Regisztráció, bejelentkezés

**Regisztráció**

Az űrlap az elküldése után a CreationRequest osztály segítségével ellenőrizve vannak az adatok.

public function rules()

    {

        return [

            'Username' => 'required|unique:users|min:6|max:20|alpha\_dash',

            'EmailAddress' => 'required|unique:users|email',

            'PwdHash' => 'required|min:8|max:24|confirmed',

            'PwdHash\_confirmation' => 'required',

            'Role' => 'integer|min:0|max:2'

        ];

    }

Amennyiben a beírt adatszerkezetben hiba található, azt jelezzük a felhasználónak. Minden mező esetében a következő módszerrel jártunk el:

@if ($errors->has('Username'))

    <span class="text-danger text-left">{{ $errors->first('Username') }} </span>

@endif

A $errors változót a Illuminate\View\Middleware\ShareErrorsFromSession middleware teszi elérhetővé az összes .blade.php fájlban.

**Bejelentkezés**

A bejelentkezés menüpontban a felhasználónak meg kell adnia a felhasználónevét és beállított jelszavát. Az esetlegesen elfelejtett jelszó esetében egy jelszó visszaállítási link is be van ágyazva, melynek segítségével a hitelesített email címre a felhasználó kap egy levelet, amelyben található linkre kattintva új jelszót tud létrehozni. A bejelentkezésnél a program megkeresi, hogy létezik-e az adott nevű felhasználó a mentett adatbázisban és a jelszó titkosítása után keres egyezést a jelszó mezőnél. Hiba esetén természetesen értesítjük a felhasználót, hogy a felhasználónév vagy a jelszó helytelen. Ha a program talált az adatbázisban egyezést az adottfelhasználóra akkor ellenőrzi, hogy hitelesítve van-e a felhasználó email címe. Ha igen, tovább engedi őt a profil oldalra, ahol a felhasználó tudja kezelni személyes adatait (pl.: jelszó, felhasználónév, profilkép szerkesztése). Abban az esetben, ha a felhasználónak nincs megerősítve az email címe, nem engedi tovább a program, megragad egy köztes oldalon, ami kéri hogy az emailjei közt keresse meg a kapott emailt és az abban található linkre kattintva hitelesítse email címét. Amennyiben nem találja azt a levelet, tud önmagának új levelet küldeni egy gombra kattintva.

**CSRF védelem**

Az oldalon használt összes <form>, ami POST kérést használ CSRF-el van védve. A <form>-ba beírt kódrészlet az App\Http\Middleware\VerifyCsrfToken middleware működéséhez szükséges. Ez alapértelmezés szerint a webes middleware csoportban található és automatikusan ellenőrzi, hogy a kérelem bemenetében lévő token megegyezik-e a munkamenetben tárolt tokennel. Ha ez a két token egyezik, akkor tudjuk, hogy a hitelesített felhasználó az, aki a kérést kezdeményezi. A CSRF token megakadályozza a Cross-Site támadást a cookie token és a szerver token összehasonlításával. A CSRF middleware és template tag (@csrf) védelmet nyújt a Cross Site Request Forgeries ellen. Ez a fajta támadás akkor következik be, amikor egy rosszindulatú weboldal tartalmaz egy linket, egy űrlapgombot vagy valamilyen JavaScript-et, amelynek célja, hogy valamilyen műveletet hajtson végre a mi weboldalunkon, a bejelentkezett felhasználó hitelesítő adatainak felhasználásával, aki a rosszindulatú weboldalra látogat a böngészőjében. A támadás becsapja a felhasználó böngészőjét, hogy valaki más hitelesítő adataival jelentkezzen be egy webhelyre.

Példa:

<form action="/register" method="post">

    @csrf

    <input type="hidden" name="\_token" value="{{ csrf\_token() }}" />

**Felhasználó jogosultság**

Az adatbázisban egy új mezőt kellett létrehozni, amit *Role*-nak hívunk. Itt alapértelmezetten minden regisztrált felhasználó 0-ás értéket kap. A 0 a sima felhasználó jogait reprezentálja. Ezt a mezőt az adminisztrátorok illetve a tulajdonosok tudják változtatni. Az adminisztrátorok az 1-es, a tulajdonosok pedig a 2-es értéket kapják a Role mezőben. Az extra jogokkal rendelkező felhasználók plusz menüpontokat és szerkesztési lehetőségeket kapnak az weboldalon.

**Források**

<https://laravel.com/docs/10.x/csrf>

<https://laravel.com/docs/10.x/validation>

Bólya Gábor tanár úr óráin tanultak