Budapesti Műszaki Szakképzési Centrum

Verebély László Technikum

5 0613 12 03 Szoftverfejlesztő és -tesztelő szakképesítés

Jössz Büfé- iskola büfé alkalmazás

Készítette: Szilágyi Péter, Svecov Szergej

Konzulens tanárok: Somogyi Erika, Horváth Attila

2024. április

NYILATKOZAT A SZAKDOLGOZAT EREDETISÉGÉRŐL

Alulírott Szilágyi Péter és Svecov Szergej a BMSZC Verebély László Technikum 5 0613 12 03 Szoftverfejlesztő és -tesztelő képzésében részt vevő tanulók büntetőjogi felelősség tudatában nyilatkoznak és aláírásukkal igazolják, hogy a Jössz Büfé című vizsgaremek saját, önálló munkánk, és abban betartottuk az iskola által előírt, a vizsgaremek készítésére vonatkozó szabályokat.

Tudomásul veszem, hogy a szakdolgozatban plágiumnak számít:

- szó szerinti idézet közlése idézőjel és hivatkozás nélkül,
- tartalmi idézet hivatkozás megjelölése nélkül,
- más publikált gondolatainak saját gondolatként való feltüntetése.

E nyilatkozat aláírásával tudomásul veszem továbbá, hogy plágium esetén szakdolgozatom visszautasításra kerül.

Budapest, 2023. április 14. _____

Tanulók aláírása

Tartalom

1. Bevezetés

Szakdolgozatunk témájának kiválasztásakor, megpróbáltunk egy olyan problémára megoldást találni, ami akár diákok, akár "felnőttek" számára megoldást nyújthat egy mindennapi problémára. Ez esetben a mindennapos büfés sorbaállást szeretnénk lerövidíteni, így akár azt hatékonyabban eltölteni, például: tanulással vagy baráti beszélgetésekkel.

A covid óta nagy népszerűségre tettek szert az akkortájt létrejövő futárszolgálatok, melyek működése nagyon egyszerű. Egy alkalmazáson vagy webes felületen keresztül rendelhetünk megadott éttermek kínálatából, amit egy futár házhoz is szállít nekünk. Ezzel elkerüljük a sorbaállást és a rendelés felvétel sem húzza a dolgozók idejét.

Hasonló módon szeretnénk átültetni ezt az iskolák világába. Az iskolák büfé tulajdonosai regisztrálhatnának oldalunkra és feltölthetik az általuk kínált termékeket (pl.: hot-dog, chips stb.). Amiket a szintén regisztráló diákok megrendelhetnek és megadott szünetekben fizethetnek érte, illetve át is vehetik a termékeket. Mivel a büfés már 1-2 szünettel előre tudhatja, hogy miket kell átadnia a következő szünetekben, korábban elkészítheti azokat. Ezzel vélhetően a kiszolgálás többszörös gyorsasággal működhetne. A büfések több diákot tudnak kiszolgálni, a diákok pedig több időt tölthetnek el sorbaállás nélkül és helyette tanulhatnak, beszélgethetnek jobban kikapcsolódhatnak.

1. Követelményspecifikáció

Az iskolai büfék rendeléseinek kezelésére szolgáló weboldal funkcionális követelményei alapján a diákoknak és esetlegesen a büfé munkatársainak lehetősége van regisztrálni és bejelentkezni az oldalra. A rendelési folyamat során a diákok könnyen választhatnak a büfé kínálatából és hozzáadhatják a kívánt ételeket vagy italokat a kosárhoz. A kosár tartalmának megtekintése, módosítása és törlése is egyszerűen megoldott. A felhasználók könnyen kezelhetik a profiljukat.

A C sharp alkalmazás követelményei közé tartozik, hogy a büfés fiókkal rendelkező felhasználók be tudjanak lépni. Belépés után kezelhetik a büfé termékeit, azok hozzávalóit illetve extra hozzáfűzhető termékeket (pl.:üdítő, köret, stb.).

Az weboldal oldal nem funkcionális követelményei között szerepel, hogy felhasználóbarát felületet kell biztosítani a diákok számára, így könnyen és gyorsan megtalálhatják az általuk keresett funkciókat. Az oldal gyors teljesítménye biztosítja, hogy az oldal azonnal reagál a felhasználó interakcióira. A biztonsági intézkedéseknek köszönhetően az adatok védelme biztosított, és az oldal képes skálázhatóságra, így bármennyi diák és büfé csatlakozhat hozzá.

A webes alkalmazás mindenféle eszközön és böngészőn keresztül elérhető, és az adatok tárolásához egy adatbázist használ.

2. Fejlesztői dokumentáció

1. Alkalmazott fejlesztői eszközök

- Kódszerkesztő: Visual Studio Code, Visual Studio
- Fejlesztői környezet: NextJS, .NET

2. Alkalmazott technológiák

- Programozási nyelvek: JavaScript (TypeScript), C#
- Webes technológiák: HTML, CSS és Bootstrap 5.3 könyvtár
- Adatbázis motor: MariaDB (MySQL)

3. Futtatási környezet

A weboldal bármely modern böngészőben használható, akár mobileszközökön is (mobil, tablet, laptop).

A C# alkalmazás csak asztali számítógépeken használható és vagy bármely eszközön melynek operációs rendszere a következők egyike: Windows 10 vagy újabb, Linux asztali verziói, macOS.

4. Fejlesztés menete

Az adatbázisunk MySQL nyelven van, MariaDB-vel működik, és megfelel a harmadik normál formának. Ez azt jelenti, hogy az adatok jól strukturáltak és lehetőség szerint nincs redundancia. A harmadik normál forma segít az adatok rendezettségében és az adatbázis hatékonyságában.

Az adatbázis 13 táblát tartalmaz. Tervezéskor a célunk az volt, hogy a lehető leglogikusabb felbontást alkossuk meg és törekedtünk a táblák megfelelő elnevezésére. Minimalizáltuk a redundanciát, hogy a tárhelyet a lehető leghatékonyabban használhassuk ki.

1. `buffets` Tábla:

Ez a tábla tárolja a büfék adatait, mint például a nevüket, címüket és regisztrációs dátumukat.

Mezők:

- **`id`:** Egyedi azonosító (INT, auto increment).

`name`: A büfé neve (VARCHAR(100)).

- **`address`:** A büfé címe (VARCHAR(100)).

- **`reg_date`:** A büfé regisztrációs dátuma (DATETIME).

2. `categories` Tábla:

Ez a tábla különböző termékkategóriákat tárol.

Mezők:

- **`id`:** Egyedi azonosító (INT, auto increment).

- **`name`:** A kategória neve (VARCHAR(30)).

3. products Tábla:

Ez a tábla tárolja a büfék által kínált termékek részletes adatait, mint például a nevüket, leírásukat, árukat és kategóriájukat.

Mezők:

- **`id`:** Egyedi azonosító (INT, auto increment).

- **`buffet_id`:** A büfé azonosítója, amelyben a termék elérhető (INT).

- **`category_id`:** A termék kategóriájának azonosítója (INT, idegen kulcs).

- **`name`:** A termék neve (VARCHAR(45)).

- **`description`:** A termék leírása (TEXT).

- **`price`:** A termék ára (INT).

- **`discount`:** A termék kedvezménye százalékban (INT(3)).

4. `users` Tábla:

Ez a tábla tárolja a felhasználók adatait, beleértve a felhasználónevet, jelszót, nevet, e-mail címet és regisztrációs dátumot. Emellett tartalmaz egy mezőt, ami megadja, hogy a felhasználó melyik büfét vezeti.

Mezők:

- **`id`:** Egyedi azonosító (INT, auto increment).

- **`managed_buffet`:** A felhasználó által vezetett büfé azonosítója, lehet NULL érték is (INT, idegen kulcs).

- **`username`:** A felhasználói név (VARCHAR(45)).

- **`password`:** A jelszó (VARCHAR(100)).

- **`name`:** A felhasználó teljes neve (VARCHAR(45)).

- **`email`:** Az e-mail cím (VARCHAR(50)).

- **`reg_date`:** A felhasználó regisztrációs dátuma (DATETIME).

5. `orders` Tábla:

Ez a tábla tartalmazza a rendelések adatait, beleértve azt, hogy melyik felhasználó adta le, melyik büféből, a teljes árat, a rendelés dátumát és hogy befejeződött-e már vagy sem.

Mezők:

- **`id`:** Egyedi azonosító (INT, auto increment).

- **`user_id`**: A rendelést leadó felhasználó azonosítója (INT, idegen kulcs).

- **`buffet_id`:** A büfé azonosítója, ahonnan a rendelést leadták (INT, idegen kulcs).

- **`price`:** A rendelés teljes ára (INT).

- **`date`:** A rendelés dátuma (DATETIME).

- **`completed`:** Logikai érték, jelzi hogy a rendelés teljesítve lett-e (TINYINT(1)).

---

6. `allergens` Tábla:

Ez a tábla tárolja az allergének adatait, amelyeket a termékek tartalmazhatnak.

Mezők:

- **`id`:** Egyedi azonosító (INT, auto increment).

- **`name`:** Az allergén neve (VARCHAR(30)).

7. `optional_ingredients` Tábla:

Ez a tábla tartalmazza az opcionális összetevők adatait, amelyeket a termékekhez adhatnak.

Mezők:

- **`id`:** Egyedi azonosító (INT, auto increment).

- **`buffet_id`:** A büfé azonosítója, amelyben az opcionális összetevő elérhető (INT, idegen kulcs).

- **`product_id`:** A termék azonosítója (INT, idegen kulcs).

- **`name`:** Az opcionális összetevő neve (VARCHAR(30)).

8. `extras` Tábla:

Ez a tábla tartalmazza az extra elemek adatait, amelyeket a rendelésekhez lehet hozzáadni.

Mezők:

- **`id`:** Egyedi azonosító (INT, auto increment).

- **`buffet_id`:** A büfé azonosítója, amelyből az extra elemek elérhetők (INT, idegen kulcs).

- **`name`:** Az extra elem neve (VARCHAR(30)).

- **`price`:** Az extra elem ára (INT).

---

9. `product_extras` Tábla:

Ez a tábla reprezentálja a termékek és az extra elemek közötti kapcsolatot.

Mezők:

- `id`: Egyedi azonosító (INT, auto increment).

- **`buffet_id`:** A büfé azonosítója, amelyben a kapcsolat elérhető (INT, idegen kulcs).

- **`product_id`:** A termék azonosítója (INT, idegen kulcs).

- **`extra_id`:** Az extra elem azonosítója (INT, idegen kulcs).

10. `product_allergens` Tábla:

Ez a tábla reprezentálja a termékek és az allergének közötti kapcsolatot.

Mezők:

- **`allergen_id`:** Az allergén azonosítója (INT, idegen kulcs).

- **`product_id`:** A termék azonosítója (INT, idegen kulcs).

11. `order_products` Tábla:

Ez a tábla tartalmazza az egyes rendelésekhez tartozó termékeket és azok mennyiségét.

Mezők:

- `id`: Egyedi azonosító (INT, auto increment).

- **`order_id`:** A rendelés azonosítója (INT, idegen kulcs).

- **`product_id`:** A termék azonosítója (INT, idegen kulcs).

- `quantity`: A termék mennyisége a rendelésben (TINYINT). (Nincs hossz meghatározva, mivel a max rendelhető db/termék 100)

12. `removed_ingredients` Tábla:

Ez a tábla rögzíti a rendelésekben eltávolított összetevőket.

Mezők:

- **`id`:** Egyedi azonosító (INT, auto increment).

- `order_product_id`: A rendelés termékeinek azonosítója (INT, idegen kulcs).

- **`order_id`:** A rendelés azonosítója (INT, idegen kulcs).

- `ingredient_id`: Az eltávolított összetevő azonosítója (INT, idegen kulcs).

13. `added_extras` Tábla:

Ez a tábla rögzíti a rendelések termékeihez hozzáadott extra elemeket.

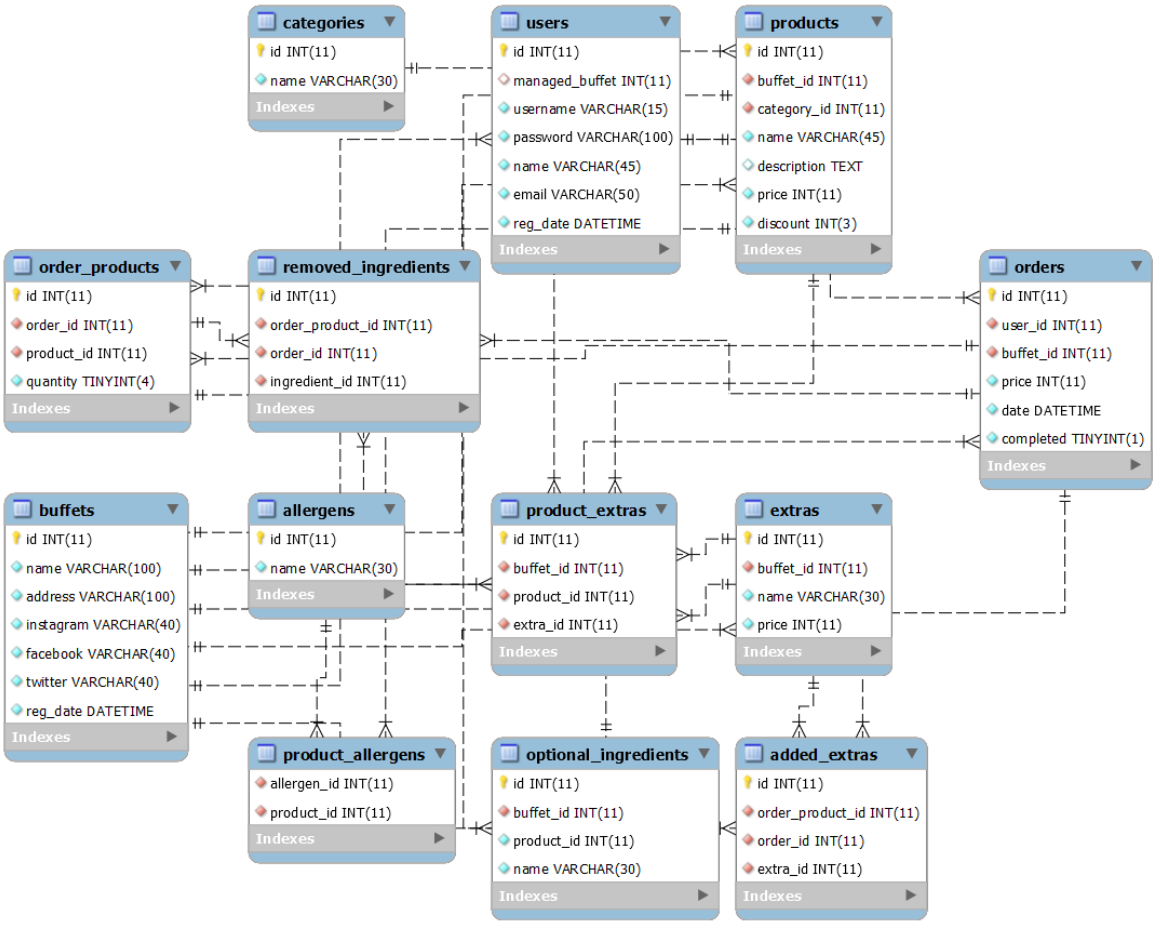
Mezők:

- `id`: Egyedi azonosító (INT, auto increment).

- **`order_product_id`:** A rendelés termékeinek azonosítója (INT).

- `order_id`: A rendelés azonosítója (INT).

- **`extra_id`:** Az extra elem azonosítója (INT).



1. ábra EER diagram a bfüé adatbázisáról

5. Főbb algoritmusok

A weboldal backendje NextJS, amely React-et használva biztosít szerver oldalt is. Azért választottuk a NextJS-t mivel megbízható és modern fejlesztési környezetet biztosít számunkra számos hasznos telepíthető modullal. Ilyen modul például a NextAuth aminek segítségével könnyedén biztosíthatjuk a felhasználók megbízható azonosítását és kezelését. A felhasználók regisztrációjakor jelszavukat sózott (salt) hash-el titkosítjuk, így egy esetleges illetéktelen hozzáféréskor a felhasználók jelszava szinte visszafejthetetlen a mai technológiákkal.

A regisztrációnál számos validációt alkalmaztunk első sorban a helyes adatok beviteléhez, másod sorban kötelezzük a felhasználókat arra, hogy megfeleljenek a minimális jelszó biztonsági feltételeknek (minimum 6 karakter). Az ellenőrzések elsősorban kliens oldalon történnek, viszont az illetéktelen visszaélések érdekében szerver oldalon is megtörténnek.

```
export async function POST(req: NextRequest) {
    const body = await req.json();
    const { name, username, email, password } = body;

    if (!validateUsername(username) || !validateEmail(email)) {
        return NextResponse.json({ error: "A megadott adatok nem voltak megfelelő formátumban" });
    }

    const rawUser = await query("SELECT * FROM users where username=?", [username]);
    const rawEmail = await query("SELECT * FROM users where email=?", [email]);

    if (rawUser == undefined || rawEmail == undefined) {
        return NextResponse.json({ error: "Hibatörtént a regisztráció közben!" });
    }

    if (rawUser.length > 0) {
        return NextResponse.json({ error: "Ez a felhasználónév már foglalt!" });
    }

    if (rawEmail.length > 0) {
        return NextResponse.json({ error: "Ez az email cím már foglalt!" });
    }

    if (password.length < 6) {
        return NextResponse.json({ error: "Ez a jelszó túl rövid!" });
    }

    const rawInsert = await query("INSERT INTO users (name, username, email, password) VALUES (?, ?, ?, ?)",
        [name, username, email, hashPassword(password)]);

    if (rawInsert == undefined) {
        return NextResponse.json({ error: "Hibatörtént a regisztráció közben!" });
    }

    return NextResponse.json({ success: true });
}
```

2. ábra regisztrációs függvény

A felhasználóknak lehetőségük van az adataik megváltoztatására (név, e-mail, jelszó). Ezek szintén elsősorban egy reaktív és informatív módon lettek megoldva kliens oldalon, visszajelzést adunk a

felhasználónak amennyiben valamelyik adatot nem tudja módosítani, mert az nem felel meg a követelményeknek (pl.: foglalt, túl rövid, stb.).

```
export async function POST(req: NextRequest) {
    const body = await req.json();
    const { id, userId, oldData, newData } = body;
    const session = await getServerSession(authOptions);

    if (!session || !session.user || session.user.id != userId) {
        return NextResponse.json({ error: 'Unauthorized' }, { status: 401 });
    }

    switch (id) {
        case "email":
            const emailValidation = await validateEmail(newData);
            if (!emailValidation.success) {
                return NextResponse.json({ success: false, error: emailValidation.error });
            }
            session.user.email = newData;
            break;
        case "username":
            const usernameValidation = await validateUsername(newData);
            if (!usernameValidation.success) {
                return NextResponse.json({ success: false, error: usernameValidation.error });
            }
            session.user.name = newData;
            break;
        case "name":
            if (newData.length > 45) {
                return NextResponse.json({ success: false, error: "A név maximum 45 karakter lehet" });
            }
            break;
        case "image":
            const fileName = `${session.user.id}.webp`;
            uploadImage(newData, "users", fileName);
            session.user.email = fileName;
            return NextResponse.json({ success: true });
        case "password":
            const rawUserData = await query("SELECT id, username, password FROM users WHERE id = ?", [userId]);
            if (!rawUserData || rawUserData.length == 0) {
                throw new Error("A megadott felhasználó nem található!");
            }
            const user = rawUserData[0];
            if (user.password !== hashPassword(oldData)) {
                return NextResponse.json({ success: false, error: "A régi jelszó nem egyezik a jelenlegivel!" });
            }
            if (newData.length < 6) {
                return NextResponse.json({ success: false, error: "A jelszónak legalább 6 karakternek kell lennie!"
});
            }
            if (oldData == newData) {
                return NextResponse.json({ success: false, error: "A régi és az új jelszó nem lehet azonos!" });
            }
            query(`UPDATE users SET ${id}=? WHERE id=?`, [hashPassword(newData), userId]);
            return NextResponse.json({ success: true });
    }

    query(`UPDATE users SET ${id}=? WHERE id=?`, [newData, userId]);

    return NextResponse.json({ success: true });
}
```

3. ábra adat módosító függvény

Nagy hangsúlyt fektettünk a kosár megfelelő működésére. Mivel az egyes termékek azon kívül, hogy betehetőek a kosárba, a felhasználó előre kivet és hozzátehet extrákat, ez azt jelenti, hogy mind a felhasználónak mind a büfésnek a lehető leginformatívabban kell megjeleníteni az információkat.

```
function addItem(element: CartItem) {
    const items = [...cartItems];
    element.addedExtras.sort((a, b) => a.id - b.id);
    element.removedIngredients.sort((a, b) => a.id - b.id);
    const itemIndex = getItemIndex(element);

    if (itemIndex > -1) {
        if (items[itemIndex].quantity >= 100) {
            toast.remove();
            toast.error("Egy termékből maximum 100 darabot rendelhetsz!", {
                duration: 5000,
                position: "top-center",
                style: errorTheme
            });
            return;
        }
        items[itemIndex].quantity++;
    } else {
        items.push(element);
    }
    toast.remove();
    save(items);
    toast.success("Sikeresen hozzáadva a kosárhoz!", {
        duration: 5000,
        position: "top-center",
        style: successTheme
    });
}
```

4. ábra kosár elem hozzáadó algoritmus

Amikor egy elemet hozzáadunk a kosárhoz az algoritmus lemásolja a lista előző állapotát, első elemnél ez természetesen egy üres tömb, majd a hozzáadott és kivett elemeket azonosítójuk alapján sorba rendezzük, hogy megkönnyítsük a termékek csoportosítását. Ezzel eltudjuk érni azt, hogy az azonos módosítással rendelkező azonos termékek ne kerüljenek többször bele a kosárba, csak a darabszámukat növeljük. Ezzel a módszerrel rengeteg memóriát spórolhatunk meg és megjelenítési szempontból is felhasználó barát módszer.

Ezután megállapítjuk a getItemIndex függvénnyel azonosítjuk az elemeket, ha már van azonos termék a megadott módosításokkal akkor megkapjuk az indexét, ha ilyen még nem létezik a függvény „-1”-et

```
function getItemIndex(cartItem: CartItem): number {
     return cartItems.findIndex(item_ =>
         cartItem.product.id == item_.product.id
         && cartItem.addedExtras.length == item_.addedExtras.length // megegyezik-e az extrák hossza
         && cartItem.addedExtras.every((extra, index) => extra.id == item_.addedExtras[index].id)
             // megegyeznek-e az extrák
         && cartItem.removedIngredients.length == item_.removedIngredients.length
             // megegyezik-e az törölt hozzávalók hossza
         && cartItem.removedIngredients.every((ingredient, index) =>
         ingredient.id == item_.removedIngredients[index].id)); // megegyeznek-e a törölt hozzávalók
 }
```

5. ábra getItemIndex függvény

ad vissza értékül így tudjuk, hogy ez egy új elemként fog a kosárba kerülni.

Az egyik legösszetettebb függvényünk a rendelések felvétele az adatbázisba. Ennek a kódnak a lefutásakor összesen öt táblába illesztünk be adatot annak érdekében, hogy később minden részletesen megjeleníthető legyen. Feljegyzésre kerül minden termék a rendelés helye és minden egyes termék módosítás amit a felhasználó végrehajt.

```
export async function POST(req: NextRequest) {
    const { userId, buffet_id, cartItems, price } = await req.json();
    const orderId = await query("INSERT INTO orders (user_id, buffet_id, price) VALUES (?, ?, ?)",
        [userId, buffet_id, price]).then(async (rawOrder: unknown) => {
        if (!rawOrder) {
            return NextResponse.json({ error: "Nem sikerült a rendelés felvétele!" });
        }
        const orderInfo = rawOrder as ResultSetHeader;
        return orderInfo.insertId;
    });

    cartItems.map(async (item: CartItem) => {
        const uniqueItemId = await query("INSERT INTO order_products (order_id, product_id, quantity) VALUES
            (?, ?, ?)", [orderId, item.id, item.quantity]).then(async (rawOrderProduct: unknown) => {
            if (!rawOrderProduct) {
                return NextResponse.json({ error: "Nem sikerült a rendelés felvétele!" });
            }
            const orderProductInfo = rawOrderProduct as ResultSetHeader;
            return orderProductInfo.insertId;
        });

        item.extras.map(extra => {
            query("INSERT INTO added_extras (order_product_id, order_id, product_extra_id) VALUES (?, ?, ?);",
            [uniqueItemId, orderId, extra]);
        });

        item.removedIngredients.map(ingredient => {
            query("INSERT INTO removed_ingredients (order_product_id, order_id, ingredient_id) VALUES (?, ?, ?);",
            [uniqueItemId, orderId, ingredient]);
        });
    });

    return NextResponse.json({ success: "Rendelés rögzítve!" });
}
```

6. ábra rendelés felvétele

A C sharp alkalmazás egyik legösszetetteb függvénye a beléptestés, melynek bonyolultsága a Next Auth azonosításra használt alkalmazás integrációjának megoldásából eredt. Alap esetben ezt csak

weboldalak biztosítására használják, de nem szerettem volna külön azonosítást írni amivel arra lyukadtam volna ki, hogy bizonyos kódokat fölöslegesen kellene megismételnem. Ahhoz, hogy az azonosítás működhessen kezelnem kellett a kommunikáció közben létrejött sütiket (cookie-kat).

```
HttpResponseMessage response = await new HttpClient().GetAsync($"http://localhost:3000/api/auth/csrf");
string csrf = response.Headers.GetValues("Set-Cookie").First().Split(";")[0].Split("=")[1];
dynamic content = JObject.Parse(await response.Content.ReadAsStringAsync());

var credentials = await WebCom.LoginPost(new Logins
{
    username = usernameInput.Text,
    password = passwordInput.Text,
    csrfToken = content["csrfToken"],
    redirect = false,
    json = true,
    callbackUrl = "http://localhost:3000"
}, csrf);
```

7. ábra a bejelentkezésre szolgáló Post kérés

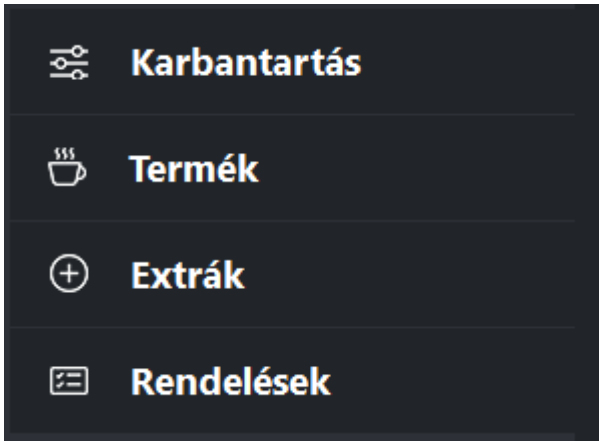
Az alkalmazást nem lehetséges használni amennyibe a web szerver nem fut, mivel azon keresztül kommunikál az adatbázissal.

A kód első sorában a webtől kérek egy Cross-Site Request Forgery-t (CSRF-et), amely segítségével tudunk küldeni egy olyan kérést is ami már egy felhasználót is visszaad egy session-el együtt amivel már letudjuk kérdezni az adatbázis adatait is.

Amennyiben a felhasználó nem rendelkezik büfével, nem használhatja a büfét erről egy üzenet tájékoztatja.

Belépés után a felhasználót a büfé aznapi adatai fogadják. Ebben a statisztikában csak azok az adatok jelennek meg, melyeknek státusza már teljesített.

A termékek kezelését a karbantartás menüponton belül éri el.



8. ábra Karbantartás fül

A „Termékek” menüpontra kattintva megjennek a büfé jelenlegi termékei. Ezen a fülön lehet akár keresni a termékre. A táblázatban kiválasztásra kerülő termék adatai jobb oldalon jelennek meg. A módosítások elvégzése után a mentés gombra kattintva frissíti az adatbázis illetve a táblázat adatait is. Ezen belül vannak olyan gombok is amik egy külön ablakot nyitnak meg (például.: „Extrák”). Ezekben az ablakokban történt módosítások nincsenek összefüggésben a termék oldalon lévő mentéssel. Ezek már korábban mentésre kerülnek!

6. Tesztelés

Az oldal működésének tesztelése kulcsfontosságú lépés volt a fejlesztés során. Kezdetben rendszeres manuális teszteket végeztünk, amelyek során kézzel ellenőriztük az oldal különböző funkcióit és szolgáltatásait. Ezek a tesztek segítettek az azonnali hibák azonosításában és javításában.

Ahogy haladtunk előre, bevezettünk automatizált tesztelést is. Ez lehetővé tette számunkra, hogy rendszeresen és hatékonyan ellenőrizzük az oldal különböző részeit, például az űrlapokat, a navigációt és az adatbázis működését. Az automatizált tesztek segítettek gyorsan reagálni az esetleges hibákra és biztosították, hogy az új funkciók bevezetése ne zavarja az oldal többi részét.

A folyamat részeként rendszeresen monitoroztuk az oldal teljesítményét is, hogy azonosítsuk és javítsuk az esetleges lassulásokat vagy egyéb problémákat.

A legtöbb tesztelést a büfé kosárrendszere igényelte, mivel igen komplex működése van. Kezdetben előfordultak olyan hibák amik miatt rosszul határoztam meg egy törlendő elem azonosítóját, így a nem megfelelő terméket törölt ki a rendszer a kosárból. Ezt úgy oldottam meg, hogy külön függvényt készítettem az index meghatározására, így egy elkülönített résszel már egy sokkal jobban átlátható kódot kaptam, aminek a tovább fejlesztése is egyszerűbb lett.

A tesztelések során egy olyan hibába ütköztünk amely azt okozza, hogy a feltöltött profilkép, nem töltődik be, ha előtte még nem volt profilképünk. Ez sajnos csak akkor tölt be ha ki és be jelentkezünk. A javítása további kutatást igényel a rendszerrel kapcsolatban.

A legtöbb problémát pedig a Form alkalmazást okozta, melybe beépítettük a weboldal biztonságáért szolgáló Next Auth szolgáltatást, hogy ne kelljen külön azonosítást készíteni az alkalmazáshoz kellő adatok lekérdezéséhez. Ennek a résznek az elkészítése rengeteg tesztelést igényelt, hogy egy webes hiba miatt az alkalmazás ne omoljon össze.

Összességében a folyamatos tesztelés és fejlesztés lehetővé tette számunkra, hogy egy stabil, megbízható és felhasználóbarát weboldalt és Form alkalmazást hozzunk létre, amely megfelel az elvárásoknak és igényeknek. Természetesen a tesztek sikeressége nem zárhatják ki az összes hibát, így azok előfordulhatnak.

3. Felhasználói dokumentáció, felhasználói kézikönyv

Az weboldalunk vendégként csak körbetekintés céljából használható. A rendelés és a profil kezelése már csak regisztrált felhasználóknak lehetséges.

1. Telepítési útmutató

Először is, győződjön meg róla, hogy telepítve van a Node.js és npm (Node Package Manager). A következő parancsokkal ellenőrizheti ezeket a függőségeket: „npm -v", „node -v".

Telepítse a projekt függőségeit a következő paranccsal: „npm i". Ez a parancs letölti és telepíti a projekt függőségeit.

Figyeljen a „.env" file tartalmára, hiánya esetén pótolja „.env.example" szerint.

Buildelje le a projectet a következő paranccsal: „npm run build". Ez a parancs létrehoz egy optimalizált verziót a next.js projektből.

Indítsa el a projektet a build után a következő paranccsal: „npm run start". Ez a parancs elindítja a buildelt next.js projektet, és alkalmassá teszi a weboldalt a böngészőben való megjelenítésre.

2. Vendég funkciók

A főoldal a vendégek és a regisztrált felhasználók számára is megegyezik. Egy átfogó képet ad a weboldal céljairól, használóról és rendelési statisztikát jelenít meg (letöbb rendeléssel rendelkező büfék, végbement rendelések száma).

Az oldalba beépítettük a sötét módot is ami alapértelmezetten a rendszer témájától függ (világos/sötét). A logó melletti gombbal viszont „véglegesen" is kilehet választani a nekünk tetsző témát.

A regisztráció és a bejelentkezés kis profilikonra kattintva érhető el. Bejelentkezés után ott érhető el a felhasználó profilja.

A kosár gombra kattintva az oldal átnavigál minket a büfék oldalára. Itt az összes olyan büfé megjelenik aminek legalább 1 terméke van. A keresési mező használatával kereshetünk a feltöltött büfék között.

Ha megtaláltuk a nekünk kellő büfét azt kiválasztva megjelennek számunkra a büfé elérhető termékei. Egy terméknek több státusza is lehet: normál, népszerű (trending), akciós, népszerű és akciós.

Akció esetén jelezzük azt egy felirattal, illetve megjelenítjük a régi és az új árát is. Trendingbe a büfé legnépszerűbb (legtöbbet vásárolt) 6 terméke kerülhet.

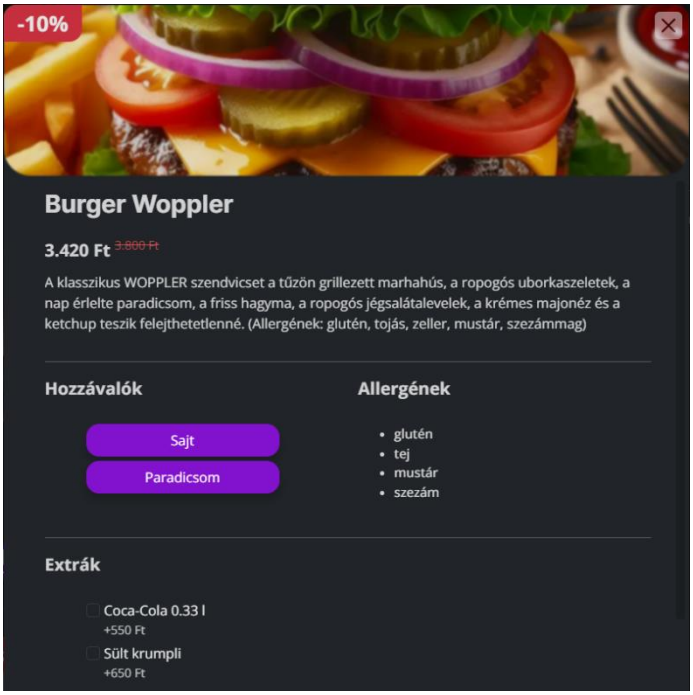
A termékekről részletes információkat már csak a regisztrált felhasználók kaphatnak, erről tájékoztatást akkor kapnak, amikor rákattintanak egy termékre, ekkor egy felugró üzenet jelzi nekik, hogy ehhez már bejelentkezve kell lennie.

9. ábra Akciós és népszerű termék megjelenése

3. Felhasználói funkciók

Amennyiben be van jelentkezve a felhasználó a büfések által feltöltött leírást láthatják, a termék allergénjeivel, a módosítható hozzávalókkal (pl.: sajt nélkül) és a hozzáadott opcionális extrákat (az extrákat szintén a büfés rendelheti hozzá).

Minden egyes módosítás azonnal megjelenik a termék összesítésében, újra számítja a termék árát extrákkal és jelzi milyen elemeket távolítottunk el belőle.



10. ábra termék részletes adatai

A módosítások nem csak ezen az ablakon jelennek meg, hanem a kosárban is részletes megjelenítést alkalmaztunk, így a felhasználó a termékek minden részletéről pontos leírást kapnak.

Amennyiben a kosárban csak egyetlen darab van egy megadott termékből, egy szemetes jelenik meg a plusz jellel szemben, viszont ha a termékből már több mint egy darab van egy mínuszjel jelenik meg helyette, ami jelzi, hogy a kattintással eltávolítjuk az t a kosárból, eltávolítás előtt erről egy felugró ablakban meg kell erősítenie döntését.

11. ábra A termékek megjelenése a kosárban módosított termékekkel

A rendelés megnyomása után a rendelés elküldésre kerül a büfés számára aki elkezdheti annak készítését. A rendelés leadásának sikerességéről egy kis felugró értesítést kap a felhasználó.

Ha a felhasználónak már van aktív megrendelése, akkor a rendelés nem történik meg és tájékoztatást kap arról, hogy folyamatban lévő rendelés esetén meg kell várnia míg az lezáródik.

A profilba belépve lehet módosítani a felhasználói adatokat (felhasználónév, jelszó, stb.), ezek módosításkor elsődlegesen kliensoldalon kerülnek ellenőrzésre és csak utána szerveroldalon (foglalt-e). A felhasználó az oldalsávban kinyitva a profil elemet, megtekintheti a rendelési előzményeit is amelyek állapotuk szerint kerülnek színkódolásra. A listanézetben a rendelésekből csak néhány infó látszik. Egy elemre kattintva pedig megtekinthetőek a rendelés részletes adatai is.

4. Büfés funkciók

A büféseknek külön C sharp Form alkalmazás készült ahol nyomon követhetik büféjük statisztikáit, kezelhetik a termékeiket és a hozzájuk tartozó információkat: extrák, hozzávalók stb.

A főoldalon köszöntő üzenetet és napi statisztikákat lehet megtekinteni. Minden olyan rendelés tartozik bele a statisztikába ami teljesítésre került. Így a valósághoz a lehető legközelebb adatokat kaphatják meg.

A baloldali menüsávba került minden a büfével kapcsolatos adat és opció. Létrehozhatunk új termékeket, részletes szerkesztési lehetőségekkel. Megadhatjuk a termék leírását, módosítható hozzávalóit, kategóriáját és hozzárendelhetünk extrákat is (például: sültkrumpli).

5. Továbbfejlesztési lehetőségek

A tervek összeírásakor rengeteg funkciót gyűjtöttünk össze, illetve fejlesztés közben is sok új ötletünk támadt, viszont figyelembe kellett vennünk a záros határidőt. Ki kellett választanunk a leglényegesebb funkciókat amik az oldal alapvető működéséhez fontosak. Sok helyen belemerültünk a részletekbe amik sajnos más funkciók lehetőségét vette el.

- WebSocket a rendelés valós idejű követéséhez:
  A WebSocket technológia lehetővé teszi, hogy a szerver valós időben kommunikáljon a kliensekkel anélkül, hogy újra és újra le kellene kérdezniük az információkat. Ezt a technológiát használnánk a rendelések állapotának frissítésére a felhasználók számára. Amint a rendelés állapota megváltozik, például elkészült, a szerver azonnal küld egy üzenetet a kliens felé. Így a felhasználó tudhatja, hogy átveheti rendelését.

- Diákokhoz való iskola rendelés a gördülékenyebb rendelés érdekében:
  Egy olyan rendszer fejlesztése, amely lehetővé teszi a diákok számára, hogy elmenthessék Ők melyik iskolához tartoznak, így a főoldalon lévő kosárgomb megnyomásakor nem a büfékeresőre, hanem egyből a büfé oldalára kerülne.
- Kupon rendszer, amit akár a büfések, akár a rendszer üzemeltetői (JösszBüfé) előállíthatnak:
  Egy integrált kupon rendszer segítségével a büfések és a rendszer üzemeltetői különféle kedvezményeket és promóciókat kínálhatnak. A felhasználók ezeket a kuponokat beválthatják a rendelések során.
- Fokozott biztonság, pl.: megerősítő e-mailek a komolyabb változásoknál:
  A biztonság fokozása érdekében bevezethetünk megerősítő e-maileket, amelyeket a felhasználóknak küldünk ki fontos fiókbeli változások, például jelszó módosítása esetén. Ezenkívül implementálhatunk egy kétlépcsős azonosítási rendszert is, amely további védelmet biztosít a felhasználói fiókok számára.
- Az API útvonalak korlátozása, az esetleges támadások ellen:
  Beállíthatunk egy korlátozást, hogy egy felhasználó csak korlátozott számú API-kérést küldhessen percenként. Ez megakadályozza az esetleges rosszindulatú használatot és túlterheléses támadásokat, így védelmet nyújtva a rendszerünk számára.
- Form alkalmazás továbbfejlesztése:
  Több beállítási funkció a büfések számára, például: büfé karbantartás, nyitvatartás megadása, rendelés átvételére alkalmas szünetek megadása.
- Statisztikák továbbfejlesztése:
  Részletes statisztikák a büfések számára, hogy nyomon követhessék a termékeik részletes adatait. Ezen kívül opció az időszak szűrésére.