WEISS MANFRÉD
BKSZC Weiss Manfréd Technikum, Szakképző Iskola és Kollégium

Vizsgaremek

RECEPT24

Készítette: Szabó Zsuzsanna,
Tóth Viktor,
Patakfalvi Ádám

Témavezető: Kovács László

Budapest, 2024

Tartalomjegyzék

Bevezetés

Témaválasztás indoklása

A receptes weboldal ötlete a csapat szenvedélyes főzős-sütni iránti szeretetén alapszik. Az egyik csapattag, aki nagyon szeret főzni és sütni, felvetette az ötletet, hogy mi lenne, ha létrehoznánk egy olyan online platformot, ahol recepteket oszthatunk meg és inspirálhatjuk egymást.

Célkitűzés

Célunk egy könnyen kezelhető receptes weboldal létrehozása, amely lehetővé teszi a felhasználók számára a receptek keresését, megosztását és véleményezését. A felhasználóknak lehetőségük van saját receptek feltöltésére, beleértve a recept leírását és képeket is. Azonban a regisztráció mellett további előnyök is várják a felhasználókat, például egyedi profil létrehozása, ami lehetővé teszi számukra a belépést saját profiljukba, hogy könnyen hozzáférjenek a receptekhez és véleményezésekhez. A regisztrált felhasználók képesek lesznek véleményezni a recepteket, megosztani tapasztalataikat és javaslatokat tenni, így aktív és interaktív közösség alakulhat ki az oldalon.

Kiknek szánjuk a weboldalt

A weboldalunkat minden olyan főzős-sütni szeretőnek szánjuk, akik szeretnének tanulni, bővíteni receptkönyvüket, vagy egyszerűen csak élvezni a gasztronómia világát.

Fejlesztői dokumentáció

Fejlesztői környezet

A receptes weboldalunk elkészítéséhez különböző fejlesztőeszközöket használunk, amelyek segítenek a fejlesztésben és az adatok kezelésében. A Visual Studio Code egy könnyen használható kódszerkesztő, amely segít a weboldal kialakításában.

A XAMPP egy olyan program, amely tartalmazza az Apache webkiszolgálót, a MySQL adatbáziskezelőt és a PHP-t, és lehetővé teszi számunkra, hogy lokálisan fejlesszük és teszteljük weboldalunkat a saját számítógépünkön anélkül, hogy szükség lenne valós webkiszolgálóra.

A phpMyAdmin segítségével pedig könnyedén kezelhetjük és karbantarthatjuk az adatbázisunkat.

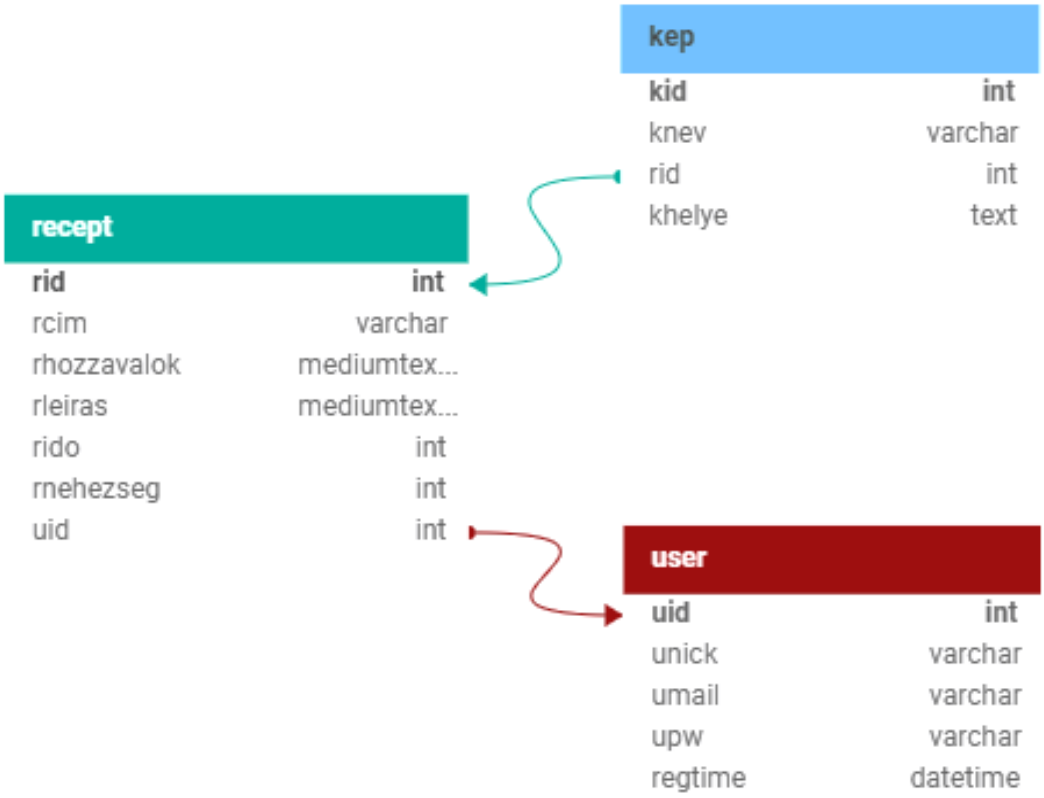
Végül a MySQL adatbázis-kezelő rendszer hatékonyan tárolja és kezeli a receptekhez, felhasználókhoz és egyéb adatokhoz kapcsolódó információkat.

Ezeket a fejlesztőeszközöket választottuk, mert tanultunk velük, és ismerjük őket. A széles körű elérhetőségük és az, hogy ingyenesen elérhetők, tovább erősíti döntésünket. Segítségükkel hatékonyan fejleszthetünk és tesztelhetünk weboldalunkat, és könnyedén kezelhetjük az adatbázist.

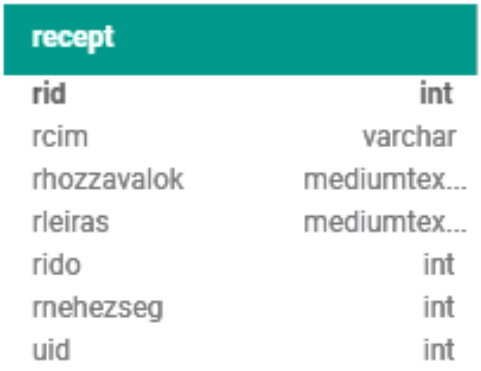
Kialakított adatszerkezet és részletes bemutatása

A receptes weboldalunk adatbázisa több táblát tartalmaz. Ezek a táblák strukturáltak és szervezettek, hogy hatékonyan kezeljék és tárolják az adatokat.

Adatbázis táblái



A ´recept´ tábla



rid (Recept azonosító): Ez a mező az adott recept egyedi azonosítója a táblában. Ez az elsődleges kulcs, amely automatikusan növekszik minden új recept létrehozásakor, és egyedi az adott recepthez. Az egyedi azonosító segít megkülönböztetni és azonosítani a különböző recepteket a táblában.

rcím (Recept címe): Ez a mező tárolja a recept címét, amely segít azonosítani és megkülönböztetni az egyes recepteket. A recept címe megjelenik a receptek listázásakor, és segít a felhasználóknak megtalálni a kívánt receptet.

rhozzavalok (Recept hozzávalók): Ez a mező tárolja a recept hozzávalóit, általában egy hosszabb szöveges mező formájában. Ide tartoznak azok az összetevők és alapanyagok, amelyeket a recept elkészítéséhez szükséges.

rleiras (Recept leírása): Ez a mező tartalmazza a recept lépéseit vagy útmutatásait, amelyeket a felhasználóknak követniük kell a recept elkészítése során. Ez a rész adja meg azokat az utasításokat vagy lépéseket, amelyek segítségével a felhasználók létrehozhatják és elkészíthetik.

rido (Recept időtartama): Ez a mező tárolja a recept elkészítésének várható időtartamát, általában percekben vagy más időegységekben. Ez segít a felhasználóknak felmérni, hogy mennyi időre van szükségük a recept elkészítéséhez.

rnehezseg (Recept nehézségi szintje): Ez a mező jelzi a recept nehézségi szintjét vagy bonyolultságát. Általában egy szám típusú mező, amely tartományban vagy értékben jelzi a recept nehézségét, például könnyű, közepes vagy nehéz.

uid (Felhasználó azonosító): Ez a mező az idegen kulcs a táblában, ami összekapcsolja a recepteket a felhasználókkal. Az user_id alapján lehet kapcsolatot kialakítani a két tábla között, és lehet látni, hogy mely felhasználók hozták létre az egyes recepteket. Ez segít abban, hogy a felhasználók megnézhessék a saját létrehozott receptjeiket.

A ´kep´ tábla

kid (Kép azonosítója): Ez a mező az adott kép egyedi azonosítója a táblában. Az "kid" mező az elsődleges kulcs, ami azt jelenti, hogy minden képhez egyedi azonosítót rendel. Ez segít az egyes képek azonosításában és hivatkozásában más táblákban.

knev (Kép neve): Ez a mező a képek nevét vagy címét tárolja a rendszerben. A "knev" mező legfeljebb 40 karaktert fogad el, és a VARCHAR adattípust használja az adatbázisban, ami lehetővé teszi a változó hosszúságú szövegek tárolását. Ez segíti az egyedi képek azonosítását és megkülönböztetését, hiszen mindegyik képhez külön név vagy cím van rendelve. Így könnyen megtalálhatjuk az adott képet az adatbázisban, amikor szükség van rá.

rid (Recept azonosítója): Ez a mező azonosítja azt a receptet, amelyhez a kép tartozik. Az adott képhez egy recept van rendelve, és ez a mező tartalmazza a kapcsolódó recept azonosítóját. Ez lehetővé teszi a képek összekapcsolását a hozzájuk tartozó receptekkel.

khelye (Kép helye): Ez a mező tárolja a kép fizikai helyét vagy elérési útvonalát. A "khelye" mező TEXT típusú, ami azt jelenti, hogy hosszabb szöveges adatokat tud tárolni, például a kép elérési útvonalát. Ez a mező segít abban, hogy az alkalmazás megtalálja és megjelenítse a képet a weboldalon.

A ´user´ tábla

uid (Felhasználó azonosító): Ez a mező az adott felhasználó egyedi azonosítója a táblában. Az "uid" mező egy szám típusú, és az elsődleges kulcsa a táblának. Automatikusan növekszik minden új felhasználó regisztrációja során, és segíti az egyedi felhasználók azonosítását és megkülönböztetését.

unick (Felhasználónév): Ez a mező tárolja a felhasználó által választott felhasználónevet. A "unick" mező VARCHAR típusú, és legfeljebb 32 karakter hosszú lehet. Ezt a felhasználónevet használja a rendszer a felhasználók azonosítására és megkülönböztetésére.

umail (E-mail cím): Ez a mező tárolja a felhasználó által megadott e-mail címet. A "umail" mező VARCHAR típusú, és legfeljebb 80 karakter hosszú lehet. Az e-mail címek segítenek a felhasználók azonosításában és a kommunikációban velük.

upw (Jelszó): Ez a mező tárolja a felhasználó jelszavát. A "upw" mező VARCHAR típusú, és legfeljebb 40 karakter hosszú lehet. Fontos, hogy a jelszavakat biztonságosan tároljuk az adatbázisban, hogy minimalizáljuk a biztonsági kockázatokat.

regtime (Regisztráció ideje): Ez a mező rögzíti a felhasználó regisztrációjának idejét. A "regtime" mező DATETIME típusú, és az adott időpontot rögzíti minden új felhasználó regisztrációja során. Segít nyomon követni a felhasználók regisztrációs időpontját és tevékenységeit a rendszerben.

Algoritmusok a backend és frontend megvalósításban

Általában a dokumentációk főként a backend részre összpontosítanak, de az algoritmusok működése és azok interakciója a frontend és backend között is fontos szerepet játszik az alkalmazás teljes rendszerének megértésében. A frontend rész bemutatása segíthet a felhasználóknak és a fejlesztőknek is abban, hogy átfogó képet alkothassanak az alkalmazás működéséről az adott alkalmazás funkcióin keresztül.

Felhasználó Regisztrációs folyamat backend oldal

```php
<?php

header('Content-Type: application/json; charset=utf-8');

if (check_fields()) {
    if (free_user()) {
        user_create();
        $createuser = [
            ['hiba'           => 0],
            [
                'unick'          => $_POST["unick"],
                'umail'          => $_POST["umail"],
                'regtime'        => date('Y-m-d H:i:s')
            ]
        ];
    } else {
        $createuser = ['hiba' => "Létezik már ilyen regiaztráció!"];
    }
} else {
    $createuser = ['hiba' => "Nincs kitöltve megfelelően a regisztráció"];
}

function check_fields()
{
    $unick = isset($_POST["unick"]) && !empty($_POST["unick"]);
    $umail = isset($_POST["umail"]) && !empty($_POST["umail"]);
    $upw = isset($_POST["upw"]) && !empty($_POST["upw"]);
    return $unick && $umail && $upw;
}

function free_user()
{
    $unick = $_POST["unick"];
    $umail = $_POST["umail"];


    require '../connection/index.php';

    $search = mysqli_query($conn, "SELECT unick, umail FROM user WHERE unick = '$unick' OR umail = '$umail'");
    $row = mysqli_fetch_assoc($search);

    mysqli_close($conn);
```

```
    if (!isset($row)) {
        return TRUE;
    } elseif ($row['unick'] == $unick || $row['umail'] == $umail) {
        $finduser = ['hiba' => "Létezik már ilyen regiaztráció!"];
        return FALSE;
    };
}

function user_create()
{


    $unick = htmlspecialchars($_POST["unick"]);
    $umail = htmlspecialchars($_POST["umail"]);
    $upw = $_POST["upw"];

    require '../connection/index.php';

    $stmt = "INSERT INTO user (unick, umail, upw, regtime) VALUES (?, ?, ?, NOW())";

    $stmt_prepare = mysqli_prepare($conn, $stmt);

    mysqli_stmt_bind_param($stmt_prepare, "sss", $unick, $umail, $upw);

    mysqli_stmt_execute($stmt_prepare);

    mysqli_close($conn);
}

$json = json_encode($createuser, JSON_UNESCAPED_UNICODE);
print $json;
?>
```

A header beállításoknál a Content-Type fejléc JSON formátumra történő beállítása történik.

Az űrlap mezők ellenőrzésekor (check_fields) ellenőrizzük, hogy a POST metódussal érkező adatokban megtalálhatóak-e a kötelező mezők: felhasználónév (unick), e-mail (umail) és jelszó (upw). Ha bármelyik mező hiányzik vagy üres, a regisztráció nem folytatódik.

A felhasználói adatok szabad már léteznek-e (free_user) lépésben ellenőrizzük, hogy a felhasználó által megadott felhasználónév vagy e-mail cím már létezik-e az adatbázisban más felhasználóhoz kötve. Ha már foglaltak, a regisztráció sikertelen lesz.

A felhasználó létrehozásakor (user_create) az adatokat beillesztjük az adatbázisba, beleértve a felhasználónév, e-mail cím, jelszó és regisztráció időpontját.

Végül JSON formátumban visszajelzést küldünk a regisztrációs folyamat eredményéről. Ha a regisztráció sikeres volt, a felhasználó adatait küldjük vissza, ellenkező esetben hibaüzenetet adunk vissza, például ha a felhasználónév vagy e-mail cím már foglalt volt.

Ez a folyamat biztosítja a felhasználók regisztrációjának hatékony és biztonságos kezelését a rendszerben, amelyet a backend rész gondoskodik az adatok helyes kezeléséről és azok biztonságos rögzítéséről az adatbázisban.

Felhasználó Regisztrációs folyamat frontend oldal

```php
<?php

if (!isset($_POST['upw']))
{
    ?>
    <div class="container">
    <div class="form">
    <div class="title">Recept24</div>
    <div class="subtitle">Regisztráció!</div>

    <form method="post" action="register.php" id="register">
    <div class="input-container ic1">
    <input type="email" id="umail" name="umail" class="input" required placeholder=" ">
    <div class="cut cutshort1"></div>
    <label class="placeholder" for="umail">Email</label>
    </div>
    <div class="input-container ic2">
    <input type="text" id="unick" name="unick" required class="input" placeholder=" ">
    <div class="cut"></div>
    <label class="placeholder" for="unick">Felhasználónév</label>
    </div>

    <div class="input-container ic2">
    <input type="password" id="upw" name="upw" required class="input" placeholder=" ">
    <div class="cut cutshort2"></div>
    <label class="placeholder" for="upw">Jelszó</label>
    </div>

    <button type="text" class="submit">Regisztrál</button>
    <div class="subtitle">Van már fiókja? <?php echo '<a href="?p=login">Bejelentkezés</a>' ?></a></div>
    </form>
    </div>
    </div>

    <?php
    echo '</div>';
}
else
{
    $hashedPassword = password_hash(($_POST['upw']),PASSWORD_DEFAULT);

    $apiUrl = 'http://recept24.szakdoga.net/api/createuser/index.php';

    $userAgent = 'Mozilla/5.0 (Windows NT 11.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.9999.99 Safari/537.36';
```

```php
    $data = [
        'umail' => $_POST['umail'],
        'unick' => $_POST['unick'],
        'upw'   => $hashedPassword,
    ];

    $ch = curl_init($apiUrl);

    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($data));
    curl_setopt($ch, CURLOPT_USERAGENT, $userAgent);

    $response = curl_exec($ch);
    $json_response = json_decode($response);



    if ($response === false) {
        echo 'cURL request failed: ' . curl_error($ch);
    } else {
        //echo $response;
    }


    curl_close($ch);
}

?>
```

Az űrlap megjeleníti a regisztrációs űrlapot, amikor a felhasználó először megnyitja az oldalt vagy újratölti azt, és tartalmazza az e-mail cím, felhasználónév és jelszó mezőket, amelyeket a felhasználó kitölt.

Amikor a felhasználó elküldi az űrlapot a regisztrációs adatokkal, a frontend először hash-eli a jelszót biztonságos továbbítás érdekében, majd összeállítja az adatokat a backend számára, ideértve az e-mail címet, felhasználónevet és a hashelt jelszót.
A frontend POST kérést küld az adatokkal a backend API-jához a regisztráció végrehajtására, majd várakozik a válaszra a backend-től az API kérésre.
Ha a válasz sikeres, azaz a regisztráció feldolgozása és rögzítése az adatbázisban sikeresen megtörtént, a frontend opcionálisan megjeleníti a sikeres regisztrációs üzenetet vagy az API által visszaküldött adatokat. Ellenkező esetben, ha a válasz sikertelen, például a backend nem érhető el vagy hiba történt a regisztráció során, a frontend hibaüzenetet jelenít meg a felhasználónak.

A frontend CURL kérést használ az adatok küldésére és a válasz fogadására a backend API-jához. Ha a CURL kérés sikertelen (például hálózati hiba vagy hibás URL), a frontend hibaüzenetet jelenít meg a felhasználónak.

Ez a folyamat biztosítja, hogy a felhasználók regisztrációs adatai megfelelően és biztonságosan kerüljenek továbbításra és rögzítésre a backend rendszerben, és hogy a frontend felhasználóbarát módon kommunikáljon a felhasználóval a regisztrációs folyamat során fellépő eseményekről.

Recept Feltöltési folyamat backend oldal

```php
<?php
// Recept hozzáadása az adatbázishoz:

header('Content-Type: application/json; charset=utf-8');

// Fő logika, az eredmény kimenetekkel:
if (check_fields()) {
    // Ha minden mező ki van töltve, létrehozzuk a receptet
    recipe_create();
    // Visszajelzés létrehozása az eredménnyel
    $createrecipe = [
        ['hiba'           => 0],
        [
            'rcim'         => $_POST["rcim"],
            'rhozzavalok'  => $_POST["rhozzavalok"],
            'rleiras'      => $_POST["rleiras"],
            'rido'         => $_POST["rido"],
            'rnehezseg'    => $_POST["rnehezseg"],
            'uid'          => $_POST["uid"]
        ]
    ];
} else {
    // Ha valamelyik mező nincs kitöltve, hibát jelzünk
    $createrecipe = ['hiba' => "Nincs kitöltve megfelelően a recept"];
}

// Ellenőrzi, hogy minden mező ki van-e töltve
function check_fields()
{
    // Ellenőrzi, hogy minden mező tartalmaz-e értéket a $_POST tömbben
    $rcim = isset($_POST["rcim"]) && !empty($_POST["rcim"]);
    $rhozzavalok = isset($_POST["rhozzavalok"]) && !empty($_POST["rhozzavalok"]);
    $rleiras = isset($_POST["rleiras"]) && !empty($_POST["rleiras"]);
    $rido = isset($_POST["rido"]) && !empty($_POST["rido"]);
    $rnehezseg = isset($_POST["rnehezseg"]) && !empty($_POST["rnehezseg"]);
    $uid = isset($_POST["uid"]) && !empty($_POST["uid"]);

    // Visszatérési érték true, ha minden mező ki van töltve, különben false
    return $rcim && $rhozzavalok && $rleiras && $rido && $rnehezseg && $uid;
}
```

```php
// Recept létrehozása az adatbázisban.
/* A POST adatok kiolvasása és kódolása, majd kapcsolat felépítése az adatbázissal.
 */
function recipe_create()
{
    /* A POST függvény kiolvasása, a htmlspecialchars() segítségével bizonyos karakterek ", ', <, >
     kikódolásával:
     */
    $rcim = htmlspecialchars($_POST["rcim"]);
    $rhozzavalok = htmlspecialchars($_POST["rhozzavalok"]);
    $rleiras = htmlspecialchars($_POST["rleiras"]);
    $rido = htmlspecialchars($_POST["rido"]);
    $rnehezseg = htmlspecialchars($_POST["rnehezseg"]);
    $uid = htmlspecialchars($_POST["uid"]);

    // Adatbázis kapcsolat felépítése
    require '../connection/index.php';

    // Beszúrási utasítás előkészítése
    $stmt = "INSERT INTO recept (rcim, rhozzavalok, rleiras, rido, rnehezseg, uid) VALUES (?, ?, ?, ?, ?, ?)";

    $stmt_prepare = mysqli_prepare($conn, $stmt);

    // Paraméterek összekapcsolása a beszúrási utasítással
    mysqli_stmt_bind_param($stmt_prepare, "sssiii", $rcim, $rhozzavalok, $rleiras, $rido, $rnehezseg, $uid);

    // Beszúrás végrehajtása
    mysqli_stmt_execute($stmt_prepare);

    // Adatbáziskapcsolat bezárása
    mysqli_close($conn);
}

// Válasz JSON formátumba alakítása és kiírása
$json = json_encode($createrecipe, JSON_UNESCAPED_UNICODE);
print $json;
?>
```

Az első lépésben a válasz Content-Type fejlécét application/json értékre állítja, ami jelzi, hogy a válasz JSON formátumban lesz visszaadva.

Ezután a fő logika ellenőrzi, hogy minden szükséges mező ki van-e töltve a POST kérésben. Ha minden mező ki van töltve, akkor létrehozza a receptet a recipe_create() függvény segítségével, majd összeállítja a választ a sikeres regisztráció adataival. Ha valamelyik mező nincs megfelelően kitöltve, akkor hibaüzenetet állít elő.

A check_fields() függvény biztosítja, hogy minden szükséges mező tartalmaz-e értéket a $_POST tömbben, és visszatérési értéke true, ha minden mező ki van töltve, különben false.

A recipe_create() függvényben a POST adatokat kikódolja bizonyos karakterekkel a biztonságos adatbázisműveletek érdekében. Ezután felépít egy kapcsolatot az adatbázissal,

előkészíti a beszúrási utasítást a recept táblába, majd végrehajtja a beszúrást és bezárja az adatbáziskapcsolatot.

Végül az eredményt JSON formátumba alakítja és kiírja a választ a kliens felé. Ez a folyamat biztosítja, hogy a regisztráció során az adatokat biztonságosan rögzítse az adatbázisban, és a válasz JSON formátumban kerüljön visszaadásra a kliensnek.

Különböző körülmények, esetek és hibakezelések

Keresési Folyamat Kezelése

Az alkalmazás keresési funkciója esetében fontos, hogy megfelelően kezeljük a felmerülő különböző körülményeket és hibákat mind a backend, mind a frontend részén.

A backend kódban először is figyelmet kell fordítanunk arra az esetre, amikor a felhasználó nem ad meg keresési paramétert, vagy üres értéket küld a GET kérésben. Ebben az esetben a backend válaszul egy hibaüzenetet küld vissza a frontendnek, amely figyelmezteti a felhasználót, hogy töltse ki a keresőablakot. Ha a keresési paraméter megfelelő és talál találatokat az adatbázisban, akkor a backend JSON objektumban küldi vissza a talált recepteket a frontend számára, hogy megjeleníthesse azokat. Amennyiben a keresési paraméterrel nem található recept az adatbázisban, a backend egy megfelelő hibaüzenetet küld vissza a frontendnek, hogy tájékoztassa a felhasználót arról, hogy nincs találat.

A frontend kódban a hiányzó keresőmezőt kezeljük először. Ha a felhasználó nem tölti ki a keresőmezőt és elküldi a formot, a frontend hibaüzenetet jelenít meg, hogy kérje a felhasználót, hogy töltse ki a keresőmezőt. Ha érvénytelen adatot ad meg a felhasználó a keresőmezőben, a frontend végzi a validációt és megjeleníti a hibaüzenetet az érvénytelen adatokról. Ha a frontend nem tud kapcsolódni a backendhez az adatok lekérdezésekor, akkor megjeleníti a hálózati hibaüzenetet a felhasználónak, és javasolja, hogy próbálkozzon később újra.

Belépési Folyamat Kezelése

A belépési folyamat során fontos megfelelően kezelni az esetleges különböző körülményeket és hibákat mind a backend, mind a frontend részén. Elsőként figyelembe kell venni azt az esetet, amikor a felhasználó nem tölti ki a felhasználónév vagy jelszó mezőt. Ebben az esetben a frontend hibaüzenetet jelenít meg, hogy kérje a felhasználót, hogy töltse ki mindkét mezőt. Amennyiben a felhasználó megadja a helyes felhasználónevet és jelszót, a backend ellenőrzi az adatokat az adatbázisban. Ha a bejelentkezés sikeres, akkor a backend JSON objektumban visszaküldi a felhasználó adatait a frontendnek, amely megjeleníti a sikeres bejelentkezést. Abban az esetben, ha a felhasználó hibás felhasználónevet vagy jelszót ad meg, a backend

hibaüzenetet küld vissza a frontendnek, hogy tájékoztassa a felhasználót a hibás bejelentkezési adatokról. Ha a frontend nem tud kapcsolódni a backend szerverhez a bejelentkezési adatok elküldésekor, akkor hibaüzenetet jelenít meg a felhasználónak, hogy tájékoztassa a kapcsolódási problémáról, és javasolja, hogy próbálkozzon később újra. Végül, ha a felhasználó sikeresen bejelentkezik, a frontend átirányítja őt a főoldalra, hogy folytathassa a rendszer használatát

Fejlesztési lehetőségek

A modern gasztronómia világában elengedhetetlen egy olyan receptes weboldal, amely egyszerűen és inspirálóan szolgál mindazok számára, akik szenvedélyesen szeretnek főzni és megosztani élményeiket. Ebben a fejlesztésben rengeteg lehetőség rejlik, melyek segítségével még vonzóbbá és hatékonyabbá tehetjük az oldalt.

Példaként néhány fejlesztési lehetőség:

- Képek és videók hozzáadása a receptekhez: Ez lehetővé teszi a felhasználóknak, hogy még érthetőbben megosszák receptjeiket, bemutassák az elkészítés folyamatát, és inspiráljanak másokat.
- Receptek kategorizálása: Segítse a felhasználókat a könnyebb böngészésben és a receptek megtalálásában a kategóriák használatával, például vegetáriánus, gyorsétel stb.
- Receptkönyv funkció: Kínáljon lehetőséget a felhasználóknak, hogy saját receptkönyvet hozzanak létre, így könnyen elérhetik és megosszák kedvenc ételeiket.
- Értesítések és tevékenységkövetés: Biztosítsa a felhasználóknak az értesítéseket az új receptekről, kommentekről vagy követett felhasználóik aktivitásáról, így folyamatosan kapcsolatban maradhatnak az oldallal.
- Chat funkció: Nyújtson lehetőséget a felhasználók számára a közvetlen kommunikációra, recepttanácsok megosztására vagy közös sütés-főzés tervezésére egy chat ablakon keresztül.

Felhasználói dokumentáció

Üdvözlöm!

Kézben tartod a Recept24 weboldalát, egy online receptmegosztó platformot, ahol könnyedén felfedezhetsz új gasztronómiai ötleteket és megoszthatod saját receptjeidet.

A Recept24 alapvető funkciói között szerepel a belépés és regisztráció lehetősége. Ha még nem regisztráltál, egyszerűen létrehozhatod a fiókodat, majd bejelentkezhetsz az oldalra, hogy teljes körű hozzáférést kapj a funkciókhoz.

Emellett a weboldalon lehetőséged van receptek böngészésére és keresésére. Fedezz fel új ízletes fogásokat, böngéssz különböző kategóriák szerint, vagy használd a keresőt a kedvenc receptjeid gyors megtalálásához.

Ha saját recepttel rendelkezel, könnyedén megoszthatod azt a közösséggel. Egyszerűen töltsd fel a recept címét, hozzávalóit, leírását, elkészítési időt és nehézségi szintet, hogy mások is élvezhessék az általad alkotott ínyencséget.

A Recept24 segítségével könnyedén inspirálódhatsz a konyhában, új fogásokat próbálhatsz ki, és tapasztalatokat cserélhetsz az ételkedvelő közösséggel.

Köszönjük, hogy csatlakoztál hozzánk, és jó étvágyat kívánunk az ízletes kalandokhoz!

Üdvözlettel,
Recept24 csapata