Домашняя работа
по предмету:

Продвинутое программирование на PHP — Laravel

Выполнил: Байбородин Александр
Урок 11. Реализация авторизации

Цели практической работы

Научиться:

— интегрировать регистрацию и аутентификацию пользователей;
— разрабатывать механизмы авторизации действий пользователей системы;
— проектировать ролевую модель системы.

Что нужно сделать:

В этой практической работе вы реализуете проект, в котором будут использованы механизмы авторизации и аутентификации пользователей.

1. Создайте новый проект Laravel или откройте уже существующий.

2. Создайте новую ветку вашего репозитория от корневой (main или master).

3. Установите библиотеку Laravel Breeze `composer require laravel/breeze`.

4. Установите файлы библиотеки `php artisan breeze:install`.

5. Соберите фронтенд проекта с помощью команд npm install && npm run dev.

6. Перейдите на ваш сайт и проверьте работу механизмов регистрации и аутентификации.

7. Создайте контроллер UsersController командой `php artisan make:controller UsersController`.

8. Создайте в классе **UsersController** функцию **index**, которая вернёт список всех пользователей системы.

9. Напишите маршрут ‘/users’ в файле web.php.

10. Создайте миграцию, которая добавит поле **is_admin** типа **boolean** в таблицу **users**.

11. Создайте политику `php artisan make:policy UserPolicy --model=User` и напишите функцию.

```
public function viewAny(User $user)
{
return $user->is_admin;
}
```

12. Зарегистрируйте политику в классе **AuthServiceProvider**.

```
protected $policies = [
User::class => UserPolicy::class,
];
```

13. Используйте авторизацию действий пользователя внутри контроллера **UsersController** в функции **index**.

```
$this->authorize('view-any', User::class);
```

14. Создайте двух пользователей, дайте одному из них роль администратора и попробуйте перейти на маршрут '/users' вашего проекта сначала за неаутентифицированного пользователя, а далее за обычного пользователя и администратора системы.

Решение:

1. **Создадим новый проект Laravel**:

```
composer create-project --prefer-dist laravel/laravel project-name
```

   - Перейдем в директорию проекта:

```
cd project-name
```

2. **Установим библиотеку Laravel Breeze**:

```
composer require laravel/breeze --dev
```

```
$ composer require laravel/breeze --dev
Cannot use laravel/breeze's latest version v2.3.6 as it requires php
^8.2.0 which is not satisfied by your platform.
./composer.json has been updated
Running composer update laravel/breeze
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
  - Locking laravel/breeze (v1.29.1)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Downloading laravel/breeze (v1.29.1)
  - Installing laravel/breeze (v1.29.1): Extracting archive
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

  INFO  Discovering packages.

  laravel/breeze
.....................................................................
..................... DONE
```

```
  laravel/sail
.......................................................................
.......................... DONE
  laravel/sanctum
.......................................................................
......................... DONE
  laravel/tinker
.......................................................................
......................... DONE
  nesbot/carbon
.......................................................................
......................... DONE
  nunomaduro/collision
.......................................................................
.................... DONE
  nunomaduro/termwind
.......................................................................
.................... DONE
  spatie/laravel-ignition
.......................................................................
.............. DONE

81 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

   INFO  No publishable resources for tag [laravel-assets].

No security vulnerability advisories found.
Using version ^1.29 for laravel/breeze
```

3. **Установим файлы библиотеки Breeze**:

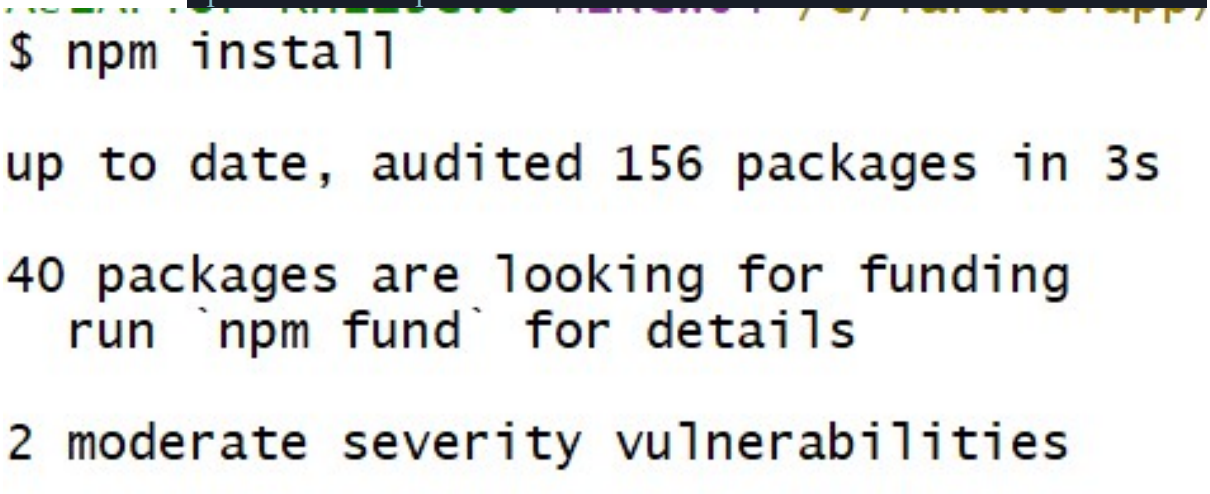
php artisan breeze:install

```
$ php artisan breeze:install

  Which Breeze stack would you like to install?
  Blade with Alpine
.......................................................................
.................................................... blade
  Livewire (Volt Class API) with Alpine
.......................................................................
............................ livewire
  Livewire (Volt Functional API) with Alpine
.......................................................................
.......... livewire-functional
  React with Inertia
.......................................................................
.................................................... react
  Vue with Inertia
.......................................................................
..................................................... vue
  API only
.......................................................................
..................................................... api
❯ blade
blade

  Would you like dark mode support? (yes/no) [no]
❯ no

  Which testing framework do you prefer? [PHPUnit]
  PHPUnit
.......................................................................
.......................................................... 0
  Pest
.......................................................................
.......................................................... 1
```

```
❯ 0
0

  INFO  Installing and building Node dependencies.


added 155 packages, and audited 156 packages in 25s

40 packages are looking for funding
  run `npm fund` for details

2 moderate severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

> build
> vite build

    vite v5.4.15 building for production...
    transforming...
    ✓ 54 modules transformed.
    rendering chunks...
computing gzip size...
public/build/manifest.json              0.27 kB | gzip:  0.14 kB
public/build/assets/app-CM7fOm3G.css   32.23 kB | gzip:  6.09 kB
public/build/assets/app-Bo-u61x1.js    79.58 kB | gzip: 29.62 kB
✓ built in 3.02s


  INFO  Breeze scaffolding installed successfully.
```

4. **Соберем фронтенд**:

```
npm install && npm run dev
```



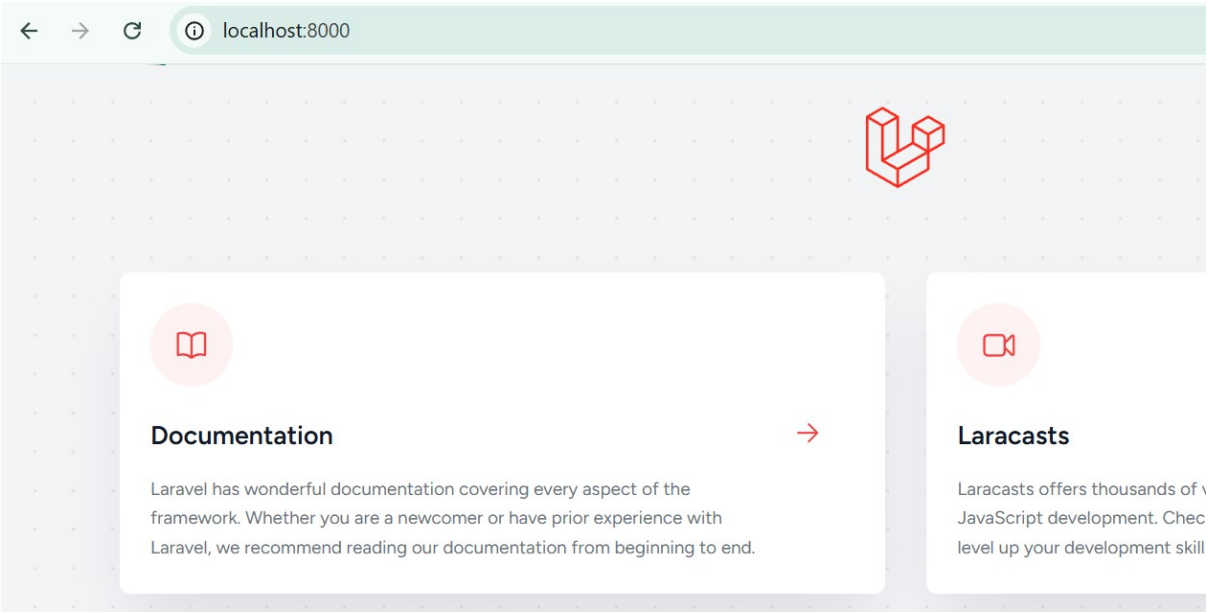

```
$ npm run dev

> dev
> vite


  VITE v5.4.15  ready in 909 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
```

```
  LARAVEL v10.48.29  plugin v1.2.0

  ➜  APP_URL: http://localhost
```

5. **Проверим работу регистрации и аутентификации запустив сервер и проверив работу регистрации и аутентификации**:

```
php artisan serve
```

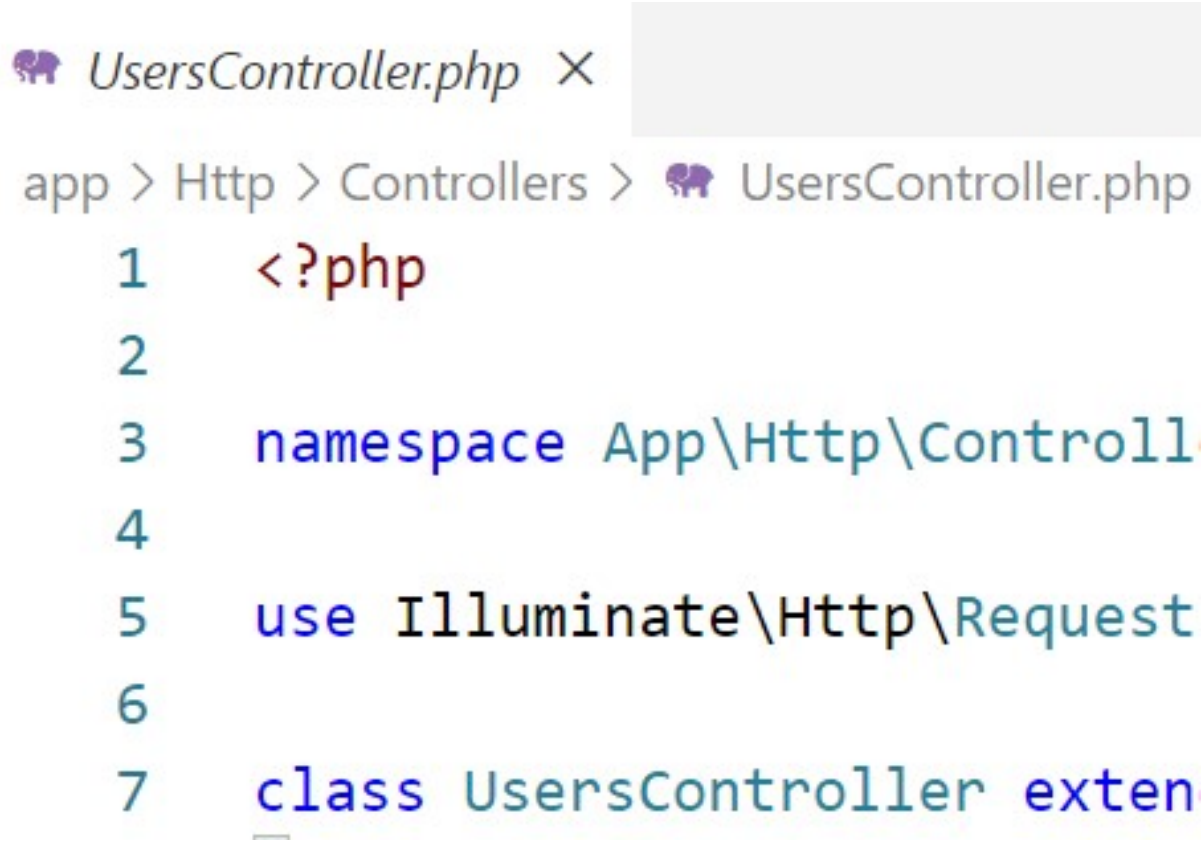
```
A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp/lesson11/project-name
$ php artisan serve

  INFO  Server running on [http://127.0.0.1:8000].

  Press Ctrl+C to stop the server
```



6. **Созданим контроллер UsersController**:

```
php artisan make:controller UsersController
```

```
$ php artisan make:controller UsersController

  INFO  Controller [C:\laravelapp\lesson11\project-
name\app\Http\Controllers\Us
ersController.php] created successfully.
```



7. **Создадим функцию index в UsersController**:
    - Откроем файл app/Http/Controllers/UsersController.php и добавим функцию:

```
public function index()
{
    $users = User::all();
    return view('users.index', compact('users'));
}
```

```
<?php

namespace App\Http\Controllers;
use App\Models\User;
use Illuminate\Http\Request;

class UsersController extends Controller
{
    public function index()
        {
            $this->authorize('viewAny', User::class);
            $users = User::all();
            return view('users.index', compact('users'));
        }

}
```

8. **Создадим маршрут /users:**
    - Откроем файл routes/web.php и добавим маршрут:

```
use App\Http\Controllers\UsersController;

Route::get('/users', [UsersController::class, 'index'])->middleware('auth');
```

```
<?php

use App\Http\Controllers\ProfileController;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\UsersController;

Route::get('/', function () {
    return view('welcome');
});

Route::get('/dashboard', function () {
    return view('dashboard');
})->middleware(['auth', 'verified'])->name('dashboard');

Route::middleware('auth')->group(function () {
    Route::get('/profile', [ProfileController::class, 'edit'])-
>name('profile.edit');
    Route::patch('/profile', [ProfileController::class, 'update'])-
>name('profile.update');
    Route::delete('/profile', [ProfileController::class,
'destroy'])->name('profile.destroy');
});

Route::get('/users', [UsersController::class, 'index'])-
>middleware('auth');

require __DIR__.'/auth.php';
```

9. **Создадим миграцию для добавления поля is_admin**:
    - Создадим миграцию:

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

```
A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp/lesson11/proje
$ php artisan make:migration add_is_admin_to_users_tab

  INFO  Migration [C:\laravelapp\lesson11\project-nam
```

    - Откроем созданный файл миграции и добавим поле:

```
public function up()
{
    Schema::table('users', function (Blueprint $table) {
        $table->boolean('is_admin')->default(false);
    });
```

```
}
```

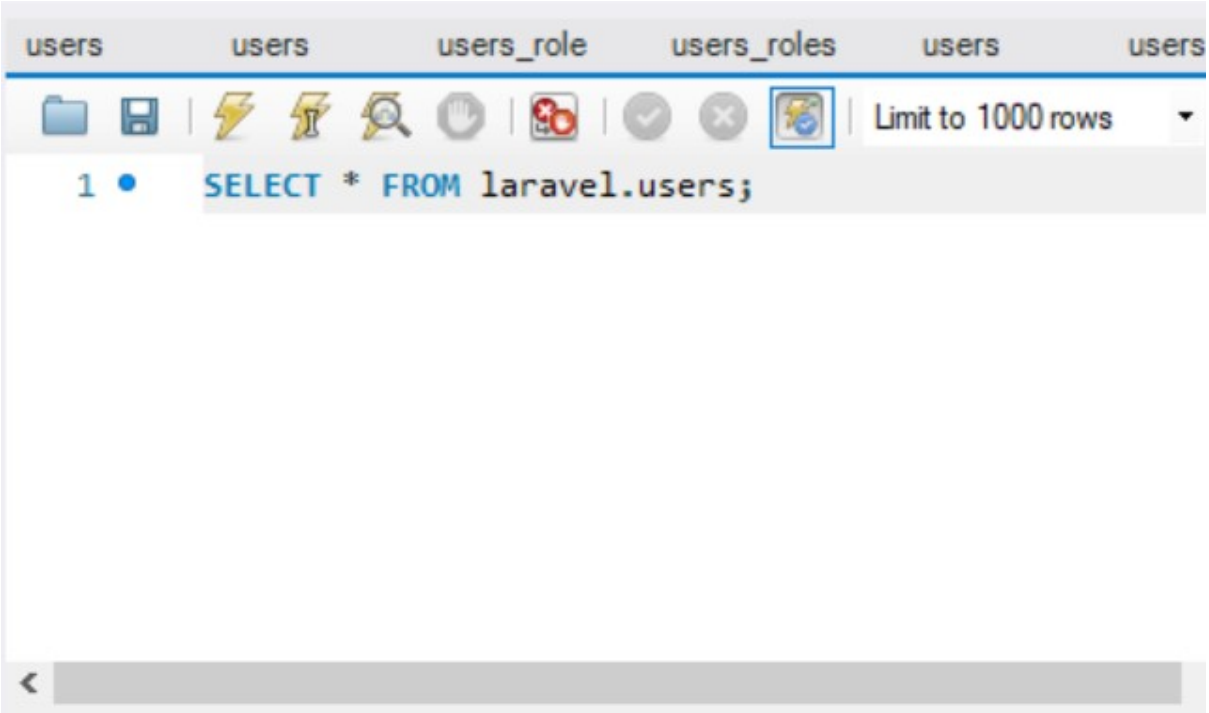
```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::table('users', function (Blueprint $table) {
            $table->boolean('is_admin')->default(false);
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::table('users', function (Blueprint $table) {
            //
        });
    }
};
```

- Выполним миграцию:

```
php artisan migrate
```

```
$ php artisan migrate

  WARN The database 'laravel' does not exist on the 'mysql' connection.

  Would you like to create it? (yes/no) [no]
› yes

  INFO Preparing database.

  Creating migration table ..................................... 49ms
DONE

  INFO Running migrations.

  2014_10_12_000000_create_users_table ........................... 33ms
DONE
  2014_10_12_100000_create_password_reset_tokens_table ............. 6ms
DONE
  2019_08_19_000000_create_failed_jobs_table ..................... 38ms
DONE
  2019_12_14_000001_create_personal_access_tokens_table ........... 39ms
DONE
```

```
  2025_03_29_131820_add_is_admin_to_users_table ..................... 5ms
DONE
```



10. **Создадим политику UserPolicy**:
    - Создадим политику:

```
php artisan make:policy UserPolicy --model=User
```

```
A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp/lesson11/proje
$ php artisan make:policy UserPolicy --model=User

  INFO  Policy [C:\laravelapp\lesson11\project-name\a
```

    - Откроем файл app/Policies/UserPolicy.php и добавьте функцию:

```
public function viewAny(User $user)
{
   return $user->is_admin;
}
```

```php
<?php

namespace App\Policies;

use App\Models\User;
use Illuminate\Auth\Access\Response;

class UserPolicy
{
    /**
     * Determine whether the user can view any models.
     */
    public function viewAny(User $user): bool
    {
```

```php
        return $user->is_admin;
    }

    /**
     * Determine whether the user can view the model.
     */
    public function view(User $user, User $model): bool
    {
        //
    }

    /**
     * Determine whether the user can create models.
     */
    public function create(User $user): bool
    {
        //
    }

    /**
     * Determine whether the user can update the model.
     */
    public function update(User $user, User $model): bool
    {
        //
    }

    /**
     * Determine whether the user can delete the model.
     */
    public function delete(User $user, User $model): bool
    {
        //
    }

    /**
     * Determine whether the user can restore the model.
     */
    public function restore(User $user, User $model): bool
    {
        //
    }

    /**
     * Determine whether the user can permanently delete the model.
     */
    public function forceDelete(User $user, User $model): bool
    {
        //
```

```
    }
}
```

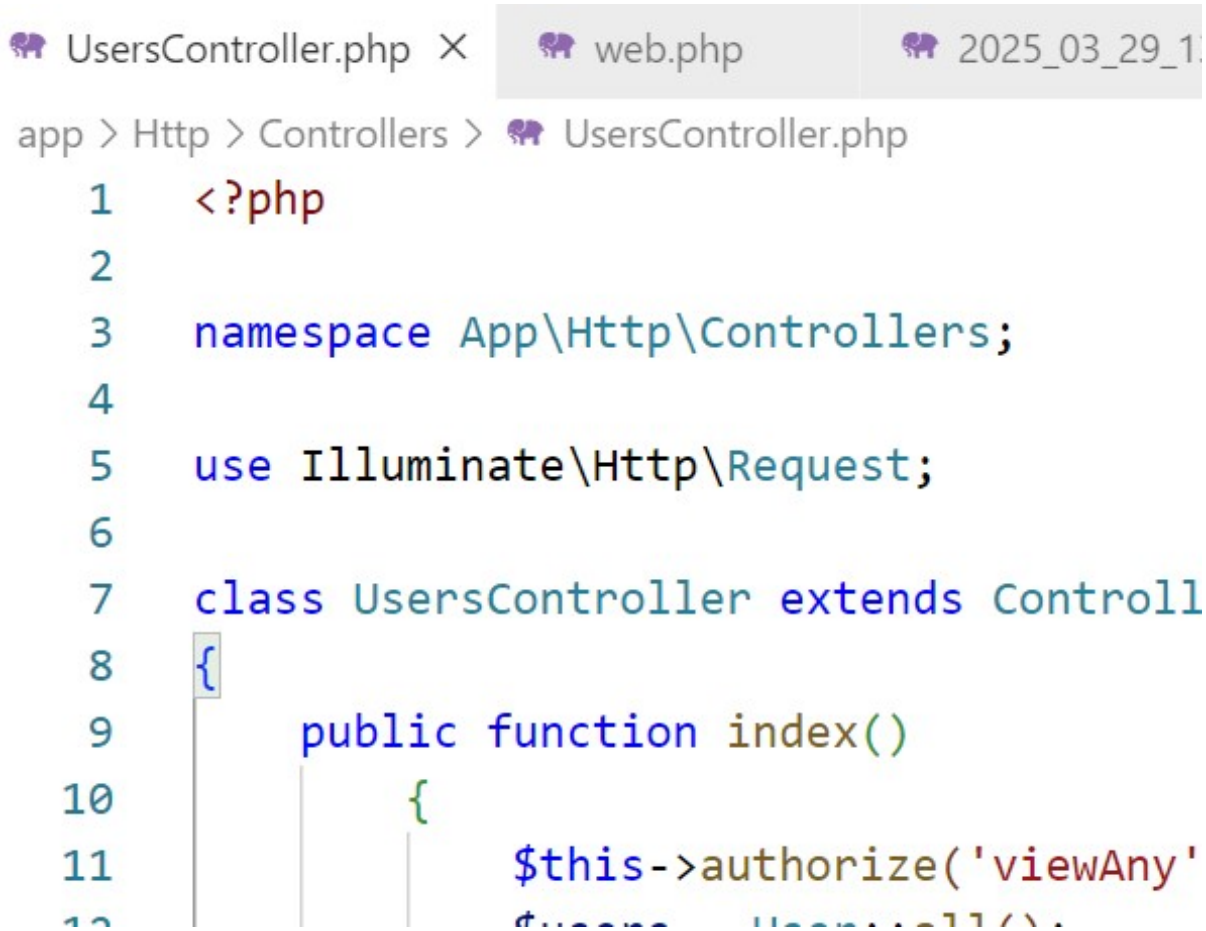
11. **Зарегистрируем политику в AuthServiceProvider**:
    - Откроем файл app/Providers/AuthServiceProvider.php и зарегистрируем политику:

```
protected $policies = [
    User::class => UserPolicy::class,
];
```

```php
<?php

namespace App\Providers;

// use Illuminate\Support\Facades\Gate;
use Illuminate\Foundation\Support\Providers\AuthServiceProvider as
ServiceProvider;

class AuthServiceProvider extends ServiceProvider
{
    /**
     * The model to policy mappings for the application.
     *
     * @var array<class-string, class-string>
     */
    protected $policies = [
        User::class => UserPolicy::class,
    ];

    public function boot(): void
    {
        //
    }
}
```

12. **Настроем использование авторизации в контроллере**:
    - Откроем файл app/Http/Controllers/UsersController.php и добавим авторизацию в функцию index:

```
public function index()
{
    $this->authorize('viewAny', User::class);
    $users = User::all();
    return view('users.index', compact('users'));
}
```



Создадим файл шаблона resources/views/users/index.blade.php:

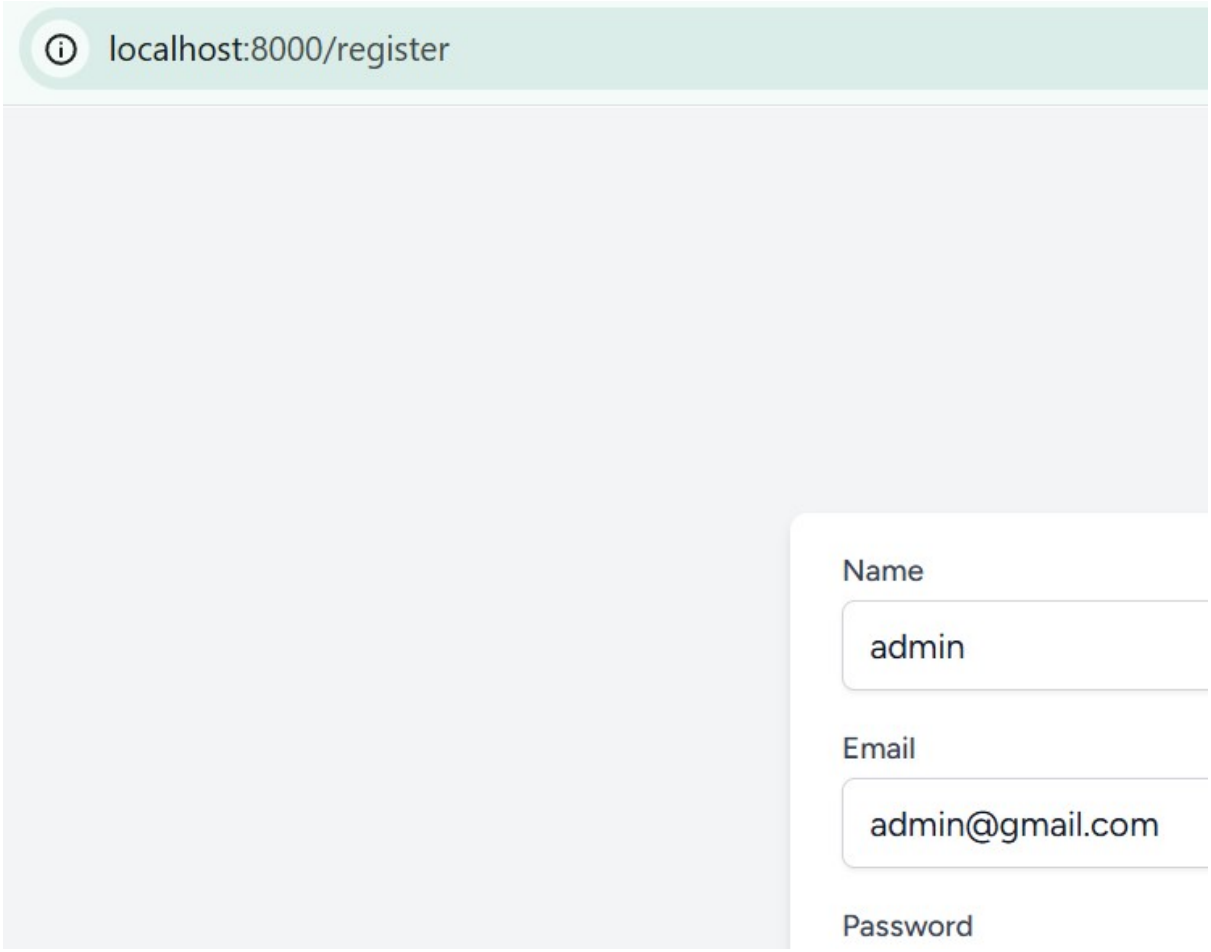
```
@extends('layouts.app')

@section('content')
    <div class="py-12">
        <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
            <div class="bg-white overflow-hidden shadow-sm
sm:rounded-lg">
                <div class="p-6 bg-white border-b border-gray-200">
                    <h1 class="text-2xl font-semibold mb-4">User
List</h1>
                    <ul class="space-y-2">
                        @foreach($users as $user)
                            <li class="p-2 bg-gray-50 rounded">
                                {{ $user->name }} ({{ $user->email
}})
                                @if($user->is_admin)
                                    <span class="ml-2 px-2 py-1
text-xs font-semibold bg-blue-100 text-blue-800 rounded-
full">Admin</span>
                                @endif
                            </li>
                        @endforeach
                    </ul>
```

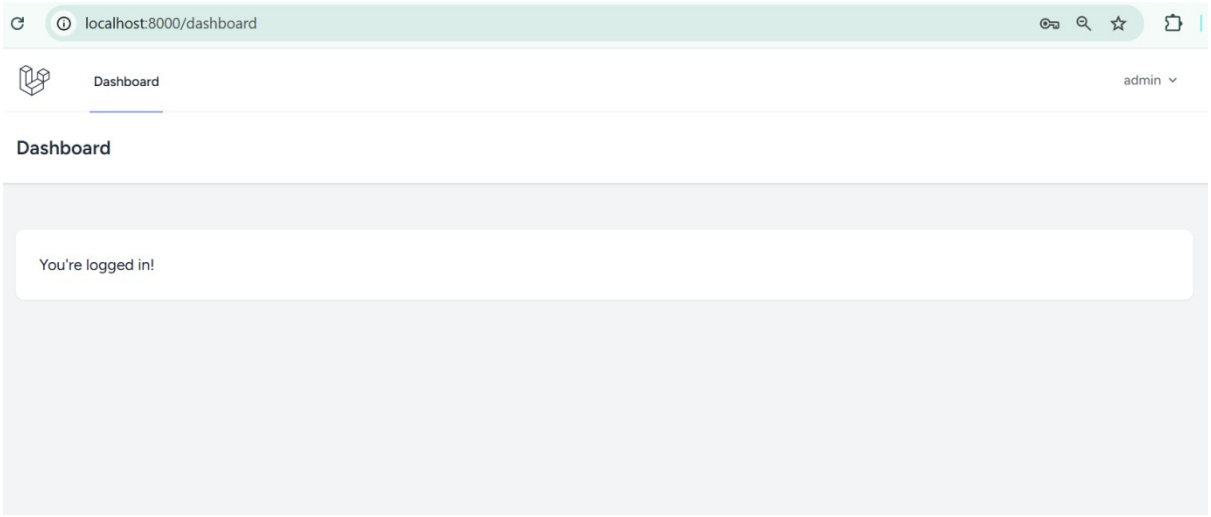
```
                </div>
            </div>
        </div>
    </div>
@endsection
```

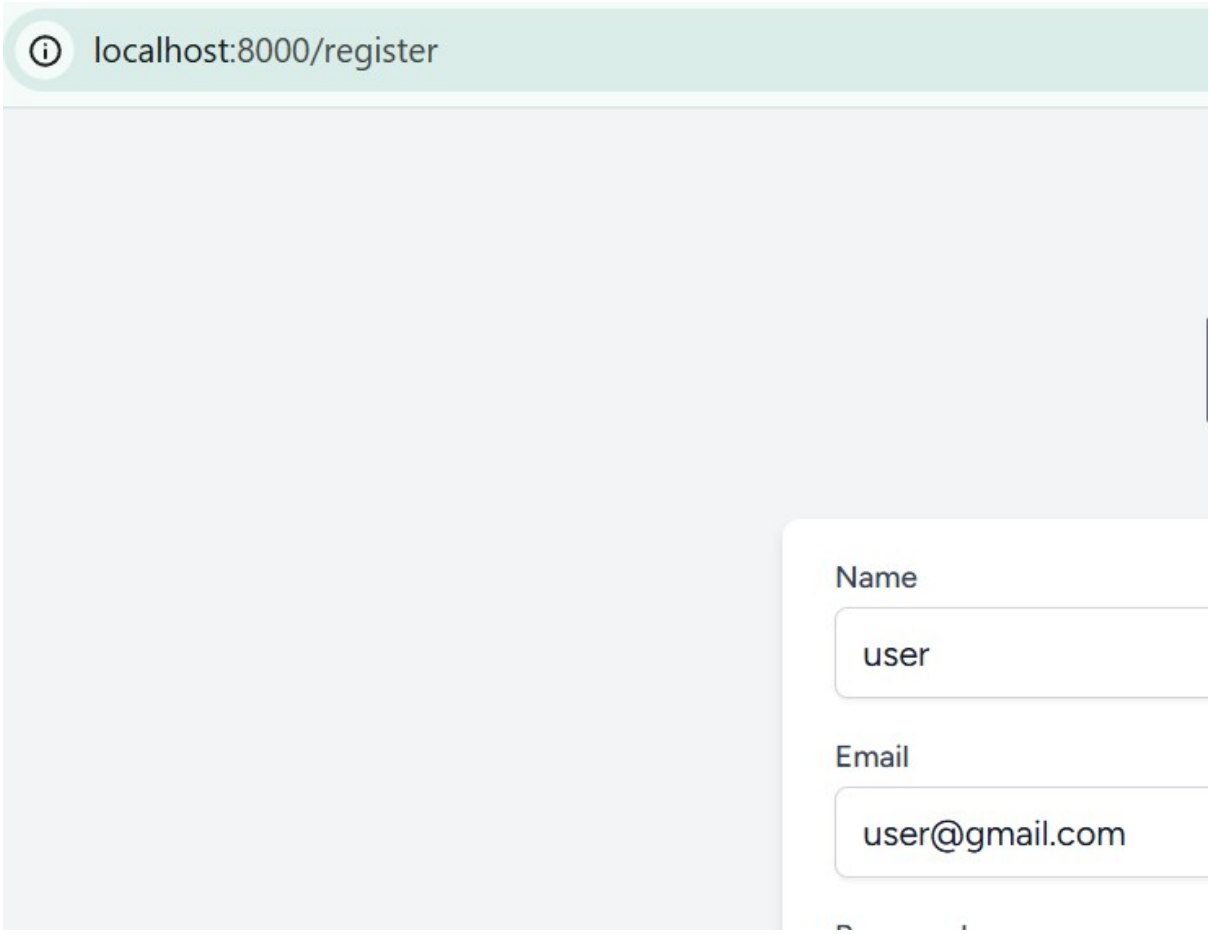
13. В файле **resources/views/layouts/app.blade.php** строку **{{ $slot }}** поменяем на **@yield('content')**:

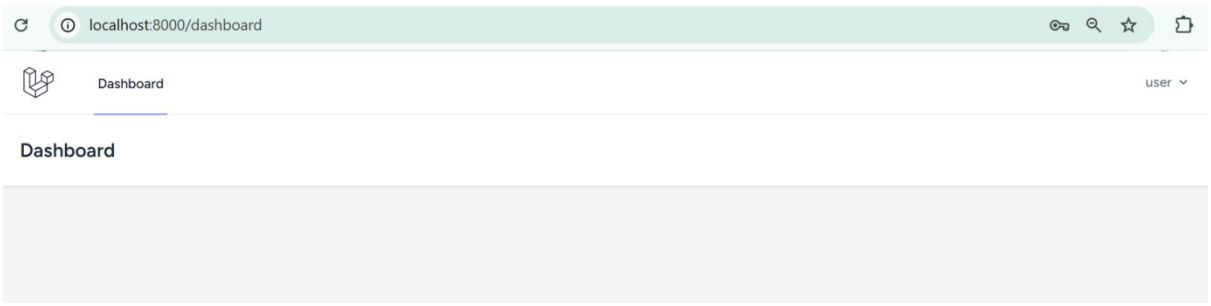
```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1">
        <meta name="csrf-token" content="{{ csrf_token() }}">

        <title>{{ config('app.name', 'Laravel') }}</title>

        <!-- Fonts -->
        <link rel="preconnect" href="https://fonts.bunny.net">
        <link
href="https://fonts.bunny.net/css?family=figtree:400,500,600&disp
lay=swap" rel="stylesheet" />

        <!-- Scripts -->
        @vite(['resources/css/app.css', 'resources/js/app.js'])
    </head>
    <body class="font-sans antialiased">
        <div class="min-h-screen bg-gray-100">
            @include('layouts.navigation')

            <!-- Page Heading -->
            @if (isset($header))
                <header class="bg-white shadow">
                    <div class="max-w-7xl mx-auto py-6 px-4
sm:px-6 lg:px-8">
                        {{ $header }}
                    </div>
                </header>
            @endif

            <!-- Page Content -->
            <main>
                @yield('content')
            </main>
        </div>
    </body>
</html>
```

14. **Создадим пользователей и проверим доступ**:
    - Создадим двух пользователей через интерфейс регистрации.

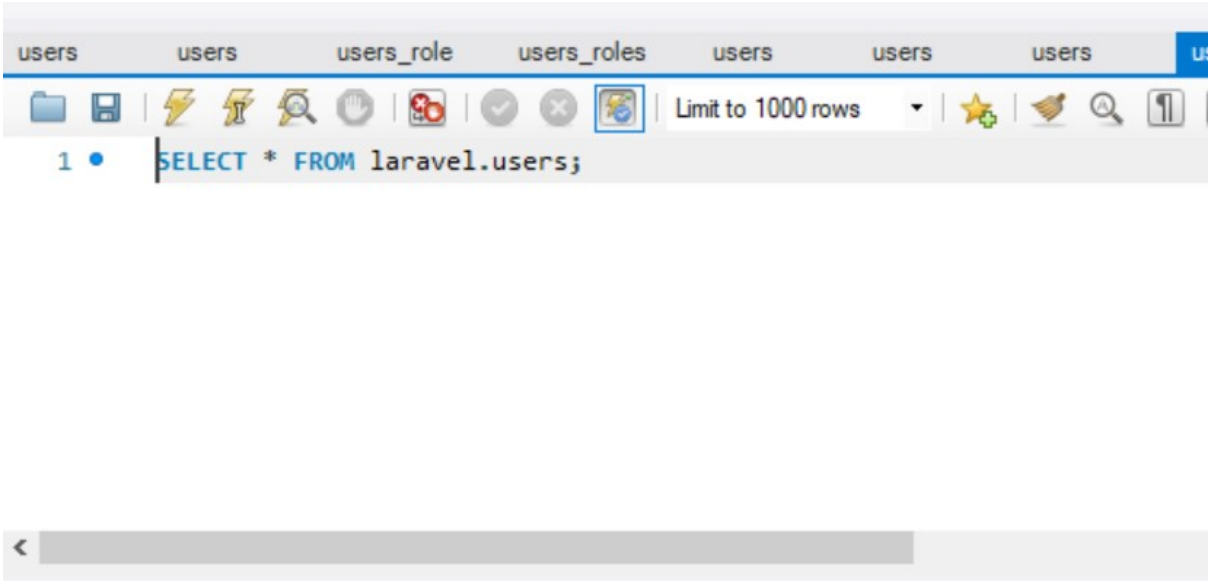






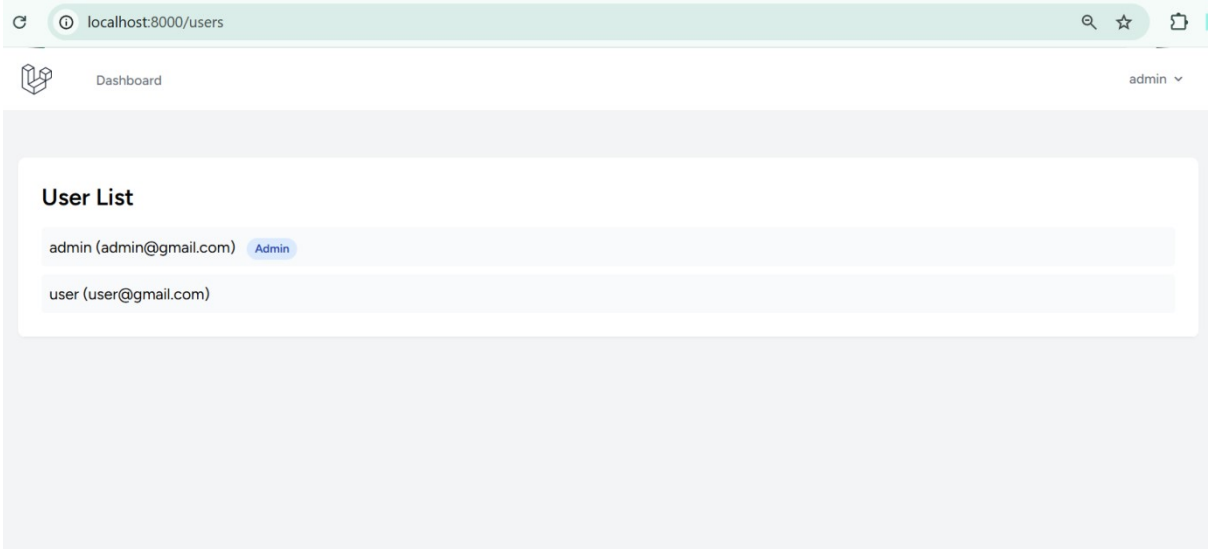


- Назначим одному из них роль администратора, обновив поле is_admin в базе данных.



- Проверим доступ к маршруту /users для неаутентифицированного пользователя, обычного пользователя и администратора.

Под пользователем admin видим список пользователей:



Под пользователем user видим, что маршрут http://localhost:8000/users не доступен:


localhost:8000/users