JEGYZŐKÖNYV

Szakmai Gyakorlat I.

Backend fejlesztési szakasz

Készítette: Orosz Kristóf

Neptunkód: EYZWG9

Dátum: 2025. június 13.

Orosz Kristóf

.....

Orosz Kristóf

Hallgató

.....

Dédesi Péter

Szakmai konzulens

Tartalomjegyzék

Bevezetés

A szakmai gyakorlatom egyik legfontosabb mérföldköve a backend fejlesztési szakasz volt, amely során lehetőségem nyílt egy valósághű, adatbázis-alapú rendszer megtervezésére és megvalósítására. A célom az volt, hogy egy időpontfoglaló alkalmazás teljes szerveroldali működését felépítsem, az alapoktól kezdve az összetettebb funkciókig – például az értékelési rendszer integrálásáig. A fejlesztést lépésről lépésre építettem fel: először létrehoztam a megfelelő lokális környezetet a XAMPP segítségével, majd kialakítottam az adatbázis szerkezetét, tesztadatokat töltöttem fel, és ehhez kapcsolódóan sorra megírtam az API végpontokat.

A fejlesztés során külön figyelmet fordítottam a hibakezelésre, az adatok validálására és arra, hogy minden funkció jól dokumentálható és Postman segítségével tesztelhető legyen. A funkcionális elemek – mint a regisztráció, bejelentkezés, szolgáltatók és időpontok kezelése, foglalások létrehozása – önállóan és együtt is működőképes rendszert alkotnak.

Fájlok tartalmának leírása

- Backend környezet kialakítása.pdf [PDF fájl az alábbi feladatrészről.]
- Adatbázis létrehozása.pdf [PDF fájl az alábbi feladatrészről.]
- Felhasználói regisztráció API.pdf [PDF fájl az alábbi feladatrészről.]
- Bejelentkezés API.pdf [PDF fájl az alábbi feladatrészről.]
- Szolgáltatók listázása API.pdf [PDF fájl az alábbi feladatrészről.]
- Szolgáltatók kezelése API.pdf [PDF fájl az alábbi feladatrészről.]
- Időpontok kezelése API.pdf [PDF fájl az alábbi feladatrészről.]
- Foglalások kezelése API.pdf [PDF fájl az alábbi feladatrészről.]
- Értékelések API.pdf [PDF fájl az alábbi feladatrészről.]
- Jogosultságkezelés, és admin funkciók.pdf [PDF fájl az alábbi feladatrészről.]
- JK – Backend fejlesztési szakasz.pdf [Jegyzőkönyv a backend szakasztól.]
- Forrásfájlok /Mappa/ [A szakasz forrásfájljait tartalmazó mappa.]

1.Feladatrészek

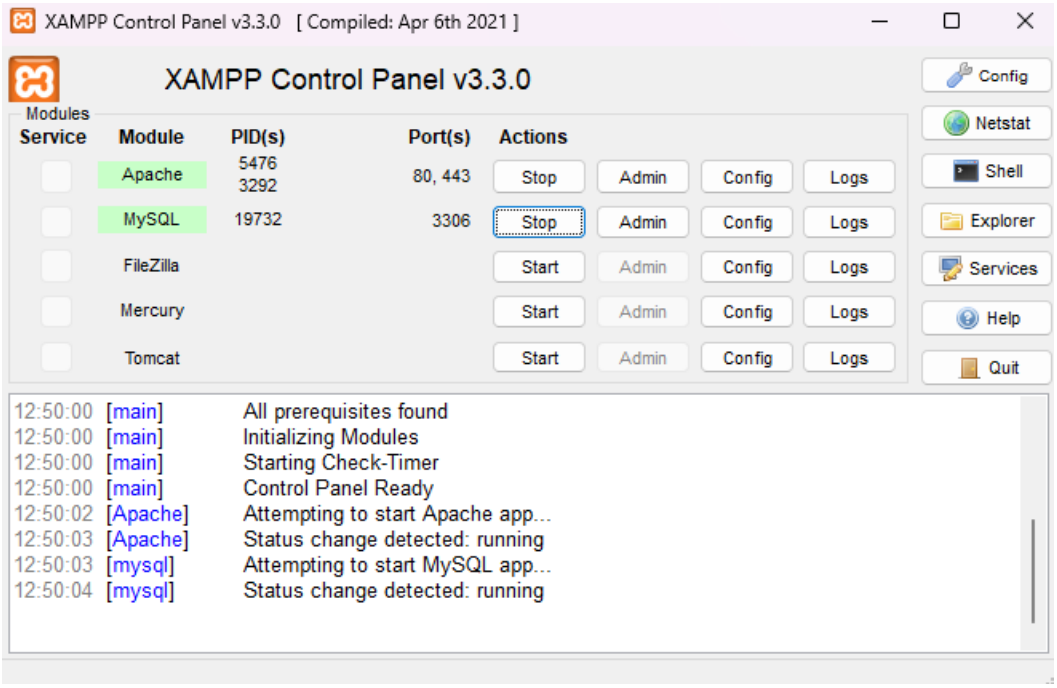
Az alábbi pontokban összefoglalom, hogy a backend fejlesztési szakaszban az egyes feladatrészek során milyen konkrét tevékenységeket végeztem.

1.1. Backend környezet kialakítása

A fejlesztés első lépéseként letöltöttem és telepítettem a XAMPP nevű helyi webszervert, amelynek segítségével saját gépen tudom tesztelni a webes alkalmazásom működését. Beállítottam az alapértelmezett útvonalat a C:/xampp/htdocs mappára, hogy a frontend fájlokat könnyen elérhessem és futtathassam. Ezen belül hoztam létre az idopontfoglalo mappát, ahol minden PHP fájl, adatbázis-elérés, és később frontend fájl is megtalálható.

A szerver elindítása után sikeresen leteszteltem a kapcsolatot egy egyszerű kapcsolat.php fájl segítségével, amely létrehozta a kapcsolatot az adatbázissal. A visszatérő válaszüzenet ({"status":"sikeres kapcsolat"}) megerősítette, hogy az adatbáziskapcsolat működik. Ezzel a lépéssel biztosítottam, hogy a backend oldali feldolgozások zökkenőmentesen kommunikáljanak az adatbázissal.

1. **XAMMP helyi szerver letöltése, elindítása.**



2. **Frontend oldali helyi webszerveres futtatás beállítása.**

   Alapértelmezett elérési frontend út: **C:/xampp/htdocs** beállítása a konfigurációs menüpont alatt.

```
#
#DocumentRoot "C:/xampp/htdocs"
#<Directory "C:/xampp/htdocs">
DocumentRoot "C:/xampp/htdocs"
<Directory "C:/xampp/htdocs">
```

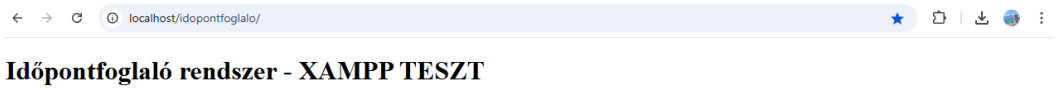
Idopontfoglalo mappa létrehozása az **xampp\htdocs** mappán belül.

C:\xampp\htdocs\idopontfoglalo\ indexe

[szülőkönyvtár]

Név	Méret	Módosítás dátuma
index.html	272 B	2025. 05. 30. 12:15:58
kapcsolat.php	393 B	2025. 05. 30. 14:28:11
teszt.php	144 B	2025. 05. 30. 14:31:19

3. Helyi XAMPP-s Frontend webszerver elindítása, tesztelése.



Időpontfoglaló rendszer - XAMPP TESZT

4. Adatbázis létrehozása, tesztadatokkal való feltöltése.

A folyamat leírása megtalálható az **Adatbázis létrehozása.pdf** fájlban.

5. Kapcsolat létesítése PHP segítségével a Backend és az Adatbázis között.

```
<?php
// Beállítom az adatbázis szerver elérhetőségét és adatait.
$dbHost = 'localhost';
$dbName = 'idopontfoglalo';
$dbUser = 'root';
$dbPass = '';

try {
    // A PDO objektum létrehozása a MySQL adatbázishoz.
    $pdo = new PDO(
        "mysql:host=$dbHost;dbname=$dbName;charset=utf8", // DSN: szerver,
adatbázis és karakterkészlet.
        $dbUser,
        $dbPass,
        [
            // Hibát íratok ki kivételként, ha bármi probléma adódik.
            PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
            // A lekérdezések eredményét asszociatív tömbként kérem.
            PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
        ]
    );
} catch (PDOException $e) {
    // Ha nem sikerül a kapcsolódás, JSON formátumban küldöm vissza a
hibaüzenetet.
    header('Content-Type: application/json');             // Beállítom a válasz
tartalmát JSON-ra.
    http_response_code(500);
    echo json_encode([
        'error' => 'Adatbázis kapcsolódási hiba: ' . $e->getMessage()

    ]);
    exit;
}
```

6. Kapcsolat ellenőrzése, tesztelése.

```php
<?php
// Betölti a kapcsolat.php fájlt, amely létrehozza az adatbázis-kapcsolatot.
require_once("kapcsolat.php");

// Beállítja, hogy a válasz JSON típusú legyen.
header('Content-Type: application/json');

// Egy JSON formátumú üzenetet küld vissza, amellyel jelzi, hogy a kapcsolat sikeres.
echo json_encode(["status" => "sikeres kapcsolat"]);
?>
```

Kapcsolat sikeres megjelenítése

1.2. Adatbázis létrehozása

Miután létrehoztam az idopontfoglalo nevű adatbázist a MariaDB-ben, következett a táblák megtervezése és kialakítása. Az alábbi táblákat definiáltam a projekt követelményei alapján.

A táblák létrehozását SQL utasításokkal végeztem el, majd mindegyikhez manuálisan feltöltöttem tesztadatokat a Postman segítségével vagy SQL parancsokkal. Ezzel biztosítottam, hogy a rendszer funkciói valós adatokkal tesztelhetők legyenek, a backend kommunikációját pedig minden ponton ellenőrizni tudjam.

Az ertekelesek tábla bevezetése kiegészítésként került a rendszerbe. Ez lehetővé teszi, hogy a felhasználók véleményt és pontszámot adjanak az igénybe vett szolgáltatásokról, de csak akkor, ha valóban foglaltak időpontot az adott szolgáltatónál.

Táblák, mezők létrehozása

Felhasznalok

```
MariaDB [idopontfoglalo]> CREATE TABLE felhasznalok (
    ->   id INT(50) PRIMARY KEY,
    ->   nev CHAR(100) NOT NULL,
    ->   szerepkor CHAR(50) NOT NULL,
    ->   email CHAR(100) NOT NULL UNIQUE,
    ->   jelszo CHAR(255) NOT NULL,
    ->   reg_datum DATETIME DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.018 sec)

MariaDB [idopontfoglalo]> DESCRIBE felhasznalok
    -> ;
+-----------+-----------+------+-----+---------------------+-------+
| Field     | Type      | Null | Key | Default             | Extra |
+-----------+-----------+------+-----+---------------------+-------+
| id        | int(50)   | NO   | PRI | NULL                |       |
| nev       | char(100) | NO   |     | NULL                |       |
| szerepkor | char(50)  | NO   |     | NULL                |       |
| email     | char(100) | NO   | UNI | NULL                |       |
| jelszo    | char(255) | NO   |     | NULL                |       |
| reg_datum | datetime  | YES  |     | current_timestamp() |       |
+-----------+-----------+------+-----+---------------------+-------+
6 rows in set (0.016 sec)
```

Szolgaltatok

```
MariaDB [idopontfoglalo]> CREATE TABLE szolgaltatok (
    ->   id INT(50) PRIMARY KEY,
    ->   szolgaltatas_tipusok_id INT(50) NOT NULL,
    ->   nev CHAR(100) NOT NULL,
    ->   leiras CHAR(255),
    ->   aktiv BOOLEAN DEFAULT TRUE,
    ->   FOREIGN KEY (szolgaltatas_tipusok_id) REFERENCES szolgaltatas_tipusok(id)
    -> );
Query OK, 0 rows affected (0.020 sec)

MariaDB [idopontfoglalo]> describe szolgaltatok;
+------------------------+------------+------+-----+---------+-------+
| Field                  | Type       | Null | Key | Default | Extra |
+------------------------+------------+------+-----+---------+-------+
| id                     | int(50)    | NO   | PRI | NULL    |       |
| szolgaltatas_tipusok_id | int(50)   | NO   | MUL | NULL    |       |
| nev                    | char(100)  | NO   |     | NULL    |       |
| leiras                 | char(255)  | YES  |     | NULL    |       |
| aktiv                  | tinyint(1) | YES  |     | 1       |       |
+------------------------+------------+------+-----+---------+-------+
5 rows in set (0.012 sec)
```

Idopontok

```
MariaDB [idopontfoglalo]> CREATE TABLE idopontok (
    ->   id INT(50) PRIMARY KEY,
    ->   szolgaltatok_id INT(255) NOT NULL,
    ->   datum DATE NOT NULL,
    ->   ido TIME NOT NULL,
    ->   foglalhato BOOLEAN DEFAULT TRUE,
    ->   FOREIGN KEY (szolgaltatok_id) REFERENCES szolgaltatok(id)
    -> );
Query OK, 0 rows affected (0.020 sec)

MariaDB [idopontfoglalo]>
MariaDB [idopontfoglalo]> describe idopotok;
ERROR 1146 (42S02): Table 'idopontfoglalo.idopotok' doesn't exist
MariaDB [idopontfoglalo]> describe idopontok;
+-----------------+------------+------+-----+---------+-------+
| Field           | Type       | Null | Key | Default | Extra |
+-----------------+------------+------+-----+---------+-------+
| id              | int(50)    | NO   | PRI | NULL    |       |
| szolgaltatok_id | int(255)   | NO   | MUL | NULL    |       |
| datum           | date       | NO   |     | NULL    |       |
| ido             | time       | NO   |     | NULL    |       |
| foglalhato      | tinyint(1) | YES  |     | 1       |       |
+-----------------+------------+------+-----+---------+-------+
5 rows in set (0.015 sec)
```

Foglalasok

```
MariaDB [idopontfoglalo]> CREATE TABLE foglalasok (
    ->   id INT(50) PRIMARY KEY,
    ->   felhasznalok_id INT(10) NOT NULL,
    ->   idopontok_id INT(10) NOT NULL UNIQUE,
    ->   foglalasi_ido DATETIME DEFAULT CURRENT_TIMESTAMP,
    ->   allapot CHAR(50) NOT NULL,
    ->   megjegyzes CHAR(50),
    ->   FOREIGN KEY (felhasznalok_id) REFERENCES felhasznalok(id),
    ->   FOREIGN KEY (idopontok_id) REFERENCES idopontok(id)
    -> );
Query OK, 0 rows affected (0.024 sec)

MariaDB [idopontfoglalo]> describe foglalasok;
+-----------------+----------+------+-----+---------------------+-------+
| Field           | Type     | Null | Key | Default             | Extra |
+-----------------+----------+------+-----+---------------------+-------+
| id              | int(50)  | NO   | PRI | NULL                |       |
| felhasznalok_id | int(10)  | NO   | MUL | NULL                |       |
| idopontok_id    | int(10)  | NO   | UNI | NULL                |       |
| foglalasi_ido   | datetime | YES  |     | current_timestamp() |       |
| allapot         | char(50) | NO   |     | NULL                |       |
| megjegyzes      | char(50) | YES  |     | NULL                |       |
+-----------------+----------+------+-----+---------------------+-------+
6 rows in set (0.014 sec)
```

Szolgaltatas_tipusok

```
MariaDB [idopontfoglalo]> CREATE TABLE szolgaltatas_tipusok (
    ->   id INT(50) PRIMARY KEY,
    ->   nev CHAR(100) NOT NULL,
    ->   leiras CHAR(100)
    -> );
Query OK, 0 rows affected (0.006 sec)

MariaDB [idopontfoglalo]>
MariaDB [idopontfoglalo]> describe szolgaltatas_tipusok;
+--------+-----------+------+-----+---------+-------+
| Field  | Type      | Null | Key | Default | Extra |
+--------+-----------+------+-----+---------+-------+
| id     | int(50)   | NO   | PRI | NULL    |       |
| nev    | char(100) | NO   |     | NULL    |       |
| leiras | char(100) | YES  |     | NULL    |       |
+--------+-----------+------+-----+---------+-------+
3 rows in set (0.014 sec)
```

Ertekelesek

```
MariaDB [idopontfoglalo]> CREATE TABLE ertekelesek (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     foglalasok_id INT NOT NULL,
    ->     ertekeles TINYINT NOT NULL CHECK (ertekeles BETWEEN 1 AND 5),
    ->     velemeny TEXT,
    ->     datum DATE DEFAULT CURRENT_DATE,
    ->     FOREIGN KEY (foglalasok_id) REFERENCES foglalasok(id) ON DELETE CASCADE
    -> );
Query OK, 0 rows affected (0.021 sec)
```

Táblák feltöltése

*Tesztadatok **a Backend – Adatbázis kapcsolat** kommunikációjának **ellenőrzéséhez.***

```
MariaDB [idopontfoglalo]> INSERT INTO felhasznalok (id, nev, szerepkor, email, jelszo) VALUES
    -> (1, 'Kiss Gábor', 'felhasznalo', 'gabor@example.com', 'pass1'),
    -> (2, 'Nagy Anna', 'admin', 'anna@example.com', 'admin123');
Query OK, 2 rows affected (0.003 sec)
Records: 2  Duplicates: 0  Warnings: 0

MariaDB [idopontfoglalo]> SELECT * FROM Felhasznalok;
+----+------------+------------+------------------+----------+---------------------+
| id | nev        | szerepkor  | email            | jelszo   | reg_datum           |
+----+------------+------------+------------------+----------+---------------------+
|  1 | Kiss Gábor | felhasznalo | gabor@example.com | pass1    | 2025-05-30 14:08:41 |
|  2 | Nagy Anna  | admin      | anna@example.com  | admin123 | 2025-05-30 14:08:41 |
+----+------------+------------+------------------+----------+---------------------+
2 rows in set (0.000 sec)

MariaDB [idopontfoglalo]> INSERT INTO szolgaltatok (id, szolgaltatas_tipusok_id, nev, leiras, aktiv) VALUES
    -> (1, 1, 'SmileDent', 'Budapesti rendelő', TRUE);
Query OK, 1 row affected (0.004 sec)

MariaDB [idopontfoglalo]> SELECT * FROM Szolgaltatok;
+----+-------------------------+----------+------------------+------+
| id | szolgaltatas_tipusok_id | nev      | leiras           | aktiv |
+----+-------------------------+----------+------------------+------+
|  1 |                       1 | SmileDent | Budapesti rendelő |    1 |
+----+-------------------------+----------+------------------+------+
1 row in set (0.000 sec)
MariaDB [idopontfoglalo]> INSERT INTO szolgaltatas_tipusok (id, nev, leiras) VALUES
    -> (1, 'Fogászat', 'Fogászati kezelések és vizsgálatok');
Query OK, 1 row affected (0.041 sec)
MariaDB [idopontfoglalo]> SELECT * FROM szolgaltatas_tipusok;
+----+-----------+--------------------------------------+
| id | nev       | leiras                               |
+----+-----------+--------------------------------------+
|  1 | Fogászat  | Fogászati kezelések és vizsgálatok   |
+----+-----------+--------------------------------------+
1 row in set (0.001 sec)
```

Kapcsolat sikeres megjelenítése

1.3. Felhasználói regisztráció API

A regisztráció során lehetőséget biztosítok arra, hogy egy új felhasználó felkerüljön az adatbázisba. Ehhez a regisztracio.php fájlt hoztam létre, amely POST metódust fogad, és ellenőrzi a bemenetet. A nev, email, jelszo és szerepkor mezők megadása kötelező, utóbbi csak felhasznalo vagy admin lehet.

A bemeneti adatok ellenőrzése után a rendszer lekérdezi, hogy az adott e-mail cím már szerepel-e az adatbázisban. Amennyiben nem, létrehozza az új felhasználót, és visszaadja az új azonosítót. A sikeres regisztráció után 201-es státuszkódot küldök vissza.

Postman segítségével leteszteltem a működést: a POST metódussal megadott JSON adat helyesen bekerült az adatbázisba, amit a MariaDB-ben is ellenőriztem.

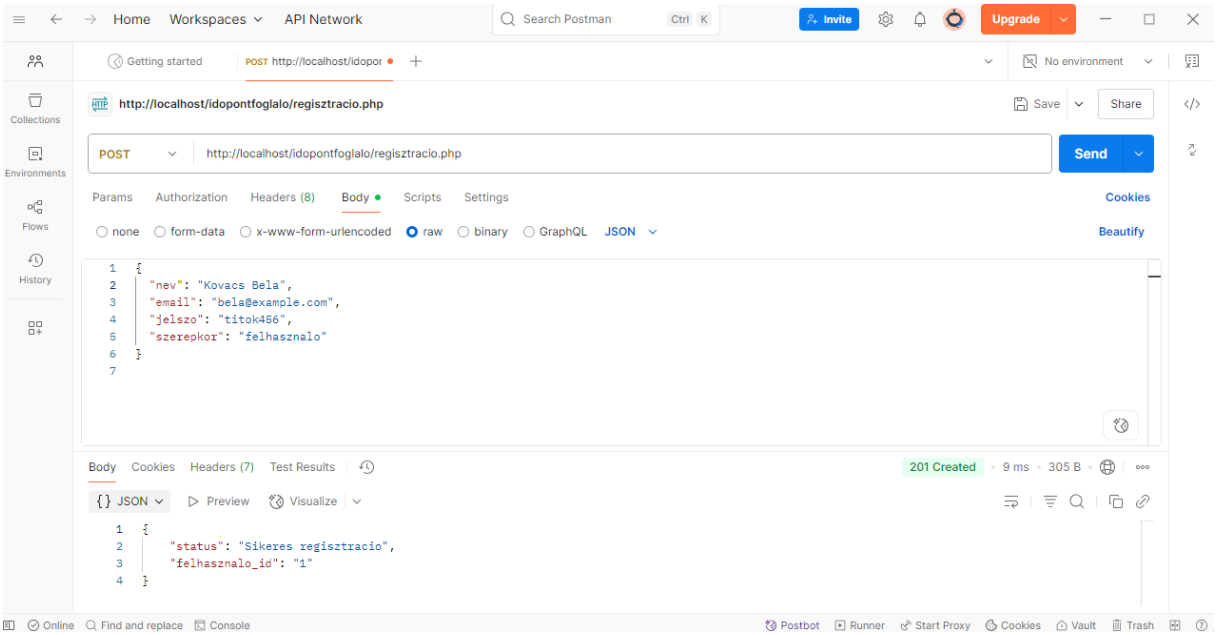
1.PHP forrásfájl

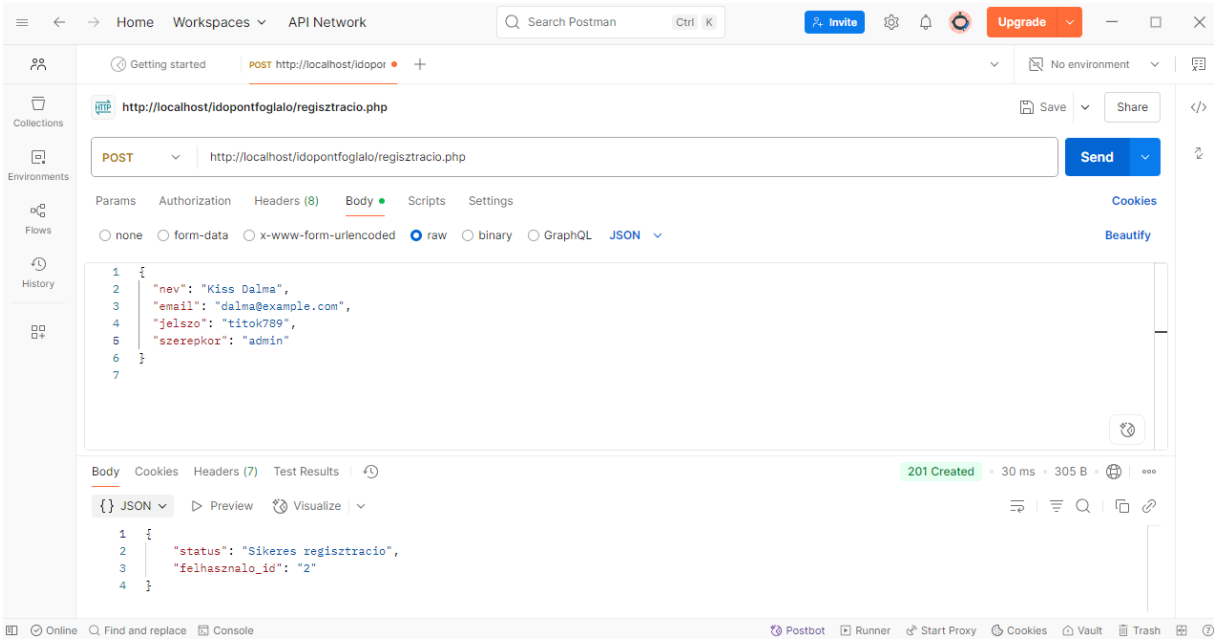
```php
<?php
// Betöltöm az adatbázis-kapcsolatot létrehozó fájlt.
require_once 'kapcsolat.php';

// Beállítom, hogy JSON-t adjak vissza.
header('Content-Type: application/json');

// Csak a POST kéréseket fogadom el.
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    http_response_code(405);
    echo json_encode(['error' => 'Csak POST metódussal érhető el ez a végpont.']);
    exit;
}
```

```
// A bejövő nyers JSON-t dekódolom tömbbé.
$input = json_decode(file_get_contents('php://input'), true);

// Ellenőrzöm, hogy minden szükséges mező létezik-e, és a szerepkor érvényes-e.
if (
    !isset($input['nev'], $input['email'], $input['jelszo'], $input['szerepkor'])
||
    !in_array($input['szerepkor'], ['felhasznalo', 'admin'], true)
) {
    http_response_code(400);
    echo json_encode(['error' => 'Nem sikerült értelmezni a JSON bemenetet, vagy
hiányzó/érvénytelen mező.']);
    exit;
}

// A bemenetből kiolvasom és megtisztítom a stringeket.
$nev = trim($input['nev']);
$email = trim($input['email']);
$jelszo = trim($input['jelszo']);
$szerepkor = $input['szerepkor'];

// Ellenőrzöm, hogy a név, email és jelszó nem üres.
if ($nev === '' || $email === '' || $jelszo === '') {
    http_response_code(400);
    echo json_encode(['error' => 'A név, email és jelszó mezők nem lehetnek
üresek.']);
    exit;
}

try {
    // Lekérdezem, hogy létezik-e már felhasználó ezzel az email címmel.
    $stmt = $pdo->prepare('SELECT COUNT(*) FROM felhasznalok WHERE email =
:email');
    $stmt->bindParam(':email', $email, PDO::PARAM_STR);
    $stmt->execute();
    if ($stmt->fetchColumn() > 0) {
        http_response_code(409);
        echo json_encode(['error' => 'Ez az email már regisztrálva van.']);
        exit;
    }
} catch (PDOException $e) {
    // Ha adatbázis-hiba történik.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba: ' . $e->getMessage()]);
    exit;
}
```

```
try {
    // Beszúrom az új felhasználót az adatbázisba.
    $stmt = $pdo->prepare('INSERT INTO felhasznalok (nev, email, jelszo,
szerepkor) VALUES (:nev, :email, :jelszo, :szerepkor)');
    $stmt->bindParam(':nev', $nev, PDO::PARAM_STR);
    $stmt->bindParam(':email', $email, PDO::PARAM_STR);
    $stmt->bindParam(':jelszo', $jelszo, PDO::PARAM_STR);
    $stmt->bindParam(':szerepkor', $szerepkor, PDO::PARAM_STR);
    $stmt->execute();

    // Lekérem az újonnan beszúrt felhasználó azonosítóját.
    $newId = $pdo->lastInsertId();
    http_response_code(201); // 201 Created
    echo json_encode([
        'status' => 'Sikeres regisztracio',
        'felhasznalo_id' => $newId
    ]);
    exit;
} catch (PDOException $e) {
    // Ha ismét adatbázis-hiba történik.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba: ' . $e->getMessage()]);
    exit;
}
```

2. API tesztelése Postman segítségével





3. Ellenőrzés MariaDB-ben.

```
MariaDB [idopontfoglalo]> SELECT * FROM Felhasznalok;
+----+-------------+-------------+-------------------+----------+---------------------+
| id | nev         | szerepkor   | email             | jelszo   | reg_datum           |
+----+-------------+-------------+-------------------+----------+---------------------+
|  1 | Kovacs Bela | felhasznalo | bela@example.com  | titok456 | 2025-06-05 14:54:59 |
|  2 | Kiss Dalma  | admin       | dalma@example.com | titok789 | 2025-06-05 14:56:09 |
+----+-------------+-------------+-------------------+----------+---------------------+
2 rows in set (0.000 sec)
```

1.4. Bejelentkezés API

A bejelentkezes.php fájlt POST metódusra terveztem. A felhasználónak meg kell adnia az email címét és jelszavát. A rendszer ellenőrzi, hogy létezik-e ilyen felhasználó, és ha az e-mailhez tartozó jelszó egyezik, akkor sikeres bejelentkezést jelez vissza.

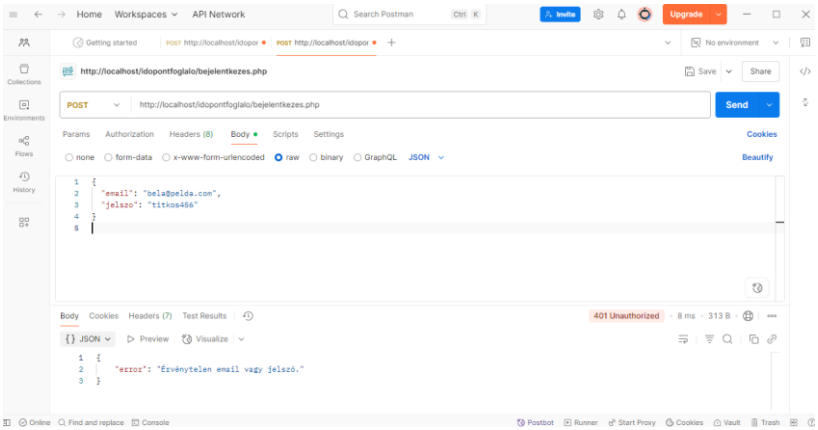
A válasz tartalmazza a felhasználó nevét, e-mail címét, azonosítóját és szerepkörét. Az API-t szintén Postman segítségével teszteltem.

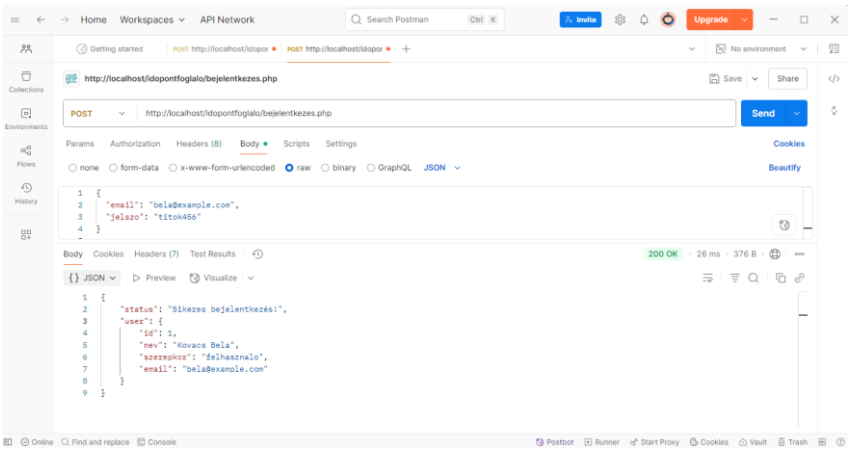
1.PHP forrásfájl

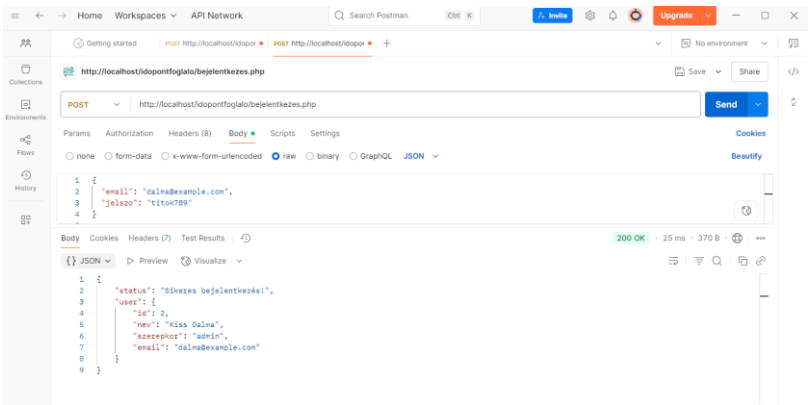
```php
<?php
// Betöltöm a kapcsolatot létrehozó fájlt, hogy használhassam a $pdo változót.
require_once "kapcsolat.php";

// A válasz tartalmát JSON formátumra állítom.
header('Content-Type: application/json');

// Csak a POST kéréseket fogadom el.
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    http_response_code(405);
    echo json_encode(['error' => 'Csak POST metódussal érhető el ez a végpont.']);
```

```
    exit;
}

// A bejövő nyers JSON-t dekódolom asszociatív tömbbé.
$input = json_decode(file_get_contents('php://input'), true);

// Ellenőrzöm, hogy sikerült-e dekódolni és valóban tömb-e.
if (!is_array($input)) {
    http_response_code(400);
    echo json_encode(['error' => 'Nem sikerült értelmezni a JSON bemenetet.']);
    exit;
}

// Ellenőrzöm, hogy a kötelező mezők (email, jelszo) megvannak-e, és nem üresek-e.
if (empty($input['email']) || empty($input['jelszo'])) {
    http_response_code(400);
    echo json_encode(['error' => 'Hiányzó email vagy jelszó.']);
    exit;
}

// Kiolvasom és elmented a változókba a bejövő emailt és jelszót.
$email = $input['email'];
$jelszo = $input['jelszo'];

// Lekérdezem az adatbázisból a felhasználót az email alapján.
$stmt = $pdo->prepare('SELECT id, nev, szerepkor, jelszo FROM felhasznalok WHERE
email = :email');
$stmt->execute([':email' => $email]);
$user = $stmt->fetch();


if (!$user || $user['jelszo'] !== $jelszo) {
    http_response_code(401);
    echo json_encode(['error' => 'Érvénytelen email vagy jelszó.']);
    exit;
}

// Ha minden rendben van, visszaküldöm a felhasználó adatait JSON-ban.
echo json_encode([
    'status'  => 'Sikeres bejelentkezés!',
    'user'    => [
        'id'        => (int)$user['id'],
        'nev'       => $user['nev'],
        'szerepkor' => $user['szerepkor'],
        'email'     => $email
    ]
]);
```

2. API tesztelése Postman segítségével







1.5. Szolgáltatók listázása API

Első lépésként feltöltöttem a szolgáltatás típusok táblát néhány alapértelmezett értékkel (pl. Fogászat, Szépségápolás stb.), majd ezekhez kapcsolódóan rögzítettem a szolgáltatók adatait is a szolgaltatok táblába. Miután az adatbázis feltöltése megtörtént, elkészítettem a szolgaltatok_listazasa.php API végpontot, amely GET kérésre visszaadja a szolgáltatók listáját, a hozzájuk tartozó típusnévvel együtt. Ezzel ellenőrizhetem, hogy az adatok helyesen tárolódnak és kapcsolódnak egymáshoz. A tesztelést Postman segítségével végeztem el. Itt láttam, hogy a status, count és data mezők szerepelnek a válaszban, tehát a funkció megfelelően működik.

1. lépés: A „Szolgaltatas_tipusok” tábla feltöltése.

```
MariaDB [idopontfoglalo]> INSERT INTO szolgaltatas_tipusok (nev, leiras) VALUES
    ->   ('Fogászat',      'Fogászati kezelések és vizsgálatok'),
    ->   ('Szépségápolás',  'Arckezelések, manikűr/pedikűr szolgáltatások'),
    ->   ('Fitness',        'Személyi edzés és csoportos alakformáló órák'),
    ->   ('Fodrászat',      'Női és férfi hajvágás, hajfestés, styling szolgáltatások'),
    ->   ('Elektronika',    'Mobiltelefon- és számítógép-javítás, szoftveres beállítások'),
    ->   ('Egészségügy',    'Kórházi kezelések és szakrendelések');
Query OK, 6 rows affected (0.004 sec)
Records: 6  Duplicates: 0  Warnings: 0

MariaDB [idopontfoglalo]> SET FOREIGN_KEY_CHECKS = 1;
Query OK, 0 rows affected (0.001 sec)

MariaDB [idopontfoglalo]> SELECT * FROM Szolgaltatas_tipusok;
+----+--------------+-------------------------------------------------------------+
| id | nev          | leiras                                                      |
+----+--------------+-------------------------------------------------------------+
|  1 | Fogászat     | Fogászati kezelések és vizsgálatok                          |
|  2 | Szépségápolás | Arckezelések, manikűr/pedikűr szolgáltatások               |
|  3 | Fitness      | Személyi edzés és csoportos alakformáló órák                |
|  4 | Fodrászat    | Női és férfi hajvágás, hajfestés, styling szolgáltatások    |
|  5 | Elektronika  | Mobiltelefon- és számítógép-javítás, szoftveres beállítások |
|  6 | Egészségügy  | Kórházi kezelések és szakrendelések                         |
+----+--------------+-------------------------------------------------------------+
6 rows in set (0.001 sec)
```

```
MariaDB [idopontfoglalo]> INSERT INTO szolgaltatas_tipusok (nev, leiras) VALUES
    ->   ('Autóápolás', 'Külső-belső tisztítás, polírozás, waxolás és motorápolás');
Query OK, 1 row affected (0.004 sec)

MariaDB [idopontfoglalo]> SELECT * FROM Szolgaltatas_tipusok;
+----+---------------+-------------------------------------------------------------------+
| id | nev           | leiras                                                            |
+----+---------------+-------------------------------------------------------------------+
|  1 | Fogászat      | Fogászati kezelések és vizsgálatok                                |
|  2 | Szépségápolás | Arckezelések, manikűr/pedikűr szolgáltatások                      |
|  3 | Fitness       | Személyi edzés és csoportos alakformáló órák                      |
|  4 | Fodrászat     | Női és férfi hajvágás, hajfestés, styling szolgáltatások          |
|  5 | Elektronika   | Mobiltelefon- és számítógép-javítás, szoftveres beállítások       |
|  6 | Egészségügy   | Kórházi kezelések és szakrendelések                               |
|  7 | Autóápolás    | Külső-belső tisztítás, polírozás, waxolás és motorápolás          |
|  8 | Autóápolás    | Külső-belső tisztítás, polírozás, waxolás és motorápolás          |
+----+---------------+-------------------------------------------------------------------+
8 rows in set (0.001 sec)
```

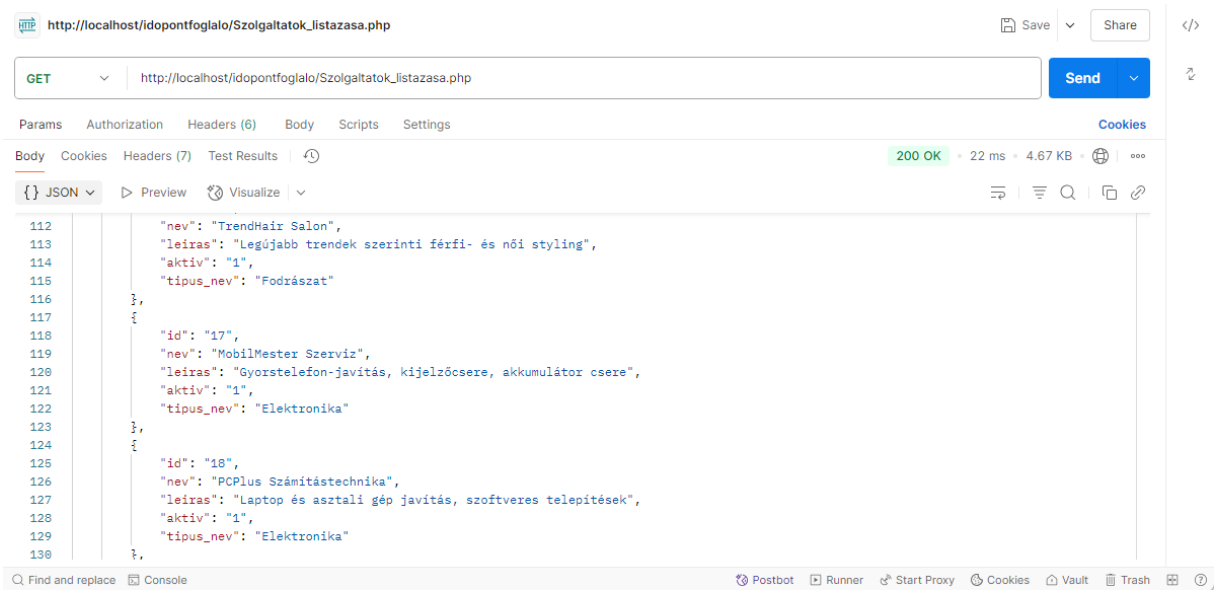
2. lépés: A „Szolgaltatok” tábla feltöltése.

```
MariaDB [idopontfoglalo]> INSERT INTO szolgaltatok (szolgaltatas_tipusok_id, nev, leiras, aktiv) VALUES
    ->   (1, 'SmileDent Rendelő',    'Teljeskörű fogászati ellátás Budapesten',                TRUE),
    ->   (1, 'EuroFogász Klinika',   'Modern implantológia és esztétikai fogászat',            TRUE),
    ->   (1, 'DentalArt Kft.',       'Gyermekfogászat és fogszabályozás profi csapattal',      TRUE),
    ->   (1, 'CityFogászat Centrum', 'Fogkő-eltávolítás, fogfehérítés gyors időpontfoglalással', TRUE),
    ->   (2, 'Beauty&You Szalon',    'Arckezelés és professzionális smink készítés',           TRUE),
    ->   (2, 'Nails&Hairs Stúdió',   'Manikűr, pedikűr és balayage hajfestés egy helyen',      TRUE),
    ->   (2, 'Elegance Spa',         'Arc- és testkezelések luxus-környezetben',               TRUE),
    ->   (2, 'PerfectBeauty Kft.',    'Kozmetikai tanácsadás, sminkoktatás, SPA csomagok',      TRUE),
    ->   (3, 'ProGym Edzőterem',     'Személyi edzés, crossfit és súlyzós edzés profi edzőkkel', TRUE),
    ->   (3, 'FitLife Csoportos',    'Csoportos alakformáló órák: zumba, pilates, jóga',       TRUE),
    ->   (3, 'IronClub Terem',       'Erőemelő szekció, funkcionális tréning és erőnléti programok', TRUE),
    ->   (3, 'Cardio4U Fitness',     'Kardiógépek, aerob és spinning órák, táplálkozási tanácsadás', TRUE),
    ->   (4, 'HairStyle Műhely',     'Férfi- és női hajvágás, profi vágók és fodrászok',       TRUE),
    ->   (4, 'ColorMaster Stúdió',   'Különleges hajfestés, balayage és tonális festések',     TRUE),
    ->   (4, 'TrendHair Salon',      'Legújabb trendek szerinti férfi- és női styling',        TRUE),
    ->   (5, 'MobilMester Szerviz',  'Gyorstelefon-javítás, kijelzőcsere, akkumulátor csere',  TRUE),
    ->   (5, 'PCPlus Számítástechnika', 'Laptop és asztali gép javítás, szoftveres telepítések', TRUE),
    ->   (5, 'GépDoktor Kft.',       'Garanciális és garancián túli javítások, szoftverlendítés', TRUE),
    ->   (6, 'Orvosi Centrum',       'Belgyógyászati és kardiológiai szakrendelések',          TRUE),
    ->   (6, 'Sürgősségi Klinikák',  'Non-stop ügyelet és sürgősségi ellátás',                 TRUE);
Query OK, 20 rows affected (0.011 sec)
Records: 20  Duplicates: 0  Warnings: 0
```

```
MariaDB [idopontfoglalo]> INSERT INTO szolgaltatok (szolgaltatas_tipusok_id, nev, leiras, aktiv) VALUES
    ->   (8, 'Expressz Autómosás',      'Gyors külső mosás, kézi szárítás, viaszvédés nélkül',             1),
    ->   (8, 'Full Service Autóápolás', 'Külső-belső takarítás, kárpittisztítás, motor tisztítás',        1),
    ->   (8, 'Nanopolírozás',           'Nanotechnológiás polírozás, felületvédelem és UV-védelem',       1),
    ->   (8, 'Motor- és Alvázmosás',    'Nagynyomású mosás motor- és alvázterületre, rozsda ellen védve', 1);
Query OK, 4 rows affected (0.003 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

```
MariaDB [idopontfoglalo]> SELECT * FROM Szolgaltatok;
+----+-------------------------+-----------------------+-----------------------------------------------------------------+--------+
| id | szolgaltatas_tipusok_id | nev                   | leiras                                                          | aktiv  |
+----+-------------------------+-----------------------+-----------------------------------------------------------------+--------+
|  1 |                       1 | SmileDent             | Budapesti rendelő                                               |      1 |
|  2 |                       1 | SmileDent Rendelő     | Teljeskörű fogászati ellátás Budapesten                         |      1 |
|  3 |                       1 | EuroFogász Klinika    | Modern implantológia és esztétikai fogászat                     |      1 |
|  4 |                       1 | DentalArt Kft.        | Gyermekfogászat és fogszabályozás profi csapattal               |      1 |
|  5 |                       1 | CityFogászat Centrum  | Fogkő-eltávolítás, fogfehérítés gyors időpontfoglalással        |      1 |
|  6 |                       2 | Beauty&You Szalon     | Arckezelés és professzionális smink készítés                    |      1 |
|  7 |                       2 | Nails&Hairs Stúdió    | Manikűr, pedikűr és balayage hajfestés egy helyen               |      1 |
|  8 |                       2 | Elegance Spa          | Arc- és testkezelések luxus-környezetben                        |      1 |
|  9 |                       2 | PerfectBeauty Kft.    | Kozmetikai tanácsadás, sminkoktatás, SPA csomagok               |      1 |
| 10 |                       3 | ProGym Edzőterem      | Személyi edzés, crossfit és súlyzós edzés profi edzőkkel        |      1 |
| 11 |                       3 | FitLife Csoportos     | Csoportos alakformáló órák: zumba, pilates, jóga                |      1 |
| 12 |                       3 | IronClub Terem        | Erőemelő szekció, funkcionális tréning és erőnléti programok    |      1 |
| 13 |                       3 | Cardio4U Fitness      | Kardiógépek, aerob és spinning órák, táplálkozási tanácsadás    |      1 |
| 14 |                       4 | HairStyle Műhely      | Férfi- és női hajvágás, profi vágók és fodrászok                |      1 |
| 15 |                       4 | ColorMaster Stúdió    | Különleges hajfestés, balayage és tonális festések              |      1 |
| 16 |                       4 | TrendHair Salon       | Legújabb trendek szerinti férfi- és női styling                 |      1 |
| 17 |                       5 | MobilMester Szerviz   | Gyorstelefon-javítás, kijelzőcsere, akkumulátor csere           |      1 |
| 18 |                       5 | PCPlus Számítástechnika | Laptop és asztali gép javítás, szoftveres telepítések         |      1 |
| 19 |                       5 | GépDoktor Kft.        | Garanciális és garancián túli javítások, szoftverlendítés       |      1 |
| 20 |                       6 | Orvosi Centrum        | Belgyógyászati és kardiológiai szakrendelések                   |      1 |
| 21 |                       6 | Sürgősségi Klinikák   | Non-stop ügyelet és sürgősségi ellátás                          |      1 |
| 26 |                       8 | Expressz Autómosás    | Gyors külső mosás, kézi szárítás, viaszvédés nélkül             |      1 |
| 27 |                       8 | Full Service Autóápolás | Külső-belső takarítás, kárpittisztítás, motor tisztítás       |      1 |
| 28 |                       8 | Nanopolírozás         | Nanotechnológiás polírozás, felületvédelem és UV-védelem        |      1 |
| 29 |                       8 | Motor- és Alvázmosás  | Nagynyomású mosás motor- és alvázterületre, rozsda ellen védve  |      1 |
+----+-------------------------+-----------------------+-----------------------------------------------------------------+--------+
25 rows in set (0.003 sec)
```

3. lépés: A „Szolgaltatok_listazasa.php” fájl tesztelése Postman segítségével.



4. Forrásfájl (Szolgaltatok_listazasa.php)

```php
<?php
// Betöltöm a PDO-kapcsolatot kezelő fájlt, így rendelkezem a $pdo objektummal.
require_once "kapcsolat.php";

// A válasz formátumát JSON-re állítom.
header('Content-Type: application/json');
```

```
// Ellenőrzöm, hogy a kérés csak GET metódussal érhető el.
// Ha más metódus érkezik, hibát (405) küldök, majd kilépek.
if ($_SERVER['REQUEST_METHOD'] !== 'GET') {
    http_response_code(405);
    echo json_encode(['error' => 'Csak GET metódussal érhető el ez a végpont.']);
    exit;
}

try {
    // Lekérdezem az összes szolgáltatót, valamint a hozzájuk tartozó típus nevét.
    $stmt = $pdo->query("
        SELECT
            s.id,
            s.nev,
            s.leiras,
            s.aktiv,
            t.nev AS tipus_nev
        FROM szolgaltatok AS s
        LEFT JOIN szolgaltatas_tipusok AS t
            ON s.szolgaltatas_tipusok_id = t.id
        ORDER BY s.id
    ");
    // A lekérdezés eredményét tömbbé alakítom.
    $szolgaltatok = $stmt->fetchAll();

    // JSON-ként visszaküldöm a státuszt, a rekordok számát és az adatokat.
    echo json_encode([
        'status' => 'siker',
        'count'  => count($szolgaltatok),
        'data'   => $szolgaltatok
    ]);
} catch (PDOException $e) {
    // Ha adatbázis-hiba történik, 500-as hibát küldök vissza a részletes
üzenettel.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba: ' . $e->getMessage()]);
    exit;
}
```

1.6. Szolgáltatók kezelése API

Ebben a feladatrészben elkészítettem a szolgáltatók létrehozására, módosítására és törlésére szolgáló API-kat. A POST, PUT és DELETE metódusokat külön-külön fájlban valósítottam meg. Fontos volt, hogy minden esetben megtörténjen az ellenőrzés is (például hogy a szolgaltatas_tipusok_id valóban létezik-e a kapcsolódó táblában), és hogy a kliens visszajelzést kapjon a művelet sikerességéről vagy hibáiról.

A Postman-nel végzett tesztelés során minden végpontra külön ellenőrzést végeztem: az új rekord hozzáadása után lekérdeztem a listát, frissítésnél módosított értékeket kértem vissza, törlés után pedig ellenőriztem, hogy valóban eltűnt-e az adott szolgáltató.

Minden teszt sikeresen lefutott, az API-k működnek, és hibakezeléssel is rendelkeznek.

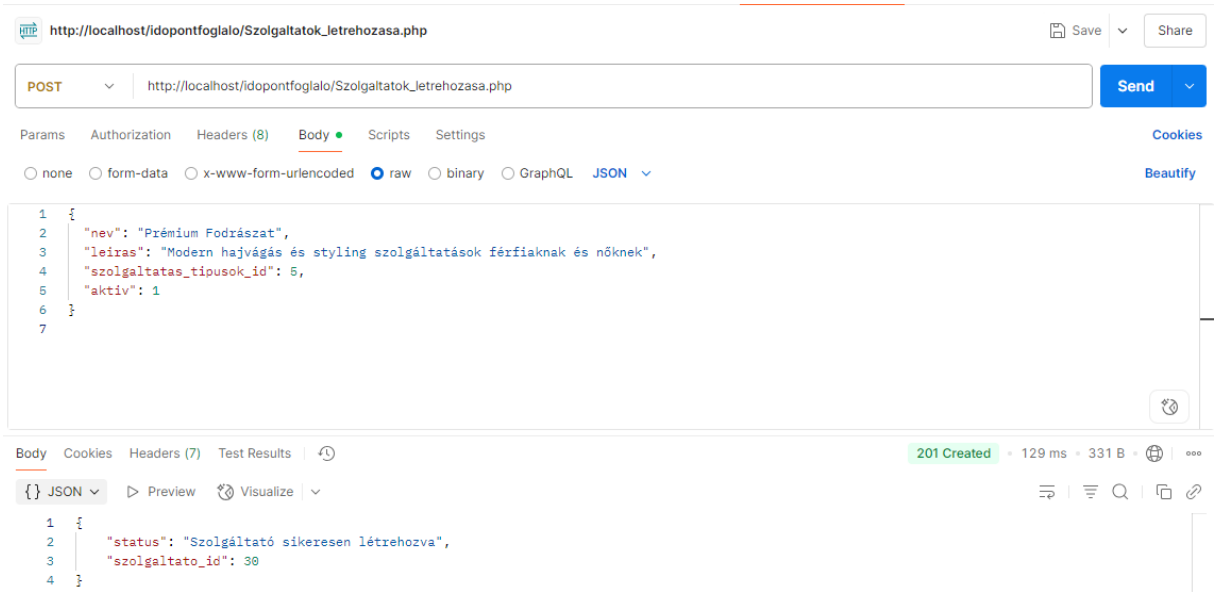
I. Szolgáltatók létrehozása:
1. Forrásfájl.

```
<?php
// Betöltöm a PDO-kapcsolatot létrehozó fájlt, hogy legyen $pdo objektumom.
require_once "kapcsolat.php";

// Beállítom, hogy a válasz mindig JSON formátumú legyen.
header('Content-Type: application/json');

// Ellenőrzöm, hogy a kliens csak POST metódussal érhesse el ezt a végpontot.
// Ha más metódus érkezik, 405-ös hibával kilépek.
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    http_response_code(405);
    echo json_encode(['error' => 'Csak POST metódussal érhető el ez a végpont.']);
    exit;
}

// Beolvasom a bejövő JSON-t, és asszociatív tömbbé alakítom.
$input = json_decode(file_get_contents('php://input'), true);

// Ellenőrzöm, hogy a JSON egy tömb legyen, és tartalmazza a 'nev', 'leiras',
'szolgaltatas_tipusok_id' kulcsokat.
// Az 'aktiv' mezőnek pedig vagy 0-nak, vagy 1-nek kell lennie.
if (
    !is_array($input) ||
    !isset($input['nev'], $input['leiras'], $input['szolgaltatas_tipusok_id']) ||
    (!isset($input['aktiv']) && $input['aktiv'] !== 0 && $input['aktiv'] !== 1)
) {
    http_response_code(400);
    echo json_encode(['error' => 'Hiányzó vagy érvénytelen mezők.']);
    exit;
}

// Kitisztítom és változókba mentem a bemenetet.
```

```
$nev                    = trim($input['nev']);
$leiras                 = trim($input['leiras']);
$szolgaltatas_tipusok_id = (int)$input['szolgaltatas_tipusok_id'];
$aktiv                  = (int)$input['aktiv'];

// Ellenőrzöm, hogy a név, leírás nem üres, és a típus-id pozitív szám-e.
if ($nev === '' || $leiras === '' || $szolgaltatas_tipusok_id <= 0) {
    http_response_code(400);
    echo json_encode(['error' => 'A név, leírás és szolgáltatástípus azonosító nem
lehet üres.']);
    exit;
}

// Megnézem, hogy valóban létezik-e a megadott szolgáltatás-típus a táblában.
// Ha nem, visszadobom a 400-as hibát.
try {
    $stmtFK = $pdo->prepare('SELECT COUNT(*) FROM szolgaltatas_tipusok WHERE id =
:tid');
    $stmtFK->bindParam(':tid', $szolgaltatas_tipusok_id, PDO::PARAM_INT);
    $stmtFK->execute();
    if ($stmtFK->fetchColumn() == 0) {
        http_response_code(400);
        echo json_encode(['error' => 'A megadott szolgáltatás-típus nem
létezik.']);
        exit;
    }
} catch (PDOException $e) {
    // Ha a lekérdezés közben hiba történik, 500-as hibát küldök a részletes
üzenettel.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba (típus ellenőrzés): ' . $e-
>getMessage()]);
    exit;
}

// Most beszúrom az új szolgáltatót az adatbázisba.
try {
    $stmt = $pdo->prepare('
        INSERT INTO szolgaltatok (szolgaltatas_tipusok_id, nev, leiras, aktiv)
        VALUES (:tid, :nev, :leiras, :aktiv)
    ');
    $stmt->bindParam(':tid',   $szolgaltatas_tipusok_id, PDO::PARAM_INT);
    $stmt->bindParam(':nev',   $nev,                     PDO::PARAM_STR);
    $stmt->bindParam(':leiras',$leiras,                  PDO::PARAM_STR);
    $stmt->bindParam(':aktiv', $aktiv,                   PDO::PARAM_INT);
    $stmt->execute();
```

```
    // Sikeres beszúrás esetén kiolvasom az új rekord AUTO_INCREMENT értékét.
    $newId = $pdo->lastInsertId();
    http_response_code(201);
    echo json_encode([
        'status'         => 'Szolgáltató sikeresen létrehozva',
        'szolgaltato_id' => (int)$newId
    ]);
    exit;
} catch (PDOException $e) {
    // Ha az INSERT során adatbázis-hiba történik, 500-as hibakódot küldök vissza.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba (szolgáltató létrehozása): ' .
$e->getMessage()]);
    exit;
}
```

2. Postman tesztelés



3. MariaDB ellenőrzés.

```
MariaDB [idopontfoglalo]> SELECT * FROM Szolgaltatok;
+----+-------------------------+-------------------------+-----------------------------------------------------------------------+-------+
| id | szolgaltatas_tipusok_id | nev                     | leiras                                                                | aktiv |
+----+-------------------------+-------------------------+-----------------------------------------------------------------------+-------+
|  1 |                       1 | SmileDent               | Budapesti rendelő                                                     |     1 |
|  2 |                       1 | SmileDent Rendelő       | Teljeskörű fogászati ellátás Budapesten                               |     1 |
|  3 |                       1 | EuroFogász Klinika      | Modern implantológia és esztétikai fogászat                           |     1 |
|  4 |                       1 | DentalArt Kft.          | Gyermekfogászat és fogszabályozás profi csapattal                     |     1 |
|  5 |                       1 | CityFogászat Centrum    | Fogkő-eltávolítás, fogfehérítés gyors időpontfoglalással              |     1 |
|  6 |                       2 | Beauty&You Szalon       | Arckezelés és professzionális smink készítés                          |     1 |
|  7 |                       2 | Nails&Hairs Stúdió      | Manikűr, pedikűr és balayage hajfestés egy helyen                     |     1 |
|  8 |                       2 | Elegance Spa            | Arc- és testkezelések luxus-környezetben                              |     1 |
|  9 |                       2 | PerfectBeauty Kft.      | Kozmetikai tanácsadás, sminkoktatás, SPA csomagok                     |     1 |
| 10 |                       3 | ProGym Edzőterem        | Személyi edzés, crossfit és súlyzós edzés profi edzőkkel              |     1 |
| 11 |                       3 | FitLife Csoportos       | Csoportos alakformáló órák: zumba, pilates, jóga                      |     1 |
| 12 |                       3 | IronClub Terem          | Erőemelő szekció, funkcionális tréning és erőnléti programok          |     1 |
| 13 |                       3 | Cardio4U Fitness        | Kardiógépek, aerob és spinning órák, táplálkozási tanácsadás          |     1 |
| 14 |                       4 | HairStyle Műhely        | Férfi- és női hajvágás, profi vágók és fodrászok                      |     1 |
| 15 |                       4 | ColorMaster Stúdió      | Különleges hajfestés, balayage és tonális festések                    |     1 |
| 16 |                       4 | TrendHair Salon         | Legújabb trendek szerinti férfi- és női styling                       |     1 |
| 17 |                       5 | MobilMester Szerviz     | Gyorstelefon-javítás, kijelzőcsere, akkumulátor csere                 |     1 |
| 18 |                       5 | PCPlus Számítástechnika | Laptop és asztali gép javítás, szoftveres telepítések                 |     1 |
| 19 |                       5 | GépDoktor Kft.          | Garanciális és garancián túli javítások, szoftverlendítés             |     1 |
| 20 |                       6 | Orvosi Centrum          | Belgyógyászati és kardiológiai szakrendelések                         |     1 |
| 21 |                       6 | Sürgősségi Klinikák     | Non-stop ügyelet és sürgősségi ellátás                                |     1 |
| 26 |                       8 | Expressz Autómosás      | Gyors külső mosás, kézi szárítás, viaszvédés nélkül                   |     1 |
| 27 |                       8 | Full Service Autóápolás | Külső-belső takarítás, kárpittisztítás, motor tisztítás               |     1 |
| 28 |                       8 | Nanopolírozás           | Nanotechnológiás polírozás, felületvédelem és UV-védelem              |     1 |
| 29 |                       8 | Motor- és Alvázmosás    | Nagynyomású mosás motor- és alvázterületre, rozsda ellen védve        |     1 |
| 30 |                       5 | Prémium Fodrászat       | Modern hajvágás és styling szolgáltatások férfiaknak és nőknek        |     1 |
+----+-------------------------+-------------------------+-----------------------------------------------------------------------+-------+
26 rows in set (0.003 sec)
```

I. Szolgáltatók módosítása:

1. Forrásfájl.

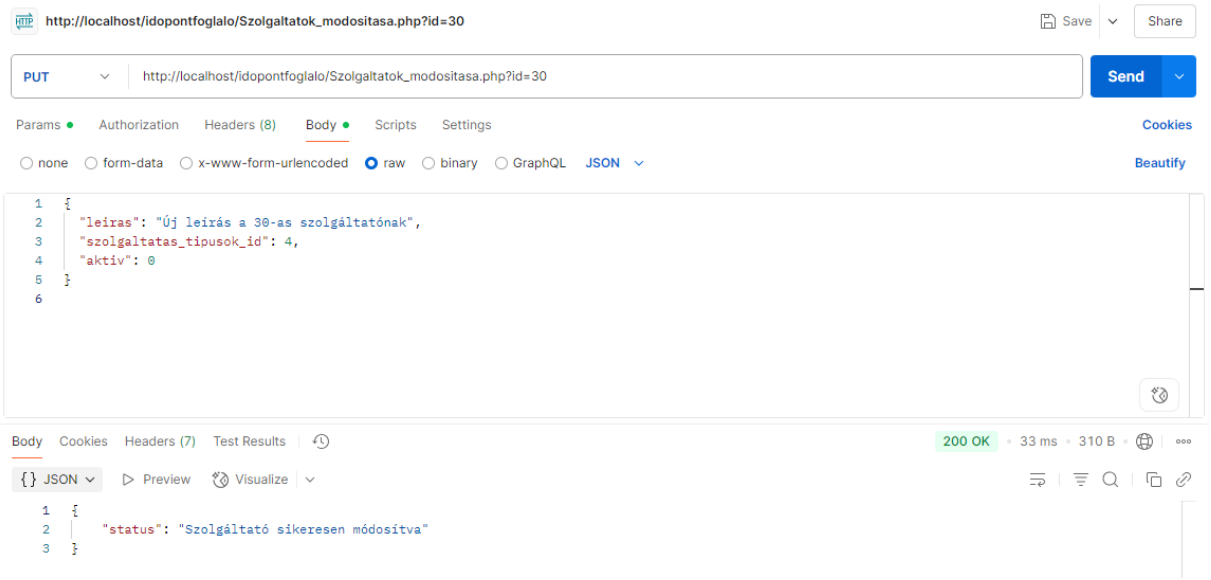
```php
<?php
// Betöltöm a kapcsolat.php fájlt, hogy a $pdo változó elérhető legyen az adatbázis-műveletekhez.
require_once "kapcsolat.php";

// Beállítom, hogy minden válasz JSON formátumban érkezzen a kliens felé.
header('Content-Type: application/json');

// Ellenőrzöm, hogy valóban PUT metódussal hívják-e meg ezt a végpontot.
if ($_SERVER['REQUEST_METHOD'] !== 'PUT') {
    // Ha nem PUT, 405-ös (Method Not Allowed) HTTP státuszkódot küldök
    http_response_code(405);
    echo json_encode(['error' => 'Csak PUT metódussal érhető el ez a végpont.']);
    exit;
}

// Ellenőrzöm, hogy szerepel-e az URL-ben az "id" paraméter, és az csak számokat tartalmaz-e.
if (!isset($_GET['id']) || !ctype_digit($_GET['id'])) {
    // Ha nincs érvényes id, 400-as (Bad Request) státuszt küldök
    http_response_code(400);
    echo json_encode(['error' => 'Hiányzó vagy érvénytelen szolgáltató-id.']);
    exit;
}
// Átalakítom az id-t integerre.
$szolgaltato_id = (int)$_GET['id'];
```

```
// A beérkező JSON-t beolvasom és asszociatív tömbbé alakítom.
$input = json_decode(file_get_contents('php://input'), true);
if (!is_array($input)) {
    // Ha nem tudtam tömbbé dekódolni a JSON-t, 400-as hibát adok vissza
    http_response_code(400);
    echo json_encode(['error' => 'Nem sikerült értelmezni a JSON bemenetet.']);
    exit;
}

try {
    // Lekérdezem, hogy létezik-e ilyen "id"-jú rekord a szolgaltatok táblában.
    $stmtCheck = $pdo->prepare('SELECT COUNT(*) FROM szolgaltatok WHERE id =
:id');
    $stmtCheck->bindParam(':id', $szolgaltato_id, PDO::PARAM_INT);
    $stmtCheck->execute();
    if ($stmtCheck->fetchColumn() == 0) {
        // Ha nem létezik, 404-es (Not Found) hibát küldök.
        http_response_code(404);
        echo json_encode(['error' => 'A megadott szolgáltató nem található.']);
        exit;
    }
} catch (PDOException $e) {
    // Ha adatbázis hiba történik a létezés-ellenőrzés során, 500-as hibát adok
vissza.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba (létezés ellenőrzés): ' . $e-
>getMessage()]);
    exit;
}

// Összeállítom, mely mezőket (oszlopokat) szeretnék frissíteni, és az ezekhez
tartozó paramétereket.
$fieldsToUpdate = [];
$params = [ ':id' => $szolgaltato_id ];

// Ha a bejövő JSON tartalmaz "nev"-et, akkor hozzáadom a frissítendő mezők közé.
if (isset($input['nev'])) {
    $fieldsToUpdate[] = 'nev = :nev';
    $params[':nev'] = trim($input['nev']);
}
// Ha tartalmaz "leiras"-t, hozzáadom.
if (isset($input['leiras'])) {
    $fieldsToUpdate[] = 'leiras = :leiras';
    $params[':leiras'] = trim($input['leiras']);
}
// Ha tartalmaz "szolgaltatas_tipusok_id"-t, hozzáadom és integer típusra cast-
```

```
olom.
if (isset($input['szolgaltatas_tipusok_id'])) {
    $fieldsToUpdate[] = 'szolgaltatas_tipusok_id = :tid';
    $params[':tid'] = (int)$input['szolgaltatas_tipusok_id'];
}
// Ha tartalmaz "aktiv"-ot, hozzáadom és integerre cast-olom.
if (isset($input['aktiv'])) {
    $fieldsToUpdate[] = 'aktiv = :aktiv';
    $params[':aktiv'] = (int)$input['aktiv'];
}

// Ha nem volt egyetlen frissítendő mező sem, 400-as hibát küldök.
if (empty($fieldsToUpdate)) {
    http_response_code(400);
    echo json_encode(['error' => 'Nincs megadva egyetlen módosítandó mező sem.']);
    exit;
}

// Ha a szolgaltatas_tipusok_id mezőt is adják, előbb ellenőrzöm, hogy az id
létezik-e a szolgaltatas_tipusok táblában.
if (isset($params[':tid'])) {
    try {
        $stmtFK = $pdo->prepare('SELECT COUNT(*) FROM szolgaltatas_tipusok WHERE
id = :tid');
        $stmtFK->bindParam(':tid', $params[':tid'], PDO::PARAM_INT);
        $stmtFK->execute();
        if ($stmtFK->fetchColumn() == 0) {
            // Ha nincs ilyen típus, 400-as hibát küldök.
            http_response_code(400);
            echo json_encode(['error' => 'A megadott (új) szolgáltatás-típus nem
létezik.']);
            exit;
        }
    } catch (PDOException $e) {
        // Ha adatbázis hiba adódik a típus ellenőrzésénél, 500-as hibát küldök.
        http_response_code(500);
        echo json_encode(['error' => 'Adatbázis hiba (típus ellenőrzés): ' . $e-
>getMessage()]);
        exit;
    }
}

// Összerakom az UPDATE SQL utasítást a kiválasztott mezőkkel.
$sql = 'UPDATE szolgaltatok SET ' . implode(', ', $fieldsToUpdate) . ' WHERE id =
:id';

try {
```

```
    // Előkészítem és végrehajtom a frissítést a paraméterekkel.
    $stmt = $pdo->prepare($sql);
    $stmt->execute($params);

    // Ha sikeres a módosítás, 200-as státuszt és egy JSON-feleletet küldök.
    http_response_code(200);
    echo json_encode(['status' => 'Szolgáltató sikeresen módosítva']);
    exit;
} catch (PDOException $e) {
    // Ha adatbázis hiba történik a frissítéskor, 500-as hibát küldök.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba (szolgáltató módosítása): ' . $e->getMessage()]);
    exit;
}
```

2. Postman tesztelés



3. MariaDB ellenőrzés.

```
MariaDB [idopontfoglalo]> SELECT * FROM Szolgaltatok;
+----+-------------------------+-----------------------+-----------------------------------------------------------------+-------+
| id | szolgaltatas_tipusok_id | nev                   | leiras                                                          | aktiv |
+----+-------------------------+-----------------------+-----------------------------------------------------------------+-------+
|  1 |                       1 | SmileDent             | Budapesti rendelő                                               |     1 |
|  2 |                       1 | SmileDent Rendelő     | Teljeskörű fogászati ellátás Budapesten                         |     1 |
|  3 |                       1 | EuroFogász Klinika    | Modern implantológia és esztétikai fogászat                     |     1 |
|  4 |                       1 | DentalArt Kft.        | Gyermekfogászat és fogszabályozás profi csapattal               |     1 |
|  5 |                       1 | CityFogászat Centrum  | Fogkő-eltávolítás, fogfehérítés gyors időpontfoglalással        |     1 |
|  6 |                       2 | Beauty&You Szalon     | Arckezelés és professzionális sminkkészítés                     |     1 |
|  7 |                       2 | Nails&Hairs Stúdió    | Manikűr, pedikűr és balayage hajfestés egy helyen               |     1 |
|  8 |                       2 | Elegance Spa          | Arc- és testkezelések luxus-környezetben                        |     1 |
|  9 |                       2 | PerfectBeauty Kft.    | Kozmetikai tanácsadás, sminkoktatás, SPA csomagok               |     1 |
| 10 |                       3 | ProGym Edzőterem      | Személyi edzés, crossfit és súlyzós edzés profi edzőkkel        |     1 |
| 11 |                       3 | FitLife Csoportos     | Csoportos alakformáló órák: zumba, pilates, jóga                |     1 |
| 12 |                       3 | IronClub Terem        | Erőemelő szekció, funkcionális tréning és erőnléti programok    |     1 |
| 13 |                       3 | Cardio4U Fitness      | Kardiógépek, aerob és spinning órák, táplálkozási tanácsadás    |     1 |
| 14 |                       4 | HairStyle Műhely      | Férfi- és női hajvágás, profi vágók és fodrászok                |     1 |
| 15 |                       4 | ColorMaster Stúdió    | Különleges hajfestés, balayage és tonális festések              |     1 |
| 16 |                       4 | TrendHair Salon       | Legújabb trendek szerinti férfi- és női styling                 |     1 |
| 17 |                       5 | MobilMester Szerviz   | Gyorstelefon-javítás, kijelzőcsere, akkumulátor csere           |     1 |
| 18 |                       5 | PCPlus Számítástechnika | Laptop és asztali gép javítás, szoftveres telepítések         |     1 |
| 19 |                       5 | GépDoktor Kft.        | Garanciális és garancián túli javítások, szoftverlendítés       |     1 |
| 20 |                       6 | Orvosi Centrum        | Belgyógyászati és kardiológiai szakrendelések                   |     1 |
| 26 |                       8 | Expressz Autómosás    | Gyors külső mosás, kézi szárítás, viaszvédés nélkül             |     1 |
| 27 |                       8 | Full Service Autóápolás | Külső-belső takarítás, kárpittisztítás, motor tisztítás       |     1 |
| 28 |                       8 | Nanopolírozás         | Nanotechnológiás polírozás, felületvédelem és UV-védelem        |     1 |
| 29 |                       8 | Motor- és Alvázmosás  | Nagynyomású mosás motor- és alvázterületre, rozsda ellen védve  |     1 |
| 30 |                       4 | Prémium Fodrászat     | Új leírás a 30-as szolgáltatónak                                |     0 |
+----+-------------------------+-----------------------+-----------------------------------------------------------------+-------+
25 rows in set (0.001 sec)
```

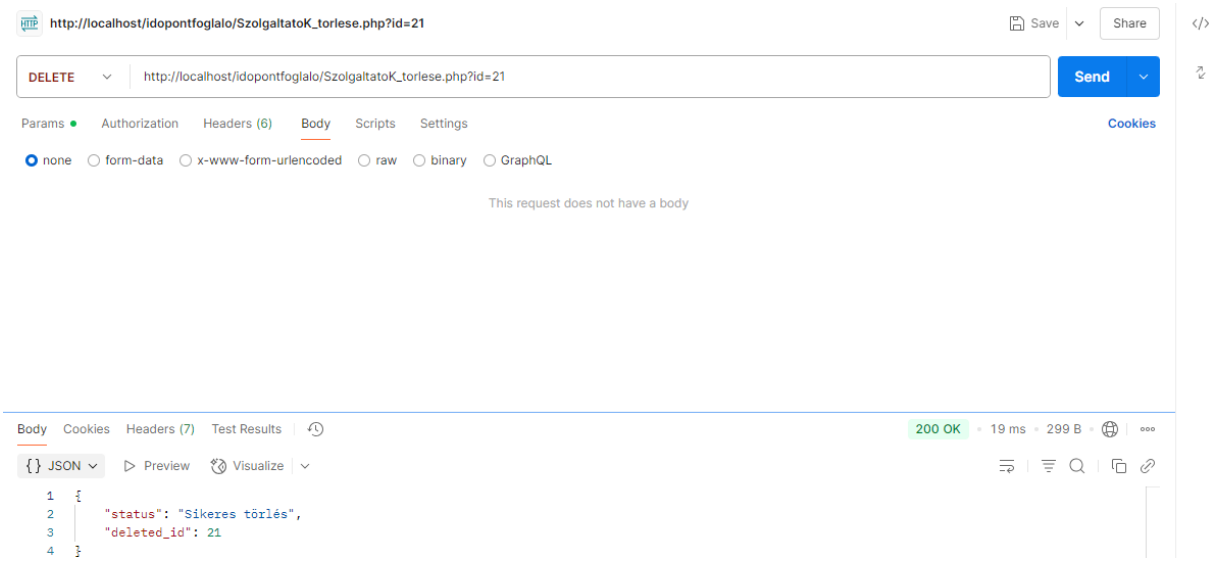
I. Szolgáltatók törlése:

1. Forrásfájl.

```php
<?php
// Betöltöm a PDO-kapcsolatot kezelő fájlt, hogy legyen $pdo objektumom.
require_once "kapcsolat.php";

// Beállítom, hogy a válasz JSON formátumban érkezzen a kliens felé.
header('Content-Type: application/json');

// Ellenőrzöm, hogy a kérés csak DELETE metódussal érhető el.
// Ha más metódust látok, 405-ös hibával kilépek.
if ($_SERVER['REQUEST_METHOD'] !== 'DELETE') {
    http_response_code(405);
    echo json_encode(['error' => 'Csak DELETE metódussal érhető el ez a
végpont.']);
    exit;
}

// Lekérem az id-t a query stringből, és ellenőrzöm, hogy számjegyekből áll-e.
// Ha hiányzik vagy nem digitális, 400-as hibát küldök.
if (empty($_GET['id']) || !ctype_digit($_GET['id'])) {
    http_response_code(400);
    echo json_encode(['error' => 'Hiányzó vagy érvénytelen id.']);
    exit;
}

$id = (int)$_GET['id']; // Ellenőrzöm, hogy egész szám az id.

// Először ellenőrzöm, létezik-e a megadott id-val szolgáltató.
// Ha nincs ilyen sor a táblában, 404-es hibát adok vissza.
```

```
try {
    $stmtChk = $pdo->prepare('SELECT COUNT(*) FROM szolgaltatok WHERE id = :id');
    $stmtChk->execute([':id' => $id]);
    if ($stmtChk->fetchColumn() == 0) {
        http_response_code(404);
        echo json_encode(['error' => 'A megadott id-val nem található
szolgáltató.']);
        exit;
    }
} catch (PDOException $e) {
    // Ha az ellenőrzés közben adatbázis-hiba történik, 500-as hibával válaszolok.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba (ellenőrzés): ' . $e-
>getMessage()]);
    exit;
}

// Ha tényleg létezik az id, végrehajtom a DELETE utasítást.
try {
    $stmt = $pdo->prepare('DELETE FROM szolgaltatok WHERE id = :id');
    $stmt->execute([':id' => $id]);

    // Sikeres törlés esetén 200-as státusszal visszaküldöm az eltávolított id-t.
    http_response_code(200);
    echo json_encode([
        'status' => 'Sikeres törlés',
        'deleted_id' => $id
    ]);
    exit;
} catch (PDOException $e) {
    // Ha a törlés közben hiba történik, 500-as hibával jelzem a részleteket.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba (delete): ' . $e->getMessage()]);
    exit;
}
```

2. Postman tesztelés



3. MariaDB ellenőrzés.

id	szolgaltatas_tipusok_id	nev	leiras	aktiv
1	1	SmileDent	Budapesti rendelő	1
2	1	SmileDent Rendelő	Teljeskörű fogászati ellátás Budapesten	1
3	1	EuroFogász Klinika	Modern implantológia és esztétikai fogászat	1
4	1	DentalArt Kft.	Gyermekfogászat és fogszabályozás profi csapattal	1
5	1	CityFogászat Centrum	Fogkő-eltávolítás, fogfehérítés gyors időpontfoglalással	1
6	2	Beauty&You Szalon	Arckezelés és professzionális smink készítés	1
7	2	Nails&Hairs Stúdió	Manikűr, pedikűr és balayage hajfestés egy helyen	1
8	2	Elegance Spa	Arc- és testkezelések luxus-környezetben	1
9	2	PerfectBeauty Kft.	Kozmetikai tanácsadás, sminkoktatás, SPA csomagok	1
10	3	ProGym Edzőterem	Személyi edzés, crossfit és súlyzós edzés profi edzőkkel	1
11	3	FitLife Csoportos	Csoportos alakformáló órák: zumba, pilates, jóga	1
12	3	IronClub Terem	Erőemelő szekció, funkcionális tréning és erőnléti programok	1
13	3	CardioHU Fitness	Kardiógépek, aerob és spinning órák, táplálkozási tanácsadás	1
14	4	HairStyle Műhely	Férfi- és női hajvágás, profi vágók és fodrászok	1
15	4	ColorMaster Stúdió	Különleges hajfestés, balayage és tonális festések	1
16	4	TrendHair Salon	Legújabb trendek szerinti férfi- és női styling	1
17	5	MobilMester Szerviz	Gyorstelefon-javítás, kijelzőcsere, akkumulátor csere	1
18	5	PCPlus Számítástechnika	Laptop és asztali gép javítás, szoftveres telepítések	1
19	5	GépDoktor Kft.	Garanciális és garancián túli javítások, szoftverlendítés	1
20	6	Orvosi Centrum	Belgyógyászati és kardiológiai szakrendelések	1
26	8	Expressz Autómosás	Gyors külső mosás, kézi szárítás, viaszvédés nélkül	1
27	8	Full Service Autóápolás	Külső-belső takarítás, kárpittisztítás, motor tisztítás	1
28	8	Nanopolírozás	Nanotechnológiás polírozás, felületvédelem és UV-védelem	1
29	8	Motor- és Alvázmosás	Nagynyomású mosás motor- és alvázterületre, rozsda ellen védve	1
30	5	Prémium Fodrászat	Modern hajvágás és styling szolgáltatások férfiaknak és nőknek	1

25 rows in set (0.001 sec)

1.7. Időpontok kezelése API

Ebben a munkafázisban az időpontok kezelésére szolgáló backend API-k elkészítése volt a feladatom. Célom az volt, hogy minden szolgáltatóhoz lehessen rögzíteni, módosítani, listázni és törölni időpontokat, valamint kezelni lehessen azok foglalhatósági állapotát is.

Első lépésként létrehoztam a GET metódust támogató végpontot, amely az összes időpontot kilistázza, szolgáltatóhoz kapcsolt névvel együtt. Ehhez csatlakoztattam az idopontok és szolgaltatok táblát, így a felhasználó a szolgáltató nevét is láthatja. A lekérdezés során rendeztem az adatokat dátum és idő szerint.

Ezután következett az új időpontok létrehozását biztosító POST API. Az inputként megadott szolgáltató ID, dátum és idő mellett lehetőség van a „foglalható” státusz megadására is. Alapértelmezetten a foglalható érték 1 (azaz igen). A rendszer validálja a mezők meglétét és helyes típusát, hibás adat esetén megfelelő hibaüzenetet küld vissza.

A PUT metódussal létrehoztam az időpontok módosítását lehetővé tevő API-t. Itt bármely mező (szolgáltató ID, dátum, idő, foglalható) frissíthető, részleges módosítás is engedélyezett. Az API megfelelően kezeli a hibákat, például ha a megadott ID nem létezik, vagy nincs módosítandó mező.

A törléshez szükséges DELETE metódus is elkészült, amely megadott ID alapján képes egy időpontot eltávolítani. A törlés előtt a rendszer ellenőrzi, hogy az adott azonosító létezik-e az adatbázisban.

Végül a fenti végpontokat egyenként leteszteltem a Postman segítségével. Az időpontok rögzítése és módosítása után a MariaDB adatbázisban manuálisan is ellenőriztem, hogy a változások rögzültek-e. Minden teszt sikeresen lefutott.

Adatbázis feltöltése tesztadatokkal.

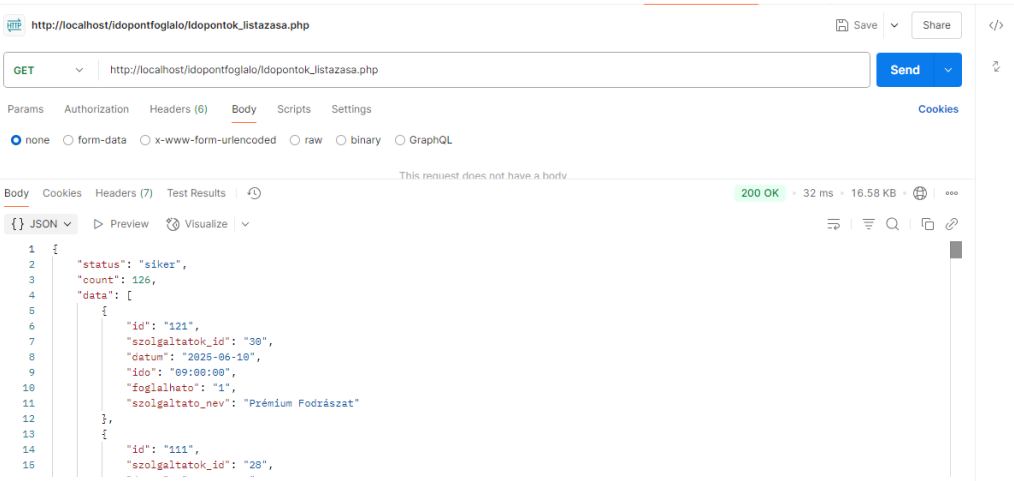
```
MariaDB [idopontfoglalo]> INSERT INTO idopontok (szolgaltatok_id, datum, ido, foglalhato)
    -> SELECT
    ->   s.id,
    ->   '2025-06-10' AS datum,
    ->   t.ido,
    ->   1 AS foglalhato
    -> FROM szolgaltatok AS s
    -> CROSS JOIN (
    ->   SELECT '09:00:00' AS ido UNION ALL
    ->   SELECT '10:00:00'        UNION ALL
    ->   SELECT '11:00:00'        UNION ALL
    ->   SELECT '12:00:00'        UNION ALL
    ->   SELECT '13:00:00'
    -> ) AS t
    -> ORDER BY s.id, t.ido;
Query OK, 125 rows affected (0.008 sec)
Records: 125  Duplicates: 0  Warnings: 0
MariaDB [idopontfoglalo]> SELECT * FROM Idopontok;
+-----+----------------+------------+----------+------------+
| id  | szolgaltatok_id | datum      | ido      | foglalhato |
+-----+----------------+------------+----------+------------+
|   1 |              1 | 2025-06-10 | 09:00:00 |          1 |
|   2 |              1 | 2025-06-10 | 10:00:00 |          1 |
|   3 |              1 | 2025-06-10 | 11:00:00 |          1 |
|   4 |              1 | 2025-06-10 | 12:00:00 |          1 |
|   5 |              1 | 2025-06-10 | 13:00:00 |          1 |
|   6 |              2 | 2025-06-10 | 09:00:00 |          1 |
|   7 |              2 | 2025-06-10 | 10:00:00 |          1 |
|   8 |              2 | 2025-06-10 | 11:00:00 |          1 |
|   9 |              2 | 2025-06-10 | 12:00:00 |          1 |
|  10 |              2 | 2025-06-10 | 13:00:00 |          1 |
|  11 |              3 | 2025-06-10 | 09:00:00 |          1 |
|  12 |              3 | 2025-06-10 | 10:00:00 |          1 |
|  13 |              3 | 2025-06-10 | 11:00:00 |          1 |
|  14 |              3 | 2025-06-10 | 12:00:00 |          1 |
|  15 |              3 | 2025-06-10 | 13:00:00 |          1 |
|  16 |              4 | 2025-06-10 | 09:00:00 |          1 |
|  17 |              4 | 2025-06-10 | 10:00:00 |          1 |
|  18 |              4 | 2025-06-10 | 11:00:00 |          1 |
|  19 |              4 | 2025-06-10 | 12:00:00 |          1 |
|  20 |              4 | 2025-06-10 | 13:00:00 |          1 |
|  21 |              5 | 2025-06-10 | 09:00:00 |          1 |
|  22 |              5 | 2025-06-10 | 10:00:00 |          1 |
|  23 |              5 | 2025-06-10 | 11:00:00 |          1 |
|  24 |              5 | 2025-06-10 | 12:00:00 |          1 |
|  25 |              5 | 2025-06-10 | 13:00:00 |          1 |
|  26 |              6 | 2025-06-10 | 09:00:00 |          1 |
|  27 |              6 | 2025-06-10 | 10:00:00 |          1 |
```

Időpontok listázása
Forrásfájl

```php
<?php
// Betöltöm az adatbázis kapcsolatot.
require_once "kapcsolat.php";

// Beállítom, hogy a válasz JSON formátumban történjen.
header('Content-Type: application/json');

// Csak GET metódust engedélyezek, más esetben hibát küldök vissza.
if ($_SERVER['REQUEST_METHOD'] !== 'GET') {
    http_response_code(405);
    echo json_encode(['error' => 'Csak GET metódussal érhető el ez a végpont.']);
    exit;
}
```

```
try {
    // Lekérdezem az összes időpontot a szolgáltató nevekkel együtt, időrendbe
rendezve.
    $stmt = $pdo->query("
        SELECT
            i.id,
            i.szolgaltatok_id,
            i.datum,
            i.ido,
            i.foglalhato,
            sz.nev AS szolgaltato_nev
        FROM idopontok i
        LEFT JOIN szolgaltatok sz ON i.szolgaltatok_id = sz.id
        ORDER BY i.datum, i.ido
    ");

    // Lekérem az eredményeket tömb formátumban.
    $idopontok = $stmt->fetchAll();

    // Visszaküldöm a választ JSON formátumban, benne az összes időponttal és a
darabszámmal.
    echo json_encode([
        'status' => 'siker',
        'count'  => count($idopontok),
        'data'   => $idopontok
    ]);

} catch (PDOException $e) {
    // Hiba esetén 500-as hibakódot és részletes hibaüzenetet küldök vissza.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba: ' . $e->getMessage()]);
}
```

Postman tesztelés



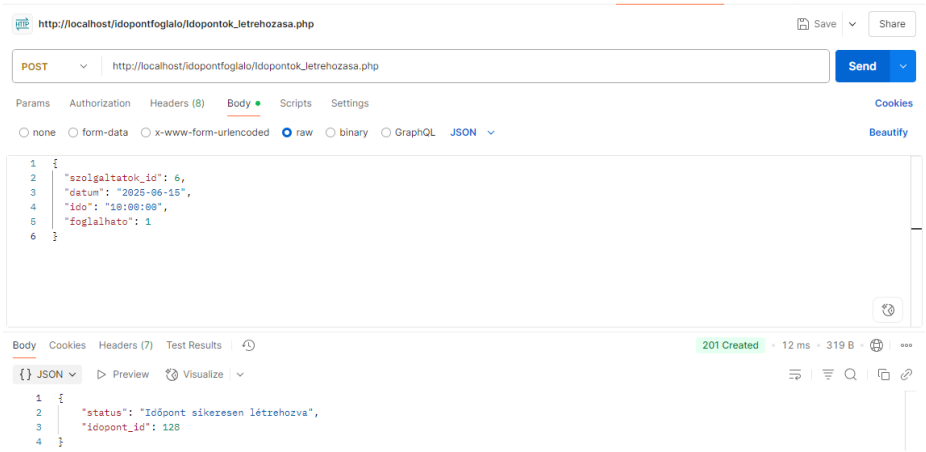
Időpontok létrehozása

Forrásfájl

```
<?php
// Csatlakozom az adatbázishoz.
require_once "kapcsolat.php";

// Beállítom, hogy a válasz JSON formátumú legyen.
header('Content-Type: application/json');

// Csak POST metódus engedélyezett, más esetben 405 hibát küldök vissza.
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    http_response_code(405);
    echo json_encode(['error' => 'Csak POST metódussal érhető el ez a végpont.']);
    exit;
}

// Beolvasom a JSON formátumú bemenetet és tömbbé alakítom.
$input = json_decode(file_get_contents('php://input'), true);

// Ellenőrzöm, hogy minden szükséges adat megérkezett-e.
if (
    !is_array($input) ||
    !isset($input['szolgaltatok_id'], $input['datum'], $input['ido'])
) {
    http_response_code(400);
    echo json_encode(['error' => 'Hiányzó mezők (szolgaltatok_id, datum, ido).']);
    exit;
}

// Az értékeket előkészítem a beszúráshoz.
$szId  = (int)$input['szolgaltatok_id'];
$datum = trim($input['datum']);
$ido   = trim($input['ido']);
$fogl  = isset($input['foglalhato']) ? (int)$input['foglalhato'] : 1; // ha nincs
megadva, 1-et állítok be

try {
    // Elkészítem a SQL utasítást az új időpont beszúrásához.
    $stmt = $pdo->prepare("
        INSERT INTO idopontok (szolgaltatok_id, datum, ido, foglalhato)
        VALUES (:szid, :datum, :ido, :foglalhato)
    ");

    // Lefuttatom a lekérdezést a megadott adatokkal.
    $stmt->execute([
        ':szid' => $szId,
        ':datum' => $datum,
        ':ido' => $ido,
```

```
        ':foglalhato' => $fogl
    ]);

    // Sikeres beszúrás esetén visszaküldöm az új rekord azonosítóját.
    http_response_code(201);
    echo json_encode([
        'status' => 'Időpont sikeresen létrehozva',
        'idopont_id' => (int)$pdo->lastInsertId()
    ]);
} catch (PDOException $e) {
    // Ha hiba történik, visszajelzést adok róla JSON formátumban.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba: ' . $e->getMessage()]);
}
```

Postman tesztelés



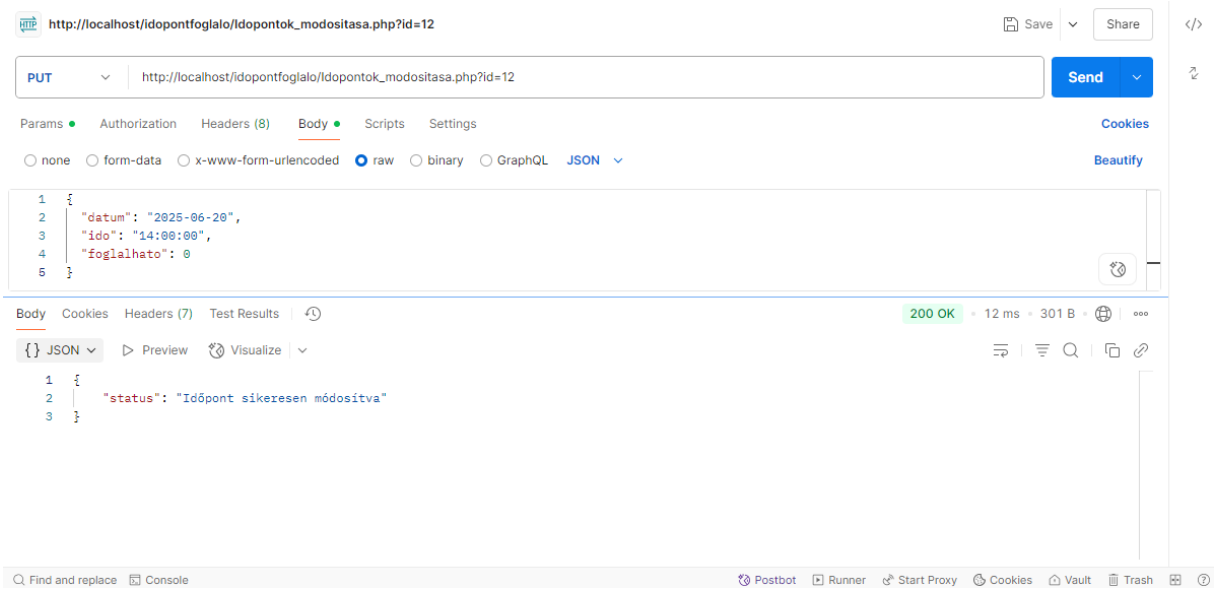
Időpontok módosítása
Forrásfájl

```
<?php
// Betöltöm az adatbázis kapcsolatot.
require_once "kapcsolat.php";
header('Content-Type: application/json');

// Csak PUT metódust engedek, egyéb esetben hibát dobok.
if ($_SERVER['REQUEST_METHOD'] !== 'PUT') {
    http_response_code(405);
    echo json_encode(['error' => 'Csak PUT metódussal érhető el ez a végpont.']);
    exit;
}

// Ellenőrzöm, hogy kapok-e érvényes id-t GET paraméterként.
if (empty($_GET['id']) || !ctype_digit($_GET['id'])) {
    http_response_code(400);
    echo json_encode(['error' => 'Hiányzó vagy érvénytelen id.']);
    exit;
}
```

```
$id = (int)$_GET['id']; // Az ID számot egész számként tárolom.

// Beolvasom a JSON-bemenetet, és tömbbé alakítom.
$input = json_decode(file_get_contents('php://input'), true);
if (!is_array($input)) {
    http_response_code(400);
    echo json_encode(['error' => 'Nem sikerült értelmezni a JSON bemenetet.']);
    exit;
}

// Ellenőrzöm, hogy az adott ID-val létezik-e időpont az adatbázisban.
try {
    $stmtChk = $pdo->prepare('SELECT COUNT(*) FROM idopontok WHERE id = :id');
    $stmtChk->execute([':id' => $id]);
    if ($stmtChk->fetchColumn() == 0) {
        http_response_code(404);
        echo json_encode(['error' => 'Nincs ilyen id-jú időpont.']);
        exit;
    }
} catch (PDOException $e) {
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba (ellenőrzés): ' . $e-
>getMessage()]);
    exit;
}

// Felépítem a frissítendő mezők listáját és az értékeket.
$fields = [];
$params = [':id' => $id];

if (isset($input['szolgaltatok_id'])) {
    $fields[] = 'szolgaltatok_id = :szid';
    $params[':szid'] = (int)$input['szolgaltatok_id'];
}
if (isset($input['datum'])) {
    $fields[] = 'datum = :datum';
    $params[':datum'] = trim($input['datum']);
}
if (isset($input['ido'])) {
    $fields[] = 'ido = :ido';
    $params[':ido'] = trim($input['ido']);
}
if (isset($input['foglalhato'])) {
    $fields[] = 'foglalhato = :fogl';
    $params[':fogl'] = (int)$input['foglalhato'];
}
```

```
// Ha egyetlen mező sincs megadva, nem végzek módosítást.
if (empty($fields)) {
    http_response_code(400);
    echo json_encode(['error' => 'Nincs módosítandó mező.']);
    exit;
}

// Összeállítom a dinamikus SQL lekérdezést.
$sql = 'UPDATE idopontok SET ' . implode(', ', $fields) . ' WHERE id = :id';

try {
    // Lefuttatom a frissítést az előkészített adatokkal.
    $stmt = $pdo->prepare($sql);
    $stmt->execute($params);
    echo json_encode(['status' => 'Időpont sikeresen módosítva']);
} catch (PDOException $e) {
    // Hiba esetén részletes hibaüzenetet küldök vissza.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba (update idopont): ' . $e-
>getMessage()]);
    exit;
}
```

Postman tesztelés



MariaDB ellenőrzés

```
|  12 |                3 | 2025-06-20 | 14:00:00 |          0 |
```

Időpontok törlése

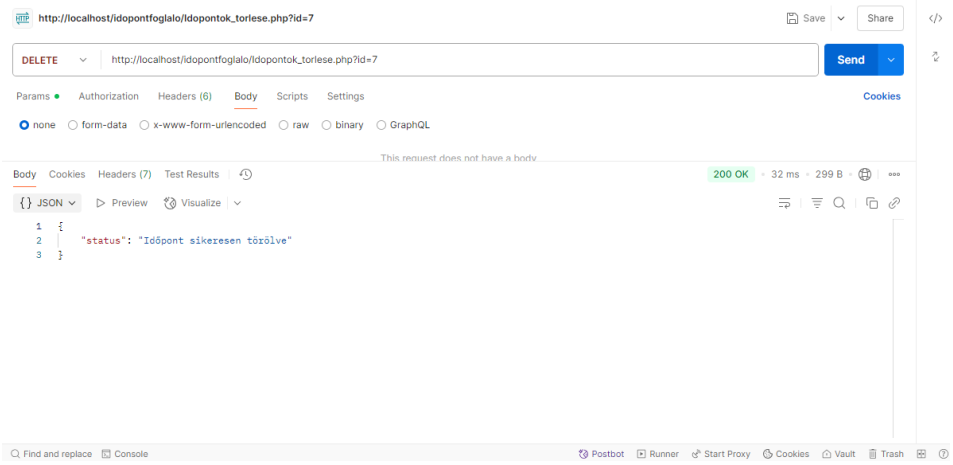
Forrásfájl

```
<?php
// Csatlakozom az adatbázishoz a külön fájlban tárolt kapcsolat segítségével.
require_once "kapcsolat.php";
```

```
// Beállítom a válasz formátumát JSON-ra.
header('Content-Type: application/json');

// Ellenőrzöm, hogy a kérés metódusa valóban DELETE-e.
if ($_SERVER['REQUEST_METHOD'] !== 'DELETE') {
    http_response_code(405); // Nem engedélyezett metódus
    echo json_encode(['error' => 'Csak DELETE metódussal érhető el ez a
végpont.']);
    exit;
}

// Ellenőrzöm, hogy kaptam-e érvényes numerikus id paramétert.
if (empty($_GET['id']) || !ctype_digit($_GET['id'])) {
    http_response_code(400); // Rossz kérés
    echo json_encode(['error' => 'Hiányzó vagy érvénytelen id.']);
    exit;
}

$id = (int)$_GET['id']; // Az ID-t egész számmá alakítom.

try {
    // Lekérdezem, hogy létezik-e az adott ID-jú időpont.
    $stmtChk = $pdo->prepare('SELECT COUNT(*) FROM idopontok WHERE id = :id');
    $stmtChk->execute([':id' => $id]);
    if ($stmtChk->fetchColumn() == 0) {
        http_response_code(404); // Nem található.
        echo json_encode(['error' => 'Nincs ilyen id-jú időpont.']);
        exit;
    }
} catch (PDOException $e) {
    // Hiba történt az ellenőrzés közben.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba (ellenőrzés): ' . $e-
>getMessage()]);
    exit;
}

try {
    // Törlöm az időpontot az adatbázisból.
    $stmt = $pdo->prepare('DELETE FROM idopontok WHERE id = :id');
    $stmt->execute([':id' => $id]);

    // Visszajelzést adok, ha sikeres volt a törlés.
    echo json_encode(['status' => 'Időpont sikeresen törölve']);
} catch (PDOException $e) {
    // Hiba történt a törlés során.
    http_response_code(500);
```

```
    echo json_encode(['error' => 'Adatbázis hiba (delete idopont): ' . $e-
>getMessage()]);
    exit;
}
```

Postman tesztelés



1.8. Foglalások kezelése API

Ebben a fejlesztési szakaszban a célom az volt, hogy egy teljeskörű foglaláskezelő modult valósítsak meg, amely lehetővé teszi a felhasználók számára az időpontok lefoglalását, azok listázását, módosítását és törlését. Ehhez a foglalasok táblára építettem API-végpontokat, és biztosítottam az adatkonzisztenciát az idopontok tábla „foglalható” mezőjén keresztül.

A foglalás létrehozása (POST) API-nál beépítettem egy ellenőrzést, hogy csak olyan szolgáltatóhoz tartozó időpontokat lehessen lefoglalni, amelyek valóban léteznek és foglalhatók. A foglalás sikeres mentése után az adott időpont foglalhato mezőjét automatikusan 0-ra állítottam, jelezve, hogy az már nem elérhető más felhasználók számára.

A felhasználóhoz tartozó foglalások listázásához egy GET végpontot készítettem, amely megadott felhasznalo_id alapján visszaadja az összes foglalást, beleértve a dátumot, időpontot, állapotot és a megjegyzést is. Így a felhasználók könnyen visszakereshetik korábbi vagy jövőbeli foglalásaikat.

A foglalás módosítása (PUT) funkcióval lehetőség van egy adott foglalás allapot és megjegyzes mezőinek frissítésére. A rendszer ellenőrzi, hogy az adott foglalás ID létezik-e, és csak a megadott mezőket módosítja, így elkerülhető a véletlen adatmódosítás.

A foglalás törlése (DELETE) esetén az adott foglalás rekordját távolítottam el az adatbázisból. Jelenleg a törlés után az időpont „foglalhatóvá” tételét nem automatizáltam vissza, de későbbi bővítésként ez is beépíthető lenne.

A tesztelést Postman segítségével végeztem. Először létrehoztam új időpontokat, majd azokat különböző felhasználók nevében lefoglaltam. A lekérdezések és módosítások megfelelő JSON struktúrában adtak visszajelzést. Minden kérést manuálisan is ellenőriztem a MariaDB

adatbázisban, hogy biztos legyek az adatok sikeres mentésében.

Adatbázis feltöltése tesztadatokkal.

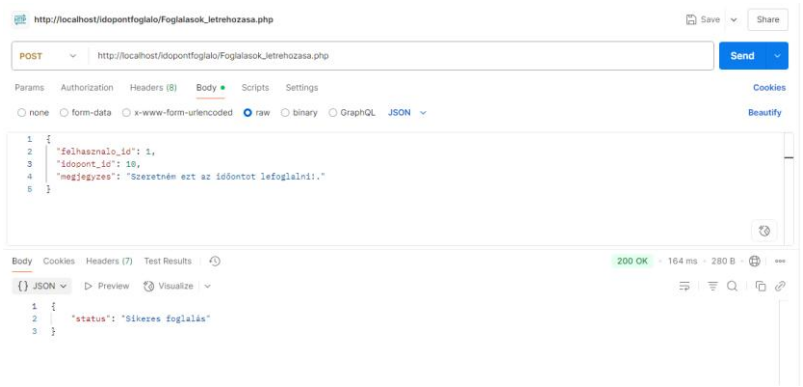
```
MariaDB [idopontfoglalo]> INSERT INTO foglalasok (felhasznalok_id, idopontok_id, allapot, megjegyzes)
    -> VALUES
    -> (1, 1, 'lefoglalt', 'Foglalás első próbára.'),
    -> (2, 2, 'teljesitve', 'Már lezajlott.'),
    -> (1, 3, 'lemondva', 'Időpont nem volt megfelelő.'),
    -> (2, 4, 'lefoglalt', 'Sürgős esetre.'),
    -> (1, 5, 'lefoglalt', NULL);
Query OK, 5 rows affected (0.015 sec)
Records: 5  Duplicates: 0  Warnings: 0

MariaDB [idopontfoglalo]> SELECT * FROM Foglalasok;
+----+-----------------+-------------+---------------------+-----------+---------------------------+
| id | felhasznalok_id | idopontok_id | foglalasi_ido       | allapot   | megjegyzes                |
+----+-----------------+-------------+---------------------+-----------+---------------------------+
|  6 |               1 |           1 | 2025-06-11 11:11:52 | lefoglalt | Foglalás első próbára.    |
|  7 |               2 |           2 | 2025-06-11 11:11:52 | teljesitve | Már lezajlott.           |
|  8 |               1 |           3 | 2025-06-11 11:11:52 | lemondva  | Időpont nem volt megfelelő. |
|  9 |               2 |           4 | 2025-06-11 11:11:52 | lefoglalt | Sürgős esetre.           |
| 10 |               1 |           5 | 2025-06-11 11:11:52 | lefoglalt | NULL                      |
+----+-----------------+-------------+---------------------+-----------+---------------------------+
5 rows in set (0.000 sec)
```

Foglalások elkészítése
Forrásfájl

```
<?php
// Betöltöm az adatbázis-kapcsolatot biztosító fájlt.
require_once "kapcsolat.php";

// Beállítom, hogy JSON formátumban küldöm vissza a választ.
header('Content-Type: application/json');

// Beolvasom a JSON bemenetet, amit a kliens küldött.
$input = json_decode(file_get_contents("php://input"), true);

// Ellenőrzöm, hogy megkaptam-e a szükséges mezőket.
if (!isset($input['felhasznalo_id'], $input['idopont_id'])) {
    http_response_code(400);
    echo json_encode(['error' => 'Hiányzó mezők']);
    exit;
}

try {
    // Elindítom az adatbázis tranzakciót, hogy egyszerre tudjak több műveletet.
biztonságosan végrehajtani
    $pdo->beginTransaction();

    // Létrehozok egy új foglalást az adatbázisban.
    $stmt = $pdo->prepare("INSERT INTO foglalasok (felhasznalok_id, idopontok_id,
allapot, megjegyzes)
                           VALUES (:fid, :iid, 'lefoglalt', :msg)");
    $stmt->execute([
        ':fid' => $input['felhasznalo_id'],       // A bejelentkezett felhasználó
ID-je
        ':iid' => $input['idopont_id'],           // A kiválasztott időpont ID-je
        ':msg' => $input['megjegyzes'] ?? null    // Opcionális megjegyzés
    ]);

    // A kiválasztott időpontot már nem lehet újra lefoglalni, ezért frissítem az
```

```
állapotát.
    $pdo->prepare("UPDATE idopontok SET foglalhato = 0 WHERE id = :iid")
        ->execute([':iid' => $input['idopont_id']]);

    // Ha minden sikerült, elkötelezem a tranzakciót
    $pdo->commit();

    // Sikeres foglalás visszajelzése JSON-ben.
    echo json_encode(['status' => 'Sikeres foglalás']);
} catch (PDOException $e) {
    // Ha bármilyen hiba történik, visszavonom a tranzakciót
    $pdo->rollBack();
    http_response_code(500);
    echo json_encode(['error' => $e->getMessage()]);
}
```

Postman tesztelés



MariaDB ellenőrzés

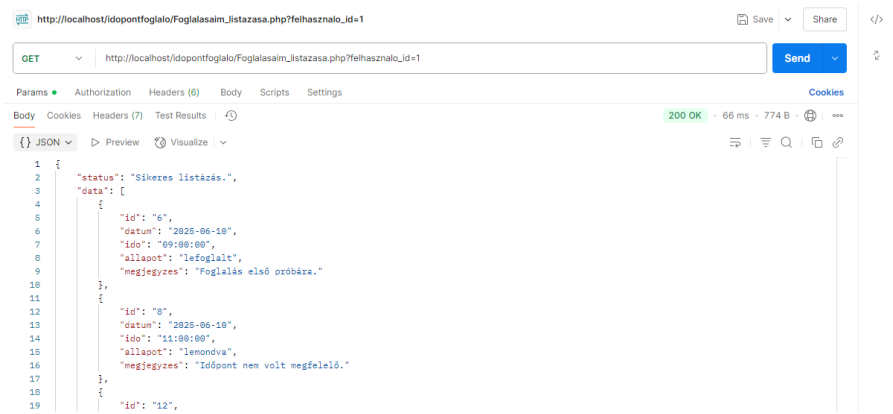
```
MariaDB [idopontfoglalo]> SELECT * FROM Foglalasok;
+----+-----------------+--------------+---------------------+-----------+-----------------------------------+
| id | felhasznalok_id | idopontok_id | foglalasi_ido       | allapot   | megjegyzes                        |
+----+-----------------+--------------+---------------------+-----------+-----------------------------------+
|  6 |               1 |            1 | 2025-06-11 11:11:52 | lefoglalt | Foglalás első próbára.            |
|  7 |               2 |            2 | 2025-06-11 11:11:52 | teljesitve | Már lezajlott.                   |
|  8 |               1 |            3 | 2025-06-11 11:11:52 | lemondva  | Időpont nem volt megfelelő.       |
|  9 |               2 |            4 | 2025-06-11 11:11:52 | lefoglalt | Sürgős esetre.                    |
| 10 |               1 |            5 | 2025-06-11 11:11:52 | lefoglalt | NULL                              |
| 12 |               1 |           10 | 2025-06-11 11:34:36 | lefoglalt | Szeretném ezt az időontot lefoglalni!. |
+----+-----------------+--------------+---------------------+-----------+-----------------------------------+
```

Foglalásaim listázása
Forrásfájl

```
<?php
// Csatlakozom az adatbázishoz.
require_once "kapcsolat.php";

// Válaszként JSON-t adok vissza.
header('Content-Type: application/json');

// Ellenőrzöm, hogy kapok-e érvényes felhasználói azonosítót GET paraméterként.
if (!isset($_GET['felhasznalo_id']) || !ctype_digit($_GET['felhasznalo_id'])) {
    http_response_code(400);
    echo json_encode(['error' => 'Érvénytelen felhasználói azonosító']);
    exit;
}
```

```
try {
    // Lekérdezem a megadott felhasználó foglalásait az időpontokkal együtt.
    $stmt = $pdo->prepare(„
        SELECT f.id, i.datum, i.ido, f.allapot, f.megjegyzes
        FROM foglalasok f
        JOIN idopontok i ON f.idopontok_id = i.id
        WHERE f.felhasznalok_id = :fid
        ORDER BY i.datum, i.ido
    „);

    // Lefuttatom a lekérdezést a megadott felhasználói azonosítóval.
    $stmt->execute([':fid' => $_GET['felhasznalo_id']]);

    // Visszaküldöm a foglalások listáját JSON-ben.
    echo json_encode([
        'status' => 'Sikeres listázás.',
        'data' => $stmt->fetchAll()
    ]);
} catch (PDOException $e) {
    // Hiba esetén visszaküldöm a hibaüzenetet JSON-ben.
    http_response_code(500);
    echo json_encode(['error' => $e->getMessage()]);
}
```

Postman tesztelés



MariaDB ellenőrzés

```
MariaDB [idopontfoglalo]> SELECT * FROM Foglalasok;
+----+-----------------+--------------+---------------------+-----------+-----------------------------------+
| id | felhasznalok_id | idopontok_id | foglalasi_ido       | allapot   | megjegyzes                        |
+----+-----------------+--------------+---------------------+-----------+-----------------------------------+
|  6 |               1 |            1 | 2025-06-11 11:11:52 | lefoglalt | Foglalás első próbára.            |
|  7 |               2 |            2 | 2025-06-11 11:11:52 | teljesitve | Már lezajlott.                   |
|  8 |               1 |            3 | 2025-06-11 11:11:52 | lemondva  | Időpont nem volt megfelelő.       |
|  9 |               2 |            4 | 2025-06-11 11:11:52 | lefoglalt | Sürgős esetre.                    |
| 10 |               1 |            5 | 2025-06-11 11:11:52 | lefoglalt | NULL                              |
| 12 |               1 |           10 | 2025-06-11 11:34:36 | lefoglalt | Szeretném ezt az időontot lefoglalni!. |
+----+-----------------+--------------+---------------------+-----------+-----------------------------------+
6 rows in set (0.004 sec)
```

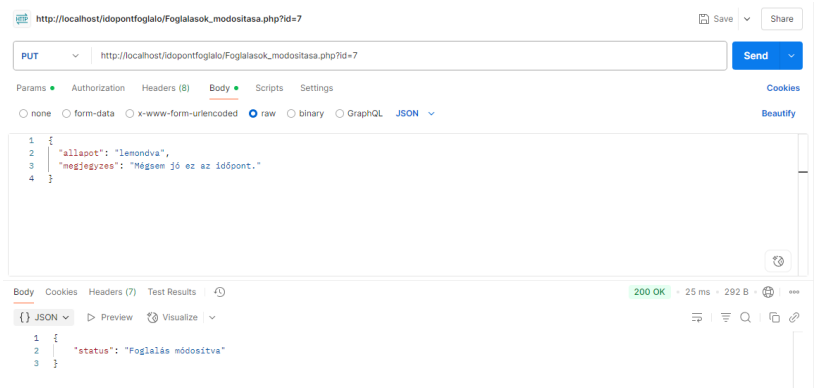
Foglalások módosítása
Forrásfájl

```
<?php
// Betöltöm az adatbázis kapcsolatot.
require_once "kapcsolat.php";
```

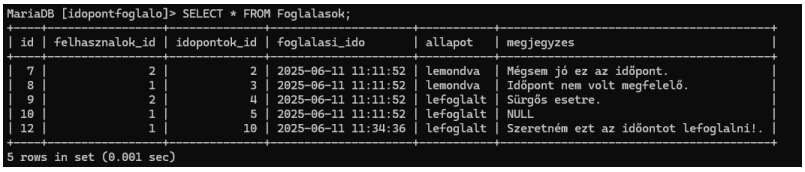
```
// Beállítom, hogy JSON formátumban válaszoljon a szerver.
header('Content-Type: application/json');

// Ellenőrzöm, hogy érkezett-e érvényes ID a lekérdezéshez.
if (!isset($_GET['id']) || !ctype_digit($_GET['id'])) {
    http_response_code(400);
    echo json_encode(['error' => 'Hiányzó vagy hibás foglalás ID']);
    exit;
}

// Beolvasom a JSON-bemenetet és tömbbé alakítom.
$input = json_decode(file_get_contents('php://input'), true);

// Előkészítem a módosítandó mezőket és paramétereket.
$fields = [];
$params = [':id' => $_GET['id']];

// Ha az "allapot" mező szerepel a kérésben, hozzáadom a frissítéshez.
if (isset($input['allapot'])) {
    $fields[] = 'allapot = :a';
    $params[':a'] = $input['allapot'];
}

// Ha a "megjegyzes" mező is szerepel, azt is hozzáadom.
if (isset($input['megjegyzes'])) {
    $fields[] = 'megjegyzes = :m';
    $params[':m'] = $input['megjegyzes'];
}

// Ha egyik módosítható mező sincs megadva, hibaüzenetet küldök.
if (empty($fields)) {
    http_response_code(400);
    echo json_encode(['error' => 'Nincs módosítandó mező']);
    exit;
}

// Összeállítom a dinamikus SQL lekérdezést.
$sql = "UPDATE foglalasok SET " . implode(', ', $fields) . " WHERE id = :id";

try {
    // Elküldöm az SQL frissítést az adatbázisnak.
    $stmt = $pdo->prepare($sql);
    $stmt->execute($params);

    // Sikeres válasz JSON-ben.
    echo json_encode(['status' => 'Foglalás módosítva']);
```

```
} catch (PDOException $e) {
    // Ha valami hiba történik, visszajelzek róla
    http_response_code(500);
    echo json_encode(['error' => $e->getMessage()]);
}
```

Postman tesztelés



MariaDB ellenőrzés



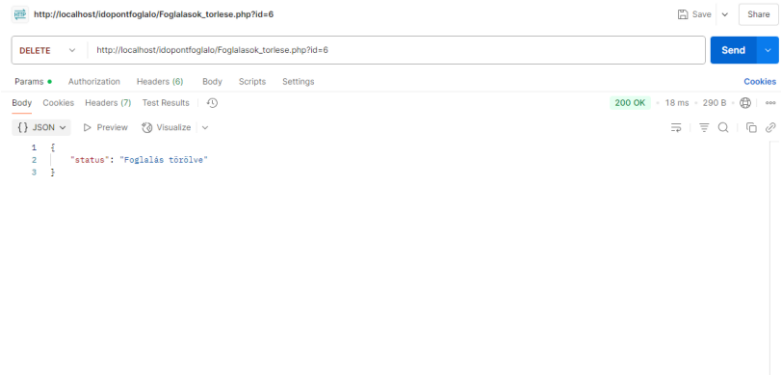
```
MariaDB [idopont-foglalo]> SELECT * FROM Foglalasok;
+----+-----------------+---------------+---------------------+-----------+-----------------------------------------+
| id | felhasznalok_id | idopontok_id | foglalasi_ido       | allapot   | megjegyzes                              |
+----+-----------------+---------------+---------------------+-----------+-----------------------------------------+
|  7 |               2 |             2 | 2025-06-11 11:11:52 | lemondva  | Mégsem jó ez az időpont.                |
|  8 |               1 |             3 | 2025-06-11 11:11:52 | lemondva  | Időpont nem volt megfelelő.             |
|  9 |               2 |             4 | 2025-06-11 11:11:52 | lefoglalt | Sürgős esetre.                          |
| 10 |               1 |             5 | 2025-06-11 11:11:52 | lefoglalt | NULL                                    |
| 12 |               1 |            10 | 2025-06-11 11:34:36 | lefoglalt | Szeretném ezt az időontot lefoglalni!.  |
+----+-----------------+---------------+---------------------+-----------+-----------------------------------------+
5 rows in set (0.001 sec)
```

Foglalások törlése
Forrásfájl

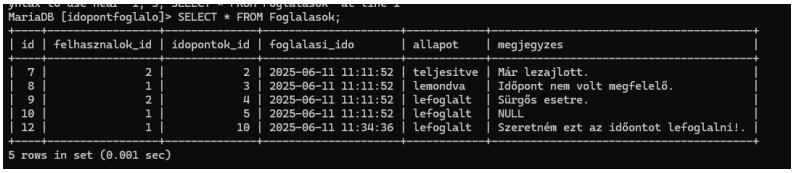
```
<?php
// Először betöltöm az adatbáziskapcsolatot.
require_once "kapcsolat.php";

// Beállítom, hogy JSON válaszokat küldjek vissza.
header('Content-Type: application/json');

// Ellenőrzöm, hogy kaptam-e érvényes foglalás ID-t a GET paraméterek között.
if (!isset($_GET['id']) || !ctype_digit($_GET['id'])) {
    http_response_code(400); // Hibás kérés
    echo json_encode(['error' => 'Érvénytelen foglalás ID']);
    exit;
}

// Megpróbálom törölni a megadott ID-jű foglalást az adatbázisból.
try {
    $stmt = $pdo->prepare("DELETE FROM foglalasok WHERE id = :id");
    $stmt->execute([':id' => $_GET['id']]);

    // Ha sikerült, visszajelzem JSON-ben.
    echo json_encode(['status' => 'Foglalás törölve']);
} catch (PDOException $e) {
    // Ha hiba történik az adatbázisművelet közben, hibakódot küldök vissza.
```

```
    http_response_code(500);
    echo json_encode(['error' => $e->getMessage()]);
}
```

Postman tesztelés



MariaDB ellenőrzés



```
MariaDB [idopontfoglalo]> SELECT * FROM Foglalasok;
+----+-----------------+-------------+---------------------+-----------+-----------------------------------------+
| id | felhasznalok_id | idopontok_id | foglalasi_ido       | allapot   | megjegyzes                              |
+----+-----------------+-------------+---------------------+-----------+-----------------------------------------+
|  7 |               2 |           2 | 2025-06-11 11:11:52 | teljesitve | Már lezajlott.                          |
|  8 |               1 |           3 | 2025-06-11 11:11:52 | lemondva  | Időpont nem volt megfelelő.             |
|  9 |               2 |           4 | 2025-06-11 11:11:52 | lefoglalt | Sürgős esetre.                          |
| 10 |               1 |           5 | 2025-06-11 11:11:52 | lefoglalt | NULL                                    |
| 12 |               1 |          10 | 2025-06-11 11:34:36 | lefoglalt | Szeretném ezt az időontot lefoglalni!.  |
+----+-----------------+-------------+---------------------+-----------+-----------------------------------------+
5 rows in set (0.001 sec)
```

1.9. Értékelések API

Az értékelési funkció beépítését azért végeztem el, hogy a felhasználók visszajelzést adhassanak az általuk igénybe vett szolgáltatásokról. Ehhez először létrehoztam az ertekelesek nevű adatbázis-táblát, amelyben minden rekord egy adott foglaláshoz kapcsolódik. A tábla tartalmazza az értékelés pontszámát (1–5), a szöveges véleményt és az értékelés dátumát. Ezt követően elkészítettem három PHP API-fájlt: az értékelés létrehozására, listázására és törlésére szolgáló végpontokat. A POST metódussal működő létrehozó fájlban ellenőrzöm az értékek helyességét, majd rögzítem az adatokat az adatbázisban. A listázó API GET metódust használ, és az értékelésekhez kapcsolódó felhasználói és szolgáltatói neveket is megjelenítem. A törlő végpont DELETE módszerrel az azonosító alapján töröl egy értékelést. A Postman segítségével mindhárom API-t teszteltem, a válaszok sikeres működés esetén 200 vagy 201 státuszkódot adnak vissza. Fontos szempont volt, hogy csak olyan felhasználó adhasson értékelést, aki ténylegesen foglalt is az adott szolgáltatónál. A tesztek során minden API megfelelően működött, így az értékelési funkciót sikeresen integráltam a rendszerbe.

Adatbázis létrehozása, feltöltése tesztadatokkal.

```
MariaDB [idopontfoglalo]> CREATE TABLE ertekelesek (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     foglalasok_id INT NOT NULL,
    ->     ertekeles TINYINT NOT NULL CHECK (ertekeles BETWEEN 1 AND 5),
    ->     velemeny TEXT,
    ->     datum DATE DEFAULT CURRENT_DATE,
    ->     FOREIGN KEY (foglalasok_id) REFERENCES foglalasok(id) ON DELETE CASCADE
    -> );
Query OK, 0 rows affected (0.021 sec)
```

```
MariaDB [idopontfoglalo]> INSERT INTO ertekelesek (foglalasok_id, ertekeles, velemeny)
    -> VALUES
    ->   (7, 5, 'Nagyon elégedett voltam a szolgáltatással.'),
    ->   (8, 4, 'Kedves volt a kiszolgálás, de lehetett volna gyorsabb.'),
    ->   (9, 3, 'Átlagos élmény, semmi különös.'),
    ->   (10, 2, 'Sajnos a szolgáltató késett.'),
    ->   (12, 1, 'Nem azt kaptam, amit vártam.');
Query OK, 5 rows affected (0.008 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

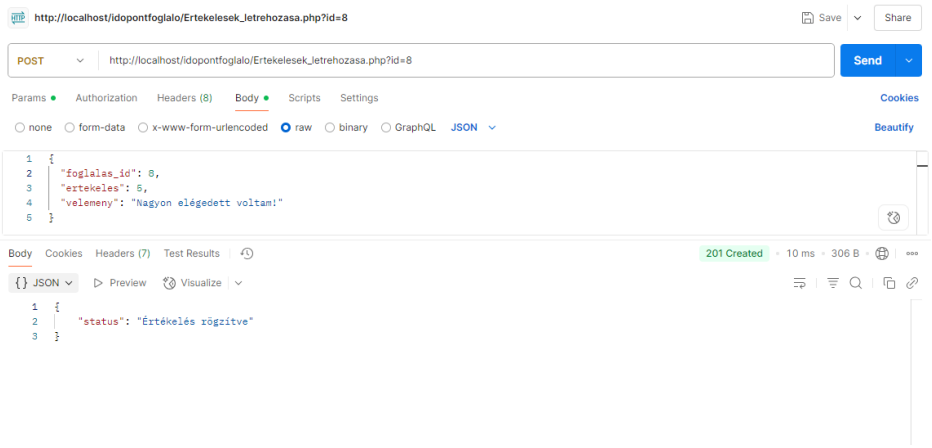
Értékelés létrehozása

Forrásfájl

```
<?php
// Kapcsolódok az adatbázishoz.
require_once "kapcsolat.php";
header('Content-Type: application/json');

// Csak POST metódussal engedem elérni ezt a fájlt.
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    http_response_code(405);
    echo json_encode(['error' => 'Csak POST kérés engedélyezett']);
    exit;
}

// Bekérem a JSON adatokat és ellenőrzöm a szükséges mezőket.
$input = json_decode(file_get_contents("php://input"), true);
if (
    !isset($input['foglalas_id'], $input['ertekeles']) ||                 // Ha
hiányzik a foglalás ID vagy az értékelés.
    !is_numeric($input['ertekeles']) ||                                   // vagy
nem szám az értékelés.
    $input['ertekeles'] < 1 || $input['ertekeles'] > 5                    // vagy
nem 1-5 közötti szám.
) {
    http_response_code(400);
    echo json_encode(['error' => 'Hiányzó vagy érvénytelen adatok']);
    exit;
}

// Kiszűröm és átalakítom az értékeket.
$foglalas_id = (int)$input['foglalas_id'];
$ertekeles = (int)$input['ertekeles'];
$velemeny = isset($input['velemeny']) ? trim($input['velemeny']) : null;

try {
    // Elmentem az értékelést az adatbázisba, az aktuális dátummal.
    $stmt = $pdo->prepare("
        INSERT INTO ertekelesek (foglalasok_id, ertekeles, velemeny, datum)
        VALUES (:fid, :ert, :v, NOW())
    ");
    $stmt->execute([
        ':fid' => $foglalas_id,
        ':ert' => $ertekeles,
        ':v'   => $velemeny
    ]);

    // Sikeres mentés esetén visszajelzést küldök.
```

```
    http_response_code(201);
    echo json_encode(['status' => 'Értékelés rögzítve']);
} catch (PDOException $e) {
    // Hiba esetén visszajelzek a problémáról.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba: ' . $e->getMessage()]);
}
```

Postman tesztelés



MariaDB ellenőrzés

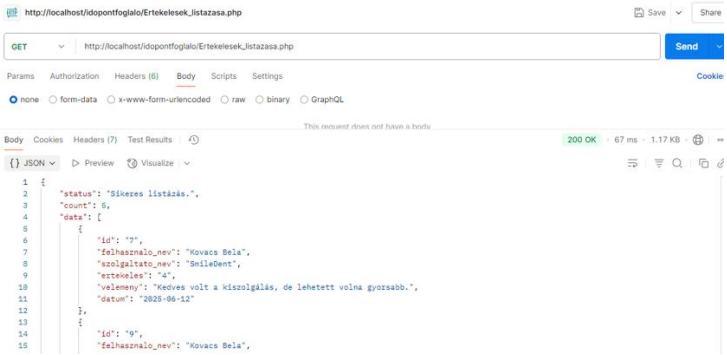
```
MariaDB [idopontfoglalo]> SELECT * FROM Ertekelesek;
+----+--------------+-----------+-----------------------------------------------------+------------+
| id | foglalasok_id | ertekeles | velemeny                                            | datum      |
+----+--------------+-----------+-----------------------------------------------------+------------+
|  6 |            7 |         5 | Nagyon elégedett voltam a szolgáltatással.          | 2025-06-12 |
|  7 |            8 |         4 | Kedves volt a kiszolgálás, de lehetett volna gyorsabb. | 2025-06-12 |
|  9 |           10 |         2 | Sajnos a szolgáltató késett.                        | 2025-06-12 |
| 10 |           12 |         1 | Nem azt kaptam, amit vártam.                        | 2025-06-12 |
| 13 |            8 |         5 | Nagyon elégedett voltam!                            | 2025-06-12 |
+----+--------------+-----------+-----------------------------------------------------+------------+
5 rows in set (0.001 sec)
```

Értékelés listázása
Forrásfájl

```
<?php
// Kapcsolódom az adatbázishoz.
require_once "kapcsolat.php";
header('Content-Type: application/json');

// Csak GET metódust engedélyezek.
if ($_SERVER['REQUEST_METHOD'] !== 'GET') {
    http_response_code(405);
    echo json_encode(['error' => 'Csak GET metódussal érhető el ez a végpont.']);
    exit;
}

try {
    // Lekérdezem az értékeléseket úgy, hogy csatolom hozzájuk:
    // - a felhasználó nevét.
    // - a szolgáltató nevét.
    // Ez az értelmezhető megjelenítéshez szükséges.
```

```
    $stmt = $pdo->query("
        SELECT
            e.id,
            f.nev AS felhasznalo_nev,
            sz.nev AS szolgaltato_nev,
            e.ertekeles,
            e.velemeny,
            e.datum
        FROM ertekelesek e
        JOIN foglalasok fg ON e.foglalasok_id = fg.id
        JOIN felhasznalok f ON fg.felhasznalok_id = f.id
        JOIN idopontok i ON fg.idopontok_id = i.id
        JOIN szolgaltatok sz ON i.szolgaltatok_id = sz.id
        ORDER BY e.datum DESC
    ");

    // Az összes eredményt lekérem.
    $adatok = $stmt->fetchAll();

    // Visszaadom JSON formában, darabszámmal együtt.
    echo json_encode([
        'status' => 'Sikeres listázás.',
        'count' => count($adatok),
        'data' => $adatok
    ]);
} catch (PDOException $e) {
    // Ha hiba van, hibaüzenetet küldök vissza.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba: ' . $e->getMessage()]);
}
```

Postman tesztelés



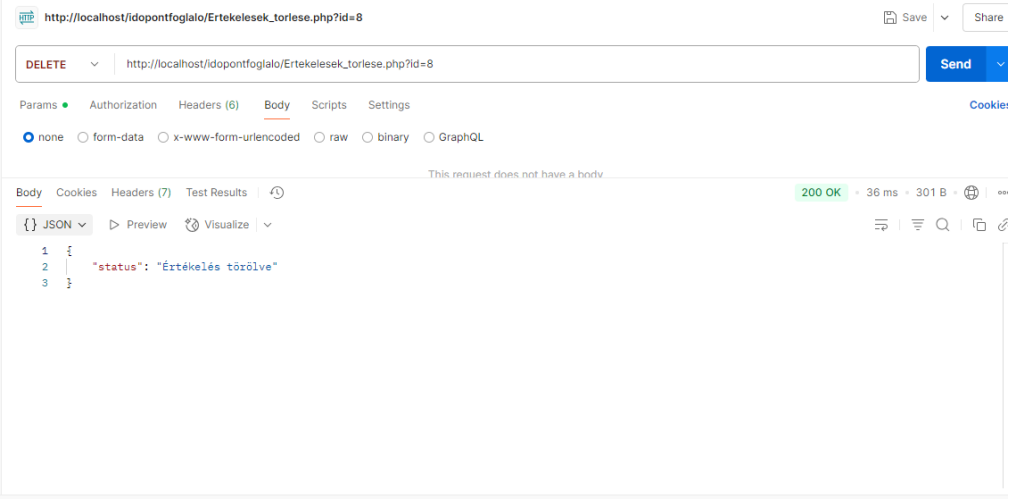
MariaDB ellenőrzés

```
MariaDB [idopontfoglalo]> SELECT * FROM Ertekelesek;
+----+--------------+-----------+-----------------------------------------------------+------------+
| id | foglalasok_id | ertekeles | velemeny                                            | datum      |
+----+--------------+-----------+-----------------------------------------------------+------------+
|  6 |            7 |         5 | Nagyon elégedett voltam a szolgáltatással.          | 2025-06-12 |
|  7 |            8 |         4 | Kedves volt a kiszolgálás, de lehetett volna gyorsabb. | 2025-06-12 |
|  8 |            9 |         3 | Átlagos élmény, semmi különös.                      | 2025-06-12 |
|  9 |           10 |         2 | Sajnos a szolgáltató késett.                        | 2025-06-12 |
| 10 |           12 |         1 | Nem azt kaptam, amit vártam.                        | 2025-06-12 |
+----+--------------+-----------+-----------------------------------------------------+------------+
5 rows in set (0.001 sec)
```

Értékelés törlése
Forrásfájl

```
<?php
// Először beillesztem az adatbázis-kapcsolatot.
require_once "kapcsolat.php";
header('Content-Type: application/json');

// Csak DELETE metódust engedek ezen az API-n keresztül.
if ($_SERVER['REQUEST_METHOD'] !== 'DELETE') {
    http_response_code(405);
    echo json_encode(['error' => 'Csak DELETE metódussal érhető el ez a
végpont.']);
    exit;
}

// Ellenőrzöm, hogy van-e 'id' megadva a kérésben, és szám-e.
if (!isset($_GET['id']) || !ctype_digit($_GET['id'])) {
    http_response_code(400);
    echo json_encode(['error' => 'Hiányzó vagy érvénytelen értékelés ID']);
    exit;
}

$id = (int)$_GET['id']; // Egész számmá konvertálom.

try {
    // Elvégzem az értékelés törlését az adott azonosító alapján.
    $stmt = $pdo->prepare("DELETE FROM ertekelesek WHERE id = :id");
    $stmt->execute([':id' => $id]);

    // Sikeres törlés után visszajelzést küldök JSON-ben.
    echo json_encode(['status' => 'Értékelés törölve']);
} catch (PDOException $e) {
    // Ha bármilyen adatbázishiba történik, azt is lekezelem.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba: ' . $e->getMessage()]);
}
```

Postman tesztelés



MariaDB ellenőrzés

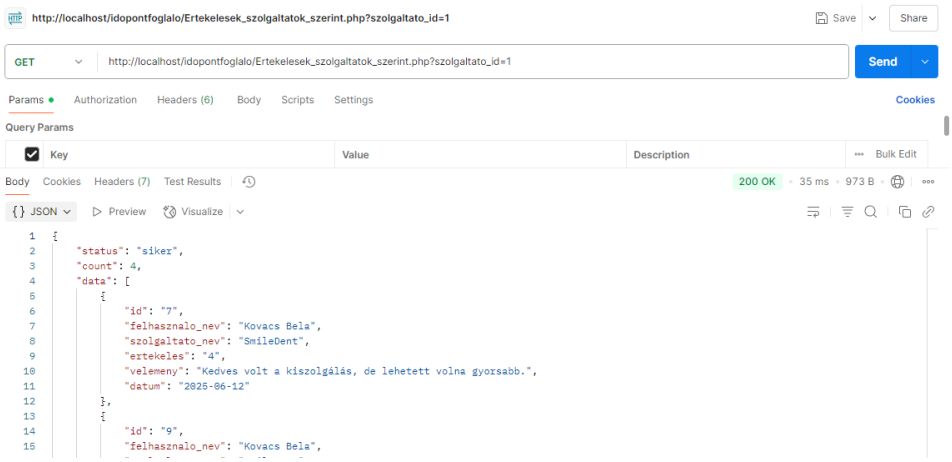
```
MariaDB [idopontfoglalo]> SELECT * FROM Ertekelesek;
+----+--------------+-----------+-----------------------------------------------------+------------+
| id | foglalasok_id | ertekeles | velemeny                                            | datum      |
+----+--------------+-----------+-----------------------------------------------------+------------+
|  6 |            7 |         5 | Nagyon elégedett voltam a szolgáltatással.          | 2025-06-12 |
|  7 |            8 |         4 | Kedves volt a kiszolgálás, de lehetett volna gyorsabb. | 2025-06-12 |
|  9 |           10 |         2 | Sajnos a szolgáltató késett.                        | 2025-06-12 |
| 10 |           12 |         1 | Nem azt kaptam, amit vártam.                        | 2025-06-12 |
+----+--------------+-----------+-----------------------------------------------------+------------+
4 rows in set (0.001 sec)
```

Értékelés szolgáltatók szerinti listázása.
Forrásfájl

```
<?php
require_once "kapcsolat.php"; // Betöltöm az adatbáziskapcsolatot biztosító fájlt.
header('Content-Type: application/json'); // Beállítom a JSON típusú válaszküldést.

// Ellenőrzöm, hogy a kérés GET típusú-e.
if ($_SERVER['REQUEST_METHOD'] !== 'GET') {
    http_response_code(405);
    echo json_encode(['error' => 'Csak GET kérés engedélyezett.']);
    exit;
}

// Megnézem, hogy a szolgáltató ID paraméter helyesen van-e megadva (szám és létezik).
if (!isset($_GET['szolgaltato_id']) || !ctype_digit($_GET['szolgaltato_id'])) {
    http_response_code(400);
    echo json_encode(['error' => 'Érvénytelen vagy hiányzó szolgáltató ID.']);
    exit;
}

$szolgaltatoId = (int)$_GET['szolgaltato_id']; // Átalakítom a bemenetet egész számmá.

try {
```

```
    // Lekérdezem az adott szolgáltatóhoz tartozó értékeléseket, a felhasználó és
szolgáltató nevével együtt.
    $stmt = $pdo->prepare("
        SELECT
            e.id,
            f.nev AS felhasznalo_nev,
            sz.nev AS szolgaltato_nev,
            e.ertekeles,
            e.velemeny,
            e.datum
        FROM ertekelesek e
        JOIN foglalasok fg ON e.foglalasok_id = fg.id
        JOIN felhasznalok f ON fg.felhasznalok_id = f.id
        JOIN idopontok i ON fg.idopontok_id = i.id
        JOIN szolgaltatok sz ON i.szolgaltatok_id = sz.id
        WHERE sz.id = :szid
        ORDER BY e.datum DESC
    ");
    $stmt->execute([':szid' => $szolgaltatoId]); // Futtatom a lekérdezést a
paraméterrel.
    $adatok = $stmt->fetchAll(); // Lekérem az összes találatot.

    // Visszaküldöm az adatokat JSON formátumban.
    echo json_encode([
        'status' => 'siker',
        'count'  => count($adatok),
        'data'   => $adatok
    ]);
} catch (PDOException $e) {
    // Hiba esetén visszajelzést küldök JSON-ben.
    http_response_code(500);
    echo json_encode(['error' => 'Adatbázis hiba: ' . $e->getMessage()]);
}
```

Postman tesztelés



MariaDB ellenőrzés

```
MariaDB [idopontfoglalo]> SELECT * FROM Ertekelesek;
+----+---------------+-----------+-----------------------------------------------------+------------+
| id | foglalasok_id | ertekeles | velemeny                                            | datum      |
+----+---------------+-----------+-----------------------------------------------------+------------+
|  6 |             7 |         5 | Nagyon elégedett voltam a szolgáltatással.          | 2025-06-12 |
|  7 |             8 |         4 | Kedves volt a kiszolgálás, de lehetett volna gyorsabb. | 2025-06-12 |
|  9 |            10 |         2 | Sajnos a szolgáltató késett.                        | 2025-06-12 |
| 10 |            12 |         1 | Nem azt kaptam, amit vártam.                        | 2025-06-12 |
| 13 |             8 |         5 | Nagyon elégedett voltam!                            | 2025-06-12 |
+----+---------------+-----------+-----------------------------------------------------+------------+
5 rows in set (0.001 sec)
```

1.10. Jogosultságkezelés, és admin funkciók

Ebben a feladatban részletesen bemutatom, hogy miként valósítottam meg az egyes PHP API-végpontokhoz tartozó szerepköralapú hozzáférés-ellenőrzést. A rendszerben három típusú jogosultságot különítettem el: „felhasznalo”, „admin” és „mindkettő”. Ezeket minden érintett fájlhoz egy külön táblázatban rendeltem hozzá, így könnyen nyomon követhető, hogy mely végpontot ki érhet el. A dokumentum tartalmazza a jogosultságkezelés technikai megvalósítását is: a session_start() indítását, a $_SESSION['szerepkor'] vizsgálatát, valamint a megfelelő HTTP válaszkódok és JSON-válaszok küldését jogosulatlan hozzáférés esetén. A rendszer így védelmet nyújt a nem hitelesített felhasználók ellen, és biztosítja, hogy csak a megfelelő szerepkörrel rendelkező személyek hajthassanak végre érzékeny műveleteket, például törlést, módosítást vagy új szolgáltatók létrehozását.

SORSZÁM	FÁJLNÉV	JOGOSULTSÁG
1.	bejelentkezes.php	Mindkettő
2.	regisztracio.php	Mindkettő
3.	kapcsolat.php	Mindkettő
4.	teszt.php	Mindkettő
5.	Idopontok_listazasa.php	Mindkettő
6.	Ertekelesek_listazasa.php	Mindkettő
7.	Szolgaltatok_listazasa.php	Mindkettő
8.	Ertekelesek_szolgaltatok_szerint.php	Mindkettő
9.	Szolgaltatok_letrehozasa.php	Admin
10.	Szolgaltatok_modositasa.php	Admin
11.	Szolgaltatok_torlese.php	Admin
12.	Idopontok _letrehozasa.php	Admin
13.	Idopontok _modositasa.php	Admin
14.	Idopontok _torlese.php	Admin
15.	Ertekelesek_torlese.php	Admin
16.	Foglalasok_letrehozasa.php	Felhasználó
17.	Foglalasaim_listazasa.php	Felhasználó
18.	Foglalasok_modositasa.php	Felhasználó
19.	Foglalasok_torlese.php	Felhasználó
20.	Ertekelesek_letrehozasa.php	Felhasználó

Jogosultságkezelés: Felhasználó

```
<?php
require_once "kapcsolat.php";
session_start();

if (!isset($_SESSION['szerepkor'])) {
    http_response_code(401);
    echo json_encode(['error' => 'Bejelentkezés szükséges.']);
    exit;
}

$engedelyezettSzerepkorok = ['felhasznalo'];

if (!in_array($_SESSION['szerepkor'], $engedelyezettSzerepkorok)) {
    http_response_code(403);
    echo json_encode(['error' => 'Nincs jogosultság.']);
    exit;
}
```

Jogosultságkezelés: Admin

```
<?php
require_once "kapcsolat.php";
session_start();

if (!isset($_SESSION['szerepkor'])) {
    http_response_code(401);
    echo json_encode(['error' => 'Bejelentkezés szükséges.']);
    exit;
}

$engedelyezettSzerepkorok = ['admin'];

if (!in_array($_SESSION['szerepkor'], $engedelyezettSzerepkorok)) {
    http_response_code(403);
    echo json_encode(['error' => 'Nincs jogosultság.']);
    exit;
}
```