PC Parts Webshop – Webprogramozás 1. beadandó

Készítette: Flámis Géza Zsolt

Neptun-kód: NGEK9N

GitHub repo: https://github.com/Geza1985/webshopngek9n

Webes tárhely: http://pcshopngek9n.great-site.net

Dátum: 2025. május 6.

Fejlesztési környezet

A projekt fejlesztése az alábbi eszközökkel történt:
- XAMPP (PHP 8.x, MySQL)
- Visual Studio Code
- GitHub + GitHub Desktop
- Chrome böngésző
- InfinityFree tárhely (ingyenes FTP + MySQL)
- Google Maps és YouTube iframe beágyazás

A projekt célja

Az alkalmazás célja egy egyszerű számítógép alkatrész webshop létrehozása, amely lehetővé teszi a látogatók számára a képek böngészését, üzenetküldést, valamint a felhasználók számára regisztrációt, belépést, képfeltöltést.

A funkciók részletes bemutatása

Front controller és oldalletöltés

Az index.php fájl dinamikusan tölti be a tartalmakat a ?page= paraméter alapján. Az oldalak külön fájlokban vannak a pages/ könyvtárban.

Dinamikus menü

A menüpontokat a core/config.php fájlban egy asszociatív tömb tárolja, a fejléc sablon ezt használja dinamikusan.

Felhasználói autentikáció

A register.php, login.php, logout.php oldalak kezelik a regisztrációt, bejelentkezést és kilépést. A jelszavak hash-eléssel kerülnek mentésre.

Galéria és képfeltöltés

A gallery.php oldal kezeli a képek megjelenítését. A feltöltés csak bejelentkezett felhasználóknak engedélyezett, a képek fájlként és adatbázisban is tárolódnak.

Videó és Google térkép

YouTube videó és Google Maps iframe az info.php oldalon kerül megjelenítésre.

Kapcsolat űrlap

A contact.php oldalon keresztül üzenet küldhető, melyet JavaScript és PHP is ellenőriz. Az üzenetek a messages táblába kerülnek.

Üzenetek megjelenítése

A messages.php oldal csak belépett felhasználó számára érhető el, az összes beküldött üzenetet időrendben jeleníti meg.

Adatbázis

Az init.sql fájl létrehozza a szükséges users, gallery és messages táblákat. Az adatbáziskapcsolat PDO-val történik a core/functions.php fájlban.

GitHub és tárhely hozzáférés

GitHub repo: https://github.com/Geza1985/webshopngek9n

Weboldal: http://pcshopngek9n.great-site.net

FTP elérhetőség:
- FTP host: ftpupload.net
- Felhasználónév: epiz_38900826
- Jelszó: uqCnPTEzuDqei
- Alap mappa: /htdocs/

Adatbázis elérés (phpMyAdmin):
- phpMyAdmin URL: https://sql213.epizy.com/phpmyadmin
- Adatbázisnév: epiz_38900826_pcparts
- Felhasználó: epiz_38900826
- Jelszó: uqCnPTEzuDqei

core/config.php

core > config.php

```php
<?php
$menu = [
    'home' => 'Főoldal',
    'products' => 'Termékek',
    'gallery' => 'Galéria',
    'contact' => 'Kapcsolat',
    'messages' => 'Üzenetek'
];

$pages = [
    'home' => 'pages/home.php',
    'products' => 'pages/products.php',
    'gallery' => 'pages/gallery.php',
    'contact' => 'pages/contact.php',
    'messages' => 'pages/messages.php',
    'login' => 'pages/login.php',
    'register' => 'pages/register.php',
    'logout' => 'pages/logout.php'
];

```

// A menü tömb tartalmazza a weboldal fő navigációs pontjait.

// A kulcs a ?page= paraméterhez, az érték pedig a megjelenítendő szöveg.

// Az oldalakhoz tartozó fájlnevek listája.

// A kulcs egyezik a ?page= paraméter értékével, az érték a betöltendő PHP fájl útvonala.

// Ezzel valósul meg az index.php dinamikus tartalombetöltése.

A core/config.php fájlban két asszociatív tömb segítségével van definiálva a menürendszer és az oldalbetöltési logika. A $menu tömb határozza meg, hogy milyen menüpontok jelennek meg a fejlécben, míg a $pages tömb alapján az index.php dinamikusan tölti be a megfelelő oldalt a ?page=... paraméter alapján.

core/functions.php

core > functions.php

```php
<?php
function getDB() {
    return new PDO(
        'mysql:host=sql102.infinityfree.com;dbname=if0_38900826_pcparts;charset=utf8',
        'if0_38900826',
        'uqCnPTEzuDqei',
        [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]
    );
}
```

// Ez a függvény hozza létre és adja vissza a PDO adatbázis-kapcsolatot.

// A kapcsolat a InfinityFree tárhelyen található MySQL adatbázishoz csatlakozik.

// A kapcsolat típusa (MySQL), a szerver host címe, adatbázis neve és karakterkódolása

// Felhasználónév az adatbázishoz

// Jelszó az adatbázishoz

// PDO hibakezelési mód: kivétel dobása hiba esetén

A getDB() függvény felelős a PDO-alapú MySQL adatbázis kapcsolat létrehozásáért. A beállítások tartalmazzák a host címet, az adatbázis nevét, a felhasználónevet és a jelszót. A PDO::ERRMODE_EXCEPTION opció biztosítja, hogy hiba esetén kivételt dobjon, így a hibák könnyen kezelhetők vagy naplózhatók. Az alkalmazás minden SQL-művelete ezt a központosított kapcsolati függvényt használja.

pages/contact.php

pages > contact.php

```php
<?php
require_once("core/functions.php");

$error = "";
$success = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = trim($_POST["name"]);
    $email = trim($_POST["email"]);
    $message = trim($_POST["message"]);

    // Szerveroldali validáció
    if (empty($name) || empty($email) || empty($message)) {
        $error = "Minden mező kitöltése kötelező!";
    } elseif (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        $error = "Érvénytelen e-mail cím!";
    } else {
        $db = getDB();
        $sender = isset($_SESSION['user']) ? $_SESSION['user']['username'] : 'Vendég';
        $stmt = $db->prepare("INSERT INTO messages (name, email, message, sender) VALUES (?, ?, ?, ?)");
        $stmt->execute([$name, $email, $message, $sender]);
        $success = "Üzenet sikeresen elküldve!";
    }
}
?>

<h2>Kapcsolat</h2>
```

```php

<?php if ($error): ?>
    <p style="color:red;"><?= $error ?></p>
<?php elseif ($success): ?>
    <p style="color:green;"><?= $success ?></p>
<?php endif; ?>

<form method="POST" onsubmit="return validateContactForm();">
    <input type="text" name="name" placeholder="Név"><br>
    <input type="text" name="email" placeholder="E-mail"><br>
    <textarea name="message" placeholder="Üzenet szövege..."></textarea><br>
    <button type="submit">Küldés</button>
</form>

<script src="js/validation.js"></script>

```

// Adatbázis kapcsolatot biztosító fájl betöltése

// Űrlapot POST metódussal

// Szerveroldali ellenőrzés

// Ha nincs hiba, mentés adatbázisba

// Ha van bejelentkezett felhasználó, a nevét menti el, különben "Vendég"

// SQL parancs előkészítése és futtatása

//Hiba- vagy sikerüzenet megjelenítése

//Űrlap, kliensoldali validációval

//JavaScript validáció betöltése

A contact.php oldal egy kapcsolatfelvételi űrlapot tartalmaz, amely szerveroldalon is ellenőrzi a beküldött adatokat. Az űrlap beküldése esetén a mezők értékét szerveroldalon ellenőrzi, és ha helyesek, akkor elmenti őket az adatbázis messages táblájába. A bejelentkezett felhasználók esetén a sender mezőbe a felhasználónév kerül, míg a vendégek esetén "Vendég". Az űrlap tartalmaz egy JavaScript fájlra való hivatkozást is (js/validation.js), ami kliensoldali validációt végez.

pages/gallery.php

pages > gallery.php

```php
<?php
require_once("core/functions.php");

$db = getDB();

// Feltöltés – csak belépett felhasználónak
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_SESSION['user'])) {
    if (isset($_FILES['image']) && $_FILES['image']['error'] === 0) {
        $targetDir = "uploads/gallery/";
        $filename = basename($_FILES['image']['name']);
        $targetFile = $targetDir . $filename;

        if (move_uploaded_file($_FILES['image']['tmp_name'], $targetFile)) {
            try {
                $stmt = $db->prepare("INSERT INTO gallery (filename) VALUES (?)");
                $stmt->execute([$filename]);
                echo "<p style='color:green;'>✅ Kép feltöltve és elmentve!</p>";
            } catch (PDOException $e) {
                echo "<p style='color:red;'>❌ HIBA: " . $e->getMessage() . "</p>";
            }
        } else {
            echo "<p style='color:red;'>❌ Hiba történt a fájl mentésekor.</p>";
        }
    }
}
```

```php
// Képek lekérése
try {
    $images = $db->query("SELECT * FROM gallery ORDER BY uploaded_at DESC")->fetchAll();
} catch (PDOException $e) {
    echo "<p style='color:red;'>❌ Galéria betöltési hiba: " . $e->getMessage() . "</p>";
    $images = [];
}
?>

<h2>Képgaléria</h2>

<?php if (isset($_SESSION['user'])): ?>
<form method="POST" enctype="multipart/form-data">
    <input type="file" name="image" accept="image/*" required>
    <button type="submit">Feltöltés</button>
</form>
<?php else: ?>
<p>🔒 Csak bejelentkezve lehet képet feltölteni.</p>
<?php endif; ?>

<hr>

<div style="display: flex; flex-wrap: wrap; gap: 10px; margin-top: 20px;">
    <?php foreach ($images as $img): ?>
        <div style="width: 200px; border: 1px solid #ccc; padding: 5px;">
            <img src="uploads/gallery/<?= htmlspecialchars($img['filename']) ?>" alt="Kép" style="width: 100%;">
            <small>🗓️ Feltöltve: <?= $img['uploaded_at'] ?></small>
        </div>
    <?php endforeach; ?>
</div>
```

//Ellenőrzi, hogy POST metódussal és bejelentkezett felhasználóként töltik-e fel a képet

//Ha a fájl rendben van, átmásolja az uploads/gallery/ mappába

//Mentést végez az adatbázis gallery táblájába (csak a fájlnév kerül bele)

//Lekérdezi az összes feltöltött képet az uploaded_at mező szerinti csökkenő sorrendben

//Hiba esetén egy üzenetet ír ki, és nem jelenít meg képeket

//A feltöltési űrlap csak belépett felhasználóknak látszik

//A képek rácsszerű elrendezésben jelennek meg

//Minden képhez megjelenik a feltöltés dátuma

A gallery.php oldal biztosítja a képfeltöltés és megtekintés funkcióját. Csak bejelentkezett felhasználók tölthetnek fel képet, amit a szerver ellenőriz, majd ment a fájlrendszerbe és az adatbázisba is. A feltöltött képek egy galériában jelennek meg feltöltési dátum szerint visszafelé sorrendben. A fájlneveket és HTML attribútumokat htmlspecialchars() segítségével biztonságosan jelenítjük meg.

pages/home.php

pages > home.php

```php
<h2>Üdvözlünk a PC Parts Shop oldalán!</h2>
<p>Nálunk megtalálod a legjobb alkatrészeket: processzorokat, videokártyákat, memóriákat és sok mást!</p>

<h3>Bemutatkozó videó</h3>
<video width="320" height="240" controls>
  <source src="uploads/intro.mp4" type="video/mp4">
  A böngésződ nem támogatja a videólejátszást.
</video>

<h3>Ismertető videó (YouTube-ról)</h3>
<iframe width="560" height="315" src="https://www.youtube.com"
        title="YouTube videó" frameborder="0" allowfullscreen></iframe>

<h3>Üzletünk helye a térképen</h3>
<iframe
    src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d10787.818824267582!2d19.0688376!3d47.52718
    width="600" height="450" style="border:0;" allowfullscreen loading="lazy"></iframe>

```

//A főoldal bevezető címsora és leírása.

//Saját, feltöltött videó (uploads/intro.mp4) lejátszása <video> taggel.

//A controls attribútum megjeleníti a lejátszás/szünet/görgetés lehetőségeket.

//Egy YouTube-videó beágyazása.

//A src attribútumba valós YouTube-videó linket kell illeszteni (jelenleg csak alap link van).

A home.php a főoldali megjelenítésért felel. Dinamikus multimédiás elemeket tartalmaz: egy saját feltöltött bemutatkozó videót, egy YouTube-linkkel beágyazott ismertetőt, valamint egy Google Maps iframe-et, amely az üzlet helyét mutatja meg. Ezzel a látogatók vizuálisan is képet kapnak a bolt működéséről és elérhetőségéről.

pages/login.php

pages > login.php

```php
<?php
require_once("core/functions.php");
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['username'];
    $password = $_POST['password'];

    $db = getDB();
    $stmt = $db->prepare("SELECT * FROM users WHERE username = ?");
    $stmt->execute([$username]);
    $user = $stmt->fetch();

    if ($user && password_verify($password, $user['password'])) {
        $_SESSION['user'] = [
            'name' => $user['lastname'] . ' ' . $user['firstname'],
            'username' => $user['username']
        ];
        header("Location: index.php");
        exit;
    } else {
        echo "<p>Hibás felhasználónév vagy jelszó!</p>";
    }
}
?>

<h2>Belépés</h2>
<form method="POST">
    <input type="text" name="username" placeholder="Felhasználónév" required><br>
    <input type="password" name="password" placeholder="Jelszó" required><br>
    <button type="submit">Belépés</button>
</form>
<p><a href='index.php?page=register'>Regisztráció</a></p>

```

//Ellenőrzi, hogy POST kérés érkezett-e (űrlap beküldése)

//A felhasználó által megadott adatokat változókba menti

//Létrehozza a PDO kapcsolatot

//Lekérdezi az adott felhasználó rekordját az adatbázisból

//Ha a felhasználó létezik és a jelszó stimmel (password_verify() használatával), akkor beállítja a session-t

//Ezáltal a felhasználó bejelentkezettnek számít

//Hibás adat esetén hibaüzenet jelenik meg

//Az űrlap bekéri a belépéshez szükséges adatokat

//Egy linket is tartalmaz a regisztrációhoz

A login.php oldal lehetővé teszi a felhasználók számára, hogy bejelentkezzenek a rendszerbe. A rendszer ellenőrzi, hogy a megadott felhasználónév létezik-e az users táblában, majd a jelszót password_verify() függvénnyel hasonlítja össze a titkosított adatbázisbeli jelszóval. Sikeres hitelesítés esetén a felhasználót bejelentkezteti és átirányítja a főoldalra, míg hiba esetén hibaüzenetet jelenít meg.

pages/logout.php

pages > logout.php

```php
<?php
session_destroy();
header("Location: index.php");
exit;

```

// Eltávolítja az aktuális munkamenetet és az összes tárolt session adatot

// Átirányítja a felhasználót a főoldalra

// Leállítja a script végrehajtását az átirányítás után

A logout.php fájl biztosítja a felhasználó kijelentkeztetését. A session_destroy() segítségével törli az összes session adatot, majd azonnal átirányítja a látogatót a főoldalra (index.php). Ezáltal a felhasználó kilép a bejelentkezett állapotból, és a rendszer újra vendégként kezeli.

pages/messages.php

pages > messages.php

```php
<?php
require_once("core/functions.php");

if (!isset($_SESSION['user'])) {
    echo "<p style='color:red;'>Ez az oldal csak bejelentkezett felhasználóknak érhető el.</p>";
    return;
}

$db = getDB();
$messages = $db->query("SELECT * FROM messages ORDER BY sent_at DESC")->fetchAll();
?>

<h2>Beküldött üzenetek</h2>

<?php if (empty($messages)): ?>
    <p>Nincs még üzenet.</p>
<?php else: ?>
    <table border="1" cellpadding="5" cellspacing="0">
        <tr>
            <th>Név</th>
            <th>E-mail</th>
            <th>Üzenet</th>
            <th>Időpont</th>
            <th>Küldő</th>
        </tr>
        <?php foreach ($messages as $msg): ?>
            <tr>
                <td><?= htmlspecialchars($msg['name']) ?></td>
                <td><?= htmlspecialchars($msg['email']) ?></td>
                <td><?= nl2br(htmlspecialchars($msg['message'])) ?></td>
                <td><?= $msg['sent_at'] ?></td>
                <td><?= $msg['sender'] ?></td>
            </tr>
        <?php endforeach; ?>
    </table>
<?php endif; ?>

```

//Betölti az adatbázis-kapcsolatot biztosító getDB() függvényt.

//Jogosultságellenőrzés: ha a felhasználó nincs bejelentkezve, hibaüzenet jelenik meg, és a script leáll.

//Kapcsolódik az adatbázishoz, és lekéri az összes üzenetet időrend szerint visszafelé (DESC).

//Ha nincs adat: "Nincs még üzenet" jelenik meg.

//Ellenkező esetben táblázatot hoz létre az alábbi mezőkkel: név, e-mail, üzenet, időpont, küldő

//A lekért adatok soronként kerülnek megjelenítésre

//A htmlspecialchars() és nl2br() használatával megelőzi az XSS támadásokat és megőrzi a sortöréseket az üzenet szövegében

A messages.php oldal kizárólag bejelentkezett felhasználók számára érhető el. Az oldal az adatbázisból kiolvassa a kapcsolatfelvételi űrlapon keresztül beküldött üzeneteket, és HTML táblázatban jeleníti meg azokat. A biztonságos megjelenítés érdekében minden adat htmlspecialchars() függvénnyel kerül kiírásra, ezzel védve a felületet a szkriptalapú támadások ellen.

pages/register.php

pages > register.php

```php
<?php
require_once("core/functions.php");
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $lastname = $_POST['lastname'];
    $firstname = $_POST['firstname'];
    $username = $_POST['username'];
    $password = password_hash($_POST['password'], PASSWORD_DEFAULT);

    $db = getDB();
    $stmt = $db->prepare("INSERT INTO users (lastname, firstname, username, password) VALUES (?, ?, ?, ?)");
    try {
        $stmt->execute([$lastname, $firstname, $username, $password]);
        echo "<p>Sikeres regisztráció! <a href='index.php?page=login'>Lépj be itt</a>.</p>";
    } catch (PDOException $e) {
        echo "<p>Hiba: felhasználónév már létezik.</p>";
    }
}
?>

<h2>Regisztráció</h2>
<form method="POST">
    <input type="text" name="lastname" placeholder="Vezetéknév" required><br>
    <input type="text" name="firstname" placeholder="Keresztnév" required><br>
    <input type="text" name="username" placeholder="Felhasználónév" required><br>
    <input type="password" name="password" placeholder="Jelszó" required><br>
    <button type="submit">Regisztráció</button>
</form>

```

//A password_hash() a jelszót titkosítja biztonságos algoritmussal

//A POST adatokból kerülnek ki az értékek

//Az SQL utasítás beszúrja az új felhasználót a users táblába

//Ha sikeres, üzenet jelenik meg, és linket ad a bejelentkezéshez

//Ha az adott felhasználónév már létezik, kivételt dob, és hibaüzenetet ír ki

//Az űrlap POST módszerrel továbbítja az adatokat

//Minden mező required, tehát üresen nem küldhető

A register.php oldal lehetővé teszi új felhasználók számára, hogy regisztráljanak a rendszerbe. Az űrlap beküldése után a rendszer ellenőrzi és biztonságosan titkosítja a jelszót. Ha a felhasználónév már létezik, hibaüzenetet jelenít meg. A sikeres regisztráció után egy link mutat a bejelentkezéshez. Ez biztosítja az új felhasználók biztonságos adatbevitelét és tárolását.

templates/footer.php

templates > footer.php

```
1    </main>
2    <footer>
3        <p>&copy; <?= date('Y') ?> PC Parts Shop</p>
4    </footer>
5    </body>
6    </html>
7
```

//<footer>: HTML5 elem, amely az oldal alján megjelenő információkat tartalmazza (pl. szerzői jog, elérhetőség stb.)

//©: HTML entitás, amely a copyright © szimbólumot jeleníti meg

//<?= date('Y') ?>: PHP rövid kiíró szintaxisa, amely kiírja az aktuális évet (pl. 2025)

A footer.php fájl a weboldal záró szekcióját tartalmazza. Dinamikusan jeleníti meg az aktuális évet a date('Y') függvény segítségével, így nem kell évente manuálisan frissíteni a láblécet. A sablon HTML-elemeket tartalmaz, és minden oldal végén megjelenik az egységes megjelenés érdekében.

templates/header.php

templates > header.php

```php
<!DOCTYPE html>
<html lang="hu">
<head>
    <meta charset="UTF-8">
    <title>PCParts MiniShop</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
<header>
    <h1>PC Parts Shop</h1>
    <nav>
        <ul>
            <?php
            foreach ($menu as $key => $value) {
                echo "<li><a href='index.php?page=$key'>$value</a></li>";
            }

            if (isset($_SESSION['user'])) {
                echo "<li><a href='index.php?page=logout'>Kilépés</a></li>";
                echo "<li>Bejelentkezett: {$_SESSION['user']['name']}</li>";
            } else {
                echo "<li><a href='index.php?page=login'>Belépés</a></li>";
            }
            ?>
        </ul>
    </nav>
</header>
<main>

```

//A HTML dokumentum fejléce (encoding, cím, CSS stílus)

//A style.css a megjelenésért felelős

//A <header> szakasz a logót/címet és a menüt tartalmazza

//A <nav> HTML5 elem a navigációs sávot jelöli

//A $menu tömbből (amit config.php tartalmaz) automatikusan kiírja az összes menüpontot

//Így elég csak egy helyen módosítani a menüt

//Dinamikusan megjeleníti, hogy a felhasználó be van-e jelentkezve

//Ha igen, megjelenik a kilépés gomb és a név

//Ha nem, akkor csak a belépés link

A header.php tartalmazza a weboldal sablon fejlécét, amely automatikusan beépül minden oldalra. A menüpontokat a config.php fájl $menu tömbje alapján generálja, így egyszerűen bővíthető vagy módosítható a navigáció. A felhasználó bejelentkezési állapota alapján dinamikusan változnak a megjelenített elemek: ha be van jelentkezve, látszik a kilépés és a neve; ha nem, csak a belépés opció érhető el.

uploads/db/init.sql

uploads > db > init.sql

```sql
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    lastname VARCHAR(50) NOT NULL,
    firstname VARCHAR(50) NOT NULL,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL
);

CREATE TABLE gallery (
    id INT AUTO_INCREMENT PRIMARY KEY,
    filename VARCHAR(255) NOT NULL,
    uploaded_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE messages (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    message TEXT NOT NULL,
    sent_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    sender VARCHAR(100) DEFAULT 'Vendég'
);
```

//id: egyedi azonosító (automatikusan növekvő)

//lastname és firstname: felhasználó nevei

//username: egyedi felhasználónév (nem lehet két azonos)

//password: titkosított jelszó (VARCHAR 255, a password_hash() miatt)

//filename: a feltöltött fájl neve

//uploaded_at: automatikusan beállított feltöltési dátum

//name, email, message: beküldött űrlap mezők

//sent_at: automatikus időbélyeg

//sender: bejelentkezett felhasználónév vagy „Vendég”

Az init.sql fájl az adatbázis struktúráját definiálja. Három táblát hoz létre: users, gallery, messages. A users táblában minden regisztrált felhasználó adatai szerepelnek, a gallery a feltöltött képek nevét és idejét tárolja, míg a messages tábla a kapcsolatfelvételi űrlapból beérkezett üzeneteket kezeli. Az időbélyegek és alapértelmezett értékek segítik az adatok naplózását és visszakereshetőségét.

uploads/js/validation.js

uploads > js > JS validation.js > ...

```
1   function validateContactForm() {
2     const name = document.querySelector('[name="name"]').value.trim();
3     const email = document.querySelector('[name="email"]').value.trim();
4     const message = document.querySelector('[name="message"]').value.trim();
5
6     if (!name || !email || !message) {
7       alert('Kérlek tölts ki minden mezőt!');
8       return false;
9     }
10
11    const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
12    if (!emailPattern.test(email)) {
13      alert('Hibás e-mail formátum!');
14      return false;
15    }
16
17    return true;
18  }
19
```

//Lekéri az űrlap három mezőjének (név, e-mail, üzenet) értékét, és eltávolítja a felesleges szóközöket.

//Ha bármelyik mező üres, alert jelenik meg és az űrlap nem kerül beküldésre.

//Egyszerű reguláris kifejezéssel ellenőrzi, hogy az e-mail cím érvényes-e (pl. valami@valami.hu)

//Ha minden ellenőrzés sikeres, az űrlap beküldhető.

A validation.js fájl biztosítja a kapcsolatfelvételi űrlap kliens oldali ellenőrzését. Ez a megoldás javítja a felhasználói élményt, mivel azonnal visszajelzést ad hibás vagy hiányzó mezőkről, még mielőtt az űrlap az adatbázisba kerülne. Az e-mail cím formátuma is ellenőrzésre kerül, hogy megelőzze az érvénytelen bejegyzéseket. A script az űrlap onsubmit eseményéhez van rendelve.

A PC Parts MiniShop projekt célja egy egyszerű, felhasználóbarát webshop létrehozása volt, amely számítógép-alkatrészek bemutatására és kezelésére alkalmas. A projekt során modern webes technológiákra (HTML, CSS, JavaScript, PHP, MySQL) építve valósult meg a teljes funkcionalitás – regisztráció, bejelentkezés, kapcsolatfelvétel, képgaléria és tartalomkezelés.

A feladat során külön figyelmet fordítottam a biztonságra (pl. jelszó-hash, szerveroldali ellenőrzés), a felhasználói élményre (kliensoldali validáció, dinamikus menük), valamint az átlátható kódstruktúrára és dokumentációra. A projekt minden fontos részét külön GitHub commitokban is rögzítettem, időben elosztva.