HÓDMEZŐVÁSÁRHELYI SZAKKÉPZÉSI CENTRUM
MAKÓI NÁVAY LAJOS TECHNIKUM ÉS KOLLÉGIUM

Warhammer 40K témájú webshop és a webshop használatát segítő program(ok) bemutatása.

ZÁRÓDOLGOZAT

Készítette:

Kádár Dávid

54 213 05 Szoftverfejlesztő szakképesítés

Makó

2021

Tartalom

Bevezető

Egy webshopot és a webshop használatát segítő alkalmazást fogok készíteni. A weblap elkészítése során két fő funkciót kell megvalósítanom az első, hogy a weblap alkalmas legyen rendelések leadására a második, hogy a weblap tartalmazzon informatív anyagokat a témában. Ezenfelül még egy jól kinéző és felhasználó barát design megtervezése és kialakítása is a feladatom.

Az asztali alkalmazás tartalmazni fog egy beléptető felületet és az itt bejelentkező felhasználó képes lesz a jogosultságai alapján a termékeket tartalmazó adatbázis tábla tartalmának frissítésére. Ezenfelül még az asztali alkalmazás tartalmazni fog egy Detachment Designer funkciót, ami a vásárlást fogja segíteni. Munkám során sok probléma megoldásában segített a StackOverflow[3] fórum használata és az ott feltett kérdések és adott válaszok.

Téma választásának indoklása

A weblapom és programom témája a Warhammer 40K nevezetű kitalált univerzum és táblajáték. Azért választottam témámnak mert már egy éve érdeklődök a Warhammer 40K iránt és ez idő alatt nagyon megtetszett és egy elégbő téma, hogy köré lehessen építeni egy záródolgozatot.

1 Fejlesztői dokumentáció

1.1 Adatbázis tervezése

Az adatbázis megtervezésekor figyelembe kellet vennem, hogy a weboldal és a asztali alkalmazás milyen funkciókkal fog rendelkezni és ehhez milyen adatbázisokra lesz szükségem. Mivel webshop a választott témám ezért szükségünk lesz a rendelések lebonyolításához egy termékek(Products) és egy Rendelések(Orders) táblára.

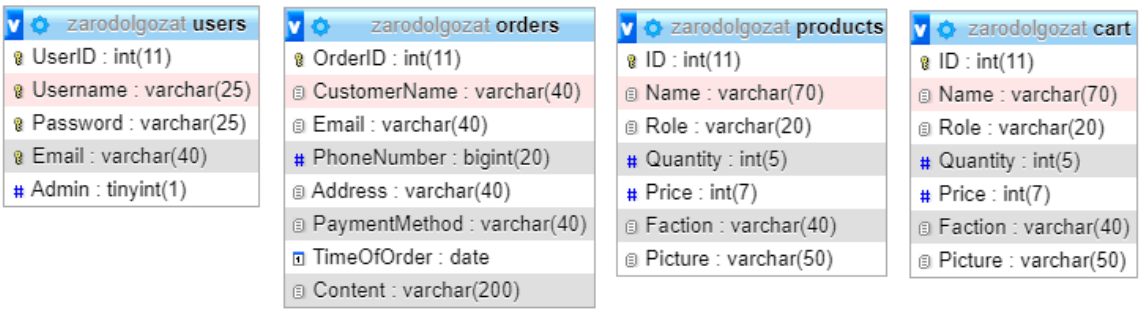
Emellett az oldal tartalmazni fog egy kosár funkciót, ahol a rendelés leadása előtt összegyűjthetjük a megrendelni kívánt termékeket és nem kell egyesével megvásárolnunk azokat. A kosár funkció megvalósításához szintén szükségünk lesz egy táblára ez a tábla a Kosár(Cart) tábla lesz.

A termékek táblánknak tartalmaznia kell a termékek megkülönböztetésére szolgáló egyedi azonosítót pár alap adatott a termékről például a két legfontosabb a név és az ár emellett tartalmaznia kell egy mennyiség mezőt is ezt az asztali alkalmazásunk használja.

A Rendelések táblának tartalmaznia kell a rendelést feladó felhasználó adatait a rendelés faladásának idejét és a rendelés tartalmát emellett a rendelések megkülönböztetésére.

A kosár tábla működésének megvalósításához formailag megegyezőnek kell lennie a termékek táblával ezért ugyan azon mezőket fogja tartalmazni.

1.2 Adatbázis felépítése



A feladathoz szükséges adatbázis létrehozására és Kezelésére a Xampp-ot azon belül MySql-t használtam. A weboldal és az asztali program négy adatbázis tartalmaz ezek a Users, Orders, Products,Cart.

Az első tábla a users tartalmazza a felhasználók adatait ez a tábla öt darab mezőt tartalmazz ezek az alábbiak:

- UserID: Ez egy egyedi azonosító, amit minden felhasználó kap a profil létrehozásakor. A táblában ez az elsődleges kulcs. Automatikus számozásra van állítva tehát a mező tartalmát magától kitölti profil létrehozáskor.
- Username: A felhasználó által a regisztráció során megadott felhasználó név. Varchar típusa miatt tartalmazhat betűket és számokat is, maximum 25 karakter hosszú lehet.
- Password: A felhasználó által a regisztráció során megadott Jelszó Varchar típusa miatt tartalmazhat betűket és számokat is maximum 25 karakter hosszú lehet.
- Email: A felhasználó által a regisztráció során megadott Email cym Varchar típusa miatt tartalmazhat betűket és számokat is maximum 40 karakter hosszú lehet.
- Admin: Egy boolean mező, ami tinyint ként jelenik meg két értéket vehet fel 0 és 1-et. Ez a mező felelős, hogy jelölje, ha egy felhasználó rendelkezik-e adminisztrátori jogosultságokkal.

A második tábla tartalmazza a rendelések adatait és az adatok tárolásához szükséges 6db mezőt.

- OrderID: Ez egy egyedi azonosítószám, amit minden rendelés megkap a rendelés leadásakor. Maximum 11 karakter hosszú lehet.
- CustomerName: A vásárló nevét tartalmazó varchar típusú mező maximum 40 karakter hosszú nevet tud tárolni.
- Email: A vásárló nevét tartalmazó varchar szöveges mező maximum 40 karakter tárolására alkalmas.
- PhoneNumber: A vásárló telefonszámát tárolja kapcsolattartás céljából.
- Address: A vásárló által megadott címet tartalmazza varchar típusú mező.
- PaymentMethod: A rendeléskor kiválasztott fizetési módot tartalmazza két értéke lehet Cash on Delivery(utánvétel) és Paid in Advance(előre fizetett).

- TimeOfOrder: A rendelés leadásának ideje a mező típusa date így dátumokat tudd tárolni.
- Content: Varchar Mező, ami a megrendelt termékek nevét és azok mennyiségét tartalmazza.

A Products(termékek) és a Cart(kosár) azonos nevű és azonos típusú mezőket tartalmaz. A két táblának a kosár funkcionalitása miatt kell meg egyeznie mivel a kosárnak képesnek kell lennie termék adatoktárolására.

- ID: csak számokat tartalmazhat 11 karakter hosszú lehet ez a mező tartalmaz egy egyedi számokból álló azonosítót automatikusan kitölti magát új rekord felvezetésekor.
- Name: Varchar mező 70 karakter hosszú szöveg tárolására alkalmas. A termékek nevét tartalmazza.
- Role: varchar típusú mező a termékhez tartozó Role(szerep)-t tartalmazza az asztali alkalmazás használja.
- Quantity: Int típusú számot tartalmazó mező a termék mennyiségét tárolja tehát hogy hány darab olyan termék van raktáron.
- Price: Maximum 7 karakter hosszú számot tartalmazó mező a termék árát tartalmazza euroban megadva.
- Faction: Varchar típusú mező maximum 40 karakter hosszú lehet. Az asztali alkalmazás használja ezt a mezőt.
- Picture: Varchar egy a terméket ábrázoló képnek a nevét tartalmazza.

1.3 Csatlakozás adatbázishoz

A weboldal megfelelő működéséhez szükségünk van az adatbázis és a weblap összeköttetésére ez azért is szükséges mert a weboldal tartalmaz PHP kódot. Ehhez egy külső PHP fájlra van szükségünk.

kapcs.php

```php
<?php
function dbkapcs() {
    $host="localhost";
    $user="root";
    $pass="";
    $db="zarodolgozat";
    global $kapcs;

    $kapcs=mysqli_connect($host,$user,$pass,$db) or die ('Hiba a kapcsolódáskor!');
    $kapcs -> set_charset("utf8");
    return $kapcs;
}

?>
```

Ahhoz, hogy a PHP fájl kapcsolatot tudjon létesíteni az adatbázissal négy tulajdonságot kell megadnunk ezek a host: a hostnak a neve jelen esettben ez a localhost. User a felhasználó nevét kell megadni, amivel hozzá férünk a mysql adatbázisunkhoz jelen eset ez a root felhasználó a következő a Pass ez a változó tárolja a jelszót mivel a root felhasználót használjuk ezért most nincs szükség jelszó megadására. Még egy db változóban meg kell adnunk az adatbázis nevét.

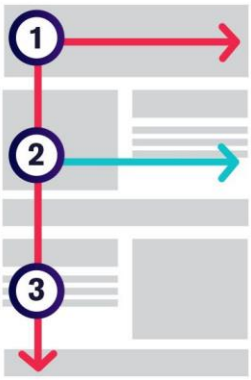
Ezután a mysql_connect functiont kell használnunk ahhoz, hogy megfelelően működjön meg kell adnunk sz előzőleg deklarált változókat. Egy biztonsági vagy feltételt is elhelyeztem a kapcsolódás után, hogy ha nem tud kapcsolódni akkor azt egy hiba üzenettel jelezze. Emellett még a kapcsolódáshoz megadjuk, hogy az adatbázis utf8 kódolású.

1.4 Weboldal

1.4.1 weboldal felépítése

A weboldal felépítésénél a cél egy egyszerű, de jól kinéző design megtervezése és megvalósítása volt. A weboldal elkészítéséhez Bootstrap-et[1] használtam. Azért döntöttem a Bootstrap használata mellett mert a Bootstrap designok használatával a megszokott html weblap készítésnél gyorsabban tudtam megvalósítani a weboldalt. Ezenfelül a weboldal a Bootstrap használata miatt automatikusan reszponzív weboldal lesz.

A weboldal felépítésének megtervezésekor az F modellt is használtam, Az F modell azt jelöli, hogy a felhasználók milyen sorrendbe tekintik meg egy webfelületet.



A felhasználók először a felület balfelső sarkát tekintik meg tehát ide érdemes elhelyezni fontosabb elemeket. Pl: Navigáció, Cég logó

Ezután a felhasználók egy F alakban fentről lefele és balról jobbra tekintik meg az oldalt.

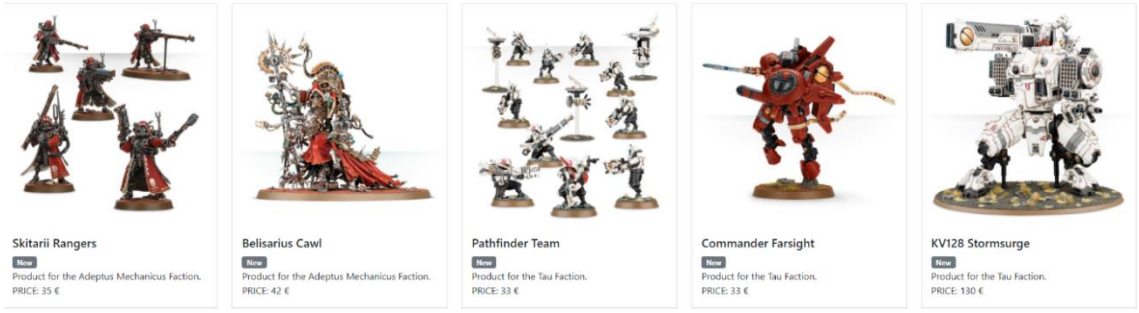
Az első belépő főoldal az index.php az oldal fő feladata a látogató figyelmének megragadása ezért ezen az oldalon sok nagy méretű képet használtam emellett hasznosítottam az F modellt is.

Az első elem a weboldalamon egy Navbar ez felesős az oldalak közötti navigáció megkönnyítéséhez az F modell használata miatt a navbar a weboldal legtetején helyezkedik el.

Az egyszerűség megőrzésének érdekében a Navbaron csak a legfontosabb elemeket helyeztem el. A navbaron a bolt témáját jelölő WH40K felirat található mellette a kezdő oldalra vezető home gomb a Termékekhez vezető faction legördülő menü és a Cart(kosár) gomb találgató.

A Navbaron navigációs pontokon kívül még egy világos és sötét téma váltó gomb is található. A téma választó gomb segítségével az oldal, mind nappal és éjszaka is kellemesen használható marad.

Ezután egy Bootstrap carousel képnézegető található itt elhelyeztem néhány nagy figyelem felkeltő képet, ami megragadhatja a felhasználó figyelmét. Emellett a képek link ként is működnek és a képhez kapcsolódó termékekhez vezetnek így a felhasználó gyorsan elérheti a neki megtetsző termékeket.

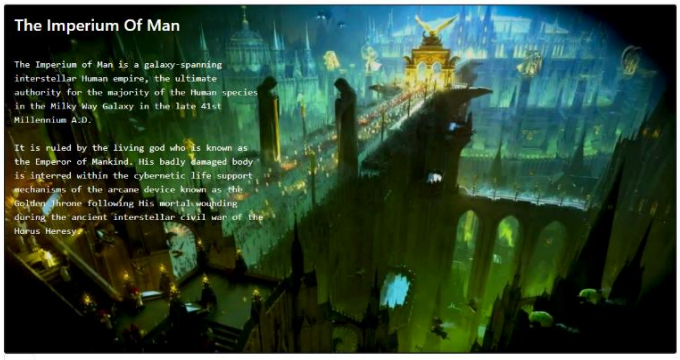


A képnézegető alatt egy a PHP kód által létrehozott Bootstrap kártya sor található. Ebben a kártya sorban mindig az öt legújabb termék jelenik meg. Ezeket a kártyákat azért helyeztem el a főoldalon, hogy a képnézegető után jelezze, hogy ez egy webshop.

A Főoldal alján található egy Bootstrap footer(lábrész) található, ami a copyright figyelmeztetést tartalmazza és a bolt elérhetőségei. Ez a footer a többi oldalon is megtalálható.

A főoldalról a képre kattintva vagy a Faction dropdown menüt használva át juthatunk a három aloldalra, amik a termékeket tartalmazza. Ezek az oldalak esetében két oldalra elhelyeztem whitespaceeket üres helyeket az oldal két szélén. Ezzel egy oszlopos formát kialakítva. Ezek az oldalak követik a főoldal sémáját fehér alap, amit színes képek szakítanak meg.

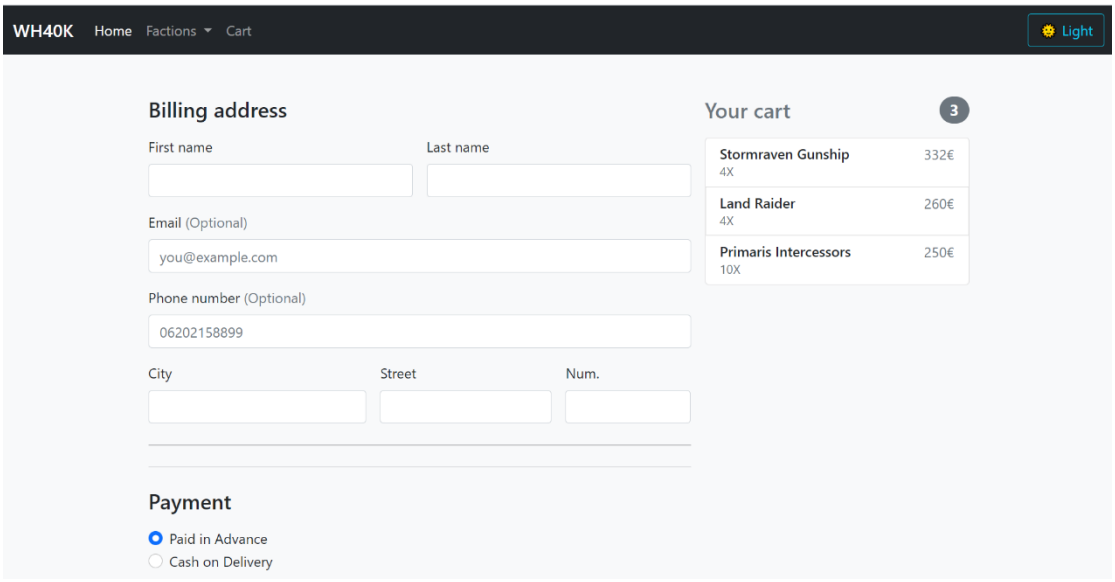
Mivel ezen az oldalon is csak Bootstrap elemeket használtam ezért itt is használható a Navbaron megtalálható téma váltó gomb. Ezen az aloldalon is megtalálható az egységesség érdekében ugyan azon navbar mint ami a főoldalon.



A navbar után egy Bootstrap overlay card-ot helyeztem el, ami tartalmaz egy a Frakcióhoz kapcsolódó képet és egy kis leírást magáról a frakcióról. A színes képeken fehér feliratot használtam, ami a kép elsötétítése után egy jól látható kontrasztos feliratot alkot.

Az információs kártya alatt egy tabs menü található ezen a menün van egy All fül és a kisebb frakciók, amelyek kötődnek egymáshoz. A különböző menüpontokra kattintva PHP és Bootstrap kártyák segítségével megjeleníti a kiválasztott frakcióhoz tartozó termékeket. Itt ugyan azon típusú és formai felépítésű kártyákat használtam, mint a főoldalon.

A tabs menü segít csoportosítani a termékeket ezzel könnyebben kezelhetővé és felhasználó barátabbá teszi az oldalt.



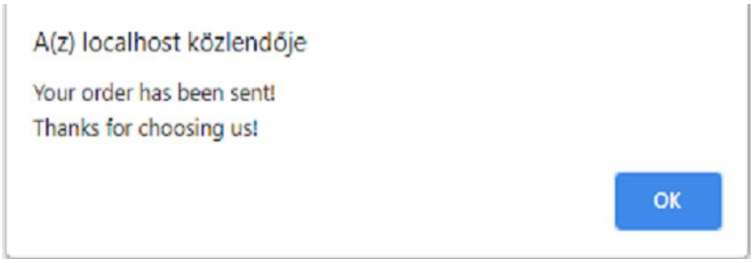
A navbaron a cart menüpontra kattintva a rendelés véglegesítésére és leadására szolgáló oldalra jutunk. Ezen az oldalon is megtalálható a faction oldalakon lévő whitespaceek.

Az oldal tetején az egységes Bootstrap navbar található ez alatt pedig a Bootstrap form inputokból kialakított információ beviteli rész.

Mellette található maga a kosár itt a felhasználó megtekintheti a kosárban eltárolt termékeket a kosár minden termék összárát mutatja.

Ezek a részek alatt található a Place Order(rendelés leadása) gomb amit az oldal színe közül kitűnő kék színnel formáztam ezzel figyelem felhívóbbá téve.

A rendelés leadása után egy alert üzenet tájékoztatja a felhasználót, hogy a rendelés sikeresen leadásra került.



1.4.2 Weboldal működése

A főoldal működéséhez szükségünk van a Bootstrap és a Theme.js ezenfelül még a kapcs.php fájlra.

A fő és aloldalak tetején lévő Navbar menüpontok és a képnézegetőben lévő képek is linkek, amik a tagek között helyezkednek el.

Mivel az oldal Bootstrap segítségével készül ezért sok elem működéséhez JavaScript szükséges mivel ezen elemek előre elkészített és maguktól működőek ezért működésüket nem részletezem.

A téma választó gomb JavaScript segítségével működik A gomb lenyomásakor a JS először a Document.GetElementByid() parancsot használva megkeresi az oldal tetején elhelyezett link element-et és eltárolja a DARK_THEME_LINK változóban.

Ezt a link elemet a weboldal tetején érdemes elhelyezni, de az első stylesheet alatt mivel, ha a betöltött sötét témánk nem tud lekezelni valamit akkor ebben az esetben az eredeti stylesheet fog betöltődni ezzel esetleges hibákat elkerülve.

A gomb működéséhez még megadunk egy sötét téma elérési útját ehhez egy külső webes css fájlt használunk, aminek a linkjét eltároljuk a DARK_THEME_PATH változóban.

```
function toggleTheme() {
  isDark = !isDark;
  if (isDark) {
    enableDarkTheme();
  } else {
    disableDarkTheme();
  }
}
```

A következő function az onclick eseménykor fut le először eldönti, hogy az oldalon dark módban van-e vagy sem és ez alapján lefuttat egy másik functiont. Az isDark változó értékét előzőleg true ként definiáltuk ezért az oldal mindig világos témában töltődik be.

```
function enableDarkTheme() {
  DARK_STYLE_LINK.setAttribute("href", DARK_THEME_PATH);
  THEME_TOGGLER.innerHTML = "🌙 Dark";
  document.getElementsByTagName("body").style.fontSize = "1rem";
}

function disableDarkTheme() {
  DARK_STYLE_LINK.setAttribute("href", "");
  THEME_TOGGLER.innerHTML = "🌞 Light";
```

Az első a sötétmódot bekapcsoló function ami a html kódban elhelyezett link href tulajdonságát beállítja az előzőleg megadott Dark_theme_Path tartalmára

A második a link href tulajdonságát kitörli ezzel vissza állítva minden elemet az eredeti világos témára. Emellett mindkét funkció megváltoztatja a gomb feliratát arra, hogy az az ép aktív témát mutassa.

A főoldalon található öt legújabb terméket tartalmazó Bootstrap kártya mivel ezek az elemeknek dinamikusan változniuk kell ahogy több és több termék kerül a bolt kínálatába ezért ezek létrehozását nem html hanem PHP segítségével hoztam létre.

```php
require ('kapcs.php');
dbkapcs();
$query='SELECT * FROM products ORDER BY ID DESC LIMIT 5';
$eredm=mysqli_query($kapcs,$query);
```

Először a PHP blokkban elhelyeztem az adatbázis kapcsolathoz szükséges két sort az első sorban megadtam, hogy melyik külső fájl tartalmazza a kapcsolódáshoz szükséges parancsokat a második sorban pedig meghívtam a kapcsolódást végrehajtó functiont.

Ezután egy query nevű változóban megadtam azt a mysql query lekérdezést, ami kiválasztja sorrendbe rendezi a termékeket az ID-jük szerint majd kiválasztja a legfelső 5-öt. Mivel a termékek ID-inek számozása 1-től kezdődik így a sorrendbe rendezés után mindig a legújabb termékek lesznek felül.

```php
echo "<div class='row row-cols-1 row-cols-md-5 g-4'>";
while ($sor=mysqli_fetch_array($eredm))
 {
   echo  "<div class='col'>
   <div class='card h-100'>
     <img src=Pictures/".$sor["Picture"]." class='card-img-top'>
     <div class='card-body'>
       <h5 class='card-title'>".$sor["Name"]."</h5>
       <p class='card-text'><span class='badge bg-secondary'>New</span> <br> Product for the ".$sor["Faction"]."
       Faction. <br> PRICE: ".$sor["Price"]." €</p>
     </div>
     <div class='card-footer'>
       <small class='text-muted'></small>
     </div>
   </div>
 </div>";

}
echo "</div>";
```

A szükséges adatok lekérdezése után a létrehoztam a Bootstrap kártyákat erre a PHP azon tulajdonságát használtam hogy egy echo parancs használatával tageket és szöveget is képes megjeleníteni így csak bemásoltam a Bootstrap kártyák felépítését a PHP blokkban lévő while parancsba ezután kicseréltem azokat a szövegrészleteket amiket az adatbázisból akartam legenerálni pl: termék neve, ára.

A while feltételbe a sor változóban eltároljuk a query lekérdezés eredményének egy sorát és végig iterálunk a lekérdezésen így minden sorban lévő adathoz létre hozunk egy-egy kártyát

A termékeket tartalmazó aloldalak működéséhez szükség van a shop.js, a Bootstrap és a Theme.js ezenfelül még a kapcs.php fájlra.

Az aloldalon is tartalmazza a navbart és az ezen található téma választó gombot ezért szükséges a Theme.js használata.

```
$query='SELECT * FROM `products` WHERE Faction LIKE "Space Marines"';
$eredm=mysqli_query($kapcs,$query);

echo "<div class='row row-cols-1 row-cols-md-3 g-4'>";
while ($sor=mysqli_fetch_array($eredm))
{
  echo  "<form action='' method='POST' onsubmit='' id=".$sor["ID"]."><div class='col'>
  <div class='card h-100'>
    <img src=Pictures/".$sor["Picture"]." class='card-img-top'>
    <div class='card-body'>
      <h5 class='card-title'>".$sor["Name"]."</h5>
      <p class='card-text'><span class='badge bg-secondary'>".$sor["Faction"]."</span> <br> Product for the ".$sor["Facti
      Faction. <br>ROLE: ".$sor["Role"]." <br> PRICE: ".$sor["Price"]." € </p>
      <div class='row g-2'>
      <div class='col-sm-6'>
     <button type='submit' class='btn btn-primary gomb' name=gomb >Add to Cart</button>
      </div>

      <div class='col-sm'>
      <input type='number' name='Quantity' class='form-control' placeholder='Quantity' min=1 aria-label='Quantity'>
    </div>

     </div>
    </div>
    <div class='card-footer'>
      <small class='text-muted'><input type=hidden  name='TID' value=".$sor["ID"]."></small>
    </div>
  </div>
</div></form>";
}
```

Ezen az oldalon is használtam a Bootstrap kártyák létrehozásához a főoldalon található PHP kódot annyi változtatással, hogy a legújabb öt termék helyett a megfelelő frakcióhoz tartozó termékeket hívtam le a query lekérdezéssel és azokat hoztam létre a megfelelő Bootstrap tab tartalmi részébe.

Emellett ezek a kártyák a tagjük-ben nem a NEW feliratot, hanem a hozzájuk tartozó frakció-t tartalmazzák. Mivel ezeknek a kártyáknak képesnek kell lenniük a termékek felvezetésére ezért ezek a kártyák külön html formokban lettek létrehozva és tartalmaznak egy mennyiség mezőt, ahol a felhasználó megadhatja, hogy a termékből hány darab kerüljön a kosárba és egy Add to cart feliratú submit típusú gombot.

A termékeket megjelenítő kártyák és a tabs menü fölött lévő picture card képét és szövegét a frakciókhoz tartozó tab-ra kattintáskor a tab tartalmához releváns képre és szövegre cserélődik. Ehhez JavaScriptett használtam.

```
function SMClick() {

        document.getElementsByClassName("Mainkep")[0].src ="Pictures/MainSPM.jpg";
        document.getElementsByClassName("Maintitle")[0].innerHTML="Space Marines<br><br>";
        document.getElementsByClassName("Maintext")[0].innerHTML="Space Marines are traditionall
```

Mindegyik tab menüponton van egy onClick esemény, ami lefuttatja a megadott functiont minden tabnak van egy külön function-je. Ezek a function-ök ugyan azon sémára ápülnek hiszen ugyan azt a feladatott látják el.

A document.getElementByClassName parancs segítségével megkeresik a Mainkép classal ellátott html elemet és beállítja a src tulajdonságot a functionben megadott értékre ez a picture cardon a képet fogja lecserélni.

Ezután a Maintitle elem InnerHTML tulajdonságát, ami a szövegét jelöli cseréli le. A Maintitle tartalmazza a címet

A harmadik sor pedig a másodikhoz hasonlóan cseréli le a picture card szövegét, amit a Maintext classal-ellátott elem tartalmaz.

A termékek kosárhoz való hozzáadását PHP kóddal oldottam meg. De ha csak PHP-t használtam volna az problémákat okoz volna az oldal működésével mivel, ha az, add to cart gombra kattintva lefuttatunk egy PHP felvezetést az adatbázisba akkor az oldal frissül és ez azt okozta volna, hogy a felhasználó visszakerül az All tabre.

```
$(function () {

  $('.gomb').click(function() {


    var adat=$(this).closest('form');

    $.ajax({
      type: 'post',
      url: 'http://localhost/webf/WH40Shop/test.php',
      data: adat.serialize(),
      success: function () {
        alert('Item has been added to the cart!');
      }
    });
    return false;

  });

});
```

Ennek elkerülésére jQuery-t és ajaxot használtam. jQuery segítségével meg keressük a $(this).closest(); paranccsal a felhasználó által megnyomott .gomb class-al ellátott html elemhez legközelebbi form tag-et.

A Jquery ezt úgy éri el, hogy az oldal DOM fáján fel felé kezd el haladni addig ameddig el nem éri a <html> tag-et vagy nem talál egy megfelelő elemet.

Miután talált egy megfelelő elemet azt eltároltam egy adat nevű változóban azután egy ajax({}) parancsot használva elküldjük a külső PHP fájlnak a szükséges adatokat az ajax megfelelő működéséhez megkell adni a kérés típusát jelen esetben ez POST típusú majd egy url-t kell megadni ami a külső PHP fájlra mutat.

Ezután az adatot kell megadnunk, amit az előzőleg megkeresett formból nyertünk ki és az adat változóban tároltam el. Ezen a változón még egy serialize() metódust kell hívni ami egy string-é alakítja az adat változóban tárolt input adatokat.

A function végén egy return false; sort használva megakadályozhatjuk, hogy az oldal frissüljön ezzel elkerülve a frissítés okozta problémákat.

Ezután lefut az urlben megjelölt külső PHP fájl. Ha a PHP parancsok sikeresen lefutottak és a cart táblához hozzá lettek adva a kiválasztott termékeke akkor a jQuery feldob egy alert üzenetet, hogy a terméket sikeresen felvezettük.

```php
<?php
require ('kapcs.php');
dbkapcs();

    $query="INSERT INTO Cart SELECT * FROM Products WHERE ID=$_POST[TID];
    UPDATE cart SET Quantity=$_POST[Quantity] WHERE ID=$_POST[TID]";
    mysqli_multi_query($kapcs,$query);
  ?>
```

A külső PHP kód-ban is megtalálható az adatbázishoz csatlakozáshoz szükséges kapcs.php-t behivatkozó és a dbkapcs() function-t meghívó két sor. Emellett tartalmazza a termék adatbázisba való felvezetéséhez szükséges parancsokat.

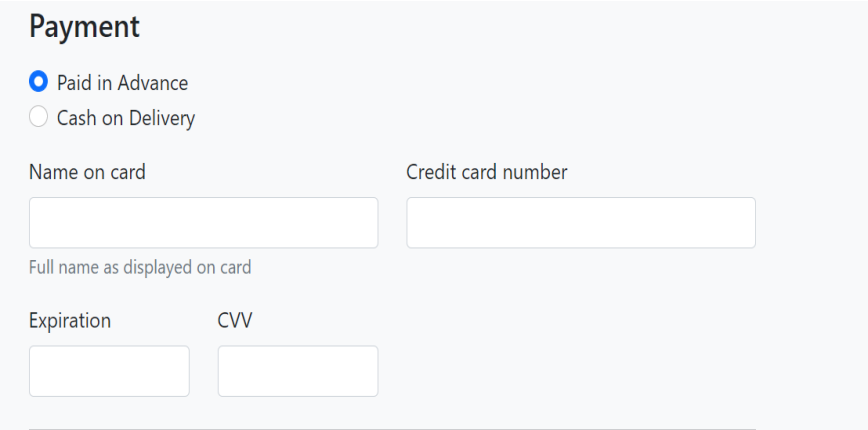
Itt is egy query nevű változóban tároljuk el a mysql által végrehajtandó SQL utasítást, ami jelen esetben egy INSERT INTO parancs, amivel a táblához tudunk adatokat hozzáadni.

Mivel az ajax functionben megadtuk típusnak a POST típust ezért nem kell változtatnunk a kódunkon kezelhetjük úgy mintha nem egy külön fájlban lenne hanem a az eredeti oldal része lenne. Ezért használhatjuk a $_POST[] szintaktikát a formból kinyert adatok felhasználására.

Mivel a termékek táblában minden terméknek már van egy Quantity(mennyiség) tulajdonsága ezért miután hozzáadtuk a terméket a táblához a hozzáadott terméknek a mennyiség tulajdonságát frissítenünk kell egy UPDATE SQL paranccsal, amivel megváltoztatjuk a termék mennyiségét az eredeti raktári darabszámról a vető által megadott megrendelni kívánt darabszámra.

Az adatok megfelelő felvezetéséhez nem csak egy SQL parancsot kell lefuttatnunk ezért ebben az esetben a mysqli_query() metódus nem megfelelő mert az csak egy SQL parancsot tud lefuttatni ezért a mysqli_multi_query() metódust kell használnunk mert az képes több SQL parancs futtatására is.

A cart kosár oldal működéséhez szükségünk van a Bootstrap Javascript és CSS fájlaira emellett a Theme.js és a cart.js JavaScript fájlokra. Az oldal tartalmazza a többi oldalon is elhelyezett navbart ezért szükségesek a bootstrap fájlok és a Theme.js pedig a navbaron lévő téma választó gomb miatt szükséges.



A webshopunk két féle fizetési módot is elfogad ezek az utánvétel tehát az átvételkor való fizetés és a előre fizetés kártyával. A cart.js azért felelős, hogy a felhasználó választása szerint változtassa, hogy milyen mezők jelennek meg az oldalon mivel utánvétel esetén a felhasználónak nem kell megadnia bank kártya adatokat.

```
function CoD() {
document.getElementById("pay").innerHTML="";
}

function PiA() {
    document.getElementById("pay").innerHTML="<div class='row gy-3'>\
    <div class='col-md-6'>\
      <label for='cc-name' class='form-label'>Name on card</label>\
      <input type='text' class='form-control' id='cc-name' placeholder=
      <small class='text-muted'>Full name as displayed on card</small>\
      <div class='invalid-feedback'>\
        Name on card is required\
      </div>\
```

A cart.js fájlban két JavaScript funkcion található a CoD function akkor fut le ha a felhasználó a Cash on Delivery utánvétel fizetési módott választja ilyenkor a pay idvel ellátott div tartalmát kitöröljük mivel nincs szükségünk további információra a felhasználótól azonban ha a felhasználó a Paid in Advance tehát az előre fizetést választja akkor a PiA function fut le ami a divben megjeleníti a kártya adatok beviteléhez szükséges mezőket.

Ahhoz, hogy a pay idvel ellátott mezők tartalmát megtudjuk változtatni először egy Document.GetElementById() metódussal megkell keresnünk azt a dokumentumban majd a választástól függően meg kell változtatnunk a megkereset elem InnerHTML tulajdonságát, ami a z elem tartalmát fogja megváltoztatni.

```php
if (isset($_POST['Rem'])) {
  $Delquery="DELETE FROM Cart WHERE ID='$_POST[TID]';";
  mysqli_query($kapcs,$Delquery);
  echo "<meta http-equiv='refresh' content='0'>";
}
```

Az oldal van lehetőség a kosárhoz adott termékek eltávolítására ezt PHP segítségével oldottam meg. Az oldal jobb oldalán megjelenő a kosárban lévő termékeket megjelenítő részben minden termék egy-egy formban van, ami egy hidden típusú inputban tárolja a megjelenített termék Id-jét. Így amikor a felhasználó a termék mellet megjelenő X-re kattint és lefut a php az ezt az id-t felhasználva kitörli a cart táblából azt a terméket. Miután kitörölte a terméket azután a kód frissíti az oldalt, hogy az oldalon is látszódjon a változás.

```php
if (isset($_POST['Orderbutton']))
{

  $nev=$_POST['firstname']." ".$_POST['lastname'];
  $address=$_POST['city']." ".$_POST['street']." ".$_POST['number'];
  $pay="";
  $order="";
  $datetime_variable = new DateTime('NOW');
  $datetime_formatted = date_format($datetime_variable, 'y-m-d');
  if ($_POST['paymentMethod'] == 'Cash on Delivery') {
    $pay="Cash on Delivery";
  }else {
    $pay="Paid in Advance";
  }
```

A rendelés gomb lenyomásakor először kinyerjük a formból az adatokat és ezeket eltároljuk változókban.

A fizetési módszert egy if elágazás segítségével tudjuk meghatározni a rendelés dátumát pedig egy new DateTime('NOW') parancs segítségével tudjuk meghatározni, ami épp a jelenlegi időt fogja megadni a rendelés idejének.

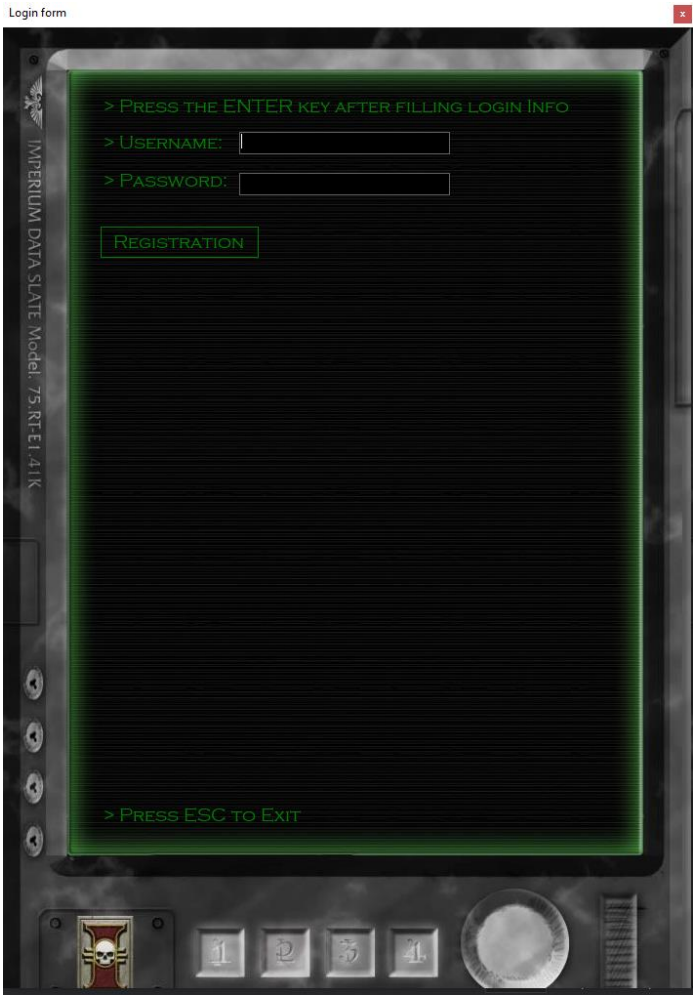
```php
$query2='SELECT * FROM Cart ORDER BY Price DESC';
$eredm2=mysqli_query($kapcs,$query2);

while ($sor2=mysqli_fetch_array($eredm2))
{
 $order=$order.$sor2['Quantity']."x ".$sor2['Name'].";";
}



    $query="Insert Into orders values
    ('','$nev','$_POST[email]','$_POST[phone]','$address','$pay','$dat
    mysqli_query($kapcs,$query);
```

Még mielőtt felvezethetnénk a rendelést először egy változóba kell egyesítenünk és formáznunk a kosárban található termékeket. A termékeket először ár szerint csökkenőbe rendezük majd egy egységes minta alapján megformázzuk és hozzáadjuk a táblához. A formázásra azért van szükségünk mert a rendelt termékeket egy cellában tároljuk és az asztali alkalmazás így tudja használni.

1.5 Asztali alkalmazás/program

Az asztali alkalmazás főbb funkciói egy bejelentkező felület, egy regisztrációs felület, egy adminisztrációs felület és egy a bolt vásárlói által elérhető vásárlást segítő rész. Ezen funkciók megvalósítása volt a főbb feladatom. Emellett egy design kialakítása a program számára.

1.5.1 Program felépítése



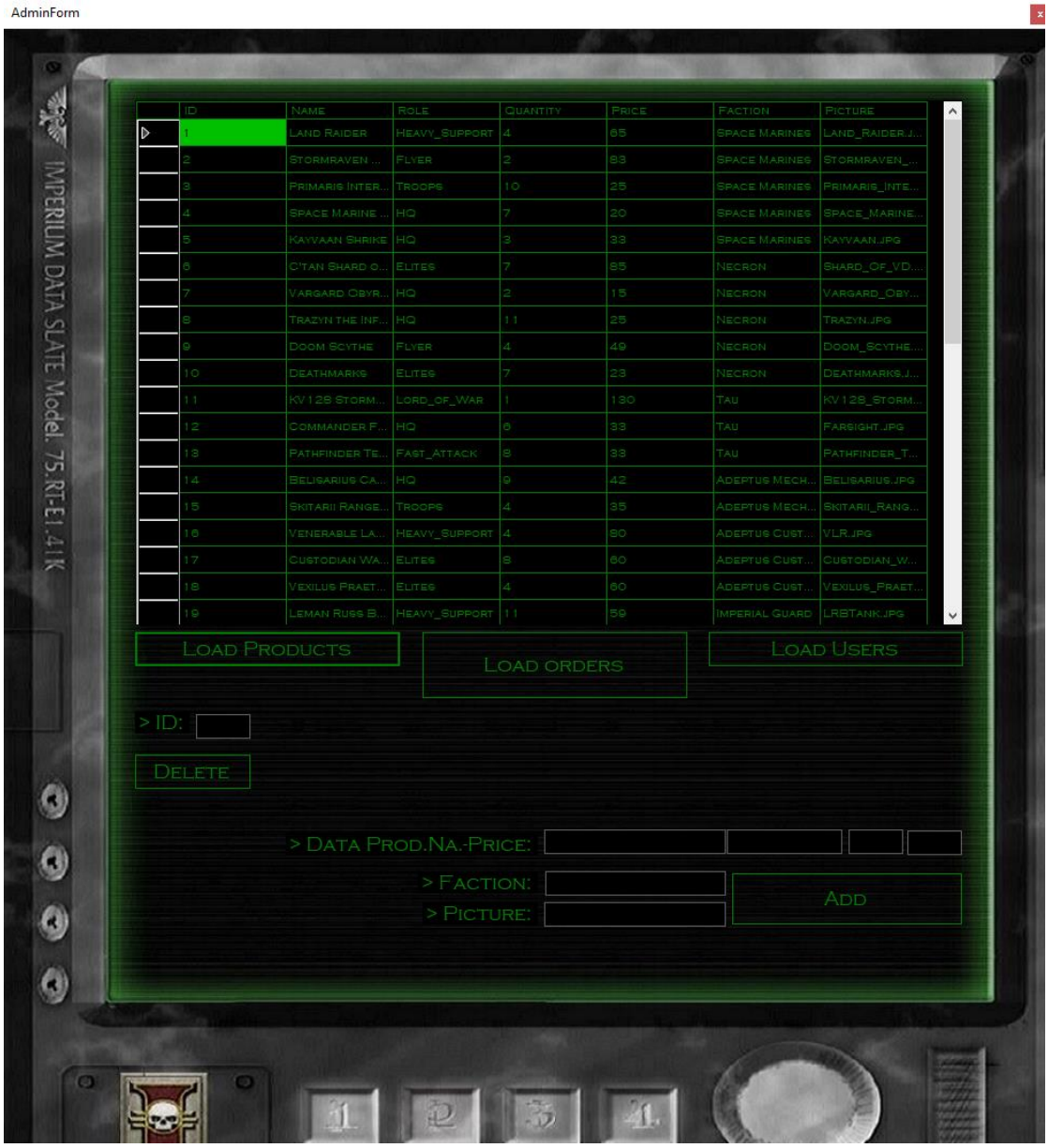
Mivel az első feladatunk a felhasználó beléptetése ezért az első form, a kezdő ablak a beléptetésért felelős.

A felhasználók azonosításához két adatra van szükségünk egy felhasználó névre és egy jelszóra ezeket a felhasználótól két textbox-on keresztül kérjük be.

Emellett elhelyeztem több labelt amit a designhoz illően formáztam. A kezdő ablakon még található egy a felhasználó által láthatatlan DataGridView amire a program működéséhez van szükségünk.

Az enter lenyomása után automatikusan a megfelelő formra vált az alkalmazás. Ha a felhasználó nem megfelelő adatokat add meg akkor erre az alkalmazás figyelmezteti és nem lépteti be, ha valami más hiba következet be akkor egy másik figyelmeztetés jelenik meg.

Ha a felhasználó nem rendelkezik fiókkal akkor a regisztráció gombra kattintva megnyílik egy regisztrációs felület. Ha egy másik formot bezárunk akkor erre a bejelentkező ablakra fog vissza irányítani minket az alkalmazás



Ha admin felhasználóval lépünk be akkor az adminisztrációs felületre nyílik meg. Ezen a formon van egy nagy DataGridView amit a táblázatban található adatok megjelenítésére használunk. A táblák között a három gomb használatával válthatunk a formon pedig dinamikusan fog változni ahogy más-más táblát töltünk be.

De a delete gomb és a hozzá tartozó textbox az mindig ott marad mert minden táblából ugyan úgy tudunk törölni. A felviteli rész változik mivel a táblák között nem inden mező egyező így más felviteli adatokat kell megadni.

Azonban az order rendelésekhez nem tudunk az alkalmazáson belül rendelést hozzáadni, de a kész rendelést tudjuk törölni ezért itt nem egy felviteli rész, hanem egy lista jelenik meg ahol láthatjuk a rendelés teljes tartalmát.

Ha nem egy admin felhasználóval lépünk be akkor a felhasználói felületre irányít minket az alkalmazás. Itt a program egy labelben üdvözli a felhasználót alatta pedig egy DataGridView-ban megjeleníti a felhasználó email címéhez tartozó rendeléseket. Alatta pedig a rendelésekhez kötődő információk találhatóak.

Ez alatt pedig a detachment designer rész található, ahol egy kis leírást található alatta pedig a designer lista része, ahol összeállíthatjuk egy lista a megvásárolni kívánt termékeket, amiket majd az export gombbal egy txt-be írhatunk

A listák között a könnyebb kezelhetőség érdekében a listák között a lista elemekre kattintva lehet őket kiválasztani és a listához adni.

A regisztrációs felület nagyon hasonló a bejelentkező felülethez, de itt még található egy email mezőis. A regisztráció gomb megnyomása után az alkalmazás automatikusan visszairányít a kezdő login felületre, ha a felhasználó bezárná az ablakot akkor is ez történik.

Összeségében minden ablakból el lehet jutni a kezdő login ablakba a könnyebb kezelhetőség érdekében. Emellett minden ablak desgin-ja ugyan olyan, de a funkciójuk alapján változó méretűek.

1.5.2 Program működése

A program működéséhez szükségünk van egy NuGet[2] package-re ami lehetővé teszi a program és a mySQL közti kapcsolat létrehozását és az adatbázis módosítását ehhez a Visual Studion belül elérhető és letölthető MySql.Data NuGet package-et fogom használni.

```
public void openConnection()
{
    if (connection.State == ConnectionState.Closed)
    {
        connection.Open();
    }
}
4 references
public void closeConnection()
{
    if (connection.State == ConnectionState.Open)

    {
        connection.Close();
    }
}
```

Minden formban megtalálható 3 az adatbázisokkal valómunkát könnyebbé tevő metódus ezek az OpenConnection(), a CloseConnection() és az executeQuery() az első kettőnek feladata hogy nyissa és zárja az adatbázishoz a kapcsolatot ezt úgy teszi hogy először egy if feltételben eldönti hogy a kapcsolat épp megvan-e nyitva vagy be van-e zárva majd ha ez true igaz értéket ad vissza akkor a connection.Open() paranccsal megnyitja vagy a connection.Close() paranccsal bezárja a kapcsolatott.

```
public void executeQuery(string query)
{
    try
    {
        openConnection();
        command = new MySqlCommand(query, connection);
        if (command.ExecuteNonQuery() >= 1)
        {

        }
        else
        {
            MessageBox.Show("Sikertelen végrehajtás");
        }
    }
    catch (Exception kivétel)
    {

        MessageBox.Show(kivétel.Message);
    }
    finally

    {
        closeConnection();
    }
}
```

Az executeQuery metódusnak egy feltétele van egy stringet kér ez lesz a sql query amit lefog futtatni. Először megnyitja a kapcsolatot az adatbázissal majd egy újpéldányt hozz létre a MySqlCommand classból aminek a query-t és a connection stringet átadva az lefuttatja az utasítást az adatbázison. Ezután pedig mindig lezárja a kapcsolatot.

Ha valami hiba következne be a csatlakozás során vagy a query futtatása során azt a metódus jelzi és kiírja a hiba üzenetet. Abban az esettben is, ha a query lefuttatható de egy sor sem érintett futása során tehát nem végzet semmilyen módosítást ez segít a szemantikai hibák kijavításában és észlelésében.

Az első formon található egy pressdown event ami akkor fut le ha a felhasználó lenyom egy gombot ez az event fogja elindítani a bejelentkeztetési folyamatot. Először egy if feltétel segítségével eldönti, hogy a felhasználó melyik gombot nyomta le és ha a felhasználó a megadott jelen esetben az enter gombot nyomta meg akkor a program eltárolja a két textboxban levő szöveget kettő string változóban.

```
if (e.KeyCode == Keys.Enter)
{
    string UserName = UserText.Text;
    string Password = PasswordText.Text;
    try
    {
        MySqlDataAdapter adapter = new MySqlDataAdapter($"SELECT * FROM Users WHERE Username='{UserName}' " +
            $"AND Password='{Password}' ", connection);
        openConnection();
        DataSet ds = new DataSet();
        adapter.Fill(ds, "Users");
        HelperGrid.DataSource = ds.Tables["Users"];
        closeConnection();
    }
    catch (Exception hiba)
    {

        MessageBox.Show(hiba.Message);
    }
```

Ezután egy try-catchen belül a Mysql.Data NuGet package ből elérhető MySqlAdapter classból létrehoz egy új példányt ennek a classnak meg kell adni egy stringet ami a SQL parancsot tartalmazza és megkell adni egy connection stringet is.

Miután létrehozta az adaptert azután meghívja a kapcsolatot nyitó metódust, amit létrehoztam, ezután létrehoz egy Dataset-et ezzel a datasettel feltöltjük az adaptert a fill metódus segítségével itt meg kell adnunk melyik táblából akarjuk az adatokat kinyerni. Végezetül pedig a DatagridView nak megadjuk, hogy a Datasourcea a Datasettünk User táblája legyen és lezárjuk a kapcsolatott.

```
if (HelperGrid.Rows[0].Cells[0].FormattedValue.ToString() == "")
{
    MessageBox.Show("Sikertelen bejelentkezés! Nem megfelelő adatok.");

}
else
{
    if (HelperGrid.Rows[0].Cells[4].FormattedValue.ToString() == "True")
    {

        var Admin = new AdminForm();
        Admin.Show();
        this.Hide();
    }
    if (HelperGrid.Rows[0].Cells[4].FormattedValue.ToString() == "False")
    {
        string Email = HelperGrid.Rows[0].Cells[3].FormattedValue.ToString();
        string name = UserText.Text;
        var User = new Userform(name, Email);
        User.Show();
        this.Hide();
    }
}
```

Miután feltöltöttük a helper gridünket azután megvizsgáljuk, hogy talált-e megfelelő elemet a SQL querry ha nem az azt jelenti hogy nem megfelelő adatott adott meg a felhasználó vagy még nem regisztrált erre figyelmezteti a program.

Ha pedig talált akkor megvizsgáljuk, hogy a gridview első sorának utolsó cellája true vagy false értéket tartalmaz-e tehát hogy a felhasználó admin jogokkal rendelkezik-e ezután pedig tovább irányítjuk a megfelelő ablakra úgy, hogy nyitunk egy új példányt a szükséges formból.

```
private void Button_ProdLoad_Click(object sender, EventArgs e)
{
    OrdersList.Visible = false;
    productsloaded = true;
    OrdersAreloaded = false;
    try
    {
        MySqlDataAdapter adapter = new MySqlDataAdapter("SELECT * FROM Products", connection);
        openConnection();
        DataSet ds = new DataSet();
        adapter.Fill(ds, "Products");
        dataGridView1.DataSource = ds.Tables["Products"];
        closeConnection();
    }
    catch (Exception hiba)
    {

        MessageBox.Show(hiba.Message);
    }

    foreach (var item in Product_Add_controlls)
    {
        item.Visible = true;
    }
    foreach (var item in Users_Add_controll)
    {
        item.Visible = false;
    }

}
```

Az adminisztrációs formban a három táblához tartozó gombok lenyomásakor a gombhoz tartozó tábla töltődik be itt is ugyan azt a kódot használtam, mint a bejelentkező felületen csak a mivel itt nem 1 sort keresünk, hanem minden adatott ezért más a SQL parancs, amit az adapternek megadunk és a kiíratni kívánt tábla is.

Az adatok megjelenítésén kívül még megjelenítjük a táblához tartozó felvitelhez szükséges mezőket vagy az order tábla esetében az orderhez tartozó listboxot ezt úgy érjük el, hogy azen elemek visible tehát láthatóság tulajdonságát false-ra vagy true-ra állítjuk attól függően, hogy melyik tábla van betöltve.Ezenfelül még megváltoztatjuk, hogy melyik tábla van betöltve ezt úgy tesszük, hogy a deklarált bool típusú változókat a gomb lenyomásakor true-ra vagy false-ra állítjuk attól függően, hogy melyiket nyomtuk meg.

A törléshez szükséges UI elemek mindig láthatóak ezért valamilyen módon el kell döntenünk, hogy ép melyik tábla van betöltve ehhez a Ordersareloaded és a productsareloaded bool változókat használjuk. A törlés gomb lenyomása után megvizsgáljuk melyik bool true és az alapján lefuttatunk egy törlést az adatbázis megfelelő tábláján és kitöröljük a felhasználó által a textboxban megadott ID-vel rendelkező recordot.

Az order táblához nincs adat felvezető felület, de tartozik egy listbox amihez hozzáadjuk a felhasználó által kiválasztott sor utolsó mezőjéből kinyert elemek. Ehhez hozzá kell adni a

DataGridViewhoz egy cellClick eventet ami akkor fogja lefuttatni a benne lévő kódot ha a felhasználó egy a gridviewban levő mezőre kattint. Miután megvizsgáltuk hogy az orderstábla van-e betöltve egy if feltétellel ezután egy stringgé convertáljuk majd a .Split(';') metódus segítségével szét daraboljuk az utolsó mező tartalmát és ezeket a listához adjuk.

A bejelentkezés után, ha a felhasználónak nincs adminisztrációs jogosultsága akkor a userform nyílik meg. Amikor a bejelentkező form megnyitja a userformot akkor átadja neki a bejelentkezett személy nevét és email címét ez azért szükséges mert a userformban az egyedi üdvözlő szövegben megjelenik a be jelentkezet felhasználó neve az email cím pedig a felhasználó rendeléseit megjelenítő DataGridViewhoz szükséges.

A DataGridView feltöltése itt is úgy történik, mint az első bejelentkező formban csak a SQL feltétel más mert itt email cím szerint kell kiszűrni a megfelelő adatokat a táblából. A refresh gomb újra lefuttatja az adatok betöltéséhez szükséges Mysql.Data parancsokat.

Ezután a Detachment designer található, ami 4db listából és egy export gombból áll. Mind a 4 lista selecteditemchange event esetén futtatja le a kódját. Az első lista elemeit egy csv fájlból nyertem ki, ami a Detachments adatait tartalmazza úgy hogy a fájl sorait soronként beolvastam a File.readalllines() parancs segítségével és pontos vesszőn ként szét daraboltam majd minden sor első elemét hozzáadtam a listához.

Ha a felhasználó kiválaszt egy elemet az első listából akkor egy forba ágyazott if feltétellel végig megyünk a beolvasott csv fájl sorait tartalmazó string listán és kiválasztjuk a megegyező névvel kezdődő sort és a requirements(Elvárások) listához adjuk a sorból split() paranccsal kinyert adatokat.

```
1 reference
private void Requirements_SelectedValueChanged(object sender, EventArgs e)
{
    Units.Items.Clear();
    string role = Requirements.SelectedItem.ToString().Split()[0];
    MySqlDataAdapter adapter = new MySqlDataAdapter($"SELECT * FROM Products WHERE Role='{role}'", connection);
    openConnection();
    DataSet ds = new DataSet();
    adapter.Fill(ds, "Products");
    Helpergrid.DataSource = ds.Tables["Products"];
    closeConnection();
                (local variable) DataSet ds

    for (int i = 0; i < Helpergrid.Rows.Count; i++)
    {
        var row = Helpergrid.Rows[i];
        Units.Items.Add($"{row.Cells[1].FormattedValue} Role:{row.Cells[2].FormattedValue}");
    }
    Units.Items.RemoveAt(Units.Items.Count-1);

}
```

Ezután ha a felhasználó kiválasz egy elemet a requirements közül akkor a harmadik listába egy SQL query segítségével

betöltjük hogy mely termékek felelnek meg a feltételnek ezután ha a felhasználó a terméket kiválasztja akkor azt a negyedik listához adjuk egy .Add() metódus segítségével. Ha felhasználó a saját listájában kiválaszt egy terméket akkor azt a .remove() metódus segítségével eltávolítjuk a listából.

Az export gomb megnyomásakor egy foreachel végig megyünk a yourlist nevű listán és minden elemét egy .ToString() metódussal stringé konvertáljuk majd a Createdlist string listához adjuk ezután egy txt fájlba kiírjuk a created list tartalmát File.WriteAlllines()

```
private void Registration_Click(object sender, EventArgs e)
{
    if (UserText.Text!="" && PasswordText.Text!="" && Emailtext.Text!="")
    {
        string reg = $"INSERT INTO Users VALUES ('','{UserText.Text}','{PasswordText.Text}','{Emailtext.Text}','0')";
        executeQuery(reg);

        this.Close();
    }
    else
    {
        MessageBox.Show("You must fill all fields!");
    }
}
```

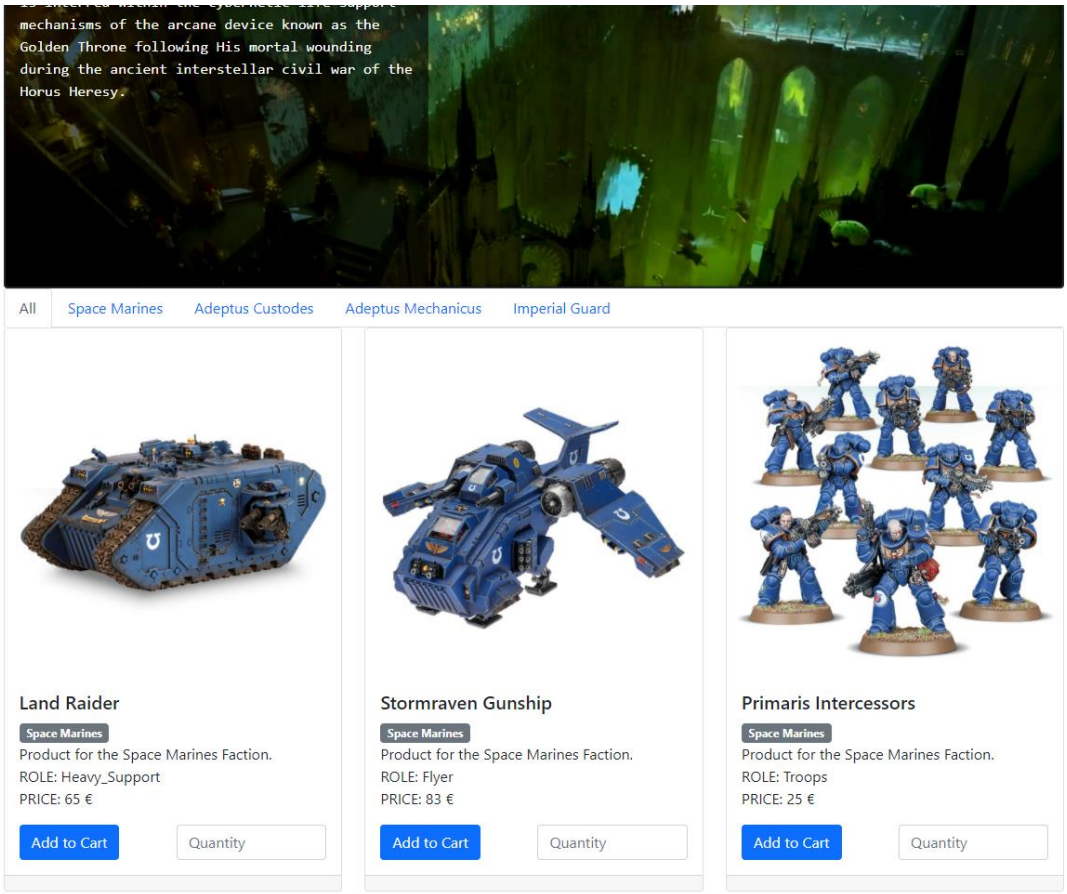
paranccsal.

Ha felhasználónak még nincs fiókja akkor a bejelentkező formról a registration gomb megnyomásával megnyílik egy Registrationform. Ezen a formon először a registration gomb lenyomása után egy if feltételben ellenőrízük hogy minden textboxban van-e beírt szöveg tehát hogy a felhasználó kitöltött-e mindent Ha nem töltött ki akkor azt jelezzük ha igen akkor egy stringbe felvesszük az adatok felviteléhez szükséges SQL parancsot majd a executeQuery() metódus segítségével ezt lefuttatjuk.

2 Felhasználói dokumentáció

2.1 Weboldal

A weboldal megnyitásához és az adatbázisok megtekintéséhez a Xampp-ra van szükségünk. A főoldalon egy a galériában lévő képre kattintva át juthatunk az ahhoz kapcsolódó aloldalra vagy a felső navbarból is kiválaszthatjuk a factions lenyitása után.



Ezután az aloldalon a leírást tartalmazó kép kártya alatt lévő fülekkel lehet szűrni a termékek között vagy ki kereshetjük az összes termék fülön is miután megtaláltuk a nekük tetsző terméket beírjuk a mennyiséget és az Add to cart gombra kattint és a sikeres hozzáadásról szóló értesítés elfogadása után folytathatjuk a termékek böngészését vagy a navbaron a cart menü pontra kattintva a rendelés oldalra átlépve leadhatjuk a rendelésünk. Ugyan ezen az oldalon tudjuk megtekinteni a kosár tartalmát és a termék neve mellett lévő X gombra kattint tudjuk azt eltávolítani a kosárból.

A rendelés oldalon meg kell adnunk a személyes adataink és ki kell választanunk a fizetési módot miután mindent kitöltöttünk azután a Place my Order gombra kattintva leadhatjuk rendelésünk.

2.2 Asztali alkalmazás/program

Az asztali alkalmazás működéséhez szükség van a Detchments.csv fájlra és a Xampp programra az adatbázisok eléréséhez. Az alkalmazás alacsony rendszer követelményekkel rendelkezik egy quad core processor, 4GB Ram és egy integrált GPU is képes probléma nélkül futtatni.

Ha bezárunk egy ablakot, ami nem a kezdő bejelentkező ablak akkor automatikus vissza kerülünk a bejelentkeztető felületre. A bejelentkezető felületről az ESC gomb megnyomásával vagy az ablak bezárásával lehet kilépni.

Az alkalmazás elindulásakor a bejelentkező felületen, ha rendelkezünk fiókkal akkor megadjuk a regisztrált felhasználó nevük és jelszavunk ezután a program átirányít az adminisztrációs felületre vagy a vásárlói felületre.

Azonban, ha nem rendelkezünk felhasználóval létrehozhatunk egyet a registration gomb lenyomása után meg nyíló ablak mezőinek kitöltésével és a registration gomb lenyomásával. A regisztráció után automatikusan visszakerülünk a bejelentkező felületre. A regisztrált felhasználónk azonnal használható.

A vásárlói felületen megjeleni az össze olyan rendelés, ami t a regisztrált email címünkkel rendeltünk. A refresh gombbal frissíthetjük a rendeléseket tartalmazó táblát.

Az ablak alján a Detachment listából egy Detachmentet kiválasztva a requirements(Elvárások) listába megjelennek a hogy milyen egységeket kell tartalmaznia. A requiremenst listából kiválaszt hatunk egy Role-t(Szerepet) és a Units listában megjelenik az összes megfelelő termék ezután a terméknevére kattintva hozzáadhatjuk a saját listánkhoz így össze állítva a saját Detachmentünk.

Ha elszeretnék távolítani egy terméket akkor a saját listánkban rá kell kattintanunk. Az export gomb megnyomásakor a program a saját listánkat egy szöveges txt fájlba exportálja, amit később felhasználhatunk.

A regisztrációs felületen három tábla betöltése közül tudunk választani a rendelések a felhasználók és a termékek. Miután betöltöttük a táblát a három gomb egyikével a Delete(törlés) gomb felett lévő mezőben megadhatjuk a törlni kívánt record ID jét és a delete gombbal törölhetjük a tábla automatikusan frissül.

A táblák között a mezők típusa és a mezős száma is különbözik ezért minden táblának egyedi felvezetési felülete van, ami a tábla betöltésekor meg is jeleni itt a mezők kitöltése után a Add gombra kattintva hozzáadhatjuk a táblához.

A rendelés tábla esetén nincs lehetőségünk rendelés hozzáadására, de ha egy mezőre kattintunk amikor a rendelések tábla van betöltve akkor a rendelt termékeket egy listába helyezi a program a jobb láthatóság érdekében.

3 Irodalomjegyzék

[1] Bootstrap Docs

https://getbootstrap.com/docs/5.0/getting-started/introduction/

(Utolsó megtekintés: 2021. 04. 07.)

[2] MySql.Data NuGet Package

https://www.nuget.org/packages/MySql.Data/

(Utolsó megtekintés: 2021. 04. 06)

[3] StackOverflow Questions

https://stackoverflow.com/questions

(Utolsó megtekintés: 2021. 04. 07)

NYILATKOZAT

Alulírott ……..…… nyilatkozom, hogy a záródolgozatomban foglalt tények és adatok a valóságnak megfelelnek, és az abban leírtak a saját munkám eredményei.

Kelt: Makó, 202_. év _____ hó ___. nap

_____
Aláírás

HSZC
Hódmezővásárhelyi Szakképzési Centrum

OM azonosító: 203039/007

MAKÓI NÁVAY LAJOS
TECHNIKUM ÉS KOLLÉGIUM
6900 Makó, Posta u. 4-6.

Tel./Fax.: 62/510-917
e-mail: keritikarsag@gmail.com

KONZULTÁCIÓS LAP
2020/2021. tanév

Tanuló neve:	
Osztálya:	5/13/1
Szak:	Szoftverfejlesztő

Téma/feladat	Időpontja	Konzulens(ek) aláírása
1. Témaválasztás (a dolgozat címének leadása) 2020. 12. 18.		
2. A szerkezeti vázlat (témavázlat) bemutatása 2021. 01. 08.		
3. 10 oldal megküldése (e-mail-ben) a konzulens szaktanár(ok)nak véleményezésre 2021. 02. 08.		
4. 20 oldal megküldése (e-mail-ben) a konzulens szaktanár(ok)nak véleményezésre 2021.03.08.		
5. A záró dolgozat beadása szaktanári véleményezésre 2021. 04. 08.		
6. A záródolgozat rendelkezésre bocsátása előzetes szakmai bírálattal együtt a vizsgabizottság elnökének a komplex vizsga előtt legalább 10 nappal		