

Webfejlesztő Vizsgadolgozat

Huszár László
2024.03.11.

1 TARTALOM

2 Fejlesztői dokumentáció	3
2.1 A szoftver célja	3
2.2 Fejlesztői környezet	3
2.2.1 A fejlesztői környezethez szükséges szoftverek	4
2.3 Adatbázis	6
2.3.1 Egyed-kapcsolat diagram	6
2.3.2 Categories tábla	7
2.3.3 Food_items tábla	7
2.3.4 Users tábla	8
2.3.5 Addresses tábla	9
2.3.6 Carts tábla	9
2.3.7 Cart_items tábla	10
2.3.8 Orders tábla	11
2.3.9 Order_addresses tábla	12
2.3.10 Order_items tábla	12
2.3.11 Order_statuses tábla	13
2.3.12 Employee tábla	14
2.4 Program felépítése	15
2.4.1 Felhasználói azonosítás	15
2.4.2 Laravel keretrendszer	15
2.4.3 Példa az adatbázis táblák létrehozására	16
2.4.4 Példa az Eloquent model-re	17
2.4.5 Példa az Eloquent model CRUD műveleteire:	19
2.4.6 Algoritmusok	24
3 Felhasználói dokumentáció	28
3.1 Vásárlói felület	28

3.1.1 Regisztráció	28
3.1.2 Belépés	29
3.1.3 Étel rendelése	30
3.1.4 Szállítási címek	32
3.1.5 Rendelések megtekintése	34
3.2 Munkavállalói felület	36
3.2.1 Operátori munkakör	36
3.2.2 Futár munkakör	37
3.3 Admin felület	37
3.3.1 Étlap szerkesztése	37
3.3.2 Ételcsoport vagy étel megjelenési sorrendjének változtatása az étlapon	41
3.3.3 Munkavállalói fiókok	41

2 FEJLESZTŐI DOKUMENTÁCIÓ

2.1 A SZOFTVER CÉLJA

Egy olyan vendéglátói webalkalmazás fejlesztése a cél, ahol az online vásárlás lehetősége a vendégek részéről, a rendelések feldolgozása a munkavállalók részéről, illetve az étlap, a munkavállalói fiókok és a rendelések szerkesztése, lekérdezése tulajdonosi részről egy alkalmazáson belül valósuljon meg. Használható legyen asztali számítógépen és mobil telefonon is. A munkavállalók az oldal frissítése nélkül lássák az új beérkezett rendeléseket és azok állapotának változását.

2.2 FEJLESZTŐI KÖRNYEZET

Laravel 10 fejlesztői környezetet választottam mert tartalmaz minden eszközt, ami egy egyoldalas alkalmazás (SPA) elkészítéséhez szükséges, ráadásul a szerver és kliens oldali fejlesztést egy projekt könyvtár alatt lehet elvégezni.

- Docker 25.0.3
- Visual Studio Code 1.87.0
- Laravel 10 PHP keretrendszer:
 - A Laravel projekt laravel sail-el lett létrehozva.
 - Laravel Breeze-el készült el a projekt váza, ami tartalmaz:
 - Alap felhasználói regisztráció és beléptetés funkciót, a hozzátartozó react komponensekkel és felhasználói fiók oldallal. Az adatbázisban létrehoz egy users táblát.
 - React frontend fejlesztői környezet
 - Inertia JS, ami az adapter a szerver és a kliens oldal között.
 - Tailwind CSS

Laravel sail-el hoztam létre a projektet, amely konténerekbe telepít mindent, ami a Laravel környezethez szükséges, így az operációs rendszerre semmit sem szükséges felrakni. Konténerek:

- PHP futtatói környezet v8.3
- MySQL Server v8.0
- Socketl v16

2.2.1 A fejlesztői környezethez szükséges szoftverek

- Docker Desktop: Windows rendszereken szükséges telepíteni a dockerhez a wsl linux futtatói környezetet. Adminisztrátorként megnyitva egy command promptot:

```
wsl --install
```

Ez feltelepíti az ubuntu-t windowsra.

- A docker telepítése után a docker beállításokban: settings -> resources -> wsl integration -> ubuntu -t szükséges bekapcsolni.
- windowsban megnyitva az ubuntu command promptot:

```
git clone ...
```

- A projekt könyvtárba belépve a projekthez szükséges vendor könyvtár telepítése:

```
docker run --rm \  
-u "$(id -u):$(id -g)" \  
-v "$(pwd):/var/www/html" \  
-w /var/www/html \  
laravelsail/php83-composer:latest \  
composer install --ignore-platform-reqs
```

- VS Code-ban a wsl bővítmény telepítése után megnyitva a projektet, majd egy terminál ablakban beírva

```
./vendor/bin/sail up
```

majd egy másik terminál ablakban

```
./vendor/bin/sail npm install
```

```
./vendor/bin/sail npm run dev
```

indul el a szerver és kliens oldal, majd a projekt beállításai egy harmadik terminál ablakban:

```
./vendor/bin/sail artisan migrate:fresh --seed
```

```
./vendor/bin/sail artisan storage:link
```

- Ezután <http://localhost> érhető el a weboldal.

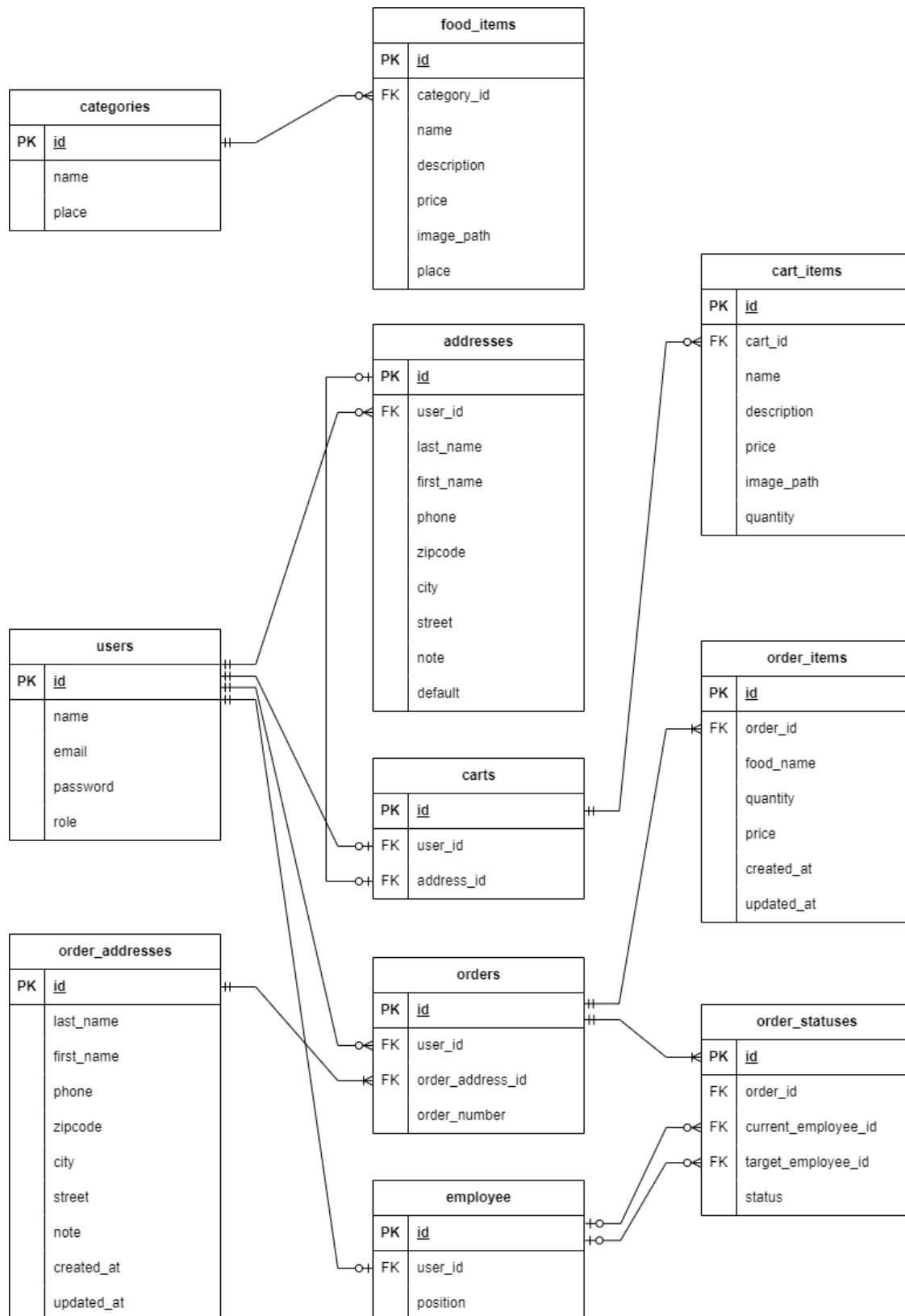
2.2.2 A weboldal tesztelése

Tesztelés céljából létrehozott felhasználói fiókok:

- Vásárló:
Email: customer1@test.com
Jelszó: password
- Operátor:
Email: operator1@test.com
Jelszó: password
- Futár:
Email: courier1@test.com
Jelszó: password
- Admin:
Email: admin@test.com
Jelszó: password

2.3 ADATBÁZIS

2.3.1 Egyed-kapcsolat diagram



2.3.2 Categories tábla

<i>Mező</i>	<i>TÍPUS, MEGKÖTÉS</i>
id	bigint, unsigned, not null, auto_increment, primary key
name	varchar(255), not null
place	int, unsigned, not null

A categories tábla tárolja az étlapon szereplő ételcsoportokat.

A name mező tartalmazza az ételcsoport nevét, a place mező az ételcsoportok megjelenési sorrendjét az étlapon.

2.3.3 Food_items tábla

<i>Mező</i>	<i>TÍPUS, MEGKÖTÉS</i>
id	bigint, unsigned, not null, auto_increment, primary key
category_id	bigint, unsigned, not null, foreign key, on delete cascade
name	varchar(255), not null
description	varchar(1000), default null
price	int, unsigned, not null
image_url	varchar(1000), default null
place	int, unsigned, not null

A food_items tábla az étlapon szereplő ételek adatait tárolja.

A category_id idegen kulcs határozza meg, hogy az egyes ételek melyik ételcsoportba tartoznak. On delete cascade megkötést tartalmaz, azaz egy adott ételcsoport bejegyzés törlésével az összes, hozzá tartozó étel is törlésre kerül.

A name mező az étel nevét, a description mező az étel összetevőit, leírását, a price mező az étel árát, az image_path mező az ételhez tartozó kép útvonalát, a place mező az étel az adott ételcsoporton belüli sorrendjét tárolja.

2.3.4 Users tábla

<i>Mező</i>	<i>TÍPUS, MEGKÖTÉS</i>
id	bigint, unsigned, not null, auto_increment, primary key
email	varchar(255), not null
password	varchar(255), not null
role	enum('customer', 'admin', 'employee'), not null

A users tábla a regisztrált felhasználók belépési adatait és felhasználói jogait tárolja. Az email mező a felhasználó email címét, a password mező a felhasználó jelszavát tárolja. A jelszó hashelt formában kerül tárolásra. A role a felhasználói jogokat tárolja. A felületen új felhasználóként regisztrálva a rendszer alapból customer jogot ad. Employee és admin joggal csak admin joggal rendelkező felhasználó regisztrálhat. Egy darab admin joggal rendelkező felhasználót tartalmaz az adatbázis.

- customer: a felületen regisztrált felhasználók, akik ételt rendelnek az oldalon
- employee: admin által regisztrált felhasználók, akik az étterem dolgozói
- admin: egy darab létező, illetve admin által regisztrált felhasználók, akik szerkesztik az étlapot, rendelésekkel kapcsolatos adatokat kérdeznek le és szerkesztik a munkavállalói fiókokat

2.3.5 Addresses tábla

Mező	TÍPUS, MEGKÖTÉS
id	bigint, unsigned, not null, auto_increment, primary key
user_id	bigint, unsigned, not null, foreign key, on delete cascade
last_name	varchar(255), not null
first_name	varchar(255), not null
phone	varchar(255), not null
zipcode	varchar(4), not null
city	varchar(255), not null
street	varchar(255), not null
note	longtext
default	tinyint(1) not null

Az addresses tábla tárolja egy adott felhasználóhoz tartozó szállítási címeket. A user_id idegen kulcs kapcsolja az adott címet egy felhasználóhoz. Egy felhasználónak több szállítási címe is lehet. Egy adott felhasználó törlésével, az összes hozzá tartozó szállítási cím is törlésre kerül. A last_name, first_name, phone, zipcode, city, street és note mezők a szállítási adatokat tartalmazzák, sorrendben: vezetéknév, keresztnév, telefonszám, irányítószám, város, utca és szállítással kapcsolatos megjegyzés. A default mező értéke 0 vagy 1 lehet, az alapértelmezett szállítási címet jelöli. Egy felhasználónak csak egy alapértelmezett szállítási címe lehet.

2.3.6 Carts tábla

Mező	TÍPUS, MEGKÖTÉS
id	bigint, unsigned, not null, auto_increment, primary key
user_id	bigint, unsigned, not null, foreign key, on delete cascade
address_id	bigint unsigned, default null

A carts tábla tárolja egy adott felhasználó által kiválasztott és kosárba rakott, megvásárolni kívánt ételeket. Egy felhasználóhoz csak egy kosár tartozhat. A user_id mező

kapcsolja a kosarat a felhasználóhoz. A kosárban a megrendelés előtt még egy szállítási címet is ki kell választani. A kiválasztott szállítási címet az address_id kapcsolja a kosárhoz. Az address_id idegen kulcs null értéket is felvehet, így egy felhasználóhoz tartozó kosár létrejöhet anélkül is, hogy lenne a felhasználónak felvéve szállítási cím a fiókjához. Új szállítási címet a felhasználó a kosárból is felvehet megrendelés előtt.

2.3.7 Cart_items tábla

<i>Mező</i>	<i>TÍPUS, MEGKÖTÉS</i>
id	bigint, unsigned, not null, auto_increment, primary key
cart_id	bigint, unsigned, not null, foreign key, on delete cascade
name	varchar(255), not null
description	varchar(1000), default null
price	int, unsigned, not null
image_path	varchar(1000), default null
quantity	int, unsigned, not null

A cart_items tábla tárolja a kosárba helyezett ételeket. A kosárban lévő ételekhez tartozó adatok ebben a táblában újra tárolásra kerülnek. A rendszer nem használ idegen kulcsot a food_items táblához, mert az admin közben szerkesztheti az adott ételt az étlapon. A kosárba helyezés pillanatában érvényben lévő étel adatokkal kerül majd megrendelésre az adott étel. A cart_id köti a kosárba helyezett tételt a kosárhoz. A name, description, price, image_path mezők az étel adatait tárolják, sorban: az étel neve, leírása, ára, hozzátartozó kép. A quantity mező tárolja, hogy egy adott ételből hány darab kerüljön rendelésre.

2.3.8 Orders tábla

<i>Mező</i>	<i>Típus, Megkötés</i>
id	bigint, unsigned, not null, auto_increment, primary key
user_id	bigint, unsigned, default null, foreign key
order_address_id	bigint, unsigned, not null, foreign key
order_number	char(36), not null

Az orders tábla tárolja a rendelt ételeket. A user_id idegen kulcs köti a rendelést ahhoz a felhasználóhoz, aki az adott rendelést leadta. A user_id null értéket is felvehet. Ha a felhasználó törli a fiókot, a rendelései megmaradnak a rendszerben csak a user_id értéke null lesz. Idegen kulcsra mégis azért van szükség, mert így a felhasználó le tudja kérdezni a rendeléseit. Az order_address_id idegen kulcs köti a megrendelést a rendeléshez tartozó szállítási adatokhoz. A rendeléshez tartozó szállítási adatok újra letárolásra kerülnek az order_addresses táblában, nem az addresses táblát használja a rendszer, mert a felhasználó szerkesztheti, törölheti a szállítási címeit, vagy törölheti a felhasználói fiókot. A szállítási címek és a rendelések kapcsolatát itt az orders táblában lévő order_addresses idegen kulcs hozza létre nem pedig az order_addresses táblában lévő order_id idegen kulcs, mert általában a felhasználók a legtöbb rendelést ugyanazzal a szállítási címmel adják le. így egy szállítási címhez több megrendelés tartozhat. Az order_number a rendelésszám, egy egyedi uuid-s azonosító.

2.3.9 Order_addresses tábla

MEZŐ	TÍPUS, MEGKÖTÉS
id	bigint, unsigned, not null, auto_increment, primary key
last_name	varchar(255), not null
first_name	varchar(255), not null
phone	varchar(255), not null
zipcode	varchar(255), not null
city	varchar(255), not null
street	varchar(255), not null
note	longtext

Az order_addresses tábla tárolja a rendelésekhez tartozó szállítási adatokat. A mezők:
last_name: vezetéknév, first_name: keresztnév, phone: telefonszám, zipcode: irányítószám,
city: város, street: utca, note: szállítással kapcsolatos megjegyzések.

2.3.10 Order_items tábla

MEZŐ	TÍPUS, MEGKÖTÉS
id	bigint, unsigned, not null, auto_increment, primary key
order_id	bigint, unsigned, not null, foreign key, on delete cascade
name	varchar(255), not null
description	varchar(1000), default null
price	int, unsigned, not null
image_path	varchar(1000), default null
quantity	int, unsigned, not null

Az order_items tábla tárolja a rendelésekhez tartozó ételeket. Az ételekhez tartozó adatok ebben a táblában újra tárolásra kerülnek, így a rendelés pillanatában érvényes étel adatok tartoznak majd a rendeléshez. Az order_id kapcsolja a rendelt ételt a rendeléshez. A

name, description, price, image_path a rendelt ételhez tartozó adatokat tartalmazza, sorban: név, leírás, ár, kép elérési útvonala. A quantity a rendelt étel mennyiségét tárolja.

2.3.11 Order_statuses tábla

Mező	TÍPUS, MEGKÖTÉS
id	bigint, unsigned, not null, auto_increment, primary key
current_employee_id	bigint, unsigned, not null, foreign key
target_employee_id	bigint, unsigned, not null, foreign key
status	enum('received', 'under_delivery', 'delivered'), not null

Az order_statuses tábla tárolja egy adott rendelés állapotát és a rendelés feldolgozásakor a vele foglalkozó munkavállalókat. Az status mező felvehető értékei:

- received: a felhasználó elküldte a rendelést, a munkavállalók látják a rendszerben
- under_delivery: az operátor az elkészült ételeket összegyűjtötte és átadta a futárnak
- delivered: a futár kiszállította a rendelést

A current_employee_id kapcsolja a rendelési státuszt ahhoz a munkavállalóhoz, aki foglalkozott a rendeléssel. A target_employee_id kapcsolja a rendelés státuszt ahhoz a munkavállalóhoz, akinek tovább küldték a rendelést. Pld.: Az operátor elküldte egy futárnak a rendelést kiszállításra, akkor a current_employee_id az operátor employee id-a, a target_employee_id a futár employee id-a lesz.

2.3.12 Employee tábla

<i>Mező</i>	<i>Típus, Megkötés</i>
id	bigint, unsigned, not null, auto_increment, primary key
user_id	bigint, unsigned, not null, foreign key
position	enum('operator', 'cook', 'courier'), not null

Az employee tábla tárolja a munkavállalók adatait. A user_id kapcsolja a munkavállalót a felhasználói fiókhoz. A position a munkavállaló beosztását tartalmazza:

- operator: operátor
- cook: szakács
- courier: futár

2.4 PROGRAM FELÉPÍTÉSE

2.4.1 Felhasználói azonosítás

A Laravel Brezee telepítésével alapvető felhasználói azonosítással kapcsolatos funkciókat kapunk. Ezek közé tartozik a felhasználói regisztráció, azonosítás és a belépési próbálkozások korlátozása. Többszöri helytelen felhasználói azonosítási adatok megadása után ip cím alapján letiltja a belépés lehetőségét egy percre.

Létrehoz egy users táblát az adatbázisban a felhasználó adatok tárolására. A jelszót hashelt formában menti.

A user táblához hozzáadtam egy role mezőt, ami alapján három féle felhasználói fiók létezhet az applikációban.

- Vásárlói: Regisztrálhatnak az oldalon, megtekinthetik az étlapot és rendelést adhatnak le. Vásárlói felhasználói fiókot regisztrációval lehet létre hozni.
- Munkavállalói: Beosztásuk szerint megjelenő felületen végeznek a beérkezett rendelések feldolgozásával kapcsolatos műveleteket. A munkavállalói fiókot csak admin jogokkal rendelkező felhasználó hozhat létre. Bejelentkezésükkor a beosztásuk szerinti felület jelenik meg.
- Admin: Szerkeszti az étlapot és a munkavállalói fiókokat. Egy darab admin felhasználó szerepel az adatbázisban. Admin felhasználói fiókot csak admin jogokkal rendelkező felhasználó hozhat létre.

2.4.2 Laravel keretrendszer

A Laravel keretrendszerben az adatbázissal kapcsolatos CRUD műveletek a beépített Eloquent Model segítségével végezhetőek el, ami egy ORM (Object Relational Mapper) eszköz.

Adatbázis táblák migration fájlokkal, azon belül Schema osztállyal hozhatók létre.

A URL mapping létrehozásához a Route osztály használható.

A Laravel keretrendszer „route model binding” szolgáltatásával a route paraméterben átadott modell azonosító megadásával a hozzá tartozó controller metódusának paraméterében megkapjuk magát a modellt, így nem kell az adatbázisban megkeresni az azonosító számhoz tartozó bejegyzést.

2.4.3 Példa az adatbázis táblák létrehozására

A categories tábla létrehozása:

```
Schema::create('categories', function (Blueprint $table) {  
    $table->id();  
    $table->string('name')->unique();  
    $table->unsignedInteger('place');  
    $table->timestamps();  
});
```

A food_items tábla létrehozása:

```
Schema::create('food_items', function (Blueprint $table) {  
    $table->id();  
    $table->foreignId('category_id')->constrained()->cascadeOnDelete();  
    $table->string('name')->unique();  
    $table->longText('description')->nullable();  
    $table->unsignedInteger('price');  
    $table->string('image_path')->nullable();  
    $table->unsignedInteger('place');  
    $table->timestamps();  
});
```

2.4.4 Példa az Eloquent model-re

A Category model:

```
class Category extends Model
{
    use HasFactory;

    protected $fillable = [
        'name',
        'place',
    ];

    public function foodItems()
    {
        return $this->hasMany(FoodItem::class);
    }
}
```

A foodItems metódus definiálja a kapcsolatot a categories és a food_items tábla között.

A FoodItem model:

```
class FoodItem extends Model
{
    use HasFactory;

    protected $fillable = [
        'category_id',
        'name',
        'description',
        'price',
        'image_path',
        'place',
    ];

    public function category()
    {
        return $this->belongsTo(Category::class);
    }

    protected static function booted()
    {
        static::deleted(function (FoodItem $foodItem) {
            if ($foodItem->image_path) {
                Storage::disk('s3')->delete($foodItem->image_path);
            }
        });
    }
}
```

A category metódot definiálja a kapcsolatot a food_items és a category tábla között. A deleted metóduval regisztrál egy függvényt a FoodItem model deleted eseményére. FoodItem deleted esemény akkor jön létre, ha a food_items táblából törlésre kerül egy bejegyzés. Ha ez megtörténik, akkor itt a törölt FoodItem-hez tartozó képet törli a tárhelyről.

2.4.5 Példa az Eloquent model CRUD műveleteire:

A Category és FoodItem model az étlap megjelenítések és szerkesztések során játszik szerepet. Az étlap megjelenítéséhez a server oldalon az adatbázisból lekérdezésre kerül az összes ételcsoport és az egyes ételcsoportokhoz tartozó összes étel, place mező szerint növekvő sorrendben.

```
public function index()
{
    $categories = Category::with(
        [
            'foodItems' => function ($query) {
                $query->orderBy('place');
            }
        ]
    )->orderBy('place')->get(['id', 'name', 'place']);

    return Inertia::render('FoodMenu/Index', ['categories' => $categories]);
}
```

Az adatok kliens oldalon react komponensben categories paraméterként érkeznek meg, amit array mapping felhasználásával jelenít meg egy listában.

```
export default function FoodMenu({ categories }) {
    return (
        <ul>
            {categories.map(category => (
                <li>...</li>
            ))}
        </ul>
    );
}
```

Új kategória felvétele az étlapra: A kliens oldalon megjelenítésre kerül egy form, ahol meg kell adni az új ételcsoport nevét.

```
export default function Create() {
  const { data, setData, errors, post } = useForm({
    name: '',
  });

  const submit = e => {
    e.preventDefault();
    post(route('admin.edit-food-menu.categories.store'));
  };

  return (
    // ...
    <form onSubmit={submit}>
      <input
        type="text"
        name="name"
        id="name"
        value={data.name}
        onChange={e => setData('name', e.target.value)}
      />

      {errors.name && <p>{errors.name}</p>}
    </form>

    // ...
  );
}
```

A form adatainak kezelését, elküldését a szerver oldalra és az esetleges szerver oldali validálási hiba miatt vissza érkező hibaüzenet megjelenítését az Inertia useForm beépített függvény segítségével végzi el:

- A form egyes mező adatait a data objektumban tárolja, beállítását a setData függvénnyel végzi. (react useState kétirányú adatkapcsolat).
- A form elküldésekor a post függvény hívásával a megadott címre elküldi az adatokat
- Szerver oldalon a requestben megérkezik az adat, ahol az előírt feltételekkel elvégzi a név adatainak ellenőrzését. Ha nem megfelelő formátumban lett megadva, vagy a

név mező már létezik a categories táblában, akkor a request->validate függvény visszaküldi a hibaüzenetet a kliens oldalra.

- A useForm errors objektumba érkezik a hibaüzenet, ami a form hibásan megadott mezője alatt megjelenik.

```
public function store(Request $request)
{
    $request->validate([
        'name' => ['required', 'string', 'max:255', 'unique:categories'],
    ]);

    $maxPlace = Category::max('place');
    if (!$maxPlace) {
        $request['place'] = 1;
    } else {

        $request['place'] = $maxPlace + 1;
    }

    Category::create([
        'name' => $request['name'],
        'place' => $request['place'],
    ]);

    return to_route('admin.edit-food-menu.index');
}
```

- Szerver oldalon, ha az adatok megfelelőek, akkor lekéri a categories táblából a maxPlace változóba a legnagyobb értéket, ami a place mezőben szerepel. Ha a categories tábla üres, akkor a maxPlace értéke 1 lesz. Egyébként a maxPlace értéke növelve 1-gyel. Majd létrehozza az új kategóriát a beérkező névvel és a maxPlace értékkel.

Meglévő kategória szerkesztése: Szerver oldalról elküldi a szerkeszteni kívánt kategória adatait a kliens oldalra.

```
public function edit(Category $category)
{
    return Inertia::render('Admin/EditFoodMenu/Categories/Edit', ['category' => $category]);
}
```

Kliens oldalon ugyan olyan form jelenik meg, mint az új kategória hozzáadásánál, csak az egyes mezők értékei az elküldött category adatainak értékeit veszik fel.

```
export default function Create(category) {
  const { data, setData, errors, post } = useForm({
    name: category.name,
  });

  // ...
}
```

Szerver oldalon ellenőrzi a bevitt adatok megfelelő formátumát, és hogy a megadott név még nem szerepel-e az adatbázisban, kivéve, ha a megadott név éppen a szerkesztett kategória neve. Ezután a category modellben frissíti az adatot, majd elmenti az adatbázisba.

```
public function update(Request $request, Category $category)
{
    $request->validate([
        'name' => ['required', 'string', 'max:255',
Rule::unique('categories')->ignore($category->id)],
    ]);

    $category->name = $request->name;
    $category->save();
    return to_route('admin.edit-food-menu.index');
}
```

Kategória törlése: A kliens oldalról elküldésre kerül a törölni kívánt kategória.

```
router.delete(route('admin.edit-food-menu.categories.destroy', categoryId));
```

A szerver oldalon az adott kategóriához tartozó ételeket egyesével törli egy foreach ciklussal, majd törli a kategóriát is az adatbázisból.

```
public function destroy(Category $category)
{
    foreach ($category->foodItems as $foodItem) {
        $foodItem->delete();
    }
    $category->delete();
}
```

Azért kell egyesével törölni az adott kategóriákhoz tartozó ételeket, mert az ételekhez tartoznak képek, amik az adott étel törlésekor a FoodItem modell deleted esemény kezelőjében kerülnek törlésre. Ha csak a kategóriát törölné az adatbázisból, akkor a food_items táblában a category_id on delete cascade megkötés miatt ugyan törölné a kategóriához tartozó összes ételt is, viszont így nem hívná meg a FoodItem model deleted eseményét és nem törölné a hozzá tartozó képeket a tárhelyről.

```
protected static function booted()
{
    static::deleted(function (FoodItem $foodItem) {
        if ($foodItem->image_path) {
            Storage::disk('s3')->delete($foodItem->image_path);
        }
    });
}
```


2.4.6 Algoritmusok

2.4.6.1 Az ételcsoportok és az ételek sorrendjének beállítása az étlapon

Az admin felület étlap szerkesztése oldal látogatásakor a szerver elküldi az összes ételcsoportot (categories) az egyes ételcsoportokhoz tartozó ételekkel, place mező szerint rendezve növekvő sorrendben.

```
public function index()
{
    $categories = Category::with(
        [
            'foodItems' => function ($query) {
                $query->orderBy('place');
            }
        ]
    )->orderBy('place')->get(['id', 'name', 'place']);

    return Inertia::render('Admin/EditFoodMenu/Index', ['categories' => $categories]);
}
```

Az oldalon „drag and drop” műveletekkel lehet az egyes ételcsoportok vagy az ételcsoporton belül az ételek megjelenési sorrendjét beállítani az étlapon. A HTML elemek, amiket meg lehet „fogni” egérrel, megadásra kerül a draggable attribútuma, és az onDragStart, onDragOver, onDrop eseménykezelő.

```
// ...

<div
  draggable
  onDragStart={e => handleCategoryDragStart(e, category.id)}
  onDragOver={handleCategoryDragOver}
  onDrop={e => handleCategoryDrop(e, category.place)}
>

  { /* ... */ }

</div>
```

- onDragStart esemény akkor keletkezik, ha a felhasználó „megfogja” az adott elemet. Ekkor az adott ételcsoport vagy étel azonosítóját hozzárendeli egy kulcshoz egy belső dataTransfer objektumban.
- Az onDragOver esemény akkor keletkezik, ha a „megfogott” elemet áthúzza a felhasználó egy másik elem fölé. Itt ellenőrzi, hogy a dataTransfer objektum kulcsa megegyezik-e egy adott értékkel, azért, hogy ételcsoportot csak ételcsoportra, ételt pedig csak ételre lehessen „dobni”.
- onDrop esemény akkor keletkezik, ha a „húzott” elemet „elengedi” a felhasználó. Itt a program kinyeri a dataTransfer objektumból a „húzott” ételcsoport vagy étel azonosítóját. A place annak az elemnek az értéke, amire „rádobták” a „húzott” elemet

```
const handleCategoryDragStart = (e, categoryId) => {
  e.dataTransfer.setData('application/category', categoryId);
};

const handleCategoryDragOver = e => {
  const isAllowed = e.dataTransfer.types.includes('application/category');
  if (isAllowed) {
    e.preventDefault();
  }
};

const handleCategoryDrop = (e, place) => {
  e.preventDefault();

  const categoryId = e.dataTransfer.getData('application/category');
  router.post(
    route('admin.edit-food-menu.categories.place', categoryId),
    {
      place,
    },
    {
      preserveScroll: true,
    }
  );
};
```

Ezután szerver oldalra elküldi az azonosítót és a place értékét. Az azonosító jelenti, hogy melyik elemet, a place, hogy hányadik helyre szeretné áthelyezni az elemet.

```

public function place(Request $request, Category $category)
{
    $targetCategory = Category::where('place', $request['place'])->first();

    if (!$targetCategory || $category->place === $request['place']) {
        return to_route('admin.edit-food-menu.index');
    }

    $targetCategoryPlace = $targetCategory->place;
    $sourceCategoryPlace = $category->place;

    if ($targetCategoryPlace < $sourceCategoryPlace) {
        $restCategories = Category::where('place', '>=', $targetCategoryPlace)->where('id', '<>',
$category->id)->orderBy('place')->get();

        $category->place = $targetCategoryPlace;
        $category->save();

        $i = $targetCategoryPlace + 1;
        foreach ($restCategories as $restCategory) {
            $restCategory->place = $i;
            $restCategory->save();
            $i++;
        }
    } else {
        $restCategories = Category::where('place', '<=', $targetCategoryPlace)->where('id', '<>',
$category->id)->orderBy('place')->get();

        $category->place = $targetCategoryPlace;
        $category->save();

        $i = 1;
        foreach ($restCategories as $restCategory) {
            $restCategory->place = $i;
            $restCategory->save();
            $i++;
        }
    }
    return to_route('admin.edit-food-menu.index');
}

```

A szerver oldalon a place metódus \$category paraméterben megjelenik az azonosítóhoz tartozó kategória, a requestben pedig a place. Ezután a program megkeresi azt a kategóriát, amelyik helyére kívánja tenni a „húzott” kategóriát. Ha olyan nincs vagy saját

magára akarja rakni, akkor nem történik semmi, visszairányít az oldalra. A `targetCategoryPlace` annak a kategóriának a sorszáma az étlapon, amire a felhasználó rá akarja rakni a kívánt kategóriát, a `sourceCategoryPlace` pedig annak a kategóriának a sorszáma, amit rá akar rakni. Ha egy olyan kategória elemet „húz” rá egy másik kategória elemre, amelynek a sorszáma az étlapon nagyobb, mint a másik kategória elem sorszáma, akkor a művelet után az „odahúzott” kategória a másik kategória helyére kerül, felveszi a `place` értéket, és az összes további kategória elem, ami alatta volt, lejjebb kerül eggyel a listában, a sorszámuk eggyel növekszik. Ha az „odahúzott” kategória elem sorszáma kisebb, akkor a művelet után felveszi a másik kategória sorszámának értékét és a listában a felette lévő összes kategória eggyel feljebb kerül, `place` értékük eggyel csökken.

3 FELHASZNÁLÓI DOKUMENTÁCIÓ

3.1 VÁSÁRLÓI FELÜLET

3.1.1 Regisztráció

A főmenüben lévő Regisztráció linkre kattintva megjelenik egy űrlap.



Az űrlap adatainak a kitöltésével hozható létre új felhasználói fiók. Hibás adatok megadásakor a kitöltési mező alatt jelenik meg a hibaüzenet. A név nem lehet hosszabb 255 karakternél, az email-t megfelelő formátumban kell megadni, hossza nem lehet több 255 karakternél. A jelszónak minimum 8 karakter hosszúságúnak kell lennie.

Sikeres regisztráció követően a rendszer be is lépteti a felhasználót az oldalra. A felhasználó neve megjelenik a főmenüben jobb oldalon.



3.1.2 Belépés

A főmenüben lévő Belépés linkre kattintva megjelenik egy űrlap.



Az űrlap adatainak helyes kitöltésével a felhasználót belépteti a rendszer. Hibás adatok megadásakor a megfelelő beviteli mező alatt megjelenik a hibaüzenet. Csak regisztrált email címmel lehet belépni. Az adatok kitöltése után a Belépés gombra kattintva a rendszer belépteti a felhasználót.

A dark blue login form with two input fields. The first field is labeled 'Email' and the second is labeled 'Jelszó'. Both fields are empty. At the bottom right of the form is a white button with the text 'BELÉPÉS' in blue capital letters.

A felhasználó neve megjelenik a főmenüben jobb oldalon.

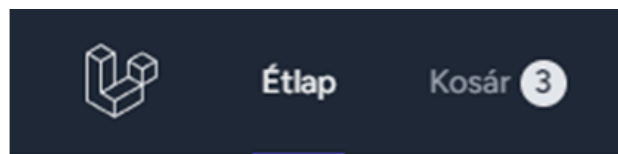
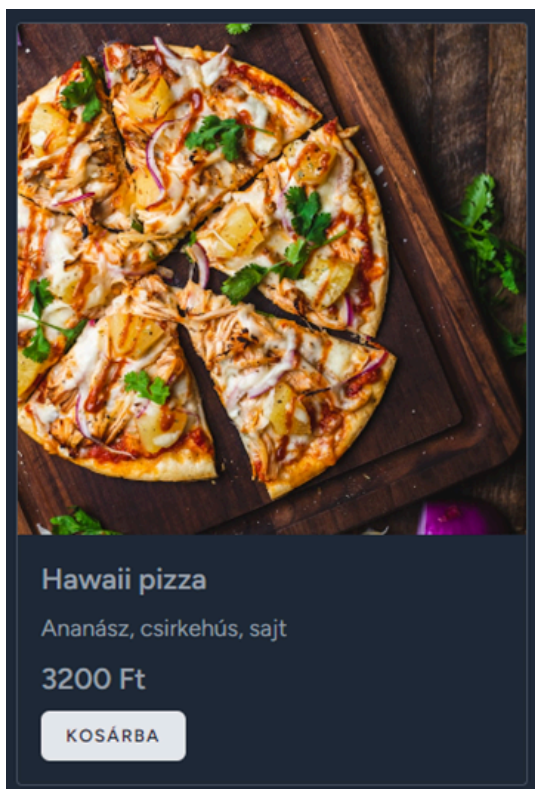


3.1.3 Étel rendelése





A főmenüben az Étlap linkre kattintva megjelenik az Étlap



Az étlapom a rendelni kívánt étel a kártya alján látható Kosárba gombbal helyezhető a kosárba. A kosárban lévő ételek mennyiségét a főmenüben a Kosár link mellett látható szám jelzi.



Az ételek kiválasztása után a rendelést a főmenüben lévő Kosár linkre kattintva lehet folytatni. Megjelenik a kosár tartalma, ahol látható a kiválasztott ételek listája.

Rendelt ételek				
	Hawaii pizza	Ananász, csirkehús, sajt	1 Db	3200 Ft 
	Fekete hamburger	Marhahús, hagyma, uborka, saláta, sajt, majonéz, szalonna	1 Db	3600 Ft 
				Összesen: 6800 Ft


Az egyes tételek rendelni kívánt mennyisége növelhető vagy csökkenthető, vagy az adott étel törölhető a kosárból a „kukába” gombra kattintva.

A rendelést a szállítási cím kiválasztásával lehet folytatni. Itt megjelennek a felhasználói fiókhoz tartozó szállítási címek. Ha nincs még mentve cím a fiókban, akkor az Új szállítási cím gombbal lehet felvenni.

Szállítási cím kiválasztása

+ ÚJ SZÁLLÍTÁSI CÍM

Név: Kiss Imre
Cím: 1212. Budapest, Halmaz u. 48/A.
Tel.: +36 223334455
Megjegyzés: Kapucsengő 6-os



Név: Kiss Edina
Cím: 1212. Budapest, Halmaz u. 48/A.
Tel.: +36 112223344
Megjegyzés: Kapucsengő 7-es

Név: Kiss Imre
Cím: 3232. Budapest, Munkahely u. 1-23.
Tel.: +36 223334455
Megjegyzés: Dolgozom kft., Humánerőforrás-osztály

A megfelelő cím kiválasztása után a Megrendelés gombra kattintva megjelenik egy összegző oldal, ahol ellenőrizhető a rendelni kívánt ételek listája és a szállítási cím. A rendelés

véglegesítése gombra kattintva küldhető el a megrendelés, a mégsem gombbal visszatérhetünk a kosárhoz.

Rendelés

Hawaii pizza	1 db	3200 Ft / db	3200 Ft
Fekete hamburger	1 db	3600 Ft / db	3600 Ft
Összesen			6800 Ft

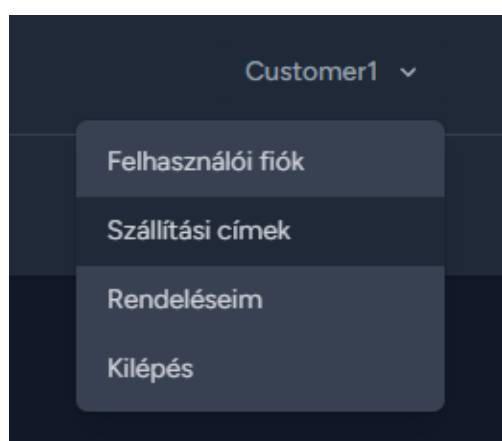
Szállítási cím

Kiss Imre
1212. Budapest
Halmaz u. 48/A.
Tel.: +36223334455
Szállítási megjegyzés: Kapucsengő 6-os


MÉGSEM RENDELÉS VÉGLEGESÍTÉSE

3.1.4 Szállítási címek

A főmenüben jobb oldalon a felhasználó nevére kattintva a megjelenő legördülő menüben kiválasztva a Szállítási címek menüpontot érhető el az oldal.





A szállítási címek oldalon egy listában látható az adott felhasználói fiókhoz tartozó összes szállítási cím.

ÉtlapKosár 2Customer1 ▾



Szállítási címek

+ ÚJ SZÁLLÍTÁSI CÍM



Név: Kiss Imre
Cím: 1212. Budapest, Halmaz u. 48/A.
Tel.: +36 223334455
Megjegyzés: Kapucsengő 6-os



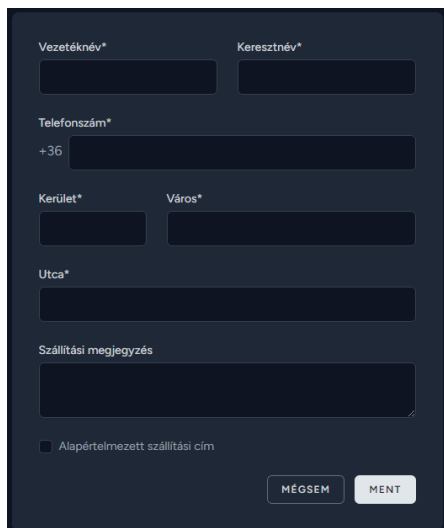
Név: Kiss Edina
Cím: 1212. Budapest, Halmaz u. 48/A.
Tel.: +36 112223344
Megjegyzés: Kapucsengő 7-es



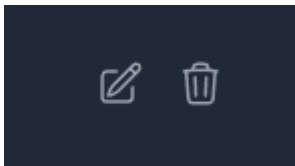
Név: Kiss Imre
Cím: 3232. Budapest, Munkahely u. 1-23.
Tel.: +36 223334455
Megjegyzés: Dolgozom kft., Humánerőforrás-osztály



Az új szállítási cím hozzáadás gombbal lehet új szállítási címet felvenni. A gombra kattintva megjelenik egy űrlap, ahol az adatokat kitöltve felvehető egy cím. Hibásan kitöltött mezők esetében a hibaüzenet az adott mező alatt látható. Az alapértelmezett cím jelölő mező kiválasztásával az adott szállítási cím lesz a legközelebbi rendelésnél alapértelmezett szállítási címnek a kosárban kijelölve. A mégsem gombbal visszatér a szállítási címek listához, a mentés gombbal felveszi az új szállítási címet.



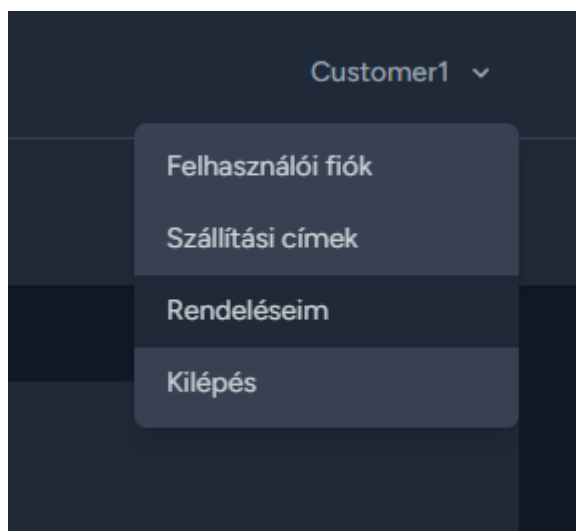
A listában megjelenő szállítási címek adatai szerkeszthetők a „ceruza” gombbal. A gombra kattintva egy ugyanolyan űrlap lesz látható, mint az új szállítási cím hozzáadásánál, kitöltve az adott szállítási cím adataival. Mégsem gombbal visszatér a szállítási címek listához, mentés gombbal menti a változásokat.



A „kukába” gombbal az adott szállítási cím törölhető.

3.1.5 Rendelések megtekintése

A főmenüben jobb oldalon a felhasználó nevére kattintva a megjelenő legördülő menüben kiválasztva a Rendeléseim menüpontot érhető el az oldal.



A megjelenő oldalon láthatóak egy listában a felhasználó elküldött rendelései.

3.2 MUNKAÁLLALÓI FELÜLET

3.2.1 Operátori munkakör

3.2.1.1 Beérkezett rendelések

A főmenüben a Beérkezett rendelések linkre kattintva megjelenik egy lista a vásárlók által leadott rendelésekről. A küldés kiszállítás gombra kattintva a kijelölt rendeléseket elküldheti kiszállításra a kiválasztott futárnak. A lista elemre kattintva lehet kijelölni adott lista elemet. A fejlécben lévő jelölő négyzettel kijelölhető az összes lista elem.

KÜLDÉS KISZÁLLÍTÁSRA		
<input type="checkbox"/> Rsz.	Cím	Ételek
<input checked="" type="checkbox"/> d6c8fbb6-c8b0-46df-871a-467f3e06a00b	Kiss, Imre 1212. Budapest, Halmaz u. 48/A. +36223334455	Hawaii pizza Fekete hamburger
<input type="checkbox"/> 0aa4f3f8-ffde-42f9-9d37-002f9b2e376c	Kiss, Edina 1212. Budapest, Halmaz u. 48/A. +36112223344	Dupla hamburger

A küldés kiszállítás gombra kattintva megjelenik egy felugró ablak, ahol egy legördülő listából kiválasztható a futár. A mégsem gombra kattintva visszatér a beérkezett rendelések listába, a Küld gombra kattintva a megrendelést elküldi a kiválasztott futárnak.

Futár

Employee3

Employee2

Employee3

Employee4

MÉGSEM

KÜLD

3.2.1.2 Szállítás alatt lévő rendelések


A főmenüben a Szállítás alatt lévő rendelések linkre kattintva megjelenik egy lista a szállítás alatt lévő rendelésekről. Látható a rendelés és a futár neve.

Rsz.	Cím	Ételek	Futár
d6c8fbb6-c8b0-46df-871a-467f3e06a00b	Kiss, Imre 1212. Budapest, Halmaz u. 48/A. +36223334455	Hawaii pizza Fekete hamburger	Employee3

3.2.2 Futár munkakör

3.2.2.1 Kiszállítandó rendelések

A főmenüben a Kiszállítandó rendelésekre kattintva megjelenik egy lista a felületre bejelentkezett futárnak kiszállításra elküldött rendelésekről.

Rsz.	Cím	Ételek	Ár	
9f7c2319-e9af-435e-b0ba-56d51eae126c	Kiss, Edina 1212. Budapest, Halmaz u. 48/A. +36112223344	Bazsalikomos pizza Rukkolás pizza	9600 Ft	

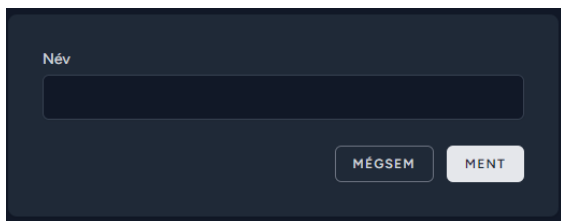
A listaelem jobb oldalán lévő kiszállítva ikonra kattintva a rendelés státusza a rendszerben felveszi a “kiszállítva” állapotot és eltűnik a kiszállítandó listából.

3.3 ADMIN FELÜLET

3.3.1 Étlap szerkesztése

3.3.1.1 Új kategória felvétele

A főmenüben az Étlap szerkesztése linkre kattintva megjelenik egy lista az étlapról. Az Új kategória gombra kattintva vehető fel új ételcsoport az étlapra. A megjelenő űrlapon meg kell adni az új ételcsoport nevét. A mégsem gombbal visszatér az étlap szerkesztése listába, a ment gombbal felveszi az új ételcsoportot a listára.



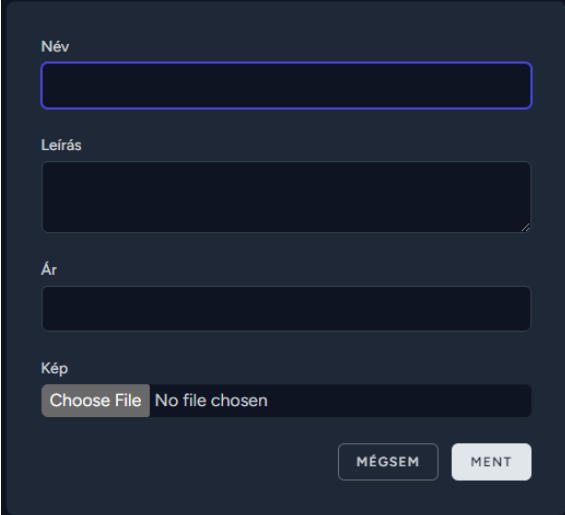
A dark-themed form for adding a new dish. It features a label 'Név' (Name) above a text input field. Below the input field are two buttons: 'MÉGSEM' (Cancel) and 'MENT' (Save).

3.3.1.2 Új étel felvétele egy adott ételcsoportba

Az ételcsoport fejlécében jobb oldalt található + gombra kattintva vehető fel új étel az ételcsoportba



A megjelenő űrlapon ki kell tölteni az új étel adatait, illetve képet lehet hozzá rendelni. A leírás és a Kép mezők kitöltése nem kötelező. A mégsem gombra kattintva visszatért az étlap szerkesztése listába, a ment gombbal felveszi az étlapra az új ételcsoportot.



The screenshot shows a dark-themed form for creating a new food item. It contains four input fields: 'Név' (Name), 'Leírás' (Description), 'Ár' (Price), and 'Kép' (Image). The 'Kép' field has a 'Choose File' button and the text 'No file chosen'. At the bottom right, there are two buttons: 'MÉGSEM' (Cancel) and 'MENT' (Save).

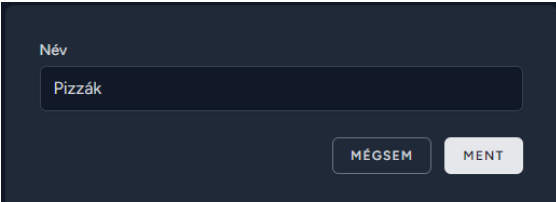
3.3.1.3 Ételcsoport szerkesztése

Az ételcsoport fejlécében jobb oldalt található „ceruza” gombra kattintva szerkeszthető az ételcsoport.



The screenshot shows a dark-themed header for a food category. On the left, there is a grid icon followed by the text 'Pizzák'. On the right, there are three icons: a plus sign, a pencil (edit), and a trash can.

A megjelenő űrlapon lehet szerkeszteni ételcsoport nevét. A mégsem gombbal visszatér az étlap szerkesztése listába, a ment gombbal felveszi az új ételcsoportot a listára.



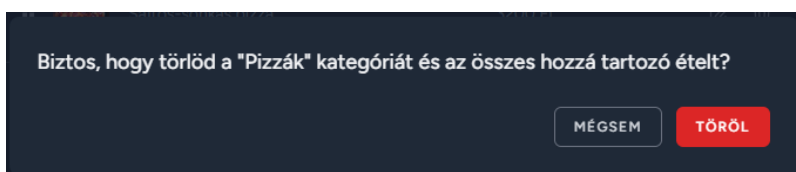
The screenshot shows a dark-themed form for editing a food category. It contains one input field labeled 'Név' with the text 'Pizzák' inside. At the bottom right, there are two buttons: 'MÉGSEM' (Cancel) and 'MENT' (Save).

3.3.1.4 Ételcsoport törlése

Az ételcsoport fejlécében jobb oldalt található „kukába” gombra kattintva törölhető az ételcsoport.

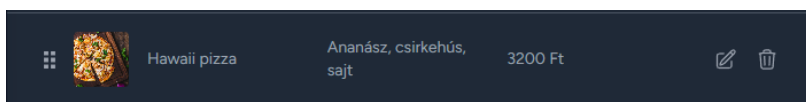


Az ételcsoport törlésével az összes hozzátartozó étel is törlésre kerül. A megjelenő felugró ablakban meg kell erősíteni a törlési szándékot. A mégsem gombbal visszatér az étlap szerkesztése listába, a töröl gombbal törli az ételcsoportot a listából.



3.3.1.5 Étel szerkesztése

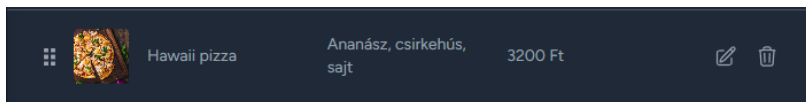
Az étel lista elemében jobb oldalt található „ceruza” gombra kattintva szerkeszthető az adott étel.



A megjelenő űrlapon lehet szerkeszteni az étel adatait. Az aktuális kép részénél látható az ételhez hozzárendelt aktuális kép. Új kép feltöltésekor az aktuális kép törlődik és az új kép lesz hozzárendelve az ételhez, a kép feltöltése mezőt üresen hagyva marad az aktuális kép. A mégsem gombbal visszatér az étlap szerkesztése listába változtatások nélkül, a ment gombbal menti az adatokat.

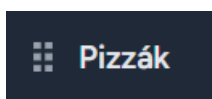
3.3.1.6 Étel törlése

Az étel lista elemében jobb oldalt található „kukába” gombra kattintva törölhető az adott étel.



3.3.2 Ételcsoport vagy étel megjelenési sorrendjének változtatása az étlapon

Adott ételcsoport vagy étel neve mellett bal oldalon látható 6 darab kis négyzet ikon segítségével, „drag and drop” műveleteket lehet a sorrendet megváltoztatni.



Ételcsoportot csak ételcsoportra, ételt csak adott ételcsoporton belüli ételre lehet „húzni”.

3.3.3 Munkavállalói fiókok

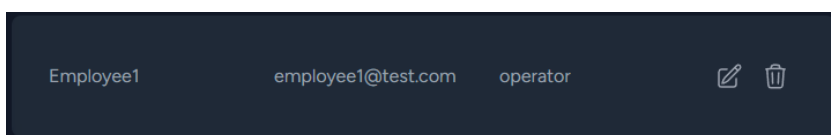
A főmenüben a Munkavállalói fiókok linkre kattintva megjelenik egy lista az admin által felvett munkavállalókról.

3.3.3.1 Új munkavállaló felvétele a listára

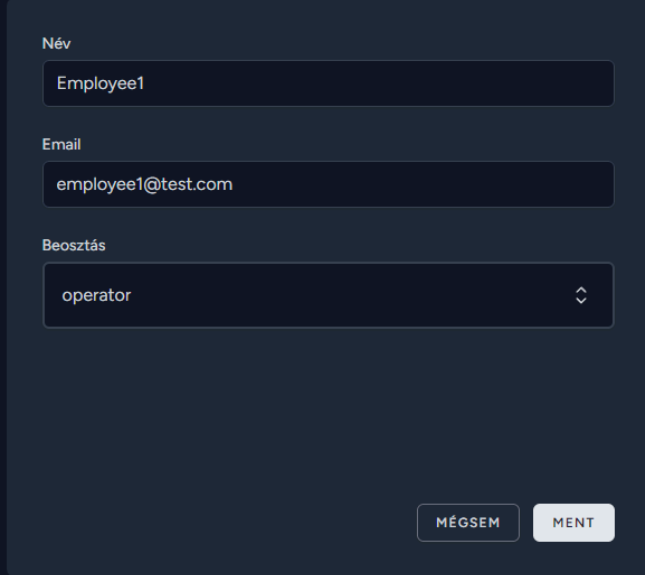
Az Új munkavállaló gombra kattintva megjelenik egy űrlap, ahol ki kell tölteni a munkavállalói fiók adatait. A mégsem gombra kattintva visszatér a munkavállalói fiókok listához, a ment gombra kattintva felveszi az új munkavállalói fiókot a rendszerbe.

3.3.3.2 Munkavállaló adatainak szerkesztése

Adott munkavállaló listaelem jobb oldalán található „ceruza” gombra kattintva lehet a munkavállaló adatait szerkeszteni.



Megjelenik egy űrlap, ahol szerkeszthető a munkavállaló fiók adatai. A mégsem gombra kattintva változtatás nélkül visszatér a munkavállalói fiókok listában. Mentés gombra kattintva a változtatások mentésre kerülnek.



A screenshot of a dark-themed form for editing employee data. The form has three input fields: 'Név' (Name) with the value 'Employee1', 'Email' with the value 'employee1@test.com', and 'Beosztás' (Position) with the value 'operator'. At the bottom right, there are two buttons: 'MÉGSEM' (Cancel) and 'MENT' (Save).

3.3.3.3 Munkavállalói fiók törlése

Adott munkavállaló listaelem jobb oldalán található „kukába” gombra kattintva lehet a munkavállalói fiókot törölni.

