Zengo teszt feladat megoldás

Takács Dániel

1. Felhasznált technológiák, frameworkok:

- backend : Laravel
- fronend : jQuery
- CSS : Bootstrap
- Docker, docker-compose

a. Konténerek

A `docker-compose.yml`-ben van definiálva a 3 futó konténer.

- mysql
  - ezen fut az adatbázis, ide kapcsolódik a backend. (3306 port)
- php
  - forráskód van rajta és futtatni az artisan commandokat rajta.
- nginx
  - szerver ami továbbítja a kéréseket a php konténernek és kiszolgálja a kéréseket.

Konténerek indítása , buildelése (**zengo_takacs_daniel** mappából).

```
docker-compose up -d --build
```

A **php** konténerre futtatni az adatbázis migration-t.

```
docker-compose exec php php /var/www/html/artisan migrate
```

A **php** konténerre futtatni az adatbázis seed-t. Feltölteni a megyék (`counties`) táblát adattal. (https://hu.wikipedia.org/wiki/Magyarorsz%C3%A1g_megy%C3%A9i alapján).

```
docker-compose exec php php /var/www/html/artisan db:seed
```

Csongrád-Csanád megye feltöltése teszt adattal.

```
docker-compose exec php php /var/www/html/artisan db:seed --
class=CityTestTableSeeder
```

Ellenőrizhetjük hogy sikeres volt-e a seed-elés.

```
docker-compose exec php php /var/www/html/artisan tinker
App\County::count()
```

b. App

Az app a http://localhost:8088/ címen érhető el.

- Elsőre csak a Megye választó látszik.
- Megye választás után megjelenik:
    - Új város felvitele az adott megyébe.
    - Megyéhez rögzített városok listája.
- Város nevére kattintva
    - Az elem inputtá változik benne a város nevével.
    - Mégse, Módosítás, Törlés gombok használhatók lesznek.
- Az összes művelet oldal újratöltődés nélkül megy végbe.

b/1. Model

App\City - Város model.

App\County - Megye model.

Model-eken végzett módosításokat és listázásokat az ORM-el csináltam.

Migration scriptek - src\zengotest\database\migrations

Seed scriptek - src\zengotest\database\seeds

b/2. Controller

- src\zengotest\app\Http\Controllers\CityController.php

Ez a Controller vezérli a műveleteket.

Actionok

- index() - Kezdő view megjelenítése
- getCities() - Város lista view megjelenítése
- addCity() - Város felvitele
- updateCity($id) - Város módosítása
- deleteCity($id) - Város törlése

b/3. Routes

- src\zengotest\routes\web.php

```
// index
Route::get('/', 'CityController@index');
// megye választás
Route::post('/getCities', 'CityController@getCities');
// város hozzáadás
Route::post('/addCity', 'CityController@addCity');
// város módosítás
Route::post('/updateCity/{id}', 'CityController@updateCity');
// város törlés
Route::get('/deleteCity/{id}', 'CityController@removeCity');
```

b/4. Ajax hívások

- src\zengotest\public\js\app.js

Megfelelő gombokra kattintva a felületen , ajax hívásra kerül a műveletnek megfelelő végpont az app-ban, a hozzá tartozó method-al.

Például **Törlés** művelet.

```
// Törlés
$('#cities-container').on('click', ".delCity", function () {
    let cityId = $(this).data('id');
    let element = this;
    $.ajax({
        type: 'get',
        url: '/deleteCity/' + cityId,
        success: function() {
            console.log('törölve');
            $(element).closest('tr').remove();
        }
    });
});
```

Üres névvel felvinni illetve módosítani nem lehet. Scriptek szűrve vannak.

Új város felvitele esetén és módosítás esetén a listát ABC sorban újra rendezi.

b/5. View-k

A view-okat struktútáltan készítettem el. Formázáshoz Bootstrap class-okat használtam, plusz egyedi ccs formázást (+src\zengotest\public\css\styles.css).

- `layouts.index` - Ezt az alap oldal strutúra , meta tag-k, ccs , scriptek.
- `main` - A kezdő view , ez töltődik a `layouts.index` **@content** szekciójába.
- `partials.cities` - a `main` view id="cities-container" div-be töltődik be, megyek választás után. (ha nem Válassz-t választunk).
- `partials.city` - `partials.cities` view-ban található táblázat sorai töltődnek be egyesével innen , minden városra külön @include.