Szegedi Szent Benedek School of Business Technikum

Minden Egy Helyen

Szakképesítés neve: Szoftverfejlesztő -és tesztelő

Témavezető: Babinszky Mónika, Bodrogi Péter

Készítők: Trájer Balázs, Lipták Dominik, Rácz Norbert

Tartalom

[Korai Frontend 4](#_Toc157755618)

[Fő oldal 4](#_Toc157755619)

[Regisztrációs felület 5](#_Toc157755620)

[Bejelentkezési felület. 5](#_Toc157755621)

[Profil 6](#_Toc157755622)

[Korai backend 7](#_Toc157755623)

[Adatbázis 7](#_Toc157755624)

[Adatbázis csatlakozás 7](#_Toc157755625)

[Regisztráció 8](#_Toc157755626)

[Bejelentkezés 11](#_Toc157755627)

[Kijelentkezés 13](#_Toc157755628)

[Korai végpontok 13](#_Toc157755629)

[Változások a projektben 14](#_Toc157755630)

[Laravel 14](#_Toc157755631)

[Mi az a Laravel? 14](#_Toc157755632)

[Laravel projekt kezdése 14](#_Toc157755633)

[Mi az a Composer? 16](#_Toc157755634)

[Laravel Backend 17](#_Toc157755635)

[Új adatbázis csatlakozás 17](#_Toc157755636)

[MVC – Model View Controller 18](#_Toc157755637)

[Model (Modell) 18](#_Toc157755638)

[View (Nézet) 19](#_Toc157755639)

[Controller (Vezérlő) 19](#_Toc157755640)

[Hitelesítés 19](#_Toc157755641)

[Regisztáció logika 19](#_Toc157755642)

[Bejelentkezés logika 20](#_Toc157755643)

[Kijelentkezés logika 20](#_Toc157755644)

[Admin felület 20](#_Toc157755645)

[Hozzáférési jogok 22](#_Toc157755646)

[Laravel Frontend 23](#_Toc157755647)

[Regisztráció nézet 23](#_Toc157755648)

[Bejelentkezés nézet 24](#_Toc157755649)

[Admin oldal 25](#_Toc157755650)

[Profil oldal 25](#_Toc157755651)

# Korai Frontend

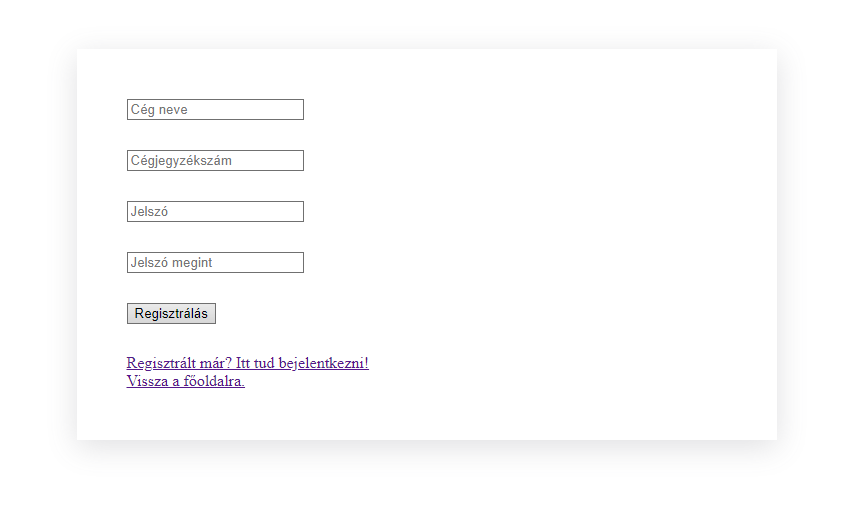
## Fő oldal

A korai megjelenése a weboldalnak nagyon minimalista volt, hiszen csak a csatlakozás volt vizsgálva a frontend és backend közt.



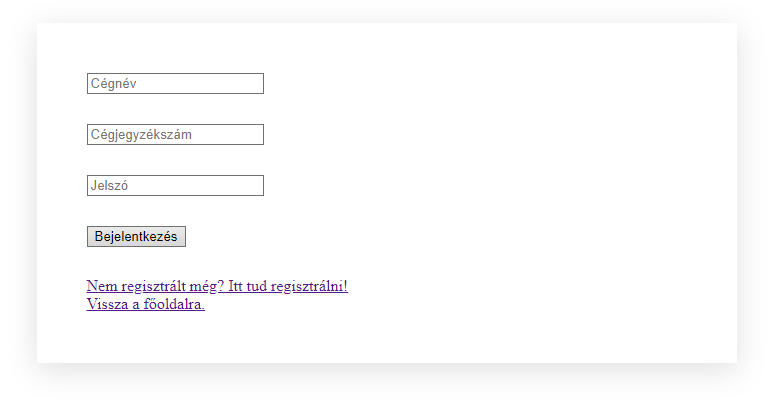
Ahogy lehet látni csak egy cím és 2 gomb jelenik meg. A „*mainPage*” kiírás csak azért jelenik itt meg, mert ez a neve a file -nak és hogy lehessen tudni, ha backendben lévő átirányítás jó helyre irányított e át. A „*regisztráció*” gomb a „*registration.php*” oldalra vezet át, a „*Belépés*” gomb a „*login.php*” oldalra vezet át.

## Regisztrációs felület



A korai regisztrációs felület csak egy formanyomtatványból áll, nincs semmi stílus beállítva az oldalnak. Csak annyi lett változtatva benne, hogy kapott árnyékot a térhatás miatt meg a margók lettek be állítva.

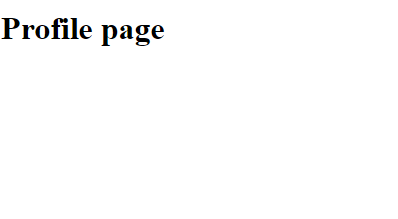
## Bejelentkezési felület.



A bejelentkezési felület sem több, mint a regisztrációs felület, mert ugyan azt a stílus file -t kapja meg.

Mind kettő felületen megjelenik egy „*Nem regisztrált még?” / „Regisztrált már?”* hivatkozás, meg egy „*Vissza a főoldalra.”* Hivatkozás, ezek, ahogy ki is olvasható, a megfelelő oldalra továbbítja a felhasználót át.

## Profil



A profil oldalon jelenleg nem jelenik meg semmi sem, itt csak az van vizsgálva, hogy a felhasználó be van e jelentkezve.

# Korai backend

A backendet eleinte 4 file -ra lehet osztani, adatbázis alap csatlakozás, regisztráció, bejelentkezés és kijelentkezés.

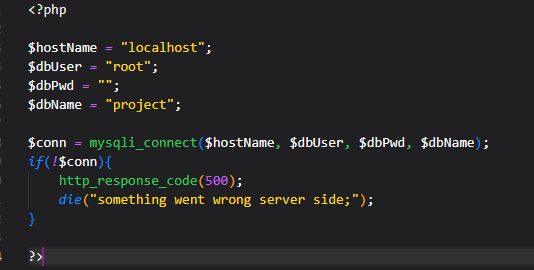
## Adatbázis

Az adatbázisban 4 oszlop van: *id, cegnev, cegjegyzekszam, jelszo*.



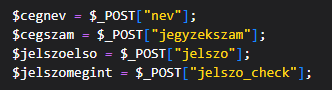
A fenti képen lehet látni azokat az oszlopokat, ahova a felhasználó adja meg az adatokat.

## Adatbázis csatlakozás



Fenti képen látható, hogy a *„mysqli\_connect()”* paranccsal lehet létrehozni a kapcsolatot, jelen esetben az adatbázis XAMPP -on keresztüli MYSQL adatbázis. Ahhoz, hogy a csatlakozás létrejöjjön, ahhoz kell 4 paraméter, **host név**, **felhasználó név**, **jelszó** és maga az **adatbázis név**. Jelen esetben a paraméterek változókba vannak eltárolva és azok vannak átadva. Ha esetleg nem tudna létrejönni a kapcsolat a kód és az adatbázis közt, akkor egy 500 -as választ kapunk, az ***500*** -as válaszkód azt jelzi, hogy valamilyen hiba történt a szerveren, amely miatt a kérés nem tudott feldolgozódni.

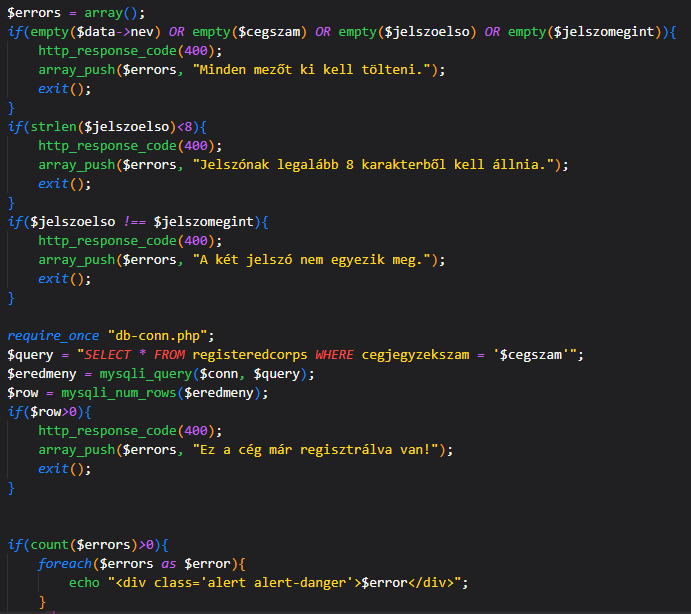
## Regisztráció



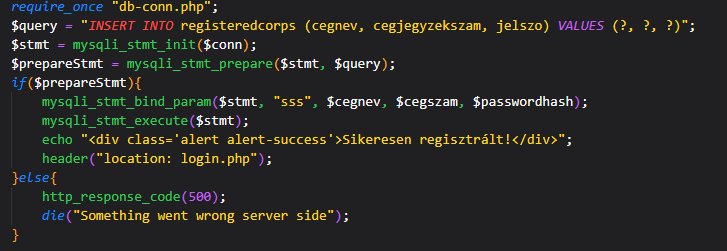
A regisztráció már egy kicsit bonyolultabb, mint a csatlakozás. A fenti képen látható, hogy a felhasználói felületen megadott adatokat így kéri át a backend. Mivel a frontenden a regisztrációs felület ***POST*** metódust használ, ezért itt is így kell megadni a változóknak.



Visszaélés meggátlása érdekében a jelszó, amit megkap a változó titkosítva van. Ez az adatbázis felületén úgy látszik, hogy emberileg érthetetlen szöveg jelenik meg a mezőben.



Első sorban a hibakezelés. Ahogy látszik, ha bármelyik mező üres, akkor ***400*** -as választ kapunk. Az 400-as válaszkód azt jelzi, hogy a kérés hibás volt, és a szerver nem tudta feldolgozni. Ha esetleg a jelszó nem lenne 8 karakter hosszúságú, vagy nem egyezik meg a kettő jelszó, akkor ugyan úgy ***400*** -as választ kap a felhasználó. Abban az esetben, ha minden mező ki lett töltve és minden adat megfelel a paramétereknek, akkor egy lekérést indít a kód, hogy ilyen cégjegyzékszámmal van e regisztrálva cég. Onnan lehet tudni, hogy van ilyen számon regisztrálva cég, hogy a lekérés eredményét a „*mysqli\_num\_rows()”* függvénnyel vizsgálja és egy számot ad vissza eredményül, ha az eredmény nagyobb mint nulla, akkor van már regisztrálva cég azon a számon és ***400*** -as választ kap. Végül, ha van valami probléma a regisztrációnál akkor azokat kiírja a végén.



Ha minden adat megfelelt a paramétereknek, akkor egy beszúrást indít a kód, amivel a táblába beszúrja az adatokat. Ha valami hiba történne az a folyamat közbe, akkor ***500*** -as választ kap a felhasználó egyéb esetben sikeres volt a regisztráció, ***200*** -as választ kap és továbbít a bejelentkezés oldalára.



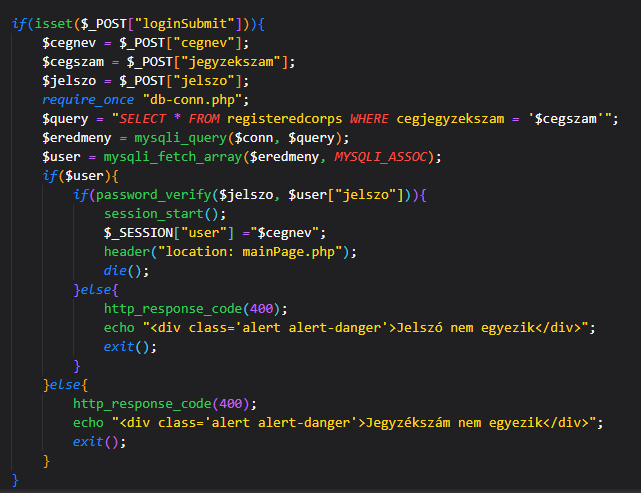
Ennél a függvénynél feltűnhet a *„sss”* paraméter, ez azt adja meg, hogy az értékek, amiket át ad azok *string -*ek, avagy szövegformátumúak.



Ha regisztrálni akar egy felhasználó akkor csak kitölti a mezőket és rányom a regisztráció gombra.



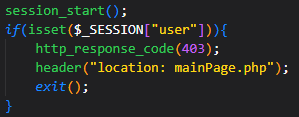
## Bejelentkezés



A regisztráció folyamata nem sokban különbözik a regisztrációtól. Ennél is a bejelentkezés felülete ***POST*** metódust használ, ezért így kapják meg a változók a megadott értékeket. A lekérdezés itt csak a cégjegyzékszámot vizsgálja. Ha nem talál ilyen számot a táblán akkor ***400*** -as választ kap. Abban az esetben, ha talált ilyen számot, akkor egy változóhoz hozzárendeli, hogy a kapott értéknek a sora egy tömb legyen, így ha a „*$user”* valamelyik értékére akarok hivatkozni, akkor ezen keresztül lehet, ez a lenti képen látszik.



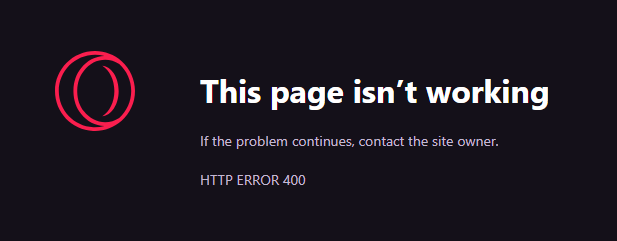
Itt az van vizsgálva, hogy a *jelszó*, amit kapott a kód az megegyezik e a lekért sor jelszó oszlopával. Ha a jelszó megegyezik, akkor elindul ez a munkamenet, a „***$\_SESSION***” eltárolja a cég nevét, ***200*** -as választ kap, majd a főoldalra irányít vissza.



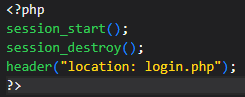
A legtöbb oldal el van látva a fenti kóddal, ami annyit vizsgál, hogy ha van már munkameneti felhasználó, akkor ***403*** -as választ kap és visszairányítja a főoldalra a felhasználót. Az 403-as válaszkód azt jelzi, hogy a kérés jogosultságok miatt el lett utasítva, és a kliensnek nincs hozzáférése az erőforráshoz. Ezeket az oldalakat csak úgy tudják munkamenet közben elérni, ha a domain címhez hozzáírják a megfelelő végpontot, ami ebben az esetben lehet „*login.php”* vagy „*registration.php*”.



A képen lehet látni, hogy egy olyan céggel akarok bejelentkezni, amely nincs regisztrálva ilyen néven, de jegyzékszámon van már regisztrálva cég. Ha így küldök egy bejelentkezést az adatbázisnak, akkor helyes válaszkódot kapok.



## Kijelentkezés



A kijelentkezés a legrövidebb kód, itt csak megsemmisíti a munkamenetet és visszairányít a bejelentkezés oldalára.

## Korai végpontok



A fenti ábrán látszódnak a korai végpontjai a projektnek.

# Változások a projektben

Mivel a projektnek kell egy keretrendszer, ezért az egész projektet át kell dolgozni. Ebben a projektben a keretrendszer a Laravel lesz.

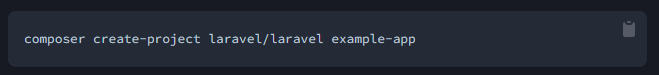
# Laravel

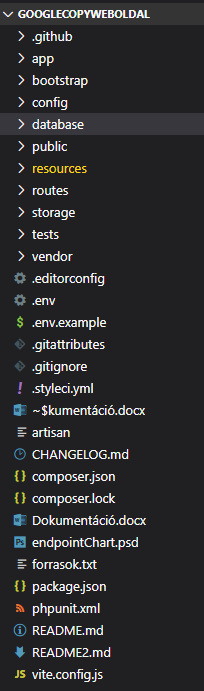
## Mi az a Laravel?

A Laravel egy nyílt forráskódú, PHP programozási nyelven írt webes alkalmazásfejlesztési keretrendszer. Az alapjaiban olyan eszközök és struktúrák gyűjteménye, amelyek segítenek fejlesztőknek a webes alkalmazások gyors és hatékony építésében. A Laravel célja, hogy egyszerűsítse a fejlesztési folyamatot, lehetővé téve a kód újra felhasználhatóságát és a könnyű karbantarthatóságot. A Laravel számoskiegészítőt, csomagot és eszközt kínál, amelyek lehetővé teszik a fejlesztők számára a gyors és hatékony fejlesztést. Ez a keretrendszer népszerűvé vált a fejlesztők körében a könnyű használhatósága, a kiterjedt dokumentációja és a nagy közössége miatt.

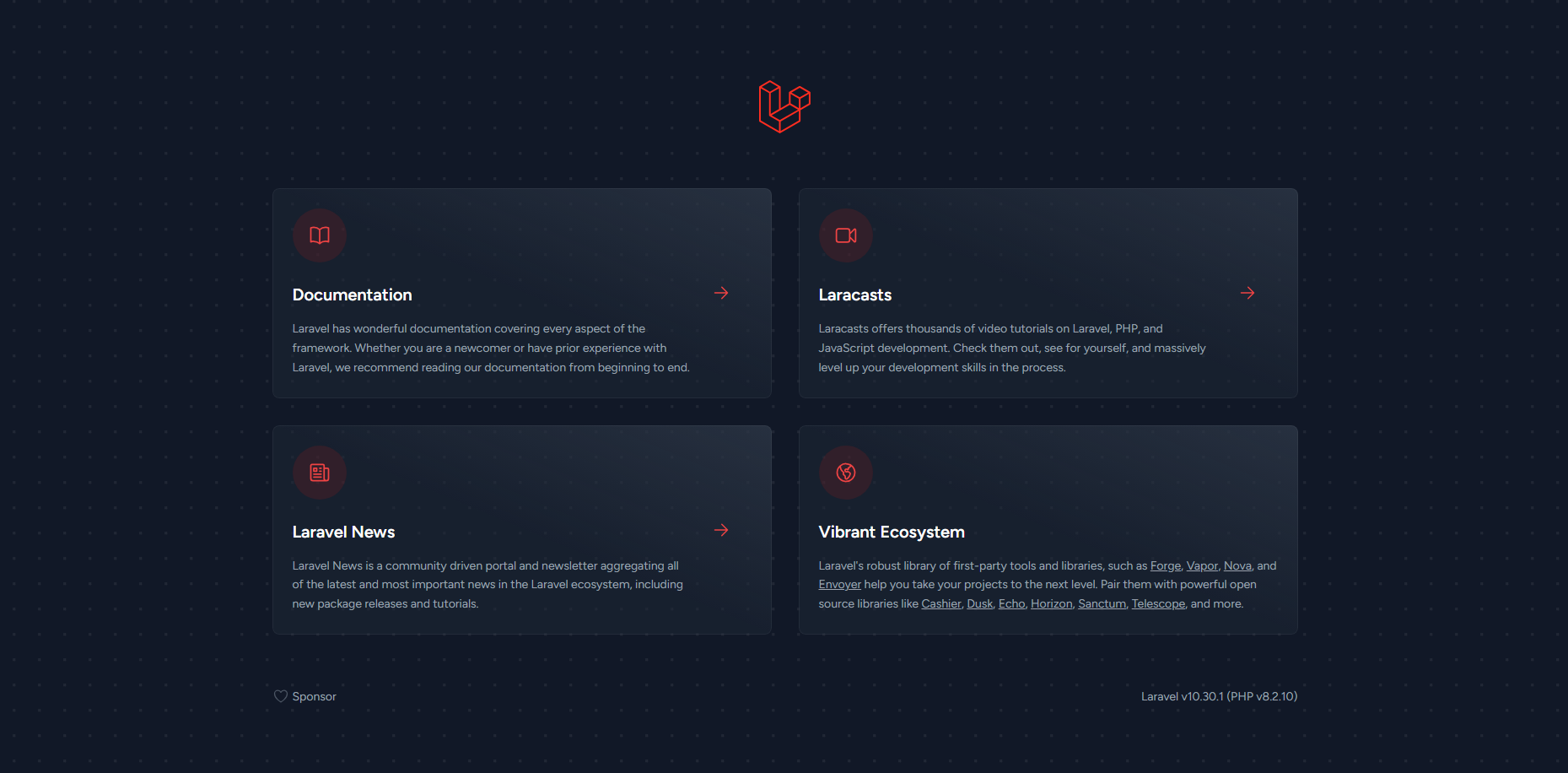
## Laravel projekt kezdése

Egy Laravel projekt úgy kezdődik, hogy elsősorban egy kiegészítő programot be kell szerezni, ez a program a Composer. A Composer -ről későbbiekben lesz szó. A Laravel -t legegyszerűbben a Laravel weboldalán megtalálható dokumentáció alapján lehet elkezdeni. Ha le van töltve a Composer, akkor egy tetszőleges helyen kell nyitni egy parancssort vagy Git Bash -t, majd az alábbi kódot kell bemásolni vagy beírni.



Hogyha ezt a kódsort futtatjuk, akkor a jelenleg megnyitott mappába egy „*example-app*” nevezetű projektet fog létrehozni, ez a folyamat számítógéptől függően változó ideig tarthat.  
  
Ha a projekt létre lett hozva, akkor valahogy így fog kinézni a projekt mappanézete, leszámítva pár filet, amit utólag lett hozzáadva. A Laravel tartalmaz egy php szervert is, ezt a „*php artisan serve*” parancsot a terminálba kell beírni és így lehet elindítani.

Hogyha el van indítva a szerver, akkor ez a weboldal fog megjelenni:



Ez az előre legenerált főoldal, amin pár Laraveles weboldal található meg és lehet támogatni a Laravel kitalálóját Taylor Otwell -t.

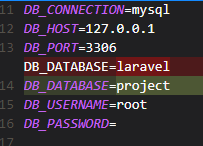
## Mi az a Composer?

A Composer egy népszerű függőségkezelő eszköz a PHP programozási nyelvhez. A fő célja, hogy segítsen a PHP-alapú projektjeinkben az egyes könyvtárak és csomagok kezelésében, valamint ezek függőségeinek hatékony kezelésében. Ez az eszköz lehetővé teszi a fejlesztők számára, hogy egyszerűen telepítsék, frissítsék, és kezeljék a különböző külső könyvtárakat, csomagokat, és azok verzióit a projektjükben. A Composer egy JSON-alapú konfigurációs fájlt használ, amely tartalmazza a projekt függőségeit, valamint azok verzióit és kapcsolódó beállításokat. Az eszköz automatizálja a külső függőségek letöltését az internetről és azok integrálását a projektbe. Például a Composer segít PHP keretrendszerek, könyvtárak vagy egyedi csomagok telepítésében a Laravel, Symfony, vagy más PHP-alapú projektek számára. Emellett gondoskodik a verziók kezeléséről és az összes függőség feloldásáról, hozzáogy biztosítsa a megfelelő működést a projektben. A Composer jelentős mértékben megkönnyíti a fejlesztők számára az alkalmazások fejlesztését és karbantartását, valamint lehetővé teszi a könnyű és hatékony együttműködést az nyílt forráskódú PHP projektek között.

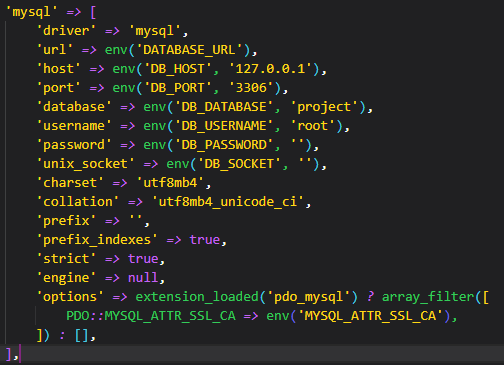
# Laravel Backend

## Új adatbázis csatlakozás

Mivel a Laravel keretrendszer hozzá lett adva a projekthez, ezért új csatlakozást kell létrehozni.



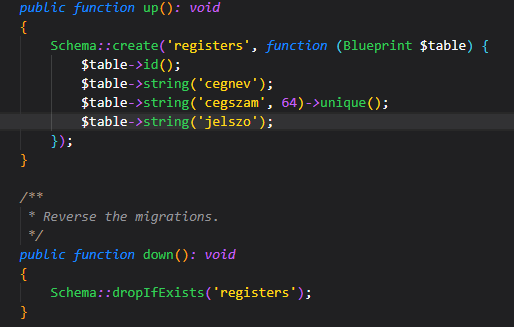
Ahhoz, hogy hogy hozzácsatoljuk, az „*.env*” fileban át kell írni a „*DB\_DATABASE*” értékét az alap „*laravel*” -ről a mi adatbázis nevünkre, ami ebben az esetben „*project*”. Emellett még a „*config/database.php*” fileban is át kell írni az adatbázis nevét és felhasználó nevét a helyesre.



Miután ezek a konfigurációs fileok át lettek írva, egy migrációt kell létrehozni, ezt a következő paranccsal lehet:



Ez létrehoz egy nyers migrációs filet, ebben meg kell adni a tábla oszlopok neveit és egyéb tulajdonságait.



Ha be lett állítva a migrációs file, akkor a következő paranccsal lehet létrehozni a táblát:



Ez a rész hibára tud futni, mivel a php tele van kiegészítőkkel, amit manuálisan kell aktiválni, ebben az esetben a „*extension=pdo*” -t kell aktiválni.

## MVC – Model View Controller

Az MVC egy tervezési minta, amelyet a szoftvertervezésben alkalmaznak a kód strukturálására és szervezésére. Az MVC elkülöníti egy alkalmazás három fő részét, hogy könnyebbé tegye a fejlesztést és karbantartást. Ezek a három fő rész:

### Model (Modell)

A modell felelős az alkalmazás üzleti logikájáért és adatmanipulációjáért. Itt találhatók az adatstruktúrák, adatbázis-interakciók és más olyan komponensek, amelyek közvetlenül kapcsolódnak az alkalmazás adattárolásához és feldolgozásához. A modellek kezelik az adatok integritását és logikáját.

### View (Nézet)

A nézet felelős az adatok megjelenítéséért és az interfész kialakításáért. Ez a rész teszi lehetővé, hogy az alkalmazás a felhasználóval kommunikáljon. A nézetek olyan elemeket tartalmaznak, mint az HTML-sablonok, amelyekkel az adatokat megjelenítik a felhasználó számára.

### Controller (Vezérlő)

A vezérlő kezeli a felhasználói bemeneteket és szabályozza a kommunikációt a modell és a nézet között. Amikor a felhasználó valamilyen műveletet hajt végre (például gombnyomás vagy űrlap beküldése), a vezérlő kezeli az eseményt, módosítja a modellt (szükség esetén) és frissíti a nézetet.

## Hitelesítés

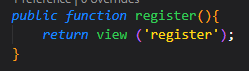
Ezen a weboldalon a cégeknek kell csak regisztrálniuk, abban az esetben, ha posztolni akarnak. A Laravelben létezik az „*php artisan make:auth*” parancs, ami előre létrehozza a hitelesítéseket a megadott modellre, de a projekt esetébe nem volt használva.

### Regisztáció logika

Ahhoz, hogy a regisztrációhoz hozzáférjen a felhasználó, létre kell hozni egy controllert. Ezt az alábbi kóddal lehet létrehozni:



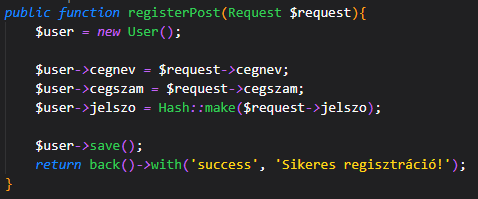
Ebben a fileban eleinte csak egy osztály található.



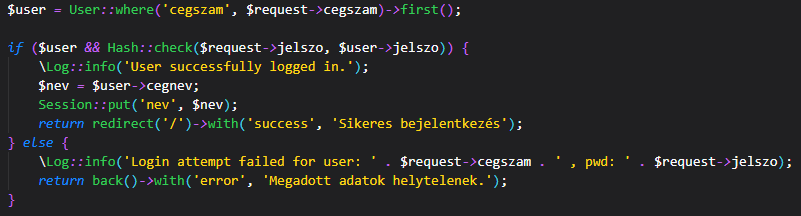
A fenti funkcióval el kell látni, mert ezt fogja meghívni amikor át akarunk menni a „*/register*” oldalra. Ahhoz, hogy működjön, a „*/routes/web.php*” -ban meg kell hívni:



Hogyha megkapja a „post” -ot a szerver, akkor egy másik funkciót küld

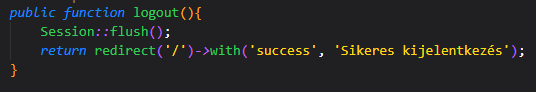


### Bejelentkezés logika



A bejelentkezési logika lényege, hogy a „*user*” model alapján létrehoz egy új usert, ez megkapja a megadott cégszámot, majd ellenőrzi, hogy a usernek megadott jelszó az megegyezik e a lekért jelszóval, ha megegyezik, akkor egy változónak átadja a nevet, amit majd késöbb a „*session*” fog használni, és visszairányítja a felhasználót a főoldalra.

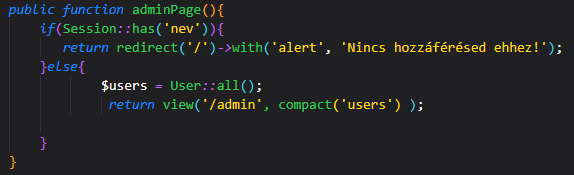
### Kijelentkezés logika

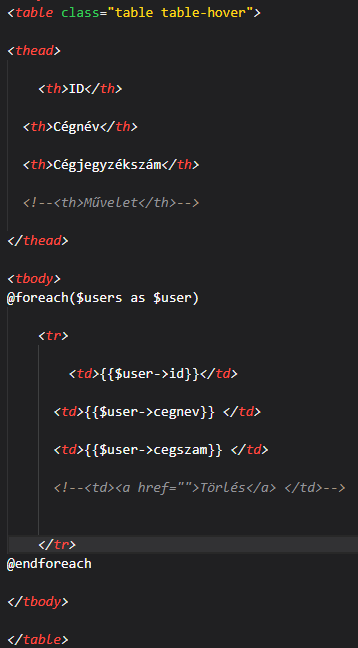


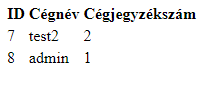
A kijelentkezés szimplán csak törli a törli a munkamenetet.

## Admin felület

Az admin felület létszükséges egy ilyen projekt során, ezért itt is implementálásra kerül.



Első sörban minden regisztrált felhasználó kiírásra kerül, ehhez meg kell hívni a „*user*” modellt.



## Hozzáférési jogok

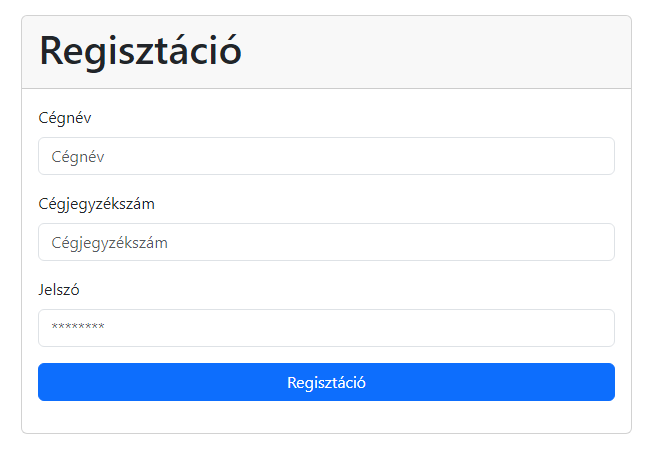
Minden weboldalhoz, amihez az általános felhasználó (azaz a nem cégek és admin) nem férhet hozzá, ahhoz egy vizsgálatot kell rakni.



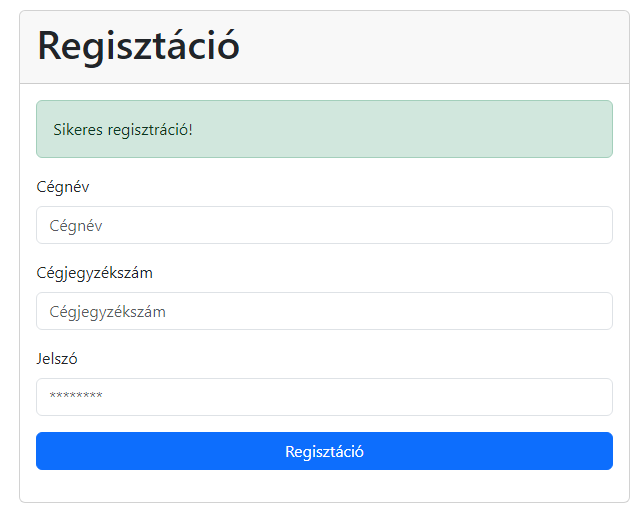
Itt azt vizsgálja a program, hogy a munkamenetnek van e „*nev*” vagy „*admin*” változója, ha van akkor visszavezeti a főoldalra a felhasználót.

# Laravel Frontend

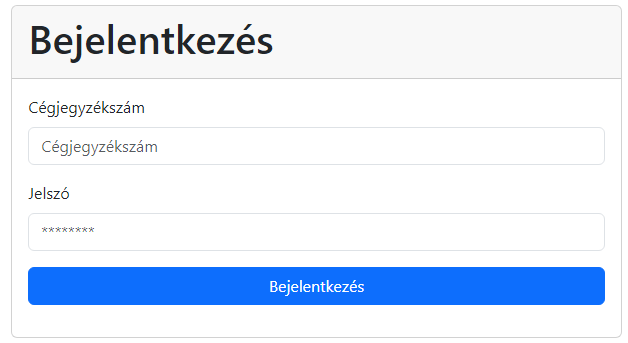
## Regisztráció nézet



A regisztrációs oldal egy egyszerű formból áll. Nevet, egy cégszámot és egy jelszót kér, majd ezeket meg az id -t, az időt amikor létre lett hozva és az időt amikor legutóbb módosítva lett hozzáadja az adatbázishoz, ha sikeres a regisztráció akkor egy zöld figyelmeztetés feljön a „cégnév” felett.



## Bejelentkezés nézet



A bejelentkezésnek is egyszerű kinézete van tesztelési indokokból. Itt ugyan úgy egy formot használ, majd ennek az adatait kapja meg a backend.

## Admin oldal

Todo

## Profil oldal

Todo

# Források

Korai register form logic: <https://www.youtube.com/watch?v=2MpZwFoBPjQ>

Korai php rest api: <https://www.youtube.com/watch?v=X51KOJKrofU>

Http válasz hiba kódok: [https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#server\_error\_responses](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status%23server_error_responses)

Laravel tutorial: <https://www.youtube.com/watch?v=ImtZ5yENzgE>