a feladat elvégzéséhez.

A weblapot illetve a mögöttes kódokat valahol futtatni is kell, erre egy normál felhasználásra szánt számítógép nem megfelelő, mivel az nem alkalmas arra, hogy azt éjjel - nappal futtassák, így hamar tönkre menne. Boltban vásárolni ilyen számítógépet túl drága lenne, ezért valamilyen felhőszolgáltatásba kellett gondolkodnom.

A szerver gép egy olyan számítógép, ami a különféle felhasználók számára elérhetővé teszi erőforrásait, más szolgáltatásait és ezek kihasználását. Ez a számítógép alkalmas arra, hogy az alkalmazást folyamatosan futtassa és elérhetővé tegye a felhasználók számára. [7]

Felhőszolgáltatások: A szolgáltatást nem egy magadott hardveren üzemeltetik, hanem a szolgáltató elosztott rendszerein (szerverek), annak üzemeltetési részleteit a felhasználó elől elrejtve. Felhasználók ezt a szolgáltatást a hálózaton keresztül érhetik el. [7]

Keresgélni kezdtem a számomra megfelelő szolgáltatást. Fő szempont volt annak ára és a könnyű felhasználási lehetőség is. A legnépszerűbb felhő (cloud) szolgáltatások:

- Microsoft Azure: Egy olyan Microsoft (MS) által felügyelt cloud platform melynek segítségével alkalmazásokat lehet készíteni, telepíteni és futtatni az ő általuk felügyelt adatközpontokon. Körülbelül 600 féle különféle szolgáltatást tudnak biztosítani. [16] [7]
- Google Cloud Platform: Google által felügyelt felhőszolgáltatás, ahol szintén alkalmazásokat és számításokat tudunk végezni a felhő segítségével. [17] [7]
- Egyéb szolgáltató(k): Termékük lehet drága, és használat szempontjából körülményes.

Összegzés: Célszerű olyan szolgáltatást választani, amelyek megbízhatóan alkalmazhatók erre a jellegű munkára, interneten, fórumokon sok oktató, valamint ismeretterjesztő anyag megtalálható hozzájuk. Kiválasztásnál lényeges szempontot játszik ezen felül, hogy a tervezést, kivitelezését mind szoftveriális, ezen kívül hardveriális oldalon is MS szoftver termékekkel készítettem, továbbá így célszerű az ő terméküket választani erre a célra is. A Google esetén az árazás drága még rövid idő-intervallumra is. Visual Studio támogatja az Azure szolgáltatásba való projekt feltöltést, így célszerű a választása, abban az esetben, ha a webshopot éles szerverre is szeretnénk elhelyezni.

Hosszas gondolkozás után arra a következtetésre jutottam, hogy a kész weboldalat nem fogom éles szerveren is beüzemelni, mivel ez dinamikus web tárhelyet követelne, ezek a szolgáltatások képesek futtatni a .net keretrendszert, így üzemeltetve a weboldalat. Ezeknek a szolgáltatásoknak az ára jelentős mértékű, amit nem szeretnék kifizetni, így a webshopot bemutatni a dolgozat II. félében DEMO jelleggel, képekkel és lírással, valamint a ppt segítségével fogok. Otthoni gépemen Apache webserver segítségével lehetne a webshopot hostolni .netet futtató környezeten, ez egy jó megoldásnak ígérkezett a fennálló probléma megoldására. Azonban a nálam lévő szolgáltatói modemben a weboldalak eléréhez szükséges 80-as és az összes többi port is a szolgáltató által alapértelmezetten és módosíthatatlanul tiltva van, így az otthoni hostolást sem tudom kivitelezni.

4. RENDSZERTERVEZÉS

4.1. Funkcionalitások:

Webáruházakról általánosan: Olyan weblap, amely termékeket, szolgáltatásokat értékesít. Az emberek számára már mindenhol elérhetővé vált az internet szolgáltatása, így a kereskedők számára megnyílt egy új platform az értékesítésre. A kereskedők az e-kereskedelem terjedésével észrevették, hogy ez új üzleti lehetőségeket rejt. Egy jól működő webáruháznak felhasználó barátnak és egyszerűnek kell lennie. Fontos szempont, hogy a felhasználók a lehető legkönnyebben el tudják érni (keresőmotor barát) és közvetlenül kapcsolatot tudjanak vele tartani. [18] [7]

Előnyei: Amikor a felhasználó egy online felületet választ egy normál bolt helyett, ott egy bevásárló kosár segítségével tud megvenni termékeket. A bolt célja, hogy a termékeit eladja, a felhasználó a termékek bő választéka közül válogathat, igényeinek megfelelően, és azt akár lakhelyétől több ezer kilométerre lévő boltban is megteheti. Így olyan termékeket is vásárolhat, amelyeket más úton nem tudott volna elérni.

Hátrányai: A felhasználók egy most induló webáruházat nehezen fogják tudni megtalálni, így azt hirdetni kell, amelyek húzós költségekkel járhatnak. A webshopot szerveren kell futtatni, azt üzemeltetni kell, az alkalmazottak fizetése is nagy költségekkel jár, amit a webshop bevételeinek fedeznie kell.

Követelményspecifikáció:

Szakdolgozatomban egy kitalált cég igényeire fogok készíteni egy webshopot, ez a cég egy informatikával foglalkozó cég, amely számítástechnikai alkatrészek széles választékával foglalkozik, alkalmazottjai is vannak. Eddig a cég csak helyi terjesztéssel foglalkozott, de most szeretnének online megrendeléseket is fogadni, ezeket kezelni és eleget tenni a rendeléseknek. Megkértek, hogy készítsek nekik egy webshopot, amit a munkájukhoz tudnak használni.

Ügyféli igény:

- A felhasználó a weboldal menürendszerének segítségével tudja megkeresni a különféle termékeket. Legyenek főmenük (mint például: Notebookok, alkatrészek, perifériák, tartozékok, mobiltelefonok TV-k, hálózati eszközök). Legyen egy központi kereső is.
- Az adott termékek között lehessen szűrési feltételt állítani. (pl.: alaplapoknál LGA1151-200 és LGA1151-300, Socket-AM4 alaplapok szűrését állítani, attól függően milyenekre szeretnénk szűrni AMD vagy Intel, stb.)
- A termékek először szűretlenül jelenjenek meg, majd szűrés után csak az adott feltételnek megfelelően tudjunk közülük választani. Képpel, árral, megnevezéssel ellátott hirdetések jelenjenek itt meg.
- A hirdetésre rákattintva megjelenjen a termék adatlapja, leírása, ára, raktárkészlet, kommentek, a kosár gomb, amivel megrendelhetjük.
- Lehetőség legyen a kosárba egy vagy több terméket is helyezni.
- A kosár alján lévő fizetés gomb segítségével tudjuk megrendelni a termékeket, ez után tudjuk a házhozszállítást, személyes átvételt, postát kérni. Szállítási címet meg tudjuk adni, és azt, van- e valamilyen további megjegyzésünk a vásárláshoz.

- A megrendelés folyamatát tudjuk nyomon követni. Raktározás állapota, szállítás folyamata.
- Tudjuk nyomon követni a cég bevételeit, kiadásait.
- A weboldal ügyintézői, kezelői számára legyen lehetőség a megrendelések, módosítására.
- Felhasználók tudjanak bejelentkezni, regisztrálni, fiókjaik segítségével megrendelni.
- A weboldal feleljen meg a jelenlegi kor elvárásainak, legyen designje.

A weboldalon megjelenő adatokat a feladat megoldása során véletlenül generált adatokkal fogom feltölteni, így szemléltetve egy esetleges éles használatot. (Megjelenő minden adat fiktív és véletlenszerű.)

Az ügyféli igények a feladat megoldása közben még változhatnak, ez függ attól, hogy az ügyfél nem tudja, pontosan mit akar/akarhat, így változások következhetnek be. (Agilis hozzáállást igényel.) [19]

Az online áruház sikeres működése érdekében, megrendelőket (felhasználókat) kell találni. Ez már lehet egy meglévő ügyfélkörből vagy új ügyfélkör oldalra történő navigálásából. Ezt megtehetjük fizetett reklámok segítségével pl.: Facebook, Instagram ; megbízunk vele külső marketing szolgáltatókat vagy mi magunk keresünk új ügyfeleket.

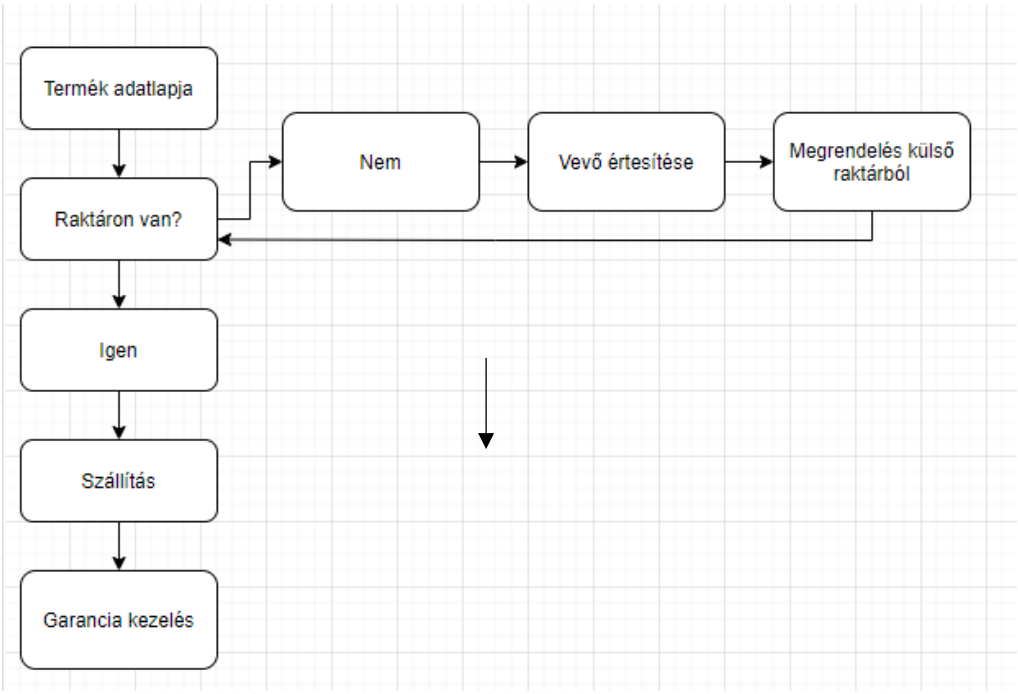
Online marketing:

Legnehezebb feladat a látogatók találása a webshopba. Szerencsére meglévő cégünk már ügyfélkörrel rendelkezik. A webshop célja, hogy a meglévő ügyfélkörnek egy online rendelési felületet biztosítson, és azok időben tájékozódjanak az újdonságokról és kedvezményekről. A hostolás után a keresőóriásokon a weboldal nem látszik csak az URL címén keresztül lehet azt elérni. SEO-t kell alkalmaznunk. (Későbbiekben részletezve.) Célszerű reklámokkal és különféle marketing fogásokkal újabb ügyfeleket keresnie vállalatnak: e-mailek, linkek, blogok, fórumok, sajtó, keresőanalitika.

4.2. Rendszertervezés:

Rendelés leadásának folyamata:

A termék adatlapjáról a kosár gomb megnyomásával tudjuk a kosárba helyezni azt. Amennyiben a fizetés gombra kattintok, meg tudom rendelni azokat a termékeket, aminek raktárkészlete nem nulla. A nulla raktárkészletűek nem megrendelhetőek. Ezeknél a termékeknél egy szállítási időt is feltüntetünk, mikor várható legközelebb készleten. Ha az raktáron van, megkezdjük a szállítást, ha nincs, várunk arra. Raktáron lévő terméknél megkezdjük a szállítást, majd a garanciakezelést. Folyamatot megtekinthetjük az 5. ábrán.

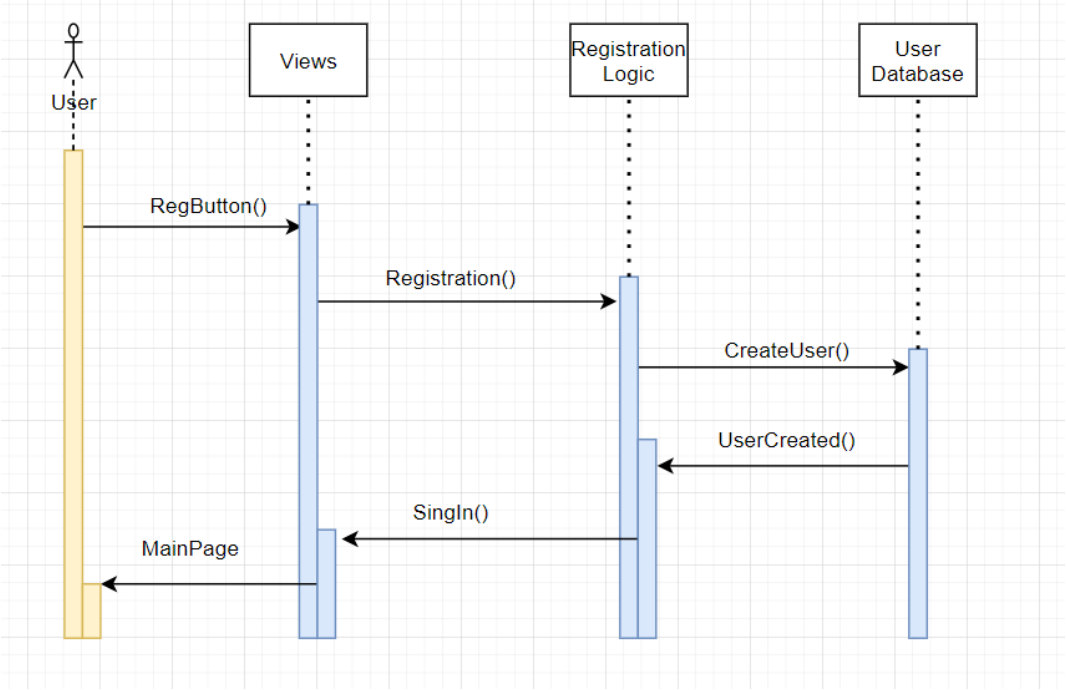


4. ábra: Rendelés leadásának folyamata

Regisztráció feldolgozásának folyamata sequence diagrammal:

Sequence diagramm: Objektumok egymás közti kapcsolatát mutatja meg időrendi sorrendben. Ábrázolja a folyamathoz szükséges dokumentumokat és a funkcionalitás végrehajtásához szükséges objektumok közt cserélt üzenetek sorrendjét. [15] [7]

A felhasználónk (User) a regisztrációs gomb megnyomásával tudja a regisztrációt elvégezni. A regisztrációs felület egy új nézeten keresztül jelenik meg. Amikor kész a kitöltésével, akkor a Regisztráció gomb lenyomásával tudja azt véglegesíteni. Elindul egy folyamat, a rétegeken keresztül a programkód létrehozza az új felhasználó példányt, majd elhelyezi ezt az adatbázisban. Adatbázis a sikeres objektum létrehozásáról tájékoztatja a felsőbb rétegeket és a felhasználót arról, hogy sikeresen létrejött az új profil. Így már be is tud jelentkezni. Sikeres bejelentkezés után a Főmenü jelenik meg. A 6. ábra szemlélteti ezt a folyamatot.

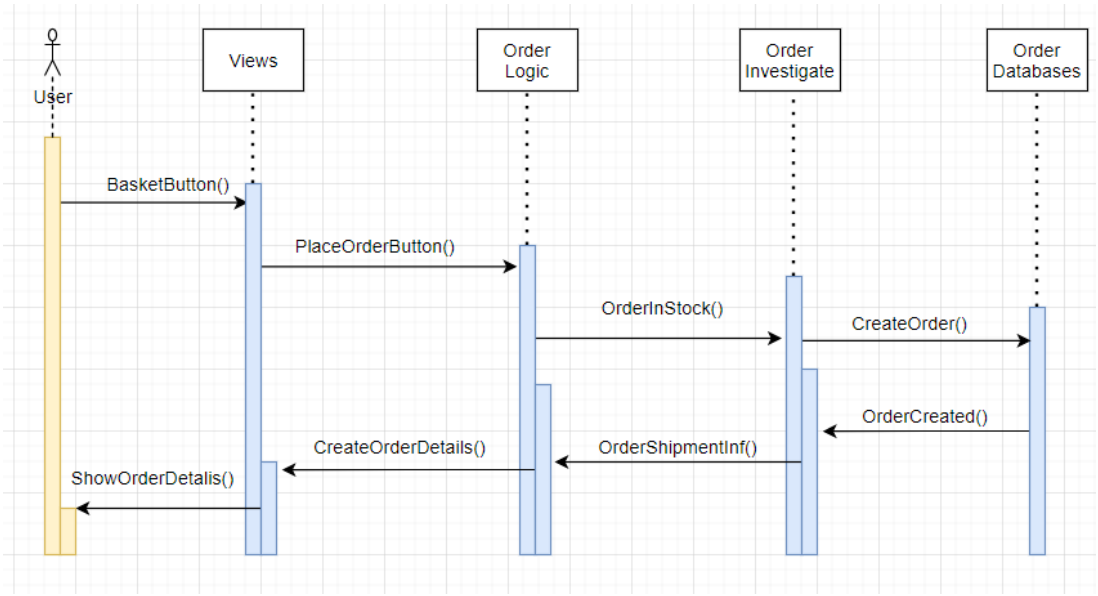


5. ábra: Regisztráció folyamata

Egy webáruház készítése során sok hasonló ilyen folyamat diagrammot lehet még készíteni. Szeretnék kiemelni a regisztráción túl meg egy érdekes folyamat diagrammot:

Az adatmodell kialakítása során figyelembe vettem a funkcionalitásokat. Olyan adatmodellt próbáltam meg kialakítani, ami a leginkább támogatja a felhasználói felület működését. Kialakításáról a 3. fejezetben olvashatunk részletesen.

A megrendelés folyamatát megtekinthetjük a 7. ábrán: A felhasználó először a termék adatlapján található kosár gomb segítségével a kosárba helyezi azt, majd ha végzett, az ott lévő termékeket a megrendelés gomb segítségével megrendelheti. A rendszer először ellenőrzi, a termék raktáron van- e, ettől függően elhelyezi azt a megfelelő adatbázisban, illetve ha nincs, elindítja a beszállítás folyamatát. Erről a megrendelőt (beszállítási idők, raktárhelyzet, stb.) tájékoztatja, felhasználó számára szükséges információkat megjeleníti.



6. ábra: Megrendelés folyamata

Az oldal felhasználása szempontjából különféle felhasználói jogköröket kell létrehoznunk, mivel nem mindegy, hogy az oldalt rendszergazdaként, vagy közönséges vásárlóként, eladóként használjuk. Ezért felhasználói jogköröket kell létrehozni a különféle user-ek között, ez a jogosultságkezelés.

Lehetséges felhasználói jogkörök:

- User: Normál felhasználó – vásárló.
- Admin: Admin – user adatokat kezel, bármihez hozzáfér.

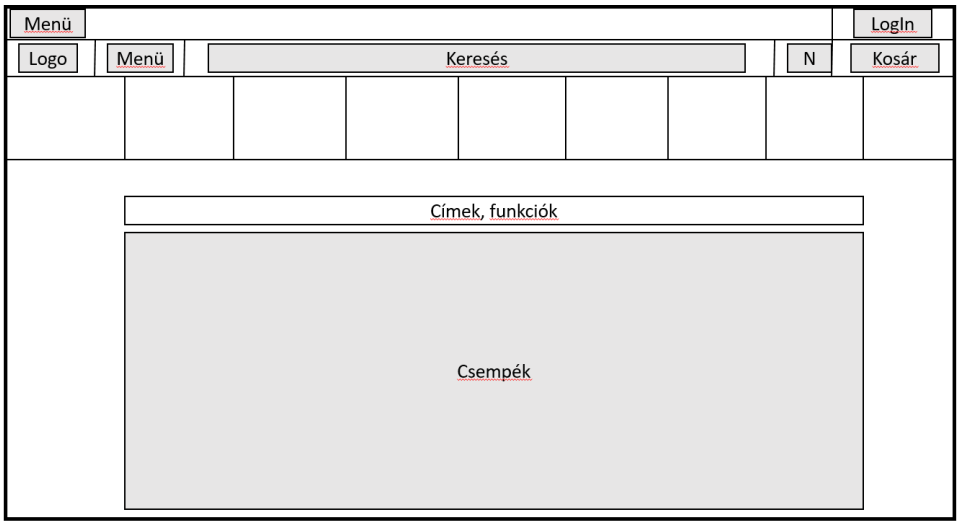
Először minden újonnan létrehozott felhasználó User. Az Admin tud esetlegesen felhasználói jogköröket kiosztani, majd a jogkörök segítségével hozzáférni a termékek szerkesztéséhez szükséges mezőkhöz. Admin bármit szerkeszthet, ő kap hozzáférést az User-ek szerkesztéséhez szükséges felülethez is. Az, hogy milyen jogkörben használjuk az oldalt, megváltoztatja annak kinézetét, funkcionalitását. Ez attól függ, melyik felhasználóval jelentkezünk be és annak milyen a felhasználói jogköre.

Gyakorlati alkalmazásban a meglévő felhasználói név után írt „–„-el lévő jogkör adja meg a felhasználó tényleges jogkörét.

Látványtervek és elképzelések:

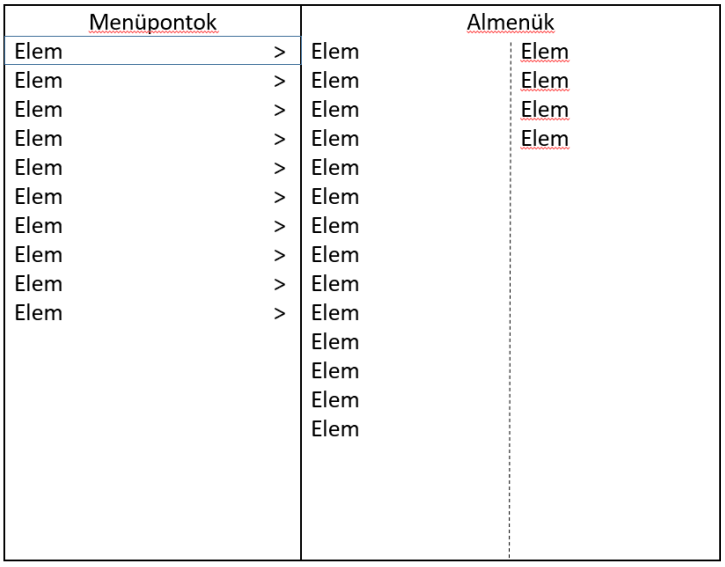
Képeket és elképzeléseket a PowerPoint program segítségével készítettem. Valószínűleg az egyes oldalak kinézete hasonló lesz ezekhez.

Főmenü és kezdőlap: Oldalra érkezéskor ezt látja a felhasználó. Rajta található a cég logója, menürendszer ikonja, fő kereső, hírek elérése, kosár, bejelentkező felület, kategóriaválasztó. A stílusát csempék alkotják, amelyeken az aktuális informatikai hírek jelennek meg. (8. ábra).



7. ábra: Főmenü

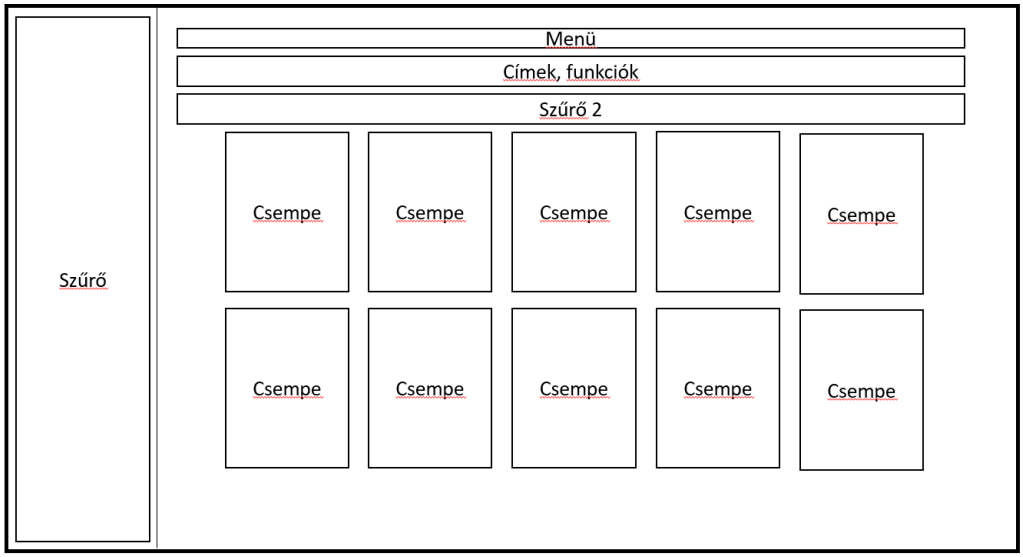
Menüválasztó és termékböngésző (menü a menüben): Menüre kattintva, ha a főmenü felé visszük az egeret, megjelennek az elemek és itt tudunk köztük böngészni. Menüpontoktól függően akár két oszlopban is lehetnek az ezek. pl.: Számítógép alkatrész → Alaplap. (9. ábra)



8. ábra: Menüválasztó

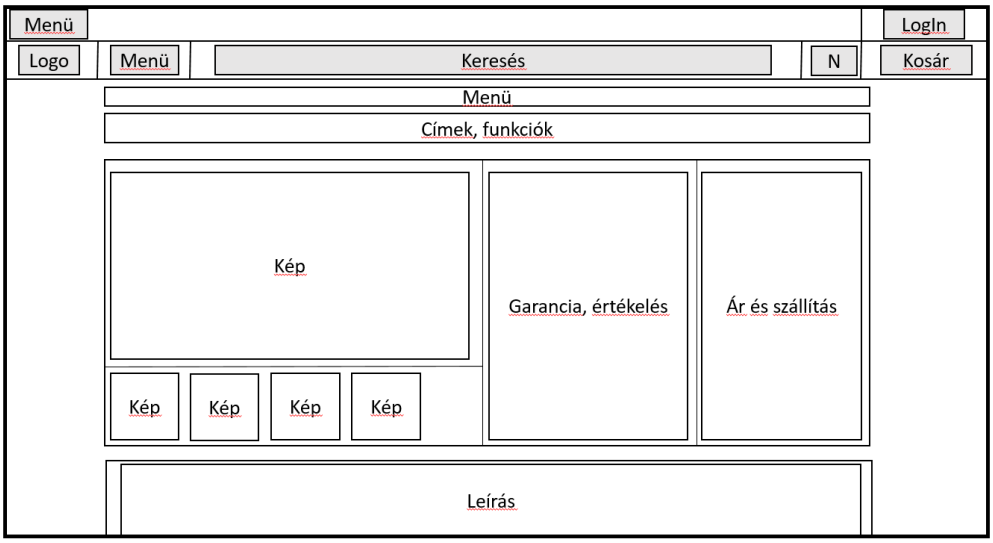
Termékböngészőbe érve, oldalsó szűrők segítségével tudunk választani az egyes szűrési feltételek között. Ezt a 10. ábrán láthatjuk.

- Szűrő 1: Típus szűrő pl.: Intel, Amd processzorok, Rizen 3, Rizen 5, DDR3 és DDR4- es ramok, stb.
- Szűrő 2: Sorrend típusát tudjuk itt megadni. Ár szerint csökken – nő, Név szerint: ABC - CBA.



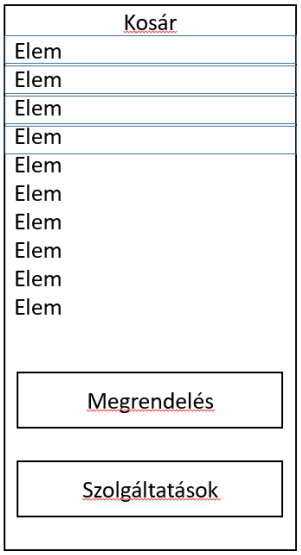
9. ábra: Termékböngésző

Termék oldal: Adott termékekhez fűződő adatokat tudjuk itt megnézni. Látunk több képet az árucikkről, garanciális feltételeiről, értékeléseket, árat, leírást. Innen is tudjuk azt kosárba tenni. (11. ábra).



10. ábra: Termék oldal

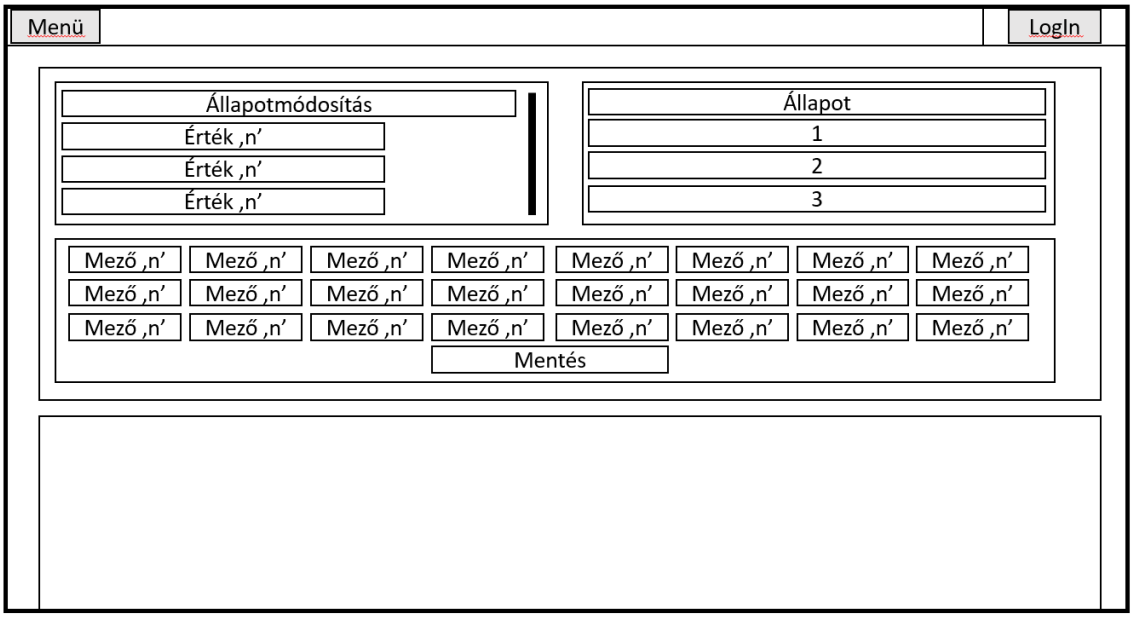
Kosár és megrendelő felület: Kosárban meg tudjuk nézni mi van abban, és ami termékeket hozzáadtunk, azt innen tudjuk megrendelni. (12. ábra). Megrendelés gombra kattintva a megrendelő felület jelenik meg. Itt pontosíthatjuk és szerkeszthetjük adatainkat, szállítási, átvételi helyeket és költségeket. Megrendelés gombbal véglegesítjük azt. (13. ábra).



11. ábra: Kosár

12. ábra: Megrendelő felület

Admin felület: Admin felhasználóként lehetőségünk van módosítani a termékek árát, nevét, képeit, garanciális feltételeit, leírását, szállítását. Beérkező megrendelések adatainak a szerkesztésére lehetőségünk van. Továbbá a főoldalon megjelenő hírfolyam híreit is itt tudjuk módosítani. (14. ábra).



13. ábra: Admin felület

Admin felületen, a különféle módosításokhoz létrehozok szerkesztési kártyákat. Vegyünk egy példát: Módosítsunk egy beérkező megrendelést: A beérkező megrendelés kiválasztása után a megrendelés állapotát tudjuk módosítani. (pl.: nem teljesíthető valamilyen indok miatt, vagy azt már szállították, stb.) Ha a megrendelő szeretne módosítani a megrendelés adatain, akkor azt a megrendelés mezőinek szerkesztésével és mentésével az Admin tudja szerkeszteni. Változtatásokat mindig menteni kell az adatbázisba, hogy a termékek állapota meg is változzon. Felület segítségével láthatjuk a megrendelés adatait, így a kollégák azt el tudják készíteni, és postára tudják adni. Módosítani kell továbbá a főoldalon megjelenítendő híreket és a meglévő terméklistát is. Azok is a fenti mintára lesznek kidolgozva.

4.3. Felhasználói élmény lehetőségei:

Célzott közönsége:

Webshop-om számítástechnikai eszközök, háztartástechnikai eszközök adás-vételével foglalkozik, így az ezek után érdeklődő emberek igényeit elégíti ki. A felhasználó egy ügyféli szempontból felhasználóbarát oldalt lát, amin megfelelő szűrési szempontok segítségével tudja megrendelni saját részére a kívánt terméket, az oldal rendelkezik termék kiválasztási hierarchiával. Napjainkban, a XXI. századi korban a számítástechnikai eszközök használata elengedhetetlen. Legyen szó a mostani járványhelyzetről vagy a munkahelyekről, kikapcsolódásról (játékok). Életünk minden területén az igények egyre inkább csak nőnek. Így ezen eszközök iránti kereslet is megnőtt, mind a fiatalabb mind az idősebb korosztály körében.

Aktív elemek, funkciók:

Felhasználói szempontból fontos, hogy egy olyan weblapot teremtsünk, aminek használata egyszerű, mégis minden figyelmet megragad. Erre az ASP.NET, és Bootstrap egyszerű és látványos megoldásokat biztosít. Ezen aktív elemek lényege, hogy a felhasználó figyelmét megragadják, legyen szó akár egy értékelés vizuális megjelenéséről vagy egy egyszerű görgetésről. Oldalamon tervezek használni ilyen vizuális elemeket is. Lehetőségek, oldalra való érkezéskor a felhasználó nemcsak csempéket lát, hanem ezeket el is tudja tüntetni, csempék megjelenhetnek, eltűnhetnek görgetés hatására. Felhasználó el tudja menteni a termék adatlapján azt kedvenc terméknek, vagy az oldal aljára érkezve megjelenik egy gomb, amivel annak az oldalnak a tetejére ugorhat. Esetlegesen, ha a termék nincs raktáron, kérhet értesítést arról, hogy mikor érhető el az újra vagy beállíthat egy árat, és ha az olcsóbb lesz, mint az adott ár, kérhet értesítést. Termék oldalán való görgetéskor az oldal tetején megjelenő egyszerűsített rendelési ablakban, meg tudja azt rendelni.

Látogatottság diagrammok, akciók:

Egy adott termék kiválasztásánál fontos a megrendelővel tudatni, hogy az a termék mennyire fog az ő igényeinek megfelelni, mennyire megbízható az a hely (jelen esetben a weboldal amit fejlesztünk), ahonnan vásárol. Ezt látogatottsági statisztikákkal, elégedettségi mutatókkal, esetlegesen független oldalakról ’beszúrt’ tartalmakkal mutatjuk meg neki. Ezt minden termék adatlapja alján minden felhasználó el tudja érni (elégedettségi mutató). Megjelennek itt a termékhez írt értékelések, kérdések, megjegyzések, eladási statisztikák (akár vizuális formában is: pl.: Ezen a héten eladott termékek száma.). Grafikonon akár tudjuk nyomon követni az elmúlt év/időszakban a termék árának változását.

Amikor egy felhasználó megrendel egy terméket, és a termékkel elégedett, valószínűleg újból szeretne vásárolni valami mást a boltból. Figyelembe kell venni, hogy más cégek is forgalmaznak hasonló termékeket és a mi célunk az, hogy a felhasználó a mi cégünket válassza. Így mikor megrendel egy terméket, a bolt kisorsol egy akció kupont, amit fel tud használni a felhasználó a következő vásárlásakor. Ezen kuponon szereplő kedvezmény függ a megvásárolt termék árától, az minél drágább annal nagyobb a kedvezmény mértéke. Kedvezmény lehet százalékos, illetve összeg formában is. Például: 15.000ft-os termék → 5%-os kupon; 80.000ft-os termék → 15.000ft-os kupon.

5. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

5.1. Igények változása

Tartalmak és szolgáltatások fejlődése:

Napjainkban minden változik, egyre modernebb technológiák jelennek meg és a technológia előre haladása megköveteli, hogy a régebbi technológiák is kövessék a jelenlegi technológiai irányvonalat. Facebook, Instagram segítségével szinte bárhol és bármilyen felhasználói réteget meg tudunk szólítani fizetett reklámokkal. Olyan helyeken tudjuk termékeinket terjeszteni, ahol az eddigi technológiák segítségével soha. Bárkiből tartalomgyártó válhat. Egy webáruház tervezése során fel kell készülni a technológiák és tartalmak fokozatos fejlődésére, fel kell készíteni azt rá. Ha az oldal sikeresen tud üzemelni, előbb- utóbb a technológia előrehaladottsága miatt szükségessé fog válni, mind a design, mind a szoftveriális fejlődés. Legyen akár szó a mesterséges intelligencia térhódításáról, vagy a blokkláncok elterjedéséről, Open Source technológiák fejlődéséről. Design javítása szükséges, mivel a felhasználók nem szívesen vásárolnak és a bizalmuk sincs meg egy olyan felületen, ami már régi (régi design) technológiát használ. Következtetnek belőle, hogy adataik nem a legnagyobb biztonságban vannak, esetlegesen a cég a csőd szélén van. Az új kinézet viszont megköveteli az új technológiák használatát, így szükségessé fog válni a szoftver javítása is. Célszerű a szoftvert úgy megtervezni, hogy ne spagetti, azaz minden mutat mindenhová kódot használni, hanem olyan kódot írni már az elején, amit később is az új technológiákhoz jól fogunk tudni használni. Így nem kell teljesen mindent átírni, mint spagetti kód esetén, hanem elég csak a frontend-et változtatgatni az időszerű igényektől függően.

A kódot a megírás során rétegzett architektúrával láttuk el, így ez már egy előkészítés a későbbi módosításra. Ezen architektúra segítségével később az alsóbb rétegekben lévő kódok, például ami az adatbázist kezeli, nem kell módosítani, vagy ha kell is, akkor azt a lehető legkisebb változtatással egyszerűen meg tudjuk tenni. Jelenlegi legjobb webfejlesztésre szánt nyelvet használtam a kornak megfelelő weboldal design-el. Ehhez is csak akkor kell hozzányúlni, ha évek múlva már a kor megköveteli a kinézet változását. A fejlesztés ezért igényel agilis hozzáállást. [19]

A weboldal további fejlesztésére megkérhetnek akár minket, akár más fejlesztő csapatokat is, igényektől (kivitelezési időtől, stb.) függően. Szerveren élesben csak fizetős szolgáltatás segítségével tudjuk nyilvánossá tenni, és ha a cég vezetése úgy dönt, hogy megéri saját szervert futtatni, a weboldalt akkor át kell írni és tenni. Jelenlegi kód támogatja ezt a fejlesztést is. Jövőben új designok (frontend) elkészítéséhez lehet, hogy jobb technológia áll rendelkezésre, mint a Bootstrap és ASP.NET , így ezeknek a cseréjére is fel kell készülni. Backend, adatbázisok terén úgy szintén. Blokkláncok egyre inkább fejlődnek és terjednek. A jövőben lehet, hogy szükségessé fog válni Bitcoin-os vagy más egyéb blokkláncot alkalmazó fizetési rendszer bevezetésére. Hozzáadhatnak Instagram és Facebook integrációkat a sikeresebb hirdetéshez, esetlegesen mesterséges intelligencia alapú cikk megjelenítés az oldal látogatóinak, így elősegítve a termék megtalálását. A webshopok jövőbeli bővítésének lehetősége szinte végtelen, ez csak pár példa volt, hogy a jövőben milyen elemekkel lehetne kiegészíteni a weboldalt.

5.2. Lehetséges újabb irányvonalak:

Manapság egyre több felhasználó használja az internetet (megrendelésekhez, ügyintézéshez). Ez a sok felhasználó leterheli a szervereket, fontosabbá válik a szerver oldalon kioptimalizált kód írása. A szerver csak azokat a feltétlenül szükséges műveleteket végezze el, amelyek a cél eléréséhez a legfontosabbak. A többi műveletet célszerű, ha a kliens számítógépe végzi el. Ez egy stabilabb működését eredményezi az oldalnak, a szerver nem áll le a működésében túlterheltség miatt. A probléma megoldására az AJAX programozás ad lehetőséget, ASP.NET-ben is megtalálható. Célja, hogy a weblap a legkevesebb információt cserélje a szerverrel, nem kell a weblapot folyamatosan újra tölteni a módosításokkor. Növeli a weblap interaktivitását is egyben. Weblap elemei közti HTML és JavaScript közti kliens oldali kommunikációt a jQuery is segíti. Alkalmazásával szintén egy optimalizáltabb szerver oldali kód hozható létre. Weblap elemei közt egy kényelmesebb kommunikáció hozható létre vele. Alapja az eseményvezérlés és a CSS szelektorok alkalmazása. Az AJAX programozás is erre épül. Jobb frontend oldal hozható létre, felhasználói élmény növekszik, több a megrendelés, gyorsabb ügyintézés. Oldalamon a fentebb említett technikák alkalmazhatók a kosárba helyezéskor, termékek adatlapján, Admin felületen. A megoldás során viszont az újra töltéses verziót használtam (szerver oldalon így optimalizálatlanabb a kódom, de ez továbbfejlesztési lehetőség). [20] [21]

A webshop tervezésekor ügyelni kell a webtárhely kihasználtságára is, mivel egy ilyen oldal akár több ezer fényképet is tárolhat. A fotók hatalmas helyet foglalnak el, ha nem megfelelően tároljuk őket. A hagyományos fájlformátumok (jpeg, png, gif, stb.) kisszámú kép esetében még alkalmazhatóak, azonban több ezres nagyságrendben a tárhelyünk hamar betelik fényképekkel. Fotók webszerveren történő optimalizált tárolásának módjára a Google kifejlesztett egy fájlformátumot ez a webp. Alkalmazásával egy gyorsabb oldalbetöltést és kevesebb tárhelyet használ az oldal. Oldalam továbbfejlesztett változata akár használhatja a képek tárolására a fent említett (webp) technikát a jpeg helyett. [22]

A projektem egy fiktív cég munkáját segíti, így a vásárlási folyamat végén szokásosan lévő bankkártyás fizetési oldalt a navigáció során már nem tettem bele. (Helyette egy üzenet jelenik meg, hogy vége a vásárlási folyamatnak, vissza a Főoldalra) Esetleges éles alkalmazásakor a webshopnak ezt az oldalt is meg kell csinálni, ami egy biztonságos fizetési felületre irányít el. (Barion, stb.). A megoldás során figyelni kell a bankrendszer visszajelzésére is. A webshopon olyan szenzitív információkat nem kérünk el a felhasználótól, mint kártyaadatok. Vigyázunk az adatbiztonságra. Esetleges adatkezelési irányelvek kidolgozása és feltüntetése a weblapon.

Webshop oldalainak olyan újabb oldalakkal való kiegészítése, amelyek a felhasználók érdekeit szolgálják, egy újabb irányvonal lehet. Például garanciális rendelések nyomon követése, kedvencekhez adás, vagy admin felület esetén újabb szerkeszthető tartalmak hozzáadása, oldal interaktívvá tétele a dolgozók számára is. A kialakítás során cél egy ügyfélorientáltabb, felhasználóbarátabb rendszer elkészítése. A fejlesztés során egy alap fikciókkal ellátott oldalt fogok elkészíteni.

6. FELADAT GYAKORLATI MEGVALÓSÍTÁSA

6.1. Kezdetek:

A feladat gyakorlati megvalósításának első lépéseként elkészítettem egy Bitbucket Repository-t és ezeken a feladat megoldásához branch-eket. Pár példa, a legfontosabbak: Mester, Develop, DevelopLogic, stb. Elhelyeztem egy commitot is a mester branchen, amiből a többi fejlesztéshez szükséges ágat elágaztattam.

Visual Studio 2019 programban pedig elkészítettem a feladat vázát. A váz részeként létrehozásra került a Repository projekt (class library - dll), a Model projekt (class library - dll), és az API-t tartalmazó ASP.NET-es web projekt is. Továbbá a feladat teszteléséhez létrehoztam egy Test projektet is (class library - dll). A projektek létrehozása után a kész részeket a dll-ek segítségével referenciaként hozzá is adtam az egyes projektekhez. A Logic réteg látja a Repository-t, a webes projekt pedig a Logic és Repository-t is egyaránt. A rétegek egymást közt kommunikálnak, metódus hívás azonban csak lefelé valósul meg, fentebbi réteg hívja a lentebbi rétegeket.

SQL Server Management Studio 18 segítségével Localhost-on létrehoztam egy adatbázist a feladat számára. Visual Studioban ADO.NET data modell segítségével hozzákapcsoltam a projektet az adatbázishoz, majd a megfelelő connection stringet a webes projektbe átmásoltam, így létesítve kapcsolatot az adatbázis elemek és a projekt között.

Egyetemi szakmai gyakorlatom során sikerült egy olyan oktatási intézményhez elhelyezkednem, ahol a gyakorlatom részeként megbíztak az intézmény weboldalának létrehozásával. Ez egy remek lehetőséget biztosított arra, hogy az eddig meglévő frontend tudásomat a gyakorlatban kamatoztassam. A gyakorlat keretében részleteiben tanulmányozhattam a frontend fejlesztési technikákat, így a dolgozat projekt részének kidolgozásához már egy biztosabb frontend ismerettel tudtam hozzákezdeni. A feladat megoldásához használt frontend fejlesztői technikák a következők voltak:

- HTML: Weboldalak leírásához készült leírónyelv, amiben a projekt alapvető kinézetét írtam le. Az egyes oldalakat külön-külön kellett kódolni. Az oldalakat a leíró nyelv legfrissebb verziójában a HTML5-ös verzióban készítettem el. Szerkezetét illetően a fejrészben (head) elhelyeztem a legfontosabb hivatkozásokat: az adott weblap stílusát leíró hivatkozást a CSS file-ra. A bootstrap hivatkozásait, kötelezően hozzá fűződő jQuery és JavaScript hivatkozásokat. Továbbá tartalmazza a meta adatokat is, mint a weboldal készítőjét (név, elérhetőség, stb.); a weboldal indexképét, a responsive (mobil nézet) alapvető hivatkozásait; kereső optimalizálás hivatkozásait, amiket a Google bootok képesek kezelni, és a webshopot a keresési találatok között listázni. Hivatkozások nélkül a weboldal például alapértelmezetten láthatatlan a legnagyobb keresőóriáson a Google-n. Keresőoptimalizálást a későbbiekben részletezem.
- CSS: Weboldalak stílusát leíró nyelv, segítségével tudom az alap HTML dokumentumomat stílusokkal ellátni. Ilyen beállítások lehetnek: betűstílus,

térközök, margók, igazítások, vagy akár gridek, amik segítségével tudok egy olyan láthatatlan hálót létrehozni, amivel a HTML elemek a megfelelő helyen és pozícióban jelennek meg. Szintén támogatja a responsive megjelenést. Használata során az egyes elemek stílusát szektorok segítségével tudom leírni. Ezeket a beállításokat akár az összes elemre, adott nevű elemekre, ezek leszármazottjaira vagy akár class-ok és id attribútomok alapján tudom állítani. Lehetőség van pszeudo-osztályok segítségével további műveleteket is elérni, mint például ha az egérrel az adott elem fölé viszem az egérmutatót, akkor más színnel vagy mérettel jelenik meg. [11]

- Bootstrap (front-end keretrendszer): CSS alapú nyílt forráskódú keretrendszer aminek a segítségével tudunk előre beállított sémák közül válogatni. Elsősorban a mobil eszközökre és a responsive webfejlesztésre irányul használata. A feladatot szeretném, ha megjelenne mobil eszközökön is, illetve más-más mérettel ellátott asztali számítógépeken. Tartalmaz előre beállított CSS beállításokat, lehetőség van adott esetben JavaScript alapú tervezési sablonok alkalmazására. Használata során például előre beállított gombokat, navigációs felületeket és form-okat tudunk használni, amiket a feladat során sokszor implementáltam.
- JavaScript: A programozási nyelvet csak kivételes esetekben használtam a feladat megoldása során: olyan esetekben, ha elég volt a megoldáshoz a script használata frontend oldalon, például: szállítási összegek számítása, felugró ablakok, esetlegesen értékek átadása az egyes elemek között. A scriptet lehetőségünk van a kódban és külső hivatkozásként is elhelyezni egyaránt, a feladat megoldása során mindkét lehetőséggel egyaránt éltem.

Frontend részekben cél volt továbbá, hogy a weboldal a legegyszerűbb és legkönnyebben kezelhető legyen a későbbiekben, ezért a megértést átgondolt mappaszerkezettel segítettem. Létrehoztam külön mappát az alap mappákon felül a CSS, JavaScript és fényképek számára. Az ASP-s nézeteket is megfelelően átgondolt mappaszerkezetben helyeztem el.

Fényképes file-k elhelyezésekor figyelembe vettem azt is, hogy az adott képek nevei a tartalmat is tükrözzék és későbbi rájuk való hivatkozások alkalmával azokat könnyebben meg lehessen találni. Például egy alaplap indexképének neve: névX-index.jpg vagy egy másodlagos képnek: névX.jpg . (X, számot jelöl meg: 1,2,…,n)

Megjelenő adatok, képek, leírások, nevek fiktív jellegűek, kitaláltak vagy random generáltak. Pár terméket, felhasználónevet, cikket példaként képekkel kidolgoztam. Többi megjelenő adatot a spawner nevű programmal random generáltam. Random generált fényképek a picsum.photos nevű oldal segítségével generáltam 1000x1000 pixel méretben. Az oldal minden generáló linkre kattintáskor/meghíváskor egy teljesen fiktív képet ad vissza, ezért a random generált termékek, egy időben mindig ugyanazzal a képpel térnek vissza.

6.2. Responsive (mobil) nézet, szép megjelenés:

Responsive nézet a mobil illetve tabletes nézetek fejlesztése, célja egy olyan alkalmazás készítése, amivel különféle mobiltelefonokon, tableteken, illetve más-más képernyő átmérővel ellátott számítógépeken akár más-más ablakméreteken egy arra tökéletesen kinéző/illő weblapot tudjunk készíteni. A feladat megoldása során ennek fő eszköze a bootstrap volt. Jelenlegi legfrissebb verziója az ötös, de én a feladat megoldásához kompatibilitási problémák miatt a 4-es verzióját használtam fel.

Bootstrap grid:

Segítségével lehetőségünk van mobilra(xs), tabletra(sm), kis laptopra(md), laptopra és asztali gépre(lg) is gridet fejleszteni. A rács elrendezés 12 egyforma oszlopot tartalmaz, amiket tetszőlegesen tudunk felosztani kisebb-nagyobb részegységekre. Minden egyes ilyen létrehozott oszlop alkalmazkodni tud a kijelző méretéhez. Például, ha van egy nagy kijelzőnk, ebben az esetben a megjelenés szempontjából előnyösebb, ha például a képeket, szöveget, hasábokat, stb. egymás melletti oszlopokban helyezünk el. Ugyanez a megjelenés mobilos nézeten már egy torzult képet eredményezne, mivel az oszlopok a kijelzőn túl szűkek lennének ahhoz, hogy a tartalmukat olvashatónak lássuk. Így a gridek ezen a nézeten, nem egymás mellett, hanem egymás alatt jelennek meg, így segítve a felhasználót, és maximális felhasználói élményt nyújtva a tartalom elérésével kapcsolatban.

Álltalános felépítése, szabályai: Minden egyes elem egy ’container-en’ belül található, ez lehet fixált méretű és teljes képernyő hosszúságú is. Ez után létre kell hozni egy ’row’ elemet, ami vízszintesen elszeparálja az egyes részegységeket egymástól (egy soron belül megjelenő elemek összessége). A sort tudjuk tagolni oszlopokra a ’col’ tulajdonsággal, lehetőség van több oszlopot is létrehozni egy sorban vagy választani az előre létrehozott 12 sor közül. [10]

Betűstílusok:

A Bootstrap-ban sok előre beállított betűstílus, címformázás közül tudunk válogatni, de vannak olyan esetek, amikor saját stílus használata célszerű, erre a CSS alapértelmezetten sok lehetőséget kínál. Élhetünk ezekkel is vagy a Google szolgáltatások segítségével akár az adott szövegre, címre optimalizált (legjobban kinéző) formázást választhatjuk.

- Font-awesome: Bootstrap és CSS betűstílus formázásokat tartalmaz előre beállítottan. Segítségével elérhetünk több száz ikont, szimbólumot és betűsítust. A projekt megvalósítása közben az 5-ös verzióját használtam fel hivatkozásként.
- Google Fonts: Nyílt forráskódú betűstílusokat tartalmazó beágyazott szolgáltatás. Hivatkozásként használtam fel pár esteben ezeket a szolgáltatásokat. [23]

Próbáltam ügyelni arra, hogy a program a felhasználó felé felhasználóbarát interakciókat adjon. Erre tökéletes eszközt nyújtott a Bootstrap-ba beépített validátorok használata. Segítségével egy rosszul beírt jelszó, karakterlánc, bejelentkezéskor/regisztrációkor megadott érték hibaüzenete is érthetően jelenik meg.

6.3.Statikus és dinamikus odalak

Statikus weboldalak nem változnak felhasználói interakciók hatására, míg a dinamikus weboldalak más néven adatbázis vezérelt weboldalak felhasználói interakcióktól függően változtatják tartalmukat.

Statikus weboldalak:

Tartalmukat a weboldal megnyitásakor tudjuk előhívni. Tartalmaik rögzítettek, nem képesek a változásra. Ilyen weboldalakat a készítője aktualizálja, de gyakran előfordul, hogy ez elmarad, így nem mindig aktuálisak. Nagy hátránya, hogy nincs könnyen kezelhető adminisztrációs felülete az adatok módosítására. Aktualizálásához beleírunk a forráskódba vagy megkérünk valakit, aki tudja azt szerkeszteni. Forráskódjukat tekintve HTML, CSS, JavaScript alapúak. Előnye a dinamikus szolgáltatásokkal szermben, hogy gyorsabbak, mivel kevesebb file-ból állnak és a méretük is jelentősebben kisebb. Weboldal költöztetésekor adatbázist nem mozgatunk, csak fileket.

Dinamikus weboldalak:

Weboldal tartalmát egy adatbázis segítségével hívjuk elő 'dinamikusan'. Tartalmukat jellemzően egy motor hajtja, felhasználói interakciótól függően változtatja azokat, így szerkesztésük is jelentősebben egyszerűbb. A rajta megjelenő adatok valós idejűek, így akár a felhasználóra szabva, a róla gyűjtött adatok alapján jelenítheti meg azokat. Alkalmazhatunk kliens és szerver oldali scripteket is egyaránt. A tartalmat adatbázisból és egy CMS (tartalomkezelő) rendszerből nyerheti. A dinamikus weboldalakat előszeretettel alkalmazzák webáruházak tervezésekor. Jelszavas bejelentkezés után így képesek leszünk megrendelést indítani, vagy egyéb tartalmakat és funkcionalitásokat elérni. Tartalmuk folyamatosan trendkövető, mivel gyorsan aktualizálhatóak. Kódolásukhoz szerveroldali nyelveket is használni kell: ilyenek: PHP, ASP.NET, JSP, Servlet, stb. [24]

Lényeges tényező egy webshop megtervezésekor, hogy a futtató környezet ára milyen költségű. Statikus és dinamikus oldalak ára közti különbég jelentős, ez a fentarthatóságból és a funkcionalitások magas fokú eltéréséből adódik. Dinamikus oldal esetében szükségünk lehet egy adatbázis szerverre és egy alkalmazást futtató operációs rendszerre is. Természetesen mindkét esetben szükségünk van a tárhelyre mutató domain címre is, aminek költségei is sok pénzbe kerülhetnek.

Domain név az internet egy meghatározott részét, tartományát egyedileg leíró megnevezés. Számítógépek azonosítására szolgált névtartomány. Kiosztásuk a DNS (Domain Name System) szabályai szerint hierarchikusan történik. [24]

Dinamikus tárhelyek nagy hátránya, hogy azok feltörhetőek, ebben az esetben az adatbiztonságról gondolkodni kell. Statikus tárhely esetében erre nincsen szükségünk, mivel nincs mögöttes tartalom, amit el lehetne lopni. A projekt céljától függően kell választanunk, melyik verziót is szeretnénk használni a megoldáshoz. A dolgozathoz én egy dinamikus weboldalt hoztam létre jellege miatt.

6.4. SEO (Search Engine Optimization): Keresőoptimalizálás

Adott kulcsszóra való optimalizációnál egy dinamikus weboldalt nehezen tudunk csak optimalizálni, mivel az dinamikus jellege miatt csak nehezen támogatja. Továbbá szintén gondot okoznak az úgynevezett „végtelen görgetésű” oldalak is, ahol a további tartalmak görgetés hatására generálódnak. Statikus weboldalt könnyebben lehet optimalizálni a mindig azonos tartalom miatt. Ezek az oldalak könnyebben is töltődnek be dinamikus társával szemben, optimalizáláskor ez sem utolsó szempont.

SEO egy marketing stratégia, amely figyelembe veszi a keresőóriásokon megjelenő találati listán a weboldal elfoglalt pozícióját. Figyelembe veszi az adott keresőmotor működését, kulcsszavakat, célzott célközönséget. SEO rövidítés vonatkozhat keresőoptimalizálásra szakosodott cégekre, szakemberekre is, de ebben a kontextusban a nem fizetett találati listában, a weboldal elfoglalt helyét jelenti. A keresőmotorokat üzemeltető cégek (pl.: Google, stb.) maguk is támogatják a SEO azon formáját, ami segíti a weboldal sikeres feltérképezését, de a keresőmotor belső rangsorolási tényezői továbbra is üzleti titkot képeznek. [25]

Találati ranglista felállításakor a keresőmotor figyelembe veszi az oldal szerkezetét, dinamikus oldalakat statikus oldallal szemben jobban preferál. Jobb helyet érhet így el a találati rangsorban. Optimalizáltalanul éles szerveren való futtatáskor a keresőóriásokon a webshop láthatatlan, így a felhasználók nem tudják azt megtalálni, csak az URL címén keresztül. Fontos egy olyan weboldalt fejleszteni, amit a felhasználó könnyen átlát, így számára könnyebben tud választani az azon megjelenő tartalmak közül, segítve a jobb eladási rátát és növelve a bevételeket.

A webshopomat éles szerveren futtatni nem fogom, de a keresőoptimalizálást elvégeztem rajta. (15. ábra) Továbbiakban nézzük át ennek lépéseit a Google keresőóriáson:

```
<meta name="robots" content="max-snippet:-1, max-image-preview:standard">
<link rel="canonical" href="https://www.jhshop.hu/index.cshtml" />
```

14. ábra: SEO

Felépítését a Google bárki számára a keresőmotorján elérhetővé tette. Elemei:

- Metatag-ok segítségével a keresőmotorok számára meg tudjuk adni az oldal feltérképezhetőségét. Tudjuk szabályozni, mely keresőrobotokra vonatkoztassanak: bármely(robots) vagy specifikus(googlebot). Szabályozni tudunk segítségével bármit. Jelen esetben a találati listán megjelenő maximális oldalt leíró, felhasználó számára bemutató szöveg hossza legyen karakterekben mérve: nem limitált. A weboldal indexképe pedig: normál méretű.
- Duplikációk (Cannonial links): Ha egy weboldalnak van másik változata is, például asztalra, telefonra, tabletre fejlesztett verzió, ezt a google duplikált verziónak tekinti. Ebben az esetben kialakít egy URL-t gyűjtőverzióként, azt feltérképezi, a többi verziót pedig ismétlődő URL-nek tekinti, így azokat ritkábban térképezi fel. Jelen esetben az asztali, telefonos, tabletes verziót a responsivitás révén ugyanaz a weboldal jeleníti meg, de éles szerveren való

futtatáskor a webshop URL címére hivatkozva és az URL/index.cshtml-re vagy a controller-re hivatkozva ugyanazt az oldalt kapnánk (Főoldal), amit célszerű ismétlődő tartalomként megjelölni.
- Robots.txt: Weboldal gyökérmappájában található szöveges állomány. Keresőrobottal tudatja, mely címekhez férhet hozzá, keresőmotorok távoltartására szolgál. Találati rangsor kialakításakor figyelembe veszik az állomány létezését, így azt célszerű létrehozni és beállításai közt megadni, hogy az adott oldal feltérképezhető.

Végleges találati rangsor felállításakor nem csak a robotok számára készült tagokat figyeli a keresőrobot. Sok más beállítást is figyelembe vesz:

- Metatagok: készítő, téma, célközönség, olvashatóság
- Címek: Fontos az egyes megjelenő weblapoknak más-más címet adni. Tartalom és a cím is változik így. Ezeknek a címek maximális hossza 50-60 karakter közé essen.
- Címsorcímkék: Címek, szövegektől eltérő tartalmak azonosítására szolgálnak a HTML nyelvben. Tartalmukat a szöveg, tartalom szervezése szempontjából figyelembe kell vennünk. Könnyebb megérteni a jól szervezett tartalmat, a keresőmotorok, mind az emberek számára egyaránt.
- Képek „alt” attribútuma: Képekhez hozzákerülő attribútum, amely leírja szövegesen annak tartalmát. Megjelenik, ha a kép letiltott vagy az valami oknál fogva nem jeleníthető meg. Keresőrobotok nem tudják látni a kép tartalmát, így fontos számukra annak leírása. A Google a keresési szóhoz így könnyebben tud olyan oldalon szereplő képeket rendelni, amelyek a legjobban megfelelnek a kritériumoknak.
- Nézetek: Nézetablakok jobban lehetővé teszik a weboldal megjelenítését egyes számítógépeken, eszközökön: Igazodik a weboldal az eszköz ablakának méretéhez. (Responsive) [26]

Keresőmotor működése nem publikus, így annak működését közvetlenül nem lehet tudni, csak következtetni rá. Említett technikák nem jelentik a ranglista elejére kerülést, csak egy jobb helyet az adott kulcsszóra való kereséskor a weboldal számára. További keresést segítő tényezők lehetnek a közösségi média tagok is (Open Graph: Facebook) vagy a sémamegjelölés, ami lehetővé teszi a keresőrobot számára, hogy ne csak olvassák a tartalmat, hanem meg is értsék, hogy miről szól a weboldal.

Megvalósítás során gondolhatunk egy olyan eshetőségre, hogy a webshopot böngészve a felhasználó elnavigál egy másik oldalra, de a mi oldalunkat nem zárja be. Lehetőség van arra, hogy a felhasználót visszacsalogassuk. Ennek eszköze például az lehet, hogy kis idő elteltével a weboldal címkét megváltoztatjuk valami szembetűnőre (színek, mozgások, animációk), így hívhatjuk vissza oldalunkra a felhasználót. [26]

6.5. Sütikezelés: (HTTP - Cookies)

Sütiket a webszerver hozza létre a felhasználó számítógépén. Ez egy információcsomag, amit a böngésző visszaküldhet a webszervernek. Erre elkülönített könyvtárban tárolódnak. Minden sütinek adhatunk nevet, értéket, lejárati időt, stb. Segítségével lehetőségünk van korábbi adatokat összekapcsolni jelenlegiekkel. Leggyakoribb felhasználási módja a webáruházakban a bevásárlókosár tartalmának tárolása egy maghatározott ideig vagy regisztrált felhasználói azonosítás. [27]

Saját webáruházam is használ sütiket. Ezek felhasználási módja a kosár tartalmának tárolása és a bejelentkezett felhasználói adatok tárolása. Lehetőségünk van a regisztráció megadásakor kiválasztani, hogy kívánjuk e menteni a bejelentkezést. Ha igen, ebben az esetben egy új süti segítségével tárolom a felhasználói profilt a lejárati dátumig vagy a kijelentkezés időpontjáig. Hasonlóan eljárva a kosár tartalmával is. Itt a lejárati dátum és a kosár ürítése gomb megnyomásáig tároljuk az adatokat.

Munkamenet – Session:

A sütik egy speciális fajtája. Számítógépek közti kommunikáció egy olyan formája, ahol az egyik számítógép a másikról adatokat tárol el átmenetileg (amíg például az oldal meg van nyitva). Egy lehetséges felhasználási módja: session segítségével tudjuk, hogy egy adott felhasználó bejelentkezett-e. Szerverrel való kommunikáció során megkapjuk az egyedi azonosítót és innen tudjuk, hogy egy időben bejelentkezett több felhasználó közül melyikkel is kommunikálunk. A kezelt kódokra vigyáznunk kell, mivel azt könnyen el tudják lopni és így adatainkkal visszaélni.(session hijacking) [28]

Projektem is használ munkameneteket, minden megnyitáskor generálódik egy új session. Segítségével tárolom a bejelentkezett felhasználót és adatait, a kosár tartalmát nem bejelentkezett felhasználó esetén vagy a felhasználói jogköröket.

ASP.NET MVC-ben a vezérlő(Controller) és a nézet(View) között kommunikációra van szükség. Lehetőségünk van modellt és objektumokat is használni egyaránt. Kommunikációt session-ok és cookie-ek segítségével is meg tudunk valósítani. Ideiglenes adatok átadásakor nem célszerű a modell osztályt használni az adatok átadására, ebben az esetben alkalmazhatjuk a ViewBag, ViewData, TempData tulajdonságokat. Rövid áttekintés a tulajdonságokról:

- ViewBag: Lehetővé teszi az értékek dinamikus megosztását a vezérlőből a nézetbe. Olyan adatok tarolására szolgál, amelyet a modell nem tartalmaz. Hozzá tudunk rendelni egy dinamikus tulajdonságot és ennek értéket adni. (pl.: .Title(cím), .MyMessageToView(üzenet)) Az átadott értékhez hozzáférni bármely HTML tagok közt a nézetben: ’@’ karakter felhasználásával.
- ViewData: Hasonló a ViewBag-hoz, annyi különbséggel, hogy a kulcsok karakterláncok (key-value). Segítségével akár komplex objektumokat is át tudunk adni. (lista, enumerable, queriable, stb.) Ezt a modellben megfelelően kell kezelni: castolással, akár razor syntax kóddal. Ehhez szükséges referenciákat a nézethez is hozzá kell adni.

- TempData: Ideiglenes adatok tárolására szolgál, későbbi kérés teljesítése után törlődik. Olyan adatok átvitelére szolgál, amelyek erre nem érzékenyek. Egy rövid életű session. Működése a ViewData-ra hasonlít, kulcs-érték párokkal lehet adatokat beletenni. Komplex adatokat is kezel. Lehetőség van az adatok további tárolására is (újra töltés utáni) speciális beállítások alkalmazásával. Természetesen ez esetben is a nézetben ügyelni kell a megfelelő típuskényszerítésre és az adatok átalakítására. [29]

Razor syntax (Razor kód):

Szerver alapú kódok weboldalakba ágyazását teszi lehetővé C# nyelven, ASP.NET-en alapul. Segítségével dinamikus tartalom hozható létre. Szerveren futó kód összetett feladatokat hajt végre: eléri az adatbázist, szűréseket hajt végre. Az oldal meghívásakor az adott razor kód lefut, majd végrehajtja az ott kódolt utasításokat. Szükséges tartalom eléréséhez a backend oldali kódokat használja.

HTML alapú kódokban akár kódblokkok segítségével tudunk létrehozni dinamikus tartalmakat. Változókat a ’@’ segítségével is leírhatunk. Kiterjesztése ezeknek a file-oknak ASP.NET-ben egy sajátos filekiterjesztés: .cshtml. Razor weboldalak kétféle tartalommal rendelkező HTML oldalként írhatók le: Razor kód és HTML kód. HTML kód a weboldal kinézetéért felelős, használhatunk díszítésre CSS és Bootstrap referenciákat is. A Razor kód pedig a dinamikus tartalmakért felelős: adatbázis elemeinek kiolvasása, egyéb változók megjelenítése.

Futtatása: Először a szerver a Razor kódot futtatja, mielőtt elküldi a HTML oldalt a böngészőnek. Szerver olyan feladatokat hajt végre, amelyeket a böngészőben nem lehet: elérni a szerveren lévő adatbázist. A szerverkód képes menet közben dinamikus tartalom létrehozására. A folyamatokat a böngészőből nézve a szerverkód nem tűnik többnek, mint egy statikus HTML oldal. [30]

Kivételkezelés a frontend oldalon:

A weboldalt frontend oldalon is fel kell készíteni olyan kivételesen képződő hibák esetére, amikkel kódolás közben nem számítunk vagy számíthatunk. Kivételek azon olyan események, amelyek szándékosan vagy nem szándékolt módon képződnének az oldal futásában és annak futását a továbbiakban megakadályozza. Erre az esetre kivételkezelést alkalmazunk, próbáljuk a hibát megoldani a keletkezéskor, kezelni azt. Így a program futását nem zavarjuk, ez a hibatoleráció. Kivételkezelést nem alkalmazva a program futását zavarjuk, amik akár adatvesztéssel is járhatnak.

6.6.Vállalatirányítási rendszerek:

Szakdolgozatomban egy olyan webáruházat készítünk, ami egy jelenleg is működő, de kitalált cég igényeire van elkészítve. A számítástechnikai hátteret ki kell egészítenie egy céges háttérnek, amiben a vállalat működését felügyeljük:

A marketingkörnyezet folyamatos figyelemmel kísérése alapvetően fontos a cégek működésének szempontjából, hiszen ha nem tud időben és megfelelően reagálni a környezet változásaira, vagy ha a megrendelők igényeit nem követi, akkor nem sokáig képes lépést tartani versenytársaival, előbb vagy utóbb elveszíti kliensk örét. Amennyiben nem mérlegeljük a körülményeket és a felkínálkozó lehetőségeket, vagy akár nem próbáljuk meg elkerülni a fennálló fenyegetéseket, akkor képtelenek leszünk a piacon maradásra és hasznot sem hoz a vállalkozásunk.

ERP rendszerek:

Manapság egy kis, közepes és nagy vállalkozás számára is elengedhetetlen egy jó vállalatirányítási szoftver, ez az ERP. A számítástechnika már mindenhova beférkőzte magát. Mérlegelni szükséges, hogy a 21. századi módszerekkel szeretnénk-e dolgozni és a partnereinkkel kapcsolatot tartani, vagy sem. A cégvezetés és a partnerek többsége is manapság már egyaránt otthon van az informatikában.

Elengedhetetlen az ERP-k használata: Kezdetekben mindenhol kézzel iktatták a papírokat. Pár év után már szekrénysorok, akár termek álltak mindenhol ezek tárolására. Már az üzleti dolgok is (Word, Excel) formában kerülnek hozzánk és mennek is ki. Ugyanez más cégekre/vevőkre is jellemző. Régi rendszer segítségével akár pár napba, hétbe is telik egy kimutatás írása (pl.: pénzügyi kimutatás, stb.). Olykor pedig már túl későn válik világossá, hogy likviditási vagy határidős problémák miatt nem jól üzemel a cég.

Vállalatirányítási információs rendszerünkbe mindazon folyamatok integrálódtak, amelyek a vállalatunk üzletmenetében előfordulnak, ide értve a vállalkozás irányításának, a termékek beszerzésének, illetve azok értékesítésének feladatait is. Fontos szempont a nyomon követhetőség.

Részegységei:

- Pénzügy, számvitel: Vállalatunk pénz mozgásinak nyomon követése: bevételeink és kiadásaink.
- Logisztikai és értékesítési folyamat: Cél a vállalat életének hatékonyabbá tétele, erőforrások megfelelő felhasználása. Tároljuk az ügyfeleink adatait, elérhetőségeit a későbbi kapcsolattartás miatt, jövőben így akár személyre szabott ajánlatokat is tudunk tenni, esetlegesen kedvezményeket alkalmazni. Jelentéseket tudunk későbbiekben generálni az adatokból; Tároljuk a beszerzésekre vonatkozó adatokat is, esetlegesen javaslatokat, kiegészítést; Árukészlet: Kezeljük (raktározunk, leltározzuk) a meglévő árukészletünket, hiánycikkekből új megrendelést indítunk a beszállításra. Beszerzett áruk paraméterezhetőek: akár ár, méret, súly szerint. ;Dolgozók: Kezeli a vállalat dolgozóinak adatait. (humán

erőforrás) – bérezés, ledolgozott munkaidő nyomon követése, táppénz, szabadság időtartama, stb., nyilvántartását; Szállítás: A cégtől való kifelé szállítást partnercégek oldják meg. Az ügyfelek megrendeléseit szállítja a cég. ; Tárgyi eszközök: Olyan eszközök listája, amit a cég, alkalmazottjai napi szinten is használnak. (pl.: csomagoló anyagok, ragasztók, kötegelők, szerszámok)

- Jelentéskészítés: Generálhatók az egyes ERP modulokból. (pl. kimutatások) Készíthetünk pénzügyi, beszerzési vagy az ügyfelekkel kapcsolatos jelentéseket is; Beszállítóértékelés: Rangsoroljuk beszállítóinkat szempontok alapján. (termék ára, beszállítási idő, minőség, mennyiség, popularitás ügyfelek között) és ezeknek a szempontok alapján döntünk, hogy következő évben ki(k) is legyenek a beszállítók. Ezekhez a döntésekhez részegységekre épülő kimutatások, információk alapján tudunk következtetést levonni, aminek akár erőforrás management-i döntés is lehet az eredménye.

Egy cég sikeres működéséhez sok dolog összetett és eredményes munkájára van szükség. A cég már eddig is sikeres, meglévő alapokkal rendelkezik. Jelenleg a működéséhez adminisztrációs személyzetet biztosít. Pár példa a dolgozók adatait feldolgozó, tároló személyzet feladataira:

- Személyes adatok kezelése, tárolása.
- Munkaköri leírások készítése, tárolása.
- Munkaszerződés archiválás.
- Esetlegesen diákmunka esetén: tanulmányi szerződések nyilvántartása.
- Baleseti értesítendők nyilvántartása
- Orvosi vizsgálatok időpontja és eredménye.
- Nyelvismeret nyilvántartása. (számlázás és bérszámfejtés)
- Végzettségek nyilvántartása.
- Előző munkahelyek nyilvántartása. Fegyelmik / Kitüntetések nyilvántartása.
- Kiadott eszközök nyilvántartása [31]

B2B - Business to Business:

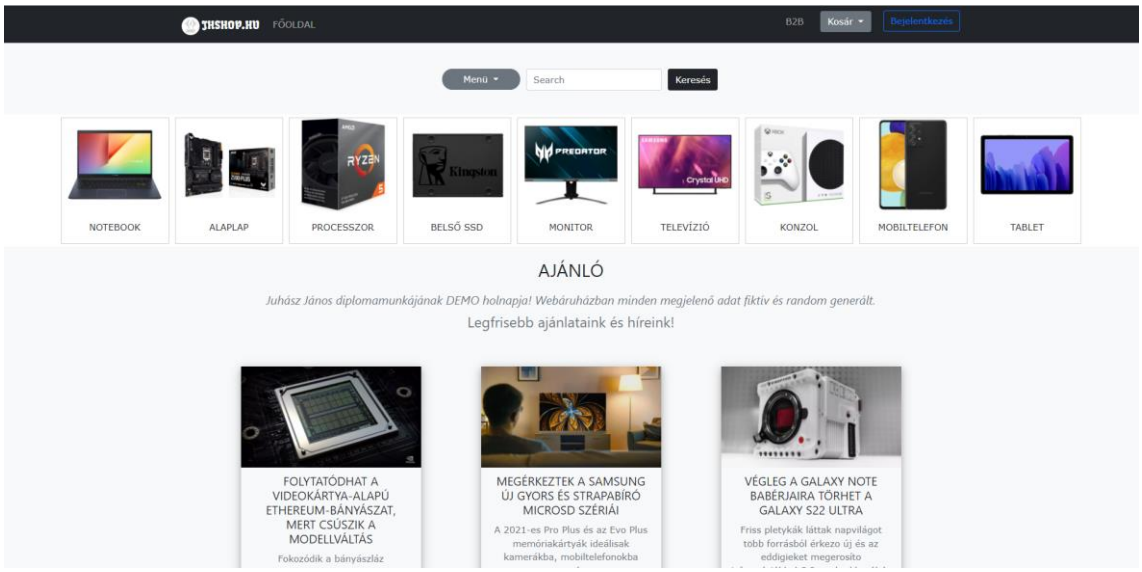
Két vállalat közti üzleti partnerség. A cég partnerségben áll olyan vállalatokkal, amelyek cégünktől vásárolják termékeiket és értékesítik azokat. Az alapvető tevékenységünk a Business to Consumer (B2C), azaz magánszemélyeknek értékesítjük a termékeket. A vállalat vezetése viszont nyitott az új partnerek-viszonteladók jelentkezésére is. Viszonteladókkal szoros kapcsolatot tartunk, ez szakmai téren, racionalitás és vállalkozás érdekek miatt is nagyon fontos. Viszonteladók csak nagy tételben vagy értékben tudnak vásárolni. Marketing szerepe miatt is fontos stratégia a B2B, mivel a vásárlók elé, csak hirdetésekkel, (online) kommunikáció által tudunk kerülni. [31]

7. A PROJEKT BEMUTATÁSA:

7.1. Főoldal

A dolgozat további részében bemutatom az elkészült projektet és annak egyes oldalait, részeit: a felhasználói funkcionalitásokkal, megoldási módszerekkel, esetleges nehézségekkel.

A 16. ábrán láthatjuk a Főoldalt és annak felépítését, kinézetét.



15. ábra: Főoldal

A felhasználót a weboldal URL címére kattintáskor vagy az URL/Index kontroller hivatkozáskor a Főoldalra juttatja el. Alapértelmezetten, minden felhasználó bejelentkezetlen az oldalra. (Bejelentkezésről a későbbiekben) Lehetőségünk van áttekinteni a kosár tartalmát (1. számú melléklet), partnerségi feltételeket áttekinteni. A címsorban találhatunk egy hivatkozást vissza a Főoldalra és egy Logót, ami a bolt nevét és logóját tartalmazza. Logóra való kattintáskor az oldal tetejére jutunk vissza, ami hasznos funkció tud lenni, ha lentebb görgetünk és szeretnénk visszajutni az oldal tetejére. Az oldal tetején lévő navigációs sáv az oldalon való navigáláskor bárhová elkísér.

A kosár tartalma az oldalra érkezéskor üres, az oldalak közti navigáció során a tartalmát megőrzi és továbbviszi azt. A Gombra való kattintáskor a tartalmát megjeleníti, esetlegesen lehetőségünk van annak törlésére és megtekinteni a weboldal szolgáltatásait is. (Elérhetőségek, szállítási feltételek, árak, átvételi címek.) A szolgáltatásokat leíró weblap kinézetét szintén az 1. számú melléklet mutatja meg. Megrendelést indítani is a kosár segítségével tudunk a „Megrendelés gomb” megnyomásával. Megrendelés csak abban az esetben indítható, ha a kosárban legalább egy termék van.

Az oldalon található egy kereső, aminek segítésével az elérhető terméklistában kulcsszavakra tudunk keresni. Ez a kereső minden termék között keres. Található egy másik kereső is a webshopban, ami viszont adott termékek közt hajt végre keresést.

A Menü gomb segítségével tudunk válogatni az elérhető termék kategóriák közül. A menürendszer menü a menüben rendszerű, ez azt jelenti, hogy a fő termék kategóriák fel vannak osztva alkategóriákra és amint egy fő kategória felé viszem az egeret, az lenyílik és megtekinthetjük az alkategóriáit. Így tudunk vele szűrést indítani. Fontos megjegyezni, hogy az egyes alkategóriákra kattintáskor nem biztosan ad ki a keresés eredményt, mivel az adatok véletlenszerűen generáltak. A 1. számú mellékleten megtekinthetjük a lenyíló Menü tartalmát is. Egy alkategóriára való kattintáskor az adott kategóriának megfelelő termékeket listázza a program, navigál a termékeket listázó oldalra.

Gyakrabban használt fő kategóriákat kiemeltem egy külön sorban. Itt képekre való kattintáskor tudunk termékek között keresni. Kattintáskor egy navigáció történik a termékeket listázó oldalra. 9 fő kategória közt tudunk válogatni, az adatok véletlen generálása miatt itt is a találatok száma eltérhet. Az alaplapok kategóriát viszont kidolgoztam, itt az első pár találat esetén bemutatva milyen lenne a webshop, ha nem véletlenszerű adatokat használnánk.

Az ajánló szekcióban megjelenik egy „hírfolyam” csempés elrendezésben. Itt lehetősége van a felhasználónak a megjelenő cikkek között olvasni. Tájékozódhat az újdonságokról, akciókról, hírekről, stb. A webshop személyzete ezeket a híreket szerkeszteni is tudja a kezelői felületről. (Kezelői felületről a későbbiekben.)

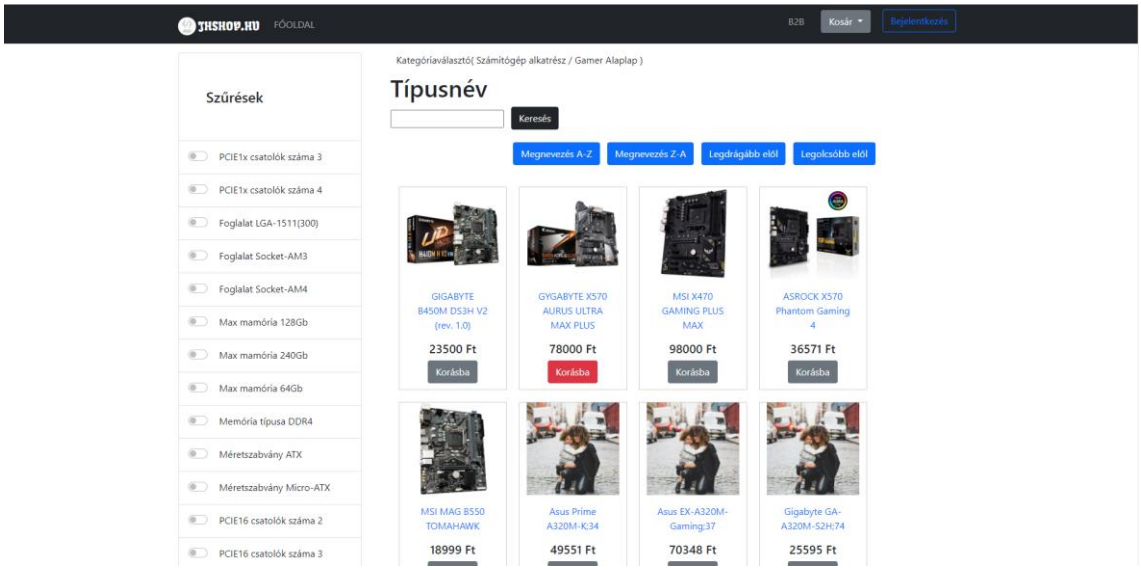
Minden hír egy fényképből, címből és alcímből áll. Egy hírt amennyiben érdekesnek találunk, kattintáskor az megnyílik egy külön oldalon és el is tudjuk olvasni a tartalmát. Egy hír megnyitását követően a 2. számú mellékleten lévő oldal fogad. Egy hír elolvasása után a Főoldalra navigálni a „Főoldal” navigációs menüpont segítségével tudunk. A hírfolyamban korlátlan mennyiségű hír elhelyezhető. A hírfolyam is alkalmazkodik a képernyő hosszához, asztali számítógépen rácsos elrendezésben jelenik meg (soronként maximálisan 3 hír), míg telefonon oszloposan.

Az Főoldal aljára érkezve a lábrész fogad minket. Ez is hasonlóan a navigációs sávhoz, az oldalon bárhová navigálva megjelenik alul. A navigációs sávval közösen egy keretet alkotnak: hasonlóan a fejléc és lábléchez. A többi oldal megalkotásakor csak az oldal középső részét kódoltam le, a keret változatlan. A láblécen található egy gomb, ami segítségével az aktuális weblap tetejére juthatunk és egy hivatkozás a készítő nevére. (én) Megtekinthetjük a lábrész tényleges kinézetét és megjelenését a 2. melléklet segítségével.

Főoldal megírása során nehézségként lépett fel, hogy először az oldalt asztali számítógépre készítettem el, majd ezt alakítottam át mobilos (responsive) nézetre. Át kellett ez indoknál fogva alakítsam az egész keresőrendszer, menük, logo megjelenést és elrendezését, mivel azt a mobilos nézetben nem lehetett volna szépen megjeleníteni. Bejelentkezés kezeléséhez meg kellett ismerkedjek, hogy a süti kezelés és a Session a C#-ban ténylegesen hogyan valósul meg. Erre rövid időn belül sikerült rájönnöm, így már lehetőség nyílt, hogy a felhasználókat be- és kijelentkeztethessük.

7.2. Termékek listázása

Termékeket tudunk listázni a főoldalon lévő kereső, vagy a kategóriaválasztók valamelyikével. Bármelyik módszert is választjuk a termékeket listázó oldalon kötünk ki, az oldalt megtekinthetjük a 17. ábrán.



16. ábra: Termék kereső

Az ábrán szereplő kategória a Számítógép alkatrészek közül a Gamer Alaplapok közé tartozó termékeket mutatja be. [1] Keresésre itt is lehetőség nyílik, de itt már nem az összes elérhető termék közt keresünk, hanem csak az adott kategóriákba tartozók közül. Például: „ULTRA” alaplapok, stb.

A keresésnek vagy a kategóriának megfelelő termékeket lehetőségünk van rendeztetni is. Alapértelmezetten nincsen rendezés beállítva, adatbázisból történő kiolvasás a rendezés alapja. Azonban ha szeretnénk, rendeztethetjük a termékeket ár szerint növekőbe vagy csökkenőbe. Név szerint is rendeztethetünk abc szerint növekvő és csökkenő sorrendbe. A vásárlás során ez a felhasználó számára egy hasznos funkció lehet, a modern webáruházak egyik alapvető kelléke. Vásárlás során az egyik elsődleges döntési szempont lehet a termék ára is.

Webshopunk számítástechnikai termékek forgalmazásával foglalkozik, fontos tényezőt jelentenek a termékek paraméterei, mivel egy alaplapba nem mindegy milyen processzort vagy memóriát választunk. A baloldalon található típusszűrő segítségével tudjuk meghatározni milyen típusú termékekre is van szükségünk. Kapcsoló segítségével beállítom, a szűrési feltételt majd a Keresés gomb megnyomásával lehet a típusszűrést elvégezni. Módosításra a szűrés után is lehetőség van, de ebben az esetben újra kell keresnünk. Szűrési feltételeket a keresés után a Törlés gomb megnyomásával tudunk

[1] Első 5 termék kidolgozott: megmutatja, hogy milyen lenne, ha nem random generált lenne az oldal. Többi termék viszont random generált. Ezért vannak egyforma képek és azonos ’alaplap’ nevek, tulajdonságok mindenhol. (pl.: TV-k)

törölni. Ez után a termékeket tartalmazó oldalra jutunk, ahol akár ismét tudunk módosítani a feltételeken.

A Főoldalon megismertetett keret (navigációs menü és lábrész) ezen az oldalon is megjelenik.

Terméklistában bármennyi termék megjelenhet, igazodik a lista a monitor méretéhez. Minden termékről megjelenik egy indexkép, a termék neve, ára és egy kosár gomb. Kosárba terméket csak bejelentkezett felhasználó tehet. Ha nem vagyunk bejelentkezve, erről az oldal egy üzenettel tájékoztat minket és felajánlja, hogy elnavigál minket a bejelentkezés oldalra. Termék nevére vagy képére kattintva megnyithatjuk annak oldalát. Ha a termék nincs raktáron, szállítása így sok időt vesz igénybe, akkor azt nem tudjuk megrendelni és a program számunkra ezt úgy jelzi, hogy a Kosár gombot pirosra változtatja. (nem tudjuk ezeket a termékeket kosárba helyezni). Adminisztrációs személyzet tudja változtatni a termékek raktárkészletét.

Felmerülő nehézségek az oldal kapcsán:

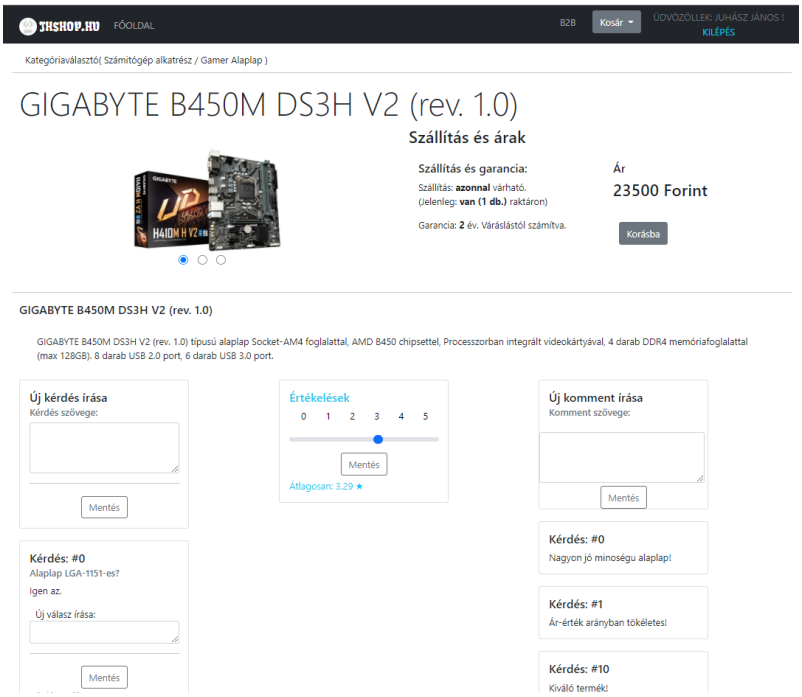
Az oldal számára egy olyan logikusan felépített kinézetet megtervezni, hogy azt az arra érkező felhasználó logikusan és könnyedén tudja használni. Továbbá ki kellett gondoljak egy olyan működési mechanizmust, amivel ugyanazt a terméklistát rendezni is tudom a-nélkül, hogy a termék típusok segítségével a listát újra lekérném. Nagy segítségemre volt a statikus mezőkről tanultak alkalmazása és az ASP.NET-es ViewData, TempData és ViewBag tulajdonságok.

Felhasznált technikák:

A felhasználó csak abban az esetben tud vásárolni, ha az be van jelentkezve. Ha nincs bejelentkezett felhasználó a program figyelmezteti egy hibaüzenet segítségével, hogy jelentkezzen be. A hibaüzenetet tartalmazó ablak alap értelmezetten rejtett, felhasználó számára láthatatlan. Bejelentkezetlen vásárlás esetén bukkan elő csak, ehhez a viselkedéshez JavaScript-et használok: Kosárba helyezés gombra kattintva a JavaScript a HTML Kártya megjelenítését láthatóra állítja, így a felhasználót figyelmezteti a bejelentkezésre.

7.3. Termék oldal

Számunkra megfelelő árucikk kiválasztása után (minden esetben ilyet a szűrési listából tudunk megtenni) a termék oldalra navigál minket el a program. Itt lehetőség nyílik arra, hogy részletesebben is áttekinthessük a terméket a vásárlás előtt. A termék oldalt a 18. ábra mutatja be számunkra.



17. ábra: Termék oldal

Minden termékről három fotó jeleníthető meg az oldalon (abból az első mindig az indexkép), a fényképek nagyíthatók és köztük lehetőség van a váltásra is. A program megjeleníti az aktuális raktárkészletet is. Ezt a kezelői személyzet minden termék esetén módosítani tudja. Amikor megvásárolnak egy terméket, a raktárkészlet csökken. A szállítás időtartamáról a rendszer a raktárkészlet függvényében dönt. A garancia időtartama minden termék esetében egyedi időtartamú, adatbázis adatokban meghatározható a garancia időtartama. A vásárlás pillanatában válik ez az időtartam aktívvá. Garanciális megrendeléseket egy erre külön létrehozott táblában tároljuk. Termék oldalon is lehetőség van a kosárba helyezésre. Ez a funkció a listázó oldalhoz hasonlóan, csak bejelentkezett felhasználó esetén működik, másként figyelmeztet a bejelentkezésre. Az oldal felépítése ebben az esetben is keretes szerkezetű.

Minden árucikkhez tartozik egy leírás is, amit az oldal szintén megjelenít. Azonban a speciális kereséshez szükséges tulajdonságait az adott árunak nem listázzuk ki.

Lehetőség van kérdéseket, értékelést és kommentet is fűznie a bejelentkezett felhasználónak az adott termékekhez. (Nem bejelentkezett felhasználó esetén ezek a funkciók csak olvashatóak.) Kérdés esetében lehetőség van új kérdés írására vagy meglévő kérdésre válaszolni. Új kérdés esetében a kérdés kap egy azonosító számot, a rendszer hozzáfűzi azt az adott termékhez és a kérést feltevő felhasználóhoz. Lehetőség

van amennyiben a bejelentkezett felhasználó a kérdés tulajdonosa, annak törlésére is. (Más esetben a kérdést a rendszer nem engedi a profilnak törölnie.) Választ egy adott kérdéshez hozzá tudunk fűzni. Több profilnak is lehetősége van egy kérdésre választ adnia. Kérdéshez hozzáfűzött választ nem tudunk törölni. Mentés gomb segítségével tudjuk a kérdést elküldeni vagy a választ hozzáfűzni.

Árucikkeket lehetőségünk van értékelni. Minden bejelentkezett felhasználó egy termékhez csak egy értékelést tud küldeni. Az értékelések leadása 0-5-ig terjedő skálán történik, csúszkás választással. Elküldeni azt a mentés gomb segítésével tudjuk. A profil amennyiben már adott le értékelést az adott termékhez, lehetősége van annak visszavonására is. Ezt a rendszer azzal tudatja, hogy a Mentés gomb Törlés gombra változik. A program egy átlagot számol minden árucikkhez, hogy tudassa a felhasználóval, hogy a többi felhasználó milyen értékelést fűzött már hozzá az adott áruhoz. Segíti a döntéshozatalt. Új értékelés leadásakor vagy meglévő visszavonásakor új átlag kerül kiszámításra.

Komment hozzáfűzéshez is módunk van. Segítségével megírhatjuk a véleményünket írásos formában is. Csak bejelentkezett felhasználó esetén működik ez a funkció is. Komment hozzáfűzésekor hasonlóan a kérdésekhez egy egyedi azonosító kerül a kérdéshez és a rendszer hozzárendeli a kommentelő profilt is. Ha a profilunk a kommentet feltevő profil, ebben az esetben lehetőségünk nyílik a komment eltávolítására. Mások kommentjeinek módosítására nincs lehetőségünk. Az oldalon bármennyi kérdés, és komment megjelenhet.

Felmerülő nehézség volt, hogy a képek váltogatását, belebegését Bootstrap segítségével nem tudtam megoldani, így ehhez CSS-t használtam fel. A feladat megoldása közben a későbbiekben észrevettem, hogy a kosár nem minden esetben viszi tovább, vagy őrzi meg az adatokat, így a süti készítést még egyszer át kellett gondoljam. A kosár törlésénél is hasonló gondok merültek fel, nem minden esetben törlődött a kosár. Ennek az oka is az elsőre rosszul elkészített HTTP-Cookie.

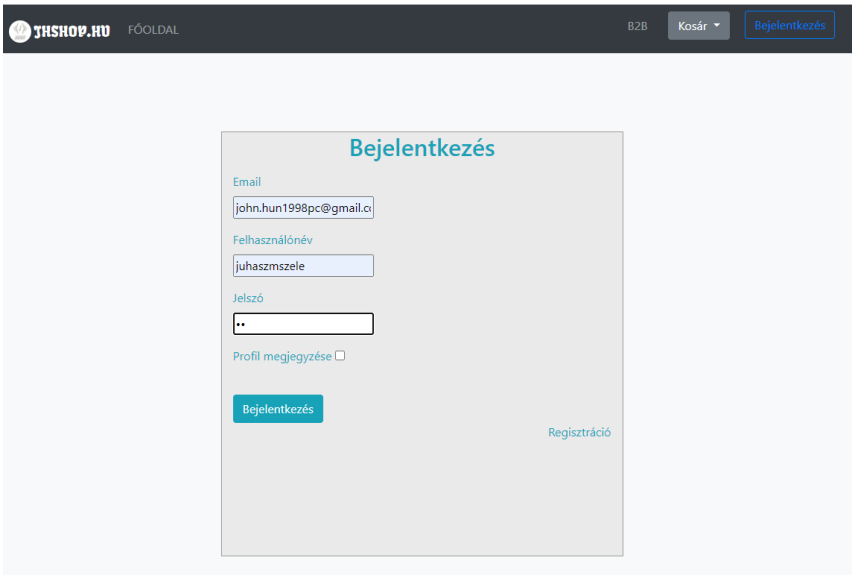
Felhasznált technikák közé tartozik a HTTP-Cookie, amit előszeretettel alkalmaznak a kosár elemeinek megőrzésére és bejelentkezés kezelésre is. A kosár elemeit a program 24 órán keresztül őrzi meg. A 3. számú mellékleten megtekinthetjük a süti készítését végző kódrészletet és a 4. számú mellékleten pedig az olvasását végző kódot.

Írása: Amennyiben a HTTP-Süti nem létezik, újat hozunk létre, kosár tartalma néven. Minden sütinek beállítok egy felhasználói azonosítót, hogy tudjuk melyik felhasználó hozta létre azt, majd hozzáfűzőm a benne lévő termékek azonosítóit egy szétválogatható szövegtípusként. (string split)

Olvasása: Abban az esetben csak, ha a Cookie létezik. Meghatározom a felhasználót, aki csinálta. Majd a kosárban lévő termékeket kódoló szövegtípust szétválogatom egy termékekből álló listába, különálló termékekre. Egy Sessiont használok arra, hogy a kosár tartalmát a továbbiakban az oldal megfelelően tudja kezelni. Hasonlóan vannak kezelve a programban azok a felhasználók, akik kérik a profiljuk elmentését bejelentkezést követően.

7.4. Bejelentkezési felület

Kulcsfontosságú felület, mivel a megrendelés elindítását és még sok más lehetőséget is csak bejelentkezett felhasználó tud az oldalon elintézni. Segítségével lehet bonyolítani a regisztrációt is. Megkülönbözteti a jogosultsággal rendelkező felhasználót a normáltól. A 19. ábra mutatja a Bejelentkező felületet.



18. ábra: Bejelentkező felület

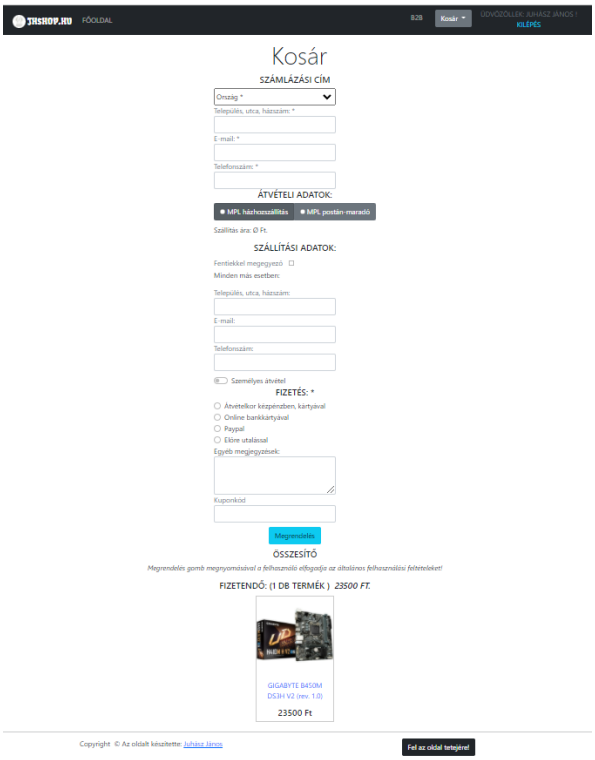
Az ábrán szereplő felületre a Bejelentkezés gomb vagy a program ajánlásainak a segítségével tudunk eljutni. Bejelentkezni e-mail cím, felhasználónév és jelszó megadását következően tudunk. Lehetőség van a profil megjegyzésére, ebben az esetben a rendszer Cookie segítségével a bejelentkezést 24 óráig megőrzi. Nélküle az oldal elhagyásáig marad meg a bejelentkezés. Bejelentkezést követően a rendszer a Főoldalra irányít minket vissza. Sikertelen bejelentkezés vagy nem létező felhasználó esetén hibaüzenetet kapunk. Lehetőség van nem létező felhasználó esetén regisztrációt készíteni. A regisztrációs felületet megtekinthetjük az 5. számú mellékleten. A bejelentkezés után az oldal címsora megváltozik: eltűnik a bejelentkezési oldalra irányító gomb és helyette megjelenik a bejelentkezett felhasználó (neve) és egy kijelentkezés gomb is, ami arra nyújt lehetőséget a felhasználónak, hogy kijelentkezzen.

A regisztráció során az új profilt személyes adatok megadásával lehet létrehozni. Az új felhasználónak meg kell adnia a nevét (vezeték és keresztnév) egy szabadon választott felhasználónevet, e-mail címét és jelszavát. A program Bootrap validációs attribútumok segítségével ellenőrzi a megadott adatokat és helyességüket. Jelszavat csak kétszeri helyes magadás után enged elfogadni. Sikeres regisztráció esetén ezt tudatja a felhasználóval és megkéri a bejelentkezésre. Jelszó típusú mezők esetében a karakterek rejtve maradnak (’*’ karakterek, adatbiztonsági okok).

A bejelentkezési felület a jogosultsággal rendelkező felhasználó esetén is a Főoldalra dobhat vissza, de azt újabb menüponttal is kiegészíti, aminek a segítségével van lehetőség a megrendelések, raktárkészlet szerkesztésére és nyomon követésére.

7.5. Megrendelési felület

Az árucikkeket azután, hogy kosárba helyeztük és úgy döntünk, hogy nem szeretnénk a kosárba újabbat tenni, meg is rendelhetjük csomagunkat. A megrendelést a cég felé elküldeni a kosárban elhelyezett „Megrendelés” gomb megnyomtatásával tudjuk elkezdeni. A gombra csak abban az esetben enged a program rákattintani, ha a kosárban van legalább egy termék. Kattintás után a 20. ábrán látható felületre jutunk el, a megrendelői felületre.



19. ábra: Megrendelői felület [2]

Az oldalon a megrendeléssel kapcsolatos információkat tudjuk a vállalat felé megadni. Az oldal alján megjelennek a kosárba tett termékek és az összesített ár. A mezők kitöltésekor a Bootstrap validátorokat használtam az adatok helyességének ellenőrzésére és arra, hogy a felhasználó a kulcsfontosságú mezőket valóban kitöltötte- e. A megrendelés véglegesítéséhez meg kell adnia a megrendelőnek a számlázási, átvételi és fizetési adatokat is, amik lehetnek: megrendelő országa (kötelező); irányítószám, település, utca, házszám (kötelező); e-mail cím (kötelező); telefonszám (kötelező). Átvételi adatokkal kapcsolatban pedig lehetőség van futárszolgálattal való házhozszállításra vagy személyes átvételre. A program minden esetben kalkulálja a szállítás díját, amelyek esetenként eltérőek (ingyenes, házhoz, postán maradó csomag). Az ár a szállítás függvényében változik. Szállítási adatok megadásakor lehetőség van olyan adatok magadására, amik a számlázási adatoktól eltérnek. Kitöltéskor lehetőség

[2] A 19. ábrán szereplő kép nagyított, hogy jobban tudja szemléltetni az oldal funkcionalitását egy képben. (nem kell görgetni)

van eltérő adatokat megadni, de egy kapcsoló megnyomásával a program a számlázási adatokkal kitöltheti, amennyiben a szállítási adatokkal megegyeznek. A szállítási adatok opcionálisak. A megrendelőnek a fizetési lehetőségek közül is választania kell kötelezően egyet. Fizetésre több féle fizetési mód közül választhatunk (személyesen, online bankkártyával, PayPal, előre utalással). Megrendeléshez egyéb megjegyzéseket is megadhatunk szöveges formátumban. Ezzel üzenve például a futárnak vagy a csomagolást végző személyzetnek. Lehetőség van kuponkód megadására is. Kuponkódok segítségével az árból a rendszer kedvezményt enged el, ami jóváírásra is kerül. Megrendeléshez kapcsolódhat kuponkód, aminek összegét a rendszer véletlenül sorsolja a megrendelést követően. Megrendelés gomb megnyomásával a megrendelés bekerül a megrendelések közé. A megrendelő ezzel együtt elfogadja a bolt felhasználási és adatkezelési feltételeit is. Innentől a bolt személyzete átveszi azt és teljesíti.

Az oldal működéséről:

Megrendelés gomb megnyomása után a megrendelés bekerül egy megrendeléseket tároló adatbázis táblába. Innen tudja majd a megrendelésekkel foglalkozó személyzet azt az alkalmazás segítségével megnézni, hogy milyen névre és címre, mely termékeket vásároltak meg, majd összekészíteni és becsomagolni a rendelést. A megrendelési listát csak admin joggal rendelkező felhasználó tudja megnézni. A bejelentkező oldal segítségével tudunk Admin nézetbe belépni. Ebben az esetben egy újabb menüpont jelenik meg a felhasználónak, amit a 6. számú mellékleten tekinthetünk meg. A bejelentkezési felület ebben az esetben mindjárt a kezelői felületre irányít bennünket.

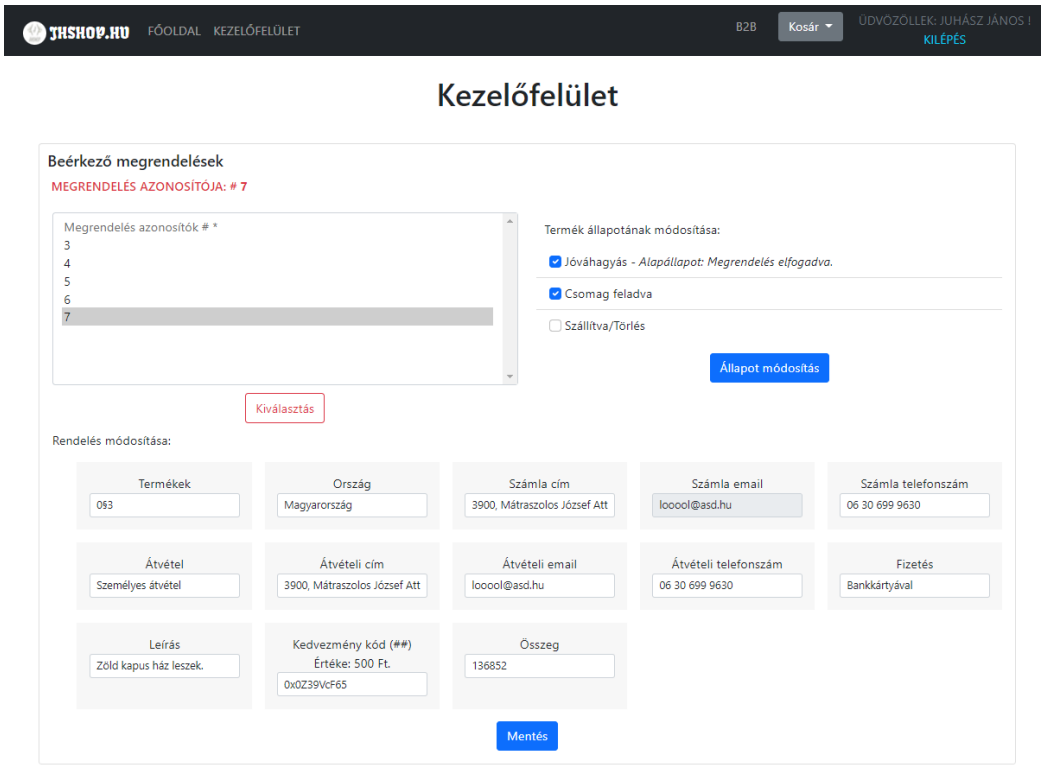
Felhasznált fejlesztési technikák, nehézségek:

Olyan megoldást kellett kigondoljak a megvalósításhoz, ami képes a lehető legegyszerűbben kezelni a megrendelésekhez kapcsolódó adatbázis táblákat. Ez nem volt minden esetben a legegyszerűbb, mivel rengeteg lehetőséget kellett számba vennem (ez adódik a beviteli mezők sokaságából, szállítási költségek megfelelő kezeléséből, stb.). A szállítási költségek esetében úgy éreztem, nem megfelelő megoldás, ha a felhasználó egyszerűen rákattint egy szállítási lehetőségre és azt ASP.NET-es kóddal valósítom meg az árhoz a szállítás hozzáadást, majd újra töltöm az oldalt. (Eddig kitöltött adatok elvesznének, szerver oldalon optimalizáltalan kód.) A felhasználói oldalon kellett megoldanom a szállítási költségek helyes hozzáadását az árhoz.

Fennálló problémára a megoldást a JavaScript jelentette: A termékek árához egy script segítségével hozzáadom a megfelelő szállítás árát. A megfelelő szállítás kiválasztásához a program segítségként, amikor rákattint a felhasználó a számára szimpatikus szállításra, kiírja annak az árát, majd hozzáadja a végső árhoz. HTML output element segítségével jelenítem meg a végső árat a felhasználónak. Alapértelmezés szerint a program nem jelenít meg árat a szállításra, de nem is hagyja a felhasználót továbblépni a szállítás kiválasztása nélkül. Ezzel a megoldással újratöltés nélkül tudom kiszámítani a szállítás árát, szerver oldalon ez egy optimálisabb megoldást eredményez és jobb felhasználói élményt biztosit a használójának.

7.6. A kezelői felület: Admin mód

Az oldal segítségével lehetőség van a megrendelések kezelésére, és a webshop szerkesztésére. Csak kiemelt felhasználók tudnak belépni a szerkesztői (admin) módba. (Jelenlegi megoldásom: regisztrált felhasználó esetén a felhasználónév-admin végződés.) A megrendelések teljesítéséhez, és a napi szintű aktualizáláshoz van az oldalnak szerepe. Amelyik oldalt nem kezelik, nem fogják a felhasználók elérni és szeretni, így elveszti a piacképességét, nem lesz sikeres a vállalkozás. A kezelő felületet megrendelésekre vonatkozó részét megtekinthetjük a 21. ábrán.



20. ábra: Kezelőfelület (megrendeléskezelés)

Lehetőség van a termékek és a cikkek szerkesztésére is. Az oldal szerkezetét tekintve a szerkesztő ablakok egymás alatt helyezkednek el. Megtekinthetjük a 7. számú mellékleten a teljes szerkesztői nézetet.

A görgethető listában megjelennek sorban a beérkező megrendelések. Minden megrendelésnek csak az azonosítóját látjuk. Alapértelmezetten nincs kiválasztva semmi. A „Kiválasztás” gomb segítségével tudok egy megrendelést kiválasztani. A megrendelésről az összes adat megjelenik (mely termékekről van szó, szállítási ország, számlázási cím, számlázási e-mail, telefonszám, átvételi lehetőség, átvételi cím, átvételi e-mail, átvételi telefonszám, fizetési mód, komment, promóciós kód, abban esetben, ha van). Mezők segítségével tudjuk megírni a címiratokat és a rendelést összekészíteni. Szerkesztésre van lehetőség, ezt befejezve a „mentés” gomb használatával tudjuk véglegesíteni. A megrendelések állapotát is tudjuk módosítani minden megrendelés esetében külön-külön. Alapértelmezés szerint minden megrendelés jóváhagyott, de ezt

különleges esetekben tudjuk módosítani (kifogyott raktárkészlet, stb.) A csomag összekészítése után feladásra kerül, erre szintén lehetőség van az alkalmazásba való bevitelre állapotmódosítással. Ha a megrendelőhöz az megérkezett a rendelést lehet törölni, megkezdődhet a garanciakezelés.

A további szerkesztési felületek esetében is (7. számú melléklet) hasonló módszer van alkalmazva a szerkesztés lehetőségének megoldására. Ezekben az esetekben nincs szükség állapot módosításra. Kiválasztás után a meglévőket tudjuk szerkeszteni és a Mentés gombbal menteni. Új termék, cikk esetében egy üres szerkesztési felületet kapunk. Adatok megadása után menteni tudjuk és így megjelenik az új cikk, termék.

Megrendeléskezelés:

A megrendelések állapotmódosításával a program áthelyezi másik táblába a módosított megrendelést. Így tudja megkülönböztetni egymástól őket (jóváhagyott, feladott, stb…). Teljesíthetetlen megrendelés esetében a IncompletedOrders táblába feltünteti a megrendelés azonosítóját, feladott termék esetében a megrendelésazonosító a ProcessedOrders táblába kerül. Kiszállított, már törlendő megrendelés törlése esetében a megrendelések közül a program nem töröli véglegesen a megrendelést, hanem elkezdődik a garanciakezelés, a garanciális megrendeléseket a WarrantyOrder táblában tárolom. A garancia lejártának időpontjáig. Az állapotjelző pipák segítésével a termékeket ki illetve be is lehet pakolni a táblákba, annak a függvényében, hogy a megrendelés állapota milyen. Az oldalon termékek közt váltani csak a kiválasztás után lehet. Esetleges továbbfejlesztési lehetőség lehet kliens oldali kóddal újra töltések nélkül megvalósítani a kiválasztást és a szerkesztést. Szerverrel csak abban az esetben kommunikálni, ha az adatbázist szerkesztenénk.

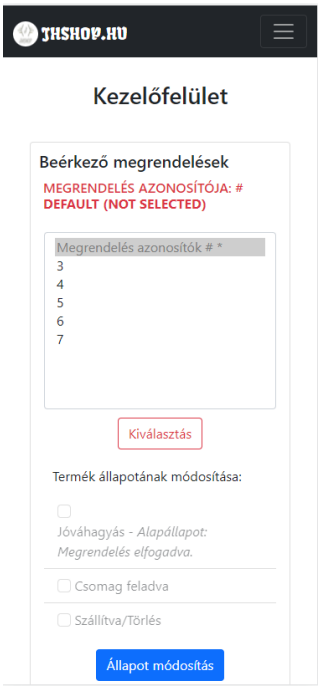
Az oldal működéséről:

Oldalra való érkezéskor egy oldalt előhívó kontroller lefut. Ez alap értékekkel felparaméterezi az oldalt, kiolvassa az adatbázisból a szükséges adatokat a rétegezés felhasználásával. Alap értékeket egy statikus osztályban tárolom. Kiválasztás után egy post metódus segítésével az adatbázisból a megfelelő „dolog” (pl. termék, cikk, megrendelés) adatait kiolvasom és ezt az alap értékek változtatásával és az oldal újra előhívásával változtatom. Abban az esetben, ha szerkeszteni vagy új „dolgot” is szeretnék kiválasztás után létrehozni, akkor ezt egy post metódusra hivatkozó Mentés gomb segítésével tudom megtenni. Ez a metódus kiolvassa és szerkeszti és menti a megfelelő sort. A frissített adatot pedig a felhasználó felé jelzi az oldal újra töltését követően. Törlésre is lehetőség van az oldalon. Ezt a kiválasztás után a „Törlés” jelölőnégyzet bepipálásával tudjuk megtenni. Megvalósítás során az azonosításkor figyelnem kellett a visszatérő azonosító típusára (szöveg, szám) és a C#-ban beépített TryParse metódus segítségével alakítanom azt a megfelelő számformátummá. A szükséges adatokat a HTML Request segítségével alakítom a számomra megfelelő formátummá. A cast-olást egy null ellenőrzés vizsgálat előzi meg, mivel lehetséges, hogy null értéket kap és így megelőzve a kód hibákat.

7.7. Utolsó simítások

Mobil nézet megvalósítása:

A frontend kinézet kialakításakor ügyeltem arra, hogy a weboldalon navigálva egy okos eszközön is élveztető tartalom jelenjen meg. Az oldalak okos eszközön (mobiltelefon, tablet, stb.) is ugyanazzal a funkcionalitásokkal és tartalommal jelenjenek meg, mint asztali számítógépen. A 21. ábra szemlélteti számunkra, hogy az Admin (szerkesztői) mód asztali számítógépen, hogyan néz ki. A 22. ábrán pedig megtekinthetjük ugyanezt az oldalt mobiltelefonon is (iPhoneX-en bemutatva). A többi oldal esetében is természetesen működik a mobil nézet. Legérdekesebb oldalakról a 8. számú mellékletben láthatunk pár példát.



21. ábra: Kezelőfelület reponsive nézete

Tesztelés kivitelezése:

A Logic réteg bizonyos metódusait teszteltem le, hogy el tudják- e látni a feladataikat. A megoldás során először egy mockolt respositoryt hoztam létre pár teszt adattal a termékekre vonatkoztathatóan. Legfontosabb publikus metódusom módosítással kapcsolatos. A 9. számú mellékleten láthatjuk a tesztkörnyezet (mockolt respository) létrehozását és a módosításért felelős metódus tesztelését. Működéséről röviden: A tesztelést NUnit Framework segítségével csináltam. A teszt adatokat pedig a Moq segítségével készítettem el. A tesztadatok kitaláltak, lényegük, hogyha a módosításért felelős metódust ezek segítségével tesztelem, akkor a változások a teszt adatokon mennek végbe és nem az éles környezeten. A tesztelés során először egy módosítást hajtok végre, majd kiolvasom az adott sort. Amennyiben minden sikeres, akkor a változtatások végbementek. (Az „id” módosításakor az megváltozott és azt az „id”-t olvasva így találunk elemet. Így a teszt sikeres.)

8. ÖSSZEFOGLALÁS:

Régóta érdekelt, hogyan is működhet egy weboldal vagy egy webshop „belülről”. Milyen kódok állhatnak a sikeres működésének hátterében, ezért választottam szakdolgozatom témájának. Részletesen utána kellett járnom a problémának, mivel még sosem csináltam ilyet eddig.

A megoldáshoz olyan programnyelveket próbáltam meg kiválasztani, amiket a tanulmányaim során használtam. A fejlesztőkörnyezetek választásánál is ez fontos szempont volt. Választásom így a C# programnyelvre és a Visual Studio 2019-es programra esett. Adatbázisokat a szintén tanult fejlesztőkörnyezettel az MSSQL Server Management Studio segítségével készítettem el.

Dolgozatom során részleteztem a projekt rétegzett felépítését, a frontend és backend oldalt egyaránt. Áttekinthettük a frontend fejlesztésre szolgáló nyelveket és ezek projektben történő alkalmazását. Backend oldalon pedig az egyes rétegek működését és feladatait. Ismertettem az adatmodellt is.

A frontend oldal a C# ASP.NET nyelven MVC fejlesztési architektúra segítségével készült el. A nézetek kialakításakor figyelembe vettem a változó ügyféli igényeket és a kor követelményeit is. A fejlesztéséhez a HTML, CSS, JavaScript és Bootstrap leíró nyelveket használtam fel. A frontend kód mobil eszközökön is jól megjelenik. Ez egy nagyon fontos tényező, mivel manapság már mindenki asztali számítógép helyett már mobiltelefonokat, tableteket használ.

A program egy olyan rendszert nyújt a dolgozók felé, amivel a megrendeléseket nyomon is tudják követni. Megkülönböztet „normál” felhasználókat és admin jogúakat. A webshop megjelenése pedig attól függ, hogyan is lépünk be.

A weboldal DEMO jelleggel készült el, egy fiktív cég igényére. A benne szereplő adatok kitaláltak. Éles használathoz tökéletesíteni kell a megrendelés véglegesítését: beiktatni a fizetési lehetőségeket. A vállalat jövőbeni céljainak eléréséhez készült a webshop. Kialakításkor fontos szerepet játszott az egyszerű használhatóság. A felhasználók elégedettsége nagy sikert jelent a jövőbeni szempontok eléréséhez.

A weboldallal kapcsolatos továbbfejlesztési lehetőségeket is bemutattam. Legfontosabb új irányvonal lehet az ügyfélorientáltság és a szerver oldali optimalizálás lehetősége. Az elkészült projektet meg is valósítottam, aminek működését bemutattam a dolgozat második felében. Webshop éles szerverre történő helyezésekor leírtam a lehetőségeket az oldal üzemeltetői számára az üzembe helyezéshez. Szükség van egy webszerverre és egy adatbázis rendszerre. Oldalanként leírtam a funkcionalitásokat és az adott oldalon használt fejlesztési technikákat és felmerülő nehézségeket.

Következtetésképpen a dolgozat hatalmas szakmai fejlődést jelentett a számomra. A megoldás során több új technikával is megismerkedhettem (HTTP-Cookie, HTML leíró nyelv, frontend fejlesztés), amiket eddig nem használtam. Részletesebben betekintést nyertem egy komplex weboldal működésébe. Úgy érzem későbbiekben is tudom majd kamatoztatni a dolgozat megírásával szerzett tudásomat.

9. SUMMARY:

I have long been interested in how a website or webshop can work “from the inside”. What codes can underlie the successful operation, so in this reason I chose this topic of my dissertation. Before the beginnings of the work I had to read some articles about the problem, because I have never done this before.

For the solution, I tried to choose the programming languages what I used in my studies. This was also an important consideration when selecting development environments. So my choice was for the C # programming language and Visual Studio 2019 enviorment. I created databases with the also learned development environment: MSSQL Server Management Studio.

During the dissertation I detailed the layered structure of the project, both the frontend and backend side. We were able to review the languages used for frontend development and their application in the project. On the backend page, we were able to review the operation and tasks of each layer. I also introduced the data model.

The frontend page was created in C # ASP.NET using MVC development architecture. When formulating the views I also consider the changing needs of the client and the requirements of the age. I used HTML, CSS, JavaScript and Bootstrap description languages to develop it. The frontend code also displays well on mobile devices. This is a very important factor since nowadays everyone has a desktop computer, but peoples already use mobile phones and tablets.

The program provides employees with a system that allows them to track orders. Distinguishes between "normal" users and admin users. And the appearance of the webshop depends on how we logged in.

The website has been created in demo format, for a fictitious company. The data which is displayed in website is fictional too. For live use, you need to consummate your order finalization: for example payment options. The webshop was created to achieve the company's future goals. Ease of use played an important role in design. Satisfied users means the great success of achieving future aspects.

I also presented the possibilities for further development of the website. The most important new direction may be customer orientation and the server-side optimization. I also implemented the completed project: the operation of the project is presented in the second half of the dissertation. When deploying a webshop to a live server, I described the options for site operators to deploy it. It needs a web server and database system.

In conclusion, the dissertation was a huge professional development for me. During the solution, I was able to learn about several new techniques (HTTP-Cookie, HTML markup language, frontend development) that I have not used so far. I gained more detailed insight into the operation of a complex website. I feel that I will be able to use the knowledge in the future what I gained by writing the dissertation.

10. IRODALOMJEGYZÉK

[1] „Stackoverflow statement,” 03 12 2020. [Online]. Available: https://insights.stackoverflow.com/survey/2018/.

[2] L. Naylor, ASP:NET MVC with Entity Framework and CSS, Apress, 2016.

[3] A. Fowler, NoSQL for DUMMIES, John Wiley & Sons, Inc.

[4] A. Molinaro, SQL cookbook [query solutions and techniques for database developers ; covers SQL server, PostgreSQL, Oracle, MySQL, and DB2], O'Reilly, 2011.

[5] J. Galloway, ASP.NET MVC Music Store Tutorial, Microsoft, 2011.

[6] D. R. Prasanna, Dependency injection: design patterns using spring and guice, Manning, 2009.

[7] „Wikipedia, the free encyclopedia,” [Online]. Available: https://en.wikipedia.org/wiki/Main_Page.

[8] J. Price, C# adatbázis programozás mesteri szinten, C# adatbázis programozás mesteri szinten, 2003.

[9] Microsoft, „Microsoft Official Page,” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp. [Hozzáférés dátuma: 03 12 2020].

[10] J. M. Benjamin Jakobus, Mastering Bootstrap 4, Packt Publishing, 2006.

[11] E. Tittel, HTML, XHTML & CSS For Dummies, Wiley, 2011.

[12] J. Bishop, C# 3.0 Design Patterns, O’Reilly Media Inc., 2008.

[13] „AutoMapper,” [Online]. Available: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller. [Hozzáférés dátuma: 10 03 2021].

[14] S. F. Guy Harrison, MySQL Stored Procedure Programming, O’Reilly Media, 2009.

[15] „OE-UML:,” [Online]. Available: https://users.nik.uni-obuda.hu/prog4/Eloadas_HU/P4HU_EA_04_05_Uml.pptx. [Hozzáférés dátuma: 15 04 2021].

[16] S. Zaal, Migrating Applications to the Cloud with Azure, Packt Publishing, 2019.

[17] V. Lakshmanan, Data Science on the Google Cloud Platform, O’Reilly, 2018.

[18] M. Khosrow-Pour és M. Khosrowpour, Encyclopedia of e-commerce, e-government, and mobile commerce, Idea Group Reference, 2006.

[19] R. C. Martin, Agile Software Development, Alan R Apt, 2003.

[20] C. Hailmann, Begining JavaScript with DOM Scripting and Ajax, Apress, 2006.

[21] L. Beighley, jQuery for Dummies, Wiley, 2010.

[22] Google, „Google Webp," Google, 2021. [Online]. Available: https://developers.google.com/speed/webp. [Hozzáférés dátuma: 20 11 2021].

[23] Google, „Google fonts," Google, 2021. [Online]. Available: https://fonts.google.com/about. [Hozzáférés dátuma: 18 10 2021].

[24] A. S. Tanenbaum, Számítógép-hálózatok, Panem Könyvek, 2011.

[25] S. S. J. S. R. F. Eric Enge, The Art of SEO Mastering Search Engine Optimalization, O'Reilly, 2012.

[26] „Open graph protocol," [Online]. Available: https://ogp.me/ open. [Hozzáférés dátuma: 15 11 2021].

[27] w3schools, „ASP.NET Cookies," [Online]. Available: https://www.w3schools.com/asp/asp_cookies.asp.

[28] w3schools, „ASP.NET Sessions," [Online]. Available: https://www.w3schools.com/asp/asp_sessions.asp. [Hozzáférés dátuma: 10 10 2021].

[29] „ASP.NET ViewData, TempData, ViewBag," [Online]. Available: https://github.com/ablealias/MVC/wiki/ViewBag,--ViewData-and-TempData. [Hozzáférés dátuma: 10 11 2021].

[30] w3schools, „ASP.NET RAZOR View Engine," [Online]. Available: https://www.w3schools.com/asp/razor_intro.asp. [Hozzáférés dátuma: 14 11 2021].

[31] L. András, eBusiness stratégia vállalai felsővezetőknek, AUla, 2002.

[32] I. Zoltán, Programozás C# nyelven, JEDLIK OKTATÁSI STÚDIÓ, 2005.

[33] T. Samara, ERP and Information Systems, Wiley, 2015.

11. ÁBRAJEGYZÉK:

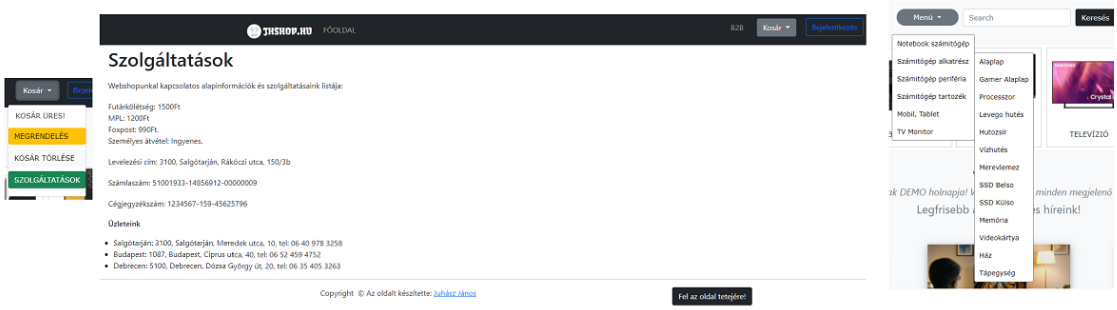
Az ábrákat a https://app.diagrams.net/ (draw.io) felhasználásával készítettem. Itt lehetőség van az egyszerű, gyors és módosítható szerkesztésre, amit a megvalósítás során sokszor használtam:

Ábrák jegyzéke:

Táblázatok jegyzéke:

12. MELLÉKLETEK:

1. számú melléklet: Az üres kosár és kinézete, a webshop szolgáltatásait tartalmazó oldal és a lenyíló Főmenü (Számítógép alkatrész menüje és almenüjei).



2. számú melléklet: Híroldal (cím, alcím, szöveg, és borító kép is megjelenik)

3. számú melléklet: HTTP-Cookie, ami a kosár elemeit tárolja 24 óráig (írás)

```
HttpCookie cookie = Request.Cookies["cartcontents"];
if (cookie == null || Request.Cookies["cartcontents"]["userID"] != Session["userid"].ToString())
{
    cookie = new HttpCookie("cartcontents");
    cookie["userID"] = Session["userid"].ToString();
    cookie["productID"] = ap.AvaliableProductID.ToString();
    cookie.Expires = DateTime.Now.AddHours(24);
}
else
{
    cookie["productID"] = Request.Cookies["cartcontents"]["productID"] + "§" + ap.AvaliableProductID.ToString();
}
```

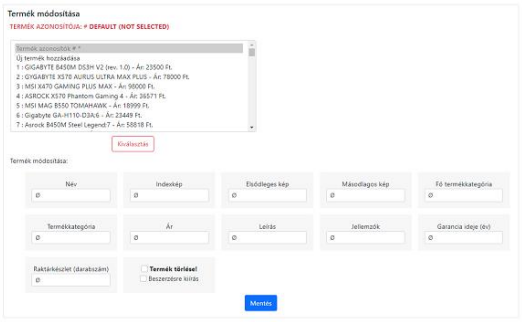
4. számú melléklet: Kosár tartalmát megőrző HTTP-Cookie olvasása, amennyiben létezik

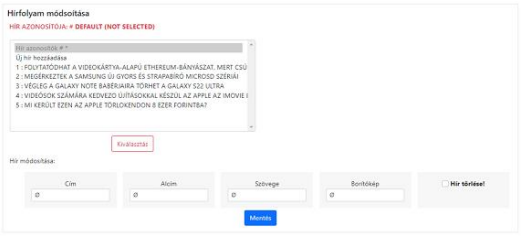
```
if (Request.Cookies["cartcontents"] != null)
{
    ID = Request.Cookies["cartcontents"]["userID"];
    string helper = Request.Cookies["cartcontents"]["productID"];
    prods = helper.Split('§');
    List<DiplomaMunka_Repository.DB.AvaliableProduct> aps = new List<DiplomaMunka_Repository.DB.AvaliableProduct>();
    foreach (var id in prods)
    {
        DiplomaMunka_Repository.DB.AvaliableProduct p =
            logic.AvaliableProductGetAll().Where(x => x.AvaliableProductID.ToString() == id).First();
        aps.Add(p);
    }

    Session["cart"] = aps;
}
```

5. számú melléklet: Regisztrációs felület és az oldal megváltozott címsora a bejelentkezést követően (bejelentkezés előtt és utána)

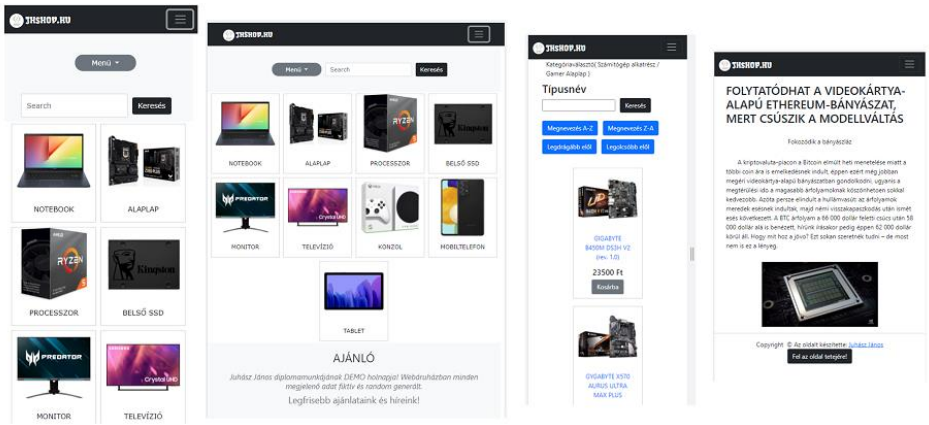
6. számú melléklet: Admin módú belépés után újabb menüpont jelenik meg

7. számú melléklet: Kezelő felület termékekre és cikkekre vonatkozó szerkesztői felülete





8. számú melléklet: Mobil nézet (responsive) a weboldalon.



9. számú melléklet: A mockolt repository (tesztkörnyezet) és a teszt

```
[SetUp]
0 references | Johnhun1998, 25 days ago | 1 author, 1 change
public void Init()
{
    var mockedrepo = new Mock<IRepository<AvaliableProduct>>();

    List<AvaliableProduct> avaliableProducts = new List<AvaliableProduct>();
    avaliableProducts.Add(new AvaliableProduct()
    {
        AvaliableProductID = 500,
        AvaliableProductName = "valami",
        AvaliableProductImageIndexLocation = "path",
        AvaliableProductImagePrimaryLocation = "path",
        AvaliableProductImageSeccundaryLocation = "path",
        AvaliableProductMainProductID = 1,
        AvaliableProductPrioductID = "1",
        AvaliableProductFeed_IDs = "52",
        AvaliableProductQuestions_IDs = "90",
        AvaliableProductComment_IDs = "22",
        AvaliableProductPrice = 5000,
        AvaliableProductDescription = "Valami leírás lenne ez itt.",
        AvaliableProductSpecDescription = "Specifikusan \t is leírom.",
        GaranteeTimeInYears = 2,
        InAStock = "1",
        StockCount = "3"
    });
    mockedrepo.Setup(m => m.AvaliableProductGetAll())
       .Returns(avaliableProducts.AsQueryable());

    this.logic = new MainLogic(mockedrepo.Object);
```

```
[Test]
0 references | Johnhun1998, 25 days ago | 1 author, 1 change
public void UpdateTest()
{
    logic.AvaliableProductUpdate(
        500,
        new AvaliableProduct()
        {
            AvaliableProductID = 570,
            AvaliableProductName = string.Empty
        });

    var AvaliableProductRead = this.logic.AvaliableProductRead(570);

    Assert.That(
        this.logic.AvaliableProductRead(570) == AvaliableProductRead);
}
```