



**Széchenyi István Katolikus Technikum és Gimnázium**

**Szoftverfejlesztő és -tesztelő projektfeladat**

**Harvest Heaven**

Készítették:

**Szurdok Bence Attila és Havrán Tibor**

**Ózd, 2025**

# TARTALOMJEGYZÉK

<b>Bevezetés .....</b>	<b>4</b>
<b>Felhasználói dokumentáció .....</b>	<b>5</b>
<i>Rendszerkövetelmény</i> .....	5
A hardverszükségletek: .....	5
A szoftverszükségletek:.....	5
<i>A webalkalmazás indítása</i> .....	6
<i>A webalkalmazás használata</i> .....	7
Általános információk a webalkalmazásról .....	7
Az oldalak leírása és képernyőképek .....	7
<b>Fejlesztői dokumentáció .....</b>	<b>13</b>
<i>Alkalmazott fejlesztői és csoportmunka eszközök</i> .....	13
A csoportmunka eszközei.....	13
Az Adatbázis eszközei .....	13
A frontend eszközei.....	13
A backend eszközei.....	14
<b>Adatbázis .....</b>	<b>16</b>
<i>A tábázati modell leírása</i> .....	16
A funkciók, táblázatra bontva .....	17
<i>Táblák, Adatszerkezetek bemutatása</i> .....	20
<i>E-K diagramm</i> .....	21
<b>Frontend.....</b>	<b>22</b>
<b>Backend.....</b>	<b>26</b>
<i>A Backend funkciói és azoknak bemutatása</i> .....	26
A „felhasznalok” tábla metódusai: .....	26
A „termek” tábla metódusai: .....	29
A „hirdetesek” tábla metódusai.....	31

A „rendelesek” tábla metódusai .....	34
A metódusok útvonalai.....	36
<b>Tesztelés .....</b>	<b>37</b>
<i>A frontend tesztje .....</i>	37
<i>A backend tesztje .....</i>	38
<b>Továbbfejlesztési lehetőségek .....</b>	<b>42</b>
<b>Irodalomjegyzék.....</b>	<b>43</b>

## BEVEZETÉS

A Harvest Heaven egy érdekes ötlet volt. Azon tűnődtünk, hogy mi legyen a projektünk témája. Hosszas gondolkozás után arra jutottunk, hogy alkotunk egy webportált, ahol a zöldárúval és ennek termelésével vagy Importálásával foglalkozó gazdák és vállalkozók egyszerűbben vásárlókra találhatnak, a vásárló kis- és nagykereskedők pedig egyszerűbben megtalálják az általuk keresett és forgalmazni kívánt terméket. Ehhez az ötlethez inspirációt adott, hogy mindenki szülei a zöldárú forgalmazási kategóriában tevékenykednek. Ennek apropójában a saját problémánkra alkottunk egy megoldást, amit nem szerettünk volna magunkban tartani, ezáltal segítve a velünk egy kategóriában helyet foglaló cégek áruinak beszerzését, eladását, ezzel felgyorsítva a kereskedelmet. Az oldalunk kizárolag adószámmal rendelkező cégek vásárolhatnak, akik ebben az iparban foglalnak tevékenységi helyet. A projektünket a GitHub-on lehet megtalálni és ingyenesen letölteni és használni.

<https://github.com/bnce4554/HarvestHeaven.git>

# FELHASZNÁLÓI DOKUMENTÁCIÓ

## RENSZERKÖVETELMÉNY

A rendszerünk használatához a legalapabb számítógép is elég ezzel széles felhasználókort elérve. A webalkalmazásunk a Windows operációs rendszerre van optimalizálva, pontosabban a Windows 10-re.

### A HARDVERSZÜKSÉGLETEK:

#### Rendszerkövetelmény

	Ajánlott:	Minimális
Processzor (CPU)	Intel(R)_Core(TM)_i3-9100F_CPU_@_3.60GHz	Intel Core i3-6100T Dual-Core 3.2GHz
Operációs rendszer (OP)	Windows 10 pro	Windows 10
RAM:	8 GB RAM	6 GB RAM
Tárhely:	6 GB	6 GB
Internet:	Széles sávú	Széles sávú
GPU:	NVIDIA GeForce GTX 1050	Intel(R) UHD Graphics 620

ábra 1: Hardverkövetelmények

### A SZOFTVERSZÜKSÉGLETEK:

A webalkalmazás frontend részéhez szükséges a Visual Studio Code nevű alkalmazás, egy webböngésző pl.: Google Chrome vagy Safari.

A backend részleghez szükséges szintén a Visual Studio Code és

## A WEBALKALMAZÁS INDÍTÁSA

A webalkalmazásunk három fő részből áll:

- Frontend (Ez a felhasználói felület amely látható a weboldalon)
- Backend (Ez az amelyet a felhasználó nem lát, segíti a frontend működését és kommunikál az Adatbázissal)
- Adatbázis (Ide kerülnek a webalkalmazás által feldolgozott adatok melyeket szükséges eltárolni)

Szükséges dolgok a webalkalmazásunk használatához:

- Visual Studio Code – A webalkalmazás minden részéhez szükséges
- Node.js – A frontend működéséhez
- Laravel és Composer – A backend működéséhez
- XAMPP – Az adatbázis működéséhez

Indítási lépések:

Miután letöltöttük a fent felsorolt alkalmazásokat ezután kezdődhet az indítási folyamat:

- A Visual Studio Codeban megnyitjuk a projektmappát
- Elindítjuk a XAMPP-ot és miután betöltött a control panel elindítjuk az Apache-t és a MySQL-t, majd megnyitunk egy bármilyen böngészőt és beírjuk hogy „<http://localhost/phpmyadmin>”.
- Visszalépve a VSC-n belül a projektmappánkba megkeressük a Backend projektmappát majd azon belül nyitunk egy terminált. A terminálba beírjuk hogy „php artisan migrate”. Ezzel a parancssal a backend felmigráltja az adatbázisunkat a phpMyAdminba.
- Ezután visszalépve a böngészőbe rányomunk az új adatbázisunkra melynek „HarvestHeaven” lesz a neve majd a database.sql nevű fájlból (Ami a projektmappában található) kimásoljuk az adatokat és a böngészőben az adatbázison belül az SQL fülben beillesztjük és feltöljtjük, így az adatbázis tele lesz adatokkal. Ezt követően belépünk a backend termináljába és beírjuk hogy „php artisan serve” ezzel futtatva a backendet.
- Ezután a VSC-be visszalépve belépünk a frontend mappájába és terminálba beírjuk hogy „npm install” így hozzáadja azt a frontendünkhez. Majd ezután már csak annyi a

dolgunk hogy beírjuk a frontend termináljába hogy „ng serve” ezzel futtatva a frontendet. Ezután rámegyünk az ott megjelent linkre, és előttünk a webalkalmazás.

## A WEBALKALMAZÁS HASZNÁLATA

A webalkalmazásunk használatát elég egyszerűre terveztük hogy mindenki hiba nélkül és könnyen tudja használni anélkül hogy akármi is nehézséget okozna.

### ÁLTALÁNOS INFORMÁCIÓK A WEBALKALMAZÁSRÓL

A Harvest Heaven célja, hogy megkönnyítse a zöldség-gyümölcs vásárlását a kereskedők számára.

A platformon egyetlen felhasználói szerepkör van, amelyen belül a felhasználó eladóként, vásárlóként vagy mindkettőként is működhet.

Az alkalmazás egy reszponzív, felhasználóbarát felületet biztosít, amelyen keresztül a felhasználók könnyedén böngészhetnek termékek között, kosárba tehetik őket, és eladóként akár saját termékeket is feltölthetnek.

- Frontend: Angular- Backend: Laravel (PHP)
- Adatbázis: MySQL
- Authentikáció: JWT token alapú

### AZ OLDALAK LEÍRÁSA ÉS KÉPERNYŐKÉPEK

Itt látható webalkalmazásunk felhasználói felülete leírással, képernyőképekkel.

#### HOME(KEZDŐLAP)

Tájékoztat a platform céljáról, navigáció a fő funkciókhoz.



ábra 2: Fóoldal

### LOGIN(BEJELENTKEZÉS)

Reactive Forms alapú űrlap, amely hitelesítési kérést küld a backendhez.

HARVEST HEAVEN

Kezdőlap Vásárlás Rölkünk Kosár Bejelentkezés Regisztráció

Bejelentkezés

Név \_\_\_\_\_

Jelszó \_\_\_\_\_

BEJELENTKEZÉS

Még nem regisztrált?

Regisztráció egy új fiókot erre kattintva!

TELEFON  
+36 70 218 2111  
+36 70 643 4611

HARVEST HEAVEN

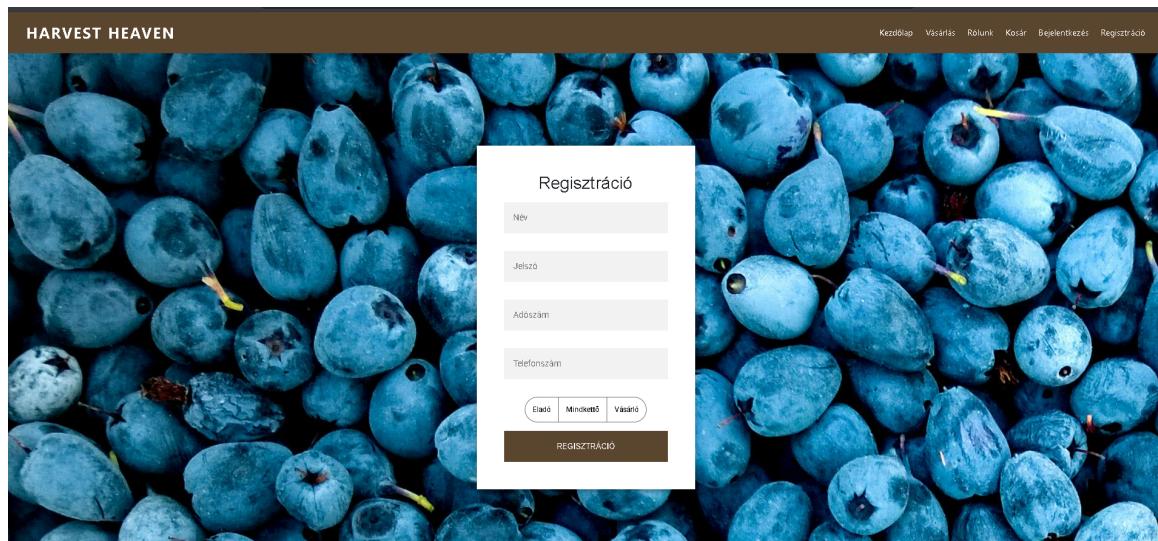
Bejelentkezés  
Regisztráció

HÍRLEVEL  
Iratkozz fel hírlevelünkre, ha nem szeretnél lemaradni a bővülő termékkínálatokról!

ábra 3: Bejelentkezés

### REGISTER(REGISZTRÁCIÓ)

Felhasználói regisztráció, a regisztrált felhasználói adatok az adatbázisba kerülnek.



ábra 4: Regisztráció

## SHOP(VÁSÁRLÁS)

Termékek böngészése kategória szerint, kosárba helyezés lehetőséggel.

TELEFON

+36 70 218 2111  
+36 70 643 5511

EMAIL

szurdok.bence@csikos-ozd.hu  
havran.tibor@csikos-ozd.hu

HARVEST HEAVEN

Bejelentkezés  
Regisztráció  
Rólunk  
Termékek  
Termék-feliratokról

HÍRLEVÉL

Iratkozz fel hírlevelünkre, ha nem szeretnél lemaradni a bővülő termékinhalatokról!

Add meg az email címed

ábra 5: Shop

## CART(KOSÁR)

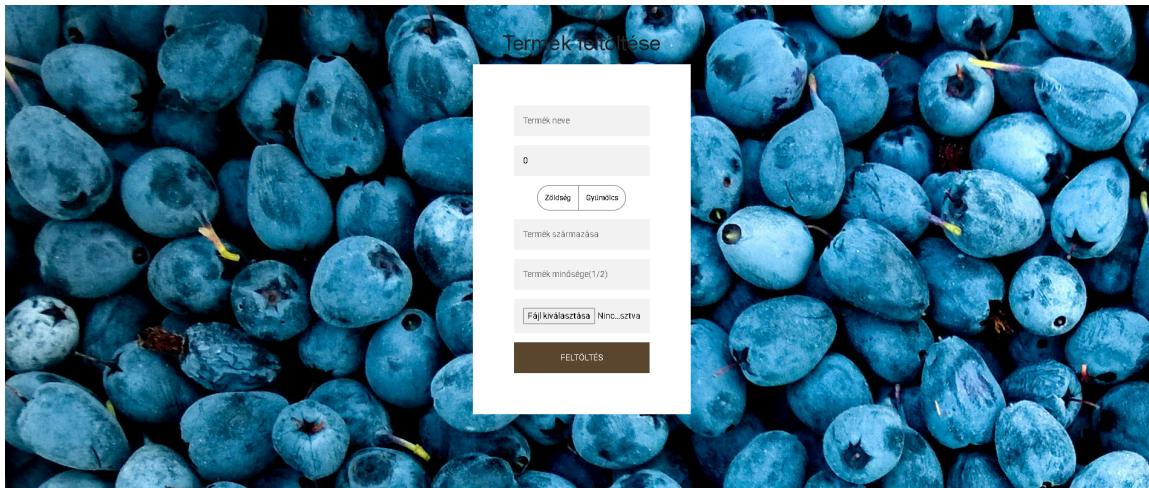
Kosár tartalmának kezelése és szerkesztése, valamit fizetés.

The screenshot shows a user interface for managing a shopping cart. At the top left is a link 'Vissza vásárláshoz'. Below it is a section labeled 'Kosár' (Cart). On the right, there is a large, dark-colored modal window titled 'Fizetési adatok' (Payment details). This modal contains fields for 'Kártyabirtokos neve' (Name of cardholder), 'Kártyabirtokos neve' (Name of cardholder), '1234 5678 9012 3457' (Card number), 'Kártyaszám' (Card number), 'MM/YYYY' (Expiry date), 'Lejárati dátum' (Expiry date), '123' (CVV number), and 'CVV szám' (CVV number). There is also a link 'Tovább a fizetéshez' (Continue to payment) and a note 'Összesen 0 - Ft' (Total 0 - Ft).

ábra 6: Kosár

## FELTOLTES(FELTÖLTÉS)

Felhasználók számára termékek feltöltése az adatbázissal megfelelő input mezőkkel.



ábra 7: Termékfeltöltés

## FELHASZNALO(FELHASZNÁLÓI OLDAL)

Felhasználói adatok megtekintése.

## ABOUT(RÓLUNK)

### Információ a projekt készítőiről.

#### Rólunk

A Harvest Heaven egy érdekes ötlet volt. Azon tűnöttünk, hogy mi legyen a projektünk témaja. Hosszas gondolkozás után arra jutottunk, hogy alkotunk egy webportált, ahol a zöldárúval és ennek termelésével vagy importálásával foglalkozó gazdák és vállalkozók egyszerűbben vásárolhatnak teáshatnak, a vásárlók kis- és nagykereskedők pedig egyszerűbben megtalálják az általuk keresett és forgalmazni kívánt termékeket.

Ehhez az ötlethez inspirációt adott, hogy mindenkiőrök szülei a zöldárú forgalmazási kategóriában tevékenykednek. Ennek apropójában a saját problémánkra alkottunk egy megoldást, amit nem szereztünk volna magunkban tartani, ezáltal segítve a velünk egy kategóriában helyet foglalo cégek áruinak beszerzését, eladását, ezzel felgyorsítva a kereskedelmet. Az oldalunk kizárolag adószámmal rendelkező cégek vásárolhatnak, akik ebben az iparban foglalnak tevékenységi helyet.



#### Csapatunk



Havrán Tibor

Frontend fejlesztő



Szurdok Bence

Backend/adatbázis fejlesztő

**ábra 8: Rólunk (1.kép)**

#### Küldetésünk

Küldetésünk, hogy folyamatosan javitsuk a friss zöldségek és gyümölcsök elérhetőségét, miközben figyelünk a fenntarthatóságra és a környezetünkre. minden egyes termékünket úgy választjuk ki, hogy az megfeleljen a legszigorúbb minőségi előírásoknak, és támogassuk az egészséges élelművet.

#### Kapcsolatfelvétel

Ha bármiremű kérdése van, ne hagyozzon kapcsolatba lépni velünk!

##### Telefon

+36 70 218 2111

+36 70 643 5511

##### Email

szurdok.bence@szikszoi-ozd.hu

havran.tibor@szikszoi-ozd.hu

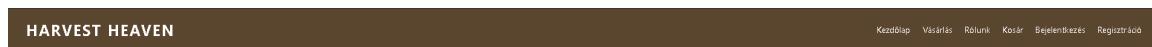
**ábra 9: Rólunk (2. kép)**

## HEADER/ FOOTER

Navigációs és információs komponensek.



ábra 10: Footer



ábra 11: Header

# FEJLESZTŐI DOKUMENTÁCIÓ

## ALKALMAZOTT FEJLESZTŐI ÉS CSOPORTMUNKA ESZKÖZÖK

### A CSOPORTMUNKA ESZKÖZEI

A csoportmunka eszközeként a Facebook Messenger, és a Discord alkalmazásokat használtuk. Csapatunknak ezzel volt a legegyszerűbb kommunikálni. A terveket főként Messengeren osztottuk meg egymással, majd mikor össze kellett dolgoznunk, a Discord nyújtott nagy segítséget az élő képernyőmegosztás funkciójával.

### AZ ADATBÁZIS ESZKÖZEI

Az adatbázist Visual Studio Code és phpMyAdmin használatával Szurdok Bence készítette. Az adatbázis 4 fő táblát tartalmaz és ezen belül mezőket. minden táblának vannak saját mezői.

### A FRONTEND ESZKÖZEI

A frontendet Havrán Tibor készítette.

A frontend Angular keretrendszerrel készült.

Fontosabb Angular elemek:

- AppRoutes: útvonalak beállítása

- Komponensek: homeComponent, loginComponent, registerComponent, shopComponent, cartComponent, feltoltesComponent, felhasznaloComponent, aboutComponent, headerComponent, footerComponent
- Szolgáltatások: AuthService, ProductService, CartService
- AuthGuard: védett oldalak biztonsága

Kommunikáció:

- A frontend HttpClient segítségével kommunikál a Laravel backenddel.
- Az adatok biztonságát JWT tokenes autentikáció biztosítja.

Validáció:

- Reactive Forms és Template-driven Forms használata mindenhol a felhasználói bemeneteknél.

## A BACKEND ESZKÖZEI

A Backend a Visual Studio Code-al Szurdok Bence Által készült. Az API fontosabb elemei melyek a működést eredményezik:

-Routes/api.php: Az útvonalak beállítása

-Controllerek: FelhasznalokController, TermekController, HirdetesekController, RendelesekController

-Migrations

A Backend működéséhez szükséges a Laravel és a Composer. Az API-ban létrehozott metódusok és elemek mind egy phpMyAdmin adatbázisból dolgozik. A Backed az adatbázis elemeit bővíti, frissíti vagy éppen törli.

# ADATBÁZIS

## A TÁBÁZATI MODELL LEÍRÁSA

Az adatbázis fájl 4 táblát tartalmaz. Ezek a felhasználó, termék, hirdetés és rendelés táblák.

A felhasználó tábla tárolja el az adott cég/személy adatait ideértve a nevet adószámot emailt, jelszót és satöbbi. Itt az elsődleges kulcs a felhasználó Id-ja lesz, amit minden személy kap.

A termékek tábla a legegyszerűbb, ez a tábla tárolja az adott termék árát, nevét, származási helyét és minden egyéb tulajdonságát. Persze itt is minden termék kap egy egyedi Id-t mint minden tábla minden eleme.

A hirdetés tábá már egy fokkal bonyolultabb mivel ebben a táblában egy adott felhasználó hirdet egy adott terméket egyedi árral és mennyiséggel. Ez a tábla elsődleges kulcsként ugyan úgy az Id-t kapja mint az összes többi tábla viszont ezen felül idegen kulcsként kapja a hirdető felhasználó és a termék Id-ját. Ezen felül eltárolja még az adott hirdetés létrehozásának dátumát a termékből eladni kívánt mennyiséget ennek mértékegységét és árát. A rendelés tábla egy adott hirdetés kap amit szállítással kértek, ebben a táblában tároljuk a rendelés idejét, az idegen kulcsként kapott eladó, vásárló, hirdetés és a termék Id-ját.

## A FUNKCIÓK, TÁBLÁZATRA BONTVA

A „felhasznalok” tábla mezői:

<u>Név</u>	<u>Típus</u>	<u>Feladat</u>
<b>id(PK)</b>	Integer	A felhasználók sorszámát tárolja el elsőleges kulcsként
<b>nev</b>	String	A felhasználó nevét tárolja el.
<b>adoszam</b>	String	Az adott felhasználó adószámát tárolja
<b>telefonszam</b>	String	A felhasználó telefonszámát tárolja
<b>jelszo</b>	String	A felhasználó jelszavát tárolja
<b>felhasznaloTipus</b>	String	A felhasználó típusát tárolja el

A „termek” tábla mezői:

<u>Név</u>	<u>Típus</u>	<u>Feladat</u>
<b>id(PK)</b>	Integer	Az adott termék sorszámát tárolja el elsőleges kulcsként
<b>termek_nev</b>	String	A termék nevét tárolja el
<b>ar</b>	Integer	Az adott termék árát tárolja el
<b>kategoria</b>	String	A termék kategóriáját tárolja

<b>szarmazasi_orszag</b>	String	A származási országot határozza meg
<b>minoseg</b>	Integer	A termék minőségét adja meg számban
<b>kep</b>	String	A termék képének az URL-jét vagy fájlnevét tárolja el

A „hirdetesek” tábla mezői:

<b>Név</b>	<b>Típus</b>	<b>Feladat</b>
<b>id(PK)</b>	Integer	A hirdetés sorszámát tárolja elsődleges kulcsként
<b>felhasznaloId(FK)</b>	Integer	A hirdetést feltöltő felhasználó sorszámát tárolja idegen kulcsként
<b>termekId(FK)</b>	Integer	A hirdetésben feltöltött termék Id-ját tárolja el idegen kulcsként
<b>termek_egysege</b>	String	A hirdetésben megjelölt termék egységét tárolja el pl.: Kg
<b>termek_mennyisege</b>	Integer	A hirdetett termék mennyiségét adja meg
<b>brutto_egysegar</b>	Integer	A feltöltött termék bruttó egységárát adja meg

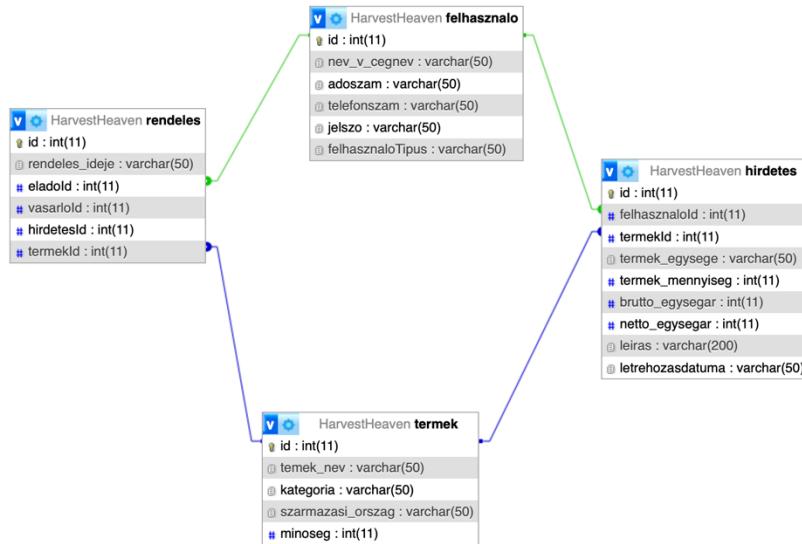
<b>netto_egysegar</b>	Integer	A feltöltött termék nettó egységárát adja meg
<b>leiras</b>	String	A hirdetés leírását tárolja el
<b>letrehozasdatum</b>	String	A hirdetés létrehozásának dátumát rögzíti

A „rendelesek” tábla mezői:

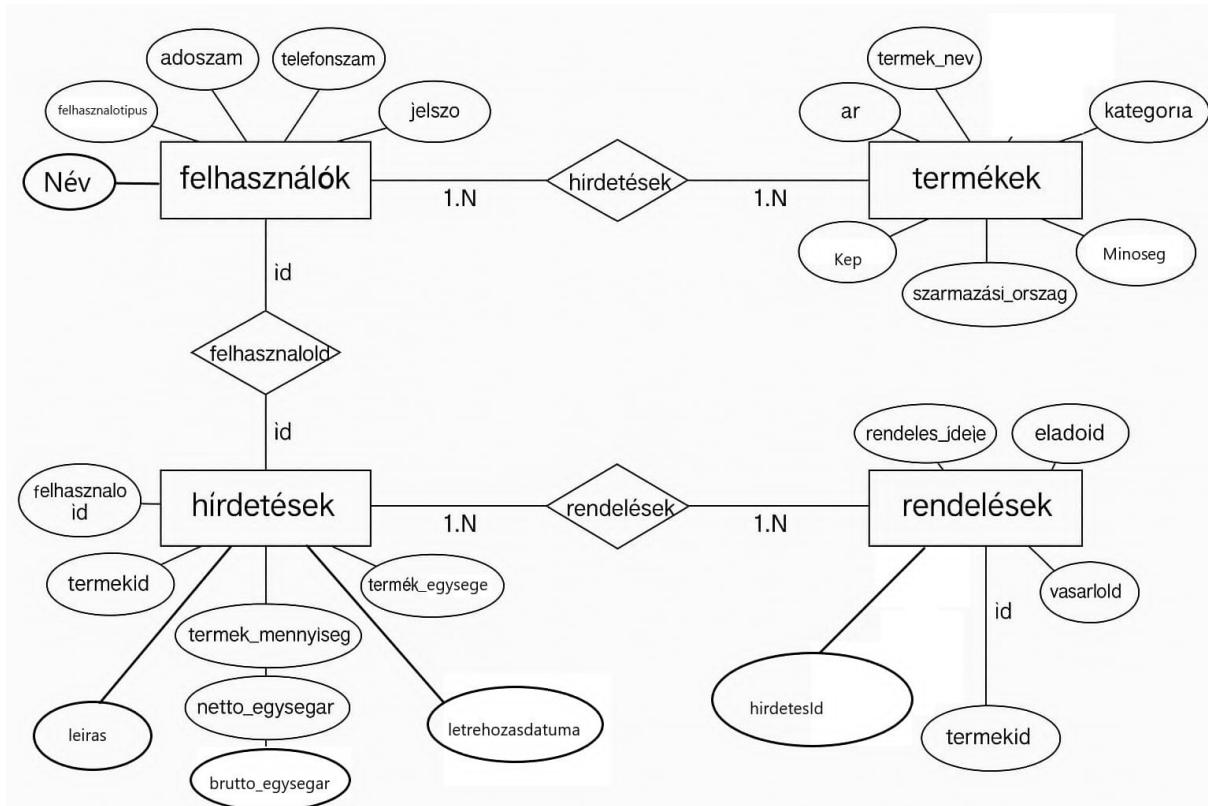
<b>Név</b>	<b>Típus</b>	<b>Feladat</b>
<b>id(PK)</b>	Integer	A rendelés sorszámát tárolja el
<b>rendeles_ideje</b>	String	A rendelés idejét tárolja el
<b>eladoId(FK)</b>	Integer	Az eladó Id-ját adja meg idegen kulcsként
<b>vasarloId</b>	Integer	A vásárló Id-ját tárolja el
<b>hirdetesId</b>	Integer	A megrendelt hirdetés Id-ját tárolja el
<b>termekId(FK)</b>	Integer	A termék sorszámát adja meg idegen kulcsként

## TÁBLÁK, ADATSZERKEZETEK BEMUTATÁSA

A weboldalunk adatbázisa több adattáblát tartalmaz. Ezek a táblák rendezettek, ezzel megkönnyítve munkánkat.



## E-K DIAGRAMM



ábra 12:E-K diagramm

# FRONTEND

Az alkalmazás Angular frontendje tiszta dizájnt követ, reszponzív kialakítással. A felhasználói felület letisztult, könnyen kezelhető. Oldal struktúra és működés:

- Kezdőlap: Áttekintés és navigáció
- Regisztráció / Bejelentkezés: Felhasználói hitelesítés

```
export class AuthService {
  private apiUrl = 'http://localhost:8000/api';
  private loggedInUser = signal<User | null>(null);
  constructor(private http: HttpClient, private router: Router) {
    try {
      const storedUser = localStorage.getItem('user');
      if (storedUser) {
        this.loggedInUser.set(JSON.parse(storedUser));
      }
    } catch (error) {
      console.error("Hibás user adatok a localStorage-ben:", error);
      localStorage.removeItem('user');
    }
  }
  register(user: User): Observable<any> {
    return this.http.post<any>(`${this.apiUrl}/felhasznalok`, user);
  }
  login(user: User): Observable<User> {
    return this.http.post<User>(`${this.apiUrl}/felhasznalok/bejelentkezes`, user).pipe(
      tap(user => {
        this.loggedInUser.set(user);
        localStorage.setItem('user', JSON.stringify(user));
        this.router.navigate(['/']);
      })
    );
  }
}
```

ábra 13: Felhasználó hitelesítése

```
export class LoginComponent {
  user: User = {nev: '', jelszo: '', telefonszam: '', adoszam: '', felhasznaloTipus: ''}; message: string = '';
  constructor(private auth: AuthService) {}
  login() {
    this.auth.login(this.user).subscribe({
      next: () => {
        alert('Sikeres bejelentkezés!');
      },
    });
  }
  logout() {
    this.auth.logout();
    alert('Kijelentkezés sikeres!');
  }
}
```

ábra 14: Be- és Kijelentkezés

```
export class RegisterComponent {
  user:User = {nev:'',adosszam:'',telefonszam:'',jelszo:'',felhasznaloTipus:''}
  constructor(private authService: AuthService, private router: Router) {}
  Windsurf: Refactor | Explain | Generate JSDoc | X
  register() {
    this.authService.register(this.user).subscribe([
      Windsurf: Refactor | Explain | Generate JSDoc | X
      next: (response) => {
        localStorage.setItem('token', response.token);
        this.router.navigate(['/login']);
      },
    ]);
  }
}
```

ábra 15: Regisztráció

- Shop: Terméklista, vásárlás

```
export class ProductService {

  private apiUrl = 'http://localhost:3000/api/termek';

  constructor(private http: HttpClient) { }

  Windsurf: Refactor | Explain | Generate JSDoc | X
  getProducts(): Observable<Termek[]> {
    return this.http.get<Termek[]>(this.apiUrl);
  }
}
```

ábra 16: Vásárlás

```

24  export class ShopComponent{
25
26    Windsurf: Refactor | Explain | Generate JSDoc | X
27    ngOnInit(): void {
28      this.httpService.gettermek().subscribe(data => [
29        this.termek = data;
30        console.log(data);
31      ]);
32    }
33
34
35    Windsurf: Refactor | Explain | Generate JSDoc | X
36    loadtermek(): void {
37      this.httpService.gettermek().subscribe(
38        (data: Termek[]) => {
39          this.termek = data;
40        },
41        (error) => {
42          console.error("Hiba történt a termékek betöltésekor:", error);
43        }
44      );
45    }
46
47    Windsurf: Refactor | Explain | Generate JSDoc | X
48    addToCart(product: Termek): void {
49      //this.cartService.addToCart(product);
50      this.cartService.addToCart(product)
51      alert(`#${product.termek_nev} hozzáadva a kosárhoz!`);
52    }
53
54    Windsurf: Refactor | Explain | Generate JSDoc | X
55    termekSzures(): void {
56      this.szurtTermek=[];
57      for (let i = 0; i < this.termek.length; i++) {
58        if (this.kiválasztottKategoria == this.termek[i].kategoria) {
59          this.szurtTermek.push(this.termek[i]);
60        }
61      }
62    }

```

ábra 17: Terméklista

- Kosár: Kosár tartalmának megtekintése és szerkesztése

```

getCart(): any[] {
  const cart = localStorage.getItem(this.cartKey);
  return cart ? JSON.parse(cart) : [];
}
Windsurf: Refactor | Explain | Generate JSDoc | X
updateCart(cartItems: any[]): void {
  localStorage.setItem(this.cartKey, JSON.stringify(cartItems));
}
Windsurf: Refactor | Explain | Generate JSDoc | X
addToCart(product: any): void {
  const cart = this.getCart();
  const existingItem = cart.find(item => item.id === product.id);
  if (existingItem) {
    existingItem.quantity++;
  } else {
    cart.push({ ...product, quantity: 1 });
  }
  this.updateCart(cart);
}
Windsurf: Refactor | Explain | Generate JSDoc | X
clearCart(): void {
  localStorage.removeItem(this.cartKey);
}
}

```

ábra 18: Kosár megtekintése

```

export class CartComponent implements OnInit {
  kosarBetoltes(): void {
    this.kosarTartalom = this.cartService.getCart();
  }

  Windsurf: Refactor | Explain | Generate JSDoc | X
  darabszamCsokken(item: any): void {
    if (item.quantity > 1) {
      item.quantity--;
    } else {
      this.eltavolitas(item);
    }
  }

  Windsurf: Refactor | Explain | Generate JSDoc | X
  darabszamNö(item: any): void {
    item.quantity++;
  }

  Windsurf: Refactor | Explain | Generate JSDoc | X
  eltavolitas(item: any): void {
    this.kosarTartalom = this.kosarTartalom.filter(cartItem => cartItem !== item);
    this.cartService.updateCart(this.kosarTartalom);
  }

  Windsurf: Refactor | Explain | Generate JSDoc | X
  osszesen(): number {
    return this.kosarTartalom.reduce((total, item) => total + item.ar * item.quantity, 0);
  }

  Windsurf: Refactor | Explain | Generate JSDoc | X
  clearCart(): void {
    localStorage.removeItem('cart');
  }

  Windsurf: Refactor | Explain | Generate JSDoc | X
  fizetes(): void {
    if (this.kosarTartalom.length > 0) {
      alert('Fizetés sikeres! 🎉');
      this.cartService.clearCart();
      this.kosarBetoltes();
    } else {
      alert('A kosárad üres! 💔');
    }
  }

  Windsurf: Refactor | Explain | Generate JSDoc | X
  sikeres(): void {
    this.cartService.addToCart(this.kosarTartalom);
    alert(`#${this.kosarTartalom} hozzáadva a kosárhoz!`);
  }
}

```

**ábra 19: Kosár tartalmának megtekintése**

### - Feltöltés: Új termék feltöltése

```

export class FeltoltesComponent {
  termekek: Termek = {termek_nev:'', ar:0, kategoria:'', szarmazasi_orszag:'', minoseg:'', kep:''}
  constructor(private httpService: HttpDataService, private authService: AuthService, private router: Router) {}

  Windsurf: Refactor | Explain | Generate JSDoc | X
  feltolt(){
    this.httpService.feltolt(this.termekek).subscribe([
      Windsurf: Refactor | Explain | Generate JSDoc | X
      next: (response) => {
        localStorage.setItem('token', response.token);
        this.router.navigate(['/shop']);
      },
    ]);
  }
}

```

**ábra 20: Új termék feltöltése**

# BACKEND

A backendben megtalálható számos metódus ami a frontendet segíti az adatbázishoz való hozzáférésben vagy éppen szerkesztésében. A backend-el tudunk adatokat menteni, lekérni, törölni vagy esetleg frissíteni az adott felhasználónál vagy éppen terméknél. Az adattáblák metódusai nagyban hasonítanak egymáshoz mivel ugyan azokra van szükség, csak más változóneveket használnak a metódusok. Ezért az első adattábla metódusainak bemutatásánál definiálva lesz az összes ott található metódus, és utána nem. Ha olyan metódus jön elő amiről még nem esett szó, az értelem szerűen definiálva lesz.

## A BACKEND FUNKCIÓI ÉS AZOKNAK BEMUTATÁSA

### A „FELHASZNALOK” TÁBLA METÓDUSAI:

```
public function login(Request $request){  
    $existingUser = Felhasznalok::where('nev', '=', $request->nev)->first();  
    if (!is_null($existingUser)) {  
        if (Hash::check($request->jelszo, $existingUser->jelszo)) {  
            return $existingUser;  
        }  
    }  
}
```

ábra 21: felhasznalok tábla: Login metódus

-Ez a login metódus. Ez segít a frontendnek bejelentkeztetni egy adott felhasználót a „nev” adatmező megvizsgálásával. Ha nemlétező névvel próbálunk bejelentkezni, a rendszer hibaüzenetet dob.

```

public function store(Request $request)
{
    $validator = Validator::make($request->all(),
    [
        'nev' => 'required',
        'adoszam' => 'required',
        'felhasznaloTipus' => 'required'
    ]);

    if($validator->fails())
    {
        return response()->json(['message' => 'Fontos adat hiányzik'], 400);
    }
    $password = Hash::make($request->jelszo);
    $felhasznalok = Felhasznalok::create([
        'nev'=>$request->nev,
        'adoszam'=>$request->adoszam,
        'telefonszam'=>$request->telefonszam,
        'felhasznaloTipus'=>$request->felhasznaloTipus,
        'jelszo'=>$password
    ]);
    return response()->json(['id' => $felhasznalok->id],202);
}

```

**ábra 22: felhasznalok tábla: Store metódus**

- Ez a „store” metódus, emberibb nevén regisztráció. Ez menti el az adattáblába az új felhasználók adatait. Vannak szükséges adatok amik nem hiányozhatnak. Ha ezek közül valamelyiket nem teljesíti az adott felhasználó akkor a regisztráció sikertelen lesz, így a rendszer hibát dob. A metódusunkba beleírtuk azt is, hogy minden egyes újonnan regisztrált felhasználónak titkosított jelszava legyen, ezáltal aki nézi az adatbázist nem láthatja a felhasználó jelszavát.

```

public function update(Request $request, $id)
{
    $felhasznalok = Felhasznalok::find($id);
    if(is_null($felhasznalok))
        return response()->json(['Hiba:'=>'A felhasználó nem létezik!'],404);

    $validator = Validator::make($request->all(),
    [
        'nev' => 'required',
        'adoszam' => 'required',
        'felhasznaloTipus' => 'required'
    ]
    );
    if($validator->fails()){
        return response()->json(['Hiba' => 'Fontos adatok hiányoznak!'],402);
    }

    $felhasznalok -> update($request->all());
    return response()->json(['Következő id-jű Felhasználó adatai megváltoztak' => $felhasznalok->i
}

```

**ábra 23: felhasznalok tábla: update metódus**

-Ez az update metódus. Egy id alapján bekér egy felhasználót, majd az adott felhasználónak az adatait frissíti az adattáblában. Persze az updatenek is vannak bizonyos mezői amelyeket kötelező kitölteni.

```

public function delete($id)
{
    $felhasznalok = Felhasznalok::where('id','','=',$id);
    if($felhasznalok->exists())
    {
        $felhasznalok->delete();
        return response('A felhasználó törölve lett!',204);
    }
    return response('A felhasználó nem létezik!',404);
}

```

**ábra 24: felhasznalok tábla: delete metódus**

-Ez a „delete” metódus. Ez a legegyszerűbb metódus amellyel minden egyes adattábla rendelkezik. Lekér Id alapján egy felhasználót és törli az adatbázisból. Ha olyan felhasználót próbálunk törölni mi nem létezik, a rendszer hibaüzenetet fog dobni.

```

public function getById($id)
{
    $felhasznalok = Felhasznalok::find($id);
    if(is_null($felhasznalok))
    {
        return response()->json(['Hiba:'=>'A felhasználó nem található!'],404);
    }

    return response()->json($felhasznalok,201);
}

```

**ábra 25: felhasznalok tábla: getById metódus**

-Ez a „geById” metódus ami annyit jelent hogy egy adott felhasználó adatait tudjuk lekérdezni az Id alapján. Ha olyan felhasználót kérdezünk le ami nem létezik, akkor. Rendszer hibaüzenetet dob.

## A „TERMEKEK” TÁBLA METÓDUSAI:

```

public function store(Request $request)
{
    $validator = Validator::make($request->all(),
    [
        'termek_nev' => 'required',
        'kategoria' => 'required',
        'szarmazasi_orszag'=> 'required'
    ]);

    if($validator->fails())
    {
        return response()->json(['message' => 'Fontos adat hiányzik'], 400);
    }

    $termek = Termek::create($request->all());
    return response()->json(['id' => $termek->id],202);
}

```

**ábra 26: termek tábla: store metódus**

-A termékek tábla „store” metódusa. A termékeket menti el, a szükséges adatmezők megadásával.

```

public function update(Request $request, $id)
{
    $termek = Termek::find($id);
    if(is_null($termek))
        return response()->json(['Hiba' => 'A termék nem létezik!'], 404);

    $validator = Validator::make($request->all(),
    [
        'termek_nev' => 'required',
        'kategoria' => 'required'
    ]);
    if($validator->fails()){
        return response()->json(['Hiba' => 'Fontos adatok hiányoznak!'], 402);
    }

    $termek->update($request->all());
    return response()->json(['Következő id-jű Termék adatai megváltoztak' => $termek->id], 201);
}

```

**ábra 27: termek tábla: update metódus**

-A termékek „update” metódusa ezzel lehet frissíteni egy már feltöltött terméke adatait.

```

public function delete($id)
{
    $termek = Termek::where('id', '=', $id);
    if($termek->exists())
    {
        $termek->delete();
        return response('A termék törölve lett', 204);
    }
    return response('A termék nem létezik!', 404);
}

```

**ábra 28: termek tábla: delete metódus**

-A tábla „delete” metódusa. Ezzel lehet törölni egy adott terméket.

```

public function getById($id)
{
    $termek = Termek::find($id);
    if(is_null($termek))
    {
        return response()->json(['Hiba'=>'A termék nem található!'],404);
    }

    return response()->json($termek,201);
}

```

ábra 29: termek tábla: getById metódus

-A tábla „getById” metódusa. Ezzel lehet a feltöltött termékeket Id alapján lekérdezni.

## A „HIRDETESEK” TÁBLA METÓDUSAI

```

public function store(Request $request)
{
    $validator = Validator::make($request->all(),
    [
        'Termek' => 'required',
        'termek_egysege' => 'required',
        'brutto_egysegar' => 'required'
    ]);

    if($validator->fails())
    {
        return response()->json(['message' => 'Fontos adat hiányzik'], 400);
    }

    $hirdetesek = Hirdetes::create($request->all());
    return response()->json(['id' => $hirdetesek->id],202);
}

```

ábra 30: hirdetesek tábla: store metódus

-A tábla „store” metódusa.

```

public function update(Request $request, $id)
{
    $hirdetesek = Hirdetes::find($id);
    if(is_null($hirdetesek))
        return response()->json(['Hiba' => 'A hirdetés nem létezik!'], 404);

    $validator = Validator::make($request->all(),
    [
        'Termek' => 'required',
        'termek_egysege' => 'required',
        'brutto_egysegar' => 'required'
    ]);
    if($validator->fails()){
        return response()->json(['Hiba' => 'Fontos adatok hiányoznak!'], 402);
    }

    $hirdetesek -> update($request->all());
    return response()->json(['Következő id-jű Hirdetés adatai megváltoztak' => $hirdetesek->id], 201);
}

```

**ábra 31: hirdetesek tábla: update metódus**

-A tábla „update” metódusa.

```

public function delete($id)
{
    $hirdetesek = Hirdetes::where('id', '=', $id);
    if($hirdetesek->exists())
    {
        $hirdetesek->delete();
        return response('A hirdetés törölve lett', 204);
    }
    return response('A hirdetés nem létezik!', 404);
}

```

**ábra 32: hirdetesek tábla: delete metódus**

-A tábla „delete” metódusa.

```
public function getById($id)
{
    $hirdetesek = Hirdetes::find($id);
    if(is_null($hirdetesek))
    {
        return response()->json(['Hiba:'=>'A hirdetés nem található!'],404);
    }

    return response()->json($hirdetesek,201);
}
```

ábra 33: hirdetesek tábla: getById metódus

-A tábla „getById” metódusa.

```
public function getMoreExpensive($brutto_egysegar){
    $hirdetesek = Hirdetes::where('brutto_egysegar', '>', $brutto_egysegar);
    if($hirdetesek->exists())
    {
        return $hirdetesek->get();
    }
    else
    {
        return response("Nem található az adott árnál drágább/nagyobb árú hirdetés!", 402);
    }
}
```

ábra 34: hirdetesek tábla: getMoreExpensive metódus

-A „hirdetesek” tábla „getMoreExpensive” metódusa. Ezzel a metódussal lehet szűrni a hirdetett termékek között. Bekéri a hirdetésben feltüntetett „brutto\_egysegar” mezőt. A felhasználó megad egy bruttó egységárat és a metódus az ennél nagyobb áron listázott elemeket fogja mutatni.

```

public function getCheaper($brutto_egysegar){
    $hirdetesek = Hirdetes::where('brutto_egysegar', '<', $brutto_egysegar);
    if($hirdetesek->exists())
        return $hirdetesek->get();
    else
        return response("Nem található az adott árnál olcsóbb/kissebb árú hirdetés!", 402);
}

```

**ábra 35:** hirdetesek tábla: getCheaper metódus

-Ez a tábla „getCheaper” metódusa. Ez ugyan azon az alapon működik, mint az előzőleg leírt metódus. A felhasználó megad egy bruttó egységárát, és a metódus az ennél kisebb árú listaelmeket fogja mutatni.

## A „RENDELESEK” TÁBLA METÓDUSAI

```

public function store(Request $request)
{
    $validator = Validator::make($request->all(),
    [
        'Termek' => 'required',
        'Elado' => 'required',
        'Vasarlo' => 'required'
    ]);

    if($validator->fails())
    {
        return response()->json(['message' => 'Fontos adat hiányzik'], 400);
    }

    $rendelesek = Rendeles::create($request->all());
    return response()->json(['id' => $rendelesek->id], 202);
}

```

**ábra 36:** rendelesek tábla: store metódus

-Ez a „rendelesek” tábla „store” metódusa. Ezzel a metódussal veszünk fel új rendeléseket az adatbázisunkba.

```
public function getById($id)
{
    $rendelesek = Rendeles::find($id);
    if(is_null($rendelesek))
    {
        return response()->json(['Hiba:'=>'A rendelés nem található!'],404);
    }

    return response()->json($rendelesek,201);
}
```

ábra 37: rendelesek tábla: getById metódus

-Ez a tábla „getById” metódusa. Ezzel a metódussal a rendelések tába elemeit tudjuk lekérdezni.

```
public function delete($id)
{
    $rendelesek = Rendeles::where('id','','=',$id);
    if($rendelesek->exists())
    {
        $rendelesek->delete();
        return response('A rendelés törölve lett',204);
    }
    return response('A rendelés nem létezik!',404);
}
```

ábra 38: rendelesek tábla: delete metódus

-Ez pedig a tábla „delete” metódusa. Ezzel töröljük a rendeléseket.

## A METÓDUSOK ÚTVONALAI

```
Route::get('/felhasznalok',[FelhasznalokController::class,'index']);
Route::post('/felhasznalok',[FelhasznalokController::class, 'store']);
Route::post('/felhasznalok/bejelentkezes',[FelhasznalokController::class,'login']);
Route::put('/felhasznalok/{id}',[FelhasznalokController::class,'update']);
Route::delete('/felhasznalok/{id}',[FelhasznalokController::class,'delete']);
Route::get('/felhasznalok/{id}',[FelhasznalokController::class,'getById']);

Route::get('/hirdetesek',[HirdetesController::class,'index']);
Route::post('/hirdetesek',[HirdetesController::class,'store']);
Route::put('/hirdetesek/{id}',[HirdetesController::class,'update']);
Route::delete('/hirdetesek/{id}',[HirdetesController::class,'delete']);
Route::get('/hirdetesek/{id}',[HirdetesController::class,'getById']);
Route::get('/hirdetesek/dragabb/{brutto_egysegar}',[HirdetesController::class,'getMoreExpensive']);
Route::get('/hirdetesek/olcsobb/{brutto_egysegar}',[HirdetesController::class,'getCheaper']);

Route::get('/termek',[TermekController::class,'index']);
Route::post('/termek',[TermekController::class, 'store']);
Route::put('/termek/{id}',[TermekController::class,'update']);
Route::delete('/termek/{id}',[TermekController::class,'delete']);
Route::get('/termek/{id}',[TermekController::class,'getById']);

Route::get('/rendelesek',[RendelesController::class,'index']);
Route::post('/rendelesek',[RendelesController::class,'store']);
Route::get('/rendelesek/{id}',[RendelesController::class,'getById']);
Route::delete('/rendelesek/{id}',[RendelesController::class,'delete']);
```

ábra 39: Útvonalak

-Ezek a metódusok útvonalai. Ezen keresztül éri el egy adott metódus a használni kívánt adattáblát.

# TESZTELÉS

A tesztelésünk a Frontenben „ng teste”-el folyt illetve Backendn esetében a Thunder Client API tesztelőt alkalmaztuk.

## A FRONTEND TESZTJE

The screenshot shows a Jasmine 4.6.1 test runner interface with the following details:

- Specs:** 17 specs, 3 failures, randomized with seed 743899.
- Spec List:** Spec 1 failed: `HttpDataService > should be created`.
- Failure Details:** `NullInjectorError: No provider for HttpClient!`
- Stack Trace:** Shows multiple error entries related to HttpClient injection across three modules: `HttpDataService`, `AuthenticationService`, and `ProductService`. Each entry includes a URL (e.g., `http://localhost:9876/_karma_webpack_node_modules/@angular/core/fesm02/core.mjs:1111:32`), a file name (e.g., `HttpClientModule.js`), and a line number (e.g., `19`).
- Spec 2 failed: `AuthenticationService > should be created`.**
- Spec 3 failed: `ProductService > should be created`.**
- Stack Trace for Spec 2:** `NullInjectorError: No provider for HttpClient!`
- Stack Trace for Spec 3:** `Unexpected value 'ProductService' imported by the module 'DynamicTestingModule'. Please add an @NgModule annotation.`

ábra 40: Frontend teszt

# A BACKEND TESZTJE

```
1  Users > szurdokbence > Desktop > ⚡ thunder-collection_HHfelhasznalok.json > ...
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
```

ábra 41: felhasznalok tábla teszt (1.kép)

```
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
```

ábra 42: felhasznalok tábla teszt (2.kép)

-Ezek a felhasználók tábla tesztjei

```

1  {
2      "clientName": "Thunder Client",
3      "collectionName": "Hirendelesek",
4      "collectionId": "75259791-5c47-4056-a243-26f3289adc32",
5      "dateExported": "2025-04-28T07:26:22.757Z",
6      "version": "1.2",
7      "folders": [],
8      "requests": [
9          {
10             "_id": "417875df-ee26-452a-bd69-d188e6572688",
11             "colId": "75259791-5c47-4056-a243-26f3289adc32",
12             "containerId": "",
13             "name": "getRendelesek",
14             "url": "http://127.0.0.1:8000/api/rendelesek",
15             "method": "GET",
16             "sortNum": 10000,
17             "created": "2025-03-25T07:34:46.139Z",
18             "modified": "2025-03-25T07:34:56.413Z",
19             "headers": []
20         },
21         {
22             "_id": "94bc24dc-10ed-4345-89c5-8b0ed913aeab",
23             "colId": "75259791-5c47-4056-a243-26f3289adc32",
24             "containerId": "",
25             "name": "postRendelesek",
26             "url": "http://127.0.0.1:8000/api/rendelesek",
27             "method": "POST",
28             "sortNum": 20000,
29             "created": "2025-04-28T06:49:45.947Z",
30             "modified": "2025-04-28T07:17:57.606Z",
31             "headers": [],
32             "body": {
33                 "type": "json",
34                 "raw": "\n    \"TermekId\":1,\n    \"eladoId\":1,\n    \"vasarloId\":1,\n    \"HirdetesId\":1\n}\n",
35                 "form": {}
36             }
37         },
38         {
39             "_id": "6a8c107e-340f-4545-8ea2-2f6446f223b6",
40             "colId": "75259791-5c47-4056-a243-26f3289adc32",
41             "containerId": "",
42             "name": "getByIdRendelesek",
43             "url": "http://127.0.0.1:8000/api/rendelesek/5",
44             "method": "GET",
45             "sortNum": 30000,
46             "created": "2025-04-28T07:18:20.911Z",
47             "modified": "2025-04-28T07:18:30.220Z",
48             "headers": []
49         },
50         {
51             "_id": "4b751aaf-84d2-41d8-bb1f-c147f402c8f8",
52             "colId": "75259791-5c47-4056-a243-26f3289adc32",
53             "containerId": "",
54             "name": "deleteRendelesek",
55             "url": "http://127.0.0.1:8000/api/rendelesek/5",
56             "method": "DELETE",
57             "sortNum": 40000,
58             "created": "2025-04-28T07:18:44.250Z",
59             "modified": "2025-04-28T07:18:53.102Z",
60             "headers": []
61         }
62     ],
63     "ref": "my0_kKr0d9_AxInC-FpWNG7-wPeMGJOYHhZmH2LWxpBlaoPrDrd3xJzimz30mCWRPSb4gM2jmEnqu0PwBnunNbQ"
64 }

```

ábra 43:rendelesek tábla teszt (1.kép)

```

38  {
39      "_id": "6a8c107e-340f-4545-8ea2-2f6446f223b6",
40      "colId": "75259791-5c47-4056-a243-26f3289adc32",
41      "containerId": "",
42      "name": "getByIdRendelesek",
43      "url": "http://127.0.0.1:8000/api/rendelesek/5",
44      "method": "GET",
45      "sortNum": 30000,
46      "created": "2025-04-28T07:18:20.911Z",
47      "modified": "2025-04-28T07:18:30.220Z",
48      "headers": []
49 },
50 {
51     "_id": "4b751aaf-84d2-41d8-bb1f-c147f402c8f8",
52     "colId": "75259791-5c47-4056-a243-26f3289adc32",
53     "containerId": "",
54     "name": "deleteRendelesek",
55     "url": "http://127.0.0.1:8000/api/rendelesek/5",
56     "method": "DELETE",
57     "sortNum": 40000,
58     "created": "2025-04-28T07:18:44.250Z",
59     "modified": "2025-04-28T07:18:53.102Z",
60     "headers": []
61 }
62 ],
63 "ref": "my0_kKr0d9_AxInC-FpWNG7-wPeMGJOYHhZmH2LWxpBlaoPrDrd3xJzimz30mCWRPSb4gM2jmEnqu0PwBnunNbQ"
64 }

```

ábra 44: rendelesek tábla teszt (2.kép)

-Ezek a rendelesek tábla tesztjei

```

Users > szurdokbence > Desktop > thunder-collection_HHirdetesek.json > ...
1
2   "clientName": "Thunder Client",
3   "collectionName": "HHirdetesek",
4   "collectionId": "753141c1-f98b-47a7-92d9-9868d8ac473c",
5   "dateExported": "2025-04-28T07:26:09.832Z",
6   "version": "1.2",
7   "folders": [],
8   "requests": [
9     {
10       "_id": "dd64d7f4-dd13-423f-b6ca-3d0ccf865b02",
11       "colId": "53141c1-f98b-47a7-92d9-9868d8ac473c",
12       "containerId": "",
13       "name": "getHirdetesek",
14       "url": "http://127.0.0.1:8000/api/hirdetesek",
15       "method": "GET",
16       "sortNum": 10000,
17       "created": "2025-03-25T07:34:08.654Z",
18       "modified": "2025-03-25T07:34:25.181Z",
19       "headers": []
20     },
21     {
22       "_id": "3fba994d-4cb2-4989-b9c-c1094a0b6e0",
23       "colId": "53141c1-f98b-47a7-92d9-9868d8ac473c",
24       "containerId": "",
25       "name": "postHirdetesek",
26       "url": "http://127.0.0.1:8000/api/hirdetesek",
27       "method": "POST",
28       "sortNum": 20000,
29       "created": "2025-04-28T07:20:28.580Z",
30       "modified": "2025-04-28T07:21:32.501Z",
31       "headers": [],
32       "body": {
33         "type": "json",
34         "raw": "{\n    \"FelhasznaloK\":1,\n    \"Termek\":1,\n    \"termek_egysege\":4,\n    \"Termek_mennyisege\":20,\n    \"brutto_egysegar\":600\n}",
35         "form": {}
36       }
37     },
38     {
39       "_id": "34e62422-a885-4813-a6c7-12b3e8f6c331",
40       "colId": "53141c1-f98b-47a7-92d9-9868d8ac473c",
41       "containerId": "",
42       "name": "getByIdHirdetesek",
43       "url": "http://127.0.0.1:8000/api/hirdetesek/2",
44       "method": "GET",
45       "sortNum": 20000,
46       "created": "2025-04-28T07:22:55.070Z",
47       "modified": "2025-04-28T07:23:13.617Z",
48       "headers": []
49     },
50   ],
51 
```

ábra 45: hirdetesek tábla teszt (1.kép)

```

Users > szurdokbence > Desktop > thunder-collection_HHirdetesek.json > ...
8   "requests": [
9     {
10       "_id": "6fc11f87-0811-4bc1-b5aa-c88b18aab00c",
11       "colId": "53141c1-f98b-47a7-92d9-9868d8ac473c",
12       "containerId": "",
13       "name": "getCheaperHirdetesek",
14       "url": "http://127.0.0.1:8000/api/hirdetesek/olcsobb/400",
15       "method": "GET",
16       "sortNum": 25000,
17       "created": "2025-04-28T07:23:31.820Z",
18       "modified": "2025-04-28T07:25:11.531Z",
19       "headers": []
20     },
21     {
22       "_id": "80ac5018-0fc6-4647-b841-db7c0248c530",
23       "colId": "53141c1-f98b-47a7-92d9-9868d8ac473c",
24       "containerId": "",
25       "name": "updateHirdetesek",
26       "url": "http://127.0.0.1:8000/api/hirdetesek/2",
27       "method": "PUT",
28       "sortNum": 30000,
29       "created": "2025-04-28T07:21:43.571Z",
30       "modified": "2025-04-28T07:22:36.970Z",
31       "headers": [],
32       "body": {
33         "type": "json",
34         "raw": "{\n    \"FelhasznaloK\":1,\n    \"Termek\":1,\n    \"termek_egysege\":5,\n    \"Termek_mennyisege\":10,\n    \"brutto_egysegar\":650\n}",
35         "form": {}
36       }
37     },
38     {
39       "_id": "abbab77f-c74b-4e94-bcd7-49171fcfb25f",
40       "colId": "53141c1-f98b-47a7-92d9-9868d8ac473c",
41       "containerId": "",
42       "name": "getMoreExpensiveHirdetesek",
43       "url": "http://127.0.0.1:8000/api/hirdetesek/dragabb/150",
44       "method": "GET",
45       "sortNum": 30000,
46       "created": "2025-04-28T07:23:29.219Z",
47       "modified": "2025-04-28T07:24:33.680Z",
48       "headers": []
49     },
50     {
51       "_id": "7866f452-0d4b-4be5-b339-00f334dac24c",
52       "colId": "53141c1-f98b-47a7-92d9-9868d8ac473c",
53       "containerId": "",
54       "name": "deleteHirdetesek",
55       "url": "http://127.0.0.1:8000/api/hirdetesek/2",
56       "method": "DELETE",
57       "sortNum": 40000,
58       "created": "2025-04-28T07:25:28.315Z",
59       "modified": "2025-04-28T07:25:37.084Z"
60     }
61   ],
62 
```

In 1 Col 1 Spaces

ábra 46: hirdetesek tábla teszt (2. kép)

-Ezek a hirdetesek tábla tesztjei

```

Users > szurdokbence > Desktop > thunder-collection_HHtermeket.json > ...
1
2   "clientName": "Thunder Client",
3   "collectionName": "HHtermeket",
4   "collectionId": "9026d134-d9e1-4bab-aac0-bff87b11da56",
5   "dateExported": "2025-04-28T07:25:48.722Z",
6   "version": "1.2",
7   "folders": [],
8   "requests": [
9     {
10       "_id": "7ef7131c-6aff-4b2c-babf-b08a24b111c5",
11       "colId": "9026d134-d9e1-4bab-aac0-bff87b11da56",
12       "containerId": "",
13       "name": "hh_getpi",
14       "url": "http://127.0.0.1:8000/api/termeket",
15       "method": "GET",
16       "sortNum": 10000,
17       "created": "2025-02-21T09:57:16.647Z",
18       "modified": "2025-03-25T07:33:41.615Z",
19       "headers": []
20     },
21     {
22       "_id": "5f77caff-de22-41d6-942a-7d494b78d708",
23       "colId": "9026d134-d9e1-4bab-aac0-bff87b11da56",
24       "containerId": "",
25       "name": "hh_post",
26       "url": "http://127.0.0.1:8000/api/termeket",
27       "method": "POST",
28       "sortNum": 20000,
29       "created": "2025-03-25T07:22:37.005Z",
30       "modified": "2025-03-25T07:32:56.786Z",
31       "headers": [],
32       "body": {
33         "type": "json",
34         "raw": "{\n  \"termek_nev\": \"sz\u00f3l\u00f3\", \n  \"kategoria\": \"gy\u00f6rcs\", \n  \"szarmazasi_orszag\": \"\u0410\u0433\u043d\u0430\u043b\u043e\u0436\u0435\u043d\u0438\u044f\", \n  \"minoseg\": 1\n}",
35         "form": []
36       }
37     },
38   ],
39

```

**ábra 47: termeket tábla teszt (1. kép)**

```

38
39   {
40     "_id": "0f01d3f7-4af7-46e7-bac4-cdca7bef5991",
41     "colId": "9026d134-d9e1-4bab-aac0-bff87b11da56",
42     "containerId": "",
43     "name": "termeket_uodate",
44     "url": "http://127.0.0.1:8000/api/termeket/4",
45     "method": "PUT",
46     "sortNum": 30000,
47     "created": "2025-03-25T07:35:55.704Z",
48     "modified": "2025-03-25T21:37:00.688Z",
49     "headers": [],
50     "body": {
51       "type": "json",
52       "raw": "{\n  \"termek_nev\": \"Uborka\", \n  \"kategoria\": \"\u0410\u0433\u043d\u0430\u043b\u043e\u0436\u0435\u043d\u0438\u044f\"\n}",
53     }
54   },
55   {
56     "_id": "22994896-ebab-46b3-8d28-45b1279d9353",
57     "colId": "9026d134-d9e1-4bab-aac0-bff87b11da56",
58     "containerId": "",
59     "name": "destroy_termeket",
60     "url": "http://127.0.0.1:8000/api/termeket/5",
61     "method": "DELETE",
62     "sortNum": 40000,
63     "created": "2025-03-25T07:40:43.134Z",
64     "modified": "2025-03-25T21:40:24.201Z",
65     "headers": []
66   },
67   {
68     "_id": "0d0fe088-7a37-401e-9a05-c736403c8c9b",
69     "colId": "9026d134-d9e1-4bab-aac0-bff87b11da56",
70     "containerId": "",
71     "name": "getByIdTermeket",
72     "url": "http://127.0.0.1:8000/api/termeket/5",
73     "method": "GET",
74     "sortNum": 50000,
75     "created": "2025-04-28T07:19:39.633Z",
76     "modified": "2025-04-28T07:19:49.320Z",
77     "headers": []
78   },
79   ],
80   "ref": "r8FwlCCVXfjYrSwHktA63DsQ_U4Hch_Uqv0D7zyryJ80sJM7RjxR3j2M030dRTyzjCM1eb8tcP0mWtIXQ0kZvg"
81

```

**ábra 48: termeket tábla teszt (2.kép)**

-Ezek a termeket tábla tesztjei-

## TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

A tovább fejlesztési lehetőségek palettája elég nagy. Számtalan lehetőség van mellyel növelni lehet a felhasználói élményt. Ezek a lehetőségek pedig itt láthatók:

- Felhasználói fiókok kiegészítése profilképpel, eladói és vásárlói visszajelzésekkel.
- Termékszűrés több paraméter alapján.
- Biztonságosabb adatkezelés, kétfaktoros hitelesítés (2FA) a bejelentkezéshez.
- Értesítések: email vagy üzenet értesítés új vásárlókról/termékekéről.
- Termékek kiemelése: fizetett hirdetések vagy termékek kiemelési lehetősége.
- Mobilalkalmazás létrehozása.
- Több felhasználó és több termék esetén adatbázis optimalizálás a több adat kezelésére.
- Külső fejlesztők vagy mobilalkalmazások számára nyitott API.

## IRODALOMJEGYZÉK

<https://www.w3schools.com/>

<https://material.angular.io/>

<https://getbootstrap.com/docs/5.3/getting-started/introduction/>

<https://nodejs.org/en>