

**Kingsconsult**

Posted on Dec 11, 2020 • Updated on Dec 17, 2020



Customize Laravel Auth (Laravel Breeze Registration and Login)

#php #laravel #beginners #webdev

Laravel Auth (4 Part Series)

- 1 Laravel 8 Auth (Registration and Login)
- 2 Basic Laravel Login and Registration using Laravel Breeze
- 3 **Customize Laravel Auth (Laravel Breeze Registration and Login)**
- 4 Customize Laravel Jetstream (Registration and Login)

Hello Artisans, after making a post on [Basic Laravel Login and Registration using Laravel Breeze](#), I got a lot of private messages on how to customize the Registration and Login where a developer can be able to add more fields because the default Laravel User Registration fields are just limited to **Name**, **Email** and **Password** right from the time of the Legacy UI till the time of writing this report. But there are instances of where a developer might want to collect not just the name, but **First Name**, **Last Name**, **Username**, **Password** or even making the **Email** not mandatory for users who don't have email, so I decide to write this little piece.

For this article, I am going to use the app from this article [Basic Laravel Login and Registration using Laravel Breeze](#)

Click on my [profile](#) to follow me to get more updates.

Step 1: Edit the RegisteredUserController

Go to `app/Http/Controllers/Auth/RegisteredUserController.php` and customize the **store** method to this

```
public function store(Request $request)
{
    $request->validate([
        'firstname' => 'required|string|max:255',
        'lastname' => 'required|string|max:255',
        'password' => 'required|string|confirmed|min:8',
        'username' => 'required|string|max:255',
    ]);

    Auth::login($user = User::create([
        'firstname' => $request->firstname,
        'lastname' => $request->lastname,
        'username' => $request->username,
        'email' => $request->email,
        'password' => Hash::make($request->password),
    ]));

    event(new Registered($user));

    return redirect(RouteServiceProvider::HOME);
}
```

In the **validate()** method, I removed **email**, and added **firstname**, **lastname** and **username**. Also in the **User::create**, I added the new fields.

Step 2: Create another migration file to add the new fields to the User and login_history table

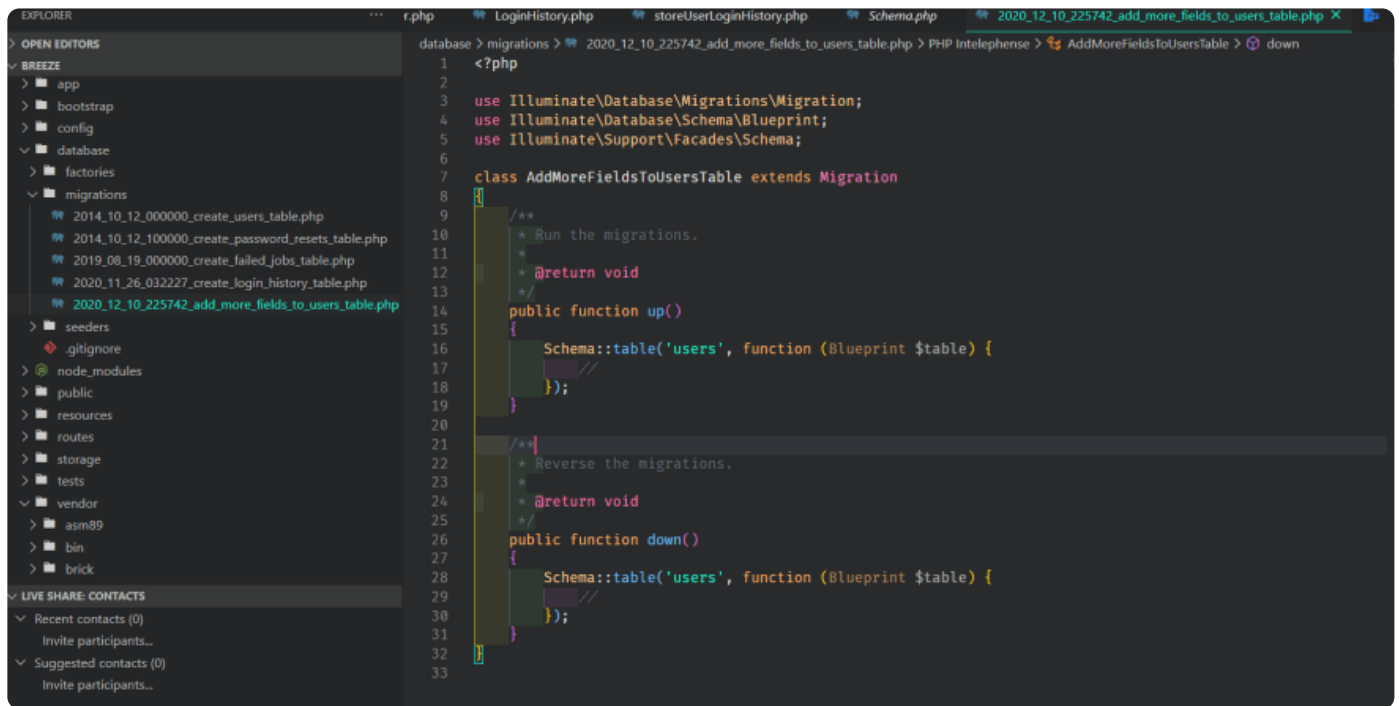
This migration field will add the new fields to our existing **users** table, so run the command below

```
php artisan make:migration add_more_fields_to_users_table --table=users
```

Do the same for the login_history table.

The **--table=users** flag will specify the type of schema facade to use.

A migrations file will be created, go to **database/migrations/**



Step 3: Add the new fields and modify the name field

We are going to modify the existing name field with firstname and also add other fields, but to modify an existing fields, we need a package **doctrine/dbal**, The Doctrine DBAL library is used to determine the current state of the column and to create the SQL queries needed to make the requested changes to your column. [Laravel docs](https://www.doctrine-project.org/en/latest/faq.html)

```
composer require doctrine/dbal
```

After this, update the composer, by running the command below

```
composer update
```

```

KINGS@DESKTOP-LCR3CE4 MINGW64 ~/Documents/dev/laravel_8/breeze
$ composer require doctrine/dbal
Using version ^3.0 for doctrine/dbal
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 4 installs, 0 updates, 0 removals
 - Installing composer/package-versions-deprecated (1.11.99.1): Loading from cache
 - Installing doctrine/event-manager (1.1.1): Loading from cache
 - Installing doctrine/cache (1.10.2): Loading from cache
 - Installing doctrine/dbal (3.0.0): Loading from cache
doctrine/cache suggests installing alcaeus/mongo-php-adapter (Required to use legacy MongoDB driver)
Writing lock file
Generating optimized autoload files
composer/package-versions-deprecated: Generating version class...
composer/package-versions-deprecated: ...done generating version class
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
75 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

KINGS@DESKTOP-LCR3CE4 MINGW64 ~/Documents/dev/laravel_8/breeze
$

KINGS@DESKTOP-LCR3CE4 MINGW64 ~/Documents/dev/laravel_8/breeze
$ composer update
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 0 installs, 26 updates, 0 removals
 - Updating symfony/var-dumper (v5.1.8 => v5.2.0): Loading from cache
 - Updating symfony/string (v5.1.8 => v5.2.0): Loading from cache
 - Updating symfony/console (v5.1.8 => v5.2.0): Loading from cache
 - Updating symfony/css-selector (v5.1.8 => v5.2.0): Loading from cache
 - Updating symfony/routing (v5.1.8 => v5.2.0): Loading from cache
 - Updating symfony/process (v5.1.8 => v5.2.0): Loading from cache
 - Updating symfony/mime (v5.1.8 => v5.2.0): Loading from cache

```

Then add the following fields to the **up()** method of the migration file created above

```

public function up()
{
    Schema::table('users', function (Blueprint $table) {
        $table->renameColumn('name', 'firstname');
        $table->string('lastname');
        $table->string('username');
        $table->string('email')->nullable()->change();
    });
}

```

You will notice how we are going to rename our name field to firstname and added lastname and username, also we modify file email from mandatory to nullable().

Don't forget to do the same for the login_history.

Finally, run the migration command, but

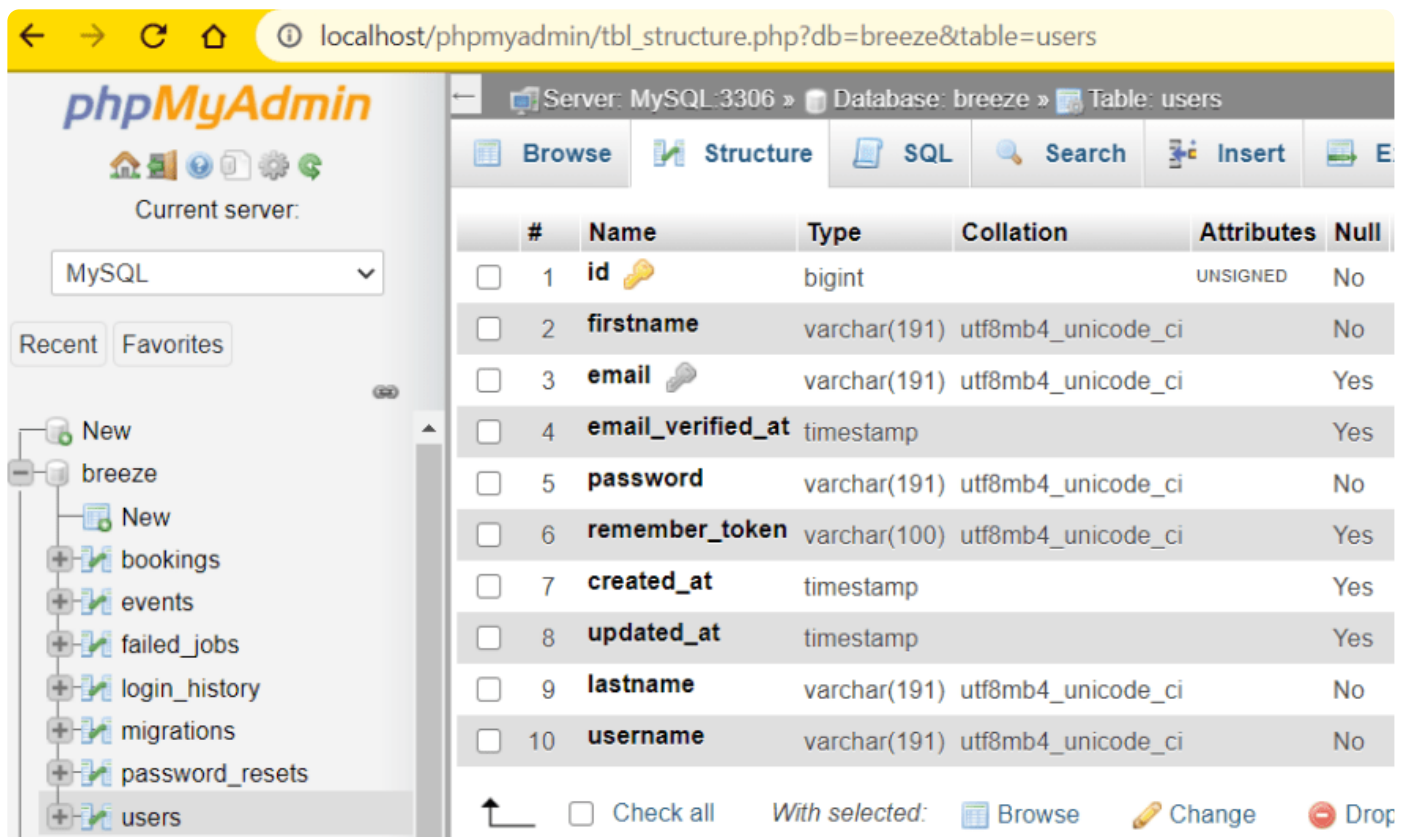
```
php artisan migrate
```

```
Discovered Package: laravel/tinker
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
77 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

KINGS@DESKTOP-LCR3CE4 MINGW64 ~/Documents/dev/laravel_8/breeze
$ php artisan migrate
Migrating: 2020_12_10_225742_add_more_fields_to_users_table
Migrated: 2020_12_10_225742_add_more_fields_to_users_table (202.35ms)

KINGS@DESKTOP-LCR3CE4 MINGW64 ~/Documents/dev/laravel_8/breeze
$
```

Database



The screenshot shows the phpMyAdmin interface. The left sidebar displays the database structure, with the 'breeze' database selected and the 'users' table highlighted. The main panel shows the 'Structure' tab for the 'users' table. The table has 10 columns with the following details:

#	Name	Type	Collation	Attributes	Null
1	id	bigint		UNSIGNED	No
2	firstname	varchar(191)	utf8mb4_unicode_ci		No
3	email	varchar(191)	utf8mb4_unicode_ci		Yes
4	email_verified_at	timestamp			Yes
5	password	varchar(191)	utf8mb4_unicode_ci		No
6	remember_token	varchar(100)	utf8mb4_unicode_ci		Yes
7	created_at	timestamp			Yes
8	updated_at	timestamp			Yes
9	lastname	varchar(191)	utf8mb4_unicode_ci		No
10	username	varchar(191)	utf8mb4_unicode_ci		No

Step 4: Modify the Users Model

We need to add the fields to be fill in users model, go to **app/Models/User.php** and edit the **protected \$fillable** to add the new fields

```
protected $fillable = [
    'firstname',
    'email',
    'password',
    'username',
    'lastname'
];
```

Step 5: Modify the Registration view

Go to **resources/views/auth/register.blade.php** and modify to this

```
<x-guest-layout>
    <x-auth-card>
        <x-slot name="logo">
            <a href="/">
                <x-application-logo class="w-20 h-20 fill-current text-gray-500"/>
            </a>
        </x-slot>

        <!-- Validation Errors -->
        <x-auth-validation-errors class="mb-4" :errors="$errors" />

        <form method="POST" action="{{ route('register') }}">
            @csrf

            <!-- Name -->
            <div>
                <x-label for="firstname" :value="__('First Name')"/>

                <x-input id="firstname" class="block mt-1 w-full" type="text" name="firstname" />
            </div>
            <!-- Name -->
            <div>
                <x-label for="lastname" :value="__('Last Name')"/>

                <x-input id="lastname" class="block mt-1 w-full" type="text" name="lastname" />
            </div>
            <!-- Name -->
            <div>
                <x-label for="username" :value="__('username')"/>

                <x-input id="username" class="block mt-1 w-full" type="text" name="username" />
            </div>
```

```

<!-- Email Address -->
<div class="mt-4">
    <x-label for="email" :value="__('Email')" />

    <x-input id="email" class="block mt-1 w-full" type="email" name="email" />
</div>

<!-- Password -->
<div class="mt-4">
    <x-label for="password" :value="__('Password')" />

    <x-input id="password" class="block mt-1 w-full" type="password" name="password" />
</div>

<!-- Confirm Password -->
<div class="mt-4">
    <x-label for="password_confirmation" :value="__('Confirm Password')" />

    <x-input id="password_confirmation" class="block mt-1 w-full" type="password" name="password_confirmation" />
</div>

<div class="flex items-center justify-end mt-4">
    <x-button>
        {{ __('Register') }}
    </x-button>
</div>
</form>
</x-auth-card>
</x-guest-layout>

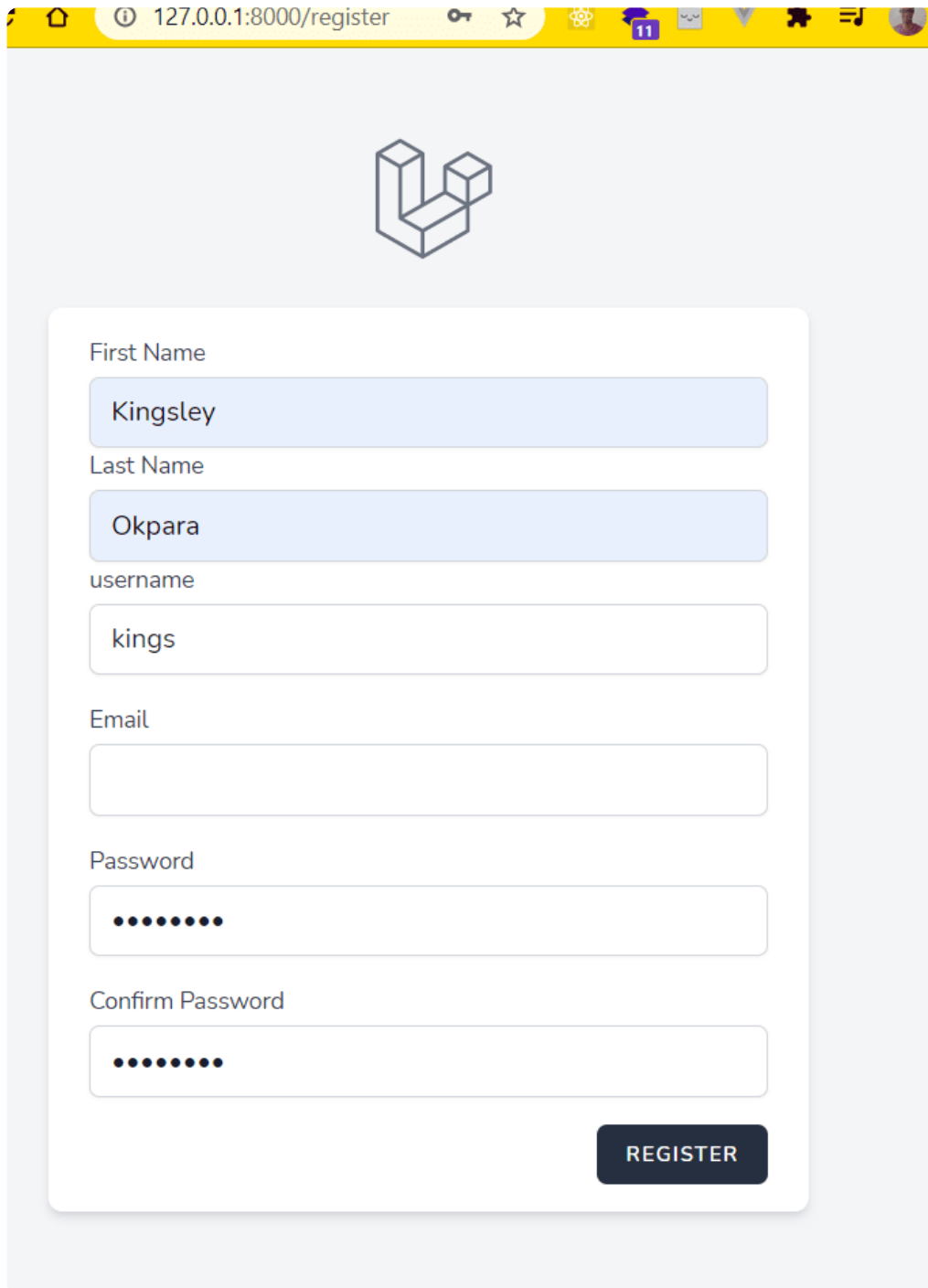
```



Start our application with

`php artisan serve`

Click on Register, we are going to have this



127.0.0.1:8000/register

First Name

Kingsley

Last Name

Okpara

username

kings

Email

Password

Confirm Password

REGISTER

Step 6: Modify the Login view

Go to **resources/views/auth/login.blade.php** and modify to this

```
<x-guest-layout>
  <x-auth-card>
    <x-slot name="logo">
      <a href="/">
        <x-application-logo class="w-20 h-20 fill-current text-gray-500"/>
      </a>
    </x-slot>

    <!-- Session Status -->
    <x-auth-session-status class="mb-4" :status="session('status')"/>
```



```

<!-- Validation Errors -->
<x-auth-validation-errors class="mb-4" :errors="$errors" />

<form method="POST" action="{{ route('login') }}">
    @csrf

    <!-- Email Address -->
    <div>
        <x-label for="username" :value="__('Username')" />

        <x-input id="username" class="block mt-1 w-full" type="text" name="username" />
    </div>

    <!-- Password -->
    <div class="mt-4">
        <x-label for="password" :value="__('Password')" />

        <x-input id="password" class="block mt-1 w-full" type="password" name="password" />
    </div>

    <!-- Remember Me -->
    <div class="block mt-4">
        <label for="remember_me" class="flex items-center">
            <input id="remember_me" type="checkbox" class="form-checkbox" />
            <span class="ml-2 text-sm text-gray-600">{{ __('Remember me') }}
        </label>
    </div>

    <div class="flex items-center justify-end mt-4">
        @if (Route::has('password.request'))
        <a class="underline text-sm text-gray-600 hover:text-gray-900" href="{{ route('password.request') }}">
            {{ __('Forgot your password?') }}
        </a>
        @endif

        <x-button class="ml-3">
            {{ __('Login') }}
        </x-button>
    </div>
</form>
</x-auth-card>
</x-guest-layout>

```

This will change the login field from email to username

Step 7: Modify the Login Request Class

This is the class that replaces the **LoginController** class in the previous Laravel Authentication. Go to **app/Requests/Auth/LoginRequest.php** and change all the email string to username string, and also make sure the validating rules, you remove the email rule and replace it with username rule

```
<?php
```

```
namespace App\Http\Requests\Auth;
```

```
use Illuminate\Auth\Events\Lockout;
use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\RateLimiter;
use Illuminate\Support\Str;
use Illuminate\Validation\ValidationException;
use App\Events>LoginHistory;
```

```
class LoginRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'username' => 'required|string',
            'password' => 'required|string',
        ];
    }

    /**
     * Attempt to authenticate the request's credentials.
     *

```

```
* @return void
*
* @throws \Illuminate\Validation\ValidationException
*/
public function authenticate()
{
    $this->ensureIsNotRateLimited();

    if (! Auth::attempt($this->only('username', 'password'), $this->filled(
        RateLimiter::hit($this->throttleKey());

        throw ValidationException::withMessages([
            'username' => __('auth.failed'),
        ]));
    }

    $user = Auth::user();

    event(new LoginHistory($user));

    RateLimiter::clear($this->throttleKey());
}

/**
 * Ensure the login request is not rate limited.
 *
 * @return void
 *
 * @throws \Illuminate\Validation\ValidationException
 */
public function ensureIsNotRateLimited()
{
    if (! RateLimiter::tooManyAttempts($this->throttleKey(), 5)) {
        return;
    }

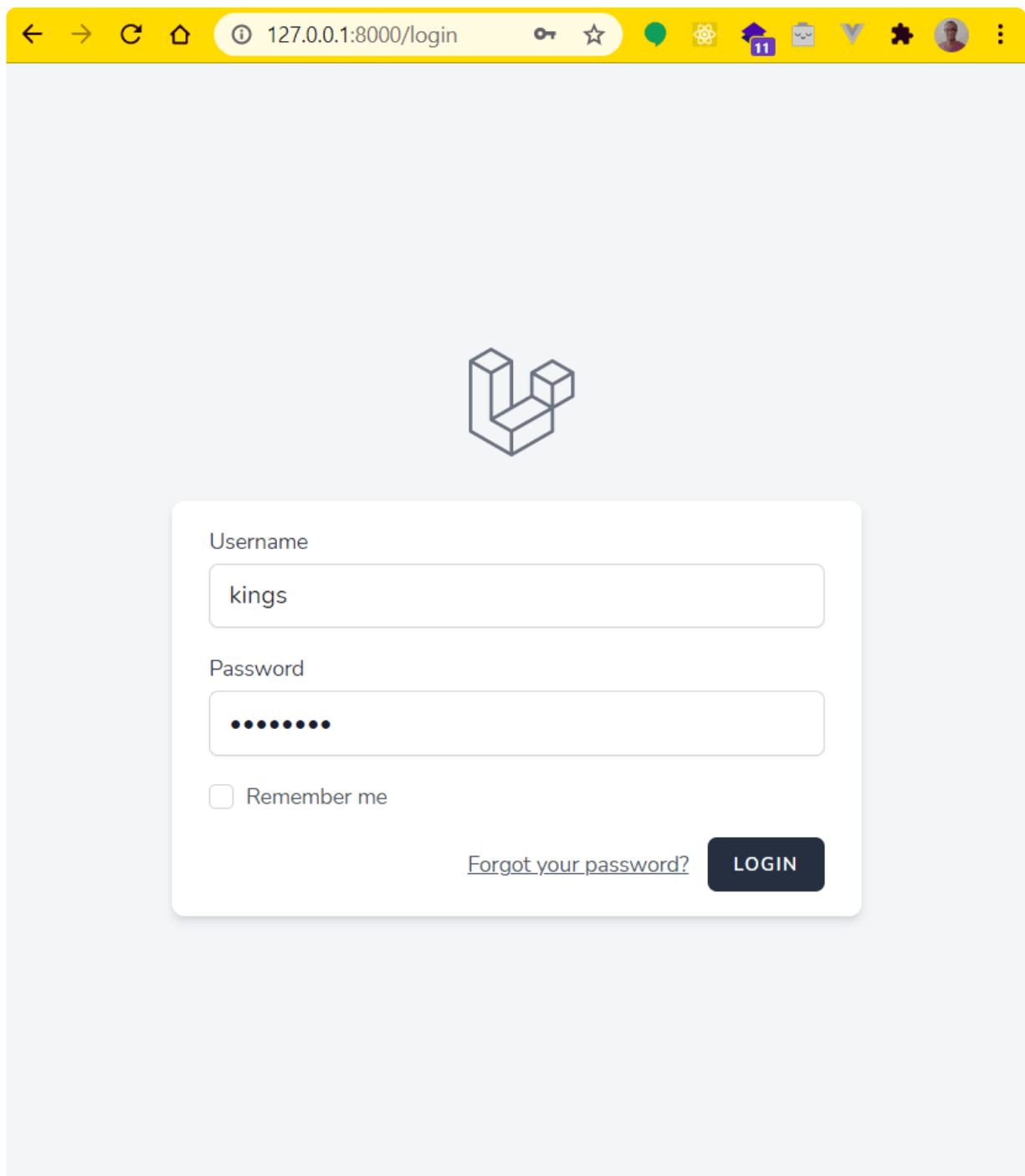
    event(new Lockout($this));

    $seconds = RateLimiter::availableIn($this->throttleKey());


    throw ValidationException::withMessages([
        'username' => trans('auth.throttle', [
            'seconds' => $seconds,
            'minutes' => ceil($seconds / 60),
        ]),
    ]);
}
```

```
/**
 * Get the rate limiting throttle key for the request.
 *
 * @return string
 */
public function throttleKey()
{
    return Str::lower($this->input('username')).'|'.$this->ip();
}
}
```

Logout from the app and click on login



← → ↻ 🏠 ⓘ 127.0.0.1:8000/login 🔑 ☆ 🗨 ⚙ 🏠 11 📅 📌 ⚙ 👤 ⋮



Username

kings

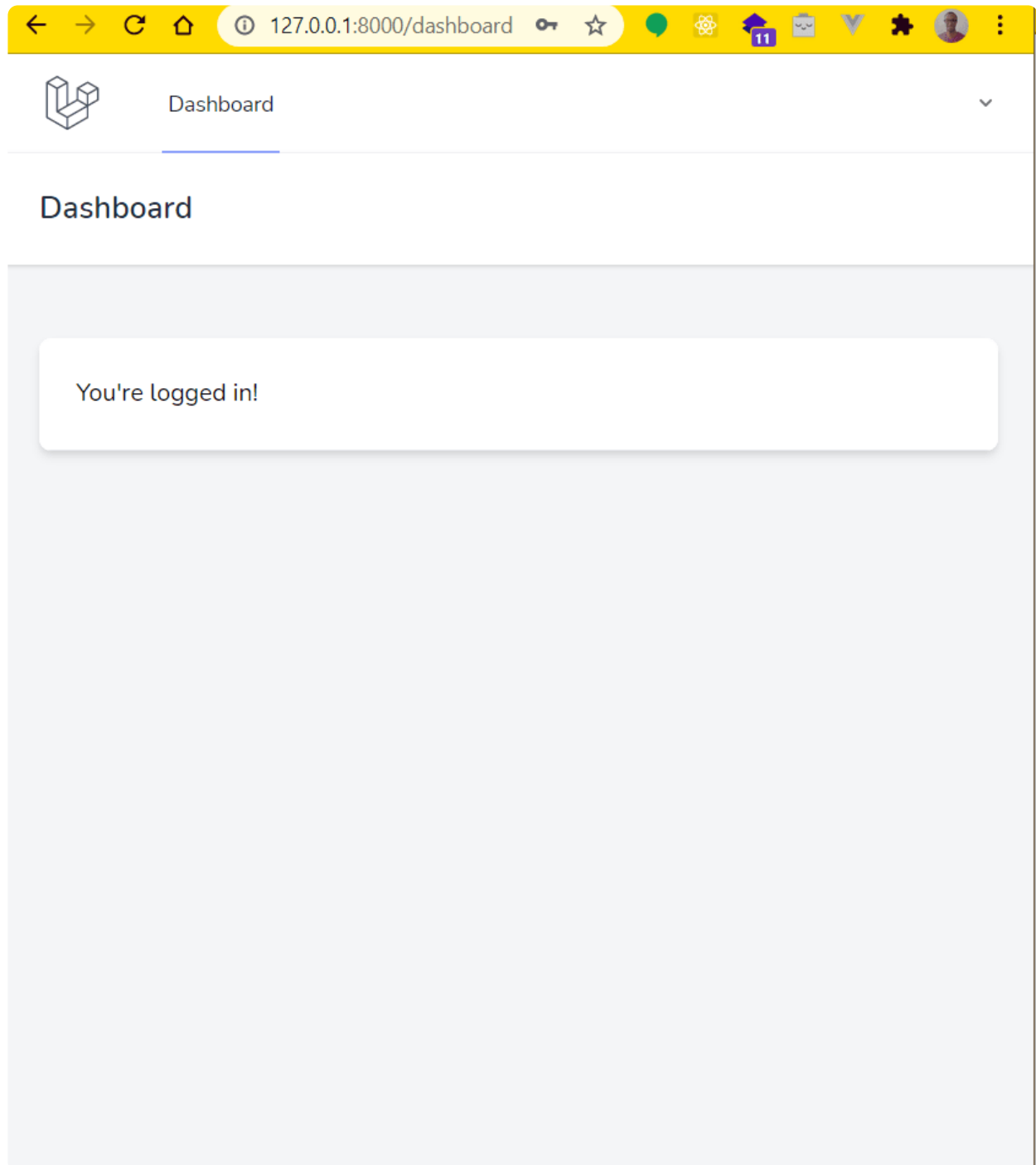
Password

●●●●●●●●

☐ Remember me

[Forgot your password?](#)

LOGIN



We can now login with username and password instead of email address. That is all
Follow me for more of my articles, you can leave comments, suggestions, and reactions.

I am open to any vacancy as a PHP backend engineer, my strength is in the Laravel framework

[click the link to view my profile and follow me](#)

Laravel Auth (4 Part Series)

- 1 Laravel 8 Auth (Registration and Login)
- 2 Basic Laravel Login and Registration using Laravel Breeze
- 3 **Customize Laravel Auth (Laravel Breeze Registration and Login)**
- 4 Customize Laravel Jetstream (Registration and Login)

Top comments (14) ↕



DreamHigh · Apr 5 '21



Hi. king.
Now, I'm trying to use jQuery ajax method to send requests to the controller.
But, I've faced such an error.

Target [Laravel\Fortify\Contracts\RegisterViewResponse] is not instantiable.

Hope your kind help.



Kingsconsult 🌟 · Apr 7 '21



Sorry, seeing this late, would be easier for me to debug if I see the way you are sending the request to the controller



DreamHigh · May 30 '21



Thank you. King.
I've solved it myself. :D
Thanks for your kindness reply again.



Federico Reinoso · Aug 5 '21



I know, maybe this is out of the scope of the tutorial, but, how you can add Roles to the register?
I just need to deny access to certain part of my site to certain role, (come can edit and some can only see)

It is possible?, or is better to go directly to Jetstream?
Thanks in advance.



Marcelo Bianco • Apr 22 '21



Hi my friend

I have the following error: `ErrorException`

Undefined index:

my password field is called `senha` and it doesn't pass this validation at all

public function validateCredentials(UserContract \$user, array \$credentials)

```
{  
  
    $plain = $credentials['password'];  
  
  
    return $this->hasher->check($plain, $user->getAuthPassword());  
  
}
```



rahulmanek • Jul 17 '21



I am also facing same issue
Any solution??



Fahrudin Yuniwinanto • Feb 26 '21



why suddenly there is `login_history`? thought in part 1 and 2 there is no tell about `login_history`, make me little bit confuse. please tell me clearly. thanks



ARINZE CHRIS HILLS • Jan 6 '21



please mine doesn't have does controllers, please can you help me out



Kingsconsult  • Jan 6 '21



Hope you run these commands

```
composer require laravel/breeze --dev
php artisan breeze:install
```



Adefowowe Adebamowo • Mar 25 '21 • Edited



Hi Kingsley, really nice series! Taking the customisation further, how might one implement login with either email or mobile phone (in one field) and password. Thanks.



wanazhad24 • Jan 22 '21



Hey wanna ask, in a situation where I'm not using a Users table for the login, where am I supposed to change the referenced table?



Sunil Thatal • Jan 22 '21



you can change it on **config/auth.php** providers configuration.



wanazhad24 • Jan 22 '21 • Edited



Is this correct? Even after I changed that, it is saying that the credentials is invalid.

```
'providers' => [
'staffs' => [
'driver' => 'eloquent',
'model' => App\Models\Staff::class,
'table' => 'staffs'
],
```



Sunil Thatal • Jan 23 '21 • Edited



yes it is correct. you don't have to add table if you are using eloquent. Add query listener above your code to monitor query.

```
\DB::listen(function($query){
\Log::debug($query->sql);
\Log::debug($query->bindings);
});
```

[Code of Conduct](#) • [Report abuse](#)



12 Rarely Used Javascript APIs You Need



Practical examples of some unique Javascript APIs that beautifully demonstrate a practical use-case.



Kingsconsult

I am Kingsley Okpara, a Python and PHP Fullstack Web developer and tech writer, I also have extensive knowledge and experience with JavaScript while working on applications developed with VueJs.

LOCATION

Lagos, Nigeria

EDUCATION

Bsc.ed Mathematics, Enugu State University of Science and Technology, Enugu, Nigeria

WORK

Mid-level Web Developer at Plexada-Si

JOINED

Aug 5, 2019

More from **Kingsconsult**

Laravel Credit Card Validation

#php #laravel #webdev #security

Schedule a task to run at a specific time in laravel (CronJob)

#php #laravel #webdev #aws

Customize Laravel Jetstream (Registration and Login)

#jetstream #laravel #php #webdev

DEV Community

...



[Reset Local Repository Branch to be Just Like Remote Repository HEAD: A Step-by-Step Guide](#)

To get your local branch back in sync with the remote repository HEAD, you'll need to reset it.

[Read full post](#)