

Fogarasi Gergő
Laravel próbafeladat
(CsigaProjectHandler)

technológiák kiválasztása

- min fejleszték → Lenovo T410, Windows 7 (most ez van kéznél)
- Nem használok webszerver, sem semmilyen hosting-céget, hanem a számítógépre telepített XAMPP segítségével próbálom 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)

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	elkezdtem 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 CsignaProjectHandler
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.
		Paginator kipróbálása (ha 10-nél több elem van, akkor jelenik csak meg).
		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.)
		Még egyszer átnézni az általam készített rendszert, hogy valóban megvalósít-e mindent, ami a feladat kiírásában szerepelt.
		Frontend jól nézzen ki. Bootstrap 4-et használtam, de esetleg még Vue.js-t is használhatnék a frontend-hez.
		Tesztelés, unittest lehetőségeken gondolkodni.

időtartam	Σ idő	leírás
		Kód alapos átnézése, esetlegesen talált szintaktikai / szemantikai hibák, indentálás, felesleges szóközök, stb. javítása.
		<p>Újratöltődés nélkül működjenek bizonyos feature-ök</p> <p>béna megadás kijavítása (pl. contacts, status)</p> <p>ne 3 lapon legyen szétszedve hanem egyben legyen minden</p> <p>legyen 10+ projekt → paginátor kipróbálása</p> <p>főoldal rögtön a „projects” legyen</p> <p>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</p> <p>Alaposan átnézni a kódot és a működést is!</p>

Miket használtam az álláshirdetésben felsorolt technológiák közül a feladat megoldásához?

-  PHP 7+ → PHP 8.0.13-at használtam
 -  HTML
 -  CSS
 -  JavaScript, ES6, Vue.js
 -  Laravel
 -  MySQL / MariaDB → MySQL-t használtam
 -  Bootstrap 4
 -  composer, npm → composer segítségével telepítettem a Laravel-t
 -  git
-
-  SASS
 -  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 működnie. → 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:

-  Projektek 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)
-  Legalább a törlés funkció aszinkron módon, az oldal újratöltése nélkül történjen
-  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