



Széchenyi István Katolikus Technikum és Gimnázium

Szoftverfejlesztő és -tesztelő projektfeladat

PÉKSÉG

Készítették:

Vakherda Ádám,
Bolyky Máté,
Iróczki Regina

Ózd, 2025

Tartalomjegyzék

RENDSZERKÖVETELMÉNY	5
Szerveroldali követelmények	5
Kliensoldali követelmények	5
WEBALKALMAZÁS HASZNÁLATA/INDÍTÁSA	6
ALKALMAZOTT FEJLESZTŐI ÉS CSOPORTMUNKA ESZKÖZÖK	7
Adatbázis	9
Frontend	9
Backend	9
ADATBÁZIS	10
Modell leírása	10
TÁBLÁK, KAPCSOLATOK	14
ER/EK DIAGRAMM	15
FRONTEND DOKUMENTÁCIÓ	16
Fő navigációs menü	17
Regisztráció és bejelentkezés	18
Profilkezelés	20
Termékek	21
Kosár és rendelési folyamat	22
Korai Design Konceptciók	24
FŐBB KÓDRÉSZLETEK	27
TESZTELÉS	39
BACKEND DOKUMENTÁCIÓ	40
TESZTELÉS	46
getById:	46
Keresés név és típus alapján:	47
Update/Destroy:	48
GetHigher/Lowerthan:	49
FilterByFizetesimod:	50
Seeder:	51
Bejelentkezési (login):	52

addFelhasznalo:	52
TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK	53
IRODALOMJEGYZÉK	54

BEVEZETÉS

Projektünk során egy olyan pékség weboldalának megtervezésén dolgoztunk, amelynek célja, hogy napi szinten friss, kiváló minőségű pékáruval lássa el az oktatási intézményeket. Az ötlet alapját saját tapasztalataink adták, hiszen diákokként sokszor szembesültünk azzal, hogy az iskolai menzán vagy büfében a kínált kenyér, kifli vagy zsemle nem volt már igazán friss, mire elfogyasztásra került.

Úgy gondoljuk, hogy egy problémát akkor lehet igazán hatékonyan megoldani, ha az ember maga is megtapasztalta azt. Ezért választottuk ezt a témát: személyesen is érezzük, mennyire fontos lenne a friss pékáru biztosítása az iskolákban, óvodákban, egyetemeken. Célunk, hogy a weboldalunk segítségével az intézmények egyszerűen és gyorsan tudjanak megrendeléseket leadni, és biztosak lehessenek abban, hogy az általuk kapott termékek mindig frissek és ízletesek lesznek.

A weboldal lehetővé teszi a rendelési folyamat digitalizálását, átláthatóságát és automatizálását. A kínálat könnyen böngészhető, a rendelés személyre szabható, a szállítás pedig megbízható és rendszeres.

Jövőbeli terveink között szerepel, hogy a pékséget tovább fejlesztjük: bővítjük a kínálatot (pl. gluténmentes vagy vegán opciókkal), és más intézményeket – például kórházakat vagy irodaházakat – is bevonunk a rendszerbe. Hosszú távon egy olyan hálózatot szeretnénk létrehozni, amely nemcsak friss pékárut biztosít, hanem hozzájárul egy egészségesebb, élhetőbb mindennapi környezet kialakításához is.

Github elérhetőségének a linkje: <https://github.com/bolykymate/Pekseg>

RENDSZERKÖVETELMÉNY

Szerveroldali követelmények

- **Operációs rendszer:** Linux vagy Windows
- **Backend:** Node.js/PHP/Python alapú szerver
- **Adatbázis:** MySQL
- **Tárhely:** Minimum 50GB SSD tárhely
- **RAM:** Minimum 4GB RAM
- **Hálózati kapcsolat:** Minimum 100 Mbps internetkapcsolat

Kliensoldali követelmények

- **Böngésző támogatás:** Chrome, Firefox, Edge, Safari (legújabb verziók)

Minimális hardverigény

- **Asztali gép/laptop:** Legalább 4GB RAM, 2 magos CPU, internetkapcsolat
- **Mobil eszköz:** Android 8.0+ vagy iOS 12.0+ támogatása

Szoftver

- **VS Code** v1.96.0 vagy újabb
- **Node.js** v22.13.0 vagy újabb
- **XAMPP** 8.1.12 vagy újabb
- **Composer** 2.8.4 vagy újabb

WEBALKALMAZÁS HASZNÁLATA/INDÍTÁSA

Előfeltételek a webalkalmazás futtatáshoz/telepítéshez:

- XAMPP (Apache + MySQL) telepítése
- PHP (XAMPP részeként települ)
- Visual Studio Code telepítése
- Composer telepítése (Laravel függőségekhez)
- Angular CLI telepítése
- Node.js és npm telepítése (Angular futtatásához)

XAMPP konfigurálása:

1. Indítsa el a XAMPP Control Panelt, az Apache és MySQL szolgáltatásokat
2. Nyissa meg a <http://localhost/phpmyadmin> felületet

Laravel backend indítása:

1. Projektmappa megnyitása Visual studio Code-ban
2. Ellenőrizzük, hogy az env-ben a pekseg adatbázis van megadva
3. Új terminál nyitása
4. Adatbázis migrálása és feltöltése: „php artisan migrate –seed” parancssal
5. Töltsük le a pékárú tábla SQL fájlját a GitHub feltöltések közül, majd importáljuk az adatbázisba (pl. phpMyAdmin segítségével).
6. Laravel szerver indítása: „php artisan serve” parancssal
7. A terminálban ekkor megjelenik egy visszajelzés, amely mutatja, hogy a szerver fut
Ezután a backend elérhető lesz

Angular frontend indítása:

1. Projektmappa megnyitása Visual studio Code-ban
2. Új terminál nyitása
3. Függőségek telepítése: npm install
4. Fejlesztői szerver indítása: ng serve
5. A terminál visszajelzést ad a sikeres indításról
Ezután a frontend a terminálban megadott címen (alapértelmezetten <http://localhost:4200>) lesz elérhető

ALKALMAZOTT FEJLESZTŐI ÉS CSOPORTMUNKA ESZKÖZÖK

A munkánk során többféle eszközt is használtunk, amelyek segítettek a tervezésben, a fejlesztésben és abban, hogy csapatként hatékonyan tudjunk együtt dolgozni.

- **Figma**

A weboldal megtervezésénél a Figma nevű online tervezőprogramot használtuk. Ebben tudtuk előre megrajzolni/megtervezni, hogyan nézzen ki az oldal. Így mindenki látta, mi, hogy fog kinézni, és könnyebb volt megbeszélni az ötleteket.

- **Trello**

A feladatok nyomon követéséhez a Trello-t vettük igénybe. Ebben létre tudtunk hozni listákat és kártyákat, amikre felírtuk, kinek mi a dolga, és mi hol tart. Ez különösen jól jött, mert segített rendszerezni a feladatokat/teendőket.

- **VSC**

A fejlesztéshez a Visual Studio Code-ot használtuk, mivel jól támogatja a különböző programozási nyelveket és keretrendszereket, mint például a HTML, CSS, JavaScript, Angular és PHP. Egyszerű kezelhetősége miatt ideális választás volt számunkra a munka során.

- **GitHub**

A GitHubot a forráskód tárolására és verziókezelésére használtuk. Segített abban, hogy egyszerre tudjunk dolgozni, anélkül, hogy egymás munkáját felülírtuk volna. Minden változtatás követhető és biztonságosan kezelhető volt, így átlátható maradt a fejlesztés folyamata.

- **Discord**

A csapat közötti kommunikációra a Discordot használtuk. A platform/felület lehetőséget biztosított a gyors üzenetváltásra, kérdések gyors megválaszolására, valamint hang- és videóhívások lebonyolítására.

Adatbázis

Az adatbázist MySQL-ben készítettük el, amit a XAMPP-on belül a phpMyAdmin felületen keresztül kezeltünk. Itt hoztuk létre a táblákat a termékek, rendelések, iskolák és egyéb adatok számára.

Frontend

A frontend fejlesztése Angular keretrendszerben történt. Ez lehetővé tette a komponensalapú, fejlesztést és a dinamikus felhasználói élményt. A komponenseket TypeScript-ben írtuk, a stílusok pedig CSS segítségével lettek kialakítva.

Backend

A backend rész Laravel keretrendszerrel készült, PHP nyelven. A Laravel segítségével hoztuk létre azokat a REST API-kat, amiken keresztül az Angular frontend adatokat kér le vagy küld el. A backend feladata az adatbázis kezelése, az adatok ellenőrzése (validálása), valamint minden olyan szerveroldali működés, ami a háttérben zajlik, de fontos a rendszer működése szempontjából.

ADATBÁZIS

Az adatbázisunk azzal a céllal készült, hogy segítsen egy online rendelési rendszer működésében, amely pékáruk árusítására lett kitalálva. Az adatbázis segítségével könnyedén nyomon követhetők a rendelések fontos részletei, például a rendelés dátuma, a kiszállítás tervezett időpontja, valamint a választott fizetési mód. Ezen kívül a Pékáru tábla információkat tárol/nyújt a termékekről, beleértve azok nevét, típusát és árait. Az adatbázis tartalmazza a felhasználók adatait is, mint például a felhasználó nevét, jelszavát és email címét, melyek segítenek a vásárlók kezelésében. A címek tábla pedig azokat az információkat tartalmazza, amelyek a kézbesítéshez szükségesek, beleértve a felhasználók címét, a telefonszámukat és a számlázási nevet is, így biztosítva, hogy a rendelések pontosan és időben eljussanak a megfelelő címre.

Modell leírása

Pékáru

Mezők	Kulcs fajtája	Adattípus	Rövid leírás
Pid	PRIMARY KEY	int	A pékáruink elsődleges azonosítója
nev		varchar(20)	A pékáruk neve
típus		varchar(20)	A pékáruk fajtáját/típusát tárolja el
ar		int	A pékáruk ára

Felhasználók

Mezők	Kulcs fajtája	Adattípus	Rövid leírás
Fid	PRIMARY KEY	int	A felhasználók elsődleges azonosítója
nev		varchar(20)	A felhasználók nevét tárolja
jelszo		varchar(30)	A felhasználók jelszavát tárolja
email		varchar(30)	A felhasználók email címeit tárolja

Cím

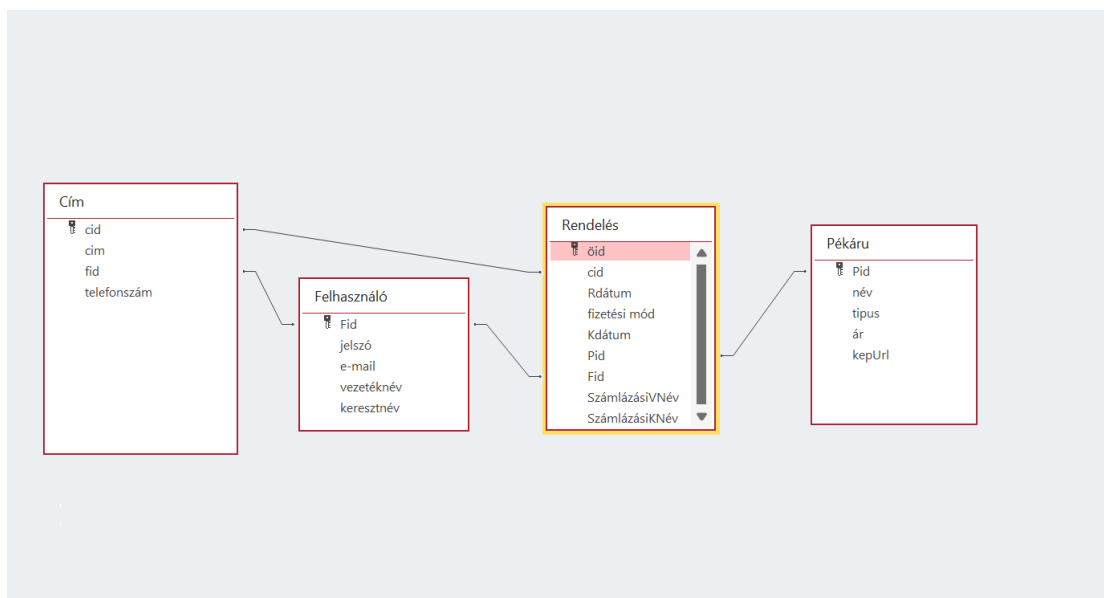
Mezők	Kulcs fajtája	Adattípus	Rövid leírás
Cid	PRIMARY KEY	int	A címtábla elsődleges azonosítója
cim		varchar(30)	A felhasználók teljes címét tárolja el
Fid	FOREIGN KEY		A felhasználók tábla mezőit meghívó idegenkulcs
SzamlazasiNev		varchar(30)	A felhasználók számlázási nevét tünteti fel
telefonszam		varchar(15)	A felhasználók telefonszámait tárolja el

Rendelés

Mezők	Kulcs fajtája	Adattípus	Rövid leírás
Rid	PRIMARY KEY	int	A rendelés azonosítója
Pid	FOREIGN KEY	int	A pékáru tábla mezőit meghívó idegenkulcs
Fid	FOREIGN KEY	int	A felhasználók tábla mezőit meghívó idegenkulcs
Cid	FOREIGN KEY	int	A címek tábla mezőit meghívó idegenkulcs
RDatum		varchar(20)	A rendelés dátumának ideje
KDatum		varchar(20)	A kiszállítás dátumának várható időpontja
FizetesiMod		boolean	A fizetési mód eldöntésére szolgál

TÁBLÁK, KAPCSOLATOK

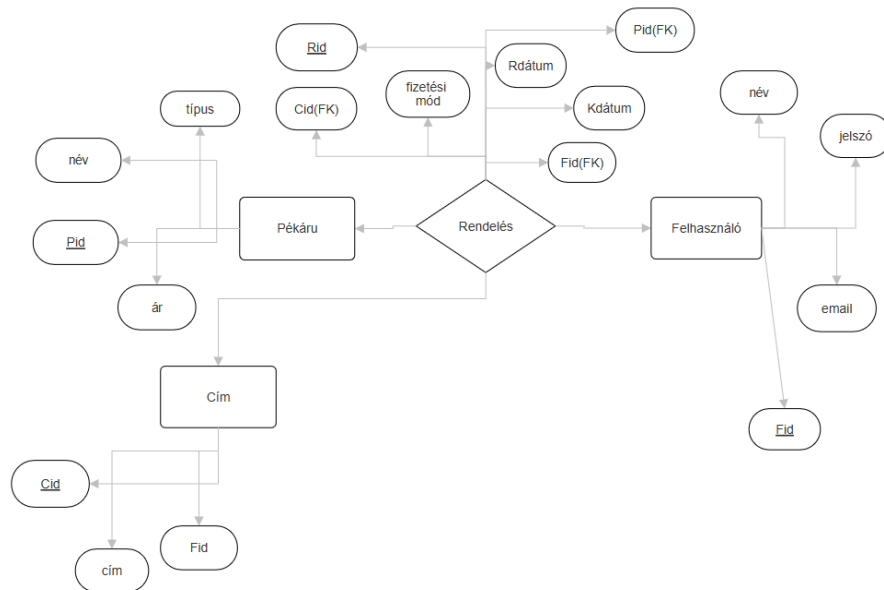
Az alábbi ábra az adatbázis táblái közötti kapcsolatokat mutatja be. Itt látható, hogyan kapcsolódnak egymáshoz a Cím, Felhasználó, Pékáru és Rendelés táblák idegen kulcsokon keresztül.



1. kép: A táblák közötti kapcsolatok ábrázolása.

ER/EK DIAGRAMM

Az alábbi ER-diagram szemlélteti, hogy a rendszer milyen adatokat tárol, ezek hogyan kapcsolódnak egymáshoz, és miként épül fel az adatbázis logikailag. Ez segít könnyebben átlátni a működését.

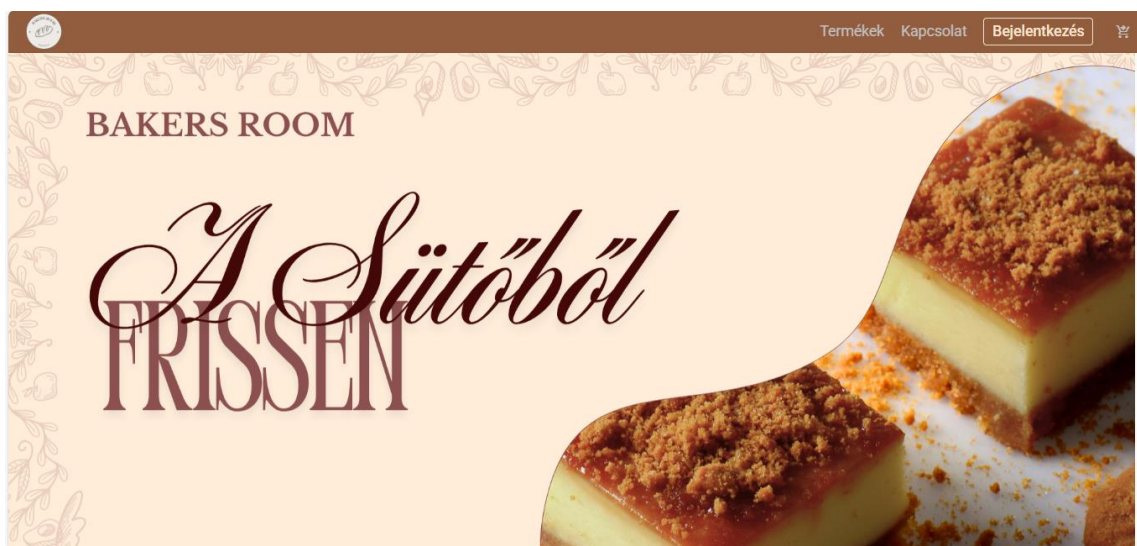


2. kép: Az entitások, azok jellemzői és a közöttük lévő kapcsolatok vizuális bemutatása.

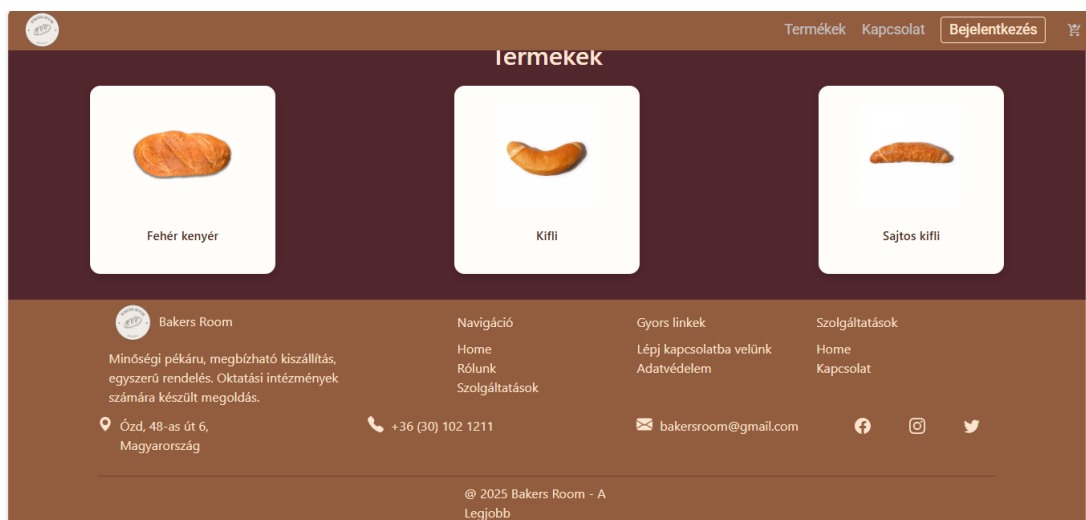
FRONTEND DOKUMENTÁCIÓ

A weboldal frontendjének fejlesztéséhez Angular keretrendszert használtunk, mert gyors, dinamikus és interaktív felhasználói élményt biztosít. Az Angular komponens alapú felépítése lehetővé teszi a kód újra felhasználását és könnyű karbantartását. Az Angular azért volt ideális választás a projekt számára, mivel egyszerre biztosít funkcionalitást és felhasználóbarát élményt. A webalkalmazás összesen 6 oldalból áll (Bejelentkezés/Regisztráció, Főoldal, Kosár, Rendelés véglegesítése, Termékek, és a Profil adatok), amikből elsőként a főoldalt tekinthetjük meg.

(FŐOLDAL KINÉZETE)



3. kép A főoldal felső része



4. kép A főoldal alsó része

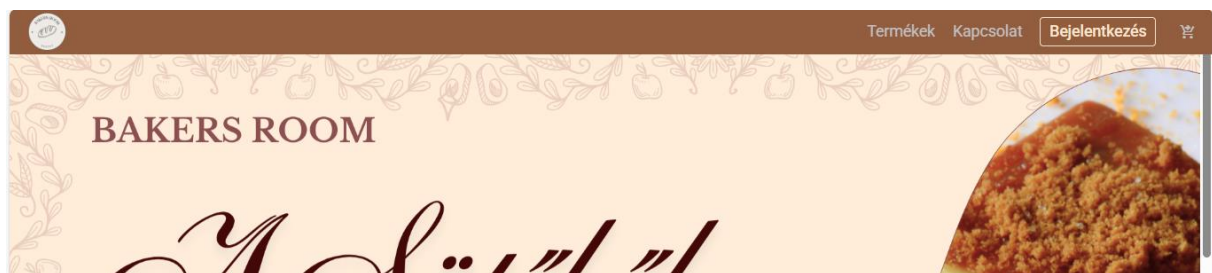
Navigációs rendszer

Fő navigációs menü

A weboldal tetején található fő navigációs sáv a következő lehetőségeket kínálja:

- Home/Logó: Visszavezet a kezdőlapra, ahol a legnépszerűbb termékek és a hirdetések láthatók
- Kapcsolat: A Bakers Room céggel kapcsolatos információk/elérhetőségek megtekintése
- Termékek: Az összes elérhető pékáru megtekintése kategóriák szerint
- Bejelentkezés/Regisztráció: Regisztrált felhasználók bejelentkezésére, vagy új felhasználók regisztrációjára szolgál
- Kosár: A felhasználó kosarában lévő termékek megtekintése és kezelése

A navigációs sáv minden oldalon konzisztensen megjelenik, lehetővé téve a gyors tájékozódást.



5. kép A minden oldalon feltüntetett navigációs sáv

Regisztráció és bejelentkezés

Regisztráció

A felhasználók számára elérhető egy regisztrációs űrlap, amely a következő kötelező mezőket tartalmazza:

- Vezetéknév
- Keresztnév
- E-mail cím
- Jelszó

Az űrlap alján elhelyezésre került egy „**Van már fiókja?**” link, amely a bejelentkezési oldalra navigálja a meglévő felhasználókat.

Regisztrációs felület

Vezetéknév*

Keresztnév*

Email*

Jelszó*

Regisztrálás

Bezárás

Van már fiókja? [Lépjen bel](#)

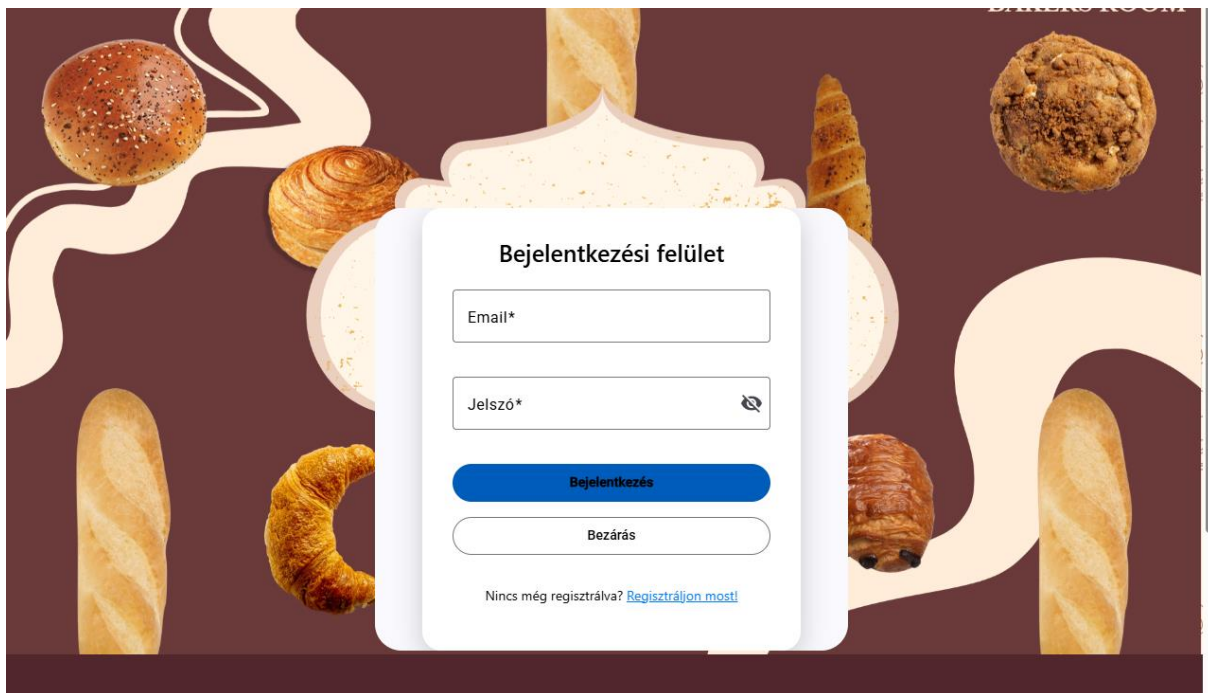
6. kép Regisztrációs felület

Bejelentkezés

A bejelentkezéshez a felhasználóknak az alábbi adatokat kell megadniuk:

- E-mail cím
- Jelszó

Amennyiben a felhasználó még nem rendelkezik fiókkal, a bejelentkezési felületen található „Nincs még regisztrálva?” link segítségével átirányítható a regisztrációs oldalra.

A screenshot of a login form titled "Bejelentkezési felület" (Login page). The form is centered on a dark brown background decorated with various breads and pastries. It contains two input fields: "Email*" and "Jelszó*" (Password*). Below the fields are two buttons: a blue "Bejelentkezés" (Login) button and a white "Bezárás" (Close) button. At the bottom, there is a link: "Nincs még regisztrálva? [Regisztráljon most!](#)".

7. kép Bejelentkezési felület

Bejelentkezetlen állapotban

"Bejelentkezés" gomb vezet a bejelentkező felületre

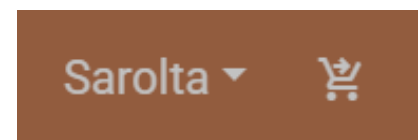
"Regisztráció" lehetőség a bejelentkezési oldalon

Bejelentkezett állapotban

A "Bejelentkezés" helyén a felhasználó neve jelenik meg dropdown menüvel



8.kép Nézet, ha a felhasználó nincs bejelentkezve



9. kép Nézet, ha be van jelentkezve


Profilkezelés

A felhasználói profil oldalon a regisztrált felhasználók megtekinthetik és kezelhetik személyes adataikat. Az oldalon az alábbi főbb funkciók érhetők el:

A rendszer lehetővé teszi a felhasználók számára, hogy megtekintsék és szükség esetén módosítsák az alapvető adataikat, beleértve például a vezetéknév- és keresztnévet, email címet, telefonszámot és lakcím információkat. A kötelezően kitöltendő mezőket csillag (*) jelzi.

A felhasználók bármikor frissíthetik adataikat. A módosítások véglegesítéséhez az "Adatok mentése" gombra kell kattintani, ezzel az adatok el lesznek mentve az adatbázisba.

Profil adatok

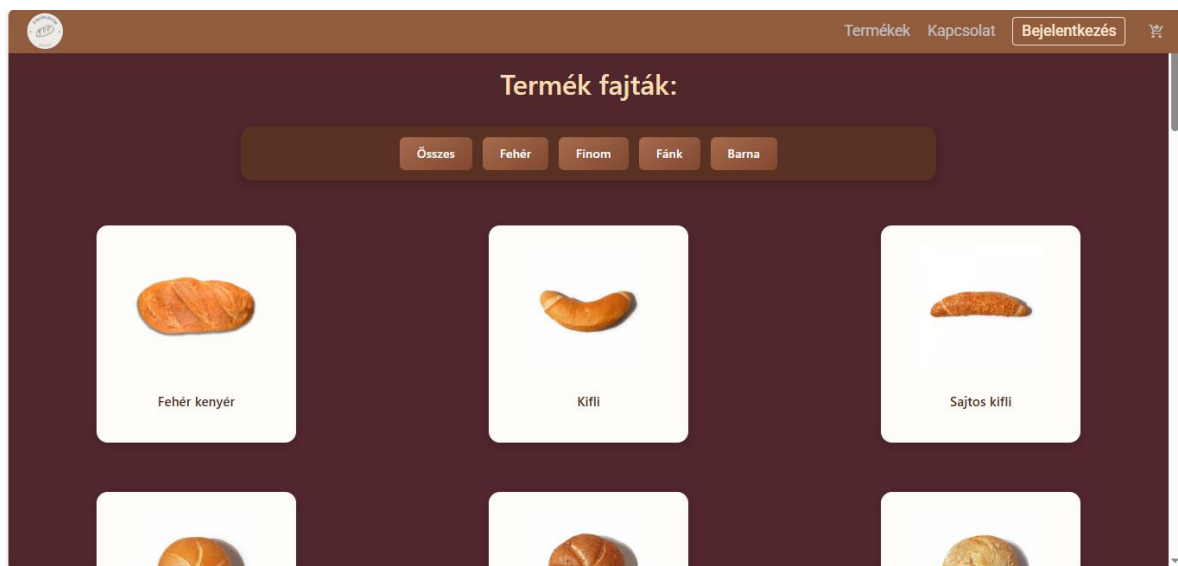
Vezetéknév* Kovács	Keresztnév* Sarlota
Email* kovacssarlota@gmail.com	Jelszó* 
Telefonszám* 06307654321 PI: 06301234567	Cím* Abigél utca PI: Példa utca 1

Adatok mentése

10. kép A profil adatok kezelésére szolgáló felület

Termékek

A termékoldalon egyszerűen szűrhetjük a pékárukat típus szerint. Az oldal tetején lévő gombok segítségével gyorsan lehet váltani a különböző termékcsoporthoz között. Az "Összes" gomb minden terméket megmutat, a "Fehér" gomb csak a fehér lisztes péksüteményeket, például a fehér kenyeret. A "Finom" gombra kattintva a vajjas, lágy kiflik és hasonló termékek jelennek meg. A "Fánk" gomb a süteményeket, a "Barna" gomb pedig a teljes kiőrlésű és barna lisztes pékárukat mutatja. A kategóriaválasztó terület kifejezetten felhasználóbarát kialakítású. A kiválasztott kategória mindig jól látható - külön színnel vagy kiemeléssel jelölve. Amikor a felhasználó másik kategóriára kattint, a terméklista azonnal frissül, így nincs várakozási idő.



11. kép A pékség termékei kilistázva egy külön felületen

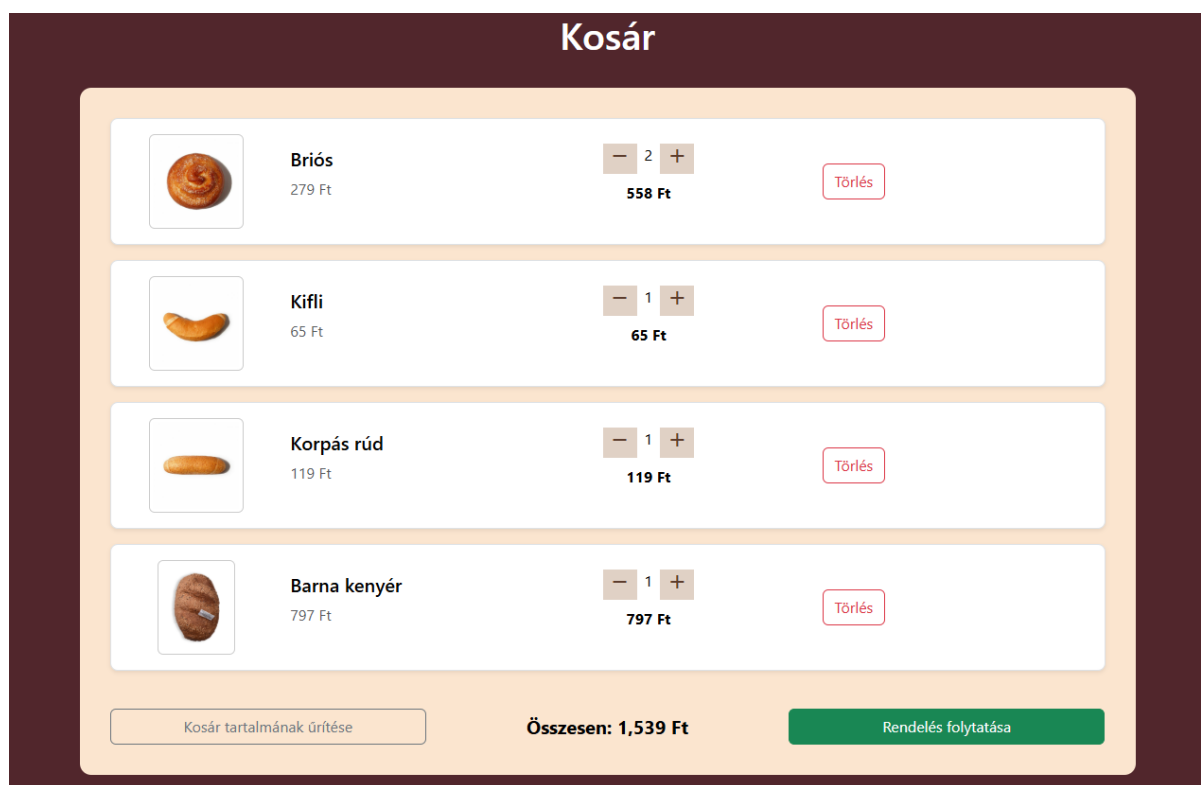
Fontos megjegyezni, hogy a kosárba helyezés funkció csak bejelentkezett felhasználók számára érhető el. Ez biztosítja, hogy csak hitelesített felhasználók tudjanak rendelést leadni.

Kosár és rendelési folyamat

Kosár

A kosár oldalon a felhasználók áttekinthetik a kiválasztott termékeket és módosíthatják a rendelés részleteit. A kosár felülete a következő funkciókat tartalmazza:

- A kosárban szereplő termékek listázása, darabszám és egységár feltüntetésével.
- A darabszám módosításának lehetősége minden termék esetében.
- Egyedi Törlés gomb minden termék mellett, a termék eltávolításához.
- Az összesített végösszeg automatikus megjelenítése és frissítése a kosár tartalmának változásakor.
- „Rendelés folytatása” gomb, amely elindítja a rendelési folyamat következő lépését (fizetési és szállítási adatok megadása).



12. kép Kosár felülete

Rendelés véglegesítése

A rendelés véglegesítésekor a felhasználó egy összefoglaló oldalon ellenőrizheti, hogy milyen termékeket tett a kosárba, hány darabot választott, és mekkora az összesített ár. Ezen a felületen szükség van néhány számlázási adat megadására is, például a vezetéknévre és a keresztnévre. Aki már regisztrált, az egyszerűen felhasználhatja a korábban megadott adatait. A rendelés során ki lehet választani a kiszállítás dátumát, valamint meg kell jelölni, hogy a vásárló bankkártyával vagy készpénzzel szeretné kifizetni a rendelést. Végül a „Rendelés elküldése” gombbal lehet leadni a megrendelést.

A rendelés véglegesítése

Briós
Darabszám: 2
Ár: 558 Ft (2 * 279)

Végösszeg: 558 Ft

☐ Regisztrált név használata

06307654321

Abigél utca

Kérjük válassza ki a kiszállítás időpontját

MM/DD/YYYY

Fizetési módok a helyszínen

☐ Kártyás fizetés

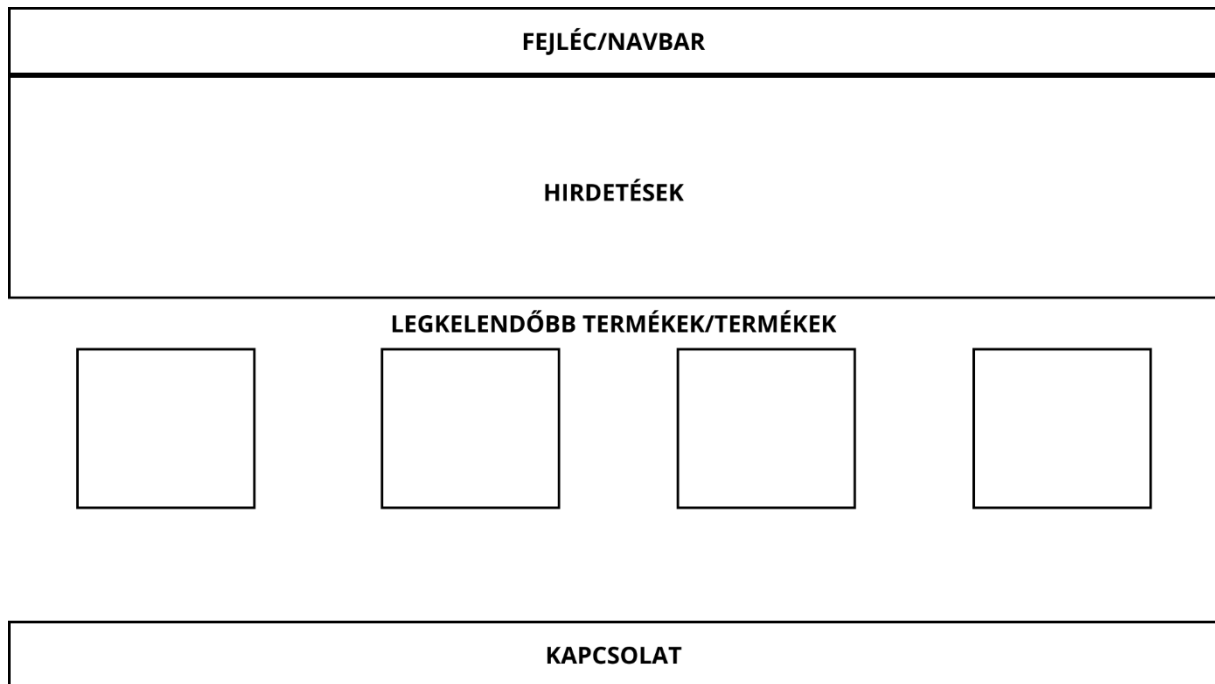
☒ Készpénz

13. kép A rendelés véglegesítését lebonyolító felület

Korai Design Konceptiók

Az alábbiakban bemutatjuk a korai design-terveket/konceptiókat, amelyekből később a jelenlegi, kiforrott felület született. Ezek a vázlatok mutatják, hogyan alakult az oldal a kezdeti ötletektől a végleges megoldásig.

FŐOLDAL KONCEPCIÓJA



14. kép Főoldal design koncepciója

A korai vázlat egy egyszerű fejléctet (navbar) és három fő szekciót mutat be:

- **HIRDETÉSEK:** Akciók, újdonságok vagy promóciók számára kijelölt terület.
- **LEGKELENDŐBB TERMÉKEK/TERMÉKEK:** A kiemelt vagy legnépszerűbb termékek listája.
- **KAPCSOLAT:** Alapvető elérhetőségek vagy üzenetküldési lehetőségek.

BEJELENTKEZÉSI FELÜLET KONCEPCIÓJA




The image shows two side-by-side wireframe diagrams for a web application. The left diagram is titled 'BEJELENTKEZÉS' (Login) and contains two horizontal input fields stacked vertically, followed by a single horizontal input field at the bottom. The right diagram is titled 'REGISZTRÁCIÓ' (Registration) and contains four horizontal input fields stacked vertically, followed by a single horizontal input field at the bottom. Both diagrams have a thick horizontal line at the very bottom, likely representing a footer or a separator.

15. kép Bejelentkezés és Regisztrációs felület design koncepciója

A kezdeti tervekben a bejelentkezési és regisztrációs felület extrém minimalista megközelítést kapott. A későbbi iterációkban ez a koncepció jelentősen átalakult. Hozzáadásra kerültek a formázott gombok, színkódolás, hover állapotok, és ezek mellett még a felhasználói élményt fokozó további funkciók.

Ez a fejlődés mutatja, hogy egy jó weboldal tervezése folyamatos finomítást igényel. A kezdeti egyszerű ötletekből alakul ki végül a kész, könnyen használható felület.

KOSÁR KONCEPCIÓJA

FEJLÉC/NAVBAR				
KOSÁR				
	TERMÉK NEVE		ÁR	+DB -
	TERMÉK NEVE		ÁR	+DB -
	TERMÉK NEVE		ÁR	+DB -
KAPCSOLAT				

16. kép Kosár design koncepciója

Ez a kosár oldal kezdeti vizuális koncepcióját mutatja be. A cél a funkcionalitás/használhatóság alapjainak lefektetése és az elrendezés irányának meghatározása a további részletezés előtt.

Tartalma

A központi blokkban/részben jelennek meg a kosárba helyezett termékek. Minden egyes tétel három fő részből áll:

- Termékkép: Vizualizáció a gyors azonosításhoz.
- Termék neve: Rövid szöveges megnevezés a kép mellett.
- Ár és Darabszám-módosító: Jobb oldalon elhelyezve, az ár megjelenítése mellett plusz és mínusz gombok biztosítják az egyszerű darabszám-változtatást.

FŐBB KÓDRÉSZLETEK

```
4      <h1>Termék fajták:</h1>
5    </div>
6    <nav>
7      <button (click)="getAll()" class="osszes">Összes</button>
8      <button class="tipusok" *ngFor="let t of tipusok" [ngClass]="{'active-btn': t === selectedType}" (click)="getTermekByTypes(t); selectedType = t">{{t}}</button>
9    </nav>
10   <div *ngIf="showProducts">
11     <div class="row sorok">
12       <div class="kartyak col-12 col-sm-6 col-md-4 col-lg-4" *ngFor="let aruk of termekek3">
13         <div class="card termekek-card" style="width: 18rem;">
14           <img class="card-img-top image-fluid" [src]="aruk?.kepUrl" alt="Card image cap">
15           <div class="card-body">
16             <h5 class="card-title">{{aruk?.nev}}</h5>
17             <a *ngIf="auth.LoggedInStatus(); else notLoggedIn" class="btn kosar-btn btn-primary" (click)="openSnackBar()" (click)="kosarba(aruk)">Kosárba</a>
18           </div>
19         </div>
20       </div>
21     </div>
22   </div>
23   <div *ngIf="showProducts2">
24     <div *ngIf="termekek2?.length !== 0; else elseBlock">
25       <div class="row sorok">
26         <div class="kartyak col-12 col-sm-6 col-md-4 col-lg-4" *ngFor="let aruk of termekek2">
27           <div class="card termekek-card" style="width: 18rem;">
28             <img class="card-img-top image-fluid" [src]="aruk?.kepUrl" alt="Card image cap">
29             <div class="card-body">
30               <h5 class="card-title">{{aruk?.nev}}</h5>
31               <a *ngIf="auth.LoggedInStatus(); else notLoggedIn" class="btn kosar-btn btn-primary" (click)="openSnackBar()" (click)="kosarba(aruk)">Kosárba</a>
32             </div>
33           </div>
34         </div>
35       </div>
36     </div>
37   </div>
38
39   <ng-template #notLoggedIn>
40     <div><p></p></div>
41   </ng-template>
42   <ng-template #elseBlock>
43     <div class="row sorok">
44       <div class="kartyak col-12 col-sm-6 col-md-4 col-lg-4" *ngFor="let aruk of termekek">
45         <div class="card termekek-card" style="width: 18rem;">
46           <img class="card-img-top image-fluid" [src]="aruk?.kepUrl" alt="Card image cap">
47           <div class="card-body">
48             <h5 class="card-title">{{aruk?.nev}}</h5>
49             <a *ngIf="auth.LoggedInStatus(); else notLoggedIn" class="btn kosar-btn btn-primary" (click)="openSnackBar()" (click)="kosarba(aruk)">Kosárba</a>
50           </div>
51         </div>
52       </div>
53     </div>
54   </ng-template>
```

17. kép Termékek komponens html

A fenti képen a `termekek.component.html` kódrészlete látható, amely a webalkalmazásban a termékek listázásáért és kosárba helyezéséért felel. Az oldal elején egy fejléc található "Termék fajták" címmel, majd navigációs gombok következnek, amelyek segítségével a felhasználó az összes termék megtekintése mellett szűrhet is terméktípusok szerint. A kiválasztott típus gombja kiemelésre kerül az `active-btn` osztály segítségével.

A termékek kártyák formájában jelennek meg, amelyeken látható a termék képe, neve, valamint egy "Kosárba" gomb is. Ez a gomb csak akkor jelenik meg, ha a felhasználó be van jelentkezve, amit az `auth.LoggedInStatus()` függvény vizsgál.

TERMÉKEK

```
export class TermekComponent {
  termek: Pekar[] = [];
  termek2?: Pekar[] = [];
  termek3: Pekar[] = [];
  tipusok: string[] = [];
  showProducts = false;
  showProducts2 = true;
  selectedType: string | null = null;
  constructor(private service: SzolgalatasService, protected auth: AuthService) {}

  _snackBar = inject(MatSnackBar);

  durationInSeconds = 3;

  openSnackBar() {
    this._snackBar.openFromComponent(SnackbarComponent, {
      duration: this.durationInSeconds * 1000,
    });
  }

  ngOnInit() {
    this.service.getPekar().subscribe(data => {
      {
        this.termek = data;
        console.log(this.termek);
        this.getTypes();
      });
    });
  }

  getAll() {
    this.service.getPekar().subscribe(data => {
      this.termek3 = data;
      this.showProducts = true;
      this.showProducts2 = false;
      this.selectedType = null;
    });
  }

  getTypes() {
    this.tipusok = [...new Set(this.termek.map(t => t.tipus))].filter(c => c)
  }

  getTermekByTypes(tipus:string) {
    this.termek2 = this.termek.filter(T => T.tipus === tipus);
    this.showProducts = false;
    this.showProducts2 = true;
  }

  kosarba(termek: Pekar) {
    this.service.addToKosar(termek);
  }
}
```

18. kép Termékek komponens ts

METÓDUSOK

ngOnInit()

- A komponens inicializálásakor fut le
- Betölti az összes terméket a SzolgalatasService segítségével
- A betöltött adatokat elmenti a termekek tömbbe
- Meghívja a getTypes() metódust a terméktípusok összegyűjtéséhez

getAll()

- Újra betölti az összes terméket a szervízből
- Az adatokat a `termek3` tömbbe menti
- Beállítja a megjelenítési flag-eket:
- `showProducts = true` (összes termék látszik)
- `showProducts2 = false` (szűrt lista nem látszik)
- Nullra állítja a kiválasztott típust

getTypes()

- Kigyűjti az összes egyedi terméktípust a `termek` tömbből
- Az üres értékeket kiszűri
- Az eredményt a `tipusok` tömbbe menti

getTermekByTypes(típus: string)

- Szűri a termékeket a megadott típus alapján
- Az eredményt a `termek2` tömbbe menti
- Beállítja a megjelenítési flag-eket:
 - `showProducts = false` (összes termék nem látszik)
 - `showProducts2 = true` (szűrt lista látszik)

kosarba(termek: Pekar)

- Hozzáadja a kiválasztott terméket a kosárhoz a szervíz segítségével

openSnackBar()

- Megjelenít egy értesítést (snackbar) a felhasználónak 3 másodpercig
- Egy külön `SnackBarComponent`-et használ ehhez

KOSÁR

```
12 @Component({
13   selector: 'app-kosar',
14   imports: [HeaderComponent, FooterComponent, NgFor, NgIf, FormsModule, MatIconModule, RouterLink, CommonModule],
15   templateUrl: './kosar.component.html',
16   styleUrls: ['./kosar.component.css']
17 })
18 export class KosarComponent {
19   osszeg: number = 0;
20   kosarTartalom: KosarElem[] = [];
21   protected actualCim?:Cim;
22
23   constructor(private szolgaltatas: SzolgaltatasService, private auth: AuthService) {
24     this.auth.getUserCimFromApi().subscribe(felhasznaloCim => {
25       if (felhasznaloCim) {
26         this.actualCim = felhasznaloCim;
27         console.log("Felhasználó címe:", this.actualCim.cim);
28       } else {
29         console.log("Nincs elementett cím.");
30       }
31     });
32     this.szolgaltatas.getKosar().subscribe((kosar) => {
33       this.kosarTartalom = kosar;
34       this.osszegSzamolas();
35     });
36   }
37
38   osszegSzamolas() {
39     this.osszeg = 0;
40     for (let elem of this.kosarTartalom) {
41       this.osszeg += elem.termek.ar * elem.darab;
42     }
43   }
44
45   novel(termek: PekarU) {
46     this.szolgaltatas.increaseQuantity(termek);
47     this.osszegSzamolas();
48   }
49
50   csokkent(termek: PekarU) {
51     this.szolgaltatas.decreaseQuantity(termek);
52     this.osszegSzamolas();
53   }
54
55   uritKosarat() {
56     this.szolgaltatas.clearKosar();
57     this.osszeg = 0;
58   }
59
60   torolElem(termek: PekarU) {
61     this.szolgaltatas.removeFromKosar(termek);
62     this.osszegSzamolas();
63   }
64 }
```

19. kép Kosár komponens ts

Ez a komponens egy bevásárlókosarat valósít meg, ahol a felhasználó megtekintheti, módosíthatja és kezelheti a kosár tartalmát.

METÓDUSOK

osszegSzamolas()

- Nullázza az összeget
- Végigmegy a kosár tartalmán
- Minden elemnél hozzáadja a termék árának és darabszámának szorzatát az összeghez

novel(termek: Pekaru)

- Növeli a kiválasztott termék darabszámát a kosárban
- Újraszámolja a kosár összértékét

csokkent(termek: Pekaru)

- Csökkenti a kiválasztott termék darabszámát a kosárban
- Újraszámolja a kosár összértékét

uritKosarat()

- Kiüríti a teljes kosarat
- Nullázza az összeget

torolElem(termek: Pekaru)

- eltávolítja a kiválasztott terméket a kosárból
- Újraszámolja a kosár összértékét

FŐOLDAL

```
<div class="page-container">
  <app-header></app-header>
  <div class="content-wrap">
    <div class="row" id="carouselId">
      <div>
        <div class="row">
          <div id="carouselExampleAutoplaying" class="carousel slide" data-bs-ride="carousel">
            <div class="carousel-inner">
              <div class="carousel-item active">
                
              </div>
              <div class="carousel-item">
                
              </div>
              <div class="carousel-item">
                
              </div>
            </div>
          </div>
        </div>
      </div>

      <div class="row cardok">
        <h1 class="text-center mt-5">Legfelkapottabb termékeink</h1>
        <h1 class="text-center mt-5"><a routerLink="/termekek">Termékek</a></h1>
        <div *ngFor="let t of termekek" class="card termekek-card" style="width: 18rem;">
          <img [src]="t.kepUrl" class="card-img-top img-fluid" alt="...">
          <div class="card-body">
            <h5 class="card-title text-center">{{t.nev}}</h5>
            <p *ngIf="auth.LoggedInStatus();else notLoggedIn" class="text-center"><button class="btn kosar-btn btn-primary" (click)="openSnackBar()" (click)="kosarba(t)">Kosárba</button></p>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
<app-footer></app-footer>
</div>

<ng-template #notLoggedIn>
  <div><p></p></div>
</ng-template>
```

20. kép Főoldal komponens html

Ez egy Angular komponens HTML szekciója/része, ami egy főoldalt valósít meg termékek megjelenítésével és egy képváltó carousellel.

Bootstrap carousel komponens automatikus lapozással (data-bs-ride="carousel") 3 hirdetést/képet jelenít meg egymás után. A hirdetések után következő szekció tetején található egy link a teljes terméklistára ("/termekek" útvonalon).


```

@Component({
  selector: 'app-fooldal',
  imports: [RouterLink, HeaderComponent, FooterComponent, NgFor, NgIf],
  templateUrl: './fooldal.component.html',
  styleUrls: ['./fooldal.component.css']
})
export class FooldalComponent {
  tipusok: string[] = [];
  termekek: Pekaruk[] = [];

  constructor(private service: SzolgaltatasService, protected auth: AuthService) {
    this.service.getPekaru().subscribe(data =>
      {
        this.termekek = data;
        console.log(this.termekek);
        this.getFeherTypes();
      });
  }

  _snackBar = inject(MatSnackBar);

  durationInSeconds = 3;

  openSnackBar() {
    this._snackBar.openFromComponent(SnackbarComponent, {
      duration: this.durationInSeconds * 1000,
    });
  }

  kosarba(termek: Pekaruk) {
    this.service.addToKosar(termek);
  }

  getFeherTypes() {
    let feherTermekek = this.termekek.filter(t => t.tipus === 'Fehér');
    this.termekek = feherTermekek.slice(0, 3);
    this.tipusok = [...new Set(this.termekek.map(t => t.tipus))].filter(c => c)
  }
}

```

21. kép Főoldal komponens ts

METÓDUSOK:

`openSnackBar()`

- Funkció: Megjelenít egy értesítést (snackbar) 3 másodpercig
- Használat: A kosárba helyezés sikerét jelzi

`kosarba(termek: Pekarú)`

- Funkció: Hozzáad egy terméket a kosárhoz a szervíz segítségével
- Paraméter: termék - A hozzáadandó termék

`getFehérTypes()`

- Kiszűri a 'Fehér' típusú termékeket
- Csak az első 3 ilyen terméket tartja meg
- Kigyűjti az egyedi terméktípusokat a típusok tömbbe

BEJELENTKEZÉS

```
<div class="login">
  <h2 class="cim">Bejelentkezési felület</h2>

  <form>
    <mat-form-field appearance="outline" class="field">
      <mat-label>Email</mat-label>
      <input matInput type="email" [(ngModel)]="user.email" name="email" required #email="ngModel" />
      <mat-error *ngIf="email.invalid && email.touched">Nem megfelelő az email</mat-error>
    </mat-form-field>

    <mat-form-field appearance="outline" class="field">
      <mat-label>Jelszó</mat-label>
      <input matInput [type]="hidePassword ? 'password' : 'text'" [(ngModel)]="user.jelszo" name="password" required #password="ngModel" />
      <button mat-icon-button matSuffix (click)="hidePassword = !hidePassword" [attr.aria-label]="Jelszó megjelenítése/elrejtése" [attr.aria-pressed]="!hidePassword">
        <mat-icon>{{ hidePassword ? 'visibility_off' : 'visibility' }}</mat-icon>
      </button>
      <mat-error *ngIf="password.invalid && password.touched">Legalább 8 karakter hosszú legyen a jelszó</mat-error>
    </mat-form-field>

    <button mat-flat-button color="primary" class="bejelentg" (click)="login()">Bejelentkezés</button>
    <button mat-stroked-button class="bezaras" (click)="close()">Bezárás</button>
  </form>

  <div class="regiszt">
    <p>Nincs még regisztrálva? <a (click)="openModal2(); close()">Regisztráljon most!</a></p>
  </div>
</div>
```

22. kép Bejelentkezési felület html kódja

A bejelentkezési oldalon két mező található: e-mail cím és jelszó. Ezek Angular Material elemek segítségével vannak megjelenítve, így az űrlap letisztult és egységes kinézetet kap. Az e-mail mező csak akkor érvényes, ha valódi e-mail címet írunk be, míg a jelszónál legalább 8 karaktert kell megadni.

A jelszó mező mellett van egy kis ikon, amivel be lehet kapcsolni a jelszó megjelenítését vagy elrejtését, attól függően, hogy mit szeretne a felhasználó.

Az űrlap alján két gomb található: a „Bejelentkezés” gomb, ami elindítja a bejelentkezési folyamatot, és egy „Bezárás” gomb, amivel az ablak bezárható.

Az oldal alján látható még egy szöveg is, ami a regisztrációra hívja fel a figyelmet. Ha valaki erre kattint, egy felugró ablakban tud új fiókot létrehozni.

```

export class ModalComponent {
  hidePassword = true;
  constructor(
    public dialogRef: MatDialogRef<ModalComponent>,
    @Inject(MAT_DIALOG_DATA) public data: { message: string }, public dialog: MatDialog, private overlay: Overlay
  ) {}

  close() {
    this.dialogRef.close();
  }

  openModal2() {
    if (window.innerWidth > document.documentElement.clientWidth) {
      document.body.style.overflow = 'hidden';
    }

    const dialogRef = this.dialog.open(RegistrComponent, {
      width: '500px',
      backdropClass: 'transparent-backdrop',
      disableClose: false,
      autoFocus: false,
      scrollStrategy: this.overlay.scrollStrategies.noop()
    });

    dialogRef.afterClosed().subscribe(() => {
      if (document.body.style.overflow === 'hidden') {
        document.body.style.overflow = 'auto';
      }
    });
  }

  user: User = {vezeteknev: '', keresztnév: '', jelszo: '', email: '' };
  message: string = '';

  login() {
    this.auth.login(this.user).subscribe({
      next: (user) => {
        if (user) {
          this.message = 'Sikeres bejelentkezés!';
          this.close();
        }
      },
      error: (err) => {
        this.snackBar.openFromComponent(LoginSnackBarComponent, {
          duration: 3000,
          horizontalPosition: 'center',
          verticalPosition: 'top',
          data: { message: 'Hibás felhasználónév vagy jelszó!' },
          panelClass: ['custom-snackbar']
        });
      }
    });
  }
}

```

23.kép A Bejelentkezési felület ts-e

A komponens felelős a bejelentkezési felület megjelenítéséért modálablakban. A felhasználó User típusú objektumba viszi be az adatait (vezetéknév, keresztnév, email, jelszó), majd a login() metódus ellenőrzi azokat. Sikeres bejelentkezés esetén bezáródik az ablak, hibás adatok esetén figyelmeztető üzenet jelenik meg. A hidePassword változó szabályozza, hogy a jelszó mező szövege látható legyen-e. A openModal2() metódus megnyitja a regisztrációs felületet egy újabb modálablakban, és kezeli a görgetés letiltását is a háttérben, amíg a modál aktív.

REGISZTRÁCIÓ

```
<div class="register">
  <h2 class="cim">Regisztrációs felület</h2>

  <form>
    <mat-form-field appearance="outline" class="fielddek">
      <mat-label>Vezetéknév</mat-label>
      <input matInput type="text" [(ngModel)]="user.vezeteknev" name="vname" required #vname="ngModel" />
      <mat-error *ngIf="vname.invalid && vname.touched">A név megadása kötelező!</mat-error>
    </mat-form-field>

    <mat-form-field appearance="outline" class="fielddek">
      <mat-label>Keresztnév</mat-label>
      <input matInput type="text" [(ngModel)]="user.keresztnev" name="kname" required #kname="ngModel" />
      <mat-error *ngIf="kname.invalid && kname.touched">A név megadása kötelező!</mat-error>
    </mat-form-field>

    <mat-form-field appearance="outline" class="fielddek">
      <mat-label>Email</mat-label>
      <input matInput type="email" [(ngModel)]="user.email" name="email" required #email="ngModel" />
      <mat-error *ngIf="email.invalid && email.touched">Nem megfelelő az email</mat-error>
    </mat-form-field>

    <mat-form-field appearance="outline" class="fielddek">
      <mat-label>Jelszó</mat-label>
      <input matInput [type]="hidePassword ? 'password' : 'text'" type="password" [(ngModel)]="user.jelszo" name="password" required minlength="8" #password="ngModel" />
      <button mat-icon-button matSuffix (click)="hidePassword = !hidePassword" [attr.aria-label]="Jelszó megjelenítése/elrejtése" [attr.aria-pressed]="!hidePassword">
        <mat-icon>{{ hidePassword ? 'visibility_off' : 'visibility' }}</mat-icon>
      </button>
      <mat-error *ngIf="password.errors?.['required'] && password.touched">A jelszó megadása kötelező!</mat-error>
      <mat-error *ngIf="password.errors?.['minlength'] && password.touched">Legalább 8 karakter hosszú legyen a jelszó!</mat-error>
    </mat-form-field>

    <button mat-flat-button color="primary" class="regisztralas" (click)="registration()">Regisztrálás</button>
    <button mat-stroked-button class="bezaras" (click)="close()">Bezárás</button>
  </form>

  <div class="login">
    <p>Van már fiókja? <a (click)="openModal()" (click)="close()">Lépjen bel</a></p>
  </div>
</div>
```

24. kép Regisztrációs felület html kódja

A regisztrációs felület célja, hogy a felhasználók új fiókot hozzanak létre az alkalmazásban. Az űrlap négy kötelező mezőt tartalmaz: vezetéknév, keresztnév, e-mail cím és jelszó. A mezők Angular Material formázással készültek, így egységes, kinézetet kapnak.

A névmezők (vezetéknév és keresztnév) egyszerű szövegmezők, amelyek csak akkor fogadhatók el, ha a felhasználó kitöltötte őket. Az e-mail mezőnél beépített ellenőrzés gondoskodik arról, hogy csak érvényes e-mail cím kerüljön beírásra.

A jelszó megadásánál minimum 8 karakter a követelmény és a mező mellett található egy ikon, amely lehetővé teszi a jelszó megjelenítését vagy elrejtését. Ez segíti a felhasználót a pontos beírásban. A rendszer minden esetben visszajelzést ad, ha egy mező hibásan lett kitöltve.

Az űrlap alján két gomb található: a „Regisztrálás” gomb a regisztrációs adatok beküldésére szolgál, míg a „Bezárás” gomb bezárja az ablakot. Az oldal alján elhelyezett link lehetőséget ad azoknak, akik már rendelkeznek fiókkal, hogy gyorsan átváltsanak a bejelentkezési felületre.

```

export class RegisterComponent {
  hidePassword = true;
  constructor(
    public dialogRef: MatDialogRef<RegisterComponent>,
    @Inject(MAT_DIALOG_DATA) public data: { message: string }, public dialog: MatDialog, private overlay: Overlay, private auth: AuthService, private snackBar: MatSnackBar
  ) {}

  close() {
    this.dialogRef.close();
  }

  login() {
    this.auth.login(this.user).subscribe({
      next: () => {
        this.message = 'Sikeres bejelentkezés!';
        this.close();
      },
      error: () => {
        this.message = 'Hibás felhasználónév vagy jelszó!';
      }
    });
  }

  openModal() {
    if (window.innerWidth > document.documentElement.clientWidth) {
      document.body.style.overflow = 'hidden';
    }
    const dialogRef = this.dialog.open(LoginComponent, {
      width: '500px',
      backdropClass: 'transparent-backdrop',
      disableClose: false,
      autoFocus: false,
      scrollStrategy: this.overlay.scrollStrategies.noop()
    });
    dialogRef.afterClosed().subscribe(() => {
      if (document.body.style.overflow === 'hidden') {
        document.body.style.overflow = 'auto';
      }
    });
  }

  @Input() user: User = {vezeteknev:'', keresztnév:'', email:'', jelszo:''};
  message: string = '';

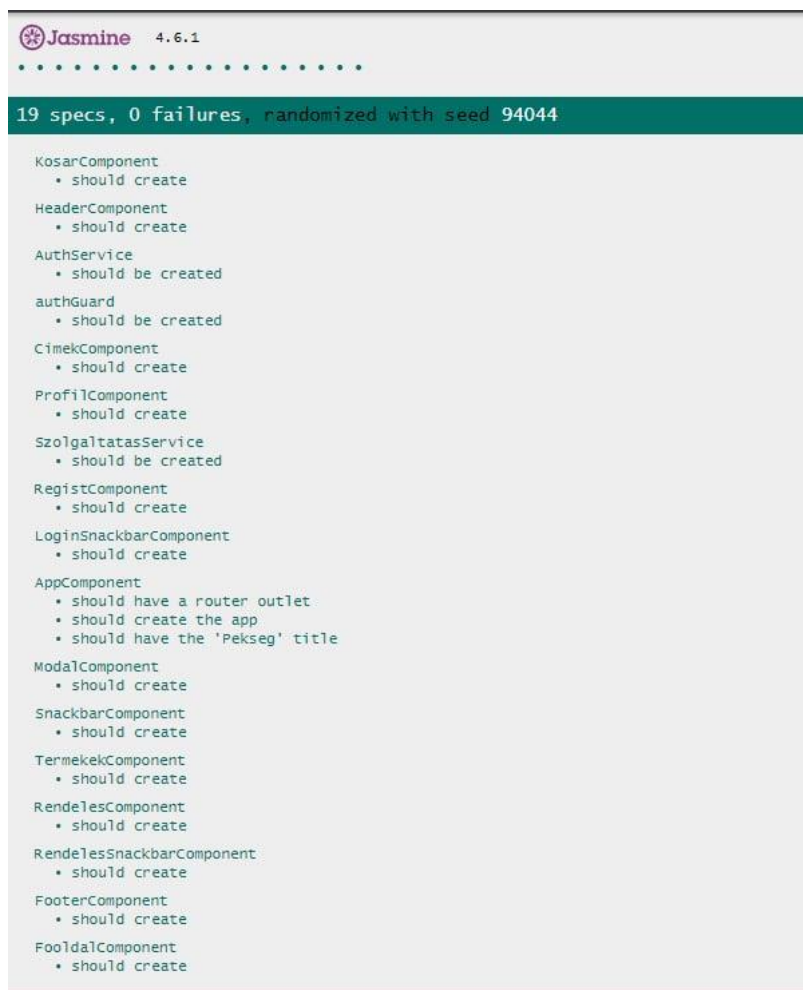
  registration() {
    if (!this.user || this.user.jelszo.length < 8) {
      this.snackBar.openFromComponent(LoginSnackBarComponent, {
        duration: 3000,
        horizontalPosition: 'center',
        verticalPosition: 'top',
        data: { message: 'Legalább 8 karakter hosszú legyen a jelszó!' },
        panelClass: ['custom-snackbar']
      });
      return;
    }
    this.auth.register(this.user)
      .subscribe({
        next: () => {
          this.message = 'Sikeres regisztráció!';
          this.login();
        },
        error: (err) => {
          if (err.status === 409 || err.error?.message === 'Ez az email már létezik') {
            this.snackBar.openFromComponent(LoginSnackBarComponent, {
              duration: 3000,
              horizontalPosition: 'center',
              verticalPosition: 'top',
              data: { message: 'Ez már regisztrált email cím!' },
              panelClass: ['custom-snackbar']
            });
          } else {
            this.snackBar.openFromComponent(LoginSnackBarComponent, {
              duration: 3000,
              horizontalPosition: 'center',
              verticalPosition: 'top',
              data: { message: 'Hibás felhasználónév vagy jelszó!' },
              panelClass: ['custom-snackbar']
            });
          }
        }
      });
  }
}

```

25. kép Regisztrációs felülethez tartozó ts

A felhasználó egy User típusú objektumba adja meg adatait (vezetéknév, keresztnév, email, jelszó), ahol a jelszónak legalább 8 karakter hosszúnak kell lennie. A registration() metódus ellenőrzi ezeket az adatokat, majd sikeres validáció után elküldi a szervernek. Ha a regisztráció sikeres, automatikusan megtörténik a bejelentkezés, hiba esetén figyelmeztető üzenet jelenik meg. A jelszó láthatósága kapcsolható, a regisztrációs felület pedig bezárható vagy átváltható bejelentkezési módra.

TESZTELÉS



```
Jasmine 4.6.1
.....

19 specs, 0 failures, randomized with seed 94044

KosarComponent
  • should create
HeaderComponent
  • should create
AuthService
  • should be created
authGuard
  • should be created
CimekComponent
  • should create
ProfilComponent
  • should create
SzolgalatasService
  • should be created
RegistComponent
  • should create
LoginSnackBarComponent
  • should create
AppComponent
  • should have a router outlet
  • should create the app
  • should have the 'Pekseg' title
ModalComponent
  • should create
SnackBarComponent
  • should create
TermekComponent
  • should create
RendelesComponent
  • should create
RendelesSnackBarComponent
  • should create
FooterComponent
  • should create
FooldalComponent
  • should create
```

26.kép Frontend tesztelése

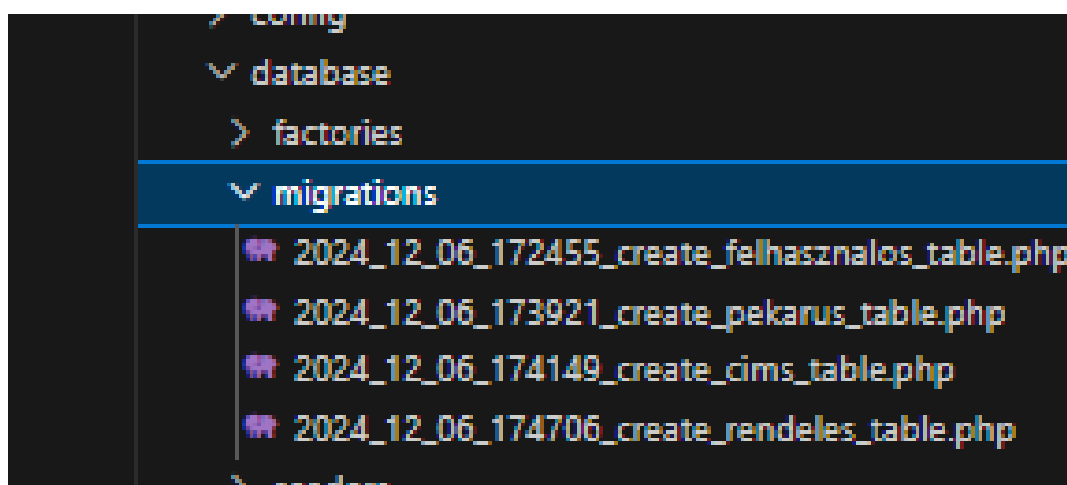
A képen egy Jasmine tesztfuttatás eredménye látható, amely 19 specifikációt (tesztesetet) tartalmaz, mindegyik sikeresen lefutott (0 hiba).

Az alkalmazás alapvető részei megfelelően működnek. (A véletlenszerű sorrend biztosítja, hogy a tesztek ne függjenek egymástól.)

BACKEND DOKUMENTÁCIÓ

Ez a dokumentáció a pékségünkhöz tartozó API működését mutatja be, amely többek között lehetőséget biztosít új termékek hozzáadására, meglévő adatok lekérdezésére, frissítésére és törlésére is. Az API REST alapú, és JSON formátumban kommunikál. Az alábbi képen a Laravel migrációs fájlokat láthatjuk. Minden fájl egy-egy táblát hoz létre az adatbázisban pl.: Pékárúk, Felhasználók stb. A táblák közötti kapcsolatok is definiálva vannak, például a rendelések táblában a pékáru, felhasználó és cím külső kulcsként szerepel. A migrációk futtatásával (php artisan migrate --seed) az adatbázis automatikusan létrejön a megadott szerkezettel.

1.)



27. kép: Laravel migrációs fájlok


```
pekseg > database > migrations > 2024_12_06_172455_create_felhasznalos_table.php
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12     public function up(): void
13     {
14         Schema::create('felhasznalok', function
15             (Blueprint $table) {
16             $table->id();
17             $table->string('nev');
18             $table->string('jelszo');
19             $table->string('email')->unique();
20         });
21     }
22
23     /**
24      * Reverse the migrations.
25      */
26     public function down(): void
27     {
28         Schema::dropIfExists('felhasznalok');
29     }
30 };
```

28. kép A felhasználók táblájának migrációs fájlja

```

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
use Illuminate\Support\Facades\DB;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('rendelesek', function (Blueprint
        $table) {
            $table->id();
            $table->foreignID('pekaru')->references('id')->on
            ('pekaruk');
            $table->foreignID('felhasznalo')->references('id')
            ->on('felhasznalok');
            $table->foreignID('cim')->references('id')->on
            ('cimek');
            $table->string('szamlazasiNev');
            $table->date('RDatum')->nullable()->default(DB::raw
            ('CURRENT_TIMESTAMP'));
            $table->date('KDatum');
            $table->boolean('fizetesiMod');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('rendelesek');
    }
};

```

29. kép A rendelések táblájának migrációs fájlja

```
2024_12_06_173921_create_pekarus_table.php ×
pekseg > database > migrations > 2024_12_06_173921_create_pekarus_table.php

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12     public function up(): void
13     {
14         Schema::create('pekaruk', function (Blueprint
15             $table) {
16             $table->id();
17             $table->string('nev');
18             $table->string('tipus');
19             $table->integer('ar');
20         });
21     }
22
23     /**
24      * Reverse the migrations.
25      */
26     public function down(): void
27     {
28         Schema::dropIfExists('pekaruk');
29     }
30 };
```

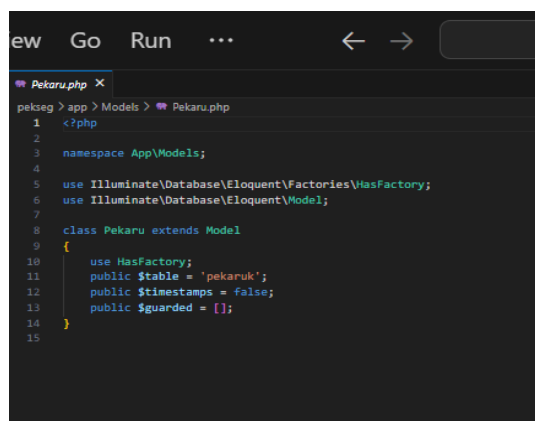
30. kép A pékáruk táblájának migrációs fájlja

```
2024_12_06_174149_create_cims_table.php X
pekseg > database > migrations > 2024_12_06_174149_create_cims_table.php
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12     public function up(): void
13     {
14         Schema::create('cimek', function (Blueprint $table)
15         {
16             $table->id();
17             $table->foreignID('felhasznalo')->references
18             ('id')->on('felhasznalok');
19             $table->string('cim')->unique();
20             $table->string('telSzam');
21         });
22     }
23
24     /**
25      * Reverse the migrations.
26      */
27     public function down(): void
28     {
29         Schema::dropIfExists('cimek');
30     }
31 };
```

31. kép A címek táblájának migrációs fájlja

2.)

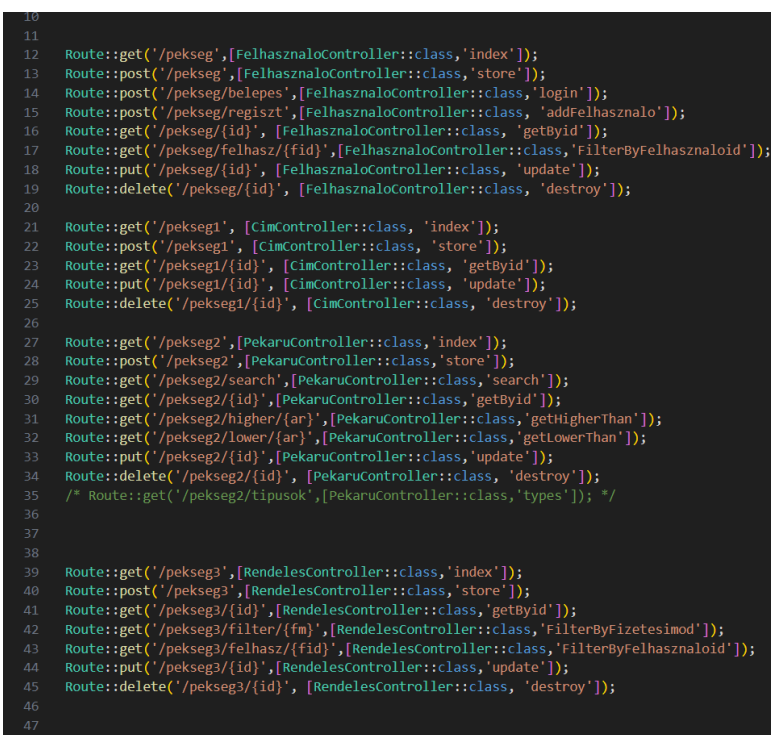
Ez a kép a Laravel modell rétegét mutatja be. A `$table` változó meghatározza a kapcsolódó táblát, a `$timestamps = false` kikapcsolja az automatikus időbélyeg mezőket, míg a `$guarded = []` engedélyezi az össze mező tömeges kitöltését. (Ez az összes táblánál el lett végezve)



```
ew Go Run ... ← →
Pekar.php X
pekseg > app > Models > Pekar.php
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Pekar extends Model
9 {
10     use HasFactory;
11     public $table = 'pekaruk';
12     public $timestamps = false;
13     public $guarded = [];
14 }
15
```

32. kép Laravel modell osztály

3.)



```
10
11
12 Route::get('/pekseg',[FelhasznaloController::class,'index']);
13 Route::post('/pekseg',[FelhasznaloController::class,'store']);
14 Route::post('/pekseg/belepes',[FelhasznaloController::class,'login']);
15 Route::post('/pekseg/regiszt',[FelhasznaloController::class,'addFelhasznalo']);
16 Route::get('/pekseg/{id}',[FelhasznaloController::class,'getByid']);
17 Route::get('/pekseg/felhasz/{fid}',[FelhasznaloController::class,'filterByFelhasznaloid']);
18 Route::put('/pekseg/{id}',[FelhasznaloController::class,'update']);
19 Route::delete('/pekseg/{id}',[FelhasznaloController::class,'destroy']);
20
21 Route::get('/pekseg1',[CimController::class,'index']);
22 Route::post('/pekseg1',[CimController::class,'store']);
23 Route::get('/pekseg1/{id}',[CimController::class,'getByid']);
24 Route::put('/pekseg1/{id}',[CimController::class,'update']);
25 Route::delete('/pekseg1/{id}',[CimController::class,'destroy']);
26
27 Route::get('/pekseg2',[PekarController::class,'index']);
28 Route::post('/pekseg2',[PekarController::class,'store']);
29 Route::get('/pekseg2/search',[PekarController::class,'search']);
30 Route::get('/pekseg2/{id}',[PekarController::class,'getByid']);
31 Route::get('/pekseg2/higher/{ar}',[PekarController::class,'getHigherThan']);
32 Route::get('/pekseg2/lower/{ar}',[PekarController::class,'getLowerThan']);
33 Route::put('/pekseg2/{id}',[PekarController::class,'update']);
34 Route::delete('/pekseg2/{id}',[PekarController::class,'destroy']);
35 /* Route::get('/pekseg2/tipusok',[PekarController::class,'types']); */
36
37
38
39 Route::get('/pekseg3',[RendelesController::class,'index']);
40 Route::post('/pekseg3',[RendelesController::class,'store']);
41 Route::get('/pekseg3/{id}',[RendelesController::class,'getByid']);
42 Route::get('/pekseg3/filter/{fm}',[RendelesController::class,'filterByFizetesimod']);
43 Route::get('/pekseg3/felhasz/{fid}',[RendelesController::class,'filterByFelhasznaloid']);
44 Route::put('/pekseg3/{id}',[RendelesController::class,'update']);
45 Route::delete('/pekseg3/{id}',[RendelesController::class,'destroy']);
46
47
```

33. kép API végpontok

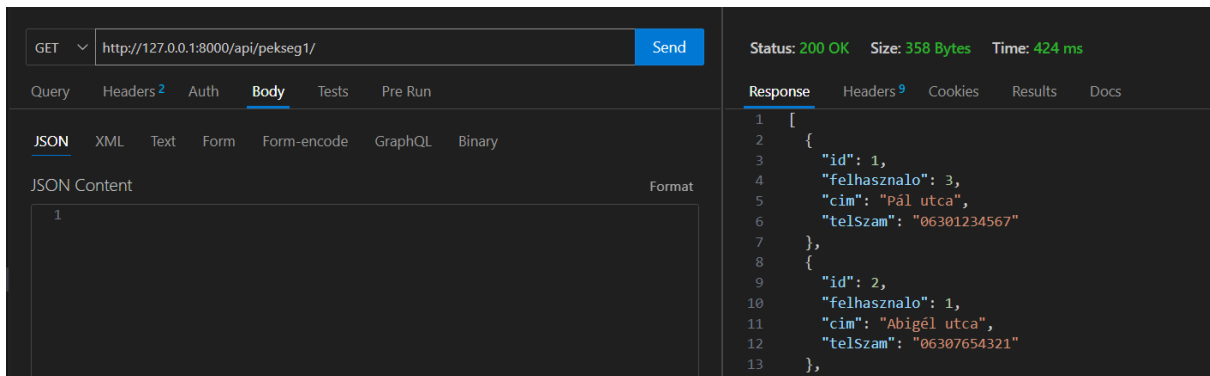
Itt a Laravel útvonalakat/útvonaldefiníciókat láthatjuk. Ezek az API végpontjait határozzák meg a különböző vezérlők számára. A **Route::get**, **Route::post**, **Route::put** és **Route::delete** metódusok különböző műveleteket végeznek, például adatokat kérnek le, hoznak létre, módosítanak vagy törölnek.

Mindegyikben egyaránt el van helyezve egy végpont a `getByid`, az `Update`, és `Delete` metódusoknak.

Ezekon kívül található `SearchBy`, `Filterby`, és `Higher/LowerThan` metódus is. Viszont ezek nem mindegyik Controllerben lettek implementálva.

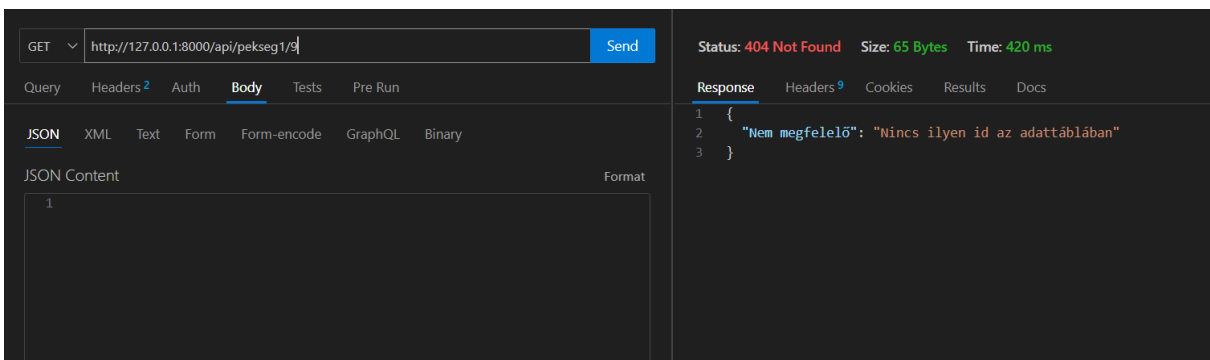
TESZTELÉS

getByid:



34. kép getByid metódus sikeres tesztelése

Reprezentáció a 4 getByid metódus egyikéről. A GET kérés a `http://127.0.0.1:8000/api/pekseg1/` URL-re történik, és egy JSON-formátumú választ ad vissza az adott id-jú felhasználó, pékárú, vagy cím részleteivel/adataival.



35. kép getByid metódus tesztje hibás adat/érték beírásakor

Itt pedig az a hibaüzenet látható, státuszkóddal együtt, amit akkor kapunk vissza, ha például nem létező vagy rossz id-t adunk meg keresésre.

Keresés név és típus alapján:

A search függvény egy request (kérés) objektumot kap bemenetként, amely alapján keresést végez az adatbázisban. Ha a kérés tartalmaz egy 'nev' (név) paramétert, akkor az adott név alapján szűri az eredményeket, míg, ha a 'tipus' (típus) paraméter szerepel benne, akkor a termék típusa szerint keres. Ha egyik paraméter sem található a kérésben, a függvény egy 400-as hibakódú válaszüzenetet küld vissza.

```
public function search(Request $request)
{
    if ($request->has('nev'))
    {
        $result = Pekar::where('nev', $request->nev)->get();
    }
    elseif ($request->has('tipus'))
    {
        $result = Pekar::where('tipus', $request->tipus)->get();
    }
    else
    {
        return response()->json(['error' => 'Hiányzó keresési paraméter'], 400);
    }

    return response()->json($result);
}
```

36. kép Search metódus

Query	Headers	Auth	Body	Tests	Pre Run
Query Parameters					
<input checked="" type="checkbox"/>	nev		Kifli		
<input type="checkbox"/>	parameter		value		

Response	Headers	Cookies
1 [
2 {		
3 "id": 2,		
4 "nev": "Kifli",		
5 "tipus": "Nostalgia",		
6 "ar": 80		
7 }		
8]		

37. kép Search metódus sikeres tesztelése

Update/Destroy:

Ez a kódrészlet egy API-t mutat be, amely a Pekarú tábla adatainak a frissítését (update) és törlését (destroy) végzi.

(A többi kontrollerben is megtalálható az a két metódus)

Update:

```
public function update(Request $request,$id)
{
    $pekaru=Pekaru::find($id);
    if(is_null($pekaru))
    {
        return response()->json(['Nem található'=>'Nincs ilyen id-jű sor az adattáblában'],404);
    }
    $validator=Validator::make($request->all(),
    [
        'nev'=>'required',
        'tipus'=>'required',
        'ar'=>'required'
    ]
    );
    if($validator->fails())
    {
        return response()->json(['Hiba!'=> 'Fontos adat hiányzik, nem lehet frissíteni'],406);
    }

    $pekaru->update($request->all());
    return response()->json(['Pékárú'=>$pekaru->nev],201);
}
```

38. kép Az update metódus

Ez a függvény egy megadott id alapján keres egy Pekarú rekordot, majd a kapott adatokkal frissíti azt.

Validálja a kérést, és ha hiányzik a nev, típus vagy ar mező, akkor 406 - Nem elfogadható hibát ad vissza.

Ha minden adat megfelelő, frissíti a rekordot.

Destroy:

Ez a függvény pedig egy adott id-jű Pekarú rekordot töröl az adatbázisból. Ha sikeres a törlés, akkor egy 205 – Reset Content státuszkódot küld vissza.

```
public function destroy($id)
{
    $pekaru=Pekaru::find($id);
    if(is_null($pekaru))
    {
        return response()->json(['valami nem jó'=>'Nincs ilyen id-jű sor az adattáblában'],404);
    }
    $pekaru->delete();

    return response('',205);
}
```

39. kép A destroy metódus

GetHigher/Lowerthan:

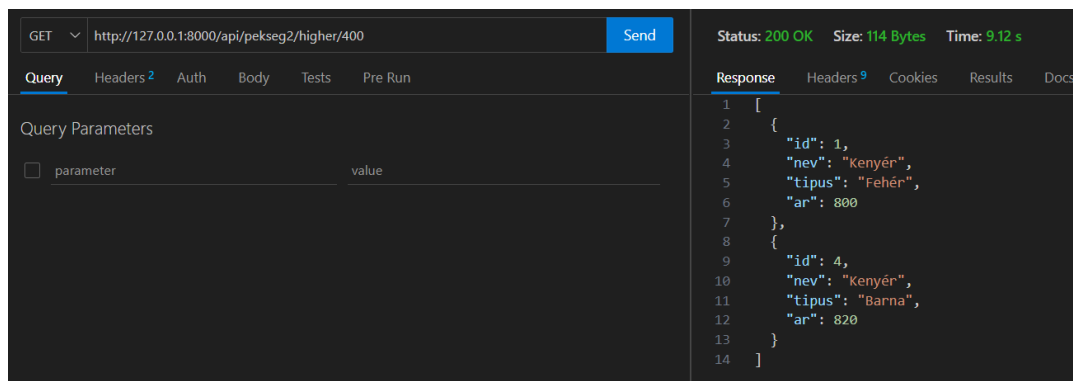
```
public function getHigherThan($ar)
{
    $pekaru=Pekaru::where('ar','>',$ar);
    if($pekaru->exists())
    {
        return $pekaru->get();
    }
    else
    {
        return response()->json(['Nem létező érték'=>'Nem lehet az keresett összegnél magasabb árú pékárut találni'],404);
    }
}

public function getLowerThan($ar)
{
    $pekaru=Pekaru::where('ar','<',$ar);
    if($pekaru->exists())
    {
        return $pekaru->get();
    }
    else
    {
        return response()->json(['Nem létező érték'=>'Nem lehet az keresett összegnél alacsonyabb árú pékárut találni'],404);
    }
}
```

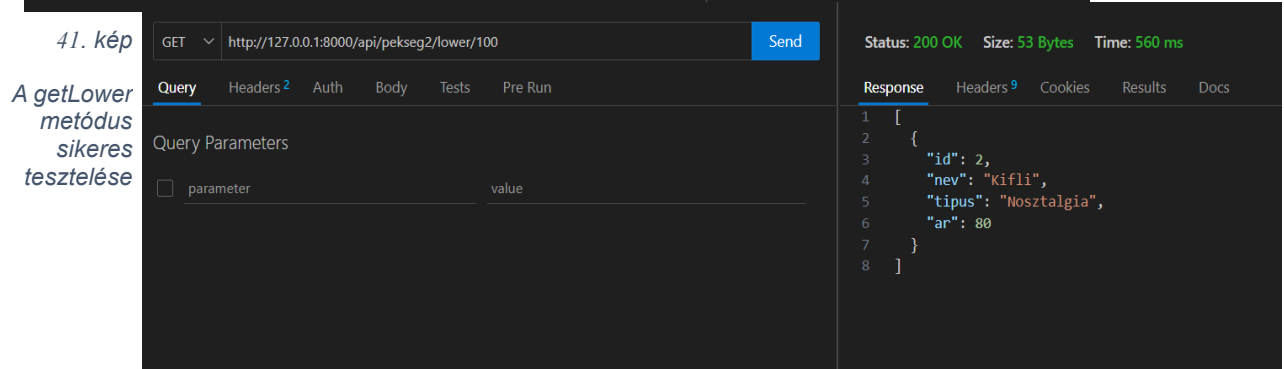
40. kép A GetLower és GetHigher metódus

Ezek a függvények visszaadják azokat a pékárukat, amelyek ára magasabb és vagy alacsonyabb, mint a megadott érték (\$ar).

Lehetővé teszi a pékáruk ár szerinti keresését egy weboldalon.



42. kép
A getHigher
metódus sikeres
tesztelése



41. kép
A getLower
metódus sikeres
tesztelése

FilterByFizetesimod:

```
public function FilterByFizetesimod($fm)
{
    $rendeles=Rendelese::where('fizetesimod','=',$fm);
    if($rendeles->exists())
    {
        return $rendeles->get();
    }
    else{
        return response()->json(['Nem létezik'=>'Nem létezik ilyen féle fizetési mód'],407);
    }
}
```

43. kép A FilterByFizetesimod metódusa

A FilterByFizetesimod lehetővé teszi a Rendelések szűrését fizetési mód alapján, és visszaadja a találatokat, ha léteznek. Két féle fizetési mód jöhet szóba: Kártyás és készpénzesHa a keresésnél 0-t adunk meg akkor a készpénzzel kívánt fizetési kérelmeket kapjuk meg. Az 1-es megjelölés pedig a kártyával való fizetést jelenti.

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://127.0.0.1:8000/api/pekseg3/filter/0
- Status:** 200 OK
- Size:** 265 Bytes
- Time:** 535 ms
- Response Body:** A JSON array containing two order objects.

```
[
  {
    "id": 1,
    "pekaru": 7,
    "felhasznalo": 2,
    "cim": 5,
    "szamlazasiNev": "Vicc Elek",
    "RDatum": "2024-11-25",
    "KDatum": "2024-11-26",
    "fizetesimod": 0
  },
  {
    "id": 3,
    "pekaru": 6,
    "felhasznalo": 5,
    "cim": 4,
    "szamlazasiNev": "Para Zita",
    "RDatum": "2024-11-27",
    "KDatum": "2024-11-28",
    "fizetesimod": 0
  }
]
```

44. kép A FilterByFizetesimod sikeres tesztelése

Seeder:

A DatabaseSeeder osztály feladata az adatbázis feltöltése kezdeti adatokkal a Laravel keretrendszerben. A seeder biztosítja, hogy a rendszer mindig rendelkezzen alapadatokkal, amelyeket a fejlesztés és tesztelés során felhasználhatunk.

Az alábbi modelleket használja:

- **Felhasznalo:** Felhasználói adatok tárolása
- **Pekaru:** Pékárúk tárolása
- **Cim:** Felhasználókhoz tartozó címek kezelése
- **Rendeles:** Megrendelések kezelése

```
public function run(): void
{
    $felhasznalok = ['Kovács Sarolta', 'Vicc Elek', 'Pál Pál', 'Molnár Ödön', 'Para Zita'];
    $jelszo = ['S1234', 'V1234', 'P1234', 'Ö1234', 'Z1234'];
    $email = ['kovacs sarolta@gmail.com', 'viccelek@gmail.com', 'palpal@gmail.com', 'molnarodon@gmail.com', 'parazita@gmail.com'];
    $fRows = count($felhasznalok);
    for ($i=0; $i < $fRows; $i++) {
        Felhasznalo::create(['nev'=> $felhasznalok[$i], 'jelszo'=>$jelszo[$i], 'email'=>$email[$i]]);
    }

    $pnev = ['Kenyér', 'Kifli', 'Zsömlé', 'Kenyér', 'Brios', 'Fánk', 'Pizzás csiga'];
    $tipus = ['Fehér', 'Nosztalgia', 'Rozsos', 'Barna', 'Fehér', 'Barackos', 'Csiga'];
    $ar = [800, 80, 120, 820, 200, 250, 275];
    $pRows = count($pnev);
    for ($i=0; $i < $pRows; $i++) {
        Pekaru::create(['nev'=> $pnev[$i], 'tipus'=>$tipus[$i], 'ar'=>$ar[$i]]);
    }

    $felid = [3, 1, 4, 5, 2];
    $cim = ['Pál utca', 'Abigél utca', 'Fecske utca', 'Petőfi utca', 'Luther utca'];
    $telszam = ['06301234567', '06307654321', '06307564231', '06303125467', '06309273638'];
    $cRows = count($felid);
    for ($i=0; $i < $cRows; $i++) {
        Cim::create(['felhasznalo'=> $felid[$i], 'cim'=>$cim[$i], 'telszam'=>$telszam[$i]]);
    }

    $pekid = [7, 5, 6, 2, 3];
    $felid = [2, 3, 5, 1, 4];
    $cimid = [5, 1, 4, 2, 3];
    $szamlazasinev = ['Vicc Elek', 'Pál Pál', 'Para Zita', 'Kovács Sarolta', 'Molnár Ödön'];
    $rendelesD = ['2024.11.25', '2024.11.26', '2024.11.27', '2024.11.28', '2024.11.29'];
    $rendelesK = ['2024.11.26', '2024.11.27', '2024.11.28', '2024.11.29', '2024.11.30'];
    $fizetesimod = [false, true, false, true, true];
    $rRows = count($pekid);
    for ($i=0; $i < $rRows; $i++) {
        Rendeles::create(['pekaru'=> $pekid[$i], 'felhasznalo'=>$felid[$i], 'cim'=>$cimid[$i], 'szamlazasiNev'=>$szamlazasinev[$i],
            'RDatum'=>$rendelesD[$i], 'KDatum'=>$rendelesK[$i], 'fizetesimod'=>$fizetesimod[$i] ]);
    }
}
```

45. kép DatabaseSeeder kódja

A run() metódus feltölti az adatbázist tesztadatokkal. Először létrehozza a felhasználókat névvel, jelszóval és email-címmel, majd hozzáadja a pékárukat típussal és árral. Ezután a felhasználók címei és telefonszámai kerülnek rögzítésre. Végül a rendelések kerülnek mentésre a kapcsolódó adatokkal, beleértve a pékáru azonosítót, számlázási adatokat, dátumokat és fizetési módokat. Az adatokat a megfelelő modellek create() metódusa menti el.

Bejelentkezési (login):

A login metódus ellenőrzi a felhasználó hitelesítő adatait. Megkeresi a felhasználót az email cím alapján, majd összehasonlítja a megadott jelszót a tárolt jelszóval. Ha minden egyezik, visszaadja a felhasználó adatait.

addFelhasznalo:

Az addFelhasznalo metódus új felhasználót regisztrál. Ellenőrzi, hogy minden kötelező mező (név, email, jelszó) ki legyen töltve. Ha valami hiányzik, hibát jelez. Ha minden rendben van, elmenti az adatokat és visszaadja az új felhasználót.

```
public function login(Request $request){
    $user = Felhasznalo::where('email','=', $request->email)->first();
    if ($user->exists()) {
        if ($request->email == $user->email && $request->jelszo == $user->jelszo) {
            return $user;
        }
    }
}

public function addFelhasznalo(request $request){
    $validator = Validator::make($request->all(),[
        'vezeteknev' => 'required',
        'keresztnev' => 'required',
        'email' => 'required',
        'jelszo' => 'required'
    ]);
    if($validator->fails()){
        return response()->json($validator->errors(),400);
    }
    $felhasznalok = Felhasznalo::create($request->all());
    return response($felhasznalok,201);
}
```

46. kép A login és addFelhasznalo metódusok

TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

A jelenlegi verzió megfelelően használható, de a fejlesztés során több olyan funkció ötlete is felmerült, amit a jövőben érdemes lenne beépíteni:

- A főoldalon lévő legfelkapottabb termékek megjelenítése megvásárolt mennyiség alapján
- Intézmények értékelhetik a kiszállítást és a termékeket
- Csak elérhető termékek szerepeljenek a kínálatban
- Mobilbarát felület vagy külön app fejlesztése

IRODALOMJEGYZÉK

1. <https://angular.io/docs>
2. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods>
3. https://inf.u-szeged.hu/~gnemeth/adatbgyak/exe/EK_diagram/az_egyedkapcsolat_diagram_elemei.html
4. <https://laravel.com/docs>
5. <https://martinjoo.dev/layered-architectures-with-laravel>
6. <https://www.fornetti.hu>