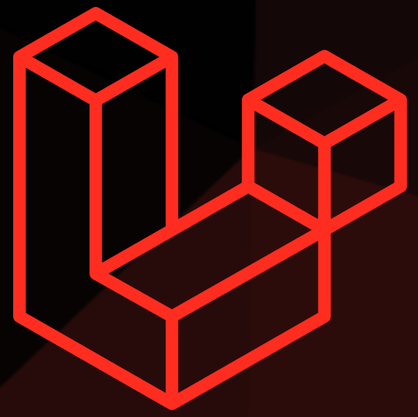


Laravel 11

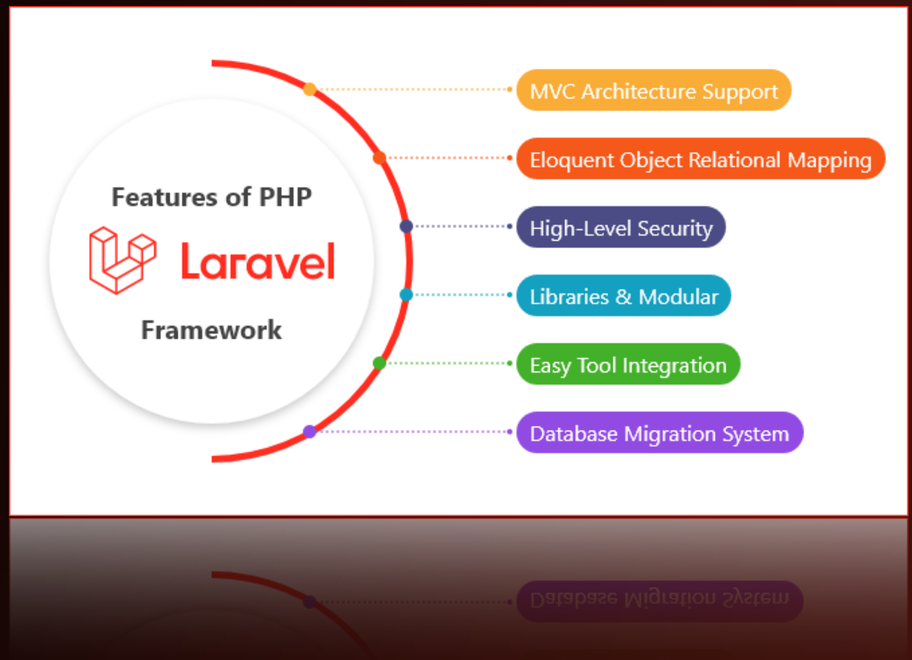
Reference/Basic Concepts

Oscar Fernando Aristizábal Cardona
Systems and Telecommunications Engineer

Laravel

What is ?

Laravel is a PHP web development framework designed to make it easier to create web applications with an elegant and simple syntax. It provides tools and resources that streamline common tasks like authentication, database management, and URL routing, allowing developers to focus more on the logic and design of their applications.



Laravel

Install

- php -v
- composer --version || composer -V
- composer create-project laravel/laravel nameapp
- cd nameapp/
- php artisan --version
- php artisan serve

Laravel

Rebuild App

> cd nameapp/

> composer install || > composer update

> touch .env

> cat .env.example > .env || > cp .env.example .env

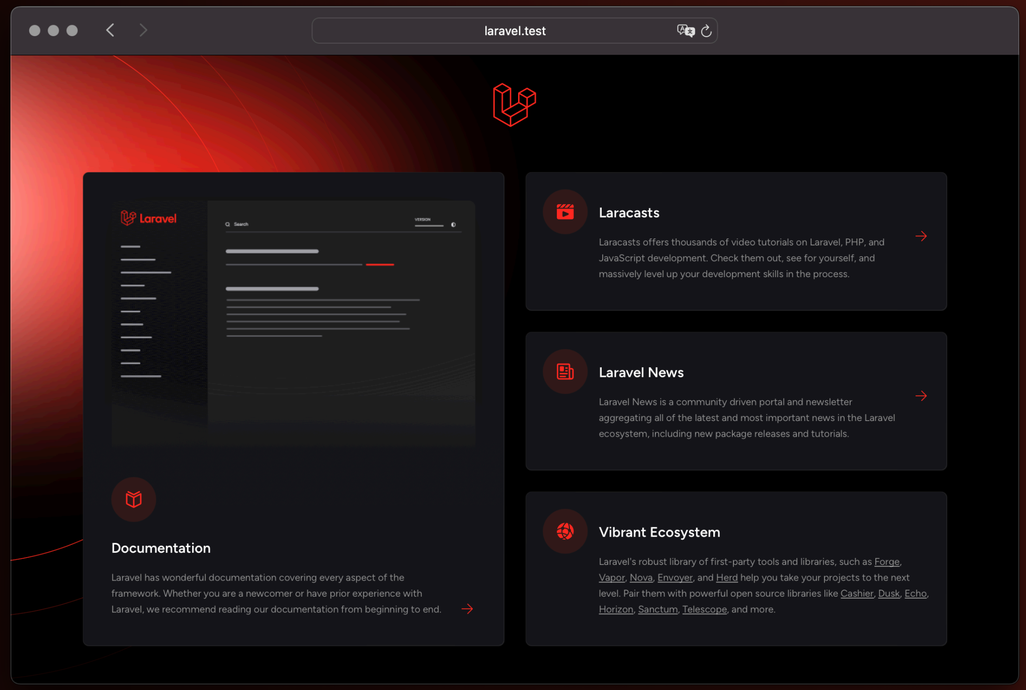
> code .env

> chmod -R 777 storage

> php artisan key:generate

> php artisan migrate

> php artisan serve



Laravel

Directory Structure

app/:

- Contains the core code of the application.
- Subdirectories include:
  - **Console/**: Contains Artisan commands.
  - **Exceptions/**: Contains the application's exception handling classes.
  - **Http/**: Handles HTTP requests, with subfolders for controllers, middleware, and form requests.
  - **Models/**: Contains the Eloquent models.
  - **Providers/**: Contains service providers that configure the application.

bootstrap/:

- Contains the application bootstrap files and the cache files generated during the compilation of routes and services.
- The **app.php** file initializes the framework.

config/:

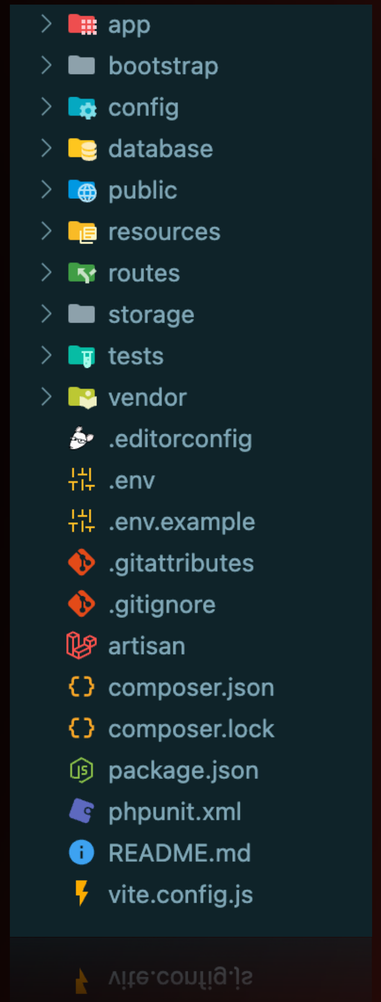
- Contains all the configuration files for the application.

database/:

- Houses database-related files.
- Subdirectories include:
  - **factories/**: Contains model factory definitions.
  - **migrations/**: Contains database migration files.
  - **seeders/**: Contains database seeders.

public/:

- The web server's root directory, which contains the front controller file **index.php**.
- Stores publicly accessible assets like images, JavaScript, and CSS.



resources/:

- Contains view templates, raw assets (Sass, JavaScript), and localization files.
- Subdirectories include:
  - **views/**: Blade templates for the application's HTML.
  - **lang/**: Localization files for different languages.

routes/:

- Contains all route definitions.
- Files include **web.php** for web routes, **api.php** for API routes, **console.php** for Artisan commands,.

storage/:

- Contains compiled Blade templates, file-based sessions, file caches, and other files generated by the application.
- Subdirectories include:
  - **app/**: For application-specific files.
  - **framework/**: For framework-generated files and caches.
  - **logs/**: For application log files.

tests/:

- Contains automated tests.
- Subdirectories include:
  - **Feature/**: For larger tests that interact with multiple parts of the application.
  - **Unit/**: For smaller, isolated tests of individual components.

vendor/:

- Contains the Composer dependencies.

Laravel

Config (DB)

Steps in terminal

- Open CMD / Terminal

> cd /Applications/XAMPP/bin/

> ./mysql -u root -p

> CREATE DATABASE nameapp;

- Open .env file in laravel directory “nameapp"

- Set DB_CONNECTION,
  DB_DATABASE,
  DB_USERNAME,
  DB_PASSWORD

Steps in phpmyadmin

- Open Web Browser

- Visit http://localhost/phpmyadmin

- Go to first tab "Data Bases”

- Enter "nameapp" and press “Create"

- Open .env file in laravel directory “nameapp"

- Set DB_CONNECTION,
  DB_DATABASE,
  DB_USERNAME,
  DB_PASSWORD

Laravel

Migrate (DataBase) ⚓

- php artisan make:migration create_categories_table
- php artisan migrate
- php artisan migrate:rollback
- php artisan migrate:rollback --step=3
- php artisan migrate:reset
- php artisan migrate:refresh
- php artisan migrate:fresh
- php artisan migrate:fresh --seed

create_categories_table

```php
public function up()
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->id();
            $table->string('name')->unique();
            $table->string('image')->default('images/no-category.png');
            $table->text('description');
            $table->timestamps();
        });
    }
```

Laravel

Models

› php artisan make:model User

User

```php
protected $fillable = [
    'document',
    'fullname',
    'gender',
    'birthdate',
    'photo',
    'phone',
    'email',
    'password',
    'role'
];

protected $hidden = [
    'password',
    'remember_token',
];

protected $casts = [
    'email_verified_at' => 'datetime',
    'password' => 'hashed',
];
```

Laravel

Seeds (DataBase)

- php artisan make:seeder UserSeeder
- composer dump-autoload
- php artisan db:seed
- php artisan db:seed --class=UserSeeder
- php artisan migrate:fresh --seed

DatabaseSeeder

```php
$this->call([
            UserSeeder::class,
            CategorySeeder::class
        ]);
```

UserSeeder

```php
// Add a record with Eloquent ORM
        $user = new User;
        $user->document  = 75000001;
        $user->fullname  = "Jeremias Springfield";
        $user->gender    = "Male";
        $user->birthdate = "1984-10-10";
        $user->photo     = "jeremias.png";
        $user->phone     = 3100000001;
        $user->email     = "jeremias@gmail.com";
        $user->password  = bcrypt('admin');
        $user->role      = "Admin";
        $user->save();

        // Add a record with Array
        DB::table('users')->insert([
            'document'   => 75000002,
            'fullname'   => 'John Wick',
            'gender'     => 'Male',
            'birthdate'  => '2000-07-06',
            'photo'      => '1709301260.png',
            'phone'      => 3100000002,
            'email'      => 'johnw@gmail.com',
            'password'   => bcrypt('12345'),
            'created_at' => now()
        ]);
```

Laravel

Factories (DataBase)

› php artisan make:factory CategoryFactory

DatabaseSeeder

```php
\App\Models\User::factory(2000)->create();
```

UserFactory

```php
return [
    'fullname' =>fake()->firstName() . " " . fake()->lastName(),
    'phone'    => fake()->phoneNumber(),
    'email'    => fake()->unique()->safeEmail()
];
```

Laravel

Factories (challenge)

Create a data factory that inserts 50 users with the following characteristics:

- FirstNames according to their gender (Male / Female)
- Random BirthDates between 1974 and 2004
- Photo URL - Store file in public directory

Laravel

RelationShips (Models)

hasMany

```php
// RelationShip: User has many games
public function games() {
    return $this->hasMany('App\Models\Game');
}
```

belongsTo

```php
// RelationShip: Game belongs to user
public function user() {
    return $this->belongsTo('App\Models\User');
}
// RelationShip: Game belongs to category
public function category() {
    return $this->belongsTo('App\Models\Category');
}
```

hasMany

```php
// RelationShip: Category has many games
public function games() {
    return $this->hasMany('App\Models\Game');
}
```

Laravel

CRUD (Tinker)

› php artisan tinker

CREATE

```php
use App\Models\User;
$usr = new User;
$usr->fullname = "Mikasa Ackerman";
$usr->email = "mikasa@gmail.com";
$usr->phone = 330987654;
$usr->birthdate = '1990-02-16';
$usr->gender = 'Female';
$usr->address = 'Cra Marley 26';
$usr->password = bcrypt('editor');
$usr->save();
```

READ

```php

$users = User::all();
$users = User::count();
$users = User::take(2)->get();
$users = User::limit(5)->get();
$users = User::limit(5)->offset(10)->get();

$user = User::find(103);
$user = User::where('email', 'homero@gmail.com')->first();

```

UPDATE

```php
$user = User::find(103);
$user->address = "Cll Paradise 26-27";
$user->email = "mikasa@shingeki.com";
$user->save();
```

DELETE

```php
$user = User::find(103);
$user->delete();

User::destroy(102);
```

Laravel

CRUD (Tinker)

› php artisan tinker

RELATIONSHIPS

```php
use App\Models\User;
$user = User::find(1);
# All games of this user.
$user->games;

use App\Models\Game;
$game = Game::find(1);

# All the user info associated with this game.
$game->user;
# Fullname of the user associated with this game.
$game->user->fullname;

# All the category info associated with this game.
$game->category;
# Name of the category associated with this game.
$game->category->name;
```

```php
$games = Game::all();
# Print all users who have games.
foreach ($games as $game) { echo $game->user . "\n\n";  }
# Print all fullnames of users who have games.
foreach ($games as $game) { echo $game->user->fullname . "\n";  }

# Print all categories that have games.
foreach ($games as $game) { echo $game->category . "\n\n";  }
# Print all names of categories that have games.
foreach ($games as $game) { echo $game->category->name . "\n";  }

```

Laravel

Routing

- php artisan route:list
- php artisan route:list -v
- php artisan route:list -vv

- php artisan route:clear
- php artisan route:cache

routes/web.php

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

```php
Route::get('/pets/show', function () {
    $pets = App\Models\Pet::all();
    //echo var_dump($pets);
    dd($pets->toArray()); // Dump & Die
});
```

```php
Route::get('/pets/view', function () {
    $pets = App\Models\Pet::all();
    return view('petsview')->with('pets', $pets);
});
```

Laravel

Routing (Challenge)

Create a route where 20 user records are displayed with their respective:

- Fullname
- Age in years
- How long it has been since the record was created

Example:

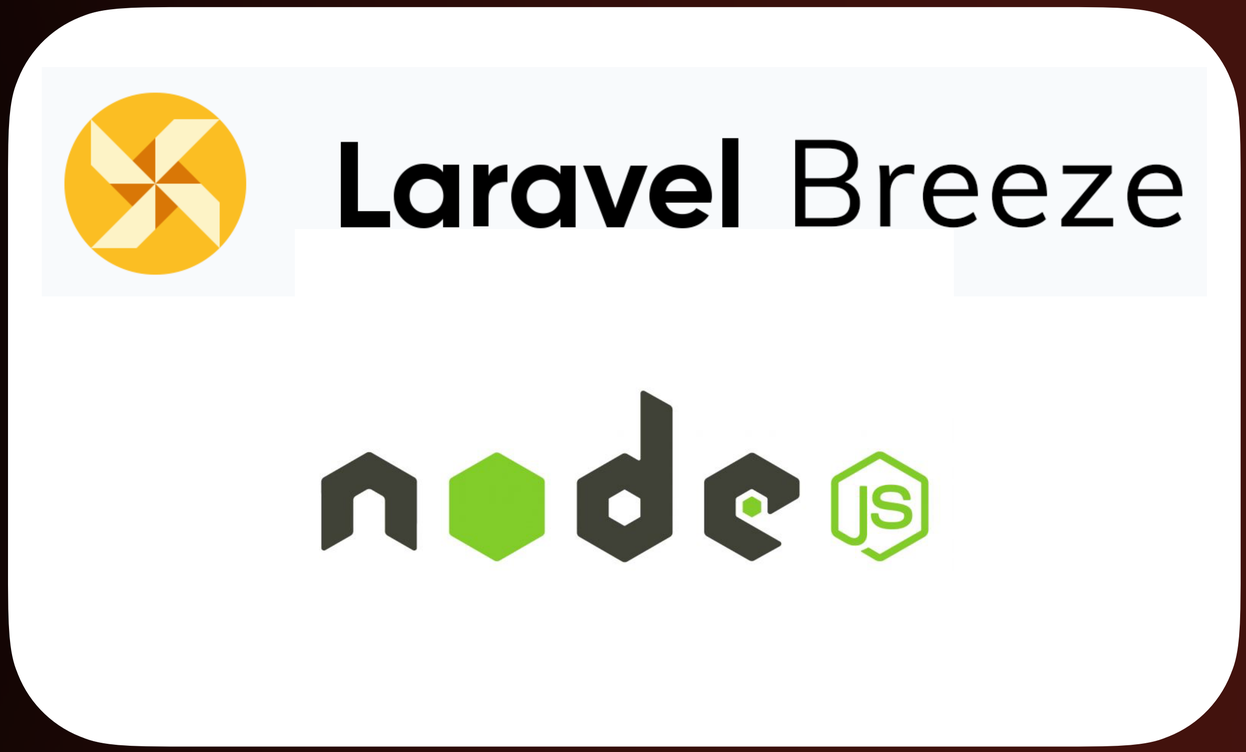
Arnoldo Suares Perez	45 years old	created 2 weeks ago
Carmen White	25 years old	created 40 minutes ago

Note: Without views

Laravel

UI for larval 11

- composer require laravel/breeze
- php artisan breeze:install
- npm install
- npm run dev

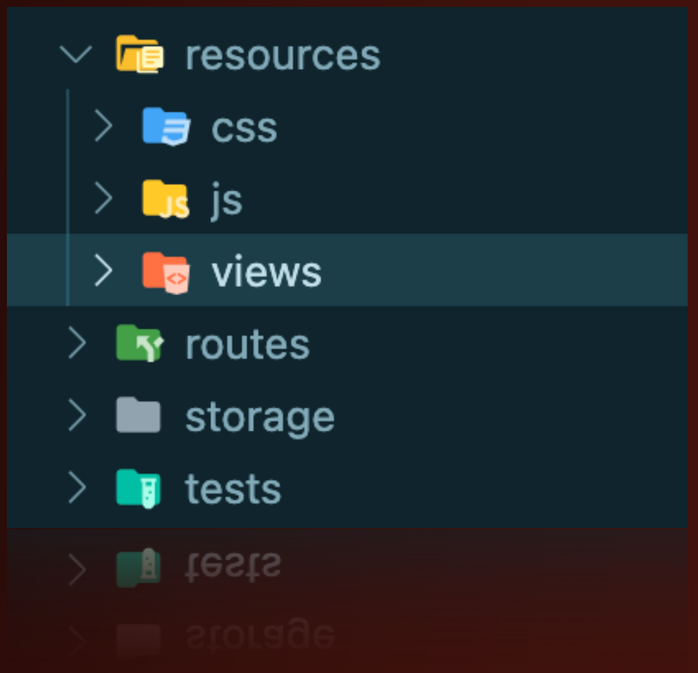


Laravel

Views

› The views contain the HTML generated by your application and separate the application logic from the presentation logic. The views are stored in the resources/views directory. A simple view might look like this:

```
1 <html>
2     <body>
3     <h1>Hello, {{ $name }}</h1>
4     </body>
5 </html>
```



Laravel

Blade

Directives

If Statements

Switch Statements

Loops

Including Subviews

Comments

PHP

Layouts Using Template Inheritance

CSRF Field

Method Field

Validation Errors

Laravel

Blade

Directives

If Statements

```blade
@if($user->isAdmin())
    <p>Welcome, Admin!</p>
@elseif($user->isModerator())
    <p>Welcome, Moderator!</p>
@else
    <p>Welcome, User!</p>
@endif
```

Switch Statements

```blade
@switch($user->role)
    @case('admin')
        <p>Welcome, Admin!</p>
        @break
    @case('moderator')
        <p>Welcome, Moderator!</p>
        @break
    @default
        <p>Welcome, User!</p>
@endswitch
```

Isset & Empty

```blade
@isset($records)
    <p>Records are available.</p>
@endisset

@empty($records)
    <p>No records found.</p>
@endempty
```

Laravel

Blade

Directives

Auth & Guest

```blade
@auth
    <p>Welcome, default user!</p>
@endauth

@auth('admin')
    <p>Welcome, Admin!</p>
@endauth

@guest
    <p>You are not logged in as a regular user.</p>
@endguest

@guest('admin')
    <p>You are not logged in as an admin.</p>
@endguest
```

Loops

```blade
@foreach($items as $item)
    <p>{{ $item }}</p>
@endforeach

@forelse($items as $item)
    <p>{{ $item }}</p>
@empty
    <p>No items found.</p>
@endforelse

@for($i = 0; $i < 10; $i++)
    <p>Iteration {{ $i }}</p>
@endfor

@while($condition)
    <p>Condition is true.</p>
@endwhile
```

Loop variables

```blade
@foreach($items as $item)
    @if($loop->first)
        <p>First iteration.</p>
    @endif
    @if($loop->last)
        <p>Last iteration.</p>
    @endif
    <p>{{ $item }}</p>
@endforeach
```

Laravel

Blade

Directives

Layout

```blade
@include('partials.header')
@includeIf('folder.filename')
@includeWhen($condition, 'partials.header')

@extends('layouts.app')

@section('content')
    <p>Content goes here.</p>
@endsection
```

Yield

```blade
// In layout file
<!DOCTYPE html>
<html>
<head>
    <title>@yield('title')</title>
</head>
<body>
    @yield('content')
</body>
</html>

// In child view
@extends('layout')

@section('title', 'Page Title')

@section('content')
    <p>Page content goes here.</p>
@endsection
```

Push & Stack

```blade
// In layout file
<head>
    @stack('scripts')
</head>

// In child view
@push('scripts')
    <script src="script.js"></script>
@endpush
```

Laravel

Blade

Directives

CSRF Field & Method

```blade
<form method="POST" action="/profile">
    @method('PUT')
    @csrf
    <!-- Form fields -->
</form>
```

Inject & Use

```blade
@inject('userModel', 'App\Models\User')

<div>
    Total Users: {{ $userModel->count() }}
</div>

@use('App\Models\Flight')
//Add an alias to it
@use('App\Models\Flight', 'FlightModel')

<p>Name: {{ $FlightModel->count() }}</p>
```

PHP

```blade
@php
    $counter = 1;
@endphp

<b> {{$counter}} </b>
```

Laravel

Blade

Validation Errors

```blade
@if (count($errors->all()) > 0)
@php $error = '' @endphp
@foreach ($errors->all() as $message )
    @php $error .= '<li>' . $message . '</li>' @endphp
@endforeach
<script>
$(document).ready(function () {
    Swal.fire({
        position: "top",
        title: "Ops!",
        html: `@php echo $error @endphp`,
        icon: "error",
        toast: true,
        showConfirmButton: false,
        timer: 5000
    })
})
</script>
@endif
```

Laravel

Controllers

- php artisan make:controller UserController
- php artisan make:controller UserController --resource
- php artisan make:controller UserController --resource --model=User

Route

```php
Route::resources([
        'users'      => UserController::class,
        'categories' => CategoryController::class
    ]);
```

Controller

```php
class UserController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        //$users = User::all();
        $users = User::paginate(20);
        return view('users.index')->with('users', $users);
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        return view('users.create');
    }
```

Laravel

Request

› php artisan make:request UserRequest

Controller

```php
public function store(UserRequest $request)
    {
        //dd($request->all());
```

Request

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

Laravel

Glossary

PHP: PHP is a widely-used open-source scripting language especially suited for web development. It is used to create dynamic web pages and applications.

Composer: Composer is a dependency management tool for PHP. It allows you to declare the libraries your project depends on and manages (installs and updates) them for you.

Framework: A PHP framework is a platform that provides a structured foundation and pre-built components for developing web applications. It simplifies and accelerates the development process by offering reusable code and tools.

PHP Artisan: PHP Artisan is the command-line interface included with Laravel. It provides a number of helpful commands for performing common tasks, such as database migrations, seeding, running tests, and generating boilerplate code, to streamline and enhance the development process.

Rebuild: Rebuild refers to recreating or reconstructing a system or component from scratch, starting with a base repository and ensuring that all necessary dependencies are installed and configured correctly.

Migrate/Migration: A migration in Laravel is a way to manage database schema changes. It allows developers to define and version-control changes to the database structure using PHP code, making it easy to deploy and share database changes across different environments.

Seed: A seed in Laravel is a way to populate the database with sample or default data. It allows developers to automate the process of inserting initial data into database tables, making it easier to set up a consistent testing or development environment.

Factory: A factory in Laravel is a tool used to generate fake data for testing purposes. It simplifies the process of creating mock database records by providing a convenient way to define the structure and attributes of model instances.

Relationships: Relationships in Laravel refer to the associations between different database tables/models. They define how one table/model is related to another, such as one-to-one, one-to-many, or many-to-many relationships. These relationships allow developers to efficiently query and manipulate related data in their applications.