**Szoftverfejlesztő és-tesztelő vizsgaremek**

**Fejlesztői dokumentáció**

**2023**



**Készítette:** Gimesi Ádám, Harka Szilárd, Lippai Ádám

**Tartalomjegyzék**

[Bevezetés 3](#_Toc133354210)

[Fejlesztőkörnyezet 4](#_Toc133354211)

[Hardver követelmények: 4](#_Toc133354212)

[Szoftver követelmények: 4](#_Toc133354213)

[Felhasznált Szoftverek 4](#_Toc133354214)

[Index 6](#_Toc133354215)

[CSS 7](#_Toc133354216)

[Árlista 8](#_Toc133354217)

[Galéria 11](#_Toc133354218)

[Adatbázis-módosítás 12](#_Toc133354219)

[Login 14](#_Toc133354220)

[Adatbázis 19](#_Toc133354221)

[Tesztelés 20](#_Toc133354222)

# Bevezetés

A dokumentum célja, hogy bemutassa a Viktória Hairstyle weboldal tervezését és fejlesztését. A weboldal célja, hogy modern és könnyen használható felületet nyújtson a fodrászati szolgáltatások igénybevételéhez és az ügyfélkapcsolatok kezeléséhez. Az ügyfélkör bővítése és Viktória munkájának támogatása a fő motiváció a weboldal létrehozására.

A dokumentáció részletesen ismerteti a weboldal fejlesztési folyamatát, a felhasznált technológiákat, valamint a hardver és szoftver követelményeket. Emellett bemutatja az adatbázis tervezését és felépítését, valamint a weboldal legfontosabb funkcióit és használatukat.

A fejlesztési folyamatról szóló részletes beszámoló mellett az eszközöket és technológiákat is ismertetjük, amelyekkel a fejlesztői csapat a weboldalt létrehozta.

Bízunk benne, hogy ez a dokumentáció segíti Viktória Hairstyle-t a weboldal hatékony működésében, és segít bővíteni az ügyfélkört, miközben további értéket és elégedettséget kínál a fodrászati szolgáltatásokat igénybe vevők számára.

# Fejlesztőkörnyezet

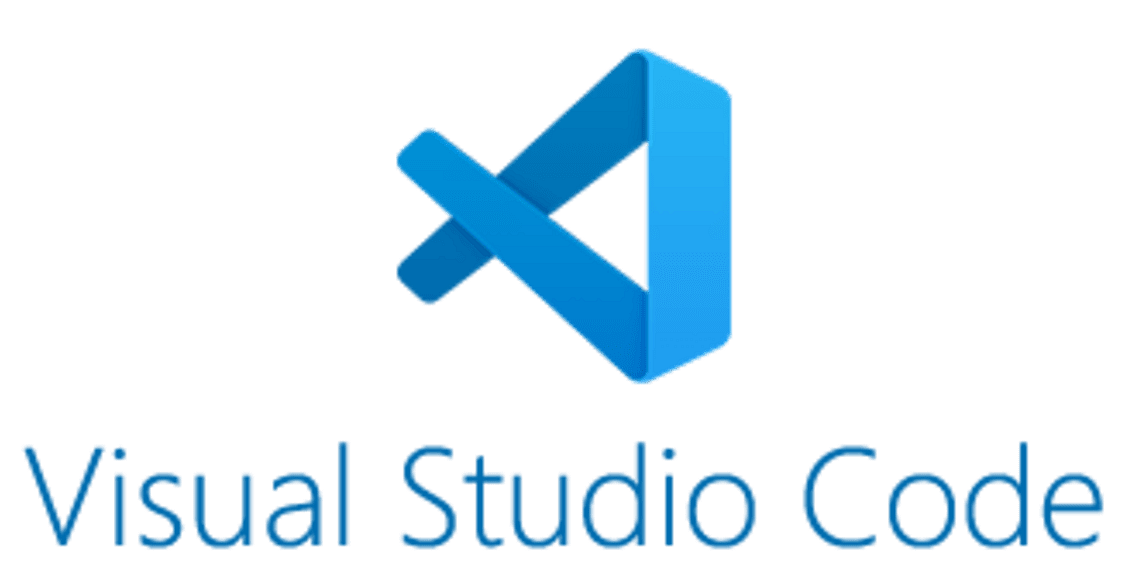
## Hardver követelmények:

* 4 Gigabyte memória
* 30 Megabyte tárhely
* Operációs rendszer (Windows, Linux, Mac)

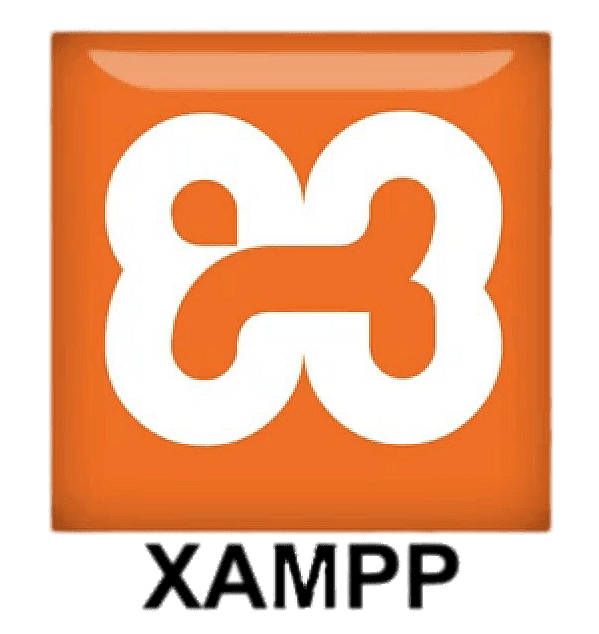
## Szoftver követelmények:

* Webböngésző (Chrome, Firefox, Opera, Microsoft Edge)
* XAMPP Control Panel v3.3.0
* PHP 8-as verzió

## Felhasznált Szoftverek

**Visual Studio Code V1.77**

A vizsgaremek nagy része ezt a szoftvert használva íródott. Egy hatékony, könnyű és ingyenes kódszerkesztő, amely integrált eszközöket biztosít a tesztelésre, illetve az egymással történő megosztáshoz.

**XAMPP Release 8.2.4**

A XAMPP teljesen ingyenes, könnyen telepíthető Apache disztribúció MariaDB, PHP és Perl komponenseket tartalmazva. Az XAMPP-ot használtuk a PHP tesztelésére, illetve az adatbázis létrehozásához.



**phpMyAdmin V5.2.0**

A phpMyAdmin egy PHP nyelven írt ingyenes szoftvereszköz, amely a MySQL webes adminisztrációjának kezelésére szolgál. A phpMyAdmin a MySQL és a MariaDB műveletek széles skáláját támogatja. A vizsgaremek adatbázisa is ebben készült.

**Bootstrap** **v5.3.0-alpha3**

A Bootstrap egy nyílt forráskódú keretrendszer (framework), mely HTML, CSS, JavaScript technológiákat használ. A pricelist.php-ban a modal ablakok is ezt a keretrendszert használják.

Weboldalunk **HTML5** segítségével készült!



Weboldalunk **CSS3** segítségével formáztuk!

Weboldalunkat **JavaScript** segítségével bővítettük! (database.php)

****

**Canva V1.63**

A Canva egy könnyen használható grafikai tervező alkalmazás, amely lehetővé teszi a felhasználók számára személyre szabott dizájnok létrehozását bármilyen célra. A weboldalunkon lévő képek, logók, „effektek” is ebben készültek.

# Index

Az index.html egy alapvető weboldal sablon, amelynek célja bemutatni egy fodrászat szolgáltatásait és elérhetőségeit. Az oldal HTML, CSS, nyelveket használ.

Az index.html fájl az oldal főoldalát tartalmazza. A dokumentum a következő HTML tag-ekből áll:

**<!DOCTYPE html>**: Az oldal dokumentum típusát határozza meg. Ebben az esetben az HTML dokumentum típusáról van szó.

**<html lang="hu">**: A weboldal nyelvének beállítása, amely a hu rövidítést használja a magyar nyelv jelölésére.

**<head>**: Az oldal fejrészét határozza meg. Itt találhatók a meta információk, a CSS és a JavaScript fájlok, valamint az oldal címe.

**<meta>**: Az oldal metaadatokat tartalmaz, mint például a karakterkódolás, az oldal leírása és kulcsszavai a kereső motornak, valamint az oldal kompatibilitása a böngészőkkel.

**<link>**: Az oldal CSS fájljait tartalmazza, amelyek a Bootstrap keretrendszerből, valamint a Google Fonts API-ból származnak.

**<title>**: Az oldal címe.

**<header>**: Az oldal fejlécét tartalmazza, amely magában foglalja a navigációs menüt és a logókat.

**<div>**: Az oldal tartalmát tartalmazza. Itt találhatók az üdvözlő üzenet, valamint az idézet Audrey Hepburn-től.

**<footer>**: Az oldal láblécét tartalmazza, amely magában foglalja a nyitvatartási időt és az elérhetőségi információkat.

A weboldal funkcionalitása kiterjed az adatbázisban található árlista, valamint a galéria oldalra való átirányításra. Az oldal alapvető funkciói a navigáció a fejlécben található menüsáv segítségével és az elérhetőségi információk megtekintése a láblécben található táblázatban.

Az index.html fájl tartalmaz egy elérhetőséget is, amely lehetővé teszi az ügyfelek számára, hogy kapcsolatba lépjenek a fodrásszal.

# CSS

Az általános leírás a style.css fájlra vonatkozóan az, hogy az összes elem margin és padding értéke 0, valamint a dobozmodell border-box-ra van állítva, ami azt jelenti, hogy a margó és padding értékek már a szélesség és magasság részét képezik az elemeknek, így nem lesznek kiszorítva más elemektől.

Ezenkívül a CSS fájl tartalmazza az oldal szín- és designelemekre vonatkozó részleteket, például a háttérszín, szegélyek, vonalak, betűtípusok és méretek meghatározását.

Az adaptív megoldásokat illetően a style.css fájlban található stílusok kialakítása és elrendezése oly módon történik, hogy azok a különböző képernyőméretekre és eszközökre való adaptálást lehetővé tegyék. Például az üdvözlőképernyőnek a közepén található kép a képernyő szélességének 100%-át elfoglalja, így bármilyen képernyőmérethez jól illeszkedik.

A reszponzív megoldásoknak az a célja, hogy az oldal tartalma és elrendezése dinamikusan változzon az eszközök képernyőméretének változásaival együtt. A style.css fájlban például a navigációs sáv magassága és szélessége, valamint az üdvözlőképernyőn található kép mérete és elhelyezkedése a képernyőmérettől függően változik, így az oldal tartalma és elrendezése minden eszközön és képernyőméretnél jól látható és elérhető marad.

# Árlista

A price.php fájlban található továbbá egy szekció a fodrász szolgáltatásainak bemutatására. Ez a szekció tartalmazza a fodrász által kínált szolgáltatásokat, valamint az áraikat. Az árak dinamikusan jelennek meg az oldalon, és azokat az adatbázisból származó adatok alapján jelenítik meg. Ezek az adatok egy Modal típusú ablakba töltődnek be, amit a Bootstrap segítségével készítettünk.

Az adatok az alapján töltődnek be mit választunk az oldalon:



Ha rákattintunk a „Kérjük válasszon:” gombra akkor megnyílik egy Modal típusú ablak, ami az alapján tölti be az árlistát, hogy mit választunk.

Ezt úgy oldottuk meg hogy a Modal ablakok meg lettek **id**-zva, tehát ha például a férfi gombra kattintunk akkor rögtön elvisz a férfi árlista Modal ablakára: **data-bs-target="#manModal"**.

A Modal ablakok kinézete a Bootstrap keretrendszer alapján lett kialakítva, azonban ahhoz, hogy az oldalunk designjához illeszkedjenek, szükséges volt néhány stílusbeli változtatás. Ezeket a változtatásokat a style.css fájlban hajtottuk végre.

**Árlista betöltődése adatbázisból**:

Mielőtt nekilátnánk lekérdezéseket írni vagy adatokat hozzáadni PHP nyelven, előtte mindenképp csatlakoznunk kell az adatbázishoz. Az adatbázisnak, minden esetben lennie kell egy szervercímének, egy felhasználónevének és egy jelszónak.

**Szintaxis:**

mysqli\_connect (SZERVERCÍM, FELHASZNÁLÓNÉV, JELSZÓ, ADATBÁZISNÉV)

A kapcsolódás valójában ebből az egyetlen egy sorból áll, de a lenti példából látni, hogy érdemes még pár sorral kiegészíteni, hogy a kódunk átláthatóbb, könnyebben módosítható legyen.

Példa a database\_conf.php-ból:

<?php

$dbhost="localhost";

$dbuser="root";

$dbpass="";

$db="fodrászat";

$con = mysqli\_connect($dbhost,$dbuser,$dbpass,$db); // csatlakozás az adatbázishoz

?>

A fenti példában látható, hogy a legelején felvesszük az adatbázis kapcsolódáshoz szükséges adatokat (szervercím, felhasználónév, jelszó, adatbázis) különböző változókba, melyeket egy sorral lentebb behelyettesítünk a mysqli kapcsolatlétesítési függvénybe.

**Adatbázis lekérdezés:**

Miután láthattuk, hogy történik a kapcsolódás ideje megemlíteni a lekérdezést.

**$shortHair = "SELECT `name`,`Rövid` FROM `pricelist` WHERE `F/N`='N'";**

**$halfLenghtHair = "SELECT `name`,`Félhosszú` FROM `pricelist` WHERE `F/N`='N'";**

**$lenghtHair = "SELECT `name`,`Hosszú` FROM `pricelist` WHERE `F/N`='N'";**

**$manHair = "SELECT `name`,`price` FROM `pricelist` WHERE `F/N`='F'";**

Ezek a változók SQL lekérdezések, amelyek az adatbázisból adatokat kérnek le. Az első oszlop mindig a "**name**" oszlop, amely a szolgáltatás nevét tartalmazza, a második oszlop pedig a megfelelő árat. A "**pricelis**t" az adatbázis tábla neve, ahol az árak és a szolgáltatások nevei tárolódnak. A "**WHERE**" feltétel azt jelenti, hogy csak azokat a sorokat kéri le, ahol az "**F/N**" oszlop értéke "**N**" vagy "**F**" (azaz férfi vagy női), tehát csak azokat az árakat és szolgáltatásokat jeleníti meg, amelyek relevánsak a felhasználó számára.

A konkrét SQL lekérdezések után szükség van az eredmények lekérdezésére és feldolgozására, amelyek általában egy "**mysqli\_query()"** függvénnyel történnek, majd egy "**while**" ciklussal feldolgozzuk az összes eredményt és kiírjuk egy újabb Modal típusú ablakban **checkbox** formában.

Ha a felhasználó többféle szolgáltatást választ akkor a checkbox-al kijelöli a kívánt szolgáltatást majd mikor a „Kész” gombra kattint (amely egy „**submit**” típusú gomb) akkor kiírja a választott szolgáltatások árát összeadva:

            <?php

            if(isset($\_GET['submit'])){

                $inputs = array('short', 'halfLenght', 'lenght', 'man');

                $hasSelection = false;

                $shipping\_subtotal = 0;

                foreach($inputs as $input) {

                    if(!empty($\_GET[$input])) {

                        $hasSelection = true;

                        foreach($\_GET[$input] as $result) {

                            $shipping\_subtotal += $result;

                        }

                    }

                }

Ez a kódrészlet először ellenőrzi, hogy a "submit" nevű gombra kattintott-e a felhasználó. Ha igen, akkor egy tömböt hoz létre négy input mezővel, amelyeknek a nevei a "short", "halfLenght", "lenght" és "man". Ezután egy ciklussal végigmegy az összes input mezőn, és ha egy adott mező nem üres, akkor a "hasSelection" változó értékét igazra állítja, majd egy másik ciklussal végigmegy az adott mezőhöz tartozó összes értéken, és hozzáadja azokat a "shipping\_subtotal" változó értékéhez.

if($hasSelection) {

                    echo('<div class="text-center"><h1 class="modalText">A választottak alapján az eredmény: <p class="modalPriceText"><u>'.number\_format($shipping\_subtotal).'Ft</u></p></h1></div>');

                } else {

                    echo "<b class='warningText'>Kérjük, válasszon legalább egy lehetőséget.</b>";

                }

            }

            ?>

Végül az utolsó "if" feltétel ellenőrzi, hogy van-e legalább egy kiválasztott mező (ha van, akkor a "hasSelection" változó igazra lett állítva), és ha van, akkor kiírja a kiválasztott szolgáltatások árát összeadva. Ha nincs kiválasztva semmilyen szolgáltatás, akkor kiírja a felhasználónak, hogy legalább egyet ki kell választania.

# Galéria

A főoldal navigációs sávjában a Galériára kattintva eljutunk a jobs.html-hez ami a fodrász eddig munkáit mutatja be. A **pictures1** osztály egy konténer div, amely egy képet és egy címet tartalmaz. Az **img** tag a képet tartalmazza, az **alt** attribútum pedig a kép alternatív szöveges leírását adja meg, amely akkor jelenik meg, ha a kép nem jelenik meg a felhasználó számára. Az **image-title** osztály az adott képhez tartozó „leírást” tartalmazza, és középre igazított div-ben helyezkedik el.

Az egész konténer egy „**welcome”** osztályú div-ben helyezkedik el. A konténerek mindegyike kerekített sarkú, és az egérmutató fölé helyezésekor egy hover hatást eredményez, amely felnagyítja a képet.

A képek a következő képpen lettek formázva a CSS-ben:

.pictures1 img{

  height: 100%;

  width: 70%;

  border: 0.125rem solid #f4b027c0;

  border-radius: 5%;

  transition: .6s ease-in-out;

  transform-origin: left;

  clip-path: circle(40%);

}

A **transition** tulajdonság beállításával az átmenet simábbá válik az egérmozdulatokra, így a kép méretének növelése vagy csökkentése animált hatással jelenik meg.

A **transform-origin** tulajdonság beállítása az átméretezett képnek a bal oldali szélénél tartja meg az eredeti helyzetét, így a kép nagyobbá válása főként jobbra történik majd.

A képeket körbefogó keretet kör alakúvá alakítja a **clip-path** tulajdonsággal, amely a **circle** függvényt használja a 40% sugárral.

A képek felnagyítása meg a következő képpen néz ki:

.pictures1 img:hover{

  z-index: 1;

  transform: scale(2);

}

Mikor az egérkurzort a kép fölé visszük akkor ez a szabály lép életbe. A **z-index: 1**-esre állítása azért volt fontos, mert így biztosítjuk, hogy a kép a többi elem fölé kerüljön.

Ezenkívül a **transform: scale(2)** CSS szabály megnöveli a kép méretét, amely az eredeti méret kétszeresére lesz skálázva, így nagyobb lesz, és jobban látható részleteket mutat.

# Adatbázis-módosítás

A database.php oldal létrehozásának célja az volt, hogy az ügyfél könnyedén tudja módosítani az árait és szolgáltatásait, amelyek egy táblázatban vannak betöltve. Az oldalt nem lehet elérni a főoldalról csak az általunk megadott link alapján lehet, és az adatok módosítása csak a bejelentkezett felhasználó számára elérhető. Amikor az ügyfél végzett a módosításokkal, a mentés gombra kattintva egy **alert** ablak jelenik meg a képernyőn, amely jelzi, hogy az adatok sikeresen frissültek.

Az oldal lehetőséget nyújt az ügyfélnek az árak és szolgáltatások könnyű és gyors módosítására, ezáltal csökkentve az adminisztrációs feladatokat. Az árak és szolgáltatások táblázatban való megjelenítése szintén lehetővé teszi az ügyfél számára, hogy egyszerűen áttekintse a kínálatot és frissítse azt.

Az oldal bejelentkezési rendszere biztonságosabbá teszi az adatokat, mivel csak az engedélyezett felhasználók érhetik el az oldalt. Az ügyfélnek csak egy felhasználónevet és jelszót kell megadnia a bejelentkezéshez, így egyszerűbb a belépés és a módosítások elvégzése.

A mentés gombra kattintva megjelenő üzenet, amely jelzi a módosítások sikeres végrehajtását, segít az ügyfélnek biztosítani, hogy az adatok frissültek, és hogy az új árak és szolgáltatások megjelennek az ügyfelek számára. Ez lehetővé teszi az ügyfél számára, hogy egyszerűen és gyorsan frissítse az oldalt, és biztosítja a pontos információkat az felhasználók számára.

A megjelenítést próbáltuk úgy megoldani, mint a kapott táblázatban, amit adott az ügyfél. Amit egy másik CSS fájlban formáztunk meg. (datastyle.css) Valamint létrehoztunk egy Modal típusú ablakot a bejelentkezési űrlapnak. (loginstyle.css) Itt nem használtunk a Bootstrap keretrendszert.

Az adatok szintén úgy töltődnek be, mint az Árlista fejezetben csak itt annyi a különbség, hogy nem **while** ciklust alkalmaztunk, hanem **for** ciklust.

Női táblázathoz való lekérdezés:

**$sql = "SELECT `name`,`Rövid`,`Félhosszú`,`Hosszú` FROM `pricelist` WHERE `ID`='$counter'";**

Férfi táblázathoz való lekérdezés:

**$sql = "SELECT `name`,`price` FROM `pricelist` WHERE `ID`='$counter'";**

Az adatokat itt is az **input** mezőkbe töltődik be, azonban ezek text típusúak, hogy a felhasználó könnyedén tudja szerkeszteni a tartalmat. A **value** érték pedig azért van megadva, hogy az alapértelmezett érték legyen megjelenítve a mezőben, ami a jelen esetben az adatbázisban tárolt árak értéke.

Az adatok módosítása a következő módon lett megoldva:

A kód egy **for** ciklussal bejárja a 1-től 16-ig az ID sorokat, majd minden sorban leellenőrzi, hogy a Submit gomb meg lett-e nyomva. Ha igen, akkor kiolvassa az input mezők értékét és frissíti az adatbázist az új értékekkel. A módosítás SQL lekérdezése az adott ID-jú sorra vonatkozik és a SET kulcsszó segítségével frissíti az adatbázisban tárolt adatokat:

**$change = mysqli\_query($con, "UPDATE `pricelist` SET `name` = '$name', `Rövid` = '$short', `Félhosszú` = '$halfLenght', `Hosszú` = '$lenght' WHERE `pricelist`.`ID`='$counter'");**

A 17-től 20-ig terjedő sorok esetében is lehetséges az árakat és a nevet tartalmazó mezőket módosítani, így a hozzájuk tartozó módosító kódban mindkét mező értékének leolvasása és a hozzá tartozó ID-val való frissítése szerepel.

**$change = mysqli\_query($con, "UPDATE `pricelist` SET `name` = '$name',`price` = '$man' WHERE `pricelist`.`ID`='$counter'");**

A módosítások elmentése után az adatok frissülnek az adatbázisban, és az árlista oldal frissítésével az új adatok láthatóvá válnak az oldalon.

# Login

**HTML szerkezet:**

Az database.php oldalon a bejelentkezési lehetőség egy **modal** ablakban jelenik meg, amit a **#logModal** azonosítójú div-en belül hoztunk létre. A bejelentkezéshez szükséges adatokat, azaz a felhasználónevet és a jelszót **input** mezők segítségével lehet megadni. A **bejelentkezés** gomb az **input** mezők alatt található, a **type="submit"** tulajdonsággal, amely az űrlap elküldésére szolgál.

<div class="modal" id="logModal">

        <form id="login-form" class="modal-form animate">

            <div class="avatar">

                <img src="./img/login\_avatar.png" alt="avatar">

            </div>

            <div class="input-container">

                <i class="fa fa-user fa-10x"></i>

                <input type="text" placeholder="Felhasználónév" name="username" id="username">

                <i class="fa fa-key"></i>

                <input type="password" placeholder="Jelszó" name="password" id="password">

                <button type="submit" class="btn">Bejelentkezés</button>

            </div>

        </form>

</div>

**CSS style:**

A **modal** stílusáért a loginstyle.css felel, amely meghatározza annak kinézetét, méretét és pozícióját. A modal ablak kódolásál kiemelt figyelmet szenteltünk a reszponzívitásnak.

**A modal stílusa:**

.modal{

    position: fixed;

    z-index: 1;

    left: 0;

    top: 0;

    width: 100%;

    height: 100%;

    overflow: auto;

    background-color: rgb(0,0,0);

    background-color: rgba(0,0,0,0.4);

}

Ez a kód a modal elhelyezkedését és áttetszőségét határozza meg. A modal **position** tulajdonsága **fixed**, így mindig a képernyő tetején fog megjelenni. A z-index tulajdonsága meghatározza a modal rétegszámát, ami azt jelenti, hogy az oldal többi tartalmánál előrébb jelenik meg. Az **overflow** tulajdonság lehetővé teszi, hogy a modal megjelenésekor az oldal mögötti tartalom gördülékenyen gördüljön le, amíg a modal megjelenik. A **background-color** tulajdonság pedig beállítja a modal mögötti áttetsző szürke szín értékét.

**Modal tartalma:**

.modal-form{

    background-color: #0e0d0c;

    margin: 0% auto 15% auto;

    border-radius: 20%;

    width: 60%;

    padding-top: 5%;

}

Ez a kód a modal tartalmát határozza meg. A **width** tulajdonság meghatározza a modal szélességét, míg a **margin** tulajdonság lehetővé teszi, hogy a modal középen jelenjen meg a képernyőn. A **background-color** tulajdonság beállítja a modal háttérszínét, a **border-radius** tulajdonság pedig kerekített sarkokat ad a modalnak.

**Reszponzivitás:**

A reszponzív megjelenést főként relatív mértékegységek és media query-k segítségével csináltuk.

@media only screen and (min-device-width: 300px) and (max-device-width: 480px){

    .modal-form{

        background-color: #0e0d0c;

        margin: 0% auto 15% auto;

        border-radius: 5%;

        width: 100%;

        padding-top: 5%;

    }

    .input-container{

        padding-right: 15%;

        padding-left: 15%;

        padding-bottom: 5%;

    }

    .btn{

        width: 100%;

        font-size: 1rem;

    }

}

Ez a **media query** felelős azért, hogy a CSS kód a készülék méretének függvényében változzon. Ha a képernyő szélessége 300 pixel és 480 pixel között van, akkor az általa meghatározott CSS szabályokat alkalmazza, amelyek megváltoztatják a modal háttérszínét, a margóját, a border-radius tulajdonságát, a szélességét és a padding-top, padding-right, padding-left és padding-bottom értékeit az input mezőkön belül. Emellett a gomb szélessége és betűmérete is változik, hogy a tartalom megfelelően jelenjen meg a kisebb képernyőméretek esetén.

**JavaScript:**

A bejelentkezési felület működéséért a login.js felel.

function checkLogin() {

    let username = localStorage.getItem('username');

    let password = localStorage.getItem('password');

    let loginTime = localStorage.getItem('loginTime');

    if (username && password && loginTime && (new Date().getTime() - loginTime < 60000)) {

        document.getElementById('logModal').style.display='none';

        return true;

    } else {

        return false;

    }

}

A **checkLogin()** függvény ellenőrzi, hogy a felhasználó korábban bejelentkezett-e már, és ha igen, akkor az oldal letiltja a bejelentkező modal megjelenítését 1 percre, miután újra felugrik és kéri a bejelentkezést. Erre biztonsági okok miatt van szükség.

function saveLogin() {

    let username = document.getElementById('username').value;

    let password = document.getElementById('password').value;

    localStorage.setItem('username', username);

    localStorage.setItem('password', password);

    localStorage.setItem('loginTime', new Date().getTime());

}

A **saveLogin()** függvény elmenti a felhasználó adatait és az időbélyeget a helyi tárolóban, ami lehetővé teszi a felhasználó számára, hogy az oldal használata során ne kelljen minden alkalommal újra megadnia az adatait például az oldal frissítése.

function validateLogin() {

    let username = document.getElementById('username').value;

    let password = document.getElementById('password').value;

    if (username == 'BadicsViki' || password == '1234') {

        alert('Sikeres bejelentkezés!');

        document.getElementById('logModal').style.display='none';

        return true;

    } else {

        alert('Hibás felhasználónév vagy jelszó!');

        return false;

    }

}

A **validateLogin()** ellenőrzi a bejelentkezési adatokat, és ha helyesek, akkor megjelenít egy üzenetet a sikeres bejelentkezésről, majd elrejti a bejelentkezési felületet. Ha az adatok helytelenek, akkor egy hibaüzenetet jelenít meg.

document.getElementById('login-form').addEventListener('submit', function(event) {

    event.preventDefault();

    if (validateLogin()) {

        saveLogin();

    }

});

Az **addEventListener()** az oldalon található bejelentkezési űrlap elküldésekor hívódik meg, és meghívja a **validateLogin()** függvényt. Ha a bejelentkezés sikeres volt, akkor elmenti a bejelentkezési adatokat a helyi tárolóban.

window.addEventListener('load', function() {

    if (checkLogin()) {

        document.getElementById('username').value = localStorage.getItem('username');

        document.getElementById('password').value = localStorage.getItem('password');

    }

});

A **window.addEventListener()** az oldal betöltésekor ellenőrzi, hogy a felhasználó be van-e jelentkezve, és ha igen, akkor betölti a felhasználónév és jelszó adatokat a megfelelő mezőkbe a helyi tárolóból.

setInterval(function() {

    document.getElementById('username').value = '';

    document.getElementById('password').value = '';

    document.getElementById('logModal').style.display='block';

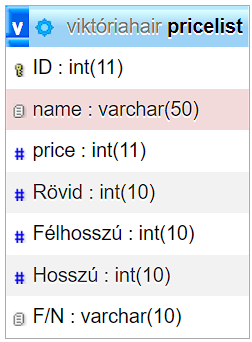
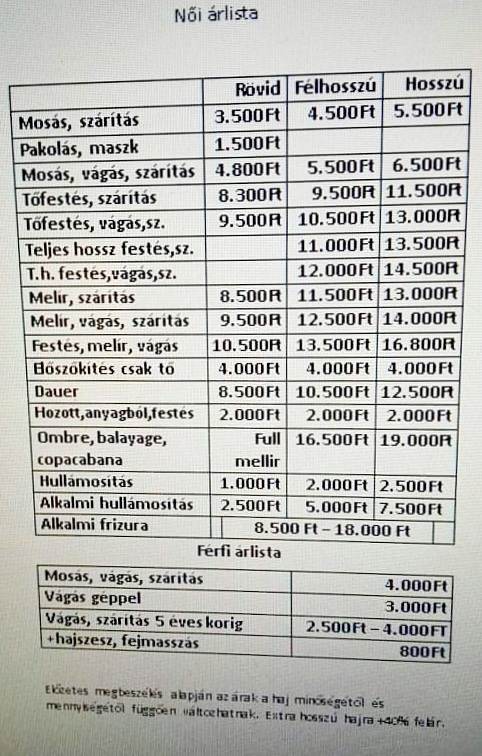
}, 61000);

A **setInterval()** időzítő 60 másodpercenként feldobja üresen a bejelentkezési felületet, ha a felhasználó nem jelentkezett be a megadott időn belül. Ez megakadályozza a felhasználói munkamenet lejártát és automatikusan kijelentkezteti a felhasználót.

# Adatbázis

A PhpMyAdmin egy webes felületű kliens a XAMPP-on belüli MySQL adatbázisok kezeléséhez. Adatbázisunk létrehozása is ebben a kliensben történt. Egy egytáblás modell-t alkalmaztunk ugyanis az ügyfél által adott árlista alapján dolgoztunk és mivel nem tartalmazott sok adatot ezért feleslegesnek tartottuk több táblásra csinálni. Illesztése utf8\_hungarian\_ci.

Az árlista, ami alapján dolgoztunk: Ez pedig az adatbázis táblán felépítése:



Adatbázis neve: viktóriahair

Tábla neve: pricelist

Mezők:

* Azonosító (ID): egész szám, elsődleges kulcs, automatikusan generált
* Név (name): karakterlánc, maximum 50 karakter
* price: egész szám, számjegyek maximális száma 11
* Rövid: egész szám, számjegyek maximális száma 10
* Félhosszú: egész szám, számjegyek maximális száma 10
* Hosszú: egész szám, számjegyek maximális száma 10
* F/N: karakterlánc, maximum 10 karakter

Adatok kapcsolódása: Az azonosító mezőt használjuk a rekordok egyedi azonosítására.

Adatbázis séma: A pricelist tábla az adatok tárolására szolgál. Az adatokat az azonosító mező alapján rendezzük. A tábla tartalmaz egy elsődleges kulcsot (azonosító mező), amelyet a rendszer automatikusan generál.

# Tesztelés

A tesztelés során különböző böngészőket használtunk, mint például a Google Chrome, Mozilla Firefox, Microsoft Edge. Az oldalt különböző készülékeken is teszteltük, beleértve a mobiltelefonokat, táblagépeket és laptopokat. A tesztelés során ellenőriztük az oldal különböző funkcióit és alkalmazásait, beleértve a navigációt, a menüket, a gombokat, az űrlapokat. Az adatbázis funkcióit XAMPP-on keresztül teszteltük. Az oldal üzemeltetéséhez egy ingyenes tárhelyet használunk, melyet a [www.nethely.hu](http://www.nethely.hu) biztosít számunkra, az oldal címe www.vikihair.nhely.hu. A tesztelés során meggyőződtünk arról, hogy az oldal megfelelően működik és megfelelően jelenik meg különböző környezetekben és készülékeken.