

Домашняя работа
по предмету:

Продвинутое программирование на PHP — Laravel

Выполнил: Байборodin Александр

Урок 7. Формирование ответа (Response)

Цели практической работы:

Научиться:

- использовать класс Laravel Response на практике;
- создавать CRUD REST API на базе фреймворка Laravel;
- передавать данные в формате PDF в ответе экземпляра класса Response.

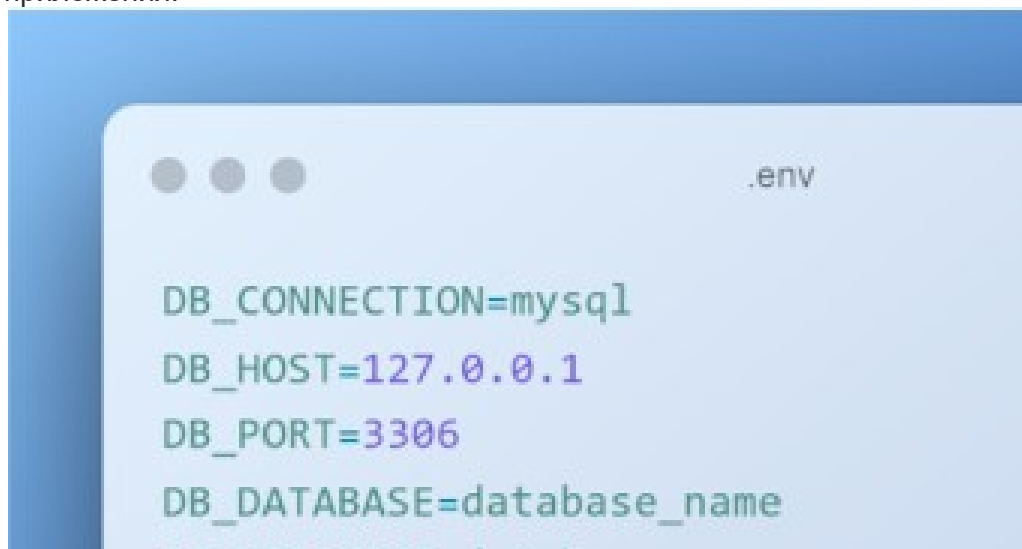
Что нужно сделать:

В этой практической работе вы будете разрабатывать контроллер, который позволит выводить информацию об одном и обо всех пользователях из базы данных, сохранять данные о новом пользователе в БД, а также создавать PDF с информацией о пользователе.

1. Установите новое приложение Laravel и настройте подключение к базе данных. Напомним, что создать новое приложение можно с помощью команды composer:

```
composer create-project laravel/laravel crud
```

Добавьте необходимые переменные окружения в ENV-файл корневого каталога приложения.

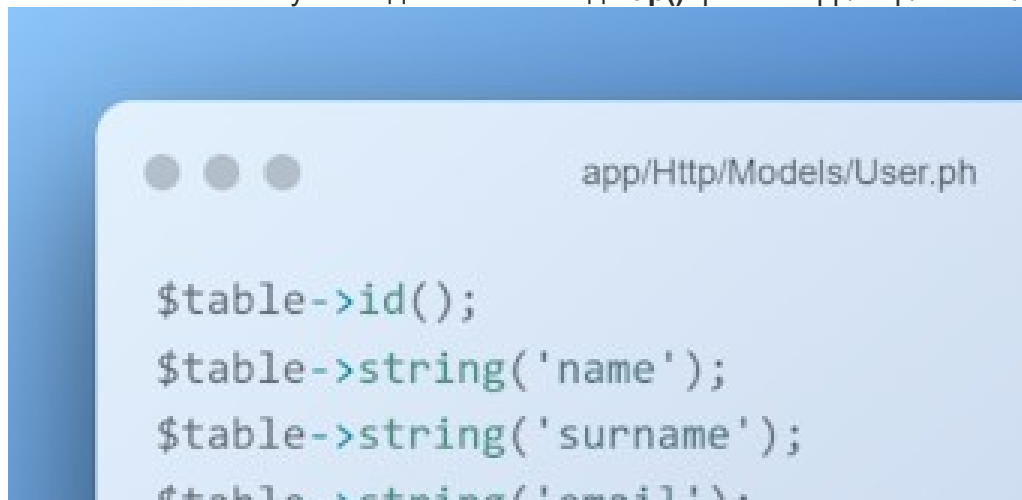


2. Создайте новую модель **Eloquent** с помощью команды:

```
php artisan make:model User -mfsc
```

Напомним, что флаг **-mfsc** создаст модель, наполнитель, контроллер и файл миграции.

После опишите схему базы данных в методе **up()** файла `app/Http/Models/User.php`.



```
app/Http/Models/User.php

$table->id();
$table->string('name');
$table->string('surname');
$table->string('email');
```

После описания схемы таблицы базы данных запустите миграцию.

3. Создайте необходимые роуты в файле `web.php`. Ваше приложение должно содержать минимум четыре эндпоинта:

- для получения всех пользователей из БД;
- получения одного пользователя через `id`, переданный в параметрах роута;
- записи нового пользователя в базу данных;
- получения данных о пользователе в виде PDF-файла.



```
Routes

Route::get('/user', [UserController::class, 'index']);
Route::get('/user/{id}', [UserController::class, 'show']);
Route::post('/store-user', [UserController::class, 'store']);
```

4. Создайте новый blade-шаблон. В blade-шаблоне создайте форму, которая будет отправлять данные о работнике. Важно, чтобы поля HTML-формы были сопоставимы с полями таблицы базы данных. При отправке запроса экземпляр класса `request` должен содержать данные об имени, фамилии и адресе электронной почты пользователя.

Форма blade-шаблона должна содержать CSRF-токен, поля формы должны быть обязательны к заполнению (используйте атрибут `required`).

5. В контроллере `UserController.php` опишите функцию `store`, которая будет сохранять данные из вашей HTML-формы. Добавьте валидацию.

```
store UserController

public function store(Request $request)
{
    //
    $request->validate([
        'name'=>'required',
        'surname'=>'required',
        'email'=>'required',
```

Дополнительно. Добавьте валидацию на количество символов (максимальное количество символов — 50) для полей Name и Surname. Для почты добавьте валидацию в виде регулярного выражения на соответствие виду example@mail.com.

```
index/get methods UserControllere

public function index(){
    return User::all();
}

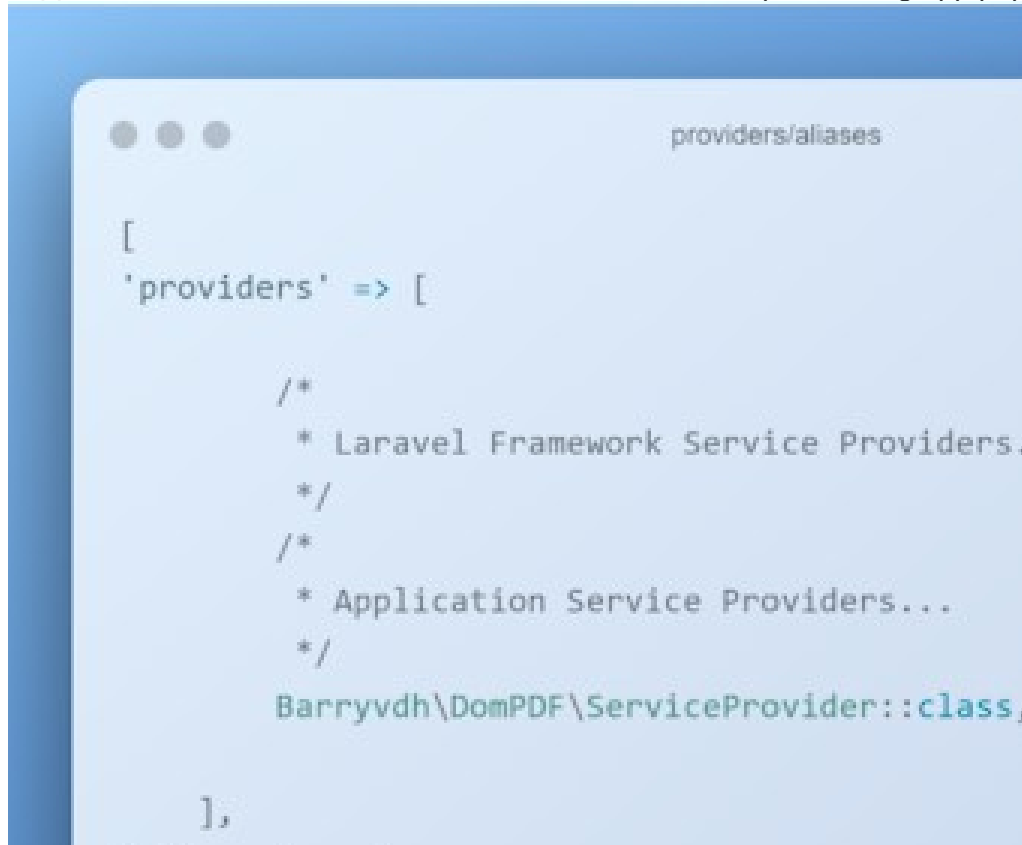
public function get(Request $request)
{
    $user = User::where('id',$id)->first();
    return $user;
}
```

6. Добавьте соответствующие методы `index` и `get`, которые будут возвращать данные обо всех пользователях и об одном пользователе по переданному `id`. Опционально можете возвращать ответ в формате JSON.

7. Чтобы генерировать PDF-документ, вам понадобится **DOMPDF**-пакет, который является сторонней библиотекой. Для его установки выполните команду:

`composer require barryvdh/laravel-dompdf`

- В файле `composer.json` добавьте строку с указанным пакетом.
- Запустите команду `composer update`.
- Добавьте необходимый Service Provider и Facade в файл `config/app.php`.



8. Создайте новый контроллер для работы с PDF:

`php artisan make:controller PdfGeneratorController`

9. Опишите функцию **index**, которая будет возвращать новый PDF-файл.

A screenshot of a code editor window with a blue header bar. The window title is "/app/Http/Controllers/PdfGeneratorControl". The code is in PHP and defines a class PdfGeneratorController. The code is as follows:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use PDF;

class PdfGeneratorController extends Controller
{
    public function index()
    {
        $data = [
            'name' => 'John',
```

10. Измените роут `Route::get('/resume')` таким образом, чтобы он принимал `id` в виде параметра. Обновите функцию «`index`» так, чтобы PDF формировался на основе данных из таблицы по переданному `id`.

Результат

Создадим проект laravel:


```

- Installing phpunit/phpunit (10.5.45): Extracting archive
- Installing spatie/error-solutions (1.1.3): Extracting archive
- Installing spatie/backtrace (1.7.1): Extracting archive
- Installing spatie/flare-client-php (1.10.1): Extracting archive
- Installing spatie/ignition (1.15.1): Extracting archive
- Installing spatie/laravel-ignition (2.9.1): Extracting archive
64 package suggestions were added by new dependencies, use `composer suggest` to see details.
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

```

INFO Discovering packages.

```

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

```

61. Настраиваем подключение к базе данных для Laravel Foundation
 Настройте подключение к базе данных, добавив соответствующие переменные в файл .env:


```
.env X
C: > laravelapp > lesson7 > crud > .env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:QDwM1YWjCJfAIp2+01XOXq2/FTL+
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
```

Создадим модель User с миграцией, фабрикой и контроллером:

```
A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp
$ php artisan make:model User -mfsc
```

В файле миграции опишем схему таблицы:

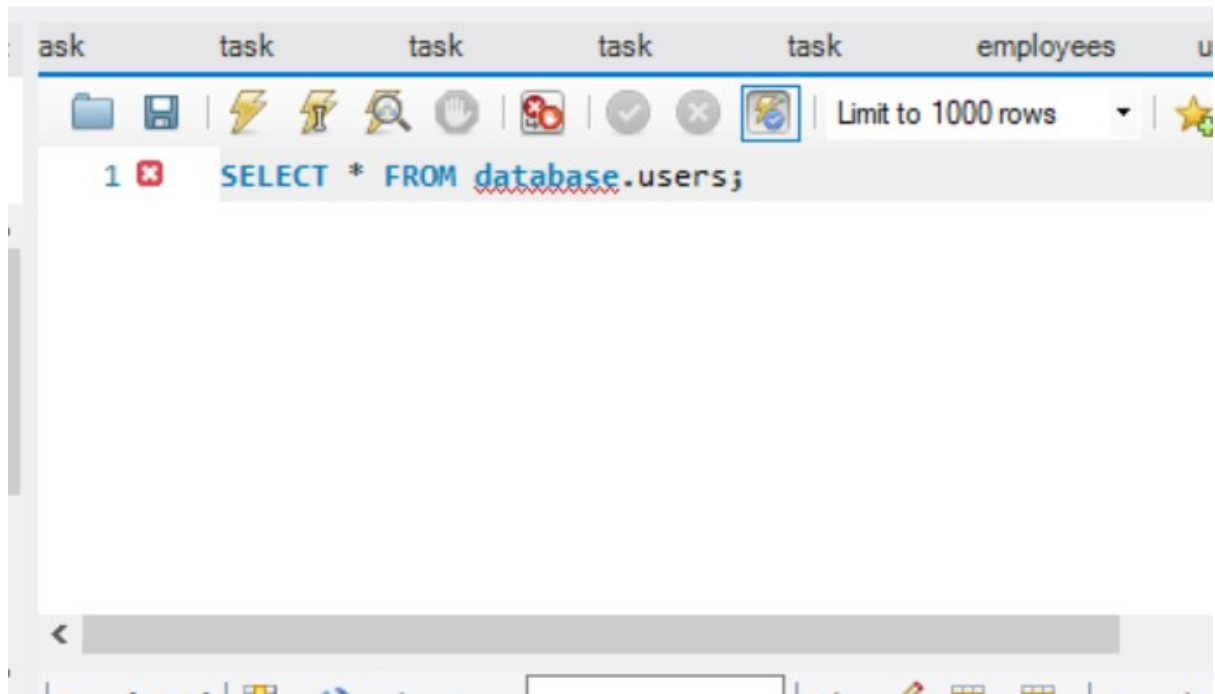
```
File Edit Selection View Go Run ... < >
.env 2014_10_12_000000_create_users_table.php X
C: > laravelapp > lesson7 > crud > database > migrations > 2014_10_12_000000_create_users_table.php
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('users', function
```


Запустим миграцию:

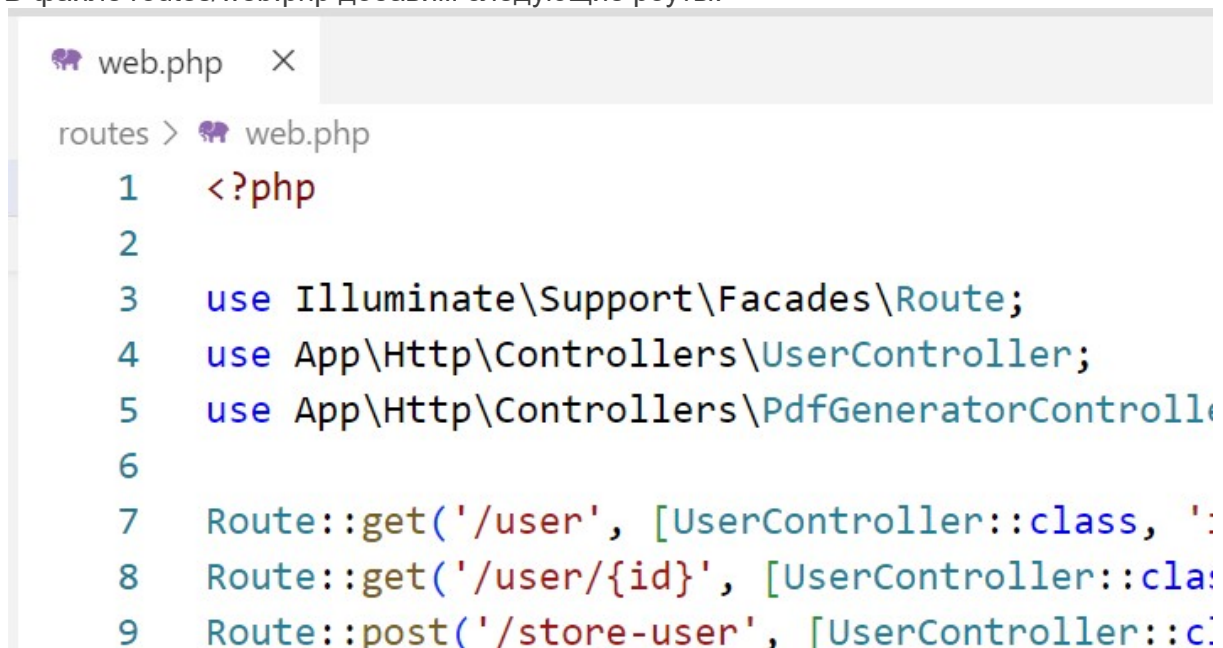
```
A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp/lesson7/crud
$ php artisan migrate
```

INFO Running migrations.

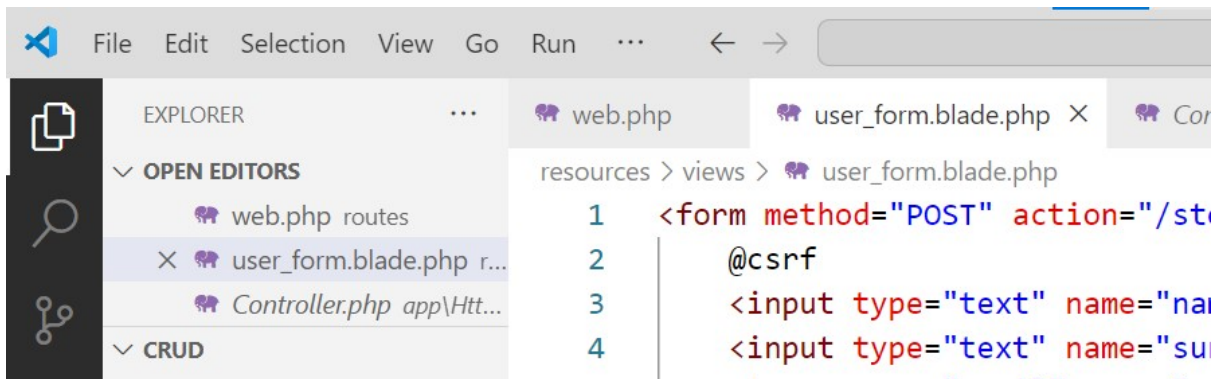
2025_03_09_000000_create_users_table



В файле routes/web.php добавим следующие роуты:



Создадим Blade-шаблон для формы, resources/views/user_form.blade.php:



Создадим контроллер UserController:

```
A@LAPTOP-RHI13GV6 MINGW64 /c:/laravelapp/lesson7/crud
$ php artisan make:controller UserController
```

Добавим в него метод store, index и get:

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use Illuminate\Http\Request;
```

```
class UserController extends Controller
```

```
{
    public function store(Request $request)
    {
        $request->validate([
            'name' => 'required|max:50',
            'surname' => 'required|max:50',
            'email' => 'required|email',
        ]);
    }
}
```

```
return User::create($request->all());
}
```

```
public function index()
{
    return User::all();
}
```

```
public function get(Request $request, $id)
{
    return User::findOrFail($id);
}
```

```
}
```

Установим пакет DOMPD:

```

A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp/1
$ composer require barryvdh/laravel-dompdf
./composer.json has been updated
Running composer update barryvdh/laravel-
Loading composer repositories with packag
Updating dependencies
Lock file operations: 6 installs, 0 updat
- Locking barryvdh/laravel-dompdf (v3.1
- Locking dompdf/dompdf (v3.1.0)
- Locking dompdf/php-font-lib (1.0.1)
- Locking dompdf/php-svg-lib (1.0.0)
- Locking masterminds/html5 (2.9.0)
- Locking sabberworm/php-css-parser (v8
Writing lock file
- Locking sabberworm/php-css-parser (v8.7.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 6 installs, 0 updates, 0 removals
- Downloading masterminds/html5 (2.9.0)
- Downloading sabberworm/php-css-parser (v8.7.0)
- Downloading dompdf/php-svg-lib (1.0.0)
- Downloading dompdf/php-font-lib (1.0.1)
- Downloading dompdf/dompdf (v3.1.0)
- Downloading barryvdh/laravel-dompdf (v3.1.1)
- Installing masterminds/html5 (2.9.0): Extracting archive
- Installing sabberworm/php-css-parser (v8.7.0): Extracting archive
- Installing dompdf/php-svg-lib (1.0.0): Extracting archive
- Installing dompdf/php-font-lib (1.0.1): Extracting archive
- Installing dompdf/dompdf (v3.1.0): Extracting archive
- Installing barryvdh/laravel-dompdf (v3.1.1): Extracting archive
3 package suggestions were added by new dependencies, use `composer suggest` to s
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

```

INFO Discovering packages.

```

barryvdh/laravel-dompdf .....
laravel/sail .....
laravel/sanctum .....
laravel/tinker .....
nesbot/carbon .....
nunomaduro/collision .....
nunomaduro/termwind .....

```

Добавим провайдер и алиас в config/app.php:

```
<?php
```

```

use Illuminate\Support\Facades\Facade;
use Illuminate\Support\ServiceProvider;

```

```

return [

    /*
    |-----
    | Application Name
    |-----
    |
    | This value is the name of your application. This value is used when
the | framework needs to place the application's name in a notification or
    | any other location as required by the application or its packages.
    |
    */

    'name' => env('APP_NAME', 'Laravel'),

    /*
    |-----
    | Application Environment
    |-----
    |
    | This value determines the "environment" your application is
currently | running in. This may determine how you prefer to configure various
    | services the application utilizes. Set this in your ".env" file.
    |
    */

    'env' => env('APP_ENV', 'production'),

    /*
    |-----
    | Application Debug Mode
    |-----
    |
    | When your application is in debug mode, detailed error messages with
    | stack traces will be shown on every error that occurs within your
    | application. If disabled, a simple generic error page is shown.
    |
    */

    'debug' => (bool) env('APP_DEBUG', false),

    /*
    |-----
    | Application URL

```

```

|-----
-----
|
| This URL is used by the console to properly generate URLs when using
| the Artisan command line tool. You should set this to the root of
| your application so that it is used when running Artisan tasks.
|
*/

'url' => env('APP_URL', 'http://localhost'),

'asset_url' => env('ASSET_URL'),

/*
|-----
-----
| Application Timezone
|-----
-----
|
| Here you may specify the default timezone for your application,
which | will be used by the PHP date and date-time functions. We have gone
| ahead and set this to a sensible default for you out of the box.
|
*/

'timezone' => 'UTC',

/*
|-----
-----
| Application Locale Configuration
|-----
-----
|
| The application locale determines the default locale that will be
used | by the translation service provider. You are free to set this value
| to any of the locales which will be supported by the application.
|
*/

'locale' => 'en',

/*
|-----
-----
| Application Fallback Locale
|-----
-----
|
| The fallback locale determines the locale to use when the current
one

```

```

| is not available. You may change the value to correspond to any of
| the language folders that are provided through your application.
|
*/

'fallback_locale' => 'en',

/*
|-----
|
| Faker Locale
|-----
|
|
| This locale will be used by the Faker PHP library when generating
fake | data for your database seeds. For example, this will be used to get
| localized telephone numbers, street address information and more.
|
*/

'faker_locale' => 'en_US',

/*
|-----
|
| Encryption Key
|-----
|
|
| This key is used by the Illuminate encrypter service and should be
set | to a random, 32 character string, otherwise these encrypted strings
| will not be safe. Please do this before deploying an application!
|
*/

'key' => env('APP_KEY'),

'cipher' => 'AES-256-CBC',

/*
|-----
|
| Maintenance Mode Driver
|-----
|
|
| These configuration options determine the driver used to determine
and | manage Laravel's "maintenance mode" status. The "cache" driver will
| allow maintenance mode to be controlled across multiple machines.
|
| Supported drivers: "file", "cache"

```



```

|
| */
|
| 'maintenance' => [
|     'driver' => 'file',
|     // 'store' => 'redis',
| ],
|
| /*
| -----
|
| Autoloaded Service Providers
| -----
|
| The service providers listed here will be automatically loaded on
the | request to your application. Feel free to add your own services to
| this array to grant expanded functionality to your applications.
|
| */

| 'providers' => ServiceProvider::defaultProviders()->merge([
|     /*
|      * Package Service Providers...
|      */
|
|     /*
|      * Application Service Providers...
|      */
|     App\Providers\AppServiceProvider::class,
|     App\Providers\AuthServiceProvider::class,
|     // App\Providers\BroadcastServiceProvider::class,
|     App\Providers\EventServiceProvider::class,
|     Barryvdh\DomPDF\ServiceProvider::class,
|     App\Providers\RouteServiceProvider::class,
|
| ])->toArray(),
|
| /*
| -----
|
| Class Aliases
| -----
|
| This array of class aliases will be registered when this application
| is started. However, feel free to register as many as you wish as
| the aliases are "lazy" loaded so they don't hinder performance.
|
| */

| 'aliases' => Facade::defaultAliases()->merge([
|     // 'Example' => App\Facades\Example::class,

```

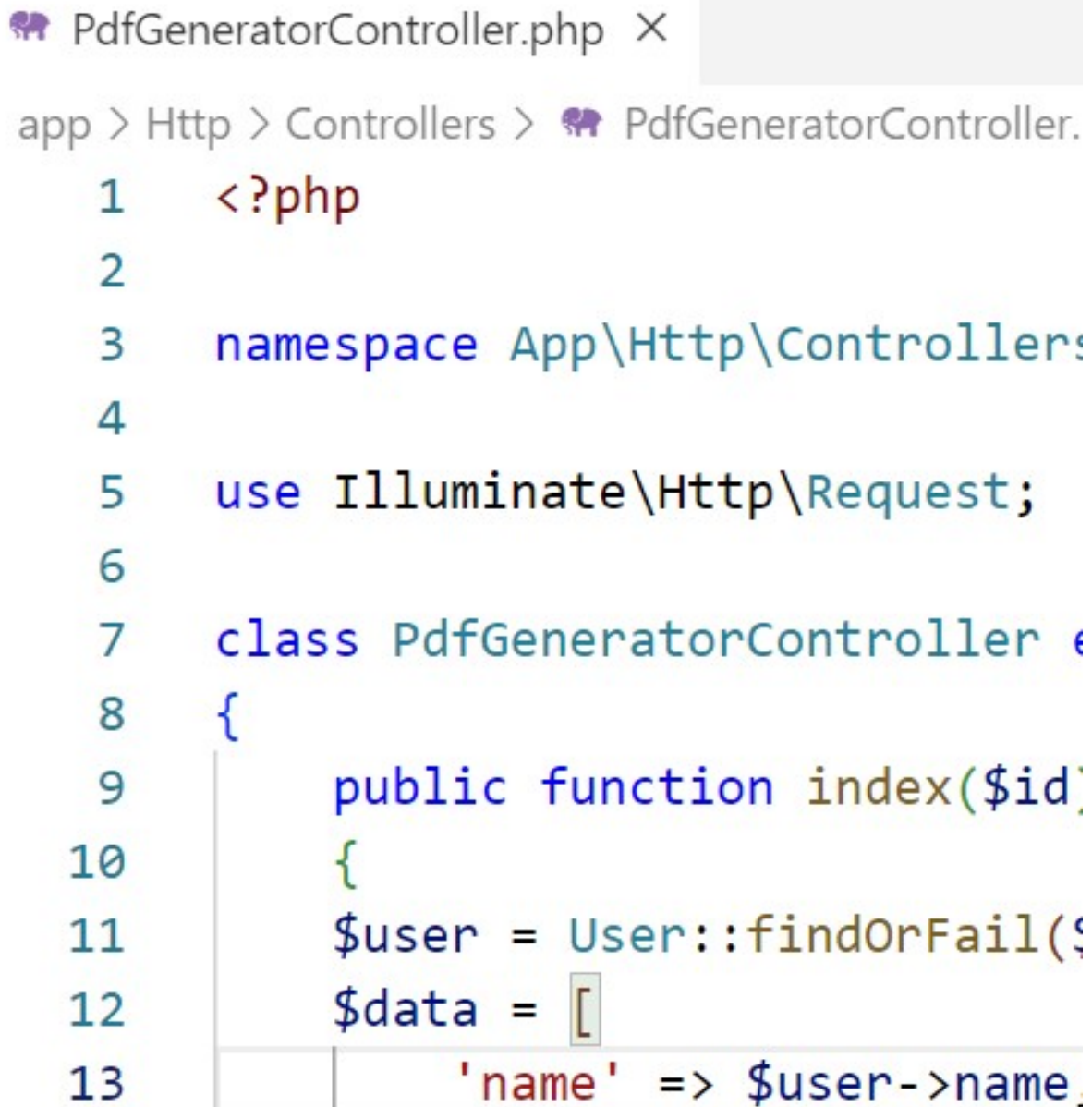
```
'PDF' => Barryvdh\DomPDF\Facade::class,  
])->toArray(),
```

```
];
```

Создадим контроллер для генерации PDF:

```
A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp/lesson7/crud  
$ php artisan make:controller PdfGeneratorController
```

Добавим в него метод index:



```
PdfGeneratorController.php X  
  
app > Http > Controllers > PdfGeneratorController.  
1  <?php  
2  
3  namespace App\Http\Controllers;  
4  
5  use Illuminate\Http\Request;  
6  
7  class PdfGeneratorController  
8  {  
9      public function index($id)  
10     {  
11         $user = User::findOrFail($id);  
12         $data = [  
13             'name' => $user->name,
```

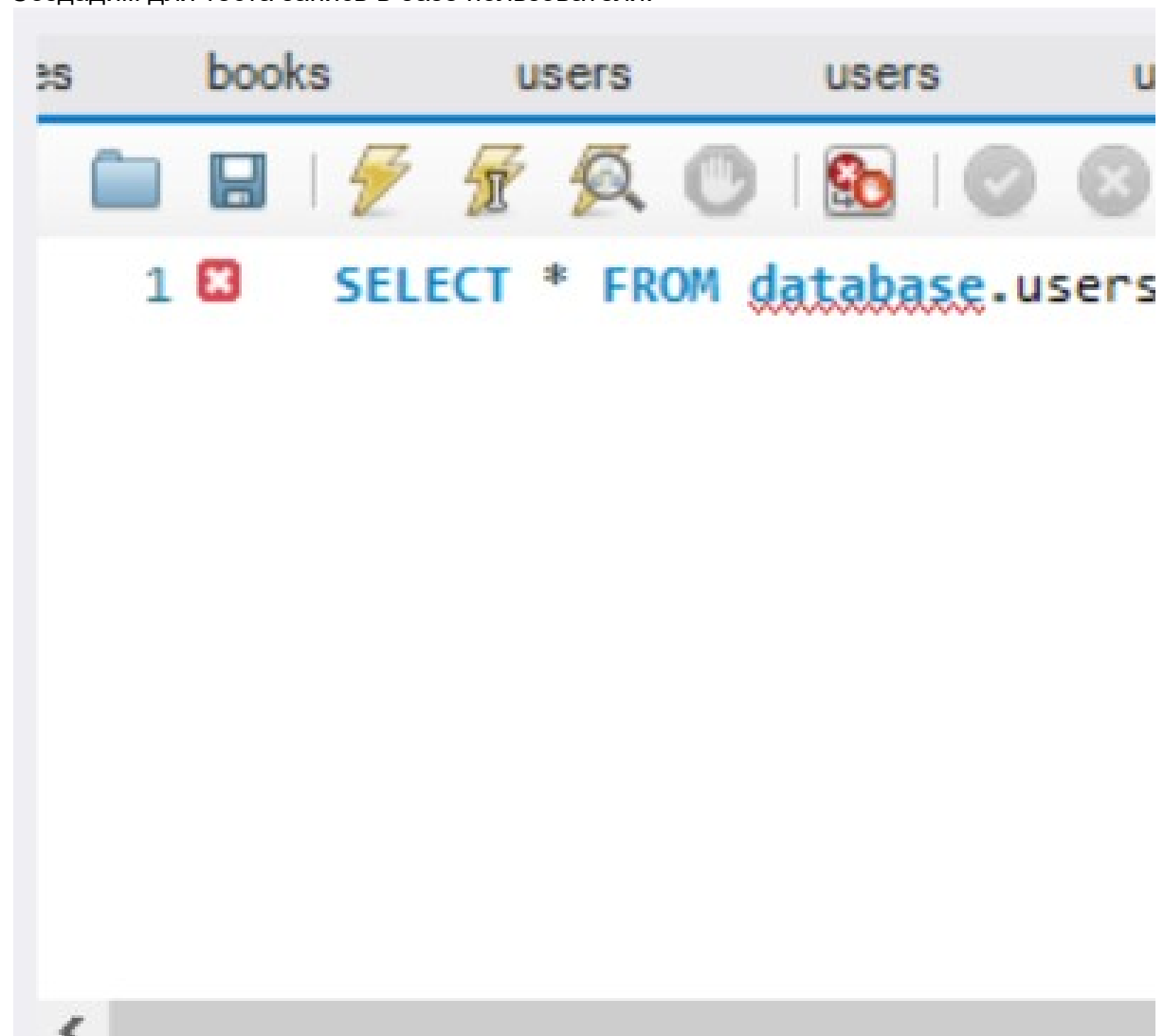
Создайте Blade-шаблон для PDF resources/views/resume.blade.php:

 resume.blade.php X

resources > views >  resume.blade.php

```
1 <h1>{{ $name }} {{ $surname }}
2 <h2>Email: {{ $email }}
```

Создадим для теста запись в базе пользователя:



Запустим сервер:

```
A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp/1
$ php artisan serve
```

```
INFO Server running on [http://127.0.
```

```
Press Ctrl+C to stop the server
```

2025-03-08 22:04:54
Протестируем получение pdf файла по пользователю через роутер:
<http://127.0.0.1:8000/resume/1>:

