**Vizsgaremek Dokumentáció**

***Kétkerekű Ördög Fattyai***

DSZC Beregszászi Pál Technikum és Kollégium

**Projekt készítői**: Ozsvárt Bence, Maták Etele Zétény

**2023 - 2024**

**Tartalomjegyzék:**

[Bevezetés 1](#_Toc164414658)

[Témaválasztás 2](#_Toc164414659)

[Téma felosztása 3](#_Toc164414660)

[Adatbázis 4](#_Toc164414661)

[Frontend 6](#_Toc164414662)

[HTML Kód 6](#_Toc164414663)

[CSS stílusok 6](#_Toc164414664)

[JavaScript Kód 6](#_Toc164414665)

[Felhasználói felület elemek 7](#_Toc164414666)

[Reszponzív design 7](#_Toc164414667)

[Tesztelés és hibakeresés 7](#_Toc164414668)

[Backend 8](#_Toc164414669)

[Konstruktor metódus 8](#_Toc164414670)

[Prepare metódus 9](#_Toc164414671)

[Regisztráció metódus 9](#_Toc164414672)

[Összesmotor metódus 11](#_Toc164414673)

[Kiválasztott motor 11](#_Toc164414674)

[Hirdetéskészítés 13](#_Toc164414675)

[Felmerült problémák 14](#_Toc164414676)

# Bevezetés

A kétkerekű ördög fattyai weboldal dokumentációja mélyebb betekintést kínál az oldal működésébe és használatába, hogy a felhasználók könnyedén eligazodhassanak az egyedi dizájnnal és általános funkciókkal rendelkező motoroknak szánt webáruház világában.

Az útmutató részletesen bemutatja a különböző funkciókat és lehetőségeket, segítve ezzel az ügyfeleket abban, hogy teljes mértékben kihasználják az oldal nyújtotta előnyöket.

Emellett az integrációs lehetőségekről is átfogó tájékoztatást nyújtunk, hogy az üzleti folyamatok egyszerűsödjenek és hatékonyabbá váljanak. Bízunk benne, hogy ez a dokumentáció segítséget nyújt az oldal használatában és hozzájárul a felhasználói élmény fokozásához.

A dokumentációban részletes útmutatókat találhatnak a regisztrációs folyamattól kezdve a termékek böngészésén és vásárlásán át egészen a fizetési és szállítási lehetőségekig.

Emellett bemutatjuk a fiókbeállítások kezelését és az ügyfélszolgálati lehetőségeket is, hogy az ügyfelek zökkenőmentesen élvezhessék az oldal minden előnyét. A dokumentáció folyamatosan frissül, hogy lépést tartson az új funkciókkal és fejlesztésekkel, így mindig az aktuális és releváns információkat nyújtja az oldal használóinak.

A dokumentáció célja az is, hogy segítséget nyújtson az esetleges problémák megoldásában és az optimalizált felhasználói élmény elérésében.

Ennek érdekében részletesen ismertetjük a leggyakoribb kérdéseket és problémákat, valamint javaslatokat és útmutatást adunk azok megoldásához.

Az oldal fejlesztése és működtetése során felmerülő újabb kihívásokra is figyelemmel vagyunk, és ennek megfelelően frissítjük és bővítjük a dokumentációt, hogy a felhasználók mindig teljes körű támogatást kapjanak az oldal használatához.

# Témaválasztás

A témaválasztás során az volt a kulcsfontosságú szempont, hogy olyan területre koncentráljunk, amely iránt szenvedélyesen érdeklődünk.

Az órák webáruháza ötlet a kezdeti lépésként született meg, azonban további kutatás és párbeszéd után felismertük, hogy a motorokkal kapcsolatos vállalkozás sokkal jobban illeszkedik a közös érdeklődésünkhöz és szakértelmünkhöz.

Az áttérés a motorokkal foglalkozó webáruházra stratégiai döntés volt, mivel ez a téma nemcsak szorosan kapcsolódik a szenvedélyünkhöz, hanem nagyobb piaci potenciált is rejt magában. Ezzel együtt a motorok világa számos lehetőséget kínál a kreativitás kibontakoztatására és az üzleti növekedésre.

Így a témaválasztás folyamata végül egy olyan irányba terelődött, amely nemcsak a személyes érdeklődésünket és szenvedélyünket tükrözi, hanem üzleti szempontból is jövedelmező lehetőséget kínál számunkra.

# Téma felosztása

A témafelosztás után a csapat két részre szakadt, azonban mindkét rész a szakértelmét és tudását mind a frontend, mind a backend területén kamatoztatta.

Ez az átfedés lehetővé tette számunkra, hogy rugalmasan reagáljunk a felmerülő kihívásokra és problémákra. Például, amikor egyikünknek problémája merült fel a frontend fejlesztés során, a másik csapattag segítséget tudott nyújtani a megoldásban vagy alternatív megközelítéseket tudott felvetni.

Ez a szoros együttműködés és kölcsönös támogatás nemcsak hatékonyabbá tette a fejlesztést, hanem lehetővé tette számunkra, hogy egymástól tanuljunk és fejlődjünk.

Ezenkívül, az átfedés révén mindketten szélesebb körű ismeretekre tettünk szert mind a frontend, mind a backend területén, ami hosszú távon előnyös lehet a szakmai fejlődésünk szempontjából.

Ez a megközelítés továbbá erősítette a csapatmunka és az együttműködés kultúráját, amely létfontosságú egy sikeres projekt megvalósításához.

Ahelyett, hogy szigorúan különválasztották volna a feladatköröket, rugalmasan alkalmazkodtunk az adott helyzethez és szükség esetén átvettük egymás feladatait, hogy biztosítsuk a projekt zavartalan haladását és sikerét.

# Adatbázis

Az oldal tartalma egy MySQL adatbázisban van tárolva, amelyet három részre lehet osztani: users, motor és vásárlás táblákra.

Ezek a táblák különféle adatokat tartalmaznak, amelyek elengedhetetlenek az oldal működéséhez és funkcionalitásához.

Az "users" tábla a felhasználók adatait tárolja, ideértve a felhasználó nevét, e-mail címét, jelszavát és egyedi azonosítóját.

Ezek az adatok regisztrációkor kerülnek rögzítésre, és szolgálnak azonosításra és hitelesítésre a felhasználó bejelentkezésekor.

A "motor" tábla a motorokkal kapcsolatos információkat tartalmazza, például a motor típusát, gyártóját, évjáratát, árát és egyedi azonosítóját.

Ezek az adatok segítenek az oldalon elérhető motorok átfogó listázásában és megjelenítésében, valamint az egyes termékek részletes leírásában.

A "vásárlás" tábla rögzíti a vásárlásokkal kapcsolatos információkat, ideértve az adott termék azonosítóját, a vásárlás időpontját és a vásárló felhasználó azonosítóját.

Ez az adatbázis segít nyomon követni és kezelni az egyes vásárlásokat az oldalon, valamint lehetővé teszi a felhasználók számára, hogy visszakereshessék korábbi vásárlásaikat és adataikat.

Ez a strukturált adatbázis réteg lehetővé teszi az oldal számára, hogy hatékonyan tárolja és kezelje a felhasználókhoz és motorokhoz kapcsolódó adatokat, valamint biztosítja a vásárlások nyomon követését és kezelését az üzleti folyamat során.

Az adatbázis táblák összekapcsolása az egyedi azonosítók (ID-k) alapján történik. A users tábla egyedi felhasználó azonosítója (userid) kapcsolódik a vásárlás tábla felhasználó azonosítójához (userid), majd ez a vásárlások táblából további kapcsolatot képez a motor tábla motor azonosítójához (motorid).

A users táblában a userid egyedi azonosítóval rendelkezik, amely egyedi azonosítót biztosít minden felhasználó számára.

A vásárlás táblában a userid oszlop kapcsolódik a users tábla userid oszlopához, így a felhasználói adatok elérhetők a vásárlásokhoz kapcsolódóan.

A vásárlás táblában a motorid oszlop kapcsolódik a motor tábla motorid oszlopához, így a vásárlásokhoz kapcsolódó motoradatok elérhetők.

# Frontend

## HTML Kód

Az oldal struktúrájának és tartalmának leírása HTML elemek segítségével definiáljuk az oldal fejlécét, láblécét, tartalmi részeit, űrlapokat, gombokat és egyéb felhasználói felületi elemeket.

## CSS stílusok

A Bootstrap 5.0.2 keretrendszer segítségével definiáljuk a CSS stílusokat az oldal megjelenésének testreszabásához. Ez a stílusrendszer lehetővé teszi a színek, betűtípusok, elrendezések és egyéb vizuális tulajdonságok egyedi beállítását, miközben egységes és reszponzív dizájnt biztosít az oldal számára. Ezen kívül a saját stílusbeállításainkat is alkalmaztuk a weboldal fejlesztése során.

## JavaScript Kód

Az oldal interaktivitásának és dinamizmusának biztosítása JavaScript segítségével történik. Ebben a részben lehetőség van az alertek, footer, menu és egyéb frontend funkcionalitások implementálására, hogy felhasználóink még jobb élményben részesüljenek az oldal használata során.

## Felhasználói felület elemek

Az oldal különböző felhasználói felületi elemeket tartalmaz, mint például gombok, űrlapok, menük, modális ablakok és egyéb vizuális elemek. Ezek az elemek teszik lehetővé a felhasználók számára az oldal interakcióját és használatát.

## Reszponzív design

A Bootstrap keretrendszer segítségével implementáljuk a reszponzív design-t az oldalra, amely lehetővé teszi az oldal megfelelő megjelenítését és működését különböző eszközökön és kijelzőméreteken, így biztosítva a felhasználók számára egyenletes és testre szabott felhasználói élményt.

## Tesztelés és hibakeresés

A frontend fejlesztés során kiemelten fontos a tesztelés és hibakeresés folyamata, amelynek során ellenőrizzük az oldal megjelenését és működését különböző böngészőkben és eszközökön. Ezen kívül javítjuk az esetleges hibákat és problémákat, hogy a felhasználók zavartalanul és hatékonyan tudjanak használni az oldalt.

# Backend

A backend részt ahogy a fentiekben is említettük PHP programozási nyelven hoztuk létre, első sorban az adatbázis osztályra szeretnénk kitérni, hogy mit tartalmaz.

Maga a Database.php többféle funkciót tárol amire egyesével részletesen kitérünk, hogy a könnyen értelmezhető és kezelhető lehessen.

## Konstruktor metódus

Elsősorban a konstruktor adatbázis kapcsolatot hoz létre egy MySQL adatbázishoz, használva a megadott felhasználónevet, jelszót és adatbázisnevet.

A kapcsolat létrehozása után beállítja a karakterkódolást UTF-8-ra. Ha bármilyen hiba történik a kapcsolódás során, a kód kezeli a kivételt, beállít egy hibajelzőt, és megjeleníti az adatbázis elérhetetlenségét jelző üzenetet.

public function \_\_construct($host, $username, $pass, $db) {

try {

$this->db = new mysqli($host, $username, $pass, $db);

$this->db->set\_charset("utf8");

} catch (Exception $ex) {

$this->error = true;

echo '<p>Az adatbázis nem elérhető!</p>';

exit();}}

## Prepare metódus

Következő a prepare metódus lenne, amit a fióktörléshez volt szükséges létrehozni ez a kód azt takarja magába, hogy előkészítse és visszaadja egy adatbázis-lekérdezés (**$sql**) előkészített állapotát.

Ha van aktív adatbázis kapcsolat (**$this->db**), akkor az előkészített állapotot visszaadja.

Ha nincs ilyen kapcsolat, kivételt dob egy megfelelő hibaüzenettel, amely jelzi, hogy nincs inicializálva az adatbázis kapcsolat.

A kód így biztosítja, hogy csak akkor próbáljon meg lekérdezést előkészíteni, ha rendelkezésre áll a szükséges adatbázis kapcsolat.

public function prepare($sql) {

if ($this->db) {

return $this->db->prepare($sql);

} else {

throw new Exception("Nincs adatbázis kapcsolat inicializálva.");}}

## Regisztráció metódus

Ez a függvény a felhasználó regisztrációját végzi az alkalmazásban. A négy paraméterként kapott érték a regisztrációhoz szükséges adatokat tartalmazza: vezetéknév, keresztnév, e-mail cím és jelszó.

Az eljárás előkészíti a felhasználó adatainak beszúrását az adatbázisba egy SQL INSERT parancs segítségével. A felhasználó által megadott adatokat a megfelelő helyekre helyezi az adatbázisban.

Ha a beszúrás sikeres volt, beállítja a munkamenetváltozókat (**$\_SESSION**) a felhasználó bejelentkezett állapotának jelzésére, és a regisztrált felhasználó azonosítóját is hozzárendeli a munkamenethez (**$\_SESSION['userid']**).

Ha a beszúrás nem sikerült, akkor a függvény visszaállítja a bejelentkezési állapotot hamisra (**$\_SESSION['login'] = false**) és kiír egy üzenetet a sikertelenségről.

Ha bármilyen kivétel (Exception) keletkezik a végrehajtás során, a függvény beállítja az **error** változót igazra, de jelenlegi formájában nem történik semmilyen további kezelés a kivétellel kapcsolatban.

$stmt = $this->db->prepare("INSERT INTO `users`(`vezeteknev`, `keresztnev`, `e-mail\_cim`, `jelszo`, `userid`) VALUES (?,?,?,?,NULL)");

$stmt->bind\_param("ssss", $vezeteknev, $keresztnev, $email, $password);

try {

if ($stmt->execute()) {

$\_SESSION['login'] = true;

$\_SESSION['userid'] = $this->db->insert\_id;

} else {

$\_SESSION['login'] = false;

echo '<p>Rögzítés sikertelen!</p>';

}} catch (Exception $ex) {

$this->error = true;}}

## Összesmotor metódus

Ez a metódus az összes motort visszaadja az adatbázisból. Az eljárás végrehajt egy SELECT lekérdezést, amely kiválasztja az összes sort a **motor** táblából.

A lekérdezés eredményét egy asszociatív tömb formájában adja vissza, amely minden sorban a mezők neveit használja kulcsokként. Ez a függvény segít megjeleníteni az összes motort az index.php-n.

public function osszesmotor() {

$result = $this->db->query("SELECT \* FROM `motor`");

return $result->fetch\_all(MYSQLI\_ASSOC);}

## Kiválasztott motor

Ez a függvény egy adott motornak adatait kérdezi le az adatbázisból a motor azonosítója alapján.

Az eljárás előkészíti a lekérdezést, amely kiválasztja az adott motor adatait a **motor** táblából, ahol a **motorid** oszlop értéke megegyezik a paraméterként kapott **$motorId**-val. Ezután végrehajtja a lekérdezést, és az eredményt egy változóban tárolja.

Ha található legalább egy sor az eredménytáblában, akkor a függvény visszaadja az első talált sort asszociatív tömbként, amely az adott motor adatait tartalmazza.

Ha nem található egyetlen sor sem az eredménytáblában, akkor a függvény null értéket ad vissza, jelezve, hogy a keresett motor nem található az adatbázisban.

Ez a módszer lehetővé teszi az alkalmazás számára, hogy lekérdezze egy adott motor adatait az azonosítója alapján, és kezelje a lehetséges hiányzó adatokat.

$stmt = $this->db->prepare("SELECT \* FROM `motor` WHERE `motorid` = ?");

$stmt->bind\_param("i", $motorId);

$stmt->execute();

$result = $stmt->get\_result();

if ($result->num\_rows > 0) {

return $result->fetch\_assoc();

} else { return null; }}

## Hirdetéskészítés

Ez a függvény lehetővé teszi egy motorhirdetés feladását az adatbázisba. A függvény paraméterei a motor tulajdonságait tartalmazzák, mint például a gyártó, típus, évjárat, állapot, köbcentiméter, jogosítvány típusa, ár és teljesítmény kW-ban.

Az eljárás előkészíti a beszúrást az adatbázisba egy INSERT parancs segítségével. A paraméterként kapott értékeket helyettesíti a megfelelő helyekre a lekérdezésben. Ezután végrehajtja a beszúrást az adatbázisba.

Ha a beszúrás sikeres volt, a függvény üzenetet jelenít meg a felhasználónak, hogy a hirdetés sikeresen fel lett adva. Ha a beszúrás nem sikerült, akkor hibaüzenetet jelenít meg, amely tartalmazza a hiba okát (**$stmt->error**), és jelez a felhasználónak, hogy a hirdetés feladása sikertelen volt.

Ha bármilyen kivétel (Exception) keletkezik a végrehajtás során, akkor a függvény beállítja az **error** változót igazra, és kiírja a hibaüzenetet (**$ex->getMessage()**), valamint kiírja a teljes végrehajtott állapotot (**var\_dump($stmt)**), hogy segítse a hibakeresést és a debuggolást.

$stmt = $this->db->prepare("INSERT INTO `motor`(`gyarto`, `tipus`, `evjarat`, `allapot`, `kobcenti`, `jogositvany`, `ar`, `kW`) VALUES (?,?,?,?,?,?,?,?)");

$stmt->bind\_param("ssisisss", $gyarto, $tipus, $evjarat, $allapot, $kobcenti, $jogositvany, $ar, $kW);

try {if ($stmt->execute()) {

echo 'Hirdetés sikeresen feladva!';} else {

echo 'Hirdetés feladása sikertelen! Hiba: ' . $stmt->error;}} catch (Exception $ex) {

$this->error = true;echo 'Hiba: ' . $ex->getMessage(); var\_dump($stmt);}}

# Felmerült problémák

A project fejlesztése alatt kisebb-nagyobb problémák is felmerültek, viszont sikeresen megoldást találtunk rájuk és most már rendeltetés szerint működik minden funkció.

Az első kisebb probléma, hogy a bejelentkezett felhasználó nevét megakartuk jelenteni, de az összes felhasználó megjelent, ami adatbázisban volt tárolva. Ezt sikeresen orvosolni tudtuk úgy, hogy egy új „currentUser” változót hoztunk létre és tároljuk le benne a „userid”-t, amivel sikeresen megoldottuk a problémát.

A vásárlás során történt probléma, hogy csak egy vásárlást tudott adatbázisban tárolni a probléma abban merült fel, hogy az adatbázisban kellett módosítani a „userid” és motorid értékeit, mert nem egyeztek meg és most már sikeresen több vásárlást is rögzít az adatbázis.

A fióktörlés esetén nem lehetett kitörölni a felhasználót, hogyha vásárlás van kötve az adatót felhasználóhoz. Ez úgy került megoldásra, hogyha a felhasználó szeretné törölni a fiókját akkor a vásárlásai is törlésre kerülnek az adatbázisból és így sikeresen végrehajtható.

A felhasználómódosítás esetén is szükség volt a „currentUser” változóra, mivel azt akartuk, hogy automatikusan betöltésre kerüljenek a bejelentkezetett felhasználó adatai az input mezőben és csak ennek a segítségével tudtuk megoldani.