

Szakképesítés neve:

Szoftverfejlesztő és -tesztelő

VIZSGAREMEK

Kalória számláló

Lipák Tibor

Vékony Márk, Pletser Norbi, Kerti Alex

Konzulens

13. A

Tartalomjegyzék

Az oldal bemutatása	4
Téma választás	4
A weboldal funkciója	4
Az oldal jövője	4
Felhasználói felület bemutatása	5
Főoldal	5
Profil	5
Regisztrálás	6
Bejelentkezés	7
Bejelentkezés nélküli használat	7
Felhasználói felület fejlesztői dokumentumok	8
Adatbázis-kapcsolat létrehozása PHP nyelven	8
Felhasználói regisztráció kezelése	9
Felhasználói bejelentkezés kezelése	10
Egyszer használatos kód ellenőrzése	11
Adatbázis-kezelő osztály	12
Étkezések lekérdezése cél és allergének alapján	13
JavaScript funkciók	15
Étkezési terv generálása (meals.js) JavaScript funkciók	16
Stíluslap (CSS)	19
Főoldal (index.php)	22
Adatbázis	28
Források és segédeszközök	28
Források (néhány étel képek):	28
Használt programok	28

Ábrajegyzék

1. ábra Főoldal	5
2. ábra profil egésze	5
3. ábra profil adatokkal	6
4. ábra Regisztrációs felület	7
5. ábra Bejelentkezési felület	7
6. ábra Bejelentkezés nélkül	8
7. ábra Adatbázis kapcsolat	8
8. ábra Regisztráció kódja	9
9. ábra Bejelentkezés kódja	10
10. ábra Verifikációs kód	11
11. ábra Adatbázis kezelés	12
12. ábra Adatbekérés az étkezésről 1.	13
13. ábra Adatbekérés az étkezésről 2.	13
14. ábra Adatbekérés az étkezésről 3.	14
15. ábra Kalória kiszámoló program 1.	15
16. ábra Kalória kiszámoló program 2.	15
17. ábra Ételek szűrése	17
18. ábra Adatfrissítések	17
19. ábra Étkezési terv megjelenítése	18
20. ábra Ételek igazítása a kalóriákhoz	18
21. ábra Kezdőfelület újra	19
22. ábra css alap stílusok	20
23. ábra css fejléc	20
24. ábra css profil	21
25. ábra css animációk	22
26. ábra Fejléc helyezkedés	23
27. ábra Regisztrációs űrlap	24
28. ábra Étkezési terv generálása	24
29. ábra Dátum frissítése	25
30. ábra Napokhoz illő dátum kiszámítása	25
31. ábra mealForm különhelyezése	27
32. ábra Adatbázis táblák kapcsolati rajza	28

Az oldal bemutatása

Téma választás

A vizsgaremek témájának azt tűztük ki célul, hogy egy edzésterv, kalória számlálós (KalCal) weboldalt készítsünk. Sok hasonló ilyen oldal létezik, mi csak szeretnénk volna megérteni, hogy pontosan ezek hogyan működnek, és szeretnénk volna készíteni egy hozzájuk hasonlót.

A weboldal funkciója

A KalCal egy olyan weboldal, ahol meg tudod nézni a kiválasztott edzéshez vagy fogyáshoz a teendőket és a folyamatokat. Az oldalra lehet regisztrálni, ezen keresztül az adatainkat elmenthetjük a folytonos edzés megkönnyítéséhez. De persze lehetséges adatok elmentése nélkül is kiszámolni dolgokat, bár alap információkat muszáj megadni a számolásokhoz, hogy sikerüljön (Testsúly, magasság, életkor stb.). A regisztrált felhasználók a 'profil' gomb segítségével a feltöltött adatokat (pl.: súlytörténet) itt lehet módosítani és nyomon követni a saját változásokat. Napról napra le tudod követni hogy milyen változásokon mentél keresztül az esetleges edzésed során

Admin felhasználók megtudják nézni a raktár jelenlegi állapotát, tudják szerkeszteni a benne szereplő adatokat (alapanyag neve, raktárban lévő mennyiség) és tudnak új alapanyagokat is felvenni. Megtudják nézni a jelenleg a rendszerben lévő pizzák adatait és tudnak és tudják szerkeszteni őket, új fajta pizzákat feltölteni, szezonális pizzákat inaktívvá tenni. A jelenleg folyamatban lévő rendeléseket a rendelések oldalon megtudják tekinteni és szükség esetén módosítani vagy törölni. Mindezeket mellett a statisztikák oldalon az elmúlt időszak rendelési statisztikáit lehet nyomon követni táblázatos és grafikonos formában.

Az oldal jövője

A csoport minden tagja úgy döntött, hogy a jövőben is szeretnénk foglalkozni az oldallal, ezért nagyon sok tervünk van a jövőre tekintve például: meg szeretnénk csinálni applikációs felületre is, szeretnénk majd bővíteni a funkciókat, hogy minél több mindent el tudj érni az oldal + applikáció segítségével. A célunk egy működő oldal fejlesztése, amit, ha esetleg én is belekezek az edzésbe, akkor szívesen használnék.

Fejlesztés alatt

A felhasználói élmény javítása és a közösségi funkciók bővítése érdekében az alábbi fejlesztések bevezetését terveztük:

Elfelejtett jelszó funkció

A bejelentkezési felületen megjelent volna egy „*Elfelejtettem a jelszót*” opció, amely lehetővé tette volna a felhasználók számára, hogy email-címen keresztül új jelszót hozzanak létre. Ez a funkció segítette volna a gyors és egyszerű jelszó-visszaállítást, ezzel növelve a rendszer használhatóságát.

Újdonságok és hírek megjelenítése

Szerettük volna lehetővé tenni kisebb újdonságok, érdekességek, tippek, illetve esetleges tévhitiek közlését, amelyek az edzéssel vagy egészséges életmóddal kapcsolatosak. Ezen kívül különböző akciók, promóciók is megjelenhettek volna, hogy a felhasználók mindig naprakészek legyenek.

Közösségi kommunikációs felület

Tervben volt egy központi oldal kialakítása, ahol a felhasználók egymással kommunikálhatnak. Ezen a felületen lehetőség lett volna posztolni, hozzászólni, illetve lájkolni mások bejegyzéseit, ezzel is ösztönözve a közösségi interakciót és támogatást.

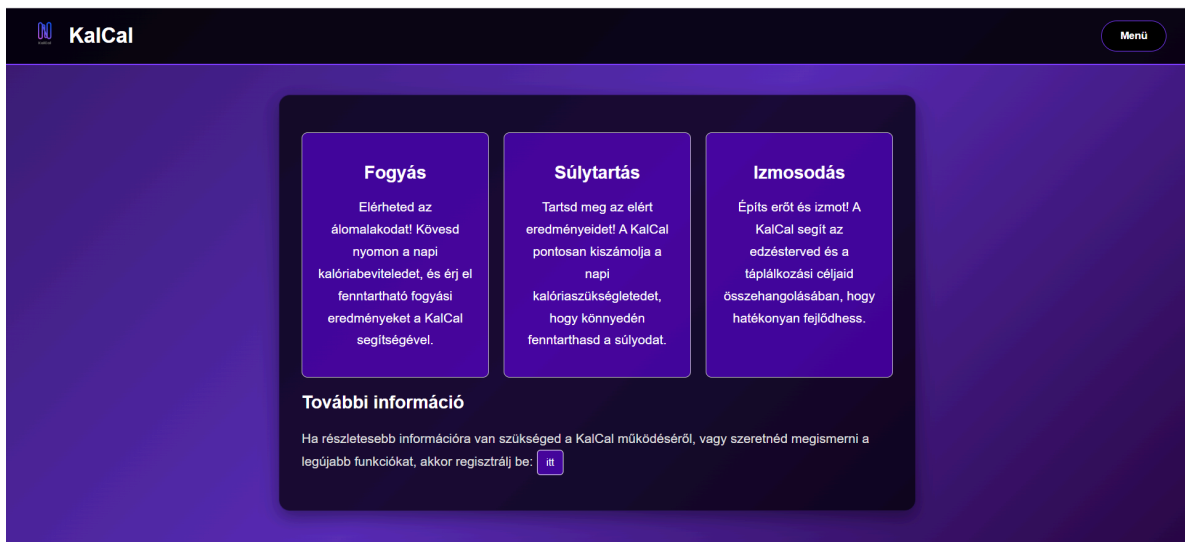
Hibabejelentő (Bug Report) rendszer

Szerettünk volna létrehozni egy egyszerűen használható hibabejelentő rendszert, ahol a felhasználók jelezhetik, ha valamilyen funkció nem működik megfelelően, illetve javaslatokat is küldhetnek a fejlesztéshez. Ez segítette volna a gyors hibakezelést és a felhasználói visszajelzések beépítését a jövőbeli fejlesztésekbe.

Felhasználói felület bemutatása

Főoldal

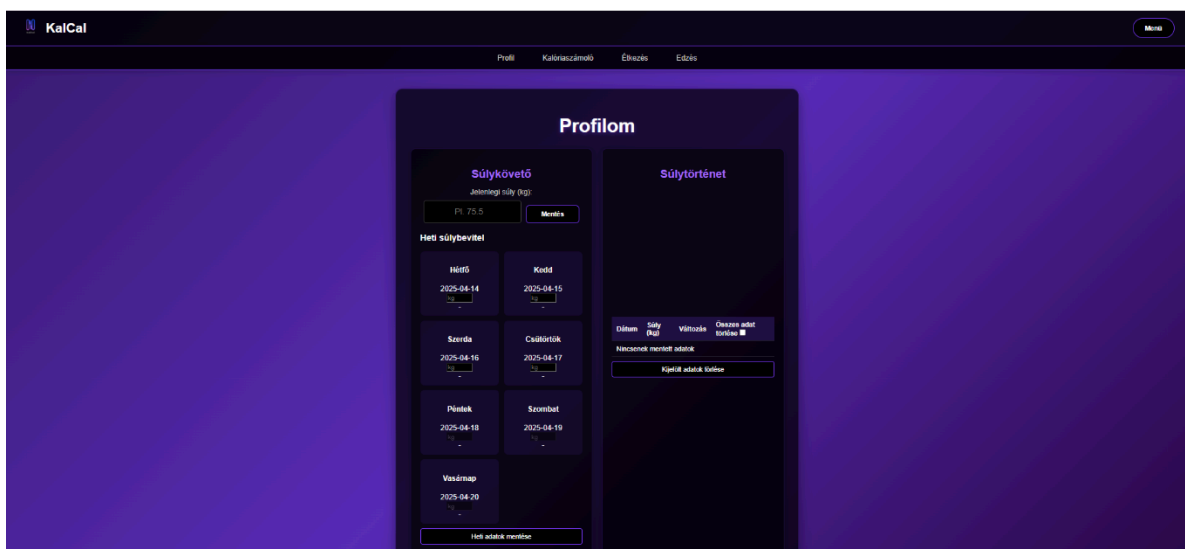
A főoldalon 3 kis leírás található a fogyásról, súlytartásról és az izmosodásról, amely segítségével vonzzuk be a kezdő edzőket. Nyilván nem csak kezdők használhatják az oldalt. Van egy gomb is 'itt' néven, ahol lehet már regisztrálni az oldalra.



1. ábra Főoldal

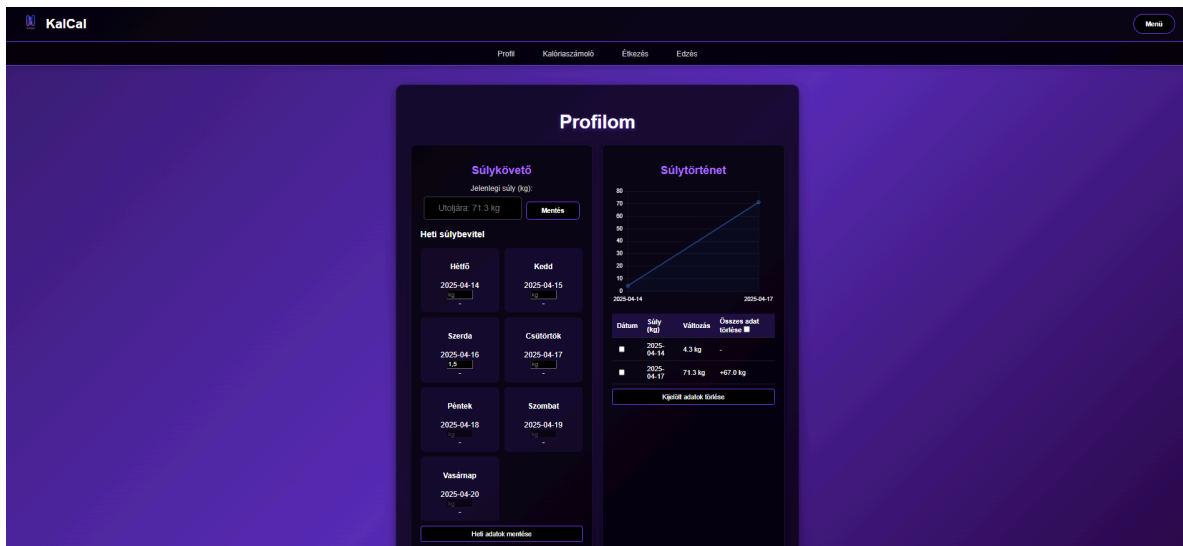
Profil

Profil oldalt már csak bejelentkezett felhasználók tudják elérni. Itt meg tudják nézni az adataikat, be tudnak írni úgymond egy hetijelentést, hogy mennyi súlyt szereztek naponta.



2. ábra profil egésze

Lehetőség van rá, hogy beírd a jelenlegi súlyodat és elmentsd. Alatta van egy olyan felület, ahol be tudod írni, hogy azon a napon mennyi súlyt vittél be vagy esetleg mennyi súlyt veszítettél (ami a második ábrán látható). Utána jobb oldalon megjelenik az elmentett adatod, ahol a súlytörténeted látszik.



3. ábra profil adatokkal

Amint elmentetted a saját magad által megadott adataidat jobb oldalon a súlytörténet funkciónál megjelenik egy diagramm, ami megmutatja neked, hogy mennyi időn belül, milyen változáson mentél át. Ez a funkció megkönnyíti az átláthatóságot és jobban ösztönözheti a felhasználót a további használatokhoz és az edzés folytatásához.

Regisztrálás

Regisztrálás létfontosságú manapság szinte minden weboldalon. Ez a mi weboldalunkon sincs másként. Bármelyik felhasználó tud regisztrálni egy olyan email-el, ami még nem lett felhasználva regisztráláshoz, és persze az sem árt, ha létező email-t adunk

meg. Ahhoz, hogy egy felhasználót „eltárolhassunk” az adatbázisban: ehhez szükség van legalább 4 információra:

- Felhasználónév
- Email
- Jelszó
- Jelszó megerősítése

4. ábra Regisztrációs felület

Ez a 4 adat nélkül nem lehetséges regisztrálni. Amint megadtuk az adatokat, és rákattintottunk a 'Regisztráció' gombra, a felhasználó azonnal át lesz irányítva a bejelentkezés felületre, ahol egyszerűen már csak be kell jelentkezzen.

Emailes megerősítés

A regisztráció folyamatát biztonságosabbá tettük azért, hogy a felhasználó a regisztráció során emailben kap egy megerősítő kódot. A kód megadása után tudja véglegesíteni regisztrációját, így csak valós email-címmel rendelkező felhasználók tudnak belépni a rendszerbe.

Bejelentkezés

Miután regisztrált a felhasználó, ezen a bejelentkező részen találja magát. Itt meg kell adnia a regisztráláskor megadott információkat. Ha mindent helyesen ad meg, akkor a 'Bejelentkezés' gombot megnyomva a főoldalra fogja irányítani a program.

5. ábra Bejelentkezési felület

Bejelentkezés nélküli használat

Ennél a funkciónál nem kell semmi 'hivatalos' adatot megadni csak, ami kell ahhoz, hogy ki tudjuk számolni a kalória szükségletet. Ehhez 7 adatot kell megadni:

- Testsúly
- Magasság
- Életkor
- Nem
- Aktivitási szint
- Cél
- Időtartam

Ezek adatok megadásával már generálja is, hogy mennyi kalóriát kell bevigyél naponta.

KalCal

Menu

Napi Kalóriaszükséglet Számító

Testsúly (kg):
75

Magasság (cm):
175

Életkor (év):
30

Nem:
Férfi

Aktivitási szint:
Mennyei aktivitás - 1.60 életmód

Cél:
Súlytartás

Időtartam (hónap):
3

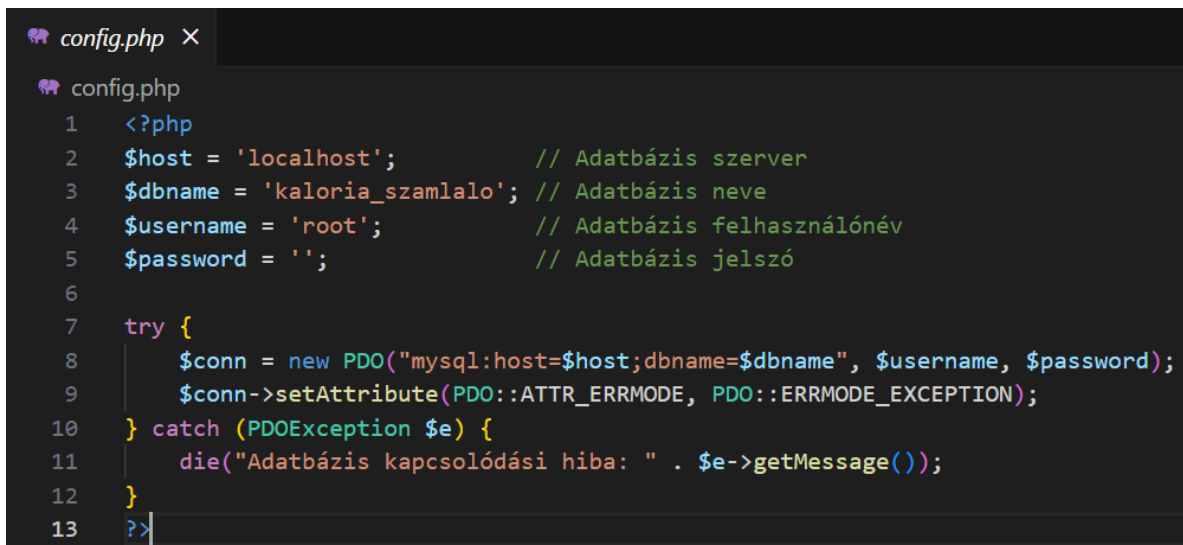
Számítás

Napi kalóriaszükséglet: - kcal

6. ábra Bejelentkezés nélkül

Felhasználói felület fejlesztői dokumentumok

Adatbázis-kapcsolat létrehozása PHP nyelven



```
config.php x
config.php
1  <?php
2  $host = 'localhost';           // Adatbázis szerver
3  $dbname = 'kaloria_szamlalo';  // Adatbázis neve
4  $username = 'root';           // Adatbázis felhasználónév
5  $password = '';               // Adatbázis jelszó
6
7  try {
8      $conn = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
9      $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
10 } catch (PDOException $e) {
11     die("Adatbázis kapcsolódási hiba: " . $e->getMessage());
12 }
13 ?>
```

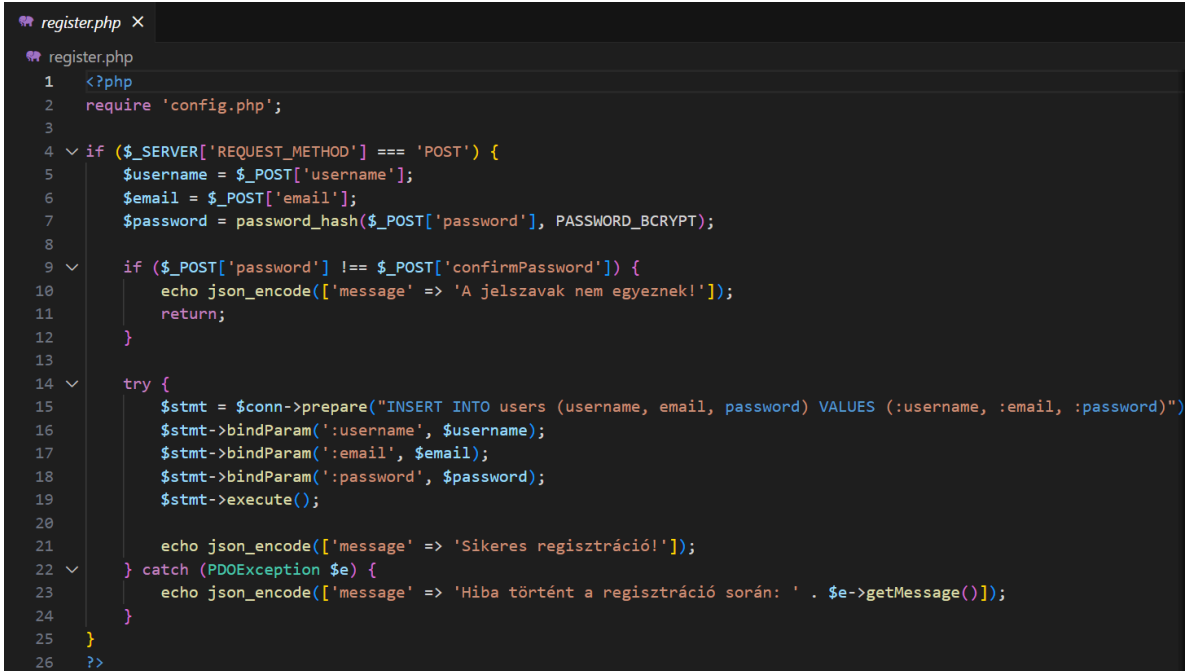
7. ábra Adatbázis kapcsolat

Ebben a szakaszban négy változóban tároljuk az adatbázis-kapcsolathoz szükséges információkat:

- `$host`: Az adatbázis szerver címe. Lokális szerver esetén ez általában `localhost`.
- `$dbname`: Az adatbázis neve, amelyhez csatlakozni szeretnénk. Jelen esetben ez a `kaloria_szamlalo`.
- `$username` és `$password`: Az adatbázishoz való hozzáféréshez szükséges felhasználónév és jelszó. Lokális környezetben (például XAMPP esetén) a `root` felhasználó jelszó nélkül is használható.
- TRY, CATCH: Ez a hibakezelő szerkezet biztosítja, hogy ha a kapcsolat létrehozása során hiba történik, azt megfelelően lekezeljük, és ne álljon le hibaüzenettel a teljes program.
- A `new PDO`: létrehozza az adatbáziskapcsolatot a megadott paraméterekkel. A kapcsolat objektumát a `$conn` változó tárolja.
- A `setAttribute`: gondoskodik arról, hogy a későbbi adatbázis-műveletek során fellépő hibák kivétel formájában kerüljenek visszaadásra. Ez átláthatóbb hibakezelést tesz lehetővé.

Ha a kapcsolat nem jön létre, a `catch` ág fut le, amely egy hibaüzenettel megszakítja a programot. Az üzenet a kivétel (`$e`) szövegét is tartalmazza a pontosabb hibakeresés érdekében.

Felhasználói regisztráció kezelése



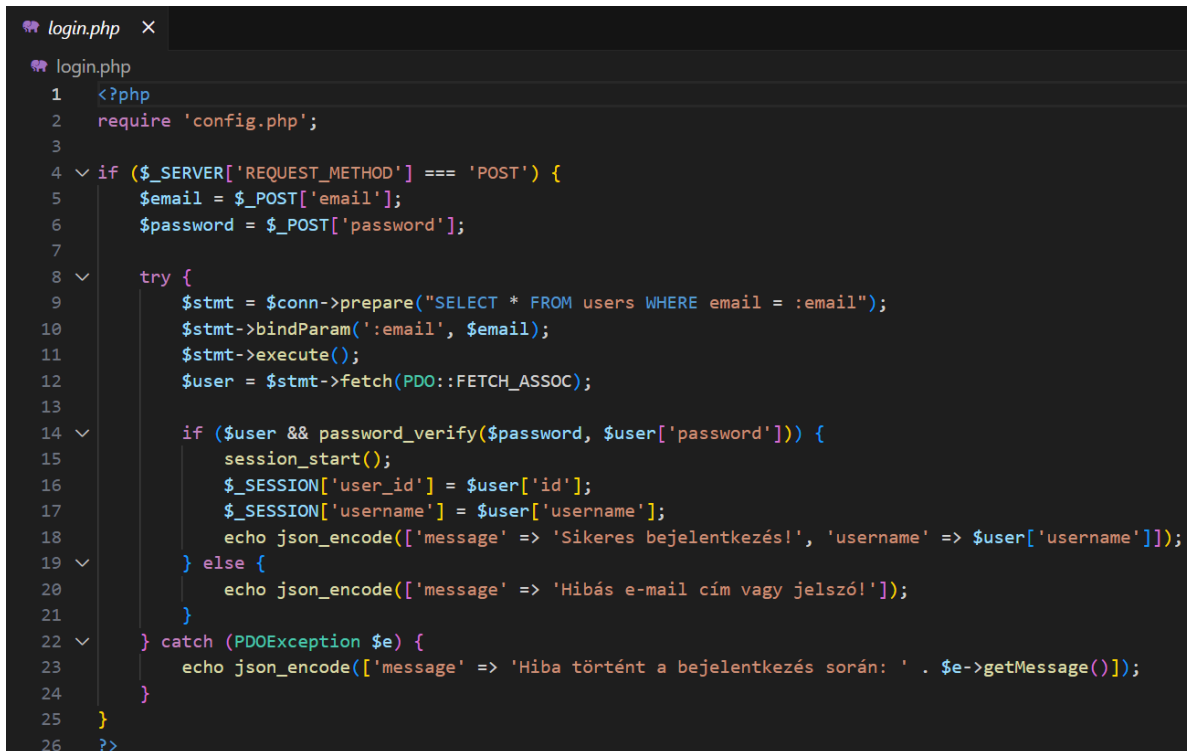
```
1 <?php
2 require 'config.php';
3
4 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
5     $username = $_POST['username'];
6     $email = $_POST['email'];
7     $password = password_hash($_POST['password'], PASSWORD_BCRYPT);
8
9     if ($_POST['password'] !== $_POST['confirmPassword']) {
10         echo json_encode(['message' => 'A jelszavak nem egyeznek!']);
11         return;
12     }
13
14     try {
15         $stmt = $conn->prepare("INSERT INTO users (username, email, password) VALUES (:username, :email, :password)");
16         $stmt->bindParam(':username', $username);
17         $stmt->bindParam(':email', $email);
18         $stmt->bindParam(':password', $password);
19         $stmt->execute();
20
21         echo json_encode(['message' => 'Sikeres regisztráció!']);
22     } catch (PDOException $e) {
23         echo json_encode(['message' => 'Hiba történt a regisztráció során: ' . $e->getMessage()]);
24     }
25 }
26 ?>
```

8. ábra Regisztráció kódja

Ez a PHP szkript a felhasználói regisztráció adatainak feldolgozásáért felel. A config.php fájl betöltésével létrejön az adatbázis-kapcsolat.

- POST kérés ellenőrzése: A művelet csak POST módszerrel indítható el.
- Űrlapadatok beolvasása: A username, email és password értékek az űrlapból kerülnek kiolvasásra. A jelszót password_hash segítségével titkosítjuk.
- Jelszó megerősítés ellenőrzése: Ha a megadott jelszavak nem egyeznek, a regisztráció megszakad.
- Adatok mentése az adatbázisba: A program előkészített SQL-lekérdezéssel beszúrja az új felhasználó adatait a users táblába, paraméterezett lekérdezéssel (SQL injection ellen védve).
- Visszajelzés: Siker vagy hiba esetén a válasz JSON formátumban kerül visszaküldésre a kliens felé.

Felhasználói bejelentkezés kezelése



```
login.php
1 <?php
2 require 'config.php';
3
4 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
5     $email = $_POST['email'];
6     $password = $_POST['password'];
7
8     try {
9         $stmt = $conn->prepare("SELECT * FROM users WHERE email = :email");
10        $stmt->bindParam(':email', $email);
11        $stmt->execute();
12        $user = $stmt->fetch(PDO::FETCH_ASSOC);
13
14        if ($user && password_verify($password, $user['password'])) {
15            session_start();
16            $_SESSION['user_id'] = $user['id'];
17            $_SESSION['username'] = $user['username'];
18            echo json_encode(['message' => 'Sikeres bejelentkezés!', 'username' => $user['username']]);
19        } else {
20            echo json_encode(['message' => 'Hibás e-mail cím vagy jelszó!']);
21        }
22    } catch (PDOException $e) {
23        echo json_encode(['message' => 'Hiba történt a bejelentkezés során: ' . $e->getMessage()]);
24    }
25 }
26 ?>
```

9. ábra Bejelentkezés kódja

Ez a szkript a felhasználói bejelentkezést valósítja meg POST kérés alapján, adatbázis ellenőrzéssel.

- Adatbázis-kapcsolat: A config.php betöltésével létrejön az adatbázis-kapcsolat.
- POST kérés feldolgozása: A beküldött email és password változók beolvasásra kerülnek.
- Felhasználó keresése: SQL lekérdezéssel lekérjük a megadott email-címhez tartozó felhasználót a users táblából.
- Jelszó ellenőrzése: A password_verify() függvény ellenőrzi, hogy a megadott jelszó megfelel-e az adatbázisban tárolt (hashelt) jelszónak.
- Sikeres bejelentkezés: Ha az ellenőrzés sikeres, elindul egy munkamenet (session_start()), és a felhasználó adatai elmentésre kerülnek a munkamenetben. A válasz JSON formátumban tartalmazza a sikeres bejelentkezés üzenetét és a felhasználónevet.
- Hibakezelés: Hibás belépési adatok vagy adatbázis-hiba esetén a válaszban hibaüzenet jelenik meg JSON formában

Admin felület

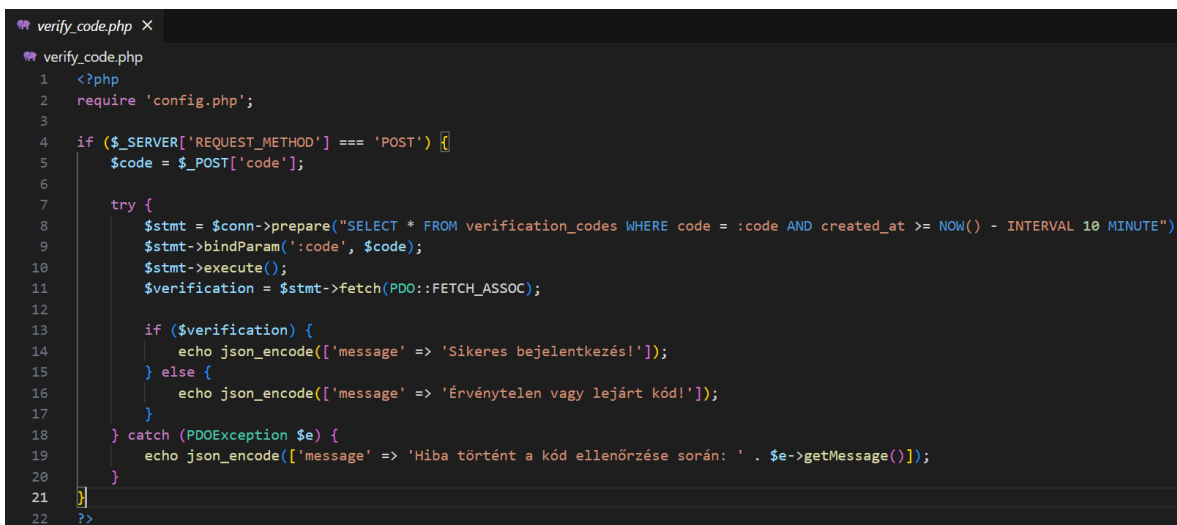
Létrehoztunk egy kizárólag fejlesztők által elérhető adminisztrációs felületet, ahol a regisztrált felhasználók adatai átláthatóan nyomon követhetők. Az admin felületen megtekinthető:

- a felhasználó email-címe,

- a választott felhasználónév,
- a regisztráció pontos időpontja.

Ezen kívül a fejlesztőknek lehetősége van a felhasználói fiókok törlésére is, amennyiben ez szükséges. Ez a funkció segíti a rendszer biztonságát, karbantartását és az esetleges visszaélések megelőzését.

Egyszer használatos kód ellenőrzése



```

1  <?php
2  require 'config.php';
3
4  if ($_SERVER['REQUEST_METHOD'] === 'POST') {
5      $code = $_POST['code'];
6
7      try {
8          $stmt = $conn->prepare("SELECT * FROM verification_codes WHERE code = :code AND created_at >= NOW() - INTERVAL 10 MINUTE");
9          $stmt->bindParam(':code', $code);
10         $stmt->execute();
11         $verification = $stmt->fetch(PDO::FETCH_ASSOC);
12
13         if ($verification) {
14             echo json_encode(['message' => 'Sikeres bejelentkezés!']);
15         } else {
16             echo json_encode(['message' => 'Érvénytelen vagy lejárt kód!']);
17         }
18     } catch (PDOException $e) {
19         echo json_encode(['message' => 'Hiba történt a kód ellenőrzése során: ' . $e->getMessage()]);
20     }
21 }
22

```

10. ábra Verifikációs kód

Ez a szkript egy időkorlátos, egyszer használatos belépési kód (pl. emailben küldött vagy 2FA) érvényességét ellenőrzi.

- Adatbázis-kapcsolat: A config.php fájl betöltésével létrejön a kapcsolat az adatbázissal.
- POST kérés fogadása: A szerver csak akkor dolgozza fel a kérelmet, ha POST metódussal érkezik.
- Kód ellenőrzése: A megadott code változó alapján lekérdezi az adatbázisból (verification_codes tábla), hogy létezik-e ilyen kód, amely az elmúlt 10 percben lett létrehozva.
- Érvényesség vizsgálata:
 1. Ha talál ilyen kódot: sikeres bejelentkezés üzenet (Sikeres bejelentkezés!).
 2. Ha nem található vagy lejárt: hibaüzenet (Érvénytelen vagy lejárt kód!).
- Hibakezelés: Az esetleges adatbázis-hibák kivételkezeléssel történnek, és JSON formátumú hibaüzenetként kerülnek visszaküldésre.

Adatbázis-kezelő osztály



```
dbvezerlo.php ×
dbvezerlo.php
1  <?php
2  class DBVezerlo {
3      private $pdo;
4
5      public function __construct() {
6          $this->pdo = new PDO('mysql:host=localhost;dbname=kaloria_szahlalo', 'root', '');
7          $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
8      }
9
10     public function executeSelectQuery($query, $params) {
11         $stmt = $this->pdo->prepare($query);
12         $stmt->execute($params);
13         return $stmt->fetchAll(PDO::FETCH_ASSOC);
14     }
15 }
16 ?>
```

11. ábra Adatbázis kezelés

Ez a PHP osztály egy egyszerű adatbázis-kezelő vezérlő, amely lehetővé teszi biztonságos SELECT lekérdezések végrehajtását PDO segítségével.

1. Kapcsolat létrehozása (__construct) Az osztály példányosításakor automatikusan létrejön a kapcsolat a kaloria_szahlalo nevű adatbázissal, a localhost szerveren, root felhasználóval. A PDO hibakezelési módja kivételkezelésre van állítva, ami átláthatóbb hibakezelést biztosít.
2. Lekérdezés futtatása (executeSelectQuery) Ez a metódus paraméterezett SQL SELECT lekérdezést hajt végre:
 - \$query: a végrehajtandó SQL utasítás szövege.
 - \$params: a lekérdezés paraméterei (pl. :id, :email, stb.). A lekérdezés eredményét asszociatív tömb formájában adja vissza (fetchAll(PDO::FETCH_ASSOC)).

Ez az osztály segít az adatbázis műveletek biztonságos és újrahasznosítható kezelésében, különösen olvasási (SELECT) műveleteknél.

Étkezések lekérdezése cél és allergének alapján

```
getMeals.php X
getMeals.php
1  <?php
2  header('Content-Type: application/json');
3  header("Access-Control-Allow-Origin: *");
4  header("Access-Control-Allow-Methods: POST, GET, OPTIONS");
5  header("Access-Control-Allow-Headers: Content-Type");
6
7  include_once('dbvezerlo.php');
8
9  try {
10     // Initialize the database controller
11     $dbvez = new DBVezerlo();
12
13     // Process the request
14     $input = json_decode(file_get_contents('php://input'), true);
15
16     if (!$input) {
17         echo json_encode(['error' => 'Invalid JSON input', 'raw_input' => file_get_contents('php://input')]);
18         exit;
19     }
20
21     $goal = $input['goal'] ?? null;
22     $allergens = $input['allergies'] ?? [];
23
24     if (!$goal) {
25         echo json_encode(['error' => 'Goal is missing', 'input' => $input]);
26         exit;
27     }
28
29     // Build the query
```

12. ábra Adatbekérés az étkezésről 1.

```
getMeals.php X
getMeals.php
9  try {
29     // Build the query
30     $query = "SELECT * FROM kaloria_szahlalo.meals WHERE goal = :goal";
31     $params = ['goal' => $goal];
32
33     // Filter by allergens (modified to work with dot-separated values)
34     if (!empty($allergens)) {
35         foreach ($allergens as $index => $allergen) {
36             $query .= " AND allergens NOT LIKE CONCAT('%', :allergen$index, '%')";
37             $params[":allergen$index"] = $allergen;
38         }
39     }
40
41     // Execute the query using DBVezerlo
42     $meals = $dbvez->executeSelectQuery($query, $params);
43
44     // Group meals by category
45     $categorizedMeals = [
46         'breakfast' => [],
47         'lunch' => [],
48         'dinner' => [],
49         'snacks' => []
50     ];
51
52     foreach ($meals as $meal) {
53         $categorizedMeals[$meal['category']][] = [
54             'id' => $meal['id'],
55             'name' => $meal['name'],
56             'image' => $meal['image'],
```

13. ábra Adatbekérés az étkezésről 2.

```

getMeals.php X
getMeals.php (mukodj)\getMeals.php (preview)
9  try {
52      foreach ($meals as $meal) {
53          $categorizedMeals[$meal['category']] = [
56              'image' => $meal['image'],
57              'recipe_link' => $meal['recipe_link'],
58              'allergens' => $meal['allergens'],
59              'calories' => $meal['calories'],
60              'protein' => $meal['protein'],
61              'carbs' => $meal['carbs'],
62              'fats' => $meal['fats'],
63              'category' => $meal['category'],
64              'goal' => $meal['goal']
65          ];
66      }
67
68      if (empty($meals)) {
69          echo json_encode([
70              'debug' => 'No meals found',
71              'query' => $query,
72              'params' => $params,
73              'meals' => $categorizedMeals // Return empty categorized structure
74          ]);
75      } else {
76          echo json_encode(['meals' => $categorizedMeals]);
77      }
78  } catch (PDOException $e) {
79      echo json_encode(['error' => $e->getMessage()]);
80  }
81  ?>

```

14. ábra Adatbekérés az étkezésről 3.

Ez az API-végpont JSON formátumban fogad POST kérést, és visszaadja a megadott célhoz (pl. fogyás, izomnövelés) illeszkedő étkezéseket, kizárva a megadott allergéneket.

1. Fejlécek beállítása: A szkript beállítja a válasz típusát JSON-ra, és engedélyezi a CORS-t (Access-Control-Allow-Origin: *), hogy más domeinekről is lehessen kérni.
2. Adatbázis-kezelő betöltése: A dbvezerlo.php fájlban definiált DBVezerlo osztály segítségével történik a biztonságos adatbázis-művelet.
3. Bemenet feldolgozása: A JSON típusú kérésből kiolvasásra kerül a goal (cél) és az allergies (allergének) mező. Ha a goal hiányzik vagy a JSON nem érvényes, hibaüzenetet küld vissza.
4. SQL lekérdezés összeállítása: A lekérdezés szűri az étkezéseket a cél szerint, és kizárja azokat, amelyek tartalmazzák a megadott allergéneket. Az allergénszűrés NOT LIKE feltételekkel történik, így rugalmasabb a keresés.
5. Eredmények kategorizálása: Az adatbázisból kapott étkezések kategóriánként (breakfast, lunch, dinner, snacks) kerülnek csoportosításra, hogy a kliens egyszerűen meg tudja jeleníteni azokat.
6. Válasz visszaadása
 - Ha található megfelelő étkezés: JSON válasz tartalmazza a kategorizált étkezéseket.
 - Ha nincs találat: üres, de strukturált válasz érkezik, hibakeresési adatokkal.
 - Hiba esetén (pl. adatbázis-hiba): a hibaüzenet JSON formátumban jelenik meg.

JavaScript funkciók

```
JS calorieCounter.js X
JS calorieCounter.js > ...
1 function handleGoalChange() {
2     const goal = document.getElementById('goal').value;
3     const targetWeightGroup = document.getElementById('targetWeightGroup');
4     const weight = parseFloat(document.getElementById('weight').value);
5     const targetWeightInput = document.getElementById('targetWeight');
6
7     if (goal === 'maintain') {
8         targetWeightGroup.style.display = 'none';
9     } else {
10        targetWeightGroup.style.display = 'block';
11        targetWeightInput.value = '';
12        targetWeightInput.placeholder = goal === 'lose' ? `Kevesebb, mint ${weight}` : `Több, mint ${weight}`;
13        targetWeightInput.setAttribute('min', goal === 'lose' ? '0' : (weight + 0.1).toFixed(1));
14        targetWeightInput.setAttribute('max', goal === 'lose' ? (weight - 0.1).toFixed(1) : '');
15    }
16}
17
18function calculateCalories() {
19    const weight = parseFloat(document.getElementById('weight').value);
20    const height = parseFloat(document.getElementById('height').value);
21    const age = parseInt(document.getElementById('age').value);
22    const gender = document.getElementById('gender').value;
23    const activity = parseFloat(document.getElementById('activity').value);
24    const goal = document.getElementById('goal').value;
25
26    let bmr;
27
28    // BMR számítás férfiakra és nőkre
29    if (gender === 'male') {
```

15. ábra Kalória kiszámoló program 1.

```
JS calorieCounter.js X
JS calorieCounter.js > ...
18 function calculateCalories() {
28     // BMR számítás férfiakra és nőkre
29     if (gender === 'male') {
30         bmr = 10 * weight + 6.25 * height - 5 * age + 5;
31     } else {
32         bmr = 10 * weight + 6.25 * height - 5 * age - 161;
33     }
34
35     // Napi kalóriaszükséglet aktivitási szint alapján
36     const dailyCalories = bmr * activity;
37
38     // let finalCalories: any
39     let finalCalories;
40     if (goal === 'lose') {
41         finalCalories = dailyCalories - 500; // Fogyás
42     } else if (goal === 'gain') {
43         finalCalories = dailyCalories + 500; // Izmosodás
44     } else {
45         finalCalories = dailyCalories; // Súlytartás
46     }
47
48     // Ellenőrzés, ha kevesebb mint 1500
49     if (isNaN(finalCalories) || finalCalories < 1500) {
50         document.getElementById('calorieResult').innerText = 'Nem lehetséges';
51         alert('A napi kalóriaszükséglet nem lehet kevesebb, mint 1500 kcal!');
52     } else {
53         document.getElementById('calorieResult').innerText = `${Math.round(finalCalories)} kcal`;
54     }
55 }
```

16. ábra Kalória kiszámoló program 2.

1. handleGoalChange() – A cél típusának kezelése

Ez a függvény a felhasználó által kiválasztott cél (fogyás, hízás vagy súlytartás) alapján dinamikusan módosítja az űrlap kinézetét és működését.

- Elrejtí a cél testsúly mezőt, ha a cél „súlytartás”.
- Megjeleníti a mezőt, ha a cél „fogyás” vagy „hízás”.
- Automatikusan beállít egy segédszöveget (placeholder), amely segíti a felhasználót a megfelelő adat megadásában.
- Értékhatárokat is meghatároz: pl. fogyásnál nem lehet célként megadni magasabb súlyt, hízásnál pedig nem lehet alacsonyabbat.

Ezzel elkerülhetők a hibás adatok megadása, növelve az űrlap használhatóságát és logikusságát.

2. calculateCalories() – Napi kalóriaszükséglet kiszámítása

Ez a függvény a felhasználó által megadott adatok alapján (testtömeg, magasság, életkor, nem, aktivitási szint) kiszámolja a napi kalóriaszükségletet, az alábbi lépésekben:

1. BMR (Basal Metabolic Rate) számítás

A nem alapján eltérő képletet használ a nyugalmi energiafelhasználás meghatározására:

- Férfiak: $BMR = 10 * \text{súly} + 6.25 * \text{magasság} - 5 * \text{életkor} + 5$
- Nők: $BMR = 10 * \text{súly} + 6.25 * \text{magasság} - 5 * \text{életkor} - 161$

2. Napi kalóriaigény meghatározása aktivitási szint alapján

Az aktivitási szorzó (pl. 1.2 ülő életmódnál, 1.55 közepes aktivitásnál) alapján:
 $\text{Napi kalóriaszükséglet} = BMR * \text{aktivitás}$

3. Cél szerinti módosítás

- Fogyásnál: csökkenti napi 500 kcal-lal
- Hízásnál: növeli napi 500 kcal-lal
- Súlytartásnál: megtartja az eredeti értéket

4. Értékel ellenőrzés

- Ha az eredmény kevesebb mint 1500 kcal, akkor hibaüzenetet ír ki (Nem lehetséges) és egy figyelmeztető ablakot is megjelenít.
- Egyébként az eredményt kerekítve megjeleníti az oldalon.

Ez a funkció gondoskodik róla, hogy a felhasználó testreszabott és életszerű ajánlást kapjon a kalóriabevitelre vonatkozóan.

Étkezési terv generálása (meals.js) JavaScript funkciók

Ez a modul lehetővé teszi az étkezési terv dinamikus generálását a felhasználó céljai és allergiái alapján. A megoldás REST API-n keresztül szerez adatokat az adatbázisból, majd ezekből a megfelelően szűrt és kategorizált ételeket jeleníti meg a felhasználói felületen.

```
function generateMealPlan(goal, allergies) {
  return fetch('getMeals.php')
    .then(response => response.json())
    .then(data => {
      // Szűrés cél és allergiák alapján
      const filteredMeals = data.filter(meal => {
        // Szűrés cél szerint
        if (meal.goal !== goal) return false;

        // Allergének szűrése
        if (allergies.length > 0) {
          const mealAllergens = meal.allergens ? meal.allergens.split(',') : [];
          return !allergies.some(allergy => mealAllergens.includes(allergy));
        }

        return true;
      });

      // Ételek kategóriák szerint csoportosítása
      const mealPlan = {
        breakfast: filteredMeals.filter(meal => meal.category === 'breakfast'),
        lunch: filteredMeals.filter(meal => meal.category === 'lunch'),
        dinner: filteredMeals.filter(meal => meal.category === 'dinner'),
        snacks: filteredMeals.filter(meal => meal.category === 'snacks'),
      };

      return mealPlan;
    })
    .catch(error => console.error('Hiba az ételek lekérésekor:', error));
}
```

17. ábra Ételek szűrése

1. generateMealPlan(goal, allergies)

Ez a függvény felelős az ételek lekéréséért és szűréséért:

- Először a getMeals.php fájlban keresztül lekéri az összes ételt.
- Ezután kiszűri azokat, amelyek **nem felelnek meg a megadott célnak** (pl. fogyás, izmosodás stb.).
- A megmaradt ételek közül további szűrést végez az **allergének alapján**, és kizárja azokat, amelyek tartalmazzák a megadott allergéneket.
- A fennmaradó ételeket **kategóriák szerint csoportosítja**: reggeli, ebéd, vacsora, snack.

Ez a függvény egy Promise-t ad vissza, amely a strukturált, kategorizált étkezési tervet (mealPlan) tartalmazza.

```
function handleMealGoalChange() {
  const goal = document.getElementById('mealGoal').value;
  // Additional logic for goal change
}
```

18. ábra Adatfrissítések

2. handleMealGoalChange()

Ez a placeholder funkció jelenleg csak az onchange eseményhez kapcsolódik. Jövőbeli bővítés esetén lehetőséget adhat például az étellista automatikus frissítésére, amikor a cél megváltozik.

```
function displayMealPlan(mealPlan) {
  const mealPlanner = document.getElementById('mealPlanner');
  mealPlanner.innerHTML = `
    <h1>Étkezési Terv</h1>
    <button class="back-button" onclick="showMealPlannerHome()">Vissza a főoldalra</button>
    <div class="meal-category">
      <h2>Reggeli</h2>
      <div class="meal-list" id="breakfastList"></div>
    </div>
    <div class="meal-category">
      <h2>Ebéd</h2>
      <div class="meal-list" id="lunchList"></div>
    </div>
    <div class="meal-category">
      <h2>Vacsora</h2>
      <div class="meal-list" id="dinnerList"></div>
    </div>
    <div class="meal-category">
      <h2>Snack</h2>
      <div class="meal-list" id="snackList"></div>
    </div>
  `;
  // Ételek hozzáadása kategóriák szerint
  addMealsToCategory(mealPlan.breakfast, 'breakfastList');
  addMealsToCategory(mealPlan.lunch, 'lunchList');
  addMealsToCategory(mealPlan.dinner, 'dinnerList');
  addMealsToCategory(mealPlan.snacks, 'snackList');
  // Az oldalsó panel megjelenítése
  document.getElementById('sidePanel').style.display = 'block';
}
```

19. ábra Étkezési terv megjelenítése

3. displayMealPlan(mealPlan)

Ez a függvény megjeleníti a kapott étkezési tervet a weboldalon:

- Törli a jelenlegi tartalmat, majd új HTML-t generál, amely négy kategóriára bontja az ételeket.
- Minden kategóriához tartozik egy div, ahová az ételek egyenként kerülnek beillesztésre az addMealsToCategory() segítségével.
- Végül megjeleníti az oldalsó panelt, amely lehetővé teszi a visszatérést a kezdőfelületre.

```
function addMealsToCategory(meals, categoryId) {
  const categoryElement = document.getElementById(categoryId);
  meals.forEach(meal => {
    const mealItem = document.createElement('div');
    mealItem.classList.add('meal-item');

    // Ellenőrizd, hogy a meal objektum tartalmazza a szükséges adatokat
    mealItem.innerHTML = `
      <h3>${meal.name}</h3>
      
      <p>Kalória: ${meal.calories || 0} kcal</p>
      <p>Fehérje: ${meal.protein || 0} g</p>
      <p>Szénhidrát: ${meal.carbs || 0} g</p>
      <p>Zsír: ${meal.fats || 0} g</p>
      <a href="${meal.recipeLink}" target="_blank">Recept megtekintése</a>
    `;

    categoryElement.appendChild(mealItem);
  });
}
```

20. ábra Ételek igazítása a kalóriákhoz

4. addMealsToCategory(meals, categoryId)

Ez a segédfüggvény hozzáadja az ételeket a megadott kategória div-jéhez:

- Dinamikusán div elemeket hoz létre, amelyek tartalmazzák az étel nevét, képét, tápanyagtartalmát és egy linket a recepthoz.
- Ellenőrzi, hogy az értékek ne legyenek null vagy undefined, és ha kell, alapértelmezett 0 értékeket használ.

```
function showMealPlannerHome() {  
    <option value="loseWeight">Fogyás</option>  
    <option value="maintainWeight">Súlytartás</option>  
    <option value="gainMuscle">Izmosodás</option>  
    </select>  
    </div>  
    <div class="form-group">  
        <label>Allergiák:</label>  
        <div>  
            <input type="checkbox" id="gluten" name="allergies" value="gluten"> Glutén<br>  
            <input type="checkbox" id="lactose" name="allergies" value="lactose"> Laktóz<br>  
            <input type="checkbox" id="nuts" name="allergies" value="nuts"> Diófélék<br>  
            <input type="checkbox" id="fish" name="allergies" value="fish"> Hal<br>  
            <input type="checkbox" id="egg" name="allergies" value="egg"> Tojás<br>  
        </div>  
    </div>  
    <div class="form-group">  
        <button type="submit">Étkezési Terv Generálása</button>  
    </div>  
    </form>  
    ;  
    document.getElementById('mealForm').addEventListener('submit', function (e) {  
        e.preventDefault();  
        const formData = new FormData(this);  
        const allergies = [];  
        formData.getAll('allergies').forEach(allergy => allergies.push(allergy));  
  
        const goal = document.getElementById('mealGoal').value;  
        generateMealPlan(goal, allergies).then(displayMealPlan);  
    });  
}
```

21. ábra Kezdőfelület újra

5. showMealPlannerHome()

Ez a függvény újra felépíti a kezdőfelületet:

- Űrlapot jelenít meg, amelyben a felhasználó kiválaszthatja a célját (goal) és az allergiáit.
- Az űrlap submit eseményét figyeli, és annak beküldésekor meghívja a generateMealPlan() függvényt a megadott adatokkal, majd az eredményt átadja a displayMealPlan()-nak.

Stíluslap (CSS)

Ez a CSS fájl felelős az alkalmazás vizuális megjelenéséért, azaz minden elem (gombok, szövegek, hátterek, menük stb.) stílusának, elrendezésének és interaktív viselkedésének meghatározásáért. A fájl célja, hogy egységes, modern, letisztult és esztétikus kinézetet biztosítson az oldal minden részének. A főbb funkciók a következők:

```

/* Alap stílusok */
body {
  font-family: Arial, sans-serif;
  background: linear-gradient(135deg, #3a1c71, #5a2db8, #2a0845);
  color: #fff;
  margin: 0;
  padding: 0;
  position: relative;
  min-height: 100vh;
}

```

22. ábra css alap stílusok

- **Alap stílusok:** Meghatározza az oldal alaphátterét (lila színátmenet), a betűtípust, és biztosítja, hogy az oldal teljes magasságú legyen.

```

/* Fejléc */
header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 15px 30px;
  background: rgba(0, 0, 0, 0.85);
  position: relative;
  z-index: 100;
  border-bottom: 2px solid #7d3cff;
}

```

23. ábra css fejléc

- **Fejléc és navigáció:** Stílusozza a felső fejléceket, a logót és a lebegő menüt, hogy jól látható, reszponzív és interaktív legyen.
- **Tartalom blokkok (konténerek):** Gondoskodik arról, hogy a fő tartalom jól elkülönüljön a háttértől, árnyékot és lekerekített sarkokat kapjon, így vizuálisan kiemelkedjen.
- **Űrlap elemek:** Egységes stílust biztosít a beviteli mezőknek, gomboknak, legördülő listáknak, és reszponzívan alkalmazkodik a különböző képernyőméretekhez.
- **Étel- és edzésterv megjelenítés:** Rácsszerű elrendezést biztosít az étel- és edzéstartalmaknak, minden elem kártyaszerűen, árnyékkal és animációval jelenik meg, így letisztult és átlátható.

```

/* Profil oldal */
.profile-container {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
}

.weight-tracker, .weight-history {
  flex: 1;
  min-width: 300px;
  background: rgba(0, 0, 0, 0.6);
  padding: 20px;
  border-radius: 10px;
  border: 1px solid rgba(125, 60, 255, 0.2);
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.3);
}

.weight-tracker h2, .weight-history h2 {
  color: #a56cff;
  margin-bottom: 15px;
  text-align: center;
}

.days-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(120px, 1fr));
  gap: 10px;
}

```

24. ábra css profil

- **Profil oldal:** Két fő szekció (súlykövetés és súlytörténet) külön kártyákban jelenik meg, modern dizájnnal, táblázattal, grafikon konténerrel és napok rácsszerű megjelenítésével.
- **Reszponzív viselkedés:** A média lekérdezések révén a weboldal minden méretű kijelzőn jól használható, legyen az telefon, tablet vagy asztali gép.

```

@media (max-width: 1024px) {
    .profile-container {
        grid-template-columns: 1fr;
    }

    .days-grid {
        grid-template-columns: repeat(4, 1fr);
    }
}

@media (max-width: 600px) {
    .days-grid {
        grid-template-columns: repeat(2, 1fr);
    }

    .weight-input-container {
        flex-direction: column;
    }
}

```

25. ábra css animációk

- **Animációk és interakciók:** Kisebb mozgások, lebegő árnyékok, színváltások és sima átmenetek segítik az interaktivitást és a felhasználói élményt.

Ez a stíluslap tehát alapvetően meghatározza az alkalmazás megjelenését és hangulatát, és biztosítja, hogy minden felület jól strukturált, modern és vizuálisan egységes legyen.

Főoldal (index.php)

Ez a PHP és HTML alapú főoldal egy tabos (füles) navigációval ellátott felhasználói felületet biztosít, amely több szekcióra osztja az alkalmazás fő funkcióit. A célja, hogy könnyen áttekinthető és egyszerűen használható módon érje el a felhasználó a profilját, étrendjét, edzéstervét és az eredményeit.

1. Session-kezelés (PHP)

A fájl elején egy `session_start()` parancs található, amely elindítja a munkamenetet, és lehetővé teszi a felhasználói adatok (pl. azonosító, felhasználónév) tárolását a munkamenet során.

2. HTML – Tabos felépítés

A weboldal HTML része egy négyfüles struktúrát definiál, amely a következő szekciókat tartalmazza:

- Profil – A felhasználó személyes adatait, fiókinformációit jelenítheti meg.
- Kaja – Itt lesz lehetőség az étrendek megtekintésére vagy kezelésére.
- Edzésterv – Az edzési programok, gyakorlatok vagy célok kezelése történik itt.
- Eredmények – Az edzés vagy étrend során elért eredmények megjelenítése történhet ebben a szekcióban.

Minden fül egy div elembe kap helyet, amely alapértelmezés szerint rejtve van, és csak akkor jelenik meg, ha a felhasználó az adott fülre kattint.

3. CSS – Alapstílusok a fülrendszerhez

Az oldal tartalmaz egy beágyazott `<style>` blokkot, amely meghatározza a tabok és tartalmak megjelenését:

- A `.tab` osztály definiálja a fülgombok konténerét.
- A `.tab button` osztály szabályozza a fülgombok kinézetét, viselkedését hover állapotban és aktívként.
- A `.tabcontent` osztály kezeli a tartalmak elrejtését és megjelenítését a megfelelő fül kiválasztásakor.

A stílus egyszerű és letisztult, mobilbarát alapelrendezés, de könnyedén továbbfejleszthető saját CSS fájlban is (`style.css`).

4. JavaScript – Tabváltás logikája

Az oldal végén található JavaScript szkript gondoskodik a tabváltás dinamikus működéséről. A `openTab(evt, tabName)` függvény:

1. Elrejtje az összes `tabcontent` osztályú tartalmat.
2. Eltávolítja az aktív osztályt az összes fülről.
3. Megjeleníti a kiválasztott fülhöz tartozó tartalmat.
4. Aktívvá teszi a megnyitott fülhöz tartozó gombot.

Az oldal betöltésekor a „Profil” fül automatikusan megnyílik, így az oldal használata intuitív.

```

index.php ×
index.php
1  <!DOCTYPE html>
2  <html lang="hu">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>KalCal - Kalóriaszámláló</title>
7      <link rel="stylesheet" href="styles.css">
8  </head>
9  <body>
10     <!-- Fejléc -->
11     <header>
12         <!-- Logo -->
13         <a href="#" class="logo" onclick="goToHome()">
14             
15             KalCal
16         </a>
17
18         <!-- Lenyíló menü -->
19         <div class="dropdown">
20             <button class="menu-button">Menü</button>
21             <div class="dropdown-content" id="menuItems">
22                 <button id="registerButton" onclick="showRegistration()">Regisztráció</button>
23                 <button id="loginButton" onclick="showLogin()">Bejelentkezés</button>
24                 <button id="guestButton" onclick="useWithoutLogin()">Folytatás bejelentkezés nélkül</button>
25                 <button id="logoutButton" onclick="logout()" style="display: none;">Kijelentkezés</button>
26             </div>
27         </div>
28     </header>

```

26. ábra Fejléc helyezkedés

Itt látható egy fejléc, aminek a bal oldalán ott van a logónk amire, ha rákattintunk, akkor az onclick segítségével a főoldalra léptet vissza. A fejléc jobb oldalán található egy lenyíló menü, ahol különböző funkciók találhatók.

```
<!-- Regisztráció űrlap -->
<div class="container" id="registration" style="display: none;">
  <h1>Regisztráció</h1>
  <form id="registerForm">
    <div class="form-group">
      <label for="regUsername">Felhasználónév:</label><br>
      <input type="text" id="regUsername" name="username" placeholder="Felhasználónév" required>
    </div>
    <div class="form-group">
      <label for="regEmail">E-mail cím:</label><br>
      <input type="email" id="regEmail" name="email" placeholder="E-mail cím" required>
    </div>
    <div class="form-group">
      <label for="regPassword">Jelszó:</label><br>
      <input type="password" id="regPassword" name="password" placeholder="Jelszó" required>
    </div>
    <div class="form-group">
      <label for="regConfirmPassword">Jelszó megerősítése:</label><br>
      <input type="password" id="regConfirmPassword" name="confirmPassword" placeholder="Jelszó megerősítése" required>
    </div>
    <div class="form-group">
      <button type="submit">Regisztráció</button>
    </div>
  </form>
</div>
```

27. ábra Regisztrációs űrlap

Itt található a regisztrációs űrlap. Elhelyeztük a bemeneti mezőket divekbe és a küldés gombot is. ezeket a style.css-ben helyeztük az oldalhoz megfelelően. Hasonlóképpen csináltuk meg a bejelentkezési űrlapot és a többi ehhez hasonló oldalakat.

```
<!-- Étkezési opciók -->
<div class="container" id="mealPlanner" style="display: none;">
  <h1>Étkezési Terv</h1>
  <form id="mealForm">
    <div class="form-group">
      <label for="mealGoal">Cél:</label>
      <select id="mealGoal" name="goal">
        <option value="loseWeight">Fogyás</option>
        <option value="maintainWeight">Súlytartás</option>
        <option value="gainMuscle">Izmosodás</option>
      </select>
    </div>
    <div class="form-group">
      <label><h1>Allergiák:</h1></label>
      <div>
        <p>Glutén </p><input type="checkbox" id="gluten" name="allergies" value="gluten">
        <p>Laktóz </p><input type="checkbox" id="lactose" name="allergies" value="lactose">
        <p>Diófélék </p><input type="checkbox" id="nuts" name="allergies" value="nuts">
        <p>Hal </p><input type="checkbox" id="fish" name="allergies" value="fish">
        <p>Tojás </p><input type="checkbox" id="egg" name="allergies" value="egg">
      </div>
    </div>
    <div class="form-group">
      <button type="submit">Étkezési Terv Generálása</button>
    </div>
  </form>
</div>
```

28. ábra Étkezési terv generálása

Ennél a lehetőségnél tudod lekérdezni a megfelelő étkezési tervet magadnak. Option szekcióban tudod kiválasztani, hogy melyik munkafolyamatot szeretnéd végezni + ki tudod választani az allergéneket, ha esetleg allergiás vagy valamire és ennek megfelelően legenerálja neked a neked illő étkezési tervet.

```
<script>
  // Profil oldal funkciók
  let weightHistory = JSON.parse(localStorage.getItem('weightHistory')) || [];
  let weekWeights = JSON.parse(localStorage.getItem('weekWeights')) || {
    monday: null, tuesday: null, wednesday: null,
    thursday: null, friday: null, saturday: null, sunday: null
  };
  let weekDates = JSON.parse(localStorage.getItem('weekDates')) || {};

  if (!window.weightChart) {
    window.weightChart = null;
  }

  function showProfile() {
    try {
      showSection('profile');
      updateWeekDates(); // Frissítjük a dátumokat minden megnyitáskor
      updateWeightDisplay();
      renderWeightHistory();
      renderWeightChart();
    } catch (error) {
      console.error('Hiba a profil megjelenítésekor:', error);
    }
  }
}
```

29. ábra Dátum frissítése

Itt egy kis javascript található, de az index.php-n belül. Itt ahogy látható a profil oldalon lévő kis részlet látszódik, ahol beállítottuk a napokat, heteket és hogy mindig frissüljön a dátum mikor belépünk.

```
function updateWeekDates() {
  const today = new Date();
  const currentDay = today.getDay(); // 0-vasárnap, 1-hétfő, ..., 6-szombat
  const currentDate = today.getDate();

  // Kiszámoljuk az aktuális hét napjainak dátumát
  const dates = {};
  const dayNames = ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday'];

  for (let i = 0; i < 7; i++) {
    const dayDiff = i - (currentDay === 0 ? 6 : currentDay - 1); // Vasárnapot a hét végére tesszük
    const date = new Date(today);
    date.setDate(currentDate + dayDiff);

    const dayElement = document.getElementById(`${dayNames[i]}Date`);
    const inputElement = document.querySelector(`.day-input[data-day="${dayNames[i]}"`);

    const dateStr = formatDate(date);
    dates[dayNames[i]] = dateStr;
    dayElement.textContent = dateStr;
  }
}
```

30. ábra Napokhoz illő dátum kiszámítása

Itt az aktuális hét napjaihoz tartozó dátumokat kiszámítja, majd ezeket megjeleníti a megfelelő HTML elemekben.

Részletes működés:

- Létrehoz egy új dátum objektumot a mai napra (today).
- Kinyeri a mai nap sorszámát (getDay()) – ez 0-tól 6-ig terjed, ahol 0 a vasárnap, 1 a hétfő stb.
- Lekéri a mai dátum napját is (getDate()), pl. 18-át.
- Egy dates nevű üres objektumban tárolja majd a hét napjaihoz tartozó dátumokat.
- Létrehoz egy dayNames tömböt, amely a hét napjait tartalmazza angolul, hétfőtől kezdve. (Ez segít a napokhoz azonosítókat rendelni.)

A ciklus (7 napra):

- Végigmegy az aktuális hét napjain (hétfőtől vasárnapig).
- A dayDiff segítségével kiszámítja, hogy hány napot kell vissza- vagy előrelepní a mai naphoz képest, hogy megkapjuk az adott naphoz tartozó dátumot.
Ha a mai nap vasárnap (currentDay === 0), akkor azt a hét végére tesszük.
- Létrehoz egy új dátum objektumot, és eltolja a napot a megfelelő irányba.
- Kikeresi az adott naphoz tartozó HTML elemeket:
dayElement: az adott naphoz tartozó dátum megjelenítéséhez.
inputElement: az adott naphoz tartozó input mező, amelynek data-day attribútuma alapján van kiválasztva.
- A dátumot formázza (feltehetően a formatDate() függvénnyel), majd:
Elmenti az adott nap nevéhez tartozó dátumot a dates objektumba.
A dátumot beírja a dayElement szövegébe.

Ez a függvény automatikusan meghatározza az aktuális hét minden napjának pontos dátumát, és feltölti vele a megfelelő HTML elemeket. A logikája külön figyelmet fordít arra, hogy a vasárnapot mindig a hét végére tegye, így a hét sorrendje mindig hétfőtől vasárnapig tart. Ez különösen hasznos heti tervek, időbeosztások, vagy étkezési naplók vizuális megjelenítéséhez.

```
// A mealForm eseménykezelőjét külön függvénybe helyezzük
function setupMealForm() {
  const mealForm = document.getElementById('mealForm');
  if (mealForm) {
    mealForm.addEventListener('submit', async function(e) {
      e.preventDefault();
      const formData = new FormData(this);
      const allergies = [];
      formData.getAll('allergies').forEach(allergy => allergies.push(allergy));
      const goal = document.getElementById('mealGoal').value;

      try {
        const mealPlan = await generateMealPlan(goal, allergies);
        displayMealPlan(mealPlan);
      } catch (error) {
        console.error('Hiba:', error);
      }
    });
  }
}
```

31. ábra mealForm különhelyezése

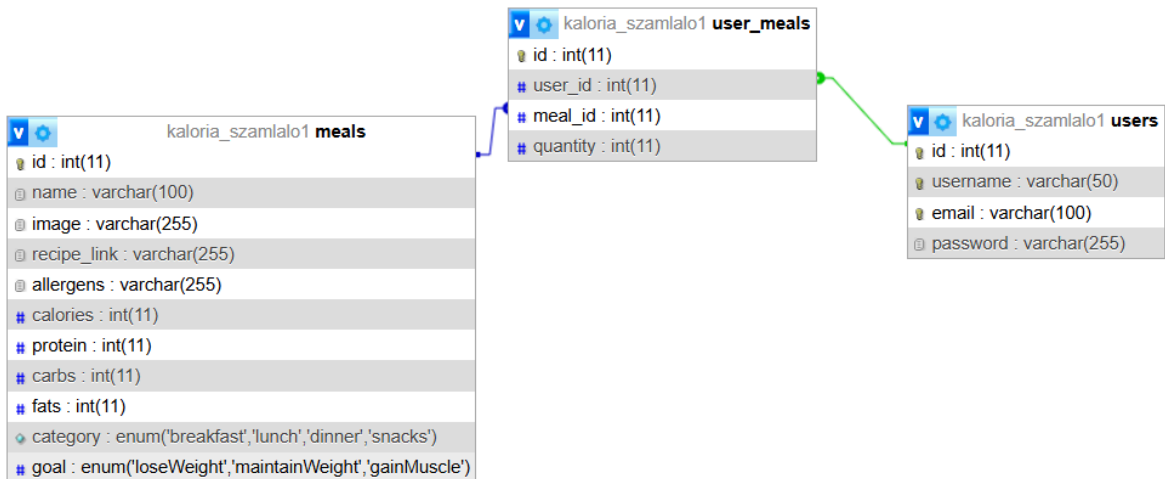
Itt beköti az eseménykezelőt a HTML-ben található mealForm nevű űrlaphoz, és gondoskodik arról, hogy a beküldés (submit) eseményre egyedi logika fusson le – ne a szokásos oldalletöltés.

Részletes működés:

- Megkeresi az oldalon a mealForm azonosítójú űrlapot, és csak akkor folytatja a működést, ha az valóban létezik.
- Hozzárendel egy submit eseménykezelőt az űrlaphoz.
- Az esemény bekövetkeztekor leállítja az alapértelmezett működést (e.preventDefault()), így az oldal nem töltődik újra.
- A FormData segítségével kiolvassa az űrlap adatait.
- Az allergies tömbbe külön kigyűjti az összes allergiát, amelyet a felhasználó megadott (pl. checkboxok esetén több is lehet).
- Beolvassa a mealGoal mező értékét, ami a felhasználó célját jelenti (pl. fogyás, izomnövelés stb.).
- Meghívja a generateMealPlan(goal, allergies) nevű aszinkron függvényt, amely egy személyre szabott étkezési tervet készít.
- A kapott tervet megjeleníti a felhasználónak a displayMealPlan(mealPlan) függvény segítségével.
- Ha hiba történik (pl. hibás válasz vagy hálózati probléma), a catch blokkban megjelenik a hiba a konzolban, így könnyebb hibát keresni.

Ez a függvény kulcsszerepet játszik abban, hogy a felhasználó által megadott cél és allergiák alapján dinamikusan generált étkezési terv létrejöhessen. Nem hagyja, hogy a böngésző újratöltse az oldalt, hanem modern, JavaScript-alapú aszinkron feldolgozással jeleníti meg az eredményt.

Adatbázis



32. ábra Adatbázis táblák kapcsolati rajza

- users: Ide rögzítjük a felhasználók regisztrálásakor keletkezett adatokat
- user_meals: Itt tároljuk el azt, hogy a felhasználó melyik ételt vagy italt fogyasztotta el és milyen mennyiségben. Összekapcsolja a felhasználókat az ételekkel, így lehet nyomon követni az egyéni étkezéseket.
- meals: Itt tároljuk az ételek, italok adatait és képeit és hogy melyik munkafolyamathoz kell fogyasztani

Források

Források (receptek, képek):

- internet

Használt programok

- PHP
- HTML
- JavaScript
- CSS
- API
- JSON

- Github
- XAMPP
- SQL