Dokumentáció

GypoWinery

Bevezetés

A Gypo Winery egy modern és elegáns borászat, amely a legkiválóbb borok kínálatával várja a látogatókat. Az oldal letisztult dizájnnal és reszponzív megjelenítéssel készült, hogy minden eszközön kényelmes böngészést biztosítson. A felhasználók számára különleges funkciók, például egyedi menük és sötét mód teszik még élvezetesebbé az élményt.

A index.php fájl egy PHP-alapú weboldal kezdőlapja, amely a Gypo Winery nevű weboldal megjelenítéséért felelős. Az oldal betöltéséhez külső konfigurációs fájlt (index\_config.php) importál, majd HTML és CSS segítségével kialakítja a vizuális megjelenést.

**Főbb elemek:**

* **Fejléc (head)**:
  + A karakterkódolás UTF-8.
  + A reszponzivitás érdekében beállított viewport.
  + Az oldal címe: Gypo Winery.
  + Betölt több CSS fájlt, köztük a Bootstrap framework legújabb verzióját, egyedi stíluslapokat (style.css, user-menu.css, darkmode.css).
* **Külső fájlok használata**:
  + Az index\_config.php tartalmazhat konfigurációs beállításokat, például adatbáziskapcsolatokat vagy egyéb környezeti változókat.

### register.php

#### 1. **Cél:**

Ez a kód egy regisztrációs űrlapot valósít meg, amely lehetővé teszi a felhasználók számára a saját adataik (vezetéknév, keresztnév, e-mail, telefonszám és jelszó) regisztrálását egy MySQL adatbázisba. Az adatok mentésre kerülnek, és a felhasználót a regisztráció után a bejelentkező oldalra irányítják.

#### 2. **Működés lépései:**

**2.1. Adatbázis kapcsolat létrehozása**

php

Másolás

$servername = "localhost"; // Adatbázis szerver címe

$username = "root"; // Felhasználónév

$password = ""; // Jelszó

$dbname = "gypowinery"; // Adatbázis neve

$conn = new mysqli($servername, $username, $password, $dbname);

* A kód egy MySQL adatbázishoz kapcsolódik a megadott szerver, felhasználónév, jelszó és adatbázis név használatával.
* Ha a kapcsolat nem sikerül, a connect\_error változóval hibát generálunk.

**2.2. Regisztrációs űrlap feldolgozása**

php

Másolás

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$vezeteknev = $\_POST['vezeteknev'];

$keresztnev = $\_POST['keresztnev'];

$email = $\_POST['email'];

$telefon = $\_POST['telefonszam'];

$jelszo = $\_POST['jelszo'];

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

$sql = "INSERT INTO login (vezeteknev, keresztnev, email, telefonszam, jelszo)

VALUES ('$vezeteknev', '$keresztnev', '$email', '$telefon', '$hashedPassword')";

* A kód a POST módszerrel elküldött adatokat (vezetéknév, keresztnév, email, telefonszám és jelszó) dolgozza fel.
* A jelszó titkosítva lesz a password\_hash() függvény segítségével.
* Az adatokat egy SQL INSERT INTO lekérdezéssel rögzíti az adatbázisban.

**2.3. Lekérdezés futtatása és eredmény kezelése**

php

Másolás

if ($conn->query($sql) === TRUE) {

header("Location: login.php");

exit();

} else {

echo "Hiba történt: " . $conn->error;

}

* Az SQL lekérdezés végrehajtása után, ha sikeres, a felhasználót átirányítjuk a login.php oldalra.
* Ha hiba történik, a hibát kiíratjuk a képernyőre.

**2.4. Kapcsolat bezárása**

php

Másolás

$conn->close();

* A MySQL kapcsolat lezárása.

#### 3. **HTML és Formázás:**

**3.1. HTML űrlap:**

* A felhasználó adatainak bekérésére szolgáló űrlap, amely a POST metódussal küldi el az adatokat a register.php oldalra.
* Az űrlapon található input mezők:
  + **Vezetéknév**
  + **Keresztnév**
  + **E-mail cím** (ellenőrzi, hogy tartalmazza az @ jelet)
  + **Telefonszám** (csak számokat enged, 8-15 karakter hosszú)
  + **Jelszó** (legalább 8 karakter, minimum 2 számot tartalmazó jelszó)

**3.2. Kép megjelenítése:**

* A showImage() JavaScript függvény biztosítja, hogy a regisztrációs űrlap alatti kép megjelenjen, amikor a felhasználó a "Sign up" feliratra kattint.
* A kép teljesen kitölti a kijelölt területet (object-fit: cover).

#### 4. **JavaScript:**

**4.1. Űrlap validálás**

javascript

Másolás

function validateForm() {

let email = document.getElementById("email").value;

let password = document.getElementById("jelszo").value;

if (!email.includes("@")) {

alert("Az e-mail címnek tartalmaznia kell egy '@' jelet!");

return false;

}

let passwordRegex = /^(?=.\*\d.\*\d)[A-Za-z\d]{8,}$/;

if (!passwordRegex.test(password)) {

alert("A jelszónak legalább 8 karakter hosszúnak kell lennie és minimum 2 számot kell tartalmaznia!");

return false;

}

return true;

}

* Az űrlap elküldése előtt a JavaScript validálja az e-mail és a jelszó mezőket.
* Az e-mailnek tartalmaznia kell egy @ jelet, és a jelszónak legalább 8 karakter hosszúnak kell lennie, valamint minimum 2 számot kell tartalmaznia.

## login.php

Ez a dokumentum a login.php fájl működését ismerteti. A szkript egy egyszerű bejelentkezési rendszert valósít meg, amely ellenőrzi a felhasználók hitelesítését egy MySQL adatbázisban.

## 2. Működés részletesen

### 2.1 Session indítás

A session segítségével a rendszer megjegyzi a bejelentkezett felhasználót:

session\_start();

### 2.2 Adatbázis kapcsolat

A szkript egy MySQL adatbázishoz csatlakozik a bejelentkezéshez:

$servername = "localhost";  
$username = "root";  
$password = "";  
$dbname = "gypowinery";  
$conn = new mysqli($servername, $username, $password, $dbname);  
  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}

* A mysqli osztály használatával csatlakozik az adatbázishoz.
* Ha a kapcsolat sikertelen, a szkript kilép egy hibaüzenettel.

### 2.3 Felhasználó adatainak ellenőrzése

A bejelentkezési adatok POST módszerrel érkeznek:

$email = $\_POST['email'];  
$jelszo = $\_POST['jelszo'];

A megadott email cím alapján a felhasználó adatait az adatbázisból lekéri:

$sql = "SELECT \* FROM login WHERE email = '$email'";  
$result = $conn->query($sql);

### 2.4 Jelszó ellenőrzés

Ha a felhasználó létezik, a megadott jelszót ellenőrizzük:

if ($result->num\_rows > 0) {  
    $row = $result->fetch\_assoc();  
    if (password\_verify($jelszo, $row['jelszo'])) {  
        $\_SESSION['user\_id'] = $row['id'];  
        echo "Sikeres bejelentkezés!";  
    } else {  
        echo "Hibás jelszó!";  
    }  
}

* **password\_verify()**: Ellenőrzi a titkosított jelszót.
* Ha helyes, a felhasználó ID-ja session-be kerül.
* Ha helytelen, hibaüzenet jelenik meg.

### logout.php

#### 1. **Cél:**

Ez a PHP kód egy egyszerű kijelentkezési funkciót valósít meg, amely törli a felhasználó session adatokat, beállít egy kijelentkezési üzenetet, majd visszairányítja a felhasználót a bejelentkező oldalra.

#### 2. **Működés lépései:**

**2.1. Session indítása**

php

Másolás

session\_start();

* A session\_start() függvény elindítja vagy folytatja a felhasználó munkamenetét (session). Ha még nem létezett munkamenet, akkor egy új session-t hoz létre, különben folytatja a meglévőt.
* Ez a függvény szükséges minden session kezelés előtt, például a session adatainak eléréséhez vagy módosításához.

**2.2. Session változók törlése**

php

Másolás

session\_unset();

* A session\_unset() függvény törli az összes session változót. Ezáltal eltávolítja azokat az adatokat, amelyeket a session során a felhasználó számára tároltak.
* A session változók csak a szerver oldalon tárolódnak, tehát a felhasználó böngészőjében nem változik semmi.

**2.3. Session megsemmisítése**

php

Másolás

session\_destroy();

* A session\_destroy() függvény véglegesen lezárja a munkamenetet, eltávolítva a session-t és annak adatait a szerverről.
* Ez azt jelenti, hogy az összes session adat (például bejelentkezési állapot, felhasználói beállítások) véglegesen törlődik.

**2.4. Új session indítása és kijelentkezési üzenet beállítása**

php

Másolás

session\_start();

$\_SESSION['logout\_message'] = "Sikeresen kijelentkeztél!";

* Egy új session-t indítunk, hogy új session változókat beállíthassunk.
* Beállítunk egy új session változót (logout\_message), amely a kijelentkezés sikerességét jelző üzenetet tartalmazza: "Sikeresen kijelentkeztél!"

**2.5. Visszairányítás a bejelentkező oldalra**

php

Másolás

header("Location: index.php");

exit();

* A header() függvénnyel beállítunk egy HTTP helyettesítést, amely a felhasználót az "index.php" oldalra irányítja. Az index.php az oldal, ahová a felhasználót a kijelentkezés után visszairányítjuk (általában egy bejelentkező oldal).
* A exit() függvény biztosítja, hogy a további kód végrehajtása ne folytatódjon, miután az irányítás átkerült az új oldalra.

#### 3. **Funkciók és viselkedés:**

* **Session Kezelés:** A kód egyszerűen törli a session adatokat, lehetővé téve a felhasználó számára a kijelentkezést.
* **Kijelentkezési üzenet:** A felhasználó egy visszajelzést kap a sikeres kijelentkezésről. Ez az üzenet az $\_SESSION['logout\_message'] változóban tárolódik, és más oldalon ki lehet olvasni.
* **Visszairányítás:** A felhasználót automatikusan visszairányítja a bejelentkező oldalra, miután sikeresen kijelentkezett.

Kviz.php

A Kviz.php fájl egy kvízoldalt biztosít a Gypo Winery weboldal számára, amely interaktív módon teszteli a felhasználók borokkal kapcsolatos tudását. Az oldal a config2.php fájlon keresztül kapcsolódik az adatbázishoz, és Bootstrap-alapú stílusokat használ a reszponzív megjelenítéshez.

A kvíz működéséhez egy külső JavaScript fájl (kviz.js) is betöltődik, amely valószínűleg az interaktív kérdés-válasz logikát kezeli. A felhasználói élmény fokozására az oldal támogatja a sötét módot, és tartalmaz egyedi stílusokat a user-menu.css és style.css fájlokból.

**Főbb funkciók:**

* Kvízkérdések megjelenítése a felhasználók számára
* Bootstrap és egyedi CSS segítségével reszponzív dizájn
* Külső JavaScript fájl biztosítja az interaktív funkciókat
* Adatbáziskapcsolat a config2.php fájlon keresztül

# add\_to\_cart.php

## Áttekintés

Ez a fájl egy PHP szkript, amely egy webshop kosárkezelő funkcióját biztosítja. A felhasználók hozzáadhatnak termékeket a kosarukhoz, és ha a termék már szerepel benne, akkor frissíti annak mennyiségét.

## Főbb Funkciók

### 1. Konfiguráció Betöltése

A fájl betölti a **config.php** fájlt, amely feltételezhetően az adatbázis-kapcsolatot és egyéb beállításokat tartalmazza:

include 'config.php';

### 2. Felhasználói Bejelentkezés Ellenőrzése

A szkript ellenőrzi, hogy a felhasználó be van-e jelentkezve. Ha nem, akkor JSON formátumban egy hibát ad vissza és kilép a végrehajtásból:

if (!isset($\_SESSION['user\_id'])) {

echo json\_encode(["error" => "Nincs bejelentkezve."]);

exit;

}

### 3. Bejövő Adatok Feldolgozása

A következő változók a **POST kéréssel érkeznek**, és a felhasználó azonosítóját, a kiválasztott termék azonosítóját és a hozzáadni kívánt mennyiséget tárolják:

$user\_id = $\_SESSION['user\_id'];

$bor\_id = $\_POST['bor\_id']; // Termék ID

$quantity = $\_POST['quantity']; // Mennyiség

### 4. Kosár Ellenőrzése

A rendszer ellenőrzi, hogy az adott termék már szerepel-e a kosárban:

$stmt = $pdo->prepare("SELECT \* FROM cart WHERE user\_id = :user\_id AND bor\_id = :bor\_id");

$stmt->execute(['user\_id' => $user\_id, 'bor\_id' => $bor\_id]);

$item = $stmt->fetch();

### 5. Termék Hozzáadása vagy Frissítése

* **Ha a termék már létezik**, akkor az új mennyiséget a meglévő mennyiséghez adja hozzá:

if ($item) {

$stmt = $pdo->prepare("UPDATE cart SET quantity = quantity + :quantity WHERE user\_id = :user\_id AND bor\_id = :bor\_id");

$stmt->execute(['quantity' => $quantity, 'user\_id' => $user\_id, 'bor\_id' => $bor\_id]);

* **Ha a termék még nincs a kosárban**, akkor egy új bejegyzést hoz létre az adatbázisban:

} else {

$stmt = $pdo->prepare("INSERT INTO cart (user\_id, bor\_id, quantity) VALUES (:user\_id, :bor\_id, :quantity)");

$stmt->execute(['user\_id' => $user\_id, 'bor\_id' => $bor\_id, 'quantity' => $quantity]);

}

### 6. Válasz Küldése a Kliensnek

A szkript végül egy JSON választ küld vissza a frontend számára:

echo json\_encode(["success" => "Termék sikeresen hozzáadva a kosárhoz."]);

## Használati Útmutató

* A szkript **AJAX POST kéréseket** vár a kosárkezeléshez.
* A felhasználónak be kell jelentkeznie, ellenkező esetben a művelet sikertelen lesz.
* Az adatbázisnak tartalmaznia kell egy **cart** nevű táblát a következő oszlopokkal:
  + id (elsődleges kulcs)
  + user\_id (felhasználó azonosító)
  + bor\_id (termék azonosító)
  + quantity (darabszám)

## Fejlesztési Javaslatok

* **Hibakezelés fejlesztése**: Érdemes részletesebb hibaüzeneteket küldeni és naplózni a hibákat.
* **SQL injekció elleni védelem**: Bár a **PDO prepared statement** használata már biztonságos, érdemes további validálásokat végezni a bemeneti adatokon.
* **Kosár összegző funkció**: Egy külön endpoint implementálható a kosár tartalmának lekérdezésére és összegzésére.

Ez a szkript egy alapvető, de jól használható kosárkezelő megoldás egy webshop számára.

# config2.php

## Áttekintés

Ez a fájl egy PHP konfigurációs szkript, amely adatbázis-kapcsolatot kezel, valamint felhasználói munkameneteket indít. Az adatbázis eléréséhez **PDO (PHP Data Objects)** interfészt használ, amely biztonságos és rugalmas módja az SQL adatbázisok kezelésének.

## Főbb Funkciók

### 1. Munkamenet Kezelés

A fájl elején található az alábbi kód:

session\_start();

Ez elindítja a **PHP munkamenetet (session)**, amely lehetővé teszi az adatok tárolását a felhasználók között történő navigáció során.

### 2. Adatbázis Kapcsolat Beállításai

Az adatbázis eléréséhez a következő változók vannak meghatározva:

$host = 'localhost';

$dbname = 'gypowinery';

$username = 'root';

$password = '';

Ezek a beállítások határozzák meg az adatbázis elérhetőségét.

### 3. Adatbázis Kapcsolódás PDO-val

A fájl egy **try-catch blokkot** használ a biztonságos adatbázis-kapcsolat létrehozásához és a hibák kezeléséhez:

try {

$pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $username, $password);

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$pdo->setAttribute(PDO::ATTR\_DEFAULT\_FETCH\_MODE, PDO::FETCH\_ASSOC);

} catch (PDOException $e) {

die("Kapcsolódási hiba: " . $e->getMessage());

}

* **PDO objektum létrehozása**: Az adatbázis kapcsolatot a **PDO** osztály kezeli.
* **Hibakezelés**: Ha a kapcsolat sikertelen, a szkript leáll és kiírja a hibát.
* **Attribútumok beállítása**:
  + ERRMODE\_EXCEPTION: Az esetleges hibák kivételt generálnak, amelyet kezelni lehet.
  + FETCH\_ASSOC: Az adatokat asszociatív tömbként adja vissza.

### 4. Bejelentkezés Ellenőrzése

A fájl egy bejelentkezési ellenőrzési részt is tartalmazhat, amely a munkamenetben tárolt felhasználói adatok alapján azonosítja a bejelentkezett felhasználót.

## Használati Útmutató

* Győződj meg róla, hogy a **MySQL adatbázis fut**, és a **beállított hitelesítő adatok helyesek**.
* Ha jelszót használ az adatbázishoz, állítsd be a megfelelő értéket a **$password** változóban.
* A kapcsolódási hibák elkerülése érdekében ellenőrizd, hogy a megfelelő **PHP PDO kiterjesztés** engedélyezve van a szerveren.

## boraink.php

Ez a dokumentum a boraink.php fájl funkcionalitását ismerteti. A szkript egy adatbázisból kinyeri a borokat, megjeleníti azokat a felhasználónak, valamint kezeli a kosárba helyezés funkciót.

## 2. Működés részletesen

### 2.1 Borok listázása

A szkript egy **SQL-lekérdezést** hajt végre a borok és a hozzájuk tartozó képek lekérdezésére:

$query = "SELECT borok.\*, bor\_kepek.kep\_url FROM borok LEFT JOIN bor\_kepek ON borok.ID = bor\_kepek.bor\_id";

$stmt = $pdo->query($query);

$borok = $stmt->fetchAll(PDO::FETCH\_ASSOC);

**borok.\***: A borok minden adatát lekéri.

**LEFT JOIN bor\_kepek**: A borokhoz tartozó képeket csatolja.

**fetchAll(PDO::FETCH\_ASSOC)**: Az eredményt asszociatív tömbként adja vissza.

### 2.2 Kosár kezelés

Amikor a felhasználó egy bort a kosárba helyez, a következő folyamat történik:

Az űrlapon keresztül elküldött **bor azonosító** és **mennyiség** lekérése:

$bor\_id = $\_POST['bor\_id'];

$quantity = $\_POST['quantity'];

Ellenőrzés, hogy a bor már a kosárban van-e:

$user\_id = $\_SESSION['user\_id'];

$check\_query = "SELECT \* FROM cart WHERE user\_id = :user\_id AND bor\_id = :bor\_id";

$check\_stmt = $pdo->prepare($check\_query);

$check\_stmt->execute(['user\_id' => $user\_id, 'bor\_id' => $bor\_id]);

$existing\_item = $check\_stmt->fetch(PDO::FETCH\_ASSOC);

**Ha a bor már benne van**, akkor a mennyiség frissítése történik:

$new\_quantity = $existing\_item['quantity'] + $quantity;

$update\_query = "UPDATE cart SET quantity = :quantity WHERE user\_id = :user\_id AND bor\_id = :bor\_id";

$update\_stmt = $pdo->prepare($update\_query);

$update\_stmt->execute(['quantity' => $new\_quantity, 'user\_id' => $user\_id, 'bor\_id' => $bor\_id]);

**Ha nincs még a kosárban**, akkor beszúrás történik:

$insert\_query = "INSERT INTO cart (user\_id, bor\_id, quantity) VALUES (:user\_id, :bor\_id, :quantity)";

$insert\_stmt = $pdo->prepare($insert\_query);

$insert\_stmt->execute(['user\_id' => $user\_id, 'bor\_id' => $bor\_id, 'quantity' => $quantity]);

## 3. Hibakezelés

**Session ellenőrzés**: A $\_SESSION['user\_id'] létezését ellenőrizni kell, hogy a felhasználó be van-e jelentkezve.

**SQL injection elleni védelem**: A prepare() és execute() metódusok használata biztosítja a védelmet.

**Hibakezelés**: A try-catch szerkezet alkalmazása segíthet a PDO hibák kezelésében.

# admin\_login.php

## Áttekintés

Ez a fájl egy **PHP alapú adminisztrátori bejelentkezési rendszer**, amely ellenőrzi a felhasználó e-mail címét és jelszavát az adatbázisban. A hitelesítés sikeressége esetén munkamenetet indít az admin számára.

## Főbb Funkciók

### 1. Konfiguráció Betöltése

A fájl elején betölti a **config2.php** fájlt, amely feltételezhetően az adatbázis-kapcsolatot tartalmazza:

include('config2.php');

### 2. POST Adatok Fogadása

A szkript ellenőrzi, hogy az adatokat **POST** módszerrel küldték-e be:

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$email = isset($\_POST["email"]) ? trim($\_POST["email"]) : '';

$password = isset($\_POST["password"]) ? trim($\_POST["password"]) : '';

### 3. Üres Mezők Ellenőrzése

Ha az e-mail vagy a jelszó mező üres, a szkript hibaüzenetet ad vissza, majd kilép:

if (empty($email) || empty($password)) {

echo "Email and password fields cannot be empty.";

exit;

}

### 4. Felhasználó Keresése az Adatbázisban

A rendszer ellenőrzi, hogy létezik-e az adott e-mail cím az **admin felhasználók adatbázisában**:

try {

$sql = "SELECT ID, jelszo, usertype FROM login WHERE email = :email";

$stmt = $pdo->prepare($sql);

$stmt->execute(['email' => $email]);

$row = $stmt->fetch(PDO::FETCH\_ASSOC);

### 5. Jelszó Ellenőrzése és Munkamenet Indítása

* **Ha az e-mail cím létezik**, a rendszer összehasonlítja a tárolt hash-elt jelszót a megadott jelszóval:

if ($row && password\_verify($password, $row["jelszo"])) {

$\_SESSION['user\_id'] = $row['ID'];

$\_SESSION['usertype'] = $row['usertype'];

echo "Login successful";

} else {

echo "Invalid email or password.";

}

* Ha a jelszó helyes, a rendszer beállítja a munkameneti változókat (user\_id és usertype).
* Ha a jelszó nem megfelelő, a rendszer hibaüzenetet küld vissza.

## Használati utmutato

* Az admin felhasználók **e-mail címükkel és jelszavukkal** jelentkezhetnek be.
* A jelszavakat **hash-elt formában kell tárolni** az adatbázisban password\_hash() függvénnyel.
* Az usertype mező használható különböző jogosultsági szintek megkülönböztetésére.

Checkout.php:

Ez a PHP script kezeli a felhasználók kosarát és a rendelés leadását a Gypo Winery weboldalon. A rendszer az alábbi funkciókat tartalmazza:

* Kosár adatainak betöltése az adatbázisból
* Végösszeg kiszámítása szállítási díjjal és esetleges kedvezményekkel
* Rendelés adatok mentése az adatbázisba
* Raktárkészlet frissítése a rendelés alapján
* Kosár törlése a rendelés leadása után
* Rendelés visszaigazolása e-mailben a Web3Forms API segítségével
* Felhasználói felület a fizetési módok kiválasztására

## Adatbázis lekérdezések

### Kosár lekérdezése

SELECT cart.ID as cart\_id, borok.ID as bor\_id, borok.ar, borok.nev, cart.quantity

FROM cart

JOIN borok ON cart.bor\_id = borok.ID

WHERE cart.user\_id = :user\_id

**Leírás:** Betölti a felhasználó kosarában levő termékeket, azok árait és mennyiségeit.

### Rendelés rögzítése

INSERT INTO rendelesek (user\_id, statusz) VALUES (:user\_id, 'pending')

**Leírás:** Létrehozza az új rendelést "pending" állapottal.

### Rendelés tételek mentése

INSERT INTO rendeles\_tetelek (rendeles\_id, bor\_id, quantity) VALUES (:order\_id, :bor\_id, :quantity)

**Leírás:** Elmenti a rendelés termékeit a rendeles\_tetelek táblába.

### Készlet frissítése

UPDATE borok SET keszlet = keszlet - :quantity WHERE ID = :bor\_id

**Leírás:** Frissíti a raktárkészletet a rendelés mennyisége alapján.

### Kosár törlése

DELETE FROM cart WHERE user\_id = :user\_id

**Leírás:** Törli a felhasználó kosarát a rendelés leadása után.

### Felhasználó e-mail lekérdezése

SELECT email FROM login WHERE ID = :user\_id

**Leírás:** Lekéri a felhasználó regisztrált e-mail címét a visszaigazolás küldéséhez.

## Rendelés visszaigazolás e-mailben

A Web3Forms API segítségével a rendszer e-mailt küld a felhasználónak a rendelésről. Az e-mail tartalmazza a rendelési azonosítót, a tételeket és az összes fizetendő összeget.

A kérést cURL-lel hajtja végre:

$ch = curl\_init("https://api.web3forms.com/submit");

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

curl\_setopt($ch, CURLOPT\_POST, true);

curl\_setopt($ch, CURLOPT\_POSTFIELDS, $post\_fields);

$response = curl\_exec($ch);

curl\_close($ch);

## Frontend: Checkout oldal

Az oldal egy Bootstrap alapú felület, amely lehetőséget biztosít a felhasználónak a fizetési mód kiválasztására:

* **Készpénz**
* **Bankkártya** (rejtett mezők a kártyaadatoknak)
* **PayPal** (e-mail megadásával)

A JavaScript biztosítja, hogy a megfelelő fizetési mezők jelenjenek meg:

function togglePaymentDetails() {

var paymentMethod = document.getElementById("payment\_method").value;

var cardDetails = document.getElementById("card-details");

var paypalDetails = document.getElementById("paypal-details");

if (paymentMethod === "Bankkártya") {

cardDetails.style.display = "block";

paypalDetails.style.display = "none";

} else if (paymentMethod === "PayPal") {

cardDetails.style.display = "none";

paypalDetails.style.display = "block";

} else {

cardDetails.style.display = "none";

paypalDetails.style.display = "none";

}

}

## Biztonsági megfontolások

* **Adatbázis lekérdezések**: A PDO prepared statement-ek védettek SQL injection ellen.
* **Bankkártya adatok kezelése**: A kártyaadatok bevitele jelenleg nincs titkosítva, ezért valódi fizetési rendszert (pl. Stripe, PayPal API) kellene integrálni.
* **Session védelem**: A felhasználó azonosítása session változókon keresztül történik.

## Összegzés

Ez a checkout rendszer biztosítja a rendeléskezelés alapvető funkcionalitásait, beleértve a termékek betöltését, rendelés leadását és visszaigazolás küldését. Javasolt a további fejlesztés biztonsági és fizetési integrációs szempontból.

## 1. **admin\_borok.php**

Ez a PHP szkript egy **adminisztrációs felületet** biztosít a borok adatbázisban való kezelésére. Az adminisztrátor a következő funkciókat érheti el:

* **Borok listázása** (képekkel együtt)
* **Borok törlése** (kapcsolódó képekkel és rendelési tételekkel együtt)
* **Új bor hozzáadása** (képpel együtt)

A szkript a **borok** és **bor\_kepek** táblákhoz kapcsolódik, valamint kezeli a **rendeles\_tetelek** tábla kapcsolódó adatait is.

## 2. **Használt táblák**

### 2.1 borok

| Oszlop | Típus | Leírás |
| --- | --- | --- |
| ID | INT | Egyedi azonosító (PRIMARY KEY) |
| nev | VARCHAR | Bor neve |
| ar | INT | Bor ára |
| leiras | TEXT | Leírás |
| keszlet | INT | Készlet darabszám |

### 2.2 bor\_kepek

| Oszlop | Típus | Leírás |
| --- | --- | --- |
| id | INT | Egyedi azonosító (PRIMARY KEY) |
| bor\_id | INT | Hivatkozás a borok.ID-re (FOREIGN KEY) |
| kep\_url | VARCHAR | Kép URL címe |

### 2.3 rendeles\_tetelek

| Oszlop | Típus | Leírás |
| --- | --- | --- |
| id | INT | Egyedi azonosító |
| rendeles\_id | INT | Rendelés azonosítója |
| bor\_id | INT | Hivatkozás a borok.ID-re |
| mennyiseg | INT | Megrendelt mennyiség |

## 3. **Funkciók**

### 3.1. Új bor hozzáadása

* Az űrlapon keresztül az adminisztrátor megadja:
  + Bor neve
  + Ár
  + Leírás
  + Készlet
  + Kép URL
* A bor adatai bekerülnek a **borok** táblába.
* A borhoz tartozó kép URL-je bekerül a **bor\_kepek** táblába.

#### Kód részlet:

php

MásolásSzerkesztés

$query = "INSERT INTO borok (nev, ar, leiras, keszlet) VALUES (:nev, :ar, :leiras, :keszlet)";

$query\_kep = "INSERT INTO bor\_kepek (bor\_id, kep\_url) VALUES (:bor\_id, :kep\_url)";

### 3.2. Bor törlése

* Az adminisztrátor egy gombnyomással törölheti a kiválasztott bort.
* Törléskor a következők történnek:
  + A borhoz tartozó rendelési tételek törlődnek a **rendeles\_tetelek** táblából.
  + A borhoz tartozó képek törlődnek a **bor\_kepek** táblából.
  + Maga a bor törlődik a **borok** táblából.

#### Kód részlet:

php

MásolásSzerkesztés

$delete\_rendeles\_query = "DELETE FROM rendeles\_tetelek WHERE bor\_id = :wine\_id";

$delete\_kepek\_query = "DELETE FROM bor\_kepek WHERE bor\_id = :wine\_id";

$delete\_bor\_query = "DELETE FROM borok WHERE ID = :wine\_id";

### 3.3. Borok listázása

* A borok listázása egy **kártyás (card)** elrendezésben történik.
* Minden kártya tartalmazza:
  + Képet
  + Bor nevét
  + Leírást
  + Árat
  + Készletet
  + Törlés gombot

#### Kód részlet:

php

MásolásSzerkesztés

$query = "SELECT borok.\*, bor\_kepek.kep\_url FROM borok LEFT JOIN bor\_kepek ON borok.ID = bor\_kepek.bor\_id";

## 4. **Megjelenés (Frontend)**

### 4.1. Fejléc és navigáció

* Logó és oldalcím
* Navigációs sáv a főbb oldalak eléréséhez (Főoldal, Történet, Boraink stb.)
* Admin linkek (Admin, Admin rendelések)

### 4.2. Sötét mód funkció

* Sötét mód kapcsoló csúszkával
* A megjelenés sötét és világos között vált, amihez a **darkmode.js** fájl kapcsolódik

## 5. **Formátumok és Stílusok**

* **Bootstrap 5.3.3** a reszponzív elrendezéshez.
* Saját CSS fájlok:
  + style.css
  + darkmode.css
  + user-menu.css
  + darkmodecard.css

## 6. **Biztonsági Megjegyzések**

### 6.1. SQL Injection elleni védelem

* **Prepared Statement-ek** használata minden adatbázis művelethez:

php

MásolásSzerkesztés

$stmt = $pdo->prepare($query);

$stmt->bindParam(':nev', $nev);

### 6.2. Adminisztrátori jogosultság ellenőrzés

* **check\_admin()** függvény hívás, ami feltételezhetően egy jogosultság-ellenőrzés:

php

MásolásSzerkesztés

check\_admin();

## 7. **Fájlkapcsolatok**

| Fájl | Funkció |
| --- | --- |
| config2.php | Adatbázis kapcsolat és admin ellenőrzés |
| darkmode.js | Sötét mód működtetése |
| translate.js | Nyelvválasztó funkció |
| user-menu.js | Felhasználói menü vezérlése |
| bootstrap.min.css | Bootstrap keretrendszer |
| style.css | Egyedi stílusok |
| darkmode.css | Sötét módhoz szükséges stílusok |
| user-menu.css | Felhasználói menühöz szükséges stílusok |
| darkmodecard.css | Kártyák stílusa sötét módban |

**tortenet.php**

Ez a dokumentáció bemutatja a "Gypo Winery" weboldal történeti oldalának kódját, amely PHP és HTML alapú. A célja, hogy bemutassa a borászat történelmét, küldetését, közösségi elkötelezettségét, fenntarthatóságát és jövőbeli terveit. A kód modern webdesign elemeket használ, mint például Bootstrap, dark mode, és felhasználói menü.

**1. Fájlok és Könyvtárak**

A projekt a következő fájlokat és könyvtárakat tartalmazza:

* **config.php**: Ez a fájl tartalmazza a projekt konfigurációs beállításait, például adatbázis-kapcsolatokat és egyéb globális beállításokat.
* **CSS fájlok**: A projekt stílusát három fő CSS fájl biztosítja:
  + bootstrap.min.css: A Bootstrap keretrendszer, amely segíti a reszponzív design megvalósítását.
  + style.css: A weboldal egyéni stílusait tartalmazza.
  + user-menu.css: A felhasználói menühöz kapcsolódó stílusok.
  + darkmode.css: A sötét módhoz kapcsolódó CSS.
* **JavaScript fájlok**: A következő fájlok felelnek a funkciók és interaktív elemek működéséért:
  + user-menu.js: A felhasználói menü működtetése.
  + translate.js: A nyelvi váltás kezelésére szolgál (nem implementált teljesen).
  + darkmode.js: A sötét mód váltásáért felelős JavaScript fájl.
  + bootstrap.bundle.min.js: A Bootstrap interaktív elemeinek működtetéséhez szükséges JavaScript fájl.

**2. PHP és Adatbázis Kezelés**

A kód elején az include('config.php') sor a config.php fájlt emeli be, amely feltételezhetően az adatbázis kapcsolódáshoz szükséges információkat és beállításokat tartalmazza.

* **Felhasználói ellenőrzés**: A kód az isset($\_SESSION['usertype']) és isset($\_SESSION['user\_id']) feltételekkel ellenőrzi, hogy a felhasználó be van-e jelentkezve, és adminisztrátori jogokkal rendelkezik-e. Ha igen, akkor megjelennek az adminisztrátori menüpontok, például az "Admin" és "Admin rendelések" lehetőségek.

**3. HTML Struktúra**

A HTML oldal felépítése világos és reszponzív:

* **Fejléc (header)**: Tartalmazza a borászat logóját, a főcímet, valamint a nyelvi zászlók és a sötét mód kapcsolóját.
* **Navigációs menü (nav)**: A navigációs menü Bootstrap segítségével készült, és különböző oldalakat linkel ki, mint a főoldal, történet, boraink, kapcsolat, és a quiz oldal. A menü elemei egyesíthetők dinamikusan a felhasználói jogosultságokkal.
* **Történet szekciói**: A történeti szakaszok HTML alapú szöveges tartalommal és képekkel vannak ellátva, így az oldal vizuálisan is vonzó.
* A borászat kezdetei, fejlődése, küldetése, közösségi elkötelezettsége és fenntarthatósága mind külön szekciókban találhatóak.
* Az egyes szakaszok mindegyike egy címekkel és szövegekkel, valamint képekkel ellátott blokkot tartalmaz.
* **Footer**: A lábléc tartalmazza a copyright információkat.

**4. Sötét Mód**

A weboldal támogatja a sötét módot, amelyet egy kapcsolóval lehet aktiválni. Ehhez a HTML-ben található egy checkbox elem, amelynek az állapotát a JavaScript figyeli. Ha a felhasználó engedélyezi a sötét módot, akkor az oldal stílusa automatikusan sötétebbé válik, biztosítva a kényelmesebb olvasási élményt gyenge fényviszonyok között.

html

Másolás

<input type="checkbox" id="darkModeToggle">

**5. Felhasználói Menü**

A felhasználói menü egy legördülő lista, amely akkor jelenik meg, ha a felhasználó be van jelentkezve. Az alapértelmezett ikon egy felhasználói profil ikont ábrázol. A menü lehetőségei a következők:

* **Rendelés**: A felhasználók hozzáférhetnek a rendelési oldalukhoz.
* **Kijelentkezés**: A felhasználó kiléphet a fiókjából.

**6. Kosár Panel (Kosár funkció)**

A kód tartalmaz egy kosár panelt, amely alapértelmezés szerint rejtett. A cartPanel elem tartalmazza a kosár tartalmát, amely dinamikusan töltődik be JavaScript segítségével. A kosár funkció később valószínűleg egy e-kereskedelmi oldal részeként kerül implementálásra.

**7. Reszponzivitás és Design**

A weboldal a Bootstrap keretrendszert használja, amely biztosítja a reszponzivitást. A navigációs menü és az egyes szakaszok rugalmasan alkalmazkodnak a különböző képernyőméretekhez, így mobil eszközökön is jól használható.

* **Mobilbarát navigáció**: A navigációs menü egy hamburger menüként jelenik meg kisebb képernyőkön, biztosítva a könnyű navigációt mobil eszközökön.
* **Képek és szövegek**: Az oldalon található képek a img-fluid osztállyal rendelkeznek, hogy azok megfelelően skálázódjanak különböző képernyőméreteken.

#### cart.php

Feladata hogy kezelje a borok készletét, lehetővé téve a borok kosárba történő hozzáadását és eltávolítását, valamint az összes bor listázását.

#### 2. **Adatbázis Kapcsolat**

Az API az alábbi adatbázis-kapcsolati beállításokat használja:

* **Szerver**: localhost
* **Felhasználó**: root
* **Jelszó**: (üres, mivel az alapértelmezett XAMPP beállítást használjuk)
* **Adatbázis**: gypowinery

A kapcsolat mysqli osztállyal történik, és a hibák kezelése JSON válaszon keresztül történik.

#### 3. **API Műveletek**

Az API két fő HTTP metódust támogat:

* **POST**: A bor készletének módosítása (hozzáadás vagy eltávolítás).
* **GET**: A borok listázása az adatbázisból.

##### 3.1 **POST Kérés: Bor készletének módosítása**

A POST kérés kétféle műveletet támogathat: **add** és **remove**.

**Műveletek:**

* **add**: A bor készletének csökkentése, ha az a kosárba kerül.
* **remove**: A bor készletének növelése, ha a bor eltávolításra kerül a kosárból.

**Kérés Paraméterek**:

* **bor\_id** (kötelező): A bor azonosítója (egész szám).
* **action** (kötelező): A végrehajtandó művelet, lehet:
  + add: Bor hozzáadása a kosárhoz (készlet csökkentése).
  + remove: Bor eltávolítása a kosárból (készlet növelése).

**Példa kérés**:

bash

Másolás

POST /api

Content-Type: application/x-www-form-urlencoded

bor\_id=123&action=add

**Válasz**:

A válasz JSON formátumban érkezik, és tartalmazza a művelet eredményét.

* **Sikeres művelet**:

json

Másolás

{

"status": "success",

"message": "Termék hozzáadva a kosárhoz."

}

* **Hibás művelet**:

json

Másolás

{

"status": "error",

"message": "Nincs elég készlet."

}

**Hibakezelés**:

* **Kapcsolódási hiba**: Ha nem sikerül csatlakozni az adatbázishoz:

json

Másolás

{

"status": "error",

"message": "Kapcsolódási hiba: [hiba üzenet]"

}

##### 3.2 **GET Kérés: Borok listázása**

A GET kérés lekérdezi az összes bor adatait az adatbázisból.

**Példa kérés**:

bash

Másolás

GET /api

**Válasz**:

A válasz JSON formátumban tartalmazza az összes bor adatát. A válasz egy tömb, amely minden bor adatokat tartalmazó objektumot tartalmaz.

**Példa válasz**:

json

Másolás

[

{

"ID": 1,

"nev": "Cabernet Sauvignon",

"keszlet": 50,

"ar": 3500

},

{

"ID": 2,

"nev": "Chardonnay",

"keszlet": 20,

"ar": 3000

}

]

A válasz minden bor esetében tartalmazza az alábbi mezőket:

* **ID**: A bor egyedi azonosítója.
* **nev**: A bor neve.
* **keszlet**: A bor aktuális készlete.
* **ar**: A bor ára.

#### 4. **Hibaüzenetek**

A rendszer a következő hibaüzeneteket adhat vissza:

* **Kapcsolódási hiba**: Ha nem sikerül csatlakozni az adatbázishoz.
* **Ismeretlen művelet**: Ha az action paraméter nem érvényes (add vagy remove).
* **Termék nem található**: Ha a megadott bor\_id nem létezik az adatbázisban.
* **Nincs elég készlet**: Ha a add műveletet próbáljuk végrehajtani, de a bor készlete 0 vagy kevesebb.