Understanding the Laravel Framework

Steve Buchanan
CLOUD ARCHITECT

@buchatech | www.buchatech.com

Laravel

Overview

Core Concepts

Components

Architecture

Databases

Install

Upgrading

Core Concepts

Core Concepts

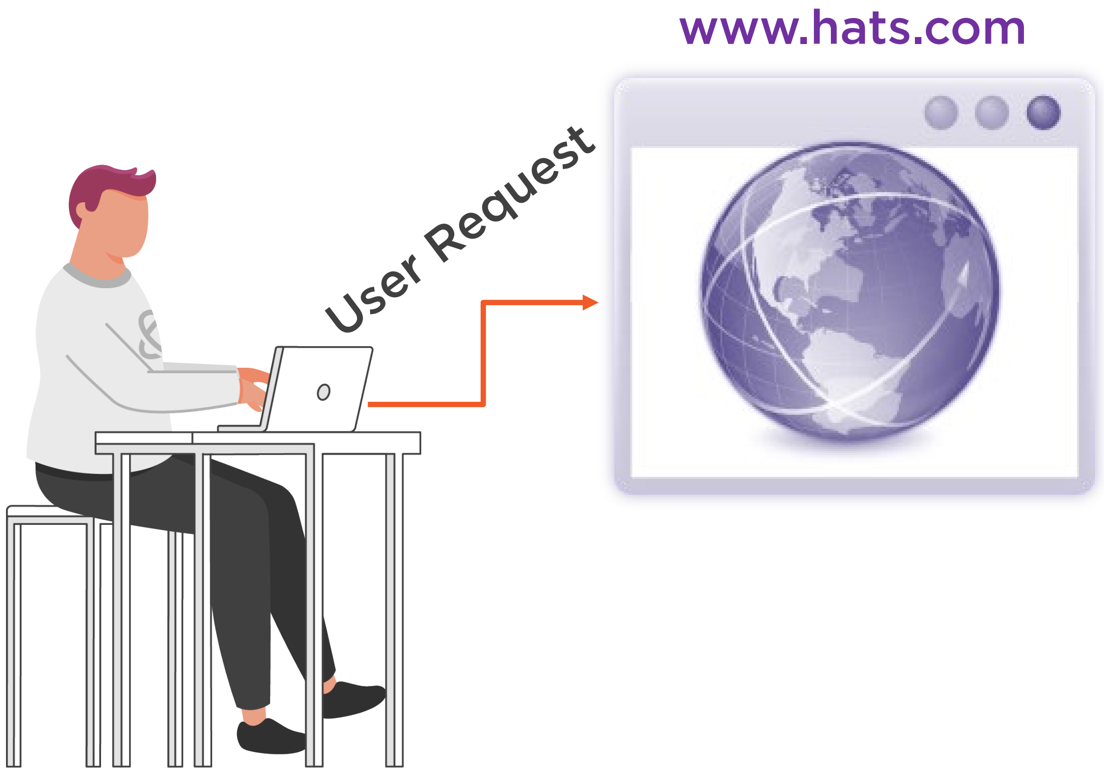
Routing	• Performs the mapping for your requests to a specific controller action • The routes/web.php file defines routes
Middleware	• A middle-man or interface acting in coordination between a request and a response • Filters incoming HTTP request before it is routed to particular controller for further processing
CSRF	• Stands for Cross-site request forgery • CSRF refers to an attack that makes the end-user perform unwanted actions within a web app that has already granted them authentication.
Controllers	• In MVC framework, the letter “C” stands for Controller • Responsible for handling the application logic • Takes on incoming HTTP requests & process them communicating with models and views, then return the results back to the browser
Models & Migrations	• In MVC framework, the letter “M” stands for Model. Models are used to interact with databases to retrieve, update, insert, and delete information • Migrations are like version control for your database. Migrations are changes to your database that are placed in the database/migrations directory with a timestamp and executed on the database when performed via artisan command
Eloquent ORM	• Programming technique known as PHP Active Record Pattern that makes interacting with Databases much easier

Core Concepts

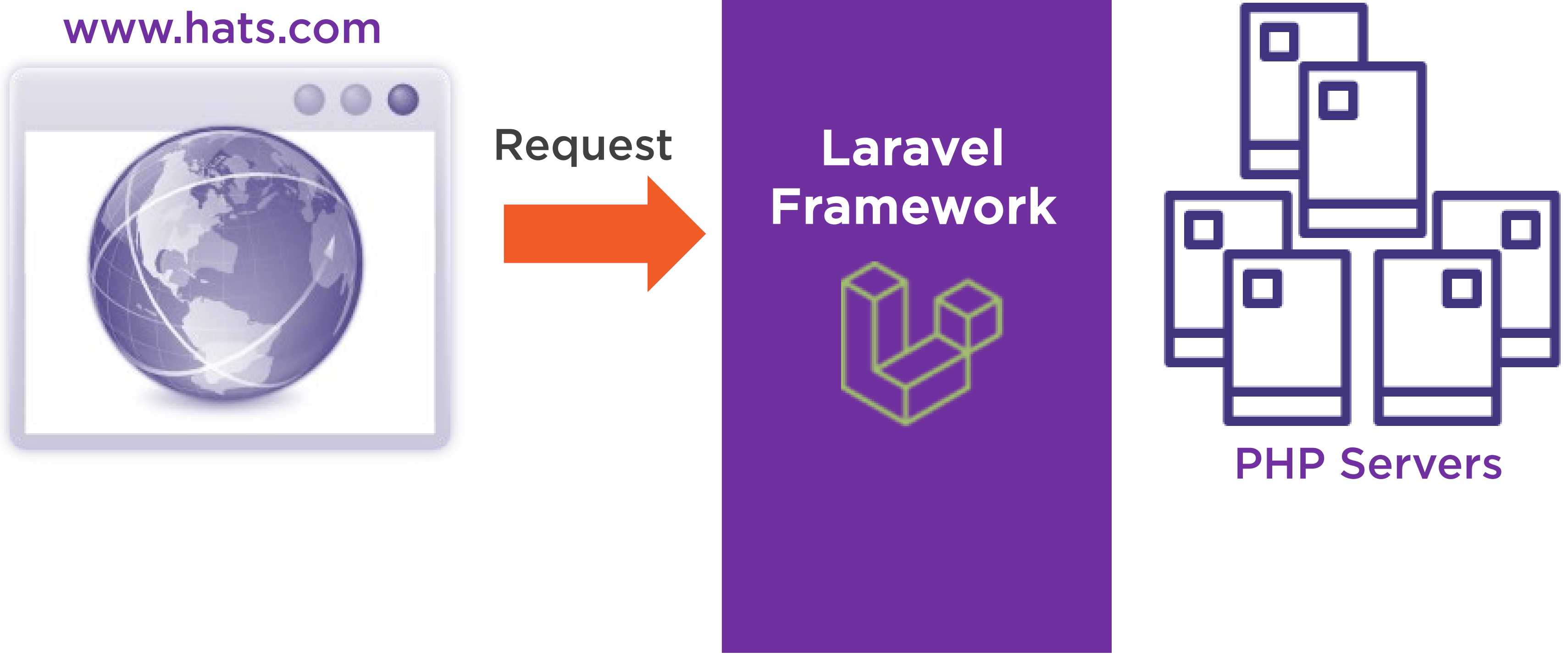
Requests	• HTTP requests from the client’s end
Responses	• Is what sent back to the user’s browser when a request made. These are handled by the controller
Views	• In MVC framework, the letter “V” stands for Views • Is used to to separate the presentation logic and business logic • Is the UI of an application and consists of HTML/CSS/JavaScript
Blade Templates	• A template engine. Views are built in this engine. Views created with the template engine have files that end in .blade.php and exist in the resources/views directory
Session	• Session is a parameter passing mechanism which enable us to store data across multiple requests

Lifecycle Overview – How Laravel Works

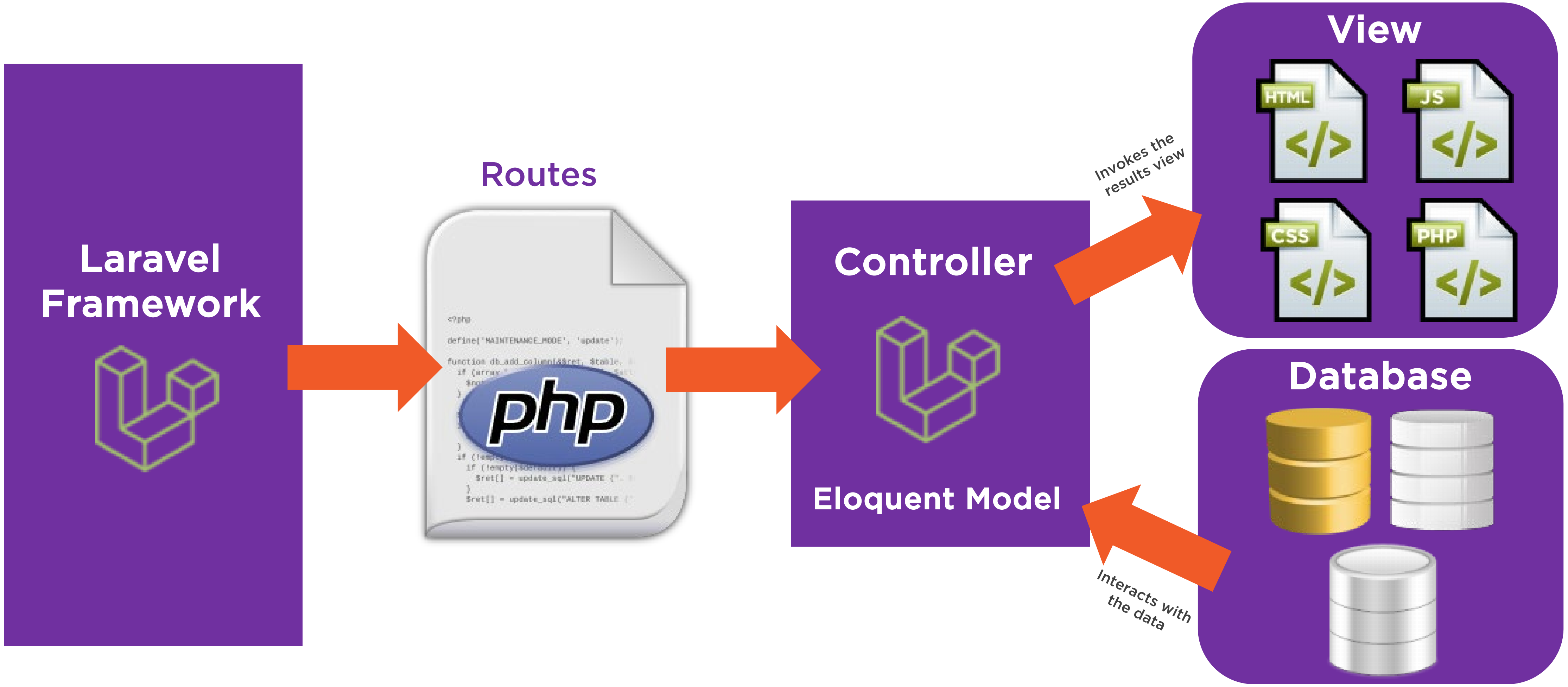
Lifecycle Overview – How Laravel Works



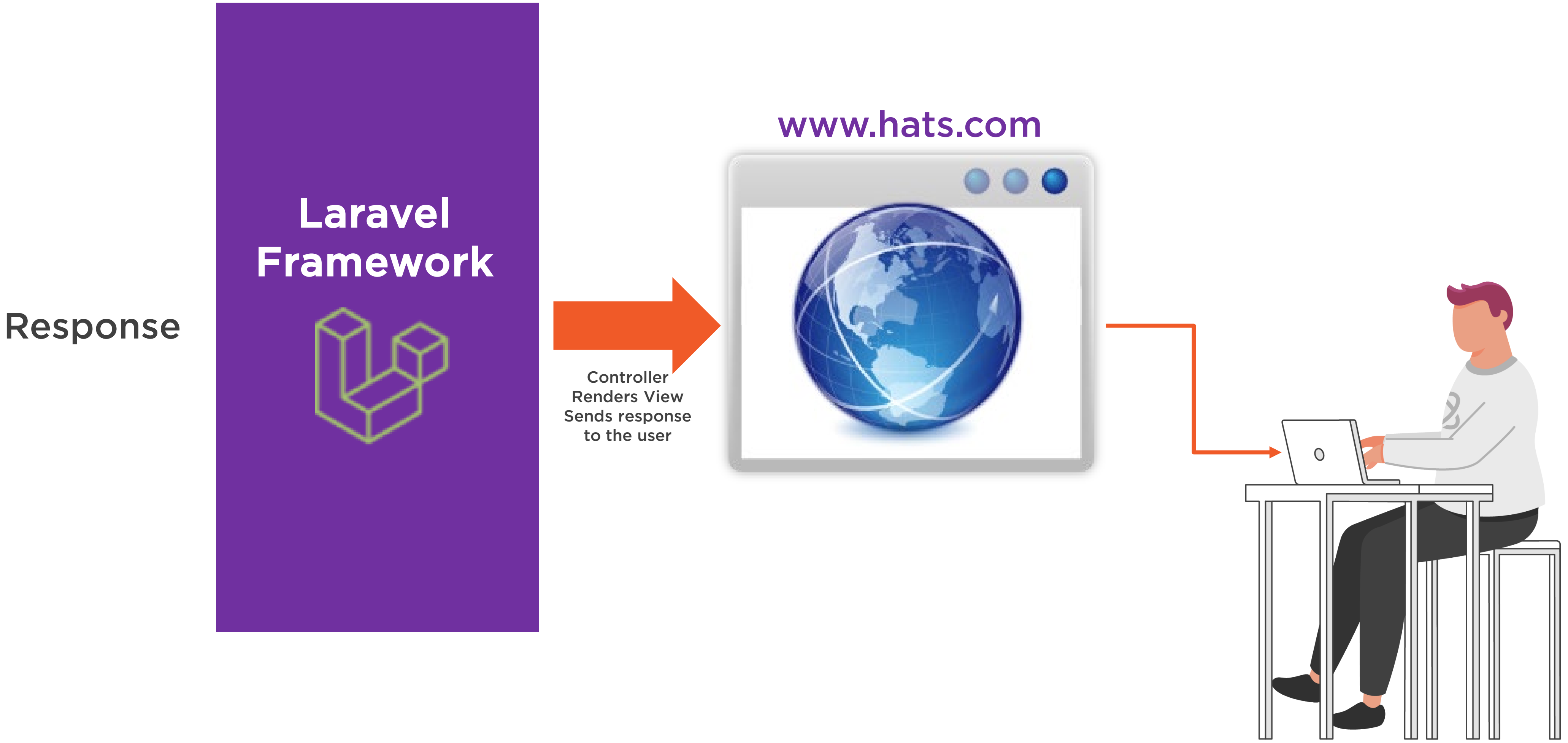
Lifecycle Overview – How Laravel Works



Lifecycle Overview – How Laravel Works



Lifecycle Overview – How Laravel Works

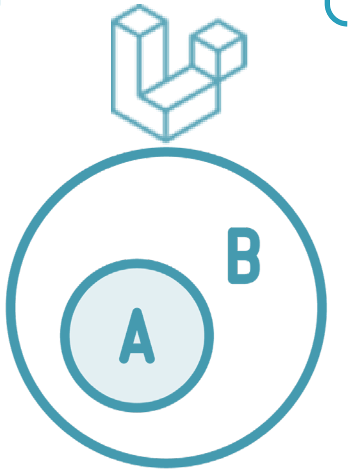


Components

Components

These are the various tech components that make up the Laravel Framework

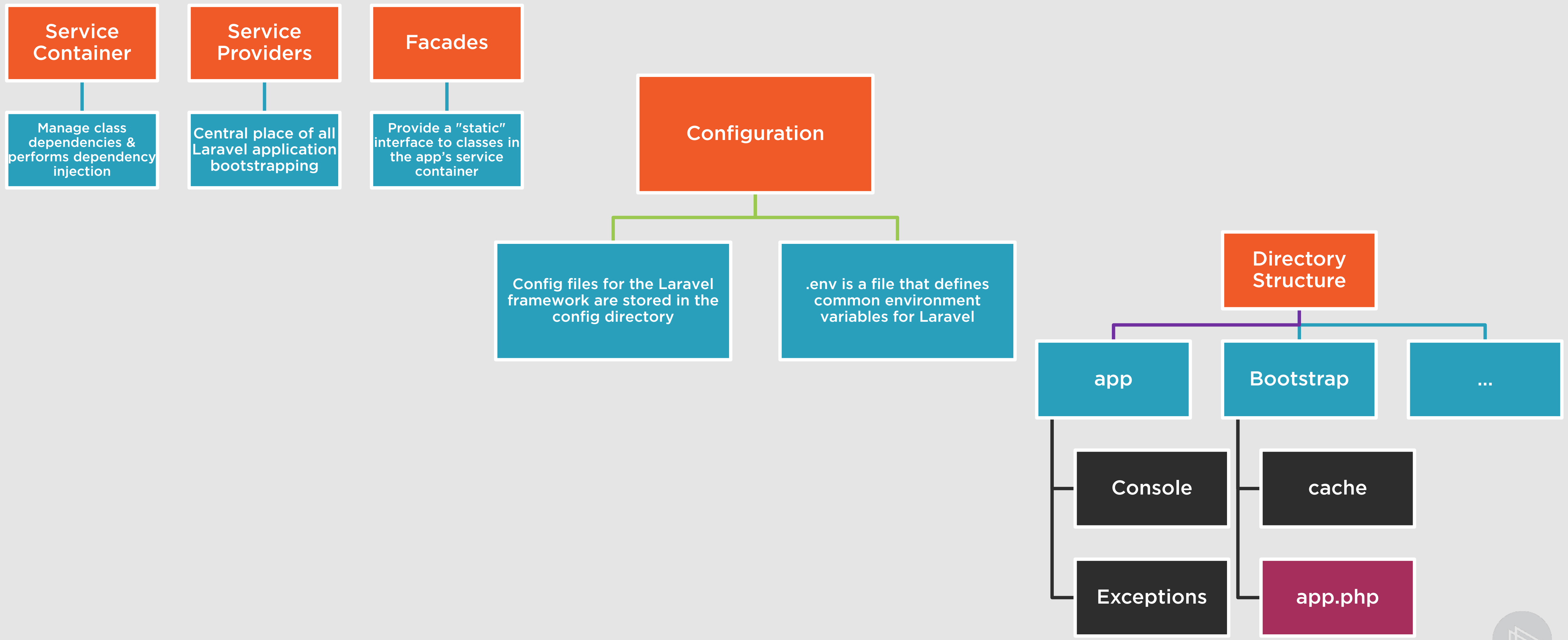
- PHP
- Artisan CLI
- Laravel Sail
- Composer
- Flysystem
- Guzzle HTTP client
- Helpers
- Authentication
- Authorization
- Encryption
- Mail (Symfony Mailer)
- Notifications (Vonage & Slack)
- Cache (Memcached or Redis)
- Command Scheduler



- API backend
- Query Builder
- Eloquent ORM
- Testing (HTTP Tests, Console Tests, Browser Tests)
- Packages

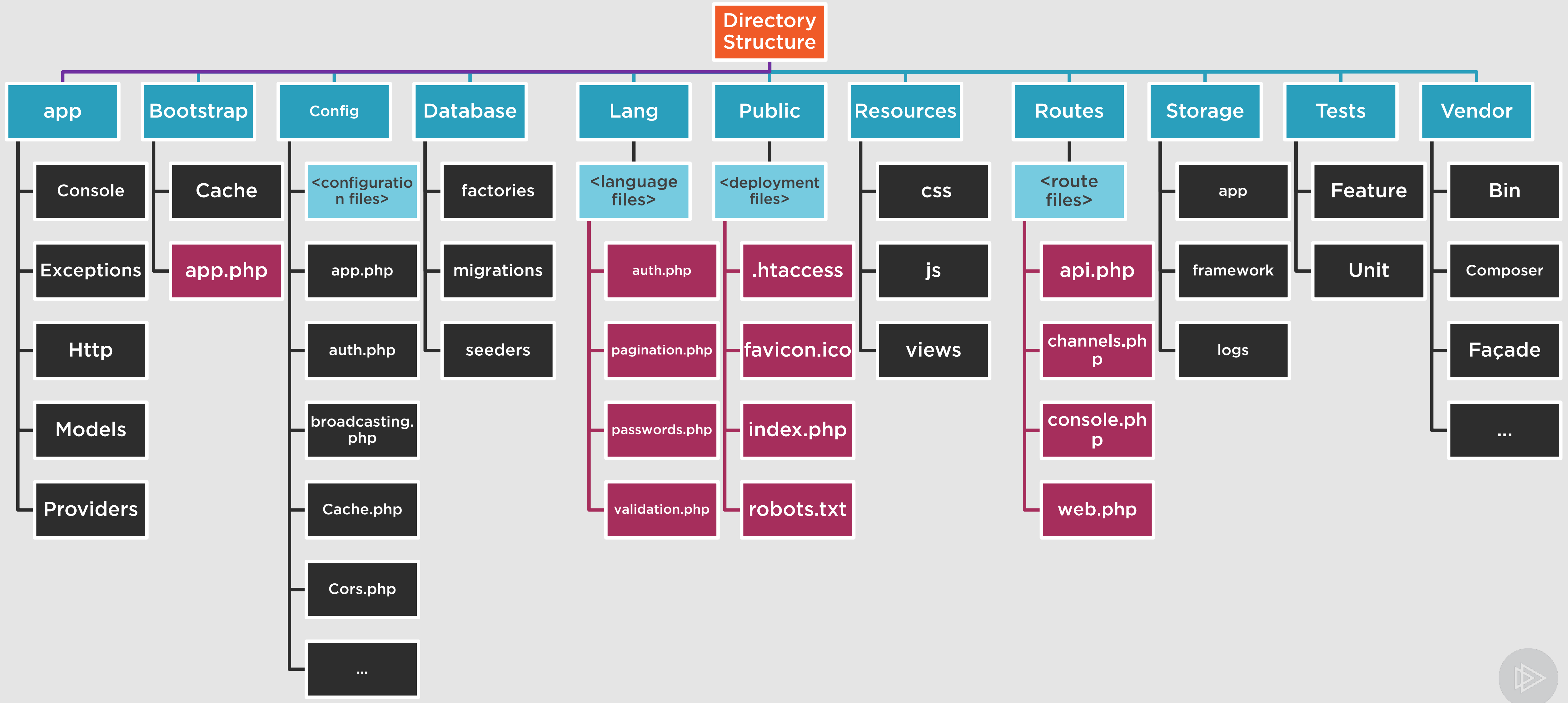
Architecture

Architecture



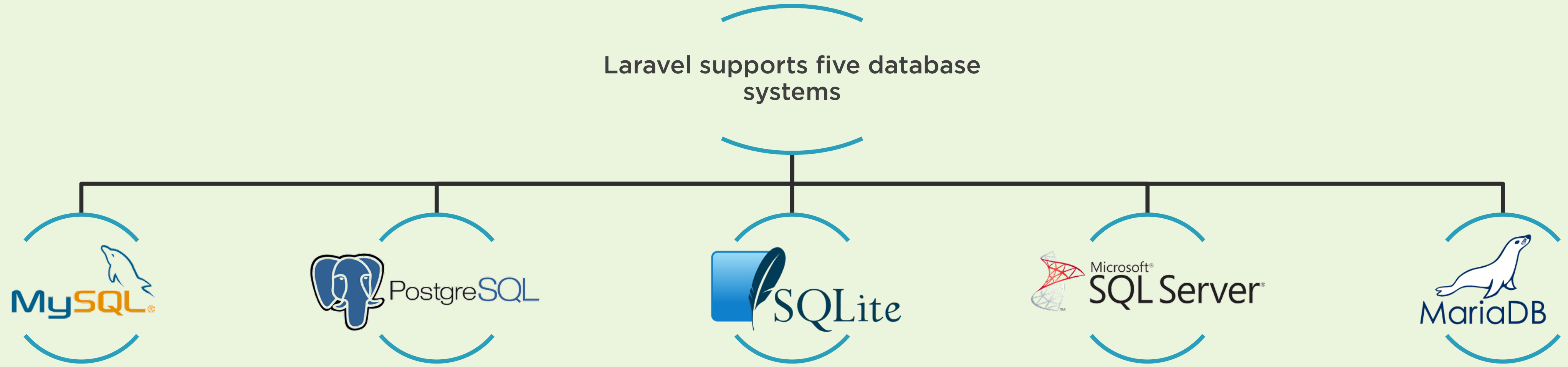
Service Container
Manage class dependencies & performs dependency injection
Service Providers
Central place of all Laravel application bootstrapping
Facades
Provide a "static" interface to classes in the app's service container
Configuration
Config files for the Laravel framework are stored in the config directory
.env is a file that defines common environment variables for Laravel
Directory Structure
app
Bootstrap
...
Console
Exceptions
cache
app.php

Architecture



Databases

What Databases are supported with Laravel


Laravel supports five database systems

Interacting with Databases via Laravel

With Laravel you have the option to work read / write to databases using

Raw Queries

Query Builder

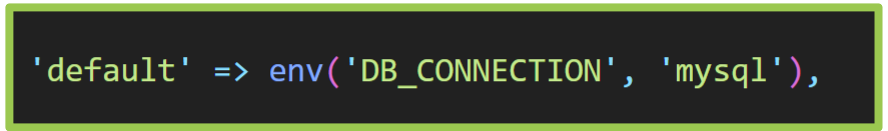
Eloquent ORM

How Laravel Connects to a Database

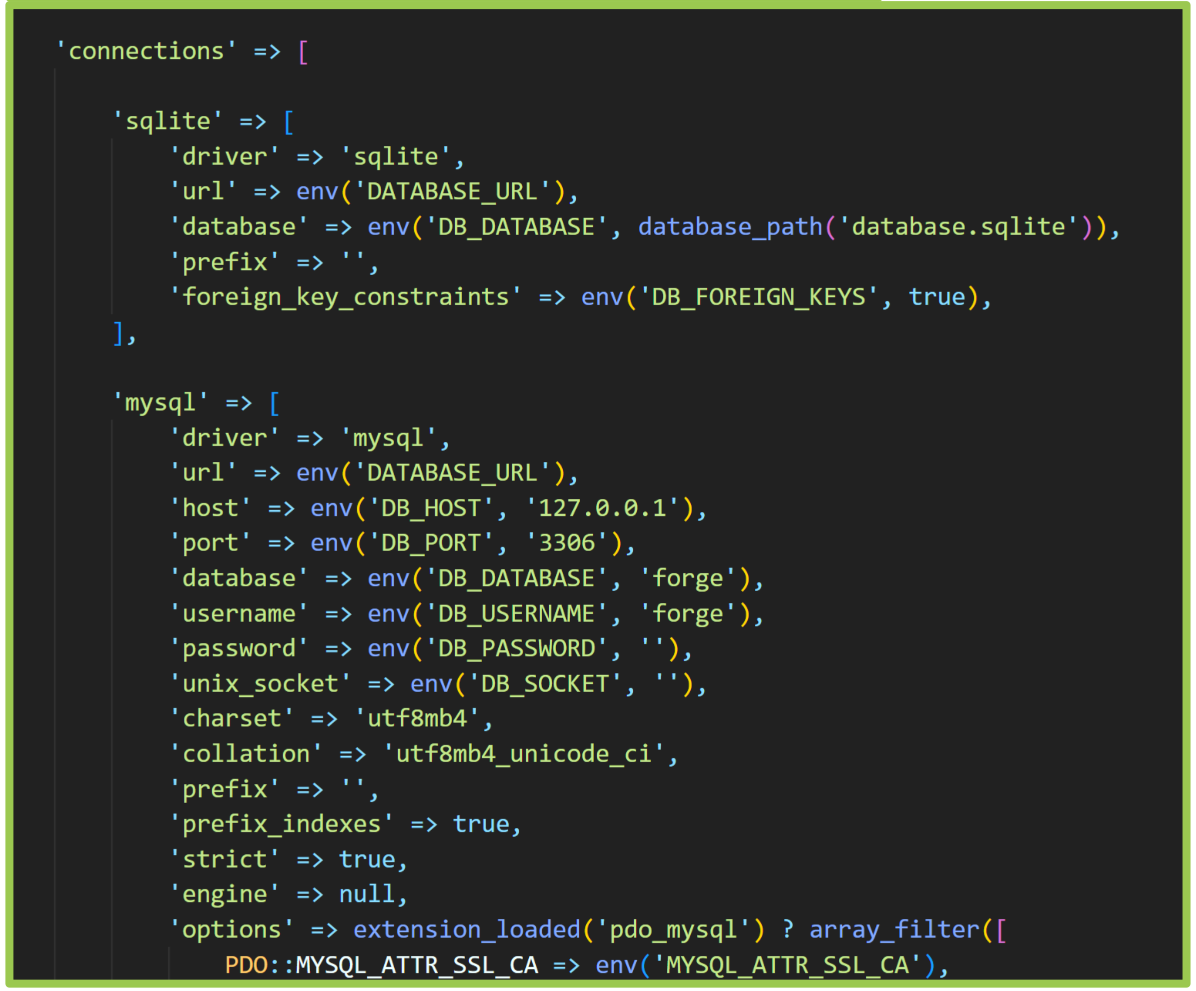
The database configuration file is **database.php** located in the **app/config** directory

You can specify all of your database connections and the default connection in this file

database connections are configured using multiple configuration values such as host, database, username, password



```php
'default' => env('DB_CONNECTION', 'mysql'),
```



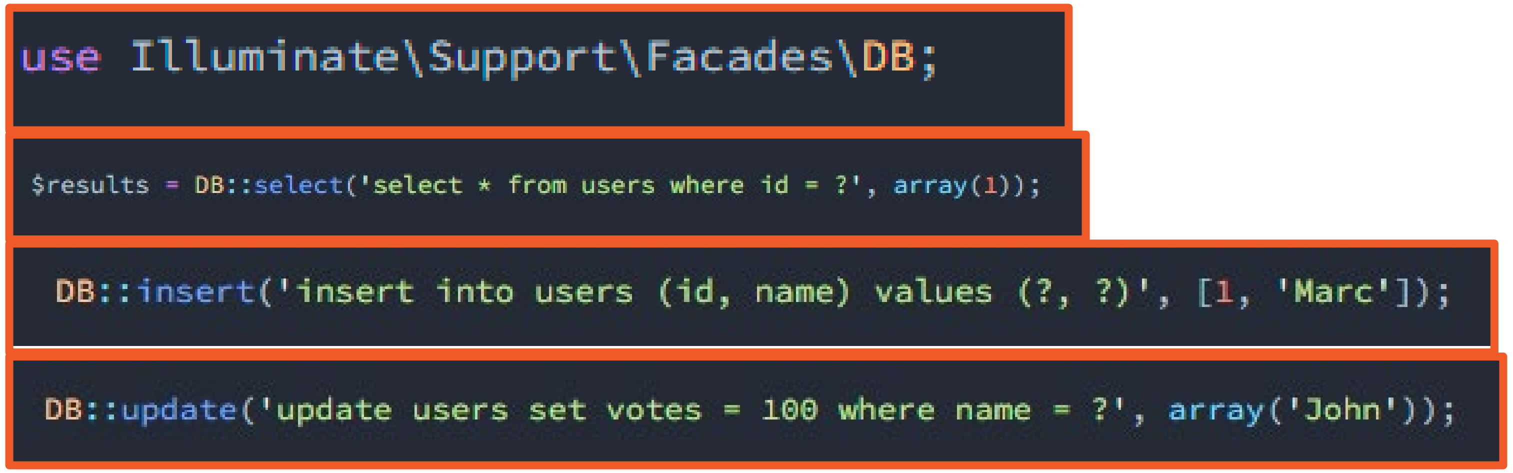
```php
'connections' => [

    'sqlite' => [
        'driver' => 'sqlite',
        'url' => env('DATABASE_URL'),
        'database' => env('DB_DATABASE', database_path('database.sqlite')),
        'prefix' => '',
        'foreign_key_constraints' => env('DB_FOREIGN_KEYS', true),
    ],

    'mysql' => [
        'driver' => 'mysql',
        'url' => env('DATABASE_URL'),
        'host' => env('DB_HOST', '127.0.0.1'),
        'port' => env('DB_PORT', '3306'),
        'database' => env('DB_DATABASE', 'forge'),
        'username' => env('DB_USERNAME', 'forge'),
        'password' => env('DB_PASSWORD', ''),
        'unix_socket' => env('DB_SOCKET', ''),
        'charset' => 'utf8mb4',
        'collation' => 'utf8mb4_unicode_ci',
        'prefix' => '',
        'prefix_indexes' => true,
        'strict' => true,
        'engine' => null,
        'options' => extension_loaded('pdo_mysql') ? array_filter([
            PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'),
```

How Laravel Connects to a Database

Connecting to multiple databases & running queries with Laravel is simplified with the Laravel query builder.

You can even query across multiple database connections i.e. you can use SELECT statements with one database, & a different database for INSERT, UPDATE, and DELETE statements

Here is an example of running SELECT, INSERT, & UPDATE queries using the DB façade in a PHP file:



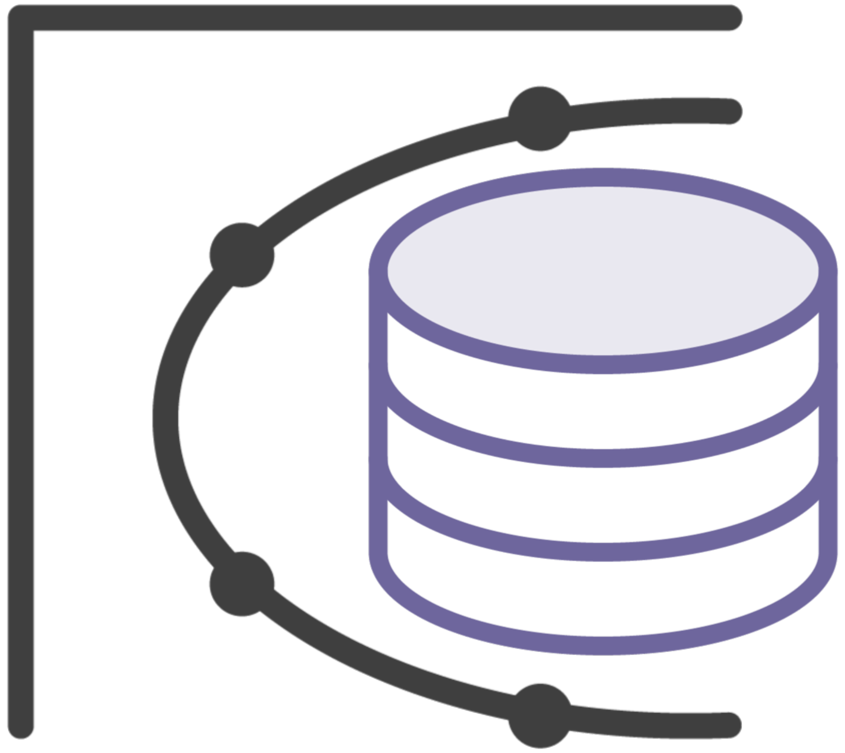
```php
use Illuminate\Support\Facades\DB;
```

```php
$results = DB::select('select * from users where id = ?', array(1));
```

```php
DB::insert('insert into users (id, name) values (?, ?)', [1, 'Marc']);
```

```php
DB::update('update users set votes = 100 where name = ?', array('John'));
```

Interacting with Databases via Laravel's Eloquent Model

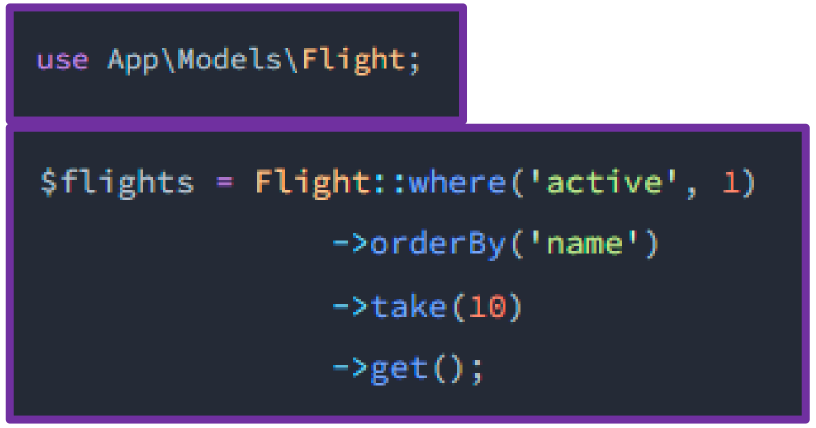


Eloquent is an object-relational mapper (ORM) that makes it easier to interact with a database

With Eloquent models you can retrieve, insert, update, & delete records from a database table

When using Eloquent, you have a models that are used to interact with associated database tables

Here is an example of a query using Eloquent:



```
use App\Models\Flight;
```

```
$flights = Flight::where('active', 1)
               ->orderBy('name')
               ->take(10)
               ->get();
```

Laravel & Redis

Redis is an open source key-value store used for caching

Redis can contain strings, hashes, lists, sets, & sorted sets

We can utilize & interact with Redis using Laravel's Redis façade

We can use Redis commands with the facade in our PHP files the façade passes these commands directly to Redis

Install

Supported OS’s

Laravel is typically run locally or....

- Browser based IDE
- Laravel Homestead
- Laravel Sail

Laravel is supported on these operating systems:

- MAC OS
- Windows
- Linux

Requirements

- PHP >= 8
- BCMath PHP Extension
- Ctype PHP Extension
- cURL PHP Extension
- DOM PHP Extension
- Fileinfo PHP Extension
- JSON PHP Extension
- Mbstring PHP Extension
- OpenSSL PHP Extension
- PCRE PHP Extension
- PDO PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension
- Composer

Laravel Install

There is no actual Laravel install

The Laravel install is creating an app

Creating an app is making a new directory for the app and populating it with the Laravel framework directories and files

Install via Sail

-Running Laravel in Docker containers can easily be done using Sail

-Sail is a light-weight command-line tool used for working with Laravel's default Docker stack

-The default Sail stack consists of mysql, redis, meilisearch, mailhog, and selenium

```
# Create and populate new app
directory

curl -s
"https://laravel.build/myapp"
| bash

# Navigate to the app
directory

cd myapp

# To start the Containers
run:

 ./vendor/bin/sail up –d

# To stop the Containers run:

./vendor/bin/sail stop
```

◀ Code / steps for creating a new Laravel application via Sail

Install via Composer

If your local machine already has PHP & Composer installed, you can create a new Laravel app using Composer directly

```
# Create and populate new app
directory

composer create-project
laravel/laravel myapp

# Navigate to the app
directory

cd myapp

# To start the Laravel app:

php artisan serve
```

◀ Code / steps for creating a new Laravel application via Composer

Install via Laravel Installer

If you are running a stack such as LAMP (Linux, Apache, MySQL, PHP) locally Laravel installer is an option for creating a new app

One major difference is needing to Install the Laravel Installer as a global Composer dependency

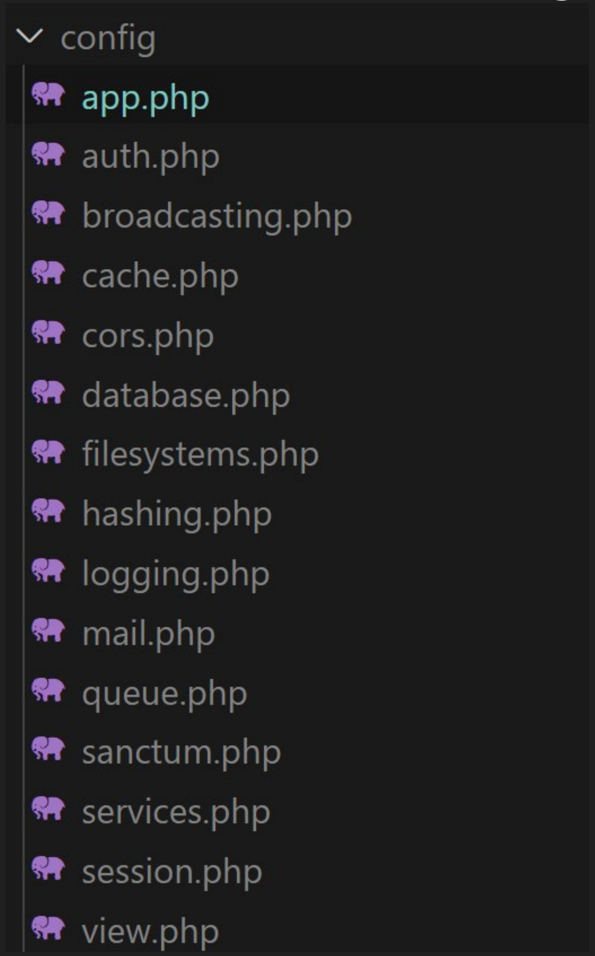
As well as setting Composer's system-wide vendor bin directory in your $PATH

```
# Set Laravel Installer as a
global Composer dependency

composer global require
laravel/installer

# Create and populate new app
directory

laravel new myapp

# Navigate to the app
directory

cd myapp

# To start the Laravel app:

php artisan serve
```

◀ Code / steps for creating a new Laravel application via Laravel Installer

Laravel Framework Configuration

All of the configuration files for the Laravel framework are stored in the config directory



```
config
  app.php
  auth.php
  broadcasting.php
  cache.php
  cors.php
  database.php
  filesystems.php
  hashing.php
  logging.php
  mail.php
  queue.php
  sanctum.php
  services.php
  session.php
  view.php
```

Laravel typically does not need additional configuration out of the box

Laravel App Configuration

The app.php file is the main configuration file for the app. It contains several options some may want to adjust such as:

- app name
- env - i.e. dev, prod etc
- debug mode
- url - of the app
- timezone
- locale - controlling the language
- providers

Laravel Environment Configuration

The .env file is the Environment configuration file. It is in the root of your app directory

- **By default Laravel has an example config file:**
  - .env.example

This file can be used to have different Configurations based on the environment where the app is running i.e.:

- local computer
- dev
- Prod

- In a shared dev team you can have different config files with the right values per environment for devs to use i.e.:
  - .env.dev
  - .env.prod

.env common values

- The common values in the .env are retrieved from various Laravel config files in the config directory using Laravel's env function

Common values in the .env file are:

- APP_NAME=myapp
- APP_ENV=local
- APP_DEBUG=true
- APP_URL=http://myapp
- DB_CONNECTION=mysql
- DB_HOST=mysql
- REDIS_HOST=redis
- REDIS_PASSWORD=null
- AWS_ACCESS_KEY_ID=
- AWS_SECRET_ACCESS_KEY=
- AWS_DEFAULT_REGION=us-east-1

Laravel App Deployment

Laravel Apps can be deployed to:

Self hosted web server (i.e. IIS, Apache, NGINX) or Containers i.e. on Kubernetes	Heroku	Azure
AWS	Digital Ocean	Laravel Forge / Vapor

Upgrading

How to upgrade Laravel

Upgrade Tips:

- Do not upgrade prod first. Upgrade dev first
- Verify your back-ups of your app before any upgrade
- After upgrade run your tests on your Laravel app & check critical pages making sure that nothing broke

How to upgrade Laravel

The CORE of the upgrade is about UPGRADING DEPENDENCIES used in the Laravel Framework

-Upgrade to the latest PHP version before you upgrade your app

-Next upgrade your app's composer.json file:

- -In the composer.json file change the laravel/framework & nunomaduro/collision lines as follows:

  - change: `laravel/framework package from ^8.12 to ^9.0
  - change: nunomaduro/collision from ^5.0 to ^6.1

  - then replace facade/ignition with "spatie/laravel-ignition": "^1.0"

Next run:

- composer update

How to upgrade Laravel

Additional high impact dependencies used in the Laravel Framework may also need to be upgraded

These items depend on what your application does & does not use

Upgrades will differ on a case by case basis. It is recommended to check the "high impact" list here for your specifics:

https://laravel.com/docs/9.x/upgrade?ref=hackernoon.com#high-impact-changes

How to upgrade Laravel

An alternative to upgrading Laravel yourself is to outsource it to Laravel Shift. Laravel Shift will automate your Laravel upgrade. More info at:

DETOUR

laravelshift.com

Summary

In this module we covered:

- The Core Concepts of Laravel & walked through how Laravel works
- The Components of Laravel & its Architecture
- What Databases are supported with Laravel & how Laravel works with them
- How to Install Laravel & how to Upgrade it

Why this is important:?

- In this module we covered the many aspects of Laravel to give you a solid understanding of the framework. This gives you a clearer picture of how Laravel works helping you understand how you can leverage it going forward