Kickstart API with Lumen

Tawsif Aqib

Lumen Framework

https://lumen.laravel.com

Micro-framework by Laravel

About ¼ of Laravel features

Start Project

terminal

```
$ composer create-project laravel/lumen lumen-api --prefer-dist
```

Missing Generator!

php artisan

```
Available commands:
  help                Displays help for a command
  list                Lists commands
  migrate             Run the database migrations
...
make
  make:migration      Create a new migration file
  make:seeder         Create a new seeder class
...
```

flipbox/lumen-generator

composer require flipbox/lumen-generator --dev

```
make:channel          Create a new channel class
make:command          Create a new Artisan command
make:controller       Create a new controller class
make:event            Create a new event class
make:job              Create a new job class
make:listener         Create a new event listener class
make:mail             Create a new email class
make:middleware       Create a new middleware class
make:migration        Create a new migration file
make:model            Create a new Eloquent model class
make:notification     Create a new notification class
make:pipe             Create a new pipe class
make:policy           Create a new policy class
make:provider         Create a new service provider class
make:request          Create a new form request class
make:seeder           Create a new seeder class
make:test             Create a new test class

route:list            Display all registered routes.
```

Docker Compose

docker-compose.yml

```
version: "3"

services:
  mysql:
    image: mysql:8
    container_name: mysql
    restart: always
    command: "--default-authentication-plugin=mysql_native_password"
    ports:
      - 3306:3306
    environment:
      MYSQL_DATABASE: lumen-api
      MYSQL_USER: user
      MYSQL_PASSWORD: secret
      MYSQL_ROOT_PASSWORD: supersecret
    volumes:
      - mysql-data:/var/lib/mysql

volumes:
  mysql-data:
```

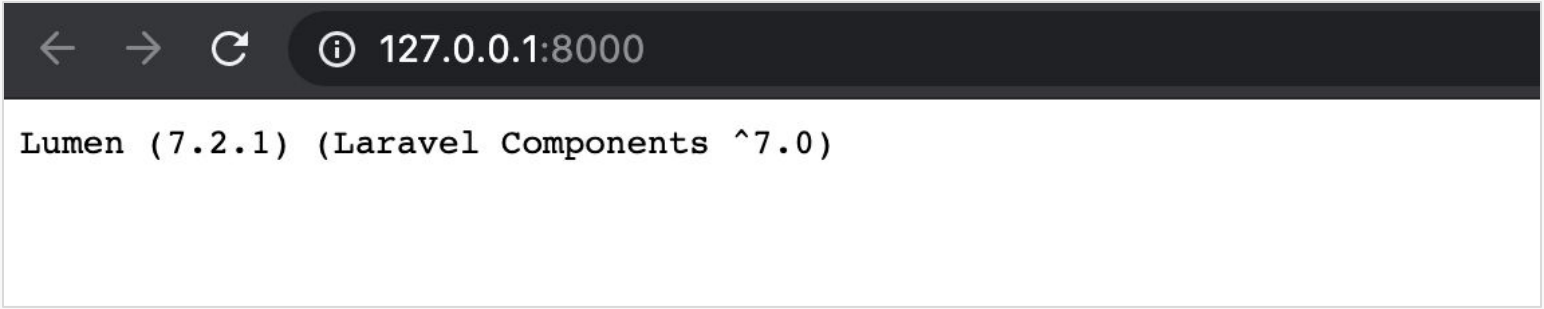
Database Config

.env

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=lumen-api
DB_USERNAME=user
DB_PASSWORD=secret
```

Start Application

```
$ docker-compose up -d
$ php artisan serve
Laravel development server started: <http://127.0.0.1:8000>
PHP 7.4.8 Development Server (http://127.0.0.1:8000) started
```



Missing Config Directory

no such file or directory: config

```
app/
bootstrap/
database/
public/
resources/
routes/
storage/
tests/
vendor/
```

Copy the Config

```
$ cp -r ./vendor/laravel/lumen-framework/config .
```

bootstrap/app.php

```
/*
|--------------------------------------------------------------------------
| Register Config Files
|--------------------------------------------------------------------------
|
| Now we will register the "app" configuration file. If the file exists in
| your configuration directory it will be loaded; otherwise, we'll load
| the default version. You may register other files below as needed.
|
*/

$app->configure('app');
$app->configure('database');
```

tymon/jwt-auth

composer require tymon/jwt-auth

routes/web.php

```
$router->post('/auth', 'AuthController@store');
$router->group(['middleware' => 'auth:api', 'prefix' => 'auth'], function ($router) {
    // protected urls
});
```

app/Http/Controllers/AuthController.php

```
class AuthController extends Controller
{
    public function store(Request $request): JsonResponse
    {
        if (!$token = app('auth')->attempt($request->input('email'),
                                           $request->input('password')) ) {
            throw new UnauthorizedException();
        }
        return response()->json(compact($token), Response::HTTP_OK);
    }
}
```

spatie/laravel-query-builder

composer require spatie/laravel-query-builder

Inside Controller/Serbvice Method

```
// Instead of using

$users = User::all();

// Use

$users = QueryBuilder::for(User::class)
    ->allowedFilters(['name'])
    ->get();
```

URL

```
GET /users?filter[name]=anam,sifat
```

Validation

Default Lumen Validation

Inside Controller/Service Method

```
$this->validate($request, [
    'name' => 'required',
    'email' => 'required|email|unique:users'
]);
```

spatie/laravel-fractal

composer require spatie/laravel-fractal

app/Transformers/UserTransfromer.php

```
class UserTransformer extends \League\Fractal\TransformerAbstract
{
    public function transform(User $user): array
    {
        return [
            'id' => (int) $user->id,
            'name' => (string) $user->name,
            'email' => (string) $user->email,
        ];
    }
}
```

To transform the response

```
fractal(User::all(), new UserTransformer())->toArray();
```

fruitcake/laravel-cors

composer require fruitcake/laravel-cors

config/cors.php

```
<?php

return [
    'paths' => [],
    'allowed_methods' => ['*'],
    'allowed_origins' => ['*'],
    'allowed_origins_patterns' => [],
    'allowed_headers' => ['*'],
    'exposed_headers' => [],
    'max_age' => 0,
    'supports_credentials' => false,
];
```

Exception Handling

app/Exceptions/Handler.php

```
class Handler extends ExceptionHandler
{
    use ExceptionRenderable ;

    public function render($request, Throwable $exception)
    {
        if ($this->isJsonRenderable($exception)) {
            return $this->renderJson($request, $exception);
        }
        return parent::render($request, $exception);
    }
}
```

Exception Handling

app/Traits/ExceptionRenderable.php

```
trait ExceptionRenderable
{
    private function renderJson(Request $request, Throwable $exception): JsonResponse
    {
        $error = fractal($exception, new ErrorTransformer(), new ErrorSerializer())->toArray();
        return response()->json($error)
            ->setStatusCode($this->getStatusCodeFromError($error))
            ->withHeaders($this->getHeadersFromException($exception));
    }

    private function getStatusCodeFromError(array $error): int
    {
        return Arr::get($error, 'data.status') ?? Arr::get($error, 'error.status')
            ?? Response::HTTP_INTERNAL_SERVER_ERROR;
    }

    private function getHeadersFromException(Throwable $exception): array
    {
        return method_exists($exception, 'getHeaders') ? $exception->getHeaders() : [];
    }
}
```

Application Structure

```
app
├── Console
├── Events
├── Exceptions
├── Http
│   ├── Controllers
│   └── Middlewares
├── Jobs
├── Listeners
├── Models
│   └── User.php
├── Providers
├── Serializers
├── Traits
└── Transformers
Accounts.php
Auth.php
```

Domain Logic

app/Accounts.php

```
 class Accounts
{
    public function getUsersWithPagination(Request $request): array
    {
        $users = User::filter($request)->paginate($request->get('per_page', 20));

        return fractal($users, new UserTransformer())->toArray();
    }
}
```

app/Auth.php

```
class Auth
{
    public function authenticateByEmailAndPassword(string $email, string $password): array
    {
        if (!$token = app('auth')->attempt(compact('email', 'password'))) {
            throw new UnauthorizedException();
        }
        return fractal($token, new TokenTransformer())->toArray();
    }
}
```

Example Project

https://github.com/munza/lumen-api-starter

Reference Links

https://github.com/flipboxstudio/lumen-generator

https://github.com/tymondesigns/jwt-auth

https://github.com/spatie/laravel-query-builder

https://github.com/spatie/laravel-fractal

https://github.com/fruitcake/laravel-cors

Thanks

For you time