


Fogarasi Gergő
Laravel próbafeladat
(CsigaProjectHandler)

technológiák kiválasztása

- min fejleszték → Lenovo T410, Windows 7 (most ez van kéznél)
- Nem használlok webszerveret, sem semmilyen hosting-céget, hanem a számítógépre telepített XAMPP segítségével próbálgatom ki a saját munkámat (localhost). (XAMPP 8.0.13 + PHP 8.0.13, az apachefriends.org oldalon ez a legújabb.) A XAMPP-ban van PHP és MySQL is, amire szükségem lesz.
- Laravel 8.65 (a composer ezt telepítette fel, mint legújabbat)
- Composer → legújabb (ezzel tudok PHP csomagokat – pl. a Laravel-t – telepíteni) (ehelyett Docker-t is lehetett volna használni)

képek (nem mindenről csináltam képet): főoldal (projektek listája)

← → 📄 localhost:8000/projects



Csiga Project Handler (Fogarasi Gergő)

Create New Project

ID	Name	Description	Status	The number of contacts	Action
4	aaa	bbb	awaiting development	1	Show Edit Delete
5	gtr	hzu	awaiting development	1	Show Edit Delete
6	dsfsdfs	sdfsdfsdf	awaiting development	1	Show Edit Delete
7	dsfsdfsdf	sdfsdfsdfsdf	awaiting development	1	Show Edit Delete
8	adatbázis-kezelő szoftver készítése	Minimalista SQL-motor készítése. Java nyelven kell elkészíteni, platformfüggetlennek kell lennie mindenképpen.	in progress	2	Show Edit Delete
9	sadfsdfs	dsfsdfsdfs	awaiting development	1	Show Edit Delete
10	dsfsdfsdf	sfsdfsdf	awaiting development	1	Show Edit Delete
11	dsfsdfsdf	sdfsdfsdfsdf	awaiting development	1	Show Edit Delete
12	dsfsdfsdfs	sdfsdfsdfsdf	awaiting development	1	Show Edit Delete
13	abc	5 kontakt van ehhez.	awaiting development	5	Show Edit Delete

Filter for status:
---show all---

status filter: (---show all---) (-1)

1 2 >

státuszra szűrve

4	aaa	bbb	awaiting development	1	Show Edit Delete
5	gtr	hzu	awaiting development	1	Show Edit Delete
6	dsfsdfs	sdfsdfsdf	awaiting development	1	Show Edit Delete
7	dsfsdfsdf	sdfsdfsdfsdf	awaiting development	1	Show Edit Delete
8	adatbázis-kezelő szoftver készítése	Minimalista SQL-motor készítése. Java nyelven kell elkészíteni, platformfüggetlennek kell lennie mindenképpen.	in progress	2	Show Edit Delete
9	sadfsdfs	dsfsdfsdfs	awaiting development	1	Show Edit Delete
10	dsfsdfsdf	sfsdfsdf	awaiting development	1	Show Edit Delete
11	dsfsdfsdf	sdfsdfsdfsdf	awaiting development	1	Show Edit Delete
12	dsfsdfsdfs	sdfsdfsdfsdf	awaiting development	1	Show Edit Delete
13	abc	5 kontakt van ehhez.	awaiting development	5	Show Edit Delete

Filter for status:
in progress

status filter: (in progress) (2)

1 1 2 >

projekt szerkesztése (ahol egyben a kapcsolattartók is szerkeszthetőek / létrehozhatóak / törölhetőek stb.)

Edit Project

Back

Name:
adatbázis-kezelő szoftver készítése

Description:
Minimalista SQL-motor készítése. Java nyelven kell elkészíteni, platformfüggetlennek kell lennie mindenképpen.

Status:
2

Contacts:
1,2

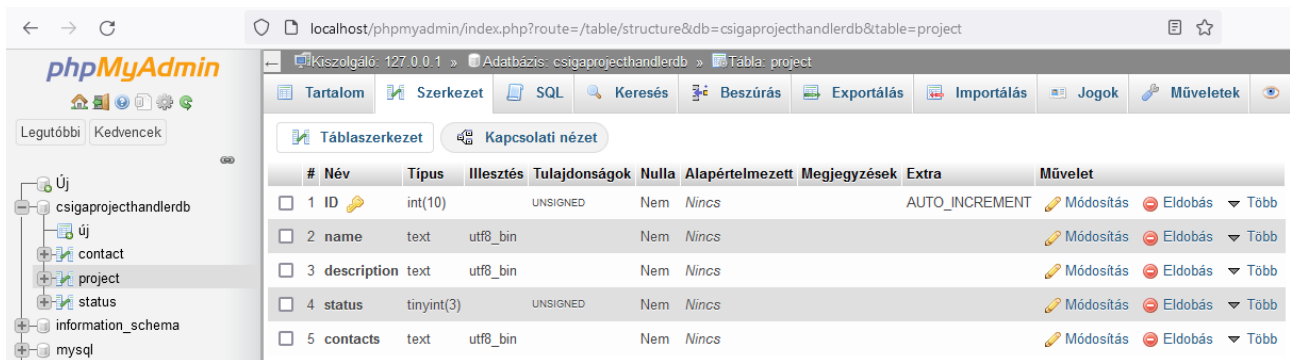
Submit

Create New Contact

ID	Name	E-mail	Action
3	Kis Odón	kisodon@citromail.hu	Show Edit Delete

adatbázis terv

- adatbázis neve: csigaprojecthandlerdb, utf8_bin
 - tábla: project
 - ID INT UNSIGNED PRIMARY A_I (pl. 17)
 - name TEXT utf8_bin (pl. „2D platformer játék fejlesztése”)
 - description TEXT utf8_bin (pl. „Android-ra kellene fejleszteni egy 2D platformer játékot, amiben egy piros színű négyzettel kell fákon ugrálni. Bla bla bla.”)
 - status TINYINT UNSIGNED (pl. 2)
 - contacts TEXT utf8_bin (pl. „1,4,6”)
 - tábla: status
 - ID INT UNSIGNED PRIMARY A_I (pl. 2)
 - description TEXT utf8_bin (pl. „awaiting development”, „in progress”, vagy „done”)
 - tábla: contact
 - ID INT UNSIGNED PRIMARY A_I (pl. 49)
 - name TEXT utf8_bin (pl. „Dr. Gipsz Jakab”)
 - email TEXT utf8_bin (pl. „drgipsz@citromail.hu”)



megjegyzések az adatbázistervvel kapcsolatban:

- project tábla / contacts → Az igazán szép megoldáshoz egy „junction table”-t kellene használni, viszont a stackoverflow-n lévő, ezzel kapcsolatos vitákat elolvasva, miniprojektek esetében az enyémhez hasonló megoldások is jók lehetnek. Ezen kívül igyekeztem beleférni a 3 órás időkeretbe, és próbáltam egyensúlyozni a jó megoldás, és a gyors megoldás között.
- Használhattam volna „foreign key”-eket a táblák összekötéséhez (pl. project tábla status ↔ illetve status tábla ID). Ehhez InnoDB engine-t kell választani a phpMyAdmin-ban. Meg lehet adni, mi történjen pl. update és delete esetben, így az ide vonatkozó kódrészletek egyszerűsödnek (az adatbázis bizonyos dolgokat magától megcsinál, ha ezeket előre definiáljuk).

időbeosztás

A feladat leírása szerint 3 óránál nem lenne szabad többet foglalkoznom a feladattal, de lehet, hogy ennél kicsit több időre lesz majd szükségem. Leírom, hogy mivel mennyi időt foglalkoztam, így látszik, hogy mihez mennyi időre volt szükségem. Egyébként eddig még sosem foglalkoztam Laravel-lel (csak egy egyetemi kurzust hallgattam végig ezzel kapcsolatban, sok évvel ezelőtt), ezért néhány dolognak utána kellett néznem.

időtartam	Σ idő	leírás
3 perc	00:03	elkezdtém a feladatot, először elolvastam a feladat leírását
10 perc	00:13	letöltöttem és telepítettem a XAMPP-ot, az Apache és MySQL modulokat elindítottam
2 perc	00:15	letöltöttem és telepítettem a composer-t
5 perc	00:20	Laravel projekt létrehozása composer segítségével, Windows parancssorban: cd C:\xampp\htdocs\ composer create-project --prefer-dist laravel/laravel composer create-project --prefer-dist laravel/laravel:^8.0 CsigaProjectHandler
5 perc	00:25	Laravel kipróbálása: cd c:\xampp\htdocs\laravel php artisan serve böngésző → localhost:8000
20 perc	00:45	adatbázis megtervezése, közben olvastam ide vonatkozó dolgokat (pl. a stackoverflow-n) adatbázisterv megvalósítása (phpMyAdmin-ban) a 3 adatbázistábla feltöltése némi adattal, kezdésnek (phpMyAdmin-ban) .env fájl szerkesztése (Laravel hogyan férjen hozzá az adatbázishoz)
10 perc	00:55	cd C:\xampp\htdocs\laravel\ php artisan make:model contact php artisan make:model project php artisan make:model status C:\xampp\htdocs\laravel\app\Models\status.php project.php contact.php Ezt a 3 fájlt kiegészítettem. (Az adatbázistáblákat a php artisan" paranccsal is legenerálhattam volna.)
5 perc	01:00	laravel/routes/web.php fájl szerkesztése (route-szabályok megadása)

időtartam	Σ idő	leírás
30 perc	01:30	<pre>cd C:\xampp\htdocs\laravel\ php artisan make:controller ContactController --resource --model=contact php artisan make:controller ProjectController --resource --model=project php artisan make:controller StatusController --resource --model=status</pre> <p>Ezután a létrejött 3 db controller-fájlt kiegészítettem (az egyes függvényeket kitöltöttem kóddal).</p> <p>CRUD (create, read, update, delete) szemlélet szerint alakítottam ki a dolgokat.</p>
30 perc	02:00	Időközben tutorial-ok, dokumentációk keresgélése, olvasása (főként a Laravel-ről).
20 perc	02:20	Időközben rövid kis dokumentáció-írás. Néha hozzáírok pár sort.
10 perc	02:30	laravel/app/Models mappában generálódott 3 db fájl kiegészítése
30 perc	03:00	<p>laravel/resources/views mappán belül 3 db mappa létrehozása: contacts, projects, statuses</p> <p>az új 3 mappán belül blade-fájlok létrehozása, és ezek kitöltése kóddal</p>
2 perc	03:02	Logo készítése (GIMP segítségével).











időtartam	Σ idő	leírás
30 perc	03:32	<p>Most már 70-80%-ban a lényeg elkészült, elvileg működni kellene. A terminálban kiadott alábbi paranccsal lehet elindítani a szervert, hogy a Laravel-ben (PHP-ban) írt kódot ki lehessen próbálni:</p> <pre>php artisan serve</pre> <p>Ezután a böngészőben lehet nézegetni a dolgokat:</p> <p>http://localhost:8000/projects</p> <p>Hasznos lehet, ha az F12-vel megnyitott developer tools-ban nézegetjük a dolgokat. (Nekem ez a chrome-ban jobban tetszik, mint a FireFox-ban, szerintem több minden van benne. A Chrome developer tools-ával kapcsolatban egészen sok munkatapasztalatom van, sokat dolgoztam vele – mondjuk ez főleg frontend-hez jó, de backend fejlesztésnél is lehet nézegetni, hogy pl. mit küldök el, és mi jön vissza a backend-ről válaszként.)</p> <p>Első próbálkozásra különféle (hasznos és szépen szerkesztett) Laravel hibaüzenetet kaptam. Nyomozgattam a stackoverflow-n, és szép lassan kigyomláltam az összes hibát, utána pedig már működött minden jól. Ez a debug nagyjából fél óráig tartott. (Jellemzően elgépelési hibák voltak.)</p>
30 perc	04:02	Blade-fájlok további szerkesztése, kiegészítése, javítása.
30 perc	04:32	Github, git → beállítások.
5 perc	04:37	Időnként commit git-be.
5 perc	04:42	Létrehoztam 10-nél több projektet, így automatikusan megjelent a paginátor. Utána ezt próbálgattam, hogy jól működik-e. Találtam egy apróbb hibát, ezt kijavítottam.
10 perc	04:52	A paginátor alatt két óriási méretű nyíl volt (ezek is a paginátorhoz tartoznak). Utánanéztam, az AppServiceProvider-ben kell megadni, hogy a paginátor kinézetéhez bootstrap-et akarok használni, innentől már normálisan néz ki a paginátor alatti rész is. Kicsit utána kellett néznie a dolognak.








időtartam	Σ idő	leírás
35 perc	05:27	<p>Ezután a státusz szerinti szűrést kezdtem el megvalósítani. Paginator-t használok. A megfelelő blade fájlban van egy „\$projects” nevű változó, amin végigmegyek egy @foreach segítségével, és így építek belőle táblázatot, ez alatt pedig ott a paginator, amivel lapozni lehet, oldalanként legfeljebb 10 elem jelenik meg.</p> <p>A @foreach-nél mindenféle feltételeket meg lehet adni, tehát lehet például rendezni, szűrni is. A gond az, hogy mire a blade fájlhoz érünk, már legenerálásra kerül az, hogy tízesével hogyan oszlanak el az egyes elemek az egyes lapozható oldalakra. Vagyis ha szűrést, rendezést, stb. akarok megvalósítani, az csakis oldalon belül fog végrehajtódni. Tehát ha van 28 darab elem (oldalanként legfeljebb 10 jelenik meg ezekből), és rászűrök valamire, akkor mondjuk az 1. oldalon semmi sem fog megjelenni, a 2. oldalon 2 db dolog fog megjelenni, a 3. oldalon pedig mondjuk 1 db dolog fog megjelenni – pedig ehelyett az lenne a helyes, ha egyetlen oldal lenne ez esetben, ahol csak a 3 db dolog jelenne meg, melyek a szűrés feltételének megfelelnek.</p> <p>Ez tehát azt jelenti, hogy nem a blade fájl @foreach részénél kell megvalósítani a szűrést (filter, where), hanem valahol jóval „korábban”. Ha ennek mélyebben utánanéznék (2-3 óra turkálás a dokumentációban, stackoverflow-n, stb.), biztosan meg tudnám oldani normális módon.</p> <p>De hogy ne tartson ilyen sokáig a dolog (mert hogy a jó megoldás mellett a relative kevés ráfordított időre is próbálok törekedni), a @foreach környékén valósítom meg a státuszra szűrést, de úgy, hogy a szűrésnek megfelelő és nem megfelelő elemeket is megjelenítem, csak eltérő háttérszínnel (kikapcsolt szűrő esetén pedig mind egyforma színnel jelenik meg). Tehát egyik sem fog eltűnni a szűréstől. Ennek hátránya, hogy a paginatorral végig kell lapozgatni, ha kíváncsiak vagyunk, hogy mik a szűrés eredményei.</p> <p>Kezdetben olyan elképzelésem is volt, hogy a blade template-fájlban a @foreach-en belülre teszek egy @if-et, és egy PHP-változótól függően rakok a <tr> HTML-element-re CSS-t, és a jQuery-s dropdown által állítgatott JS-változót AJAX segítségével átadom a PHP-nak. Ez viszont túl bonyolult lett volna, úgyhogy azt csináltam, hogy a @if-ek segítségével kamu CSS class-okat rakok a <tr> element-ekre, majd jQuery-vel megnézem, melyik <tr> element-en milyen kamu CSS class van, és ettől függően rakok rájuk jQuery-vel CSS formázást. Tehát először a Laravel legenerálja a DOM-ot, majd utána a jQuery a dropdown állapotától függően azonnal (az oldal újratöltése nélkül) megformázza a dolgokat, azaz elvégzi a szűrést. Konkrétan, a kiszűrt elemeket majdnem teljesen elhomályosítja, míg a többieket nem. A megoldás hátránya még, hogy lapozáskor újra be kell állítani a dropdown-t a kívánt értékre (nem jegyzi meg). Ez megoldható lenne pl. cookie, localStorage, stb. segítségével.</p>
5 perc	05:32	<p>A routes/web.php fájl apró módosításával megoldottam, hogy a főoldal (localhost:8000) automatikusan irányítódjon át a projekteket kezelő oldalra (localhost:8000/projects). Ezen kívül van még egy státuszokat szerkesztő felület amire sehonnan sem mutat link (és nincs is rá szükség, mert ezek fixek), illetve egy kontaktokat szerkesztő/kezelő felület (erre van szükség, de sehonnan sem mutat rá link, és a projekteket kezelő felületbe beépítve kellene lennie).</p>
10 perc	05:42	<p>A főoldalon, ahol a projekteket kilistázó, paginátoros táblázat van, ott eddig a kontaktok felsorolása volt látható az egyik oszlopban, de a feladat nem ezt kérte, hanem helyette a darabszámukat. Szóval átalakítottam a kódot, hogy ezt mutassa.</p>

időtartam	Σ idő	leírás
15 perc	05:57	<p>A főoldalon, ahol a projekteket kilistázó, paginátoros táblázat van, ott eddig a státusznál egy ID volt (pl. 1, 2, vagy 3), és a státuszokat tartalmazó MySQL-adatbázistáblából lehetett megnézni, hogy ez mit is jelent. Ezt egy trükkal átalakítottam úgy, hogy ID helyett a státusz-leírás jelenjen meg (pl. „Done”).</p> <p>Ehhez pár dolognak utána kellett olvasnom.</p> <p>Kb. ez a megoldásom lényege:</p> <pre><td>{{ DB::table('status')->select('description')->where('ID', \$project->status)->get()->first()->description }}</td></pre> <p>A blade-template-fájlba bele-hack-elt közvetlen DB-elérés nem feltétlenül nagyon szép megoldás, úgyhogy ezen még lenne mit refaktorálni, de működik.</p> <p>(Közben gyorsan azt is megoldottam, hogy ne dupla csiga logó kép jelenjen meg, hanem csak egy darab.)</p>
5 perc	06:02	<p>A kapcsolattartók létrehozásánál és szerkesztésénél az e-mail-címre vonatkozóan nem volt validálás. Utánanéztem, és van ilyen feature beépítve a Laravel-be. Szóval ezt:</p> <pre>'email' => 'required'</pre> <p>ki kellett cserélnem erre:</p> <pre>'email' => 'required email'</pre> <p>Kipróbáltam, és jól működik. Szándékosan rosszul megadott e-mail-címet nem enged elfogadni:</p>

időtartam	Σ idő	leírás
10 perc	06:12	<p>A feladat azt kérte, hogy a projekt létrehozásánál, valamint szerkesztésénél lehessen kezelni mindent a kapcsolattartókkal kapcsolatban. Tehát ne legyen egy külön view a kapcsolattartók kezelésére.</p> <p>Ezt úgy oldottam meg, hogy van egy külön view a kapcsolattartók teljeskörű kezelésére (létrehozás, szerkesztés, kilistázás, egyéni nézet, törlés), és ezt beletettem egy iFrame-be, és ez jelenik meg a projekt létrehozásánál, illetve szerkesztésénél is.</p> <p>Az iFrame talán kicsit nyomi és divatjamúlt dolog, de végül is teljesül az adott elvárás, és működik. Esetleg érdemes lenne gondolkodni valamilyen jobb megoldáson, de egyelőre ezt tudtam kitalálni.</p>
30 perc	06:42	<p>A getChanges() segítségével le lehet kérdezni, hogy melyik paraméterek módosultak projekt-szerkesztés esetén, így ezt el lehet küldeni a projekthez kapcsolódó emberek e-mail-címeire. A vonatkozó SMTP-beállításokat nem csináltam meg, de a többi részét megírtam.</p>

Miket használtam az álláshirdetésben felsorolt technológiák közül a feladat megoldásához?

-  PHP 7+ → PHP 8.0.13-at használtam
-  HTML
-  CSS → készíthettem volna szép CSS-fájlokat, van is ebben tapasztalatom, ehelyett csak néhány ide-oda betett inline CSS-t használtam, ami nem túl szép
-  JavaScript, ES6 → egy kevés JS-et és jQuery-t használtam
-  Vue.js → npm-mel telepítettem volna Vue.js-t, és azzal csinálhattam volna valami szép frontend-et, így az AJAX azt is megoldotta volna, hogy ne töltsdjön mindig újra az oldal bármilyen műveletnél. Az AngularJS tapasztalatom miatt szerintem nem okozna gondot Vue.js-sel foglalkozni.
-  Laravel
-  MySQL / MariaDB → MySQL-t használtam
-  Bootstrap 4
-  composer, npm → composer segítségével telepítettem a Laravel-t
-  git

-  SASS → ebben is van némi tapasztalatom, de már nem maradt idő arra, hogy rendes CSS fájlokat, vagy akár SASS-t használjak
-  Tailwind CSS
-  Inertia.js
-  REST API tervezés
-  Clean code
-  SOLID elvek
-  tesztelhető kód, TDD (test-driven development)

Mik készültek el a feladat leírásából?

























A megoldást lehetőség szerint egy általad választott git repoba feltöltve küldd el nekünk (pl. GitHub).



Reszponzív megjelenés nem szükséges, a rendszernek elég asztali gépen, 1080p felbontáson működni. → Mivel használtam Bootstrap-et (*amiben egyébként van is munkatapasztalatom az Attrecto-tól*), valamennyire rezponzív lett, például a show-edit-delete gombok keskeny képernyőn egymás alá rakódnak, szélesebb képernyőn egymás mellé kerülnek.

Feladat: egy egyszerű projektkezelő rendszer építése. A rendszerben lehessen:

-  Projekteket listázni, létrehozni, szerkeszteni, törölni
 -  Egy projekt rendelkezzen névvel, leírással és státusszal. Minden adat megadása kötelező
 -  A státusz lehet “fejlesztésre vár”, “folyamatban”, illetve “kész”
-  A projektekhez lehessen kapcsolattartókat felvenni
 -  Egy kapcsolattartó rendelkezzen névvel és e-mail címmel
 -  Egy projekthez tetszőleges számú kapcsolattartó tartozhasson
 -  A kapcsolattartókat a projekt létrehozása / szerkesztése során lehessen szerkeszteni és törölni, ne kelljen hozzá külön űrlapot használni
-  A rendszer egy projekt módosításakor küldjön automatikus e-mailt a projekthez tartozó kapcsolattartók számára, amelyben szerepelnek a megváltozott adatok (de csak azok)
→ megfelelő SMTP-szervert nem állítottam be, de egyébként megírtam a szükséges kódot [ide vonatkozóan](#)
-  Legalább a törlés funkció aszinkron módon, az oldal újratöltése nélkül történjen →
feltételezem, hogy ehhez AJAX kellene, és akkor már valami rendes frontend keretrendszer sem ártana, mint amilyen a Vue.js. Jelenleg minden műveletnél újratöltődik az oldal.

-  Két screen szükséges
 -  Projektlista (kezdőképernyő)
 -  Projektek listázása, a listában a következő adatok megjelenítése:
 -  Név
 -  Státusz
 -  Kapcsolattartók száma
 -  Szerkesztés illetve törlés gomb
 -  Legfeljebb 10 projekt megjelenítése, 10 feletti projektszám esetén lapozó
 -  A projekteket lehessen státusz szerint szűrni
 -  Projekt űrlap
 -  A projekt adatainak szerkesztése (név, leírás, státusz)
 -  Kapcsolattartók szerkesztése
-  Felhasználókezelés, azonosítás nem szükséges

Amit még lehetett volna, de nem került rá sor / nem maradt rá idő:

- Projekt szerkesztésénél/létrehozásánál a státuszt az ID begépelésével kell megadni, és nincs semmilyen validáció. Ez így nyilván nem jó. Egy adatbázisból feltöltött dropdown kellene.
- A megadott adatokat legalább backend oldalon validálni kellene (mert az nehezebben meghackelhető).
- Projekt szerkesztésénél/létrehozásánál a kapcsolattartókat az egyes ID-k vesszővel elválasztva történő begépelésével lehet megadni, validáció nélkül. Ez így nyilván nem jó. Adatbázisból generált checkbox-lista kellene, és ott lehetne ki-be pipálgatni az egyes kapcsolattartókat.
- Bootstrap-et, jQuery-t, stb. letölteni, és fájlként hivatkozni, nem pedig internetes linkként hivatkozni (mert onnan eltűnhet) Kerülni kell a függőségeket, ha lehet.
- A TAB-okat find&replace segítségével kicserélni 4 db space-re minden kódfájlban, majd egy kicsit átnézni a kódfájlokat, hogy minden rendben van-e. (Bizonyos nyelveken – pl. GNUmake – a TAB-nak nyelvtani jelentése van, és nem használhatunk helyette pl. 4 db space-t. Most viszont nem használtam ilyen nyelvet.)
- Újratöltődés nélkül működjenek bizonyos feature-ök → például projekt törlése, ehhez valószínűleg AJAX kellene, és akkor már használhatnánk valamilyen JS frontend keretrendszert (pl. Vue.js)
- 3-féle státusz van, ezek ID-ja 1, 2 és 3. Ez bizonyos helyeken be van építve a kódba, pedig van ehhez egy adatbázistábla. Inkább mindent dinamikuson onnan kellene kiolvasni, és ha változik, akkor elég lenne a status nevű táblát módosítani. Ezt kb. 10-20 perc lenne normálisra átírni, refaktorálni.
- bizonyos esetekben (pl. a logó-kép megadásánál) a kódba be van építve az, hogy localhost-on fut a webalkalmazás → ez így nem jó, deployment-nél gond lehet, hogy sok helyen kézzel át kell írni
- kódot átnézni, indentálást, kód szép formázottságát, esetleges elgépeléseket, stb. javítani
- Frontend jól nézzen ki. Bootstrap 4-et használtam, de esetleg még Vue.js-t is használhatnék a frontend-hez.
- Tesztelés, unittest lehetőségeken gondolkodni.