



Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

Gaál Réka Dorottya

SZOFTVERFEJLESZTÉS .NET PLATFORMON

WEBFEJLESZTÉS .NET BLAZOR KERETRENDSZEREN MATBLAZOR KÜLSŐ KOMPONENS

KÖNYVTÁR SEGÍTSÉGÉVEL

KONZULENS:

DR. SIMON BALÁZS

BUDAPEST, 2022

Tartalomjegyzék

Összefoglaló	4
A megvalósított webalkalmazás leírása	5
1 Frontend modulok	8
1.1 Pages	8
1.1.1 Cart.....	8
1.1.2 Home.....	8
1.1.3 IceCream.....	9
1.1.4 Login.....	9
1.1.5 Product.....	9
1.1.6 Profile.....	10
1.1.7 SignUp.....	10
1.2 Shared	10
1.2.1 MainLayout.....	10
2 BackEnd modulok.....	11
2.1 Pages	11
2.1.1 Home.razor.cs – public partial class Home	11
2.1.2 IceCream.razor.cs – public partial class IceCream.....	11
2.1.3 Login.razor.cs – public partial class Login.....	12
2.1.4 Product.razor.cs – public partial class Product.....	12
2.1.5 SignUp.razor.cs – public partial class SignUp	12
2.2 Shared	13
2.2.1 MainLayout.razor.cs – public partial class MainLayout	13
2.3 További modulok.....	14
2.3.1 Allergen.cs – public class Allergen	14
2.3.2 Card.cs – public class Card.....	14
2.3.3 CartElement.cs – public class CartElement.....	15
2.3.4 GenderDef.cs – public class GenderDef.....	15
2.3.5 Global.cs – public static class Global	15
3 Felhasznált MatBlazor komponensek.....	17
3.1 Felugró értesítések	17
3.1.1 SnackBar.....	17

3.2 Form Control-ok	18
3.2.1 Checkbox	18
3.2.2 DatePicker.....	19
3.2.3 NumericUpDownField.....	20
3.2.4 Select.....	21
3.2.5 Slider.....	22
3.2.6 TextField.....	22
3.2.7 Validation via EditContext	23
3.3 Gombok	25
3.3.1 Button.....	25
3.3.2 Icon	25
3.3.3 IconButton	26
3.4 Layout-ok.....	27
3.4.1 Card.....	27
3.4.2 Divider	28
3.4.3 Layout Grid.....	29
3.4.4 List	30
3.4.5 Paper & Elevation.....	31
3.4.6 Themes.....	32
3.4.7 Typography	33
3.5 Navigáció	33
3.5.1 AppBar.....	33
3.5.2 ButtonLink.....	34
3.6 Táblázatok.....	35
3.6.1 Table	35

Összefoglaló

A feladat egy .NET Blazor alapú weboldal elkészítése volt a MatBlazor külső komponens könyvtár minél széleskörűbb felhasználásával és tesztelésével. Ennek érdekében egy Webshophoz hasonló webes felületet valósítottam meg, ahol főleg a komponensek változatos felhasználására törekedtem a funkciók tényleges megvalósítása helyett. A Webshop fagyaltokat árul és a vásárláshoz és a felhasználókezeléshez tartozó oldalakat igyekeztem külsőleg elkészíteni, amik magába foglalják az oldalak közti navigációt és az oldalon belüli interaktivitások lehetőségét. Ehhez a Visual Studio által biztosított C# nyelvű Blazor WebAssembly App projekt sablonból indultam ki 5.0-ás .NET verzióval és ezen hajtottam végre a szükséges módosításokat és kiegészítéseket.

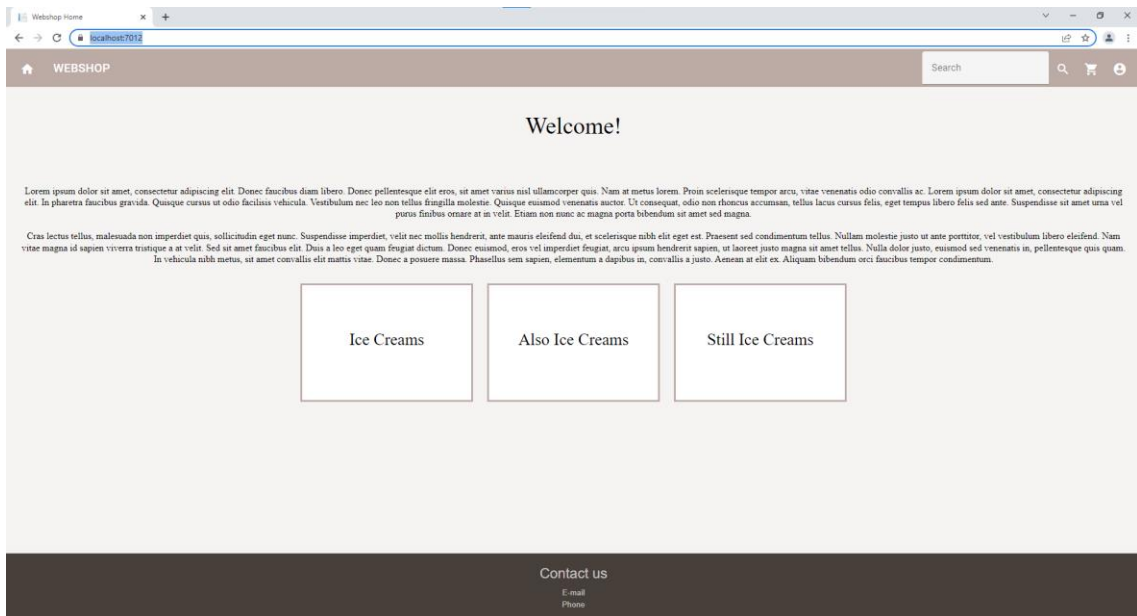
A webalkalmazás rengeteg kiaknáztatlan lehetőséget rejt még magában, amikkel a tárgy keretein belül nem volt alkalmam foglalkozni, de maga a keretrendszer lehetővé teszi, hogy egy teljes értékű és tökéletesen működő webshopot írjon a segítségével az elegendő tudással rendelkező fejlesztő.

Összességében úgy érzem, hogy a feladatból rengeteget tudtam tanulni és fejleszteni magamat. Egy teljesen új technológiával ismerkedhettem meg, ami nem csak korszerű, de rengeteg lehetőséget is rejt magában a fejlesztők számára. Korábban nem fejlesztettem webes felületet .NET platformon, de a Blazor egy olyan keretrendszert biztosít ehhez, ami nagyban megkönnyítette és élvezhetővé tette a tanulást, a komponensekben mutatott hiányosságokat pedig széles körben tudtam pótolni a hozzá elérhető külső komponenskönyvtárak, azon belül is a fent említett MatBlazor segítségével.

Ezen pozitív tapasztalatok birtokában és a konzulensemtől, Dr. Simon Balázstól kapott segítségnek köszönhetően egy tartalmas félévet tudhatok magaménak az Önálló laboratórium tárgy keretein belül és szívesen fogom bővíteni a későbbiekben is az itt megalapozott tudásomat a témakörben.

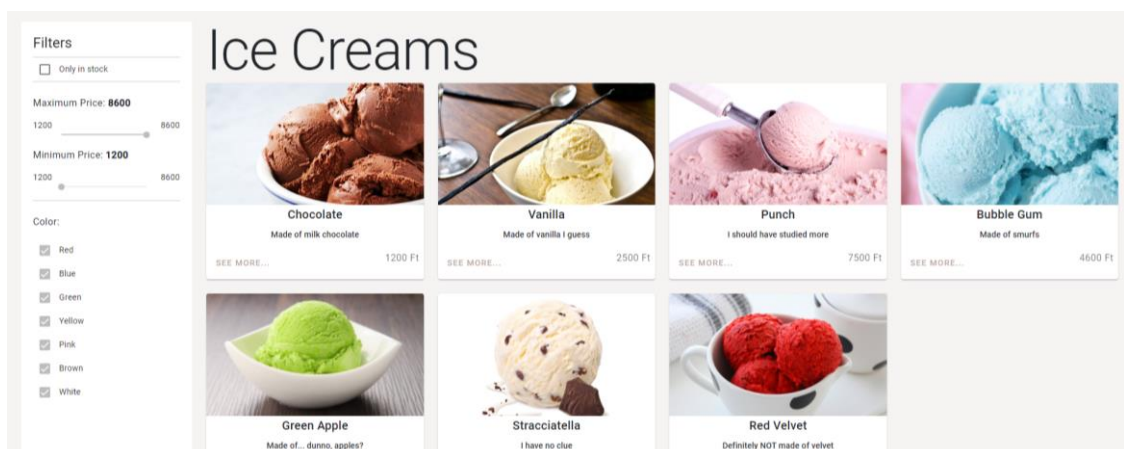
A megvalósított webalkalmazás leírása

Az oldal megnyitásakor a felhasználó egy home oldalon találja magát. Innen navigálhat tovább a termékkategóriákra, esetleg a keresősáv segítségével konkrét termékekre, a kosarára vagy a bejelentkező felületre, bejelentkezést követően pedig már egyből a saját profiljára.



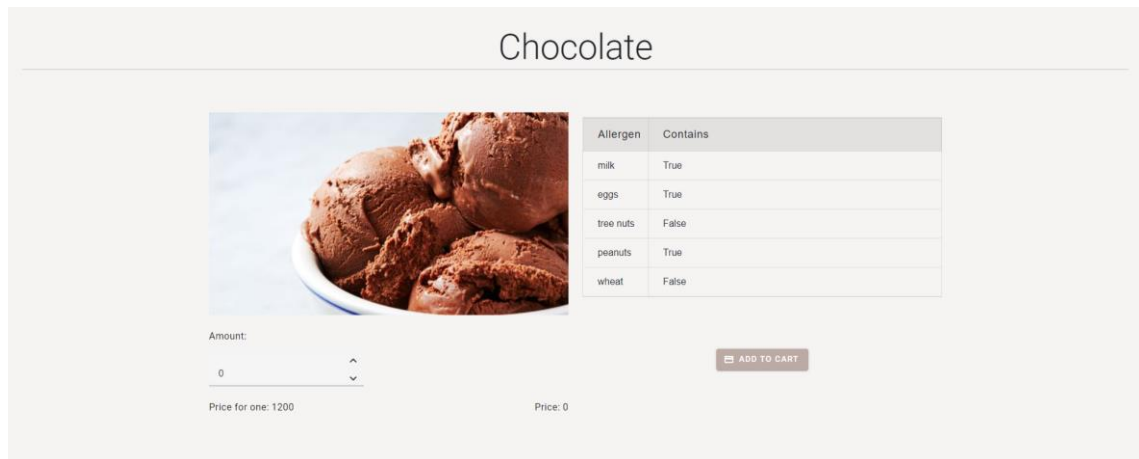
1. ábra: Home Page

Egy termékkategóriát kiválasztva egy gyűjtő oldal jelenik meg a felhasználó előtt, ahol bal oldalt találhatóak a szűrők, jobb oldalt pedig a termékek és az azokhoz tartozó adatok, a név, leírás, ár és egy gomb, amivel az oldalukra lehet navigálni.



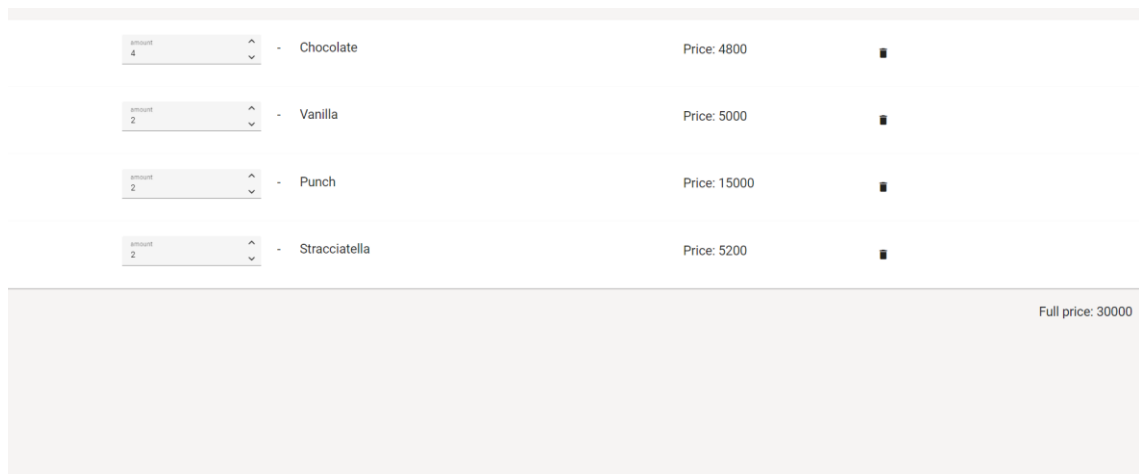
2. ábra: Termékek oldala

Egy terméket kiválasztva az oldal tetején megjelenik a termék neve, alatta elválasztva pedig a termék képe, az allergén táblázat és a vásárlási felület. Kiválaszthatja a felhasználó a megvásárolni kívánt mennyiséget és hozzá is adhatja a kosarához.



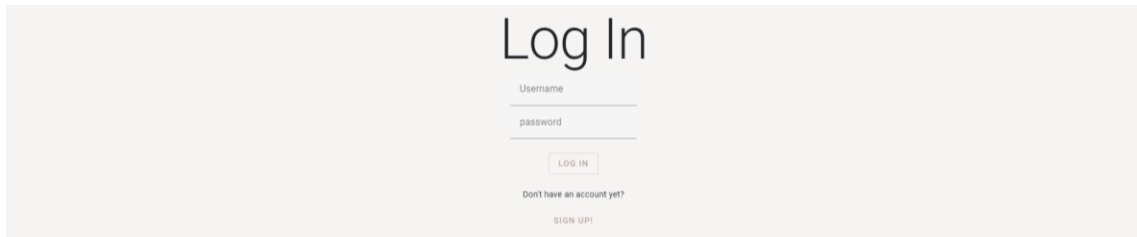
3. ábra: Termékek önálló oldala

A kosárra navigálva láthatjuk a kosarunkban található termékek neveit és mennyiségét, valamint az ehhez tartozó árat. A megvásárolni kívánt mennyiség még itt is állítható és termékek teljesen el is távolíthatók a kosárból. A lista alatt megtalálható a teljes fizetendő összeg.



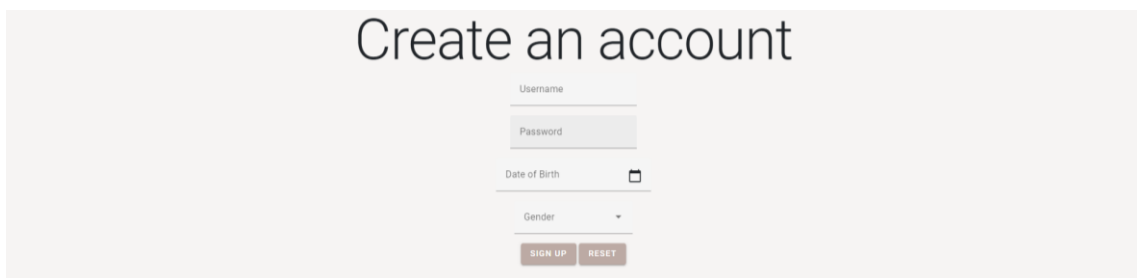
4. ábra: Kosár

A bejelentkező felületen egy egyszerű kép tárul a felhasználó szeme elé, csupán az oldal címe és a két beviteli mező, egy a felhasználónévnek és egy a jelszónak. Alattuk fel van kínálva a regisztráció lehetősége.

A screenshot of a 'Log In' form. At the top, the text 'Log In' is displayed in a large, dark font. Below it, there are two input fields: 'Username' and 'password'. A 'LOG IN' button is positioned below the password field. At the bottom of the form, there is a link that says 'Don't have an account yet?' followed by a 'SIGN UP!' link.

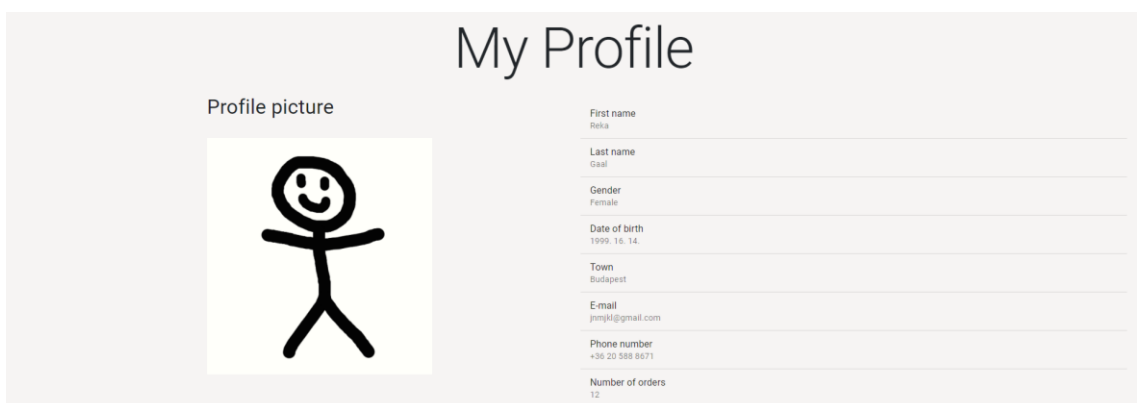
5. ábra: Bejelentkezés

A regisztrációs oldal a bejelentkezéséhez hasonló, beviteli mezők találhatók rajta, azonban többféle típusú adatra és csak megfelelően kitöltött mezők esetén fogunk tudni regisztrálni.

A screenshot of a 'Create an account' form. The title 'Create an account' is at the top in a large, dark font. Below it, there are four input fields: 'Username', 'Password', 'Date of Birth' (with a calendar icon), and 'Gender' (with a dropdown arrow). At the bottom, there are two buttons: 'SIGN UP' and 'RESET'.

6. ábra: Regisztráció

Végezetül sikeres bejelentkezés után elérhető még a saját profilunk oldala. Itt a profilkép és különböző személyes adatok találhatók meg.

A screenshot of a 'My Profile' page. The title 'My Profile' is at the top in a large, dark font. On the left, there is a section labeled 'Profile picture' with a placeholder image of a stick figure. On the right, there are several input fields for personal information: 'First name' (with the value 'Rita'), 'Last name' (with the value 'Gaal'), 'Gender' (with the value 'Female'), 'Date of birth' (with the value '1999. 10. 14.'), 'Town' (with the value 'Budapest'), 'E-mail' (with the value 'jonyk@gmail.com'), 'Phone number' (with the value '+36 20 588 8671'), and 'Number of orders' (with the value '12').

7. ábra: Profil oldal

Bármelyik oldal alján és tetején megtalálhatók a fejléc és a lábléc, amik a navigációs lehetőségeket adják és a kapcsolattartási információkat.

1 Frontend modulok

Az általam fejlesztett frontend modulok felsorolása és rövid ismertetésük. Az egyes modulokhoz tartozó .razor file-ok tartalmazzák a megjelenítésért felelős HTML kódot és a .css file-ok tartalmazzák a HTML kódban felhasznált saját stílusokat.

1.1 Pages

1.1.1 Cart

Ez az osztály valósítja meg a kosárban található elemek megjelenítését. Tartozik hozzá egy .razor és egy .css file.

.razor: Kosárelemenként áll egy Paper hátterű sorból, ami tartalmazza az elem mennyiségét állítható módon, nevét, árát és egy eltávolítás gombot. Legalul jobb oldalt látható a teljes ár.

.css: Található benne egy formázás az árak megjelenítésére és a teljes összeg megjelenítésére.

1.1.2 Home

Ez az osztály valósítja meg az üdvözlő oldal megjelenítését. Tartozik hozzá egy .razor, egy .css file.

.razor: Egy négyzetrácsos elrendezésben tartalmaz egy üdvözlő szöveget, alatta az oldalhoz tartozó leírást, legalul pedig a termékkategóriákat tartalmazó három kártyaként formázott div blokk.

.css: Tartalmaz egy stílust az üdvözlő szövegre, az oldal leírását tartalmazó szövegre és itt található a három termékkategóriát tartalmazó blokk formázása is.

1.1.3 IceCream

Ez az osztály valósítja meg a fagyik listájának megjelenítését egy szűrési felülettel együtt. Tartozik hozzá egy `.razor` és egy `.css` file.

.razor: Egy négyzetrácsos elrendezés bal oldalán találhatók a szűrési feltételek, jobb oldalt pedig a fagyikat tartalmazó kártyák. A szűrők közt található egy checkbox csak a készleten lévő elemek megjelenítésére. Alatta található két csuszka a minimum és a maximum ár beállítására dinamikusan. Legalul végezetül megtalálhatók a fagyik színére szűrő checkbox-ok. Bal oldalt a kártyákat egy foreach ciklus jeleníti meg, minden fagyira egy új cellát hoz létre és elhelyezi benne a fagyik adataival kitöltött kártyát.

.css: Tartalmaz egy stílust a szűrőket tartalmazó cellára, a kártyákra és egy stílust, amit akkor alkalmazunk egy kártyán, ha a szűrési feltételek alapján azt nem szeretnénk megjeleníteni.

1.1.4 Login

A bejelentkező felületet megvalósító osztály. Egy `.razor` file-ból áll.

.razor: Az oldal címe alatt található a két beviteli mező a felhasználónévnek és a jelszónak. Ez alatt található a bejelentkezés gomb, ami alatt a regisztrációs oldalra irányító gomb, amennyiben a felhasználó még nem rendelkezik fiókkal. Ha a felhasználó rossz felhasználónévvel vagy jelszóval próbál bejelentkezni, egy snackbar üzenet figyelmezteti erre.

1.1.5 Product

Az egyes fagyikat egyenként megjelenítő rendelési felület. Tartozik hozzá egy `.razor` és egy `.css` file.

.razor: A termék neve alatt két oszlopban jeleníti meg a termék információkat és a rendelési felületet. Bal oldalt látható a termék képe, alatta pedig a rendelési mennyiség és annak az ára. Jobb oldalt található az allergén táblázat és a kosárba tétel gomb. Amennyiben a termék nincs készleten, a gomb helyett a Sold out felirat jelenik meg.

.css: A kép szélességét megadó stílus.

1.1.6 Profile

A felhasználói profilt megjelenítő oldal. Egy .razor file-ból áll.

.razor: Az oldal címe alatt két oszlop található. Bal oldalt látható a profilkép , jobb oldalt pedig a felhasználó adatai egy kétsoros listában vonallal elválasztva.

1.1.7 SignUp

A regisztrációs oldalt megjelenítő modul. Egy .razor file-ból áll.

.razor: Az oldal címe alatt találhatók egymás alatt sorban a beviteli mezők: felhasználónév, jelszó, születési dátum kiválasztása, nem. A beviteli mezők alatt található két gomb. Az egyikkel regisztrálni lehet, a másikkal pedig kiüríteni a már kitöltött mezőket. Végezetül található a file alján egy validációs kód, ami vizsgálja, hogy minden mező ki van-e töltve, a jelszó legalább 8 karakteres-e és a születési dátum nem későbbi-e, mint a mai dátum. Ehhez tartozik egy SignUpModell osztály, ami a beviteli mezők változóit adja meg követelményekkel együtt. Ha sikeres a regisztráció, megjelenik az üzenet, hogy most már be lehet jelentkezni.

1.2 Shared

1.2.1 MainLayout

Az oldal keretét megvalósító modul. Ebben található a fejléc és a lábléc. Tartozik hozzá egy .razor és egy .css file is.

.razor: A header tag-ben található egy téma beállítás, ami az alkalmazás színeit Htározza meg. Ezután egy appbar biztosítja az oldal fejlécét. Ezen található a home gomb és az oldal neve balra igazítva, majd egy keresősáv, kosár és profil gomb jobbra igazítva. Ezután található a body, ahová a különböző oldalak layout-jai kerülnek navigációtól függően. Végezetül található még a keret alján egy lábléc a különböző elérhetőségekkel. A kód legalján található a sikertelen keresés esetén megjelenő snackbar.

.css: Található benne formázás a body-ra, a láblécre és a helykitöltő panelre.

2 BackEnd modulok

Az általam írt backend modulok felsorolása és rövid ismertetése. Az oldal hátterét C# kód adja, az ezen a nyelven írt osztályokat, változókat és függvényeiket mutatom be.

2.1 Pages

2.1.1 Home.razor.cs – public partial class Home

Változók:

- **szoveg1: string** – A leírást tartalmazó szöveg első fele.
- **szoveg2: string** – A leírást tartalmazó szöveg második fele.

Függvények:

- **onCategory1(): void** – Az első kategória kiválasztása esetén a fagyik oldalára irányít.
- **onCategory2(): void** – A második kategória kiválasztása esetén a fagyik oldalára irányít.
- **onCategory3(): void** – A harmadik kategória kiválasztása esetén a fagyik oldalára irányít.

2.1.2 IceCream.razor.cs – public partial class IceCream

Változók:

- **onlyInStock: bool** – Be van-e pipálva a checkbox, hogy csak a készleten lévő elemeket jelenítsük meg.
- **setMax: int** – A beállított maximum ár.
- **setMin: int** – A beállított minimum ár.
- **minPrice: int** – A legkisebb elérhető ár.
- **maxPrice: int** – A legnagyobb elérhető ár.
- **checkboxes: bool[]** – A színekhez tartozó checkboxok közül melyik van bepipálva.

Függvények:

- + **IceCream()** – A konstruktorban állítjuk be a minimum és maximum árakat.

2.1.3 Login.razor.cs – public partial class Login

Változók:

- **username: string** – A beírt felhasználónév.
- **password: string** – A beírt jelszó.
- **snackBarIsOpen: bool** – Látható-e a snackbar.

Függvények:

- **loginOnClick(): void** – A bejelentkezés gomb callback függvénye. Helyes felhasználónév és jelszó esetén a profil oldalra irányít, egyéb esetben láthatóvá teszi a snackbar-t, hogy a beírt bejelentkezési adatok valamelyike nem helyes.

2.1.4 Product.razor.cs – public partial class Product

Változók:

- + **amount: int** – A megadott mennyiség a termékből.
- **allergens: Allergen[]** – A táblázatban megjelenítendő allergének.

Függvények:

- + **Product()** – Konstruktor. A mennyiséget 0-ra állítja.
- + **toPay(): int** - visszatér a termék ára és a kiválasztott mennyiség szerinti fizetendő összeggel.
- + **onAddToCart(): void** – A kosárba helyezés callback függvénye. Ha a termék még nincs a kosárban, akkor belerakja mennyiséggel együtt. Ha benne van, akkor növeli a mennyiségét.

2.1.5 SignUp.razor.cs – public partial class SignUp

Változók:

- **myModel: SignUpModel** – A regisztrációs mezők validálását segítő modell.

Függvények:

- **Reset(): void** – A reset gomb callback függvénye. Visszaállítja az összes beviteli mező értékét.

2.2 Shared

2.2.1 MainLayout.razor.cs – public partial class MainLayout

Változók:

- **searchString: string** – A keresési szöveg.
- + **theme: MatTheme** – A MatBlazor téma, amit az oldalra alkalmazunk.

Függvények:

- **HomeOnClick(): void** – A home gomb callback függvénye, ami a home oldalra navigál.
- **profileOnClick(): void** – A profil ikon gomb callback függvénye, ami ha a felhasználó be van már jelentkezve, akkor a profiljára irányítja őt, ha nincs, akkor a bejelentkező felületre.
- **cartOnClick(): void** – A kosár ikon gomb callback függvénye, ami a kosárra navigálja a felhasználót.
- **searchOnClick(): void** – A keresés gomb callback függvénye, ami a beírt kulcsszó alapján a termék oldalára navigál vagy jelzi, hogy nem találja a terméket

2.3 További modulok

2.3.1 Allergen.cs – public class Allergen

Az allergéneket megvalósító osztály az allergén táblázat kitöltéséhez.

Változók:

- **name: string** – Az allergén neve, tartozik hozzá property.
- **contains: bool** – Tartalmazza-e a fagyit az adott allergént. Mivel egy konstans táblázatot jelenít meg minden termék mellett terméktől függetlenül, így egyszerűbb volt itt elmenteni ennek a változónak az értékét. Valós működés mellett ez a megoldás nem működne. Tartozik hozzá property.

Függvények:

- + **Allergen(name: string, contains: bool)** – Konstruktor, ami beállítja a paraméterként kapott értékre a név és a tartalmazás értékét.

2.3.2 Card.cs – public class Card

A fagyik kártyáira kerülő információkat tároló objektum.

Változók:

- **name: string** – A fagyit neve, tartozik hozzá property.
- **price: int** – A fagyit ára, tartozik hozzá property.
- **description: string** – A fagyit leírása, tartozik hozzá property.
- **imageUrl: string** – A fagyit képének URL-je, tartozik hozzá property.
- **color: int** – A fagyit színének sorszáma, tartozik hozzá property.
- **inStock: bool** – A fagyit van-e éppen raktáron, tartozik hozzá property.

Függvények:

- + **Card(name: string, price: int, description: string, imageUrl: string, color: int, inStock: bool)** – Konstruktor, ami beállítja a változók értékeit a paraméterben kapottakra.
- + **onSelected(): void** – Ez a függvény fut le, amikor a felhasználó kiválaszt egy fagyit.

2.3.3 CartElement.cs – public class CartElement

A kosárban lévő elemek.

Változók:

- **product: Card** – A kosárban lévő termék, tartozik hozzá property.
- **amount: int** – A kosárban lévő termék mennyisége, tartozik hozzá property.

Függvények:

- + **CartElement(card: Card, amount: int)** – Konstruktor, beállítja a változók értékeit a paraméterként kapottakra.
- + **remove():void** – Törli a kosárból az adott elemet.

2.3.4 GenderDef.cs – public class GenderDef

A legördülő listából választható nemek.

Változók:

- **name: string** – A nem neve, tartozik hozzá property.

Függvények:

- + **GenderDef(name: string)** – Konstruktor, beállítja a name változó értékét a paraméterként kapottra..

2.3.5 Global.cs – public static class Global

Statikus osztály a globális változók tárolására.

Változók:

- + **loggedIn: bool** – A felhasználó be van-e éppen jelentkezve.
- + **cards: List<Card>** - A termékek listája.
- + **names: string[]** – A fagyik neveinek listája.
- + **prices: int[]** – A fagyik árainak listája.
- + **descriptions: string[]** – A fagyik leírásainak listája.
- + **urls: string[]** – A fagyik képeinek url-jeinek listája.
- + **colors: int[]** – A fagyik színeinek sorszámának listája.

- + **stocks: bool[]** – A fagyik közül melyek vannak készleten.
- + **theChosenOne: Card** – Az aktuálisan kiválasztott termék, ha a felhasználó kiválaszt egyet, hogy megtekintse.
- + **products: List<CartElement>** - A kosárban található elemek listája.
- + **FullPrice: int** – A kosárelemekért összesen fizetendő összeg.

Függvények:

- + **createCards(): void** – Létrehozza a fagyikhoz tartozó objektumokat.

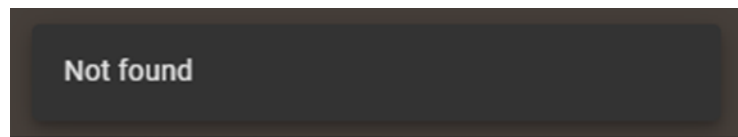
3 Felhasznált MatBlazor komponensek

Ebben a fejezetben ismertetem a MatBlazor külső komponens könyvtárból felhasznált elemeket. Komponens típusonként végigmenve mutatom be az egyes komponenseket, a felhasználásuk módját, bennük rejlő lehetőségeket és nehézségeiket.

3.1 Felugró értesítések

3.1.1 SnackBar

A SnackBar általában az oldalon felugró rövid, szöveges üzenet esetleg egy gombbal kiegészítve. Alap esetben 10000ms után az üzenet eltűnik, de ennek az időtartama a Timeout property-n keresztül módosítható. Minden esetben az oldal alján jelenik meg, alap esetben középen, de ez a bal alsó sarokra is módosítható.



8. ábra: SnackBar

SnackBar-t használtam a Login és a MainLayout felületen. HTML kódban szinte akárhol elhelyezhető, így célszerű a kód aljára rakni a többi tag-tól elkülönítve.

A felhasználáshoz először el kell helyezni egy MatSnackBar komponens és összekötni egy bool értékkel, amit egy abban elhelyezett MatSnackBarContent komponens segítségével tudunk feltölteni tartalommal.

```
<MatSnackBar @bind-IsOpen="változó_neve">  
    <MatSnackBarContent>Üzenet</MatSnackBarContent>  
</MatSnackBar>
```

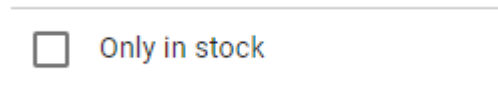
Elhelyezés után a C# kódban az isOpen értéként megadott bool típusú változóval lehet engedélyezni a megjelenítést és frissíteni kell a státuszt. A SnackBar eltűnéséről a könyvtár gondoskodik, így a fejlesztőnek nem kell a változó értékét visszaállítania ekkor.

```
snackBarIsOpen = true;  
this.StateHasChanged();
```

3.2 Form Control-ok

3.2.1 Checkbox

A Checkbox az egyik legalapvetőbb form control komponens, ami egy doboz kipipálását vagy üresen hagyását foglalja magába. Ennek állása alapján hozhatunk a kódban további döntéseket. A doboz értéke igaz vagy hamison kívül lehet még „indeterminate”, ez az az állapot, amikor a felhasználó még nem választott sem igaz, sem pedig hamis értéket az adott kérdésre, viszont egyszer választás után már nem állítható vissza ugyan erre az állapotra többet. Ezen kívül letiltható rajta a felhasználói interakció, ekkor csak kódból lesz állítható az értéke.



9. ábra: Checkbox

Checkbox-ot az IceCream felületen használtam, azaz azon az oldalon, ahol a termékek listája található meg szűrőkkel együtt. Ezen az oldalon belül a szűrők közt található.

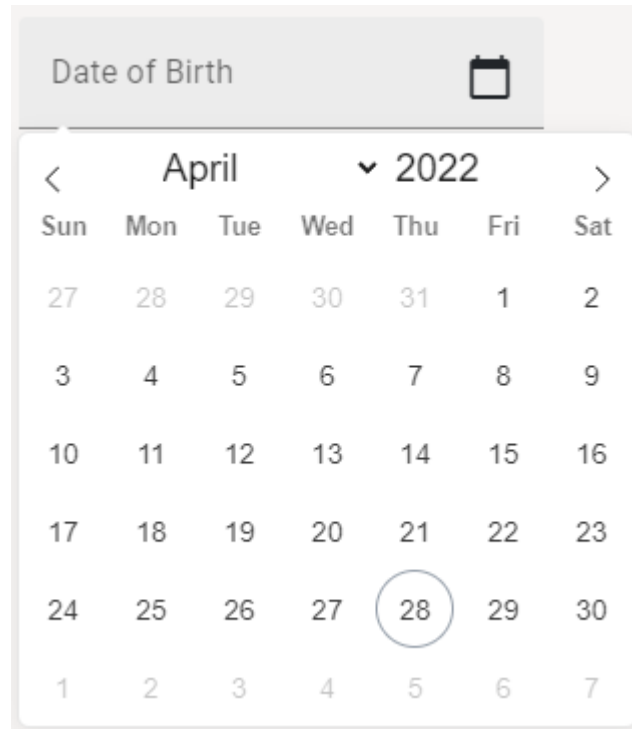
Felhasználáshoz el kell helyezni a HTML kódunk megfelelő részén egy MatCheckbox komponenst és össze kell kötni az állását tároló bool értékkel. Ezen kívül érdemes még megadni a Label property értékét, ami a Checkbox melletti szöveg értékét állítja.

```
<MatCheckbox @bind-Value="@változó" Label="Felirat"></MatCheckbox>
```

Ezek után kódból könnyen elérhető bármikor a checkbox állása, hiszen csak a hozzá bindolt bool változó értékét kell lekérdeznünk.

3.2.2 DatePicker

A DatePicker egy Textfield beviteli mezőhöz kapcsolt előre elkészített kész naptár komponens, ami segít a dátum kiválasztásában és a megfelelő formátumban kitölti vele a beviteli mezőt. A mező végében elhelyezett naptár ikonnal nyitható meg és a többi elem fölött jelenik meg egy kis ablakban.



10. ábra: DatePicker

DatePicker-t a SignUp felületen használtam a születési dátum megadásának megkönnyítésére.

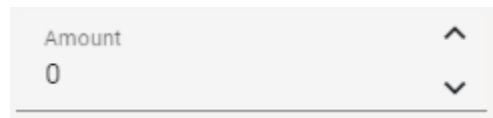
Felhasználása a vártnál egyszerűbb, csupán el kell helyezni a HTML egy MatDatePicker komponenst és meg kell adni a rá kerülő feliratot és a hozzá kötött DateTime típusú változó nevét.

```
<MatDatePicker Label="Felirat" @bind-Value="változó" />
```

A C# kód részeként pedig a változón keresztül egyből DateTime formában kezelhetjük a DatePicker-ből kapott inputot.

3.2.3 NumericUpDownField

A NumericUpDownField egy nagyon sok lehetőséget magában rejtő form control komponens, amin keresztül egyszerre adhatunk meg begépelve vagy a fel-le nyilak segítségével szinte bármilyen típusú számokat. Megadhatunk neki alsó és felső határt, fogadhatunk benne integer-eket, megadhatjuk, hány tizedes jegyig érjenek az egyes számok vagy akár százalékot vagy pénznemek értékét is kérhetjük a segítségével. Ezen kívül megjeleníthetünk az input mezőjében ikont az intuitívabb értelmezhetőségért vagy letilthatjuk a gépeléses inputok fogadását.



11 ábra: NumericUpDownField

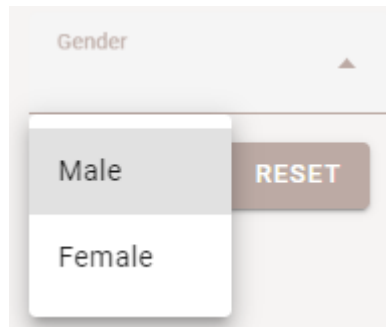
Ilyen komponenst használtam a Cart és a Product felületen, azaz azokon az oldalakon, ahol a termékmennyiség adható meg, hogy mennyit szeretnénk a kosárba helyezni.

Felhasználásához alap esetben csak a hozzákötött változó értékét kell megadni, de általánosabb esetben megadjuk neki a hozzá tartozó feliratot és egy minimum és maximum értéket, amit felvehet.

```
<MatNumericUpDownField Label="Felirat" @bind-Value=@<változó> Minimum=0  
Maximum=10></MatNumericUpDownField>
```

3.2.4 Select

A Select komponens egy legördülő listát jelent a felhasználó számára, ahonnan az előre megadott értékek közül választhat, a programozó pedig különféle típusú értékeket adhat meg a különböző választási lehetőségekhez, ami alapján később a választott elem ismert lesz. A választást könnyítése érdekében megadható hozzá HelperText. A legördülő lista a többi elem fölött, a beviteli mező mellett jelenik meg.



12. ábra: Select

Select komponens a SignUp felületen használtam a nem kiválasztásához, hiszen itt megadott lehetőségek közül lehet és csak egyet kiválasztani.

Ennek a komponensnek a felhasználása kicsit bonyolultabb. Először el kell helyezni egy MatSelect komponens, majd ezen belül minden lehetőséghez egy-egy MatOptionString-et. MatSelect tag-en belül meg kell adni a hozzá tartozó feliratot, hogy a felhasználó tudja, mit választ, majd hozzá kell kötni egy változó értéket is. A MatOptionString tag használható arra, hogy a listában megjelenő elemeket és a hozzájuk kötött értékeket definiáljuk.

```
<MatSelect Label="Gender" @bind-Value="myModel.Gender">
  <MatOptionString Value="Male">Male</MatOptionString>
  <MatOptionString Value="Female">Female</MatOptionString>
</MatSelect>
```

3.2.5 Slider

A Slider egy olyan csuszka, ahol a minimum és a maximum érték között választhatunk értéket. A rajta beállítható értékek lehetnek folytonosak vagy diszkrét, diszkrét esetben pedig a diszkrét pontok megjelölhetők. Megadhatunk emellett akár lépésközt is, hogy csúsztatásra mennyivel nőjön vagy csökkenjen a kiválasztott érték.



13. ábra: Slider

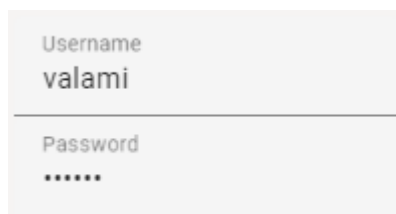
Slidert használtam az IceCream felületen a szűrők közt a minimum és a maximum ár beállításához. Nagy előnye, hogy a minimum és a maximum érték is dinamikusan változtatható, arra viszont sajnos nincs lehetőség, hogy két csúszkával egyazon sávon állítsunk be intervallumot. Hiányzik még, hogy a minimum és maximum pontokhoz tartozó, valamint a kiválasztott értéket az adott helyek fölött megjeleníthessük, így ezeket külön komponens segítségével kell kiírni.

A felhasználás ennek ellenére nagyon programozóbarát, mindössze három változóra, egy minimum, egy maximum és egy beállított értékre van szükségünk, valamint el kell helyezni egy MatSlider komponenst a HTML kódban. Itt állíthatjuk be azt is, hogy az értékek azonnal változzanak, már csúsztatás közben.

```
<MatSlider      @bind-Value="@<változó>"      ValueMin="0"      ValueMax="10"  
Discrete="true" Immediate="true"></MatSlider>
```

3.2.6 TextField

A TextField egy egyszerű szövegbeviteli mező rengeteg formázási lehetőséggel. Hozzáadhatunk ikonokat jobb vagy bal oldalra, hogy a felhasználó számára intuitívabbá váljon, emellett megadható az inputként várt típus is, ekkor a bevitt érték csak akkor marad meg a mezőben, ha az megfelel az elvárt típusnak.



14. ábra: TextField

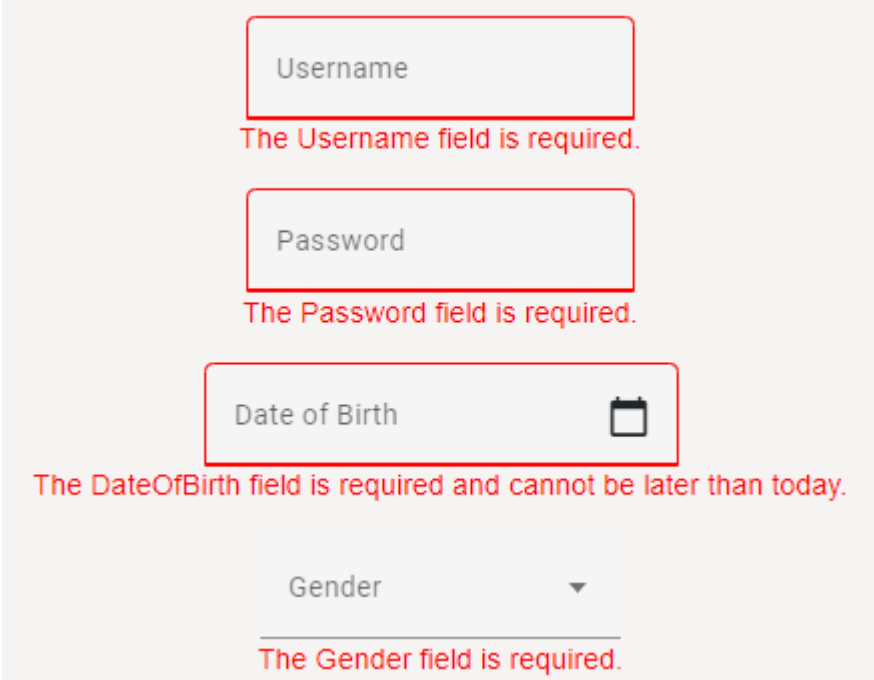
TextField elemet használtam a Login, SignUp és a MainLayout felületen is. Hatalmas segítség volt, hogy beépített módon megadható neki, hogy „password” típusú legyen és ebben az esetben a begépeltek szöveg nem válik láthatóvá, hanem automatikusan kis körök jelennek meg karakterenként. Külön érdekessége még, hogy Microsoft Edge böngészőben tartozik hozzá egy gomb a beviteli mező végén, amivel láthatóvá tehető vagy újra elrejtethető a begépeltek szöveg. Chrome felületen erre valamiért nincs lehetőség.

A felhasználás alap esetben egy egyszerű MatTextField komponens helyezéséből áll és a hozzá tartozó felirat és változó nevének megadásából. Ez bővíthető tovább a különféle formázások vagy extra követelmények megadásával.

```
<MatTextField @bind-Value="<változó>" Label=" Felirat "></MatTextField>
```

3.2.7 Validation via EditContext

A Validation via EditContext egy olyan lehetőség, ami a felhasználó inputjait hasonlítja össze a fejlesztő által megadott kritériumokkal.



The image shows a registration form with four input fields, each with a red border indicating a validation error. The fields are: Username, Password, Date of Birth (with a calendar icon), and Gender (a dropdown menu). Below each field is a red error message. The Username error is 'The Username field is required.', the Password error is 'The Password field is required.', the Date of Birth error is 'The DateOfBirth field is required and cannot be later than today.', and the Gender error is 'The Gender field is required.'.

15. ábra: Validation via EditContext

Ezt az elemet a SignUp felületen építettem be és mind közül ezt volt a legnehezebb megérteni.

Első lépésként a C# kódban célszerű létrehozni egy validációs modellt, amiben az inputok értékeit tároljuk és hozzá szögletes zárójelben megadhatunk feltételeket. A 8. ábrán látható beviteli mezőkhöz tartozó modell a követelményekkel együtt a következőképpen néz ki:

```
public class SignUpModel
{
    [Required]
    public string Username { get; set; }

    [Required]
    [MinLength(8)]
    public string Password { get; set; }

    [Required]
    public string Gender { get; set; }

    [Required]
    [CustomValidation(typeof(SignUpModel), nameof(RequiredDateTime))]
    public DateTime DateOfBirth { get; set; }
}
```

Ebben az esetben 4 beviteli mezőről beszélünk, melyek mindegyikén megadása kötelező. Ezen kívül a Password mezőben megadott értéknek legalább 8 karakter hosszúnak kell lennie és a DateOfBirth mező értékének vizsgálatához saját metódust alkalmazunk egy saját függvénnyel, melynek neve RequiredDateTime és egy ValidationResult értékkel tér vissza. Ebben adhatunk meg személyre szabott validációs üzenetet is. A fenti példához a kód a következőképpen néz ki:

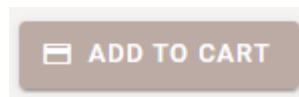
```
public static ValidationResult RequiredDateTime(DateTime value,
ValidationContext vc)
{
    return value > DateTime.MinValue && value <= DateTime.Today
        ? ValidationResult.Success
        : new ValidationResult($"The {vc.MemberName} field is required and
cannot be later than today.", new[] { vc.MemberName });
}
```

Ezután a HTML kódban a megfelelő beviteli mezőkhöz meg kell adni ValidationMessage (validációs üzenet) tag-et is.

3.3 Gombok

3.3.1 Button

Az egyszerű gombok nagyon gyakori elemei a felhasználókkal interaktáló weblapoknak, hiszen gyakran szeretnénk, ha egyes folyamatok a felhasználó jelzésére hajtódjanak végre. A formázásokon kívül lehetőség van még a gombnyomáshoz tartozó parancsok változatos megadására és a gombok letiltására is.



16. ábra: Button

Gombokat használtam a Login, a SignUp és a Product felületen is, hogy a felhasználó érvényre juttathassa a többi komponensben megadott inputjait.

A felhasználáshoz mindössze egy MatButton komponens elhelyezésére van szükség a HTML kódban és az OnClick parancs összekötésére egy megadott függvénnyel, ami akkor hajtódik végre, ha a felhasználó megnyomja a gombot. Lehetőség van még link megadására, ha azt szeretnénk, hogy a gombnyomás átirányítsa a felhasználót, de erre célszerűbb a külön erre a célra létrehozott ButtonLink komponenst használni. A gombon olvasható feliratot a nyitó és a záró tag között adhatjuk meg.

```
<MatButton OnClick="@Click">Felirat</MatButton>
```

3.3.2 Icon

Az ikonok a nevüknek megfelelően egyszerű kis képek, amik bárhol elhelyezhetők egy oldalon. Alap esetben nem tartozik hozzájuk semmilyen működés, ha pedig mégis szeretnénk hozzá valamilyen parancsot rendelni, az IconButton használata javasolt, melynek viselkedését a 3.3.3 fejezet írja le.

Ikonokat főleg más komponensekbe ágyazva, az Icon property megadásával vagy gombként használtam, így önálló, működés nélküli ikonok nincsenek a projektben.

A MatBlazor könyvtár szerencsére rengeteg különféle ikonnal rendelkezik¹, amik számomra minden igényt lefedtek, akármilyen célra kerestem képet.

¹ Az itt megtalálható ikonok listája az alábbi linken érhető el: <https://www.matblazor.com/Icon>

3.3.3 IconButton

Az IconButton olyan komponens, amit egyből a hozzá tartozó paranccsal is elláthatunk, hogy az végrehajtódjon az ikonra kattintáskor. A működése ugyan olyan, mint a 3.3.1 fejezetben leírt egyszerű gomboké, annyi különbséggel, hogy nem adható meg neki felirat, viszont kötelezően ki kell választani, hogy melyik ikon jelenítse meg az a gombot. További lehetőség még a sima ikon mellett egy ToggleIcon property megadása. Ekkor a gombnak tulajdonképpen két állása lesz. Egyik állásban az Icon, másikban pedig a ToggleIcon property-ben megadott kép jelenik meg és egy bool érték is hozzákapcsolható, hogy bármikor lekérdezhessük, hogy éppen melyik állapotában van a gomb.



17. ábra: IconButton

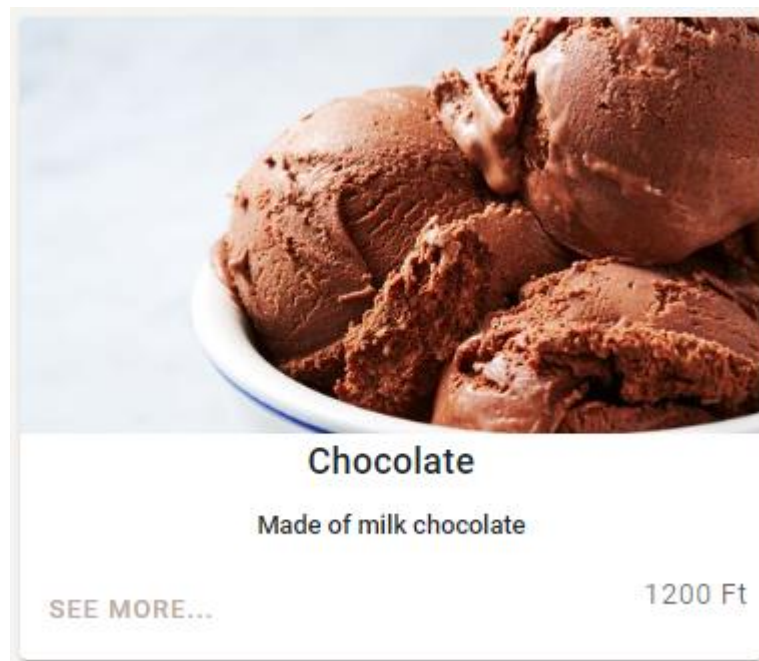
IconButton-t a MainLayout fejlécébn használtam a home page-re navigáláshoz, kereséshez, valamint a kosár és a profil eléréséhez. Mivel ezek a gombok egyértelműek a felhasználó számára, így nem kell hozzájuk külön felirat, elég a kép. Elhelyezésük a HTML kódban egy MatIconButton tag-gel lehetséges és a megfelelő property-k kitöltésével.

```
<MatIconButton Icon="ikon_neve" OnClick="@függvény"></MatIconButton>
```

3.4 Layout-ok

3.4.1 Card

A Card, vagy más néven kártya egy egységes megjelenítést ad összetartozó adatok megjelenítésére. Egy téglalap alakú területet oszt fel kisebb egységekre. A tetején jeleníthető meg egy kép. A képen vagy a kép alatt legfelső elemként helyezhető el egy cím vagy egy kiemelt megjegyzés. Ezután következik egy leírásnak dedikált rész. Ide bármilyen szöveg kerülhet, ez alapvetően nincs kiemelve. A kártya legalján találhatóak végezetük a gombok. Bal oldalt egy egyszerű gomboknak fenntartott régió, míg jobb oldalt egy Icon Button-öknek fenntartott terület van. Ez egy jól kitalált koncepció egy kártyára szánt adatok megjelenítésének, de ezekben a régiókban nem feltétlenül szükséges tartani magunkat ehhez, mint ahogy azt én se tettem.



18. ábra: Card

Ezt a komponenszt az IceCream felületen, a fagyaltok listázásához használtam fel. A rengeteg szekció könnyen kezelhetővé tette magát a kártyát, így nem volt szükség semmilyen további formázásra és rugalmasan tudtam a szekciók adta formázásokat kihasználni.

A kártya létrehozásához mindenekelőtt szükség van egy MatCard komponens elhelyezésére. Ebbe fognak kerülni a további komponensek, a MatCardContent és a MatCardActions. A MatCardContent része a MatCardMedia, ami a kép megjelenítéséért

felelős, majd ezután helyezhetők el valamilyen kiemelt szövegtípussal a képhez tartozó fontos információk és valamilyen alap stílussal a leírások. A `MatCardActions` rész jelenti az alsó, gombokat tartalmazó sávot. Bal oldalt a `MatCardActionButtons` tag-be írhatók a gombok, vagy egyéb elemek, amiket balra igazítva szeretnénk megjeleníteni, valamint jobb oldalt a `MatCardActionIcons` tag-be írhatók az ikon gombok, vagy egyéb elemek, amiket bal oldalt szeretnénk megjeleníteni.

```
<MatCard class="demo-mat-card">
  <MatCardContent>
    <MatCardMedia Wide="true" ImageUrl="url"></MatCardMedia>
    <MatHeadline6>Cím</MatHeadline6>
    <MatBody2>leírás</MatBody2>
  </MatCardContent>
  <MatCardActions>
    <MatCardActionButtons>
      <MatButton>Gomb</MatButton>
    </MatCardActionButtons>
    <MatCardActionIcons>
      <MatIconButton Icon="ikon"></MatIconButton>
    </MatCardActionIcons>
  </MatCardActions>
</MatCard>
```

3.4.2 Divider

A `Divider` komponens tulajdonképpen egy két másik komponenst elválasztó egyenes vonal. Vízszintesen húzható ki bármely két komponens közé vagy listaelem mellé folytatólagosan az `Inset` property igazra állításával. Emellett beállítható, hogy az öt tároló komponens széleitől tartson-e valamennyi távolságot, vagy teljes széltében töltsse ki azt.

Filters

☐ Only in stock

19. ábra: `Divider`

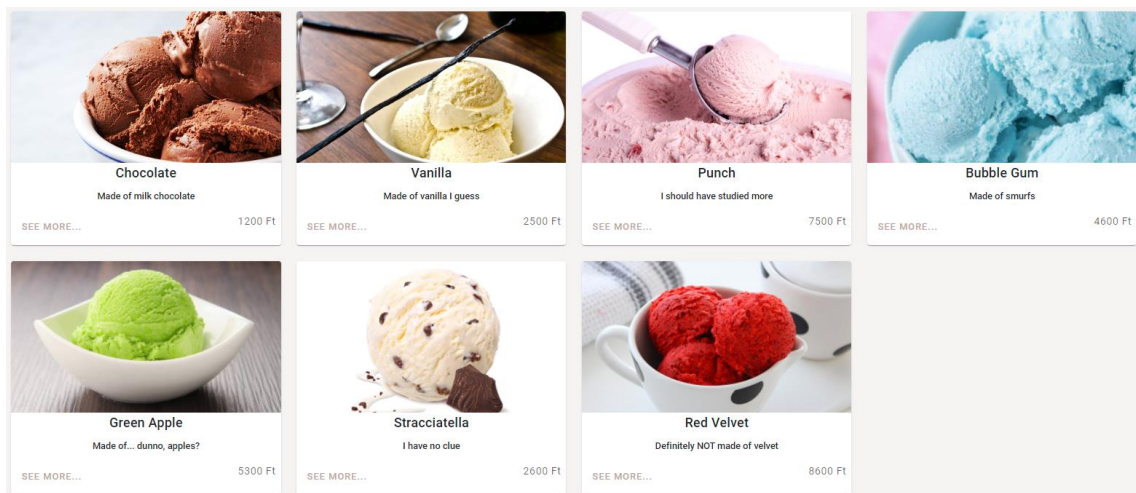
`Divider`-t használtam az `IceCream` oldalon a szűrők elválasztásához és a `Product` felületen a termék nevének az oldal többi részétől való elválasztására.

Az egyik legegyszerűbben felhasználható komponens, mindenféle paraméter megadása nélkül beilleszthető a HTML kódba.

```
<MatDivider></MatDivider>
```

3.4.3 Layout Grid

A Layout Grid egy láthatatlan komponens, ami 12 egyforma egységre osztja széltében az őt tartalmazó elemet. Ezáltal oszlopokra és sorokra bonthatóvá válik az ablak. A cellák egymásba is ágyazhatók, valamint megadható, hogy egy cella a 12-ből hányat foglaljon el. Amennyiben egy cella nem fér már el a sorban található 12 egység közül még üresen állókban, úgy a következő sorba tolódik. Beállítható a cellák jobb vagy bal oldalhoz igazítása, vagy a felső és alsó határtól függően is igazíthatók felülre, középre és alulra.



20. ábra: Grid Layout

Ezt az elrendezést használtam a Cart, Home, IceCream, Product és Profile oldalakon is.

A felhasználása a többi komponensétől eltérő. Önálló tag-ek helyett egyszerű div tag-eket kell csak elhelyezni és ezen tag-eknek kell megadni a megfelelő css osztályokat a class property-ben. Az elrendezés legfelső szintjébe kerül a mat-layout-grid tag, ez adja meg az elrendezés alapját. Itt adható meg továbbá, hogy a cellákat jobbra vagy balra szeretnénk-e igazítani a mat-layout-grid-align-<irány> osztállyal, ahol az irány right vagy left lehet. Ebben kerül a mat-layout-grid-inner tag, ami megadja, hogy az elrendezésen belül cellákat szeretnénk lerakni. Levégül egy legbelső tagben beállítjuk a mat-layout-grid-cell osztályt. Ez utolsó div-ben adható még meg, hogy hány egységet foglaljon el az adott cella, amihez a mat-layout-grid-cell-span-<n> osztály használható, ahol n megegyezik az elfoglalni kívánt egységek számával. Itt adható még meg az alsó és felső határhoz igazítás is a mat-layout-grid-cell-align-<irány> osztállyal, ahol az irány top, bottom vagy middle lehet.

Ez HTML kódban így néz ki:

```
<div class="mat-layout-grid">
  <div class="mat-layout-grid-inner">
    <div class="mat-layout-grid-cell mat-layout-grid-cell-span-2">
      ...
    </div>
  </div>
</div>
```

3.4.4 List

A List egy beépített lista komponens, ami a felsorolásokat teszi esztétikusabbá és átláthatóbbá. A listában szereplő listaelemeket egymás alatt jeleníti meg. Ezekhez köthető link, ami kattintásra átirányítja a felhasználót vagy tovább ágyazható bele elsődleges és másodlagos szöveg.

First name Reka
Last name Gaal
Gender Female
Date of birth 1999. 16. 14.
Town Budapest
E-mail jnmjkl@gmail.com
Phone number +36 20 588 8671
Number of orders 12

21. ábra: List

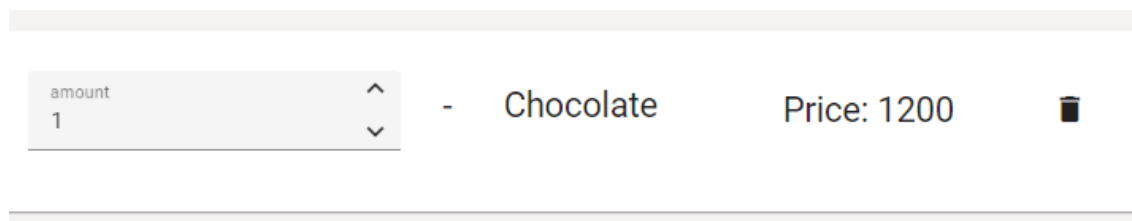
Listát Profile oldalon használtam a személyes adatok felsorolására két soros listaelemekkel egymástól elválasztva.

A MatList tag szolgál a komponens elhelyezésére és ezen belül a MatListItem tag-ekben helyezhetők el a listaelemek egyesével, amik egy vízszintes vonallal választhatók el a MatListDivider tag-gel. Az egyes listaelemekbe ezen túl ágyazható elsődleges és másodlagos szöveg, ehhez azonban előbb érdemes beállítani a MatList tag-en belül a TwoLine property-t igazra, hogy elég helyet hagyjon a komponens a kétsoros listaelemeknek. Magukat a szövegeket egy MatListItemText tag-be helyezett MatListItemPrimaryText és MatListItemSecondaryText tag-ekkel lehet beépíteni.

```
<MatList TwoLine="true">
  <MatListItem>
    <MatListItemText>
      <MatListItemPrimaryText>szöveg1</MatListItemPrimaryText>
      <MatListItemSecondaryText>szöveg2
    </MatListItemSecondaryText>
    </MatListItemText>
  </MatListItem>
  <MatListDivider></MatListDivider>
</MatList>
```

3.4.5 Paper & Elevation

A Paper komponens egy kiemelhető egységes hátteret ad az azon elhelyezett többi elemnek. Az Elevation property adja meg a kiemelés mértékét, ami tulajdonképpen egy 3D-s hatást kelt árnyékolás segítségével.



22. ábra: Paper & Elevation

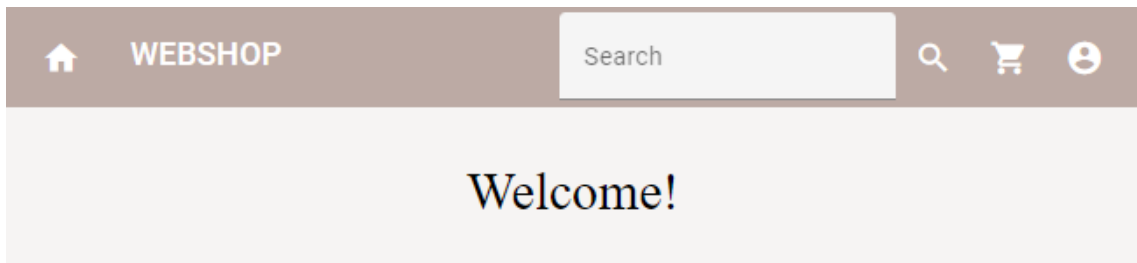
Ezt a komponens a Cart felületen használtam a kosárba helyezett elemek hátterének, hogy egységet biztosítson az egyes soroknak.

Felhasználása egy egyszerű MatPaper tag elhelyezésével lehetséges. Ezen belül adható meg az Elevation property értéke 0 és 24 között. Ezen kívül a div tag is kiemelhető, nem csak a MatPaper. Ehhez a class property beállítása szükséges mat-elevation-z<n> - re, ahol n a kiemelés mértéke.

```
<MatPaper Elevation="5">  
...  
</MatPaper>
```

3.4.6 Themes

A témák az alkalmazásban elhelyezett komponensek fő színeit adják meg, így azoknak az oldalon belül egységes megjelenítést biztosít.



23. ábra: Themes

Az általam beállított témát az egész oldalra alkalmaztam, így a MainLayout felületét ágyaztam ebbe a komponensbe.

Felhasználásához mindössze egy MatThemeProvider tag-et kell rakni a komponensek köré a HTML kódban. Ebben a tag-ben beállítjuk a Theme property értékét egy, a kódban létrehozott MatTheme típusú változóéra. C# kódban a MatBlazor könyvtár importálásával lesz elérhető a MatTheme típus. Létrehozása az alábbi módon történik.

```
public MatTheme theme = new MatTheme()  
{  
    Primary = "#bcaaa4",  
    Secondary = "#bdbdbd"  
};
```

Itt meg kell adni neki egy elsődleges és egy másodlagos színt, amit megtehetünk string-ként egy kettőskereszt után a szín hexadecimális kódjával, vagy beépített MatBlazor színeket is megadhatunk.

3.4.7 Typography

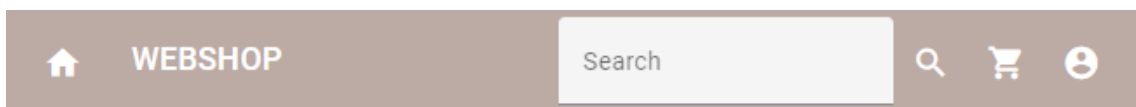
A Typography a beépített MatBlazor betűtípusokat jelenti. Ezek beállíthatók tag-ekkel vagy class property megfelelő beállításával. Vannak HeadLine, Subtitle, Body, Caption és Overline szövegtípusok. A fejlesztő találhat akármekkora méretű és kiemelt betűtípusokat is, könnyen felhasználható és változatos.

```
<Math2>  
    Szöveg  
</Math2>
```

3.5 Navigáció

3.5.1 AppBar

Az AppBar egy felső navigációs sávot biztosít az oldalnak. Elhelyezhető rajta az oldal neve, gombok vagy akár keresősáv is. Ezen a komponensen belül helyezhető el az oldal tartalma is.



24. ábra: AppBar

Az AppBar-t a MainLayout-ban helyeztem el, így az összes oldalon elérhető és könnyen átnavigálhatunk vele a főbb oldalakra vagy kereshetünk a termékek közt.

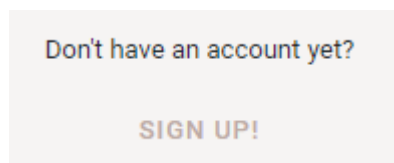
Az elhelyezés egy MatAppBarContainer tag-gel kezdődik, ez biztosítja az AppBar helyét, a nevéből is mutatja, hogy ez lesz a tároló. Ezen belül helyezhető el maga a MatAppBar, ami további sorokra osztható és azon belül szekciók hozhatók létre, ezzel könnyítve az elrendezést. MatAppBar után helyezhető el még az AppBarContainer-en belül a MatAppBarContent.

A HTML kódban ez az alábbi módon néz ki:

```
<MatAppBarContainer>
  <MatAppBar Fixed="true">
    <MatAppBarRow>
      <MatAppBarSection>
        <MatAppBarTitle>Oldal címe</MatAppBarTitle>
      </MatAppBarSection>
      <MatAppBarSection Align="@MatAppBarSectionAlign.End">
        ...
      </MatAppBarSection>
    </MatAppBarRow>
  </MatAppBar>
  <MatAppBarContent>
    ...
  </MatAppBarContent>
</MatAppBarContainer>
```

3.5.2 ButtonLink

A ButtonLink egy olyan gomb komponens, ami egyből magába foglalja, hogy valamilyen navigációs feladatot fog ellátni, tehát megnyomására egy másik oldalra navigálja a felhasználót. Kinézhet egyszerű szöveggént, de adható neki keret vagy egyéb módokon is formázható, teljes mértékben a fejlesztőre van bízva, alap esetben egyszerű, átszínezett szöveggént jelenik meg a beállított téma színeitől függően.



25. ábra: ButtonLink

ButtonLink-et a Login felületen használtam, hiszen itt volt olyan lehetőség, hogy ha a felhasználó még nem rendelkezik fiókkal, akkor átirányíttassa magát a regisztrációs felületre.

Elhelyezése A MatButtonLink tag-gel lehetséges, ezen belül pedig a Href property beállításával adható meg az útvonal, hogy hova irányítson tovább a gomb megnyomása. A gomb felirata a nyitó és záró tag-ek közt adható meg.

```
<MatButtonLink Href="/masik_oldal ">Felirat</MatButtonLink>
```

3.6 Táblázatok

3.6.1 Table

A Table komponens egy táblázatmegjelenítő eszköze a MatBlazor komponenskönyvtárnak, beállíthatók a fejléc szövegei és a sorokba kerülő adatok. Ezen kívül lehetőség van még a komponensen belül rendezések megadására saját függvényekkel vagy a sorok kiválaszthatóvá tételére. Az elemek lapozhatóvá tételéhez rendelkezésre áll egy elrejtető vagy megjeleníthető lapozó eszköz a táblázat alatt, de még a Table komponensen belül.

Allergen	Contains
milk	True
eggs	True
tree nuts	False
peanuts	True
wheat	False

26. ábra: Table

Táblázatot a Product felületen alkalmaztam az allergének megjelenítésére. Az egyik oszlop tartalmazza az allergének nevét, a másik pedig egy igaz vagy hamis értéket, hogy a termék tartalmazza-e az adott allergént.

A táblázat beépítéséhez egy MatTable komponens elhelyezésére van szükség a HTML kódban. Ezen belül szükséges megadni az Items property-t ami a táblázatban szereplő objektumok listáját jelenti. Ezután a táblázaton belül megadjuk a MatTableHeader tag-ben a fejléc elemeit, majd a MatTableRow tag-ben, hogy az egyes sorokat milyen elemekkel töltsse fel oszloponként az Items property-ben megadott objektumok alapján.

A HTML kódban ez az alábbi módon néz ki:

```
<MatTable Items="@allergens" ShowPaging="false">
  <MatTableHeader>
    <th>első oszlop neve</th>
    <th>második oszlop neve</th>
  </MatTableHeader>
  <MatTableRow>
    <td>@context.<első_megjelenítendő_adat></td>
    <td>@context.<második_megjelenítendő_adat></td>
  </MatTableRow>
</MatTable>
```