**Vizsgaremek Dokumentáció**

***Kétkerekű Ördög Fattyai***

DSZC Beregszászi Pál Technikum és Kollégium

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

**2023 - 2024**

**Tartalomjegyzék:**

[Bevezetés 1](#_Toc165020377)

[Témaválasztás 2](#_Toc165020378)

[Téma felosztása 3](#_Toc165020379)

[Adatbázis 5](#_Toc165020380)

[Frontend 7](#_Toc165020381)

[HTML Kód 7](#_Toc165020382)

[CSS stílusok 7](#_Toc165020383)

[JavaScript Kód 7](#_Toc165020384)

[Felhasználói felület elemek 8](#_Toc165020385)

[Reszponzív design 8](#_Toc165020386)

[Tesztelés és hibakeresés 8](#_Toc165020387)

[Backend 9](#_Toc165020388)

[Konstruktor 9](#_Toc165020389)

[Prepare 10](#_Toc165020390)

[Regisztráció 10](#_Toc165020391)

[Összesmotor 13](#_Toc165020392)

[Kiválasztott 13](#_Toc165020393)

[Hirdetéskészítés 15](#_Toc165020394)

[Felhasználó módosítás 17](#_Toc165020395)

[Felhasználó törlés 19](#_Toc165020396)

[GetUserByEmail 20](#_Toc165020397)

[HashPassword és VerifyPassword 20](#_Toc165020398)

[Vásárlás 22](#_Toc165020399)

[Felmerült problémák 24](#_Toc165020400)

# 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 alapvető szempont volt, hogy olyan területre összpontosítsunk, amely iránt szenvedélyesen érdeklődünk és ahol elmélyült szakértelmünk van.

Kezdetben az órák webáruháza ötletének megvalósításán gondolkodtunk, de miután további kutatást végeztünk, és mélyebb párbeszédet folytattunk csapaton belül, rájöttünk, hogy a motorokkal kapcsolatos vállalkozás sokkal jobban összhangban van mind az érdeklődésünkkel, mind a szakértelmünkkel.

Az órák webáruházának ötletéről a motorokkal foglalkozó webáruházra történő áttérés több szempontból is stratégiai döntés volt. Először is, a motorok világa számos olyan aspektust kínál, amelyhez már szoros kötődésünk van. Ezek az aspektusok magukban foglalják a technikai tudást, a motorozás kultúráját és a közösségi hálózatot. Ez az ismeret nemcsak előnyt jelent számunkra az üzleti életben, hanem segít a hitelesség és a szakértelem megerősítésében is.

Másodszor, a motorok piaca jelentős lehetőségeket kínál a növekedésre és a kreatív üzleti modellek kidolgozására. Az órák piaca viszonylag telített és hagyományos üzleti modellekre épül, míg a motorok területe sokkal több innovációs lehetőséget nyújt. A motoros közösség erősen összetartó és aktív, ami lehetőséget biztosít számunkra, hogy közvetlenül kapcsolódjunk az ügyfeleinkhez, és személyre szabott szolgáltatásokat nyújtsunk.

Továbbá, a motoros világ számos kapcsolódó üzleti területet is magában foglal, amelyek lehetőséget adnak a portfólió bővítésére. Ilyen területek lehetnek például a motoros kiegészítők, ruházat, karbantartási szolgáltatások, és a motoros turizmus. Ezek az ágazatok lehetővé teszik számunkra, hogy szélesebb körben gondolkodjunk az üzleti stratégiánkról, és több lábon álljunk az üzleti növekedés során.

Végül, a témaválasztás folyamatában az is fontos volt számunkra, hogy olyan üzleti területet találjunk, amely hosszú távon is fenntartható. A motorok piaca nem csak dinamikus és izgalmas, hanem állandó fejlődést és megújulást is kínál, ami biztosítja, hogy üzletünk hosszú távon is releváns és sikeres maradjon.

Összefoglalva, az órák webáruházától a motorokkal foglalkozó webáruházra való áttérés stratégiai döntés volt, amely tükrözi közös szenvedélyünket, miközben üzleti szempontból is jelentős potenciált kínál számunkra. A motorok világa nemcsak üzletileg vonzó, hanem számos lehetőséget kínál a kreativitás kibontakoztatására és az üzleti növekedésre is.

# Téma felosztása

A témafelosztás után a csapat két részre szakadt, azonban a feladatkörök összefonódása és a kölcsönös szakértelem kiaknázása révén mind a frontend, mind a backend területén képesek voltunk hatékonyan együtt dolgozni. Ez a megközelítés számos előnnyel járt, mivel lehetővé tette számunkra, hogy rugalmasan kezeljük a felmerülő problémákat és gyorsan reagáljunk a projekt követelményeire.

Az átfedés révén mindketten nemcsak a saját munkánkra összpontosíthattunk, hanem segítséget is tudtunk nyújtani másiknak, amikor szükséges volt. Például, ha a frontend oldalon felmerült egy komplikált PHP-hiba, a backend fejlesztő is segíthetett a probléma megoldásában a saját programozási ismereteivel. Hasonlóképpen, ha a backend oldalon volt szükség adatbázis-optimalizálásra vagy API-finomításra, a frontend fejlesztő is hozzá tudott járulni ötletekkel vagy javaslatokkal.

Ez a fajta együttműködés erősítette a csapatszellemet és a kölcsönös támogatás kultúráját, ami nemcsak a projekt haladását tette gördülékenyebbé, hanem lehetővé tette mindkettőnk számára, hogy egymástól tanuljunk és szélesebb körű ismeretekre tegyünk szert. A projekt során szerzett tapasztalatok hosszú távon is előnyösek lesznek, mivel a sokoldalú szakértelem a gyorsan változó technológiai környezetben előnyt jelent.

Az átfedés és az együttműködés megközelítése továbbá növelte a csapat összetartását, ami elengedhetetlen a sikeres projekt megvalósításához. Ahelyett, hogy szigorúan elkülönített feladatkörökkel dolgoztunk volna, mindketten hajlandóak voltunk rugalmasan alkalmazkodni az aktuális helyzethez és szükség esetén átvette a másik szerepét. Ez a hozzáállás biztosította a projekt folyamatos haladását, még a kihívásokkal teli időszakokban is.

Ezenkívül az ilyen típusú együttműködés elősegítette az innovációt és a kreatív gondolkodást. A különböző területekről származó ötletek és nézőpontok keveredése segített új megoldások kidolgozásában és a projekt színvonalának növelésében. Az együttműködés során kialakult erős bizalom és kommunikációs készség szintén kulcsfontosságú volt a projekt sikeres lebonyolításához és a csapat hatékonyságának növeléséhez.

Összefoglalva, a csapat rugalmas és együttműködő megközelítése nemcsak a projekt hatékonyságát növelte, hanem hozzájárult a csapat szakmai fejlődéséhez és a projekt általános sikeréhez. Ez a fajta együttműködés nemcsak az egyéni képességek kiaknázását tette lehetővé, hanem erősítette a csapatszellemet és az innovatív gondolkodást, amelyek kulcsfontosságúak egy sikeres vállalkozás számára.

# 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.

A képen szöveg, szoftver, szám, Betűtípus látható

Automatikusan generált leírás

# 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

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

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

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;}}

A képen szöveg, képernyőkép, multimédia, Mobiltelefon látható

Automatikusan generált leírás

## Összesmotor

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

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; }}

A képen szöveg, kerék, képernyőkép, Autóalkatrész látható

Automatikusan generált leírás

## 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);}}

A képen szöveg, képernyőkép, tervezés látható

Automatikusan generált leírás

## Felhasználó módosítás

Ez a PHP függvény frissíti egy felhasználó adatait az adatbázisban a felhasználó azonosítójának, vezetéknévnek, keresztnévnek, e-mail címnek és opcionálisan a jelszónak a használatával. Ha a jelszó meg van adva, akkor a függvény a jelszót biztonságosan hashelve tárolja az adatbázisban. A frissítés eredményétől függően sikeres vagy sikertelen frissítési üzenetet hoz létre, amelyet a felhasználói felületnek küld tovább egy munkameneti változóban, majd átirányítja a felhasználót egy másik oldalra (index.php). Ha hiba történik a frissítés során, akkor azt is naplózza, és megjeleníti az üzenetben.

public function frissitFelhasznalo($userid, $vezeteknev, $keresztnev, $email, $jelszo) {

try {

if (!is\_null($jelszo)) {

$jelszo = password\_hash($jelszo, PASSWORD\_DEFAULT);

$sql = "UPDATE `users` SET `vezeteknev` = ?, `keresztnev` = ?, `e-mail\_cim` = ?, `jelszo` = ? WHERE `userid` = ?";

$stmt = $this->db->prepare($sql);

$stmt->bind\_param('ssssi', $vezeteknev, $keresztnev, $email, $jelszo, $userid);

} else {

$sql = "UPDATE `users` SET `vezeteknev` = ?, `keresztnev` = ?, `e-mail\_cim` = ? WHERE `userid` = ?";

$stmt = $this->db->prepare($sql);

$stmt->bind\_param('sssi', $vezeteknev, $keresztnev, $email, $userid);}

if ($stmt->execute()) {$frissitesUzenet = "Sikeres frissítés!";}

else {$frissitesUzenet = "A frissítés nem sikerült! Hiba: " . $stmt->error;}} catch (Exception $e) {$frissitesUzenet = "Hiba a felhasználó frissítésekor: " . $e->getMessage();}

$\_SESSION['frissitesUzenet'] = $frissitesUzenet;

header("Location: index.php");

exit();}

A képen szöveg, képernyőkép, Mobiltelefon látható

Automatikusan generált leírás

## Felhasználó törlés

Ez a függvény töröl egy felhasználót az adatbázisból az e-mail cím és a jelszó alapján. Először elkészít egy SQL törlési parancsot, majd biztonságosan hasheli a megadott jelszót. A törlési parancsot az e-mail cím és a hashelt jelszó felhasználásával hajtja végre. Ha a művelet sikeres, hibanaplóban rögzíti a sikeres törlést és igaz értékkel tér vissza. Ha a törlés sikertelen, akkor az okát is naplózza, és hamis értékkel tér vissza.

public function felhasznaloTorles($email, $jelszo) {

$sql = "DELETE FROM users WHERE `e-mail\_cim` = ? AND `jelszo` =?";

$stmt = $this->db->prepare($sql);

if ($stmt) {

$hashedPassword = $this->hashPassword($jelszo);

$stmt->bind\_param("ss", $email, $hashedPassword);

if ($stmt->execute()) {

error\_log('Sikeres felhasználó törlés: ' . $email);

return true;}

else {

error\_log('Sikertelen felhasználó törlés: ' . $stmt->error);

return false;}}

else {

error\_log('Felhasználó törlés előkészítése sikertelen: ' . $stmt->error);

return false;}}

### GetUserByEmail

A függvény visszaadja a felhasználó adatait az adatbázisból az e-mail cím alapján. Ha talál egyező felhasználót, a teljes felhasználói adatokat egy társított tömbben adja vissza. Ha nincs ilyen felhasználó, null értéket ad vissza.

public function getUserByEmail($email) {

$stmt = $this->db->prepare("SELECT \* FROM users WHERE `e-mail\_cim` = ?");

$stmt->bind\_param("s", $email);

$stmt->execute();

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

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

return $result->fetch\_assoc();}

else {return null;}}

### HashPassword és VerifyPassword

A hashPassword függvény a megadott jelszót hasheli egy biztonságos algoritmussal, jelen esetben a PASSWORD\_DEFAULT használatával, amely a legújabb biztonsági szabványoknak megfelelően generál hashelt jelszót.

public function hashPassword($jelszo) {

return password\_hash($jelszo, PASSWORD\_DEFAULT);

}

A verifyPassword függvény a megadott jelszót ellenőrzi egy már hashelt jelszóval. Ha a jelszó megfelel a hashelt értéknek, igaz értékkel tér vissza, egyébként hamissal. Ezt a módszert használják például felhasználói hitelesítéskor.

public function verifyPassword($jelszo, $hashedPassword) {

return password\_verify($jelszo, $hashedPassword);

}

A képen szöveg, képernyőkép, multimédia, kütyü látható

Automatikusan generált leírás

## Vásárlás

Ez a függvény egy új vásárlási rekordot szúr be az adatbázisba, tartalmazva az irányítószámot, várost, utca nevet és házszámot, emelet és ajtót, telefonszámot, felhasználói azonosítót, valamint a motor azonosítóját. Ha a beszúrás sikeres, a függvény egy üzenetet jelenít meg a sikeres vásárlásról, majd átirányítja a felhasználót a kezdőlapra. Ha a beszúrás sikertelen, hibaüzenetet jelenít meg és visszaviszi a felhasználót a bejelentkezési oldalra.

public function vasarlasLeadasa($irszam, $varos, $unev\_hszam, $emelet\_ajto, $telefonszam, $userid, $motorid){

$stmt = $this->db->prepare("INSERT INTO `vasarlas` (`irszam`, `varos`, `unev\_hszam`, `emelet\_ajto`, `telefonszam`, `userid`, `motorid`) VALUES (?,?,?,?,?,?,?)");

$stmt->bind\_param("issssii", $irszam, $varos, $unev\_hszam, $emelet\_ajto, $telefonszam, $userid, $motorid);

try{

if ($stmt->execute()){

echo '<p>Vásárlás sikeresen leadva!</p>';

header("Location: index.php");}

else{

echo '<p>Rögzítés sikertelen</p>';

header("Location: index.php?menu=Bejelentkezes");}}

catch (Exception $ex) {

$this->error = true;}

}

A képen szöveg, kerék, képernyőkép, Autóalkatrész látható

Automatikusan generált leírás

# 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.