



Széchenyi István Katolikus Technikum és Gimnázium

Szoftverfejlesztő és -tesztelő projektfeladat

PÉKSÉG

Készítették:

Vakherda Ádám,
Bolyky Máté,
Iróczki Regina

Ózd, 2025

Tartalomjegyzék:

Bevezetés

Felhasználói dokumentáció

Rendszerekvetelmény Hardver

Szoftver

Webalkalmazás indítása

Webalkalmazás használata

Általános információk a webalkalmazásról

Főoldal leírása + képernyőképek

Aloldal1 leírása + képernyőkép

Aloldal2...

Fejlesztői dokumentáció

Alkalmazott fejlesztői és csoportmunka eszközök

Adatbázis

Frontend

Backend

Adatbázis

Modell leírása (adatmezők – típusok - leírásuk)

Táblák, kapcsolatok

E-K diagram

Frontend

Alkalmazás frontend részének részletes leírása + képernyőfotók

Backend

Alkalmazás backend részének részletes leírása + képernyőfotók

Tesztelés

Továbbfejlesztési lehetőségek

Irodalomjegyzék

PROJEKT BEVEZETŐ DOKUMENTÁCIÓ

A projektünk célja egy olyan weboldal létrehozása, amely megkönnyíti az általános iskolák számára a péksütemények, uzsonnák, reggelik...stb rendelését egy helyi pékségből. Az oldal egy egyszerű és felhasználóbarát felületet biztosít az intézmények számára, ahol előre leadhatják rendeléseiket, megadhatják a kiszállítás időpontját, a fizetés módját és nyomon követhetik a rendelési folyamatot.

A weboldal nemcsak az iskolák számára biztosít kényelmes megoldást az ételek beszerzésére, hanem a pékség számára is hatékonyabb rendeléskezelést és kiszállítást tesz lehetővé.

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

ADATBÁZIS DOKUMENTÁCIÓ

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

Táblák bemutatása:

Pékáru

Mezők	Kulcs fajtája	adattípus	rövid leírás
Pid	PRIMARY KEY	int	A pékáruink elsődleges azonosítója
nev		varchar(20)	a pékáruk neve
tipus		varchar(20)	a pékáruk fajtáját/típusát tárolja el
ar		int	A pékáruk ára

2. Felhasználók

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

Cím

Mezők	Kulcs fajtája	adattípus	rövid leírás
-------	---------------	-----------	--------------

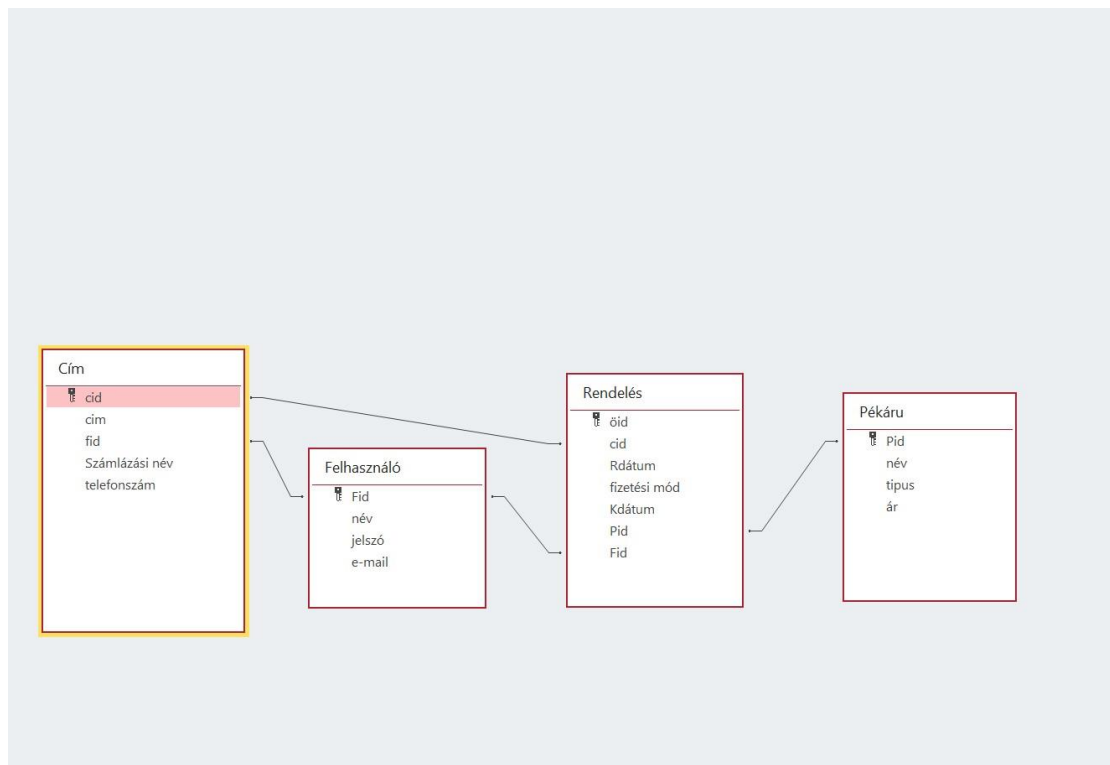
Cid	PRIMARY KEY	int	a cím tábla elsődleges azonosítója
cim		varchar(30)	a felhasználók teljes címét tárolja el
Fid	FOREIGN KEY		A felhasználók tábla mezőit meghívó idegenkulcs
SzamlazasiNev		varchar(30)	A felhasználók számlázási nevét tünteti fel
telefonszam		varchar(15)	A felhasználók telefonszámait tárolja el

Rendelés

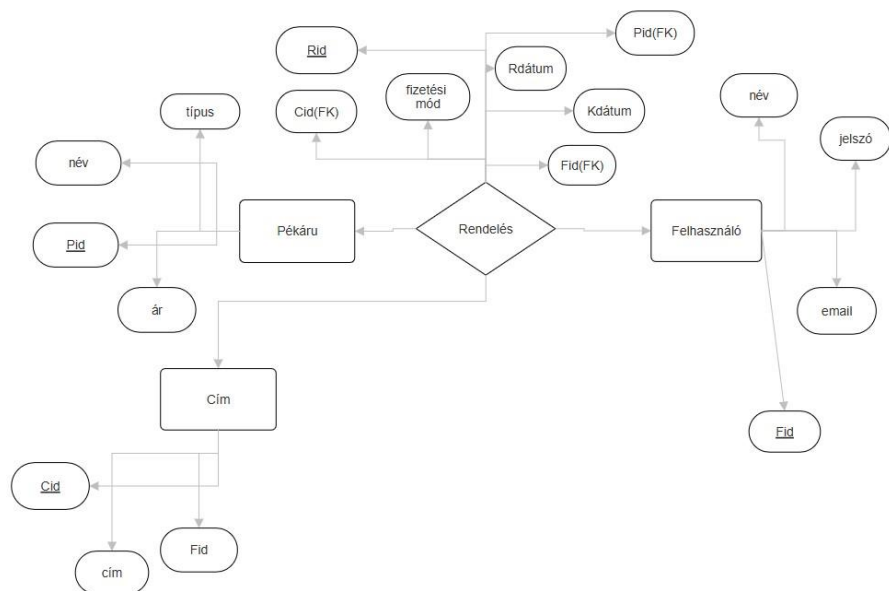
Mezők	Kulcs fajtája	adattípus	rövid leírás
Rid	PRIMARY KEY	int	a rendelés azonosítója
Pid	FOREIGN KEY	int	a pékáru tábla mezőit meghívó idegenkulcs
Fid	FOREIGN KEY	int	A felhasználók tábla mezőit meghívó idegenkulcs

Cid	FOREIGN KEY	int	a címek tábla mezőit meghívó idegenkulcs
RDatum		varchar(20)	a rendelés dátumának ideje
KDatum		varchar(20)	a kiszállítás dátumának várható időpontja
FizetesMod		boolean	a fizetési mód eldöntésére szolgál

KAPCSOLATÁBRA:



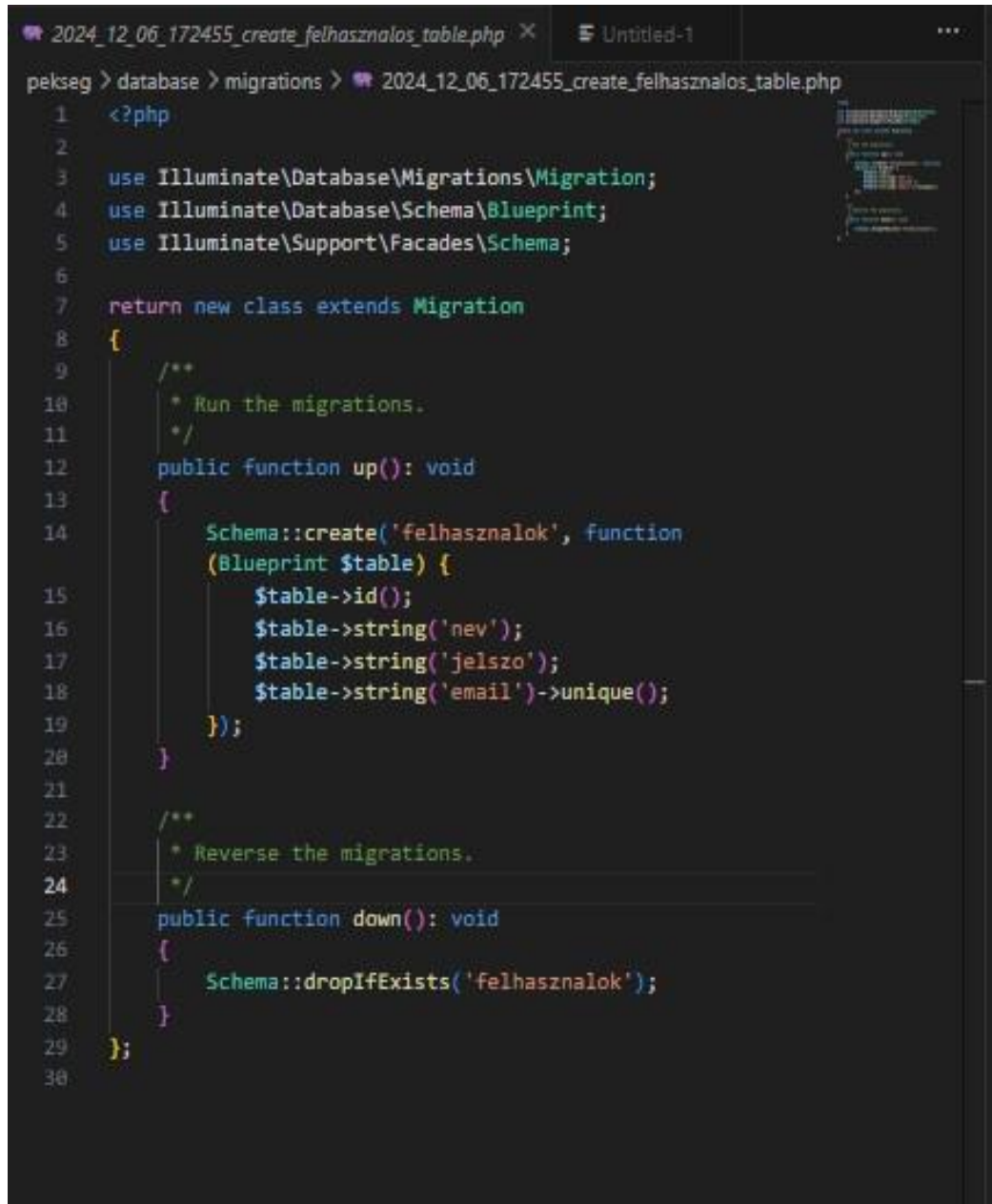
ER/EK DIAGRAMM:



BACKEND DOKUMENTÁCIÓ

Ez a dokumentáció a pékségünkhöz tartozó API működését mutatja be, amely többek között lehetőséget biztosít új termékek hozzáadására, meglévő adatok lekérdezésére, frissítésére és törlésére is. Az API REST alapú, és JSON formátumban kommunikál.

1.)



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

1. kép


```

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

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

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

```

2. kép

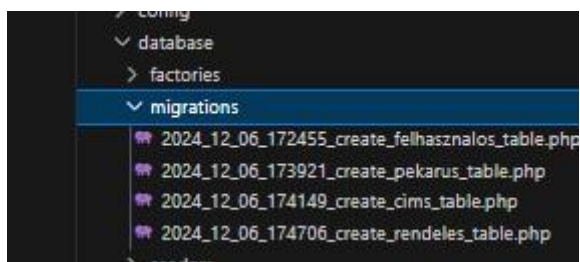
```
2024_12_06_173921_create_pekarus_table.php ×
pekseg > database > migrations > 2024_12_06_173921_create_pekarus_table.php

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

3. kép

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

4. kép



5.

kép

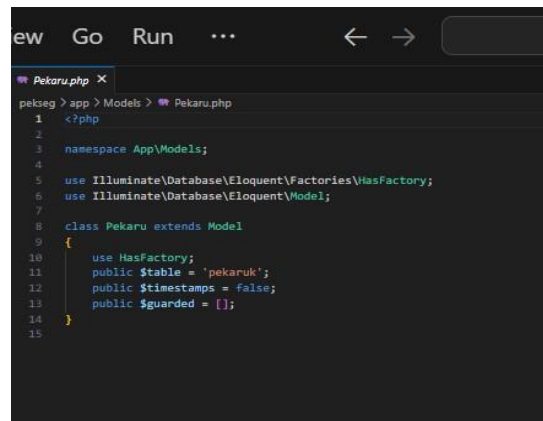
Ezen a képen a Laravel migrációs fájlokat láthatjuk.

Minden fájl egy-egy táblát hoz létre az adatbázisban pl.: Pékaruk, Felhasználók stb... A táblák közötti kapcsolatok is definiálva vannak, például a rendelések táblában a pékáru, felhasználó és cím külső kulcsként szerepel. A migrációk futtatásával (php artisan migrate - seed) az adatbázis automatikusan létrejön a megadott szerkezettel.

2.)

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

6. kép



```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Pekaruk extends Model
9 {
10     use HasFactory;
11     public $table = 'pekaruk';
12     public $timestamps = false;
13     public $guarded = [];
14 }
15
```

3.)

```

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

```

7. kép

Itt a Laravel útvonalakat/útvonaldefiníciókat láthatjuk.

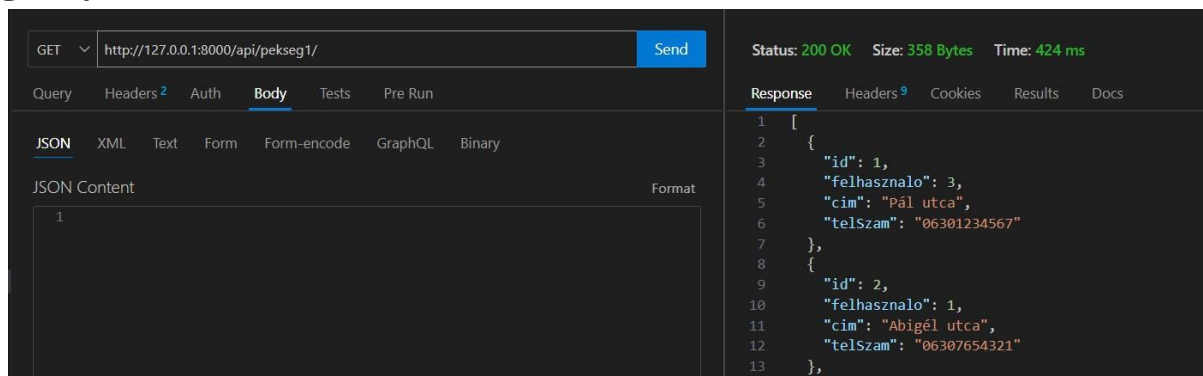
Ezek az API végpontjait határozzák meg a különböző vezérlők számára.

A **Route::get**, **Route::post**, **Route::put** és **Route::delete** metódusok különböző műveleteket végeznek, például adatokat kérnek le, hoznak létre, módosítanak vagy törölnek.

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

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

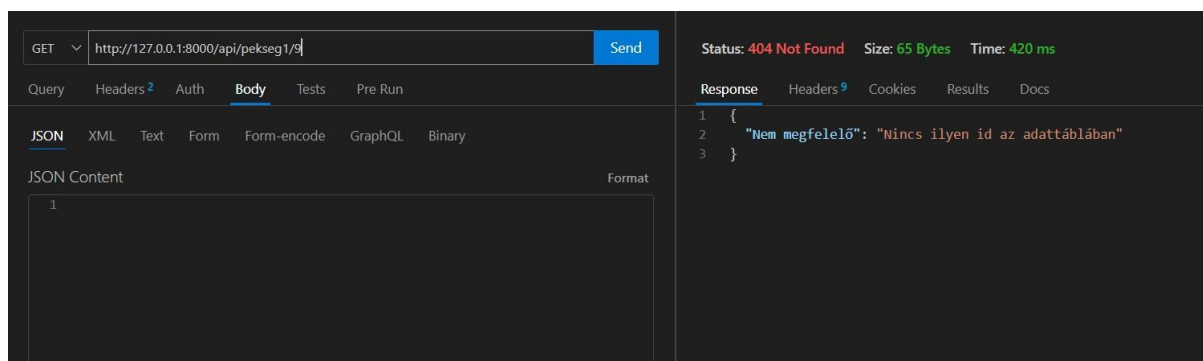
getByid:



8. kép

Reprezentáció a 4 getByid metódus egyikéről.

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



9. kép

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

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

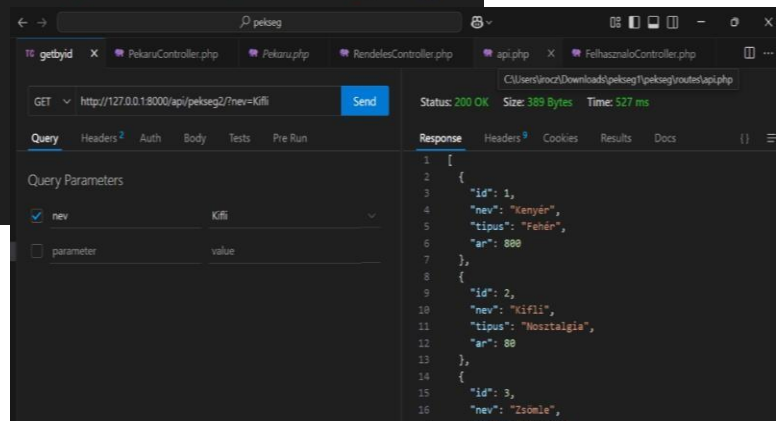
```

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

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

```

10. kép



11. kép

Update/Destroy:

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

(A többi controllerben is megtalálható az a két metódus)

Update:

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

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

12. kép

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

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

Ha minden adat megfelelő, frissíti a rekordot.

Destroy:

Ez a függvény pedig egy adott id-jú Pekarú rekordot töröl az adatbázisból.

Ha sikeres a törlés, akkor egy 205 – Reset Content státuszkódot küld vissza.

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

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

13. kép

GetHigher/Lowerthan:

14. kép

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

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

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

Lehetővé teszi a pékáruk ár szerinti keresését egy weboldalon.

16. kép

15. kép

15. kép

FilterByFizetesimod:

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

17. kép

A FilterByFizetesimod lehetővé teszi a Rendelések szűrését fizetési mód alapján, és visszaadja a találatokat, ha léteznek.

Két féle fizetési mód jöhet szóba: Kártyás és készpénzes

Ha a keresésnél 0-t adunk meg akkor a készpénzzel kívánt fizetési kérelmeket kapjuk meg. Az 1-es megjelölés pedig a kártyával való fizetést jelenti.

The screenshot shows a REST client interface. The top bar displays the method 'GET' and the URL 'http://127.0.0.1:8000/api/pekseg3/filter/0'. A 'Send' button is visible. Below the URL bar, there are tabs for 'Query', 'Headers', 'Auth', 'Body', 'Tests', and 'Pre Run'. The 'Query' tab is active, showing 'Query Parameters' with a table with columns 'parameter' and 'value'. The 'Response' tab is also active, showing the JSON response. The status bar at the top right indicates 'Status: 200 OK', 'Size: 265 Bytes', and 'Time: 535 ms'.

Query Parameters

parameter	value
-----------	-------

Response

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

18. kép

SEEDER:

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

Az alábbi modelleket használja:

- **Felhasználó:** Felhasználói adatok tárolása
- **Pékaru:** Pékárúk tárolása
- **Cím:** Felhasználókhoz tartozó címek kezelése
- **Rendelés:** Megrendelések kezelése

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

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

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

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

19. kép

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