Gradus Vol 8, No 1 (2021) 211-218
ISSN 2064-8014

WEB-DEVELOPMENT WITH LARAVEL FRAMEWORK

Zoltán Subecz[1*]

[1] Department of Information Technology, GAMF Faculty of Engineering and Computer Science, John von Neumann University, Hungary

https://doi.org/10.47833/2021.1.CSC.006

Keywords:
Web-development
PHP Framework
Laravel
MVC model
Web-application

Article history:
Received 30 March 2021
Revised 20 April 2021
Accepted 22 April 2021

Abstract
Laravel is a useful PHP framework and we can create web-application easily with this application. It uses the popular MVC (model–view–controller) design pattern and based the Symfony system. Laravel uses a modular package system, therefore we can expand our application with new modules. It reuses several existing components of other frameworks, which helps to create a secure operable application quickly. Some of its features and advantages: supports accessing of different databases; presents utilities which helps in web- application development; simple and fast routing engine; the web application becomes more scalable; considerable time is saved in designing. Laravel is open-source software, licensed by MIT.

1 Introduction

Laravel is the most usable PHP framework for beginner and advanced programmers as well. It can reduce the time of web- application development and getting to the market with modern object oriented PHP methods. Its expressive syntax and modern functions are attractive for developers who want to create robust applications [7]. Using a framework eases the process of development because it supplies several modules with their connections together. The web-framework gives us a starting point to the application creation and enables the developer to concentrate on more interesting issues. Laravel provides such powerful features as expressive database abstract layer and dependency injection and it's highly scalable.

In this paper I introduce the main characteristics of this useful and rightly popular framework.

2 Routing

The main function of a web-application framework is to receive the requests from the users and send responses usually via HTTP(S). It means that the first task during the application process is to determine the necessary routes. Without the routes the system can't get in touch with the users. The **route** is essentially a URI that makes possible the communication with the outside world with a known URL-address. Basically a client gives a request for a determined route, which according the requirements, forwards it to a controller that processes the request. The /routes folder in the project's root directory contains the route files.

2.1 Some examples for routing in *routes/web.php* file

Route::get('route1', function () { return view('welcome'); });

In this example the 'Route' has a method 'get' that returns a 'welcome' view, which presents a web page. When a user visits the home page, he is taken to the 'welcome' page. This 'welcome' view is actually a PHP file: 'welcome.blade.php' in *resources/views* folder..

Route::post('route2', 'App\Http\Controllers\FileController@read');

[*] Corresponding author. E-mail address: subecz.zoltan@gamf.uni-neumann.hu

In this case the Route’s post method calls the read method of the FileController.

3 MVC architecture

Laravel is based on MVC architecture. **The Model-View-Controller (MVC)** is a design pattern that contains three main parts: model, view and controller. These components are so configured to handle the special development aspects of the application. The **View** component is intended for user interface logic. The **Controller** component receives input data and processes it. It works as interface between the model and view components. The **Model** component is the logic that has connection to the data related to the users. It is the main component of the pattern and represents the transferred data between the view and controller.

In a typical MVC web-framework (Figure 1.) the controllers handle the user requests and the business logic, the models represent the data and the views generate the output data [2].

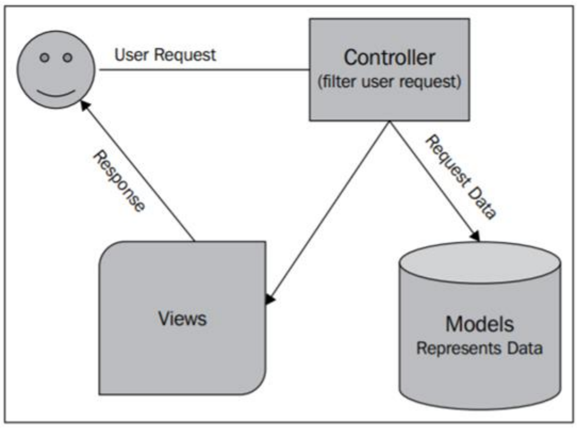

Figure 1. MVC architecture

4 Views and Templates

The views responsible for displaying the response from the controller in suitable format, usually as a webpage. They can be built easily with the **Blade template** language or simply with regular PHP scripts. The view is the most common object that return from the routes. The views get data from routes or controllers and inject them into a template, so they help to separate the business logic from the presentation logic.

In the next example you can find some Blade directives:

```
@extends("mainpage")
@section("content")
  <h1>Connect page content:</h1>
  <h2>{{ $data1}}</h2>
  Lorem ipsum dolor …
@stop
```

mainpage view:

```
  <body>
    <div class="header">
      @include("header")
    </div>
    <div class="content">
      @yield("content")
    </div>
```

```
</body>
```

The above example contains a Blade echo statement **{{……. }}**. Blade lets you build hierarchical layouts by allowing the templates to be nested and extended. The **@yield directives** act as placeholders for the different sections that a child view can populate and override. The **@section ... @stop directives** delimit the blocks of content that are going to be injected into the master template. Unlike some PHP templating engines, **Blade** does not restrict you from using plain PHP code in your templates. In fact, all Blade templates are compiled into plain PHP code and **cached** until they are modified, meaning Blade adds essentially zero overhead to your application

5 Controller

In an MVC framework the **controller** controls the dataflow between the datasets and views. This is a transport mechanism that takes the users to the presentation layer [2]. The controllers receive requests, process them and send appropriate responses. The controllers deal with such tasks as requesting data from database, handling received data from forms and saving data into the database. The controllers are inside the app/http/controllers directory.

An example for a controller file:

```
<?php
namespace App\Http\Controllers;
class TaskController extends Controller
{
   public function CallView()
   {
            $data = array('data1' => "example string",'data2' => 5);
            return view('view1', $data);
   }
}
```

In this example the TaskController has a CallView method with an array definition and passing this array to a view. Our controller has been created in the *app/controllers* directory which Laravel has created for us. There are two types of controllers in Laravel, **standard controllers** and **resource controllers**. Their job is to make sense of the incoming requests and to send an appropriate response.

5.1 RESTful Controller

The **REST (Representational State Transfer)** is a software architectural style that uses the subset of HTTP. It is used for creating interactive applications that use web-services. The web-services that use these principles are called RESTful. It provides such HTTP methods as POST, GET, PUT, and DELETE. Each URL represents an imaginary resource. Laravel simplifies the creation of REST APIs with its resource controllers. The resource controllers make easy to build RESTful controllers for resources [5].

6 Interacting with databases

6.1 Migration

The **migrations** are PHP scripts that are used for manipulate the structure and content of the database. A developer who works in a team can well synchronize the database tasks with colleagues. In this sense it operates as a version control. The migrations are supplied with time-stamps, so these are executed in correct order. Using migrations have the same database structure in consistent state [4].

We can create a migration file automatically with the next artisan command:
php artisan migrate:make create_users_table

For creating the users table we can use such a migration class:

```
class CreateUsersTable extends Migration
        {
            // Run the migrations.
           public function up()
           {
              Schema::create('userss', function (Blueprint $table) {
                 $table->id();
                 $table->string('first_name');
                 $table->string('last_name');
                 $table->string('login');
                   $table->integer('age');
                 $table->timestamps();
              });
           }
            // Reverse the migrations.
           public function down()
           {
              Schema::dropIfExists('workers');
           }
        }
```

To run all of your outstanding migrations, execute **migrate** Artisan command:
php artisan migrate

6.2 Database Seeding

We can populate our database with seeding helpers, rather than do it manually. With the seeder you can upload test data to the database using a simple way. You can find the seeder files in the database/seeders directory. We can generate a seeder with the next Artisan command:
php artisan make:seeder UserSeeder

As an example, let's create a seeder for the workers table and add a database insert statement to the run method:

```
<?php
namespace Database\Seeders;
use Illuminate\Support\Facades\DB;
use Illuminate\Database\Seeder;
class WorkerSeeder extends Seeder
{
   /**
    * Run the database seeds.
    *
    * @return void
    */
   public function run()
   {
      DB::table('workers')->insert([
         'first_name' => 'John',
         'last_name' => 'Hall',
         'login' => 'admin',
         'age' => '35',
            ]);
        }
    }
```

6.3 Query builder

Laravel provides the easy to use query builder that is intended for creating and executing database queries. The developer can use it for most of the database operations and works perfectly with Laravel’s all supported database systems. With the query builder we can connect methods into a chain for bulding complex queries.

7 Model and Eloquent

Models are associated with the resources in the application and work often with database records. They are classes that represent entities in the application, e.g. users, news articles or events.

Laravel ships the Eloquent, which is an efficient **object-relational mapper (ORM)** and the developer can define entities, assign them to related database tables and use PHP methods with it instead of raw SQL statements. We can perform with ORM efficient database queries, without writing any SQL instruction. The Eloquent operates as a model-layer in our applications [4].

In the next diagram we can see the main components of a typical web-application and the interactions between them. (Figure 2) (with the test cats data-table) [1]:

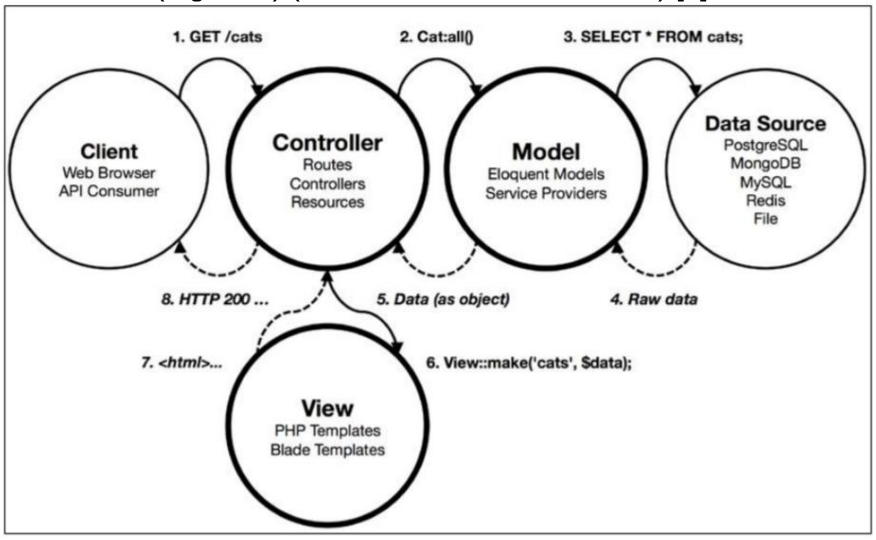

Figure 2. Interactions between all main constituents

7.1 Eloquent relationships

The database programmers are familiar with such relationships as one-to-one, one-to-many, many-to-one and many-to-many. Laravel Eloquent provides these concept in object-oriented environment. Besides them with Eloquent we can use other effective relations such as the polymorph relation, where the entities can have connections to several entities.

8 Authentication and Authorization

The user registration and login in web-applications are general and important features. . Laravel provides several different tools to use these functions safer and easier. The next figure contains the main view of the Laravel’s **authentication** service (Figure 3)

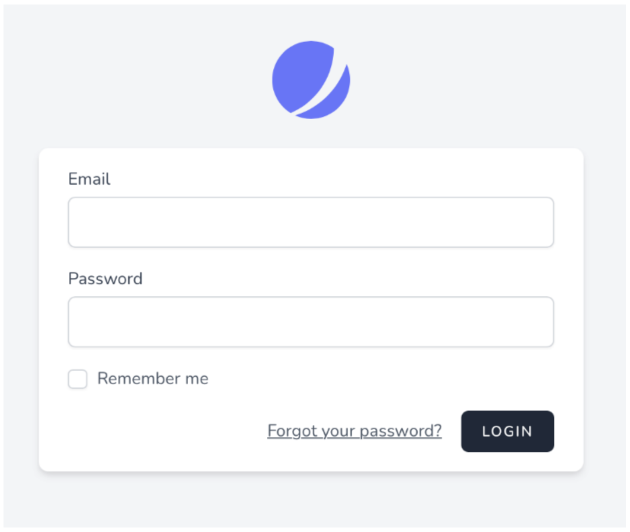

Figure 3. Laravel Authentication

Besides the authentication system we can use a simple method for the **authorization** of the user operations when the user tries to access a given resource. Even if the user is authenticated, it's not sure that he has right to read or write some Eloquent models handled by the application or database records. Laravel's authorization services give simple and organized methods to solve such tasks.

Laravel 8 ships Jetstream that gives a better starting point to new projects. It contains the next components: **two-factor authentication**; API support; session management; functions for login and registration; email verification [5].

9 Middleware

Middleware is used for filtering the HTTP-requests entering the application. The developer can use middleware for such tasks: authentication and authorization checks; session validation and session variable modification; reading, setting or modifying the header of the request and response; transaction logging; API calls, …[5].

10 Validation

Laravel's **Validator** class helps with full functionality the validation of e.g. forms and database models. The validator supports receiving input data; declaring special validation roles and declaring special validation messages.

This is a validation example with error messages in a blade form:

```
<form method="post" action="form5">
  @csrf
  @if (count($errors) > 0)
  <ul>
          @foreach ($errors->all() as $error)
                  <li>{{ $error }}</li>
          @endforeach
  </ul>
  @endif
  <h1>Form</h1>
  @if ($errors->has('email'))<strong>{{ $errors->first('email') }}</strong><br>@endif
  <label>E-mail:</label><input type="email" name="email"><br>
………………………..
</form>
```

And these are form validation rules in a controller file:

```
public function validateform(Request $request) {
   $this->validate($request,[
      'email'=>'required|min:8',
      'name'=>'required|max:15',
      'passw'=>'required',
      'age'=>'required',
   ]);
   // If the validation rules pass, the code will keep executing normally:
   return view('form5', compact('request'));
}
```

11 Securing the application

Before locating the application into an open environment, the developer has to consider several security questions. Laravel protects the application against several frequent attacks. Some of them: Cross-site request forgery (CSRF), Cross-site scripting (XSS), SQL injection, mass assignment vulnerability.

12 Some other useful Laravel concepts and the Laravel ecosystem

You can find detailed information about these useful Laravel concepts in Laravel main page: *Service Container, Service Providers, HTTP Requests, HTTP Responses, URL Generation, HTTP Session, Error Handling, Logging, Artisan Console, Broadcasting, Cache, Collections, Compiling Assets (Mix), Contracts, Events, File Storage, Helpers, HTTP Client, Localization, Mail, Notifications, Package Development, Queues, Task Scheduling, Database: Pagination, Testing, ..*

*In **Laravel ecosystem** you can find several other Laravel products and integrations, such as: Vapor, Forge, Envoyer, Horizon, Nova, Echo, Lumen, Sail, Spark, Valet, Mix, Cashier, Dusk, Santum, Scout, Socialite, Telescope, Jetstream,* …[5]

13 Version history

Laravel version schema uses these conventions: paradigm.major.minor. The major versions are released evefy year (in September) and the minor versions and patches can be released frequently. The minor versions can't include breaking changes, but the major releases can contain such changes [5]. The next table (Table 1.) contains information about the latest Laravel versions.

Table 1. Laravel version history

Version	*Release*
4	May 2013
5	February, 2015
6 (LTS)	September, 2019
7	March, 2020
8	September, 2020
9 (LTS)	September, 2021

14 Summary

In this paper I introduced the main characteristics this useful and rightly popular framework, with covering these topics: routing, MVC architecture, views and templates, controller, interacting with databases, model and Eloquent, authentication and authorization, middleware, validation, the service container and request lifecycle, securing the application and testing. I'd recommend Laravel framework for developers who want to create web-applications in a modern and fast way.

Acknowledgment

This research is supported by **EFOP-3.6.1-16-2016-00006 "The development and enhancement of the research potential at John von Neumann University"** project. The Project is supported by the Hungarian Government and co-financed by the European Social Fund.

References

[1] Martin Bean: Laravel 5 essentials, Published by Packt Publishing Ltd., 2015, ISBN: 978-1785283017

[2] Hardik Dangar: Learning Laravel 4 Application Development, Published by Packt Publishing Ltd., 2013 ISBN: 978-1783280575

[3] Jesse Griffin: Domain-Driven Laravel, Learn to Implement Domain-Driven Design Using Laravel, 2020, Apress; 1st ed. edition, ISBN: 1484260228, https://doi.org/10.1007/978-1-4842-6023-4

[4] Arda Kılıçdağı, H. İbrahim YILMAZ: Laravel Design Patterns and Best Practices, Published by Packt Publishing Ltd., 2014, ISBN: 978-1783287987

[5] Laravel - The PHP Framework For Web Artisans, https://laravel.com/docs/8.x, Accessed 25 Marc. 2021.

[6] Shawn McCool: Laravel Starter, Published by Packt Publishing Ltd., Leanpub, 2012, ISBN 978-1-78216-090-8

[7] Christopher John Pecoraro: Mastering Laravel, Published by Packt Publishing Ltd., 2015, ISBN: 978-1785285028

[8] Dayle Rees: Laravel: Code Bright, Publisher: Leanpub, 2013, ISBN: 978-1471777493

[9] Raphaël Saunier: Getting started with Laravel 4, Published by Packt Publishing Ltd., 2014, ISBN: 978-1783287031

[10] Sanjib Sinha: Beginning Laravel, Build Websites with Laravel 5.8, Howrah, West Bengal, India, 2017 ISBN: 978-1484225370, https://doi.org/10.1007/978-1-4842-2538-7

[11] Matt Stauffer: Laravel: Up and Running, A Framework for Building Modern PHP Apps, Publisher: O'Reilly Media, 2016, ISBN-13: 978-1491936085