**Reszponzív weboldal**

Az index.html fájlt nyissa meg. Állítsa be a bootstrap osztályokat úgy, hogy 3 kártya legyen egymás mellett képernyőn, 2 kártya tableten, és 1 kártya egy sorban mobilos nézetben.

**Backend feladat**

Adatbázis létrehozása:

Hozza létre a products2025.sql fájlt a következő tartalommal:

CREATE DATABASE IF NOT EXISTS product\_catalog;

USE product\_catalog;

-- Tábla létrehozása

CREATE TABLE IF NOT EXISTS products (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

description TEXT NOT NULL,

price DECIMAL(10, 2) NOT NULL,

image\_name VARCHAR(255) NOT NULL

);

-- Adatok beszúrása

INSERT INTO products (name, description, price, image\_name) VALUES

('Mountain Bike', 'A durable bike perfect for off-road trails and rough terrain.', 499.99, 'mountain\_bike.jpg'),

('Smartphone', 'A high-end smartphone with a sleek design and powerful features.', 899.99, 'smartphone.jpg'),

('Coffee Maker', 'Brew your perfect cup of coffee with this advanced coffee maker.', 79.99, 'coffee\_maker.jpg'),

('Gaming Laptop', 'A powerful laptop designed for gamers and heavy-duty tasks.', 1499.99, 'gaming\_laptop.jpg'),

('Wireless Headphones', 'Noise-cancelling headphones with crystal-clear sound.', 199.99, 'wireless\_headphones.jpg');

**Hozzon létre egy connect.php nevű fájlt, amelyben létrehozza az adatbázissal való kapcsolatot, és annak létrejöttét ellenőrzi.**

**Feladatleírás: REST API Készítése PHP-ben Termékkezeléshez**

Készítsen egy **REST API**-t PHP-ben, amely lehetővé teszi a termékek adatainak kezelését (lekérdezés, hozzáadás, frissítés, törlés). Az API a megadott **product2025** adatbázist fogja használni, amelyben a **products** tábla található.

**Elérhető Adatbázis Információk:**

Az **products** tábla a következő oszlopokat tartalmazza:

* **id** (INT, AUTO\_INCREMENT, PRIMARY KEY) – A termék egyedi azonosítója.
* **name** (VARCHAR) – A termék neve.
* **description** (TEXT) – A termék részletes leírása.
* **price** (DECIMAL) – A termék ára.
* **image\_name** (VARCHAR) – A termék képének fájlneve.

**Feladatok az API Létrehozásához:**

1. **Projektmappa szerkezetének kialakítása:**
   * Hozzon létre egy projektmappát **product\_api** néven.
   * Helyezze el benne az alábbi fájlokat:
     + **connect.php** – Az adatbázis kapcsolatot kezeli.
     + **api.php** – Az API végpontokat tartalmazó fájl.
2. **Kapcsolódás az adatbázishoz (connect.php):**
   * Hozza létre a **connect.php** fájlt.
   * Ebben a fájlban csatlakozzon a **product2025** adatbázishoz a következő adatokkal:
     + **Szerver:** localhost
     + **Felhasználónév:** root
     + **Jelszó:** (üres)
   * Ellenőrizze a kapcsolat sikerességét:
     + Sikeres kapcsolat esetén ne adjon vissza üzenetet.
     + Hiba esetén adjon vissza egy JSON formátumú hibaüzenetet, például:  
       {"status": "error", "message": "Adatbázis kapcsolódási hiba: <hiba leírása>"}
3. **API fájl létrehozása (api.php):**
   * Hozza létre az **api.php** fájlt, amely kezeli az API végpontokat.
   * Csatolja a **connect.php** fájlt a kapcsolat biztosításához.
4. **HTTP Fejlécek Beállítása:**
   * Állítsa be a következő HTTP fejléceket:
     + **Content-Type:** application/json; charset=utf-8
     + **Access-Control-Allow-Origin:** \*
     + **Access-Control-Allow-Methods:** GET, POST, PUT, DELETE, OPTIONS
5. **HTTP Metódusok Kezelése:**
   * Határozza meg az aktuális HTTP metódust a **$\_SERVER['REQUEST\_METHOD']** segítségével.
   * Az alábbi metódusokat kell kezelni:
     + **GET** – Adatok lekérése.
     + **POST** – Új adat hozzáadása.
     + **PUT** – Meglévő adat frissítése.
     + **DELETE** – Adat törlése.
6. **Azonosító (id) paraméter kezelése:**
   * Ellenőrizze, hogy az **id** paraméter átadásra került-e (**$\_GET['id']**).
   * Az **id** paraméter szükséges a **GET**, **PUT**, és **DELETE** kérések esetén.
7. **GET Kérés Kezelése:**
   * **Minden termék lekérése:**  
     Ha nincs megadva **id**, kérje le az összes terméket a **products** táblából.
     + Adja vissza az adatokat JSON formátumban:

{

"status": "success",

"data": [

{

"id": 1,

"name": "Mountain Bike",

"description": "A durable bike perfect for off-road trails.",

"price": 499.99,

"image\_name": "mountain\_bike.jpg"

},

...

]

}

* + **Egy adott termék lekérése:**  
    Ha meg van adva az **id**, kérje le az adott terméket.
    - Ha a termék létezik, adja vissza a részleteket JSON formátumban.
    - Ha a termék nem található, adjon vissza 404-es hibát:

{

"status": "error",

"message": "A megadott azonosítóval nem található termék: <id>"

}

1. **POST Kérés Kezelése:**
   * Hozzon létre egy új terméket a **products** táblában.
   * Az adatok JSON formátumban érkeznek a kérés törzsében:

{

"name": "New Product",

"description": "Product description",

"price": 99.99,

"image\_name": "new\_product.jpg"

}

* + Ellenőrizze, hogy minden szükséges mező (**name**, **description**, **price**, **image\_name**) meg van-e adva.
  + Sikeres létrehozás után adjon vissza 201-es státuszkódot és az új termék azonosítóját:

{

"status": "success",

"message": "Termék sikeresen létrehozva.",

"id": <új azonosító>

}

* + Hiányos adatok esetén küldjön vissza 400-as hibát:

{

"status": "error",

"message": "Hiányos adatok. Kérjük, töltse ki a név, leírás, ár és kép mezőket."

}

1. **PUT Kérés Kezelése:**
   * Frissítse egy meglévő termék adatait az **id** alapján.
   * Az új adatok JSON formátumban érkeznek.
   * Ellenőrizze, hogy minden mező (**name**, **description**, **price**, **image\_name**) meg van-e adva.
   * Sikeres frissítés esetén adjon vissza 200-as státuszkódot:

{

"status": "success",

"message": "Termék sikeresen frissítve."

}

* + Ha a termék nem található, adjon vissza 404-es hibát.

1. **DELETE Kérés Kezelése:**
   * Törölje az adott azonosítójú terméket.
   * Sikeres törlés esetén adjon vissza 204-es státuszkódot (nincs tartalom).
   * Ha a termék nem található, adjon vissza 404-es hibát:

{

"status": "error",

"message": "A megadott azonosítóval nem található termék: <id>"

}

1. **Hibakezelés:**
   * Minden művelet után biztosítson megfelelő hibakezelést:
     + **200** – Sikeres lekérdezés/frissítés.
     + **201** – Sikeres létrehozás.
     + **204** – Sikeres törlés, nincs tartalom.
     + **400** – Hiányos vagy hibás kérés.
     + **404** – Nem található a termék.
     + **500** – Szerverhiba.
2. **SQL Injekció Elleni Védelem:**
   * Használja a **mysqli\_real\_escape\_string** függvényt a bemeneti adatok biztonságos kezeléséhez.
3. **API Válaszok Formázása:**
   * Az összes válasz legyen JSON formátumban, és használja a következő opciókat a formázáshoz:
     + **JSON\_PRETTY\_PRINT** – Szép formázás.
     + **JSON\_UNESCAPED\_UNICODE** – Unicode karakterek megfelelő megjelenítése.
4. **API Tesztelése:**
   * Tesztelje az API végpontjait **Thunder Client** (VS Code kiegészítő) segítségével.
   * Ellenőrizze, hogy minden **GET**, **POST**, **PUT**, és **DELETE** kérés megfelelően működik.

### ****API Tesztelése Thunder Clienttel****

Miután a bővítmény telepítve van, kezdheti az API tesztelését.

#### **A. GET Kérés – Összes termék lekérése**

1. Nyissa meg a **Thunder Client**-et a bal oldali oldalsávon lévő villám ikonra kattintva.
2. Kattintson a **New Request** gombra.
3. A felső sávban válassza ki a **GET** metódust.
4. Az URL mezőbe írja be a következő címet:

http://localhost/product\_api/api.php

1. Kattintson a **Send** gombra.

**Mit várunk?**

* Ha az adatbázisban vannak termékek, a válasznak tartalmaznia kell az összes termék adatait JSON formátumban.
* Ha nincs adat, egy üres tömböt kell visszakapnia:

{

"status": "success",

"data": []

}

#### **B. GET Kérés – Egy adott termék lekérése**

1. Hozzon létre egy új kérés-t a **Thunder Client**-ben.
2. Válassza a **GET** metódust.
3. Az URL mezőbe írja be az alábbi címet, ahol az **id** paraméter a termék azonosítója:

http://localhost/product\_api/api.php?id=1

1. Kattintson a **Send** gombra.

**Mit várunk?**

* Ha az **id=1** termék létezik, a válasz tartalmazni fogja az adott termék adatait:

{

"status": "success",

"data": {

"id": 1,

"name": "Mountain Bike",

"description": "A durable bike perfect for off-road trails.",

"price": 499.99,

"image\_name": "mountain\_bike.jpg"

}

}

* Ha az adott **id** nem létezik, 404-es hibát kell kapnia:

{

"status": "error",

"message": "A megadott azonosítóval nem található termék: 1"

}

#### **C. POST Kérés – Új termék hozzáadása**

1. Hozzon létre egy új kérés-t a **Thunder Client**-ben.
2. Válassza a **POST** metódust.
3. Az URL mezőbe írja be:

http://localhost/product\_api/api.php

1. Lépjen a **Body** fülre, majd válassza ki a **JSON** formátumot.
2. Adja meg az új termék adatait a következő formátumban:

{

"name": "Wireless Mouse",

"description": "A sleek wireless mouse with ergonomic design.",

"price": 29.99,

"image\_name": "wireless\_mouse.jpg"

}

1. Kattintson a **Send** gombra.

**Mit várunk?**

* Sikeres létrehozás esetén a válasznak 201-es státuszkódot kell adnia:

{

"status": "success",

"message": "Termék sikeresen létrehozva.",

"id": 6

}

* Ha bármelyik mező hiányzik, 400-as hibát kell kapnia:

{

"status": "error",

"message": "Hiányos adatok. Kérjük, töltse ki a név, leírás, ár és kép mezőket."

}

#### **D. PUT Kérés – Meglévő termék frissítése**

1. Hozzon létre egy új kérés-t a **Thunder Client**-ben.
2. Válassza a **PUT** metódust.
3. Az URL mezőbe írja be a következőt (például a 6-os azonosítójú termék frissítéséhez):

http://localhost/product\_api/api.php?id=6

1. Lépjen a **Body** fülre, és válassza ki a **JSON** formátumot.
2. Adja meg a frissítendő adatokat:

{

"name": "Wireless Mouse Pro",

"description": "An upgraded wireless mouse with additional features.",

"price": 39.99,

"image\_name": "wireless\_mouse\_pro.jpg"

}

1. Kattintson a **Send** gombra.

**Mit várunk?**

* Sikeres frissítés esetén a válasz:

{

"status": "success",

"message": "Termék sikeresen frissítve."

}

* Ha a termék nem található, 404-es hibát kell kapnia.

#### **E. DELETE Kérés – Termék törlése**

1. Hozzon létre egy új kérés-t a **Thunder Client**-ben.
2. Válassza a **DELETE** metódust.
3. Az URL mezőbe írja be a törölni kívánt termék azonosítóját (például):

http://localhost/product\_api/api.php?id=6

1. Kattintson a **Send** gombra.

**Mit várunk?**

* Ha a törlés sikeres, a válasznak **204 No Content** státuszkódot kell visszaadnia (nem fog adatot mutatni).
* Ha a termék nem létezik, 404-es hibát kell kapnia:

{

"status": "error",

"message": "A megadott azonosítóval nem található termék: 6"

}

### ****Hibák Ellenőrzése Thunder Clienttel****

1. **Hiányos adatok esetén (POST vagy PUT kérésnél):**
   * Ha egy mező hiányzik, például az ár (**price**), a rendszer 400-as hibát ad vissza.
   * Próbáljon meg egy adatot kihagyni a kérés törzsében, hogy tesztelje a hibakezelést.
2. **Nem létező termék lekérése vagy törlése:**
   * Adjon meg egy nem létező **id**-t (pl. **id=9999**), és ellenőrizze, hogy megfelelő hibát kap.

**Frontend:**

**Termékkatalógus Megjelenítése AngularJS-sel az index.html Fájlban**

Készítsen egy **index.html** fájlt, amely **AngularJS** segítségével beolvassa a termékadatokat a **product.json** fájlból, és táblázatos formában jeleníti meg azokat a felhasználó számára. A termékekhez tartozó képek az **img** mappában találhatók, és szintén meg kell jeleníteni azokat.

**Feladatok az index.html Fájl Elkészítéséhez**

1. **Hozza létre az index.html fájlt** a **frontend** mappában.
2. **Állítsa be az alap HTML struktúrát**:
   * Definiálja az AngularJS alkalmazást az **ng-app** direktíva segítségével.
   * Használjon **ng-controller** direktívát a vezérlő meghatározásához.
3. **Importálja az AngularJS könyvtárat** a Google CDN-ről az **<head>** szakaszban.
4. **Hozza létre a táblázatot** az alábbi oszlopokkal:
   * **Kép** – a termékhez tartozó kép megjelenítése.
   * **Név** – a termék neve.
   * **Leírás** – a termék leírása.
   * **Ár** – a termék ára, helyi pénznemben megjelenítve.
5. **Olvassa be a product.json fájlt** AngularJS **$http** szolgáltatásával:
   * Használja az **ng-repeat** direktívát a termékek dinamikus megjelenítéséhez.
6. **Képek megjelenítése:**
   * A képeket az **img** mappából töltse be az **ng-src** direktívával.
   * Biztosítson hibakezelést arra az esetre, ha egy kép nem töltődne be.
7. **Hibaüzenet megjelenítése**:
   * Ha az adatok beolvasása sikertelen, jelenítsen meg egy hibaüzenetet a felhasználó számára.
   * Használja az **ng-if** direktívát a hibaállapot kezelésére.
8. **Stílusok hozzáadása**:
   * Formázza a táblázatot, hogy átlátható és reszponzív legyen.
   * A képek méretét állítsa egységesre (pl. 100px széles).
9. **Tesztelje a működést**:
   * Nyissa meg az **index.html** fájlt böngészőben, és ellenőrizze, hogy az adatok és képek helyesen jelennek meg.
   * Próbáljon meg hibás vagy hiányzó képfájlokat tesztelni, hogy ellenőrizze a hibakezelés működését.
10. **Mentse el a munkáját**, és győződjön meg arról, hogy a kód olvasható és jól strukturált.

minta:

