C# vizsgarész jegyzet

Konzolos:

Fájl betallózása (Add -> Existing item), majd Properties, Copy Always

A class (osuztályok) mindig legyenek publikusak (public), különben nem fogjuk elérni!

Létrehozásuk: Add -> New Class prop/propg/propfull stb….

Konstruktor létrehozása: ctor

program.cs-ben használjuk: static

JSON fájnál Dependencies -> Menage NuGet package

Új projekt hozzáadása: Solution Explorerben Solution -> Add New Projekt

Projekt hivatkozás: Dependencies -> Add project reference

WPF:

Oszlopok létrehozása:

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"></ColumnDefinition>

<ColumnDefinition Width="2\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

Adatok betöltése:

Dependencies -> Menage NuGet package : Dapper, MySqlConnector

Loaded="Window\_Loaded" -> F12

MySql connection:

string cs = "server=localhost;uid=root;pwd=;database=ingatlan";

Függőség kialakítása a másik projekthez:

Dependencies -> Add Project Reference

Betöltés:

using (var conn = new MySqlConnection(cs))

{

conn.Open();

string sql = "SELECT \* FROM adatbázisnév";

}

Listbox:

<ListBox x:Name="lbSellers" SelectionChanged="lbSellers\_SelectionChanged"></ListBox> -> F12

Button:

<Button x:Name="btnLoad" Content="Hirdetés betöltése" Click="btnLoad\_Click"></Button> -> F12

Frontend vizsgarész

Backend vizsgarész

Laravel new *projekt\_nev*

composer create-project laravel/laravel *projekt\_nev*

php artisan serve //szerver indítása

php artisan make:model -m *Modelnév (-m migráció létrehozása)*

* *Felesleges migrációk törlése a database->migrations mappából*

*composer remove laravel/sanctum //config mappából sanctum.php törlése*

*vendor\laravel\sanctum\database\migrations\table.php (sanctum mappa törlése)*

*Migrációkban az angol többesszám „s” törlése kategorias -> kategoriak, majd táblák létrehozása a Schema alatt:*

Schema::create('kategoriak', function (Blueprint $table) {

$table->id();

$table->string('nev')->unique();

});

App-> models- kategoria.php fájlon belül:

class Kategoria extends Model

{

use HasFactory;

public $table = 'kategoriak'; //kategoriak tábla

public $timestamps = false; //időbélyegzők megtiltása

public $guarded = []; //messas assigment művelet védelem tiltása

}

php artisan make:model -mc --api Ingatlan

* -mc //model és controller létrehozása
* --api CRUD műveletekhez szükséges műveletek létrehozása

$table->foreignId('kategoriak')->references('id')->on('kategoriak');

* foreignId // idegenkulcs
* references //hivatkozás id-ra
* on // hivatkozás a kategoriak táblán keresztül

$table->string('leiras')->nullable();

* string //tábla adatok típusa
* nullable //nem kötelezően kitöltendő adatok

$table->boolean('tehermentes');

$table->integer('ar');

$table->string('kepUrl')->nullable();

* boolean //logikai
* integer //egész szám típusú
* string // szöveg típusú
* ha nincs nullable, akkor alapértelmezetten kötelező
* nullable // nem kötelező

$table->date('hirdetesDatuma')->nullable()->default(DB::raw('CURRENT\_TIMESTAMP'));

* date // dátum típusú
* nullable // nem kötelező
* default //alapértelmezett tábla ->
  + ehhez szükséges: use Illuminate\Support\Facades\DB;

$table->string('type')->unique(); //unique azt jelenti, hogy egyedi

Database->seeders->databaseSeeder.php //ebben kell a táblákat feltölteni

$kategoriak = ['Ház', 'Lakás', 'Építési telek', 'Garázs', 'Mezőgazdasági terület', 'Ipari ingatlan'];

foreach ($kategoriak as $key => $value) {

Kategoria::create(['nev' => $value]);

}

* $kategoriak //kategóriák tábla, bene az adatokkal
* foreach járja be, a név érték lesz feltöltve

php artisan migrate –seed //migráció létrehozása adatbázis feltöltéssel

php artisan migrate:fresh // minden migrációt kidob és újra létrehozza

belongsTo -> belong tab+tab és kiegészíti

public function kategoria()

{

return $this->belongsTo(Kategoria::class, 'kategoria');

}

Routes-> Api.php

use App\Http\Controllers\IngatlanController;

Route::get('/ingatlan', [IngatlanController::class, 'index']);

App->http->Controllers->IngatlanController

public function index()

{

return Ingatlan::all(); //mindent visszaad  
 return Ingatlan::with('kategoria')->get(); // kategóriákkal tér vissza

}

Api.php-ban:

* Route::get('/orders', [OrderController::class, 'index']);

Az OrderControllerben:

* return Order::all(); // Mindent visszaad
* return Order::with('category')->get(); //kategóriákkal adja vissza

Tunderclient tesztelés

* <http://127.0.0.1:8000/api/orders> (SEND)

A teszteléshez Thunderclient extension telepítése (vagy postman), majd Collections -> New

Parancs segítségek:

php artisan help make:model

php artisan migrate:fresh /töröl minden migrációt és létrehozza újra

Az App/http/Models mappában az elkészült modellekhez beírni:

* public $timestamps = false;
* public $table = 'categories'; //categories, magyarul kategóriák

class Category extends Model

{

use HasFactory;

public $timestamps = false;

public $table = 'categories';

}

public function show($id)

{

$order = Order::where('id', '=', $id);

if (! $order->exists())

{

return response()->json(['message' => 'A megrendelés nem létezik.'], 404);

}

else

{

return $order->get();

}

}