**Laravel 5.8**

Laravel support mvc architecture

MVC :model, view and controller

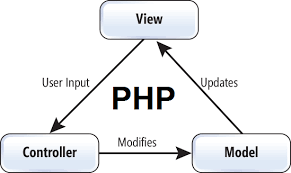
Model :

Model support db connection

View : support gui part of application

Controller : support logic or coding part of application

**MVC architecture :**

****

**Laravel installations :**

**How to install Laravel :**

Step1 :

Laravel.com

Step 2:

<https://laravel.com/docs/5.8>

step 3:

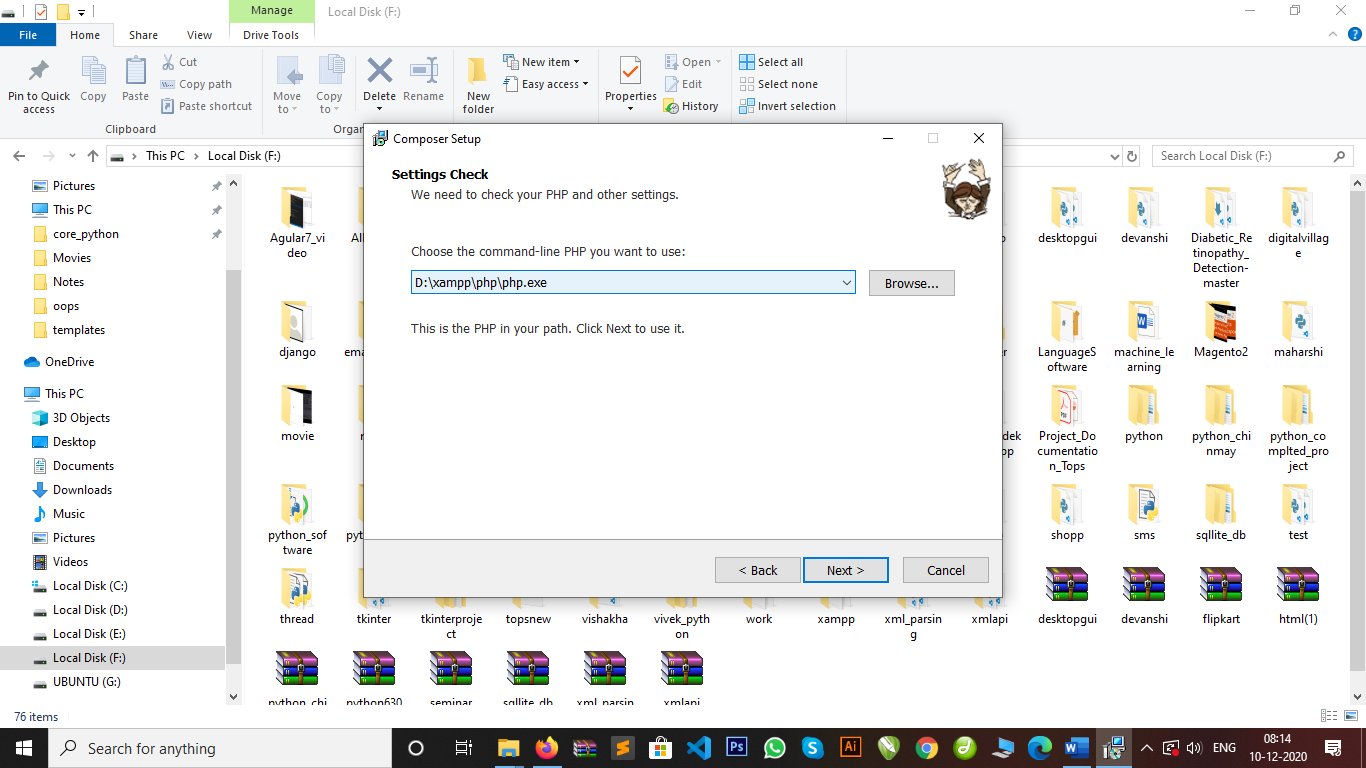
download composer

Composer :

<https://getcomposer.org/download/>

composer is a dependency manager i.e used for create or install Laravel without comoposer we cant install Laravel

step 4: composer will configure with xampp

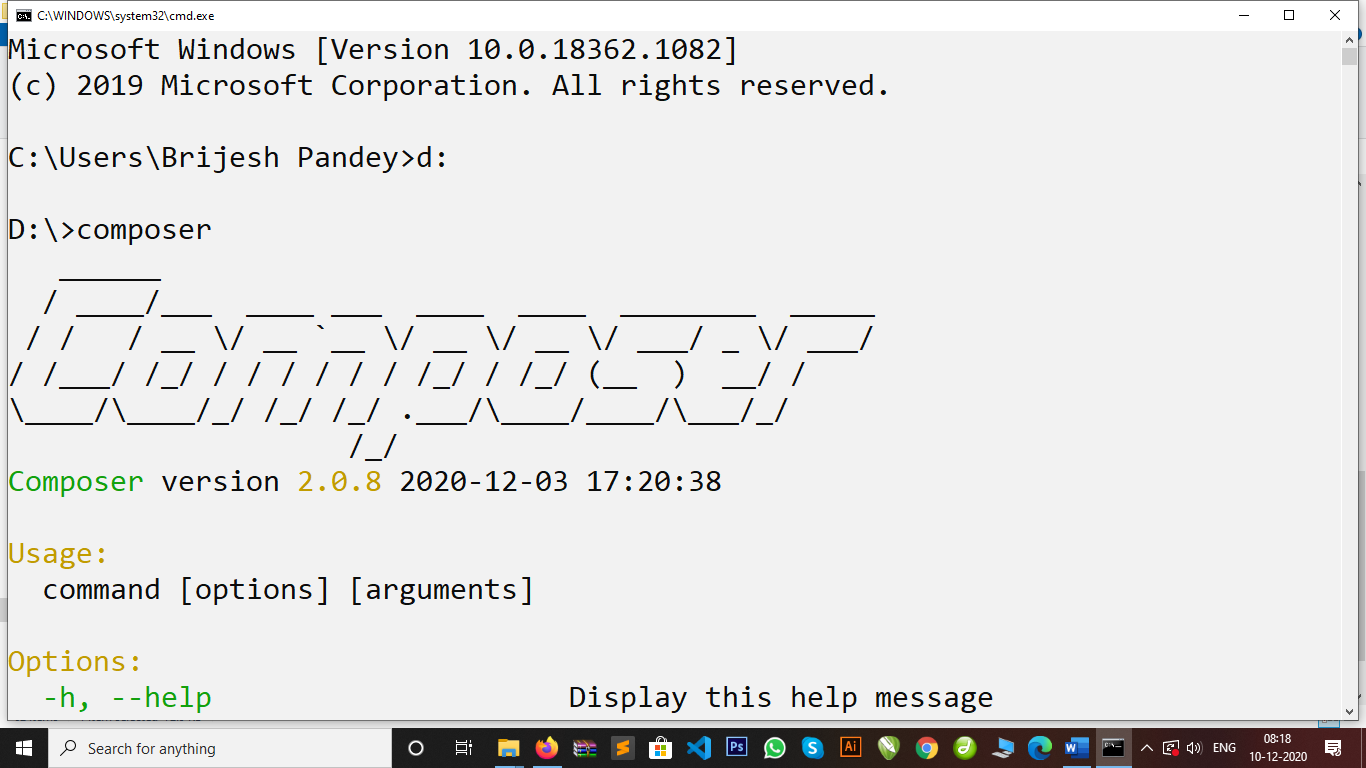


dependency manager :

composer is used to install all the dependency of Laravel or library and components of Laravel.

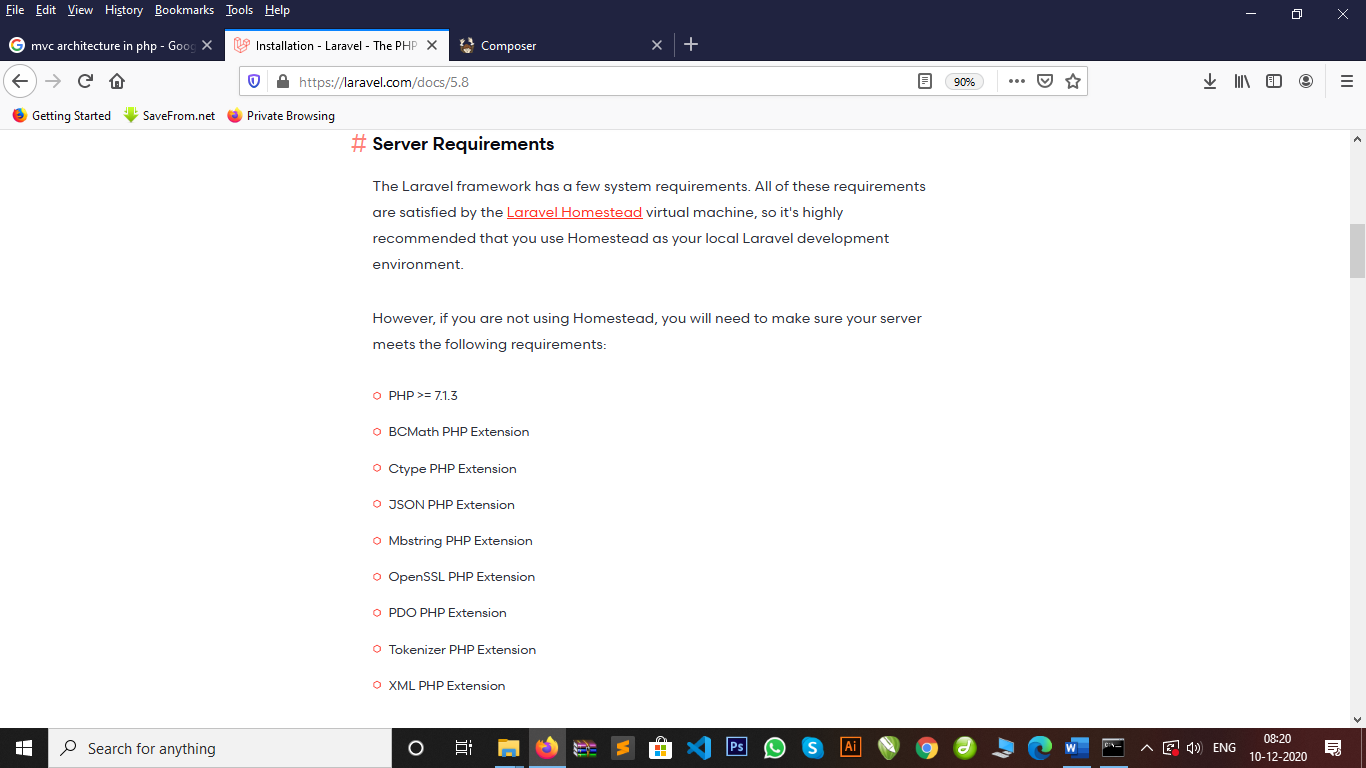
Step :5

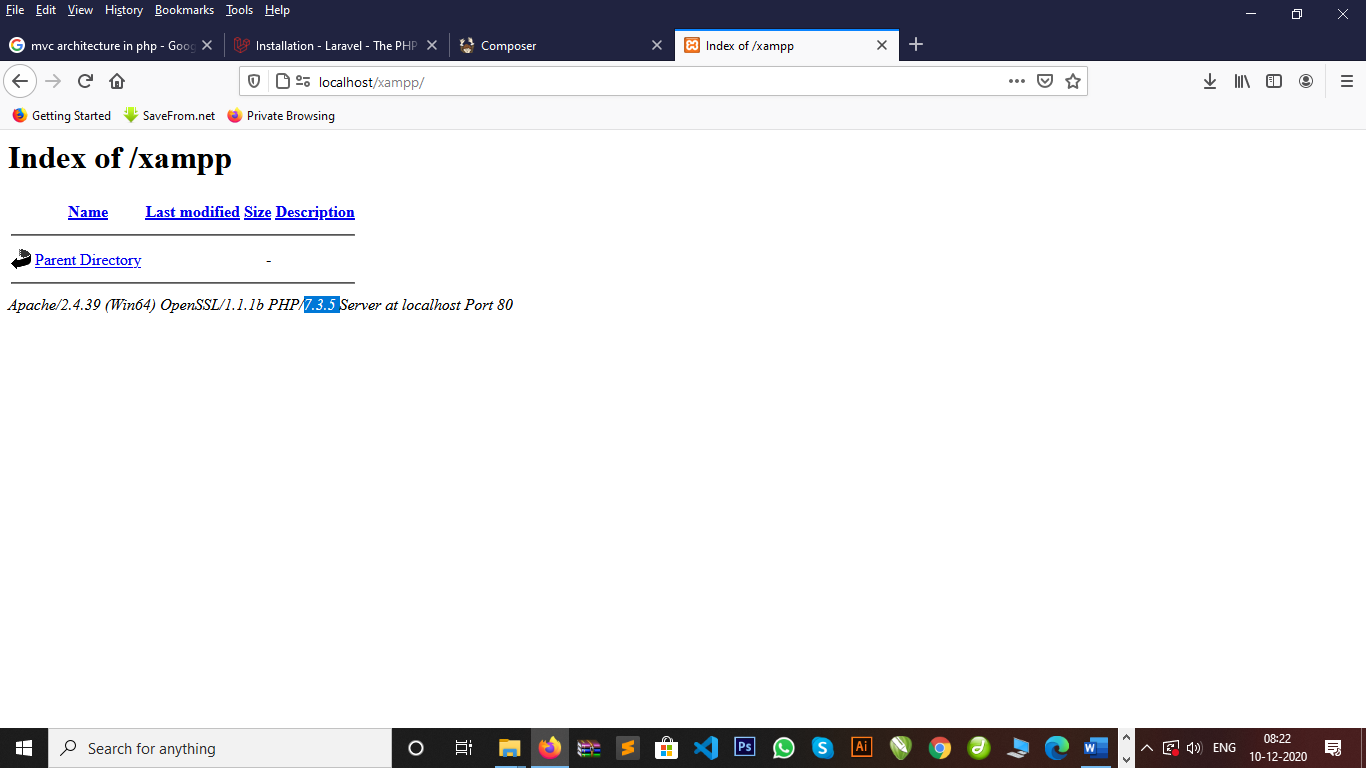
How to check composer install or not



Step 6:

System requirements



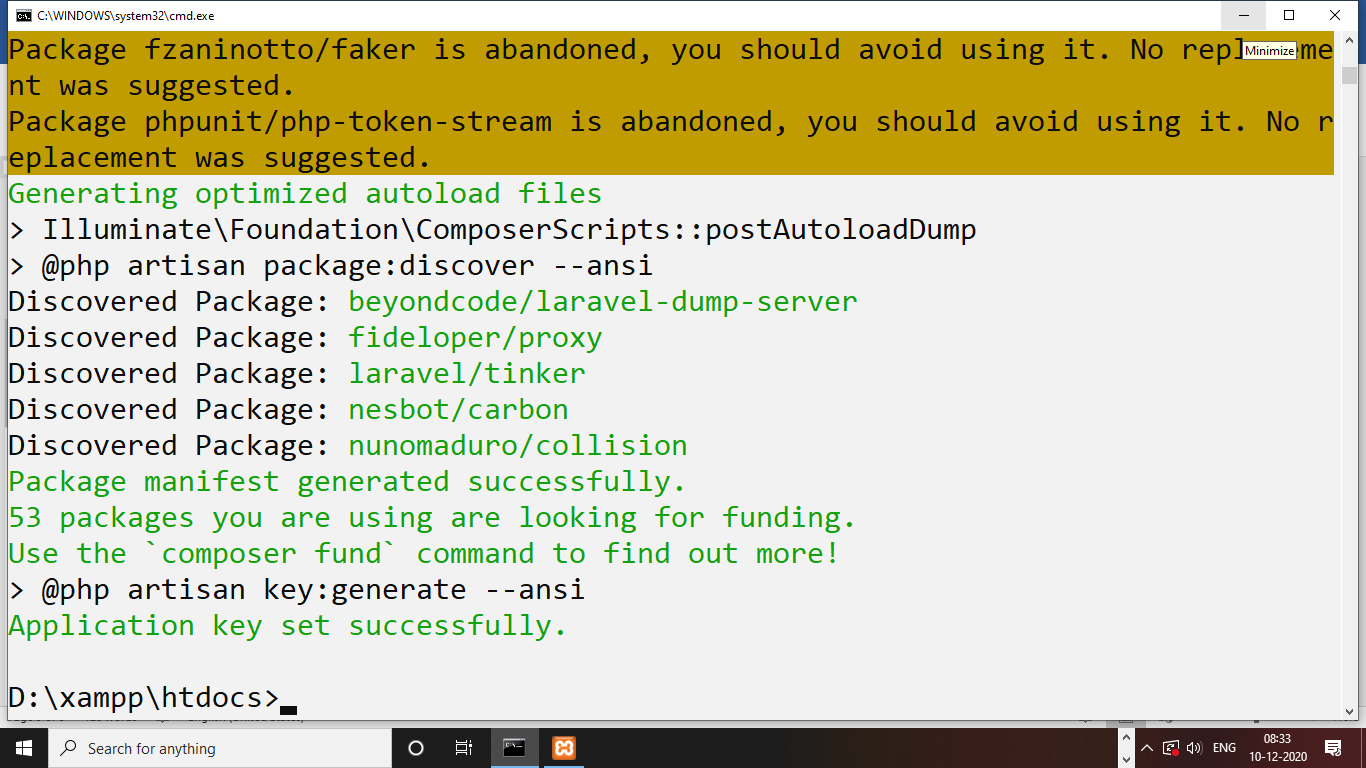


Step 7:

Install Laravel 5.8

Cmd :

composer create-project --prefer-dist laravel/laravel blog "5.8.\*"



How to run Laravel project :

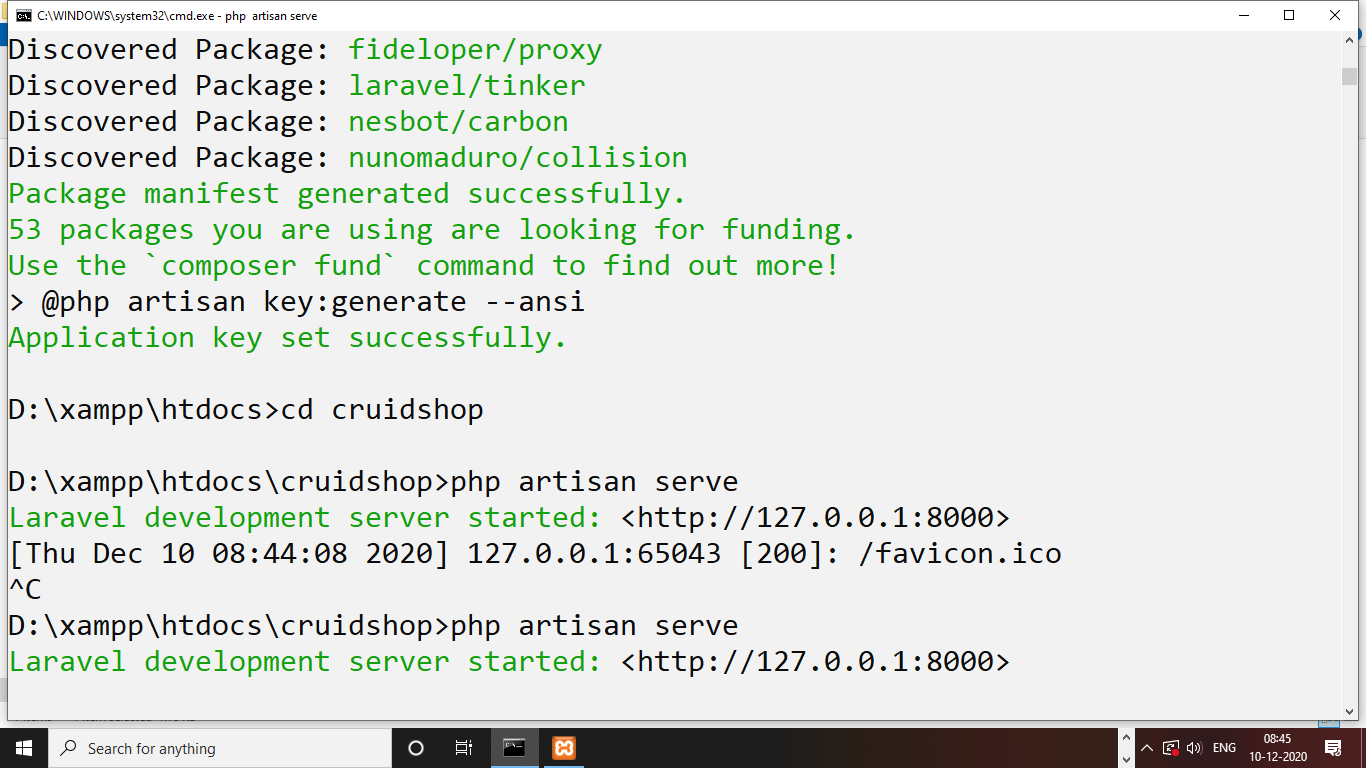
How to run Server development URL :

Create a artisan command of Laravel :

Artisan command:

Laravel provides a **artisan command** to create a model, controller, or database or run Laravel on server or localhost.

Cmd : php artisan serve



**How to load Laravel on localhost :**

Step 1 :

Go in project main directory

Step 2:

Open server.php file

Rename with index.php

<http://localhost/cruidshop/>

**Database Migrations :**

Database migrations is a version control i.e used to provide database connection and create a table structure in Laravel.

How to provide a database connection :

Step 1: open .env file in Laravel structure.

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=cruidapp

DB\_USERNAME=root

DB\_PASSWORD=

Step 2: open config->database.php

        'mysql' => [

            'driver' => 'mysql',

            'url' => env('DATABASE\_URL'),

            'host' => env('DB\_HOST', '127.0.0.1'),

            'port' => env('DB\_PORT', '3306'),

            'database' => env('DB\_DATABASE', 'cruidapp'),

            'username' => env('DB\_USERNAME', 'root'),

            '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'),

            ]) : [],

        ],

Step 3: Laravel default tables migrations provide.

Open database->migrations->2014\_10\_12\_000000\_create\_users\_table.php

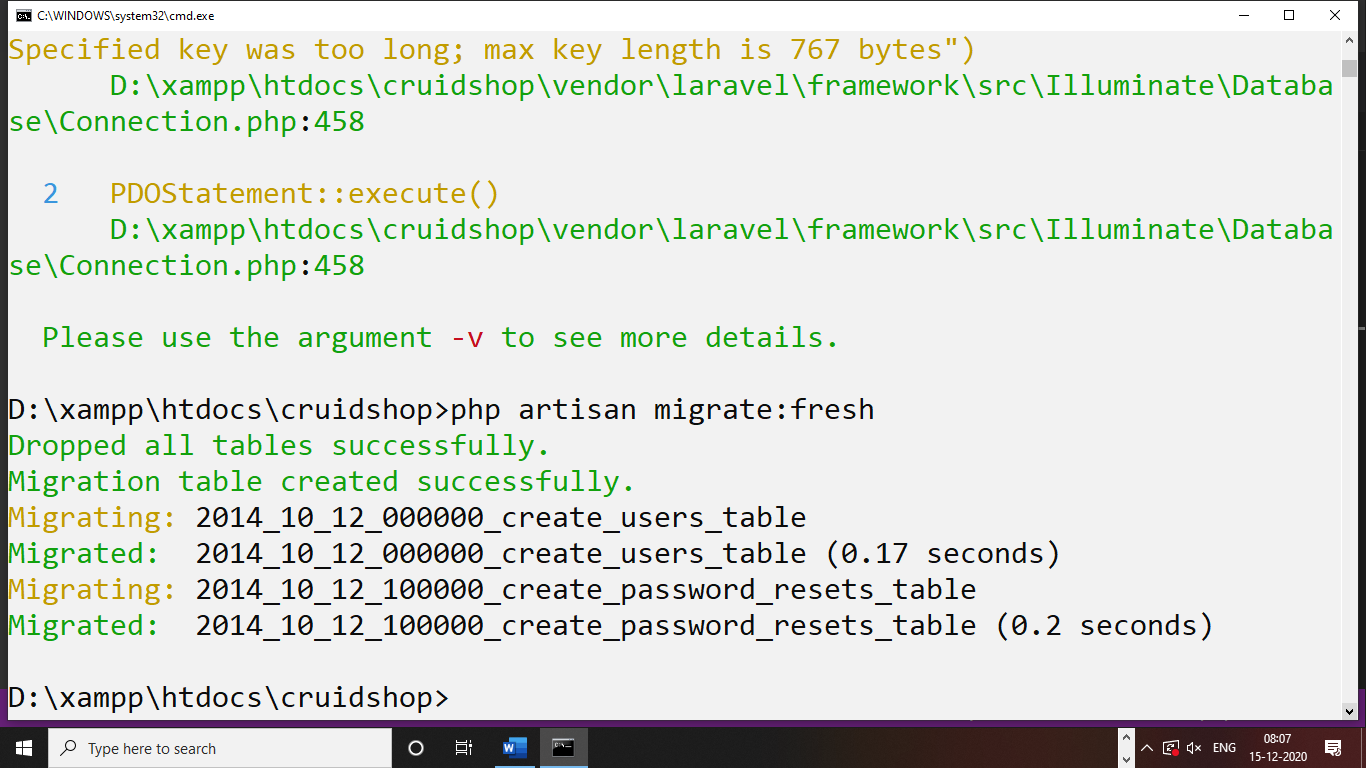
Step 4: how to migrate database in Laravel

Cmd : database migrations

Php artisan migrate

Step 5: how to drop tables and migrate furthure

Cmd : php artisan migrate:fresh

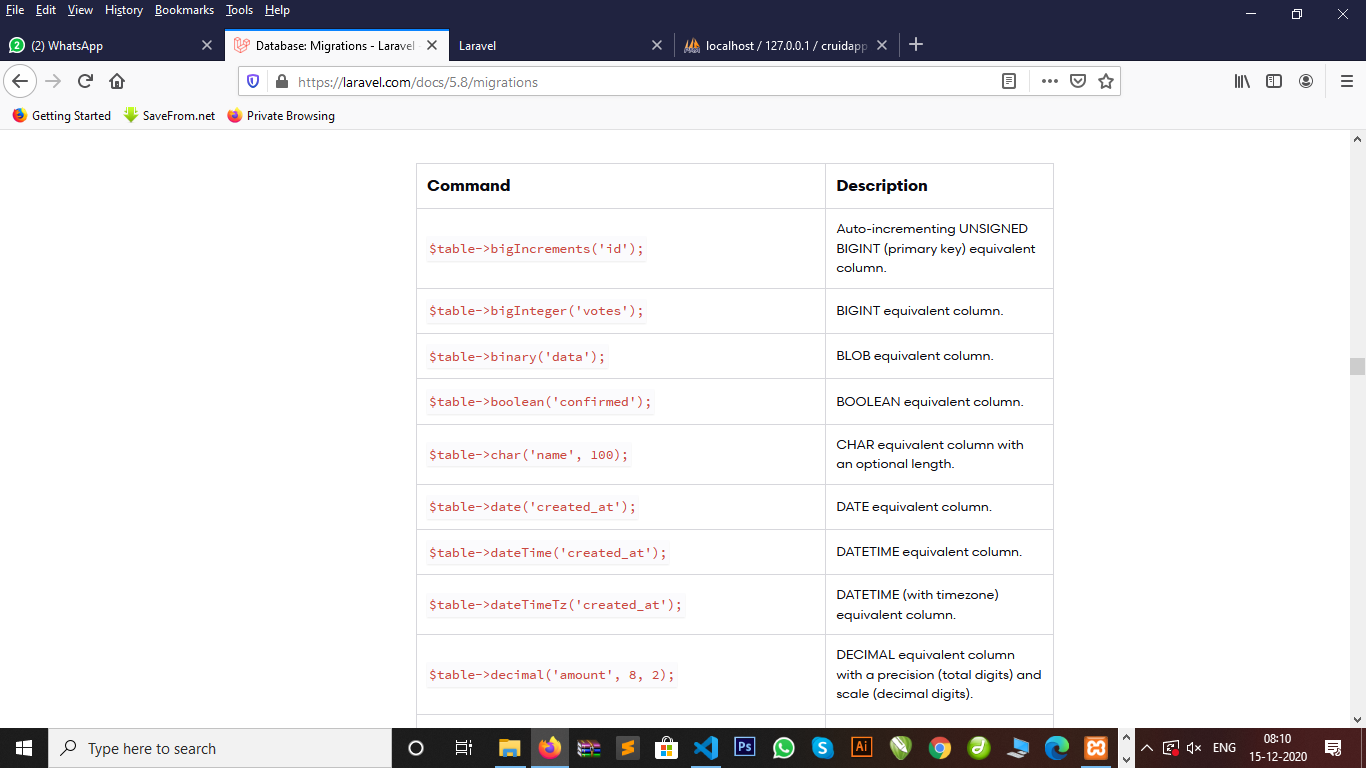


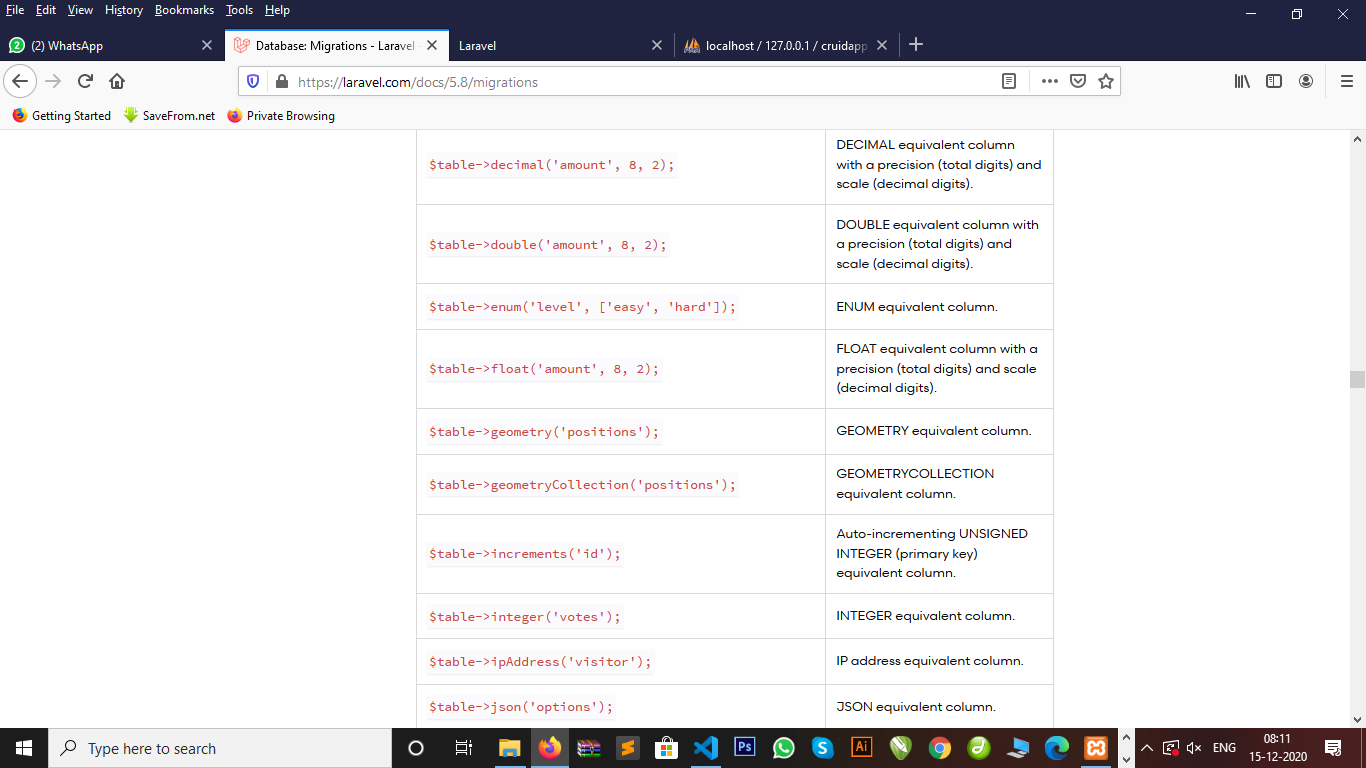
How to create a table in Laravel :

Step 1: in Laravel a table will created by database migrations command

Step 2: in Laravel a table will include columns and row

Step 3: in Laravel a table had a different-2 datatypes





Command of migrations to creating tables in Laravel :

Step 1: php artisan make:migration create\_feedbacks\_table

Or

Step 2: php artisan make:migration create\_registers\_table –create=registers

Default migrations table file structure Laravel :

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateRegisters extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('registers', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->timestamps();

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('registers');

    }

}

How to create Foreign Key in Laravel database Migrations :

Step 1: php artisan make:migration create\_countries\_table –create=countries

Step 2: php artisan make:migration create\_states\_table –create=states

Step 3: php artisan make:migration create\_cities\_table –create=cities

Step 4: php artisan migrate:fresh –force

Php artisan force is used to delete and rollback all tables forcefuly.

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateCountriesTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('countries', function (Blueprint $table) {

            $table->increments('id');

            $table->string('cname');

            $table->timestamps();

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('countries');

    }

}

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateStatesTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('states', function (Blueprint $table) {

            $table->increments('id');

            $table->integer('cid')->unsigned();

            $table->foreign('cid')->references('id')->on('countries');

            $table->string('sname');

            $table->timestamps();

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('states');

    }

}

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateCitiesTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('cities', function (Blueprint $table) {

            $table->increments('id');

            $table->integer('sid')->unsigned();

            $table->foreign('sid')->references('id')->on('states');

            $table->string('ctname');

            $table->timestamps();

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

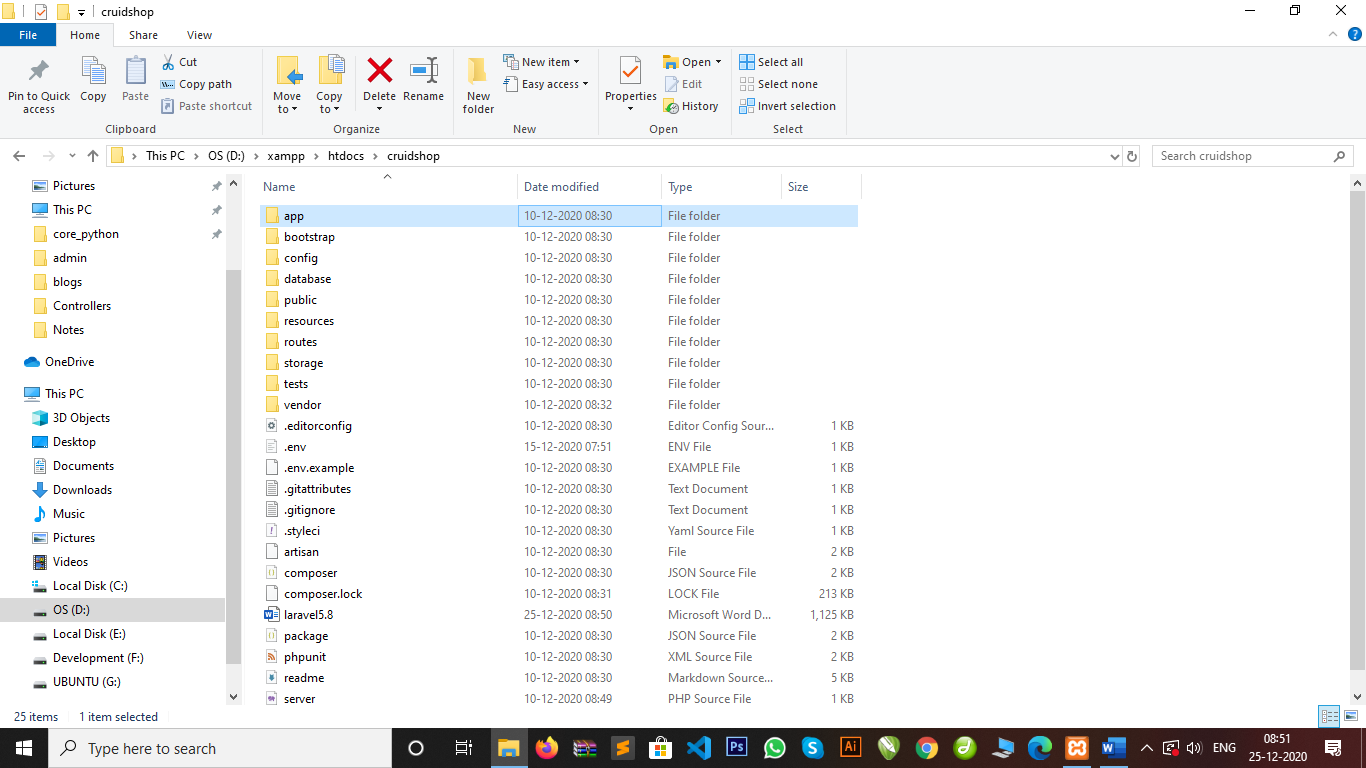
    {

        Schema::dropIfExists('cities');

    }

}

File structure or directory structure :



Laravel blade templating :

1. Default views file structure in Laravel build with blade templating and follow inheritance concepts
2. Extends keywords inheritance concepts
3. Ex: index.blade.php

Routing : routing is just load your appropriate user url that user wants.

:Laravel routes=>web.php

Ex: Route::get(‘/damini’, function(){

echo “Hi damini how are you”;

});

Routing: 1) get

2) post

3) resource

Resource : we used resource to post or get data using routing then we used only controller and its accept all method.

How to create a controller :

Cmd : 1) php artisan make:controller TestController

Cmd : 2) php artisan make:controller Test1Controller –resource

Note : --resource will create all function or methods of controller index(), create(), store(), update(), destroy(), edit()

How to create a Model :

Cmd : php artisan make:model TestModel

How to create a Controller

Cmd : php artisan make:controller Test2Controller –resource

Blade Templating :

Blade template is a default Laravel used for template integration

Blade template is used a inheritance concepts

Ex: index.blade.php

Structure of blade templating

1)master.blade.php =>master will handle all your assets file here

2)header.blade.php

3)navigation.balde.php

4)slider.blade.php

5)leftsidebar.blade.php

6)rightsidebar.blade.php

7)contant.blade.php

8)footer.blade.php

Public=>user=>assets ->css

->js

->images

->fonts

Laravel CRUID operations :

Using Laravel auth operations:

Auth =>auth is a Laravel default authentications operations i.e provide a default CRUID operations.

ORM Model : orm model is a Laravel mdel i.e stands for object relation mapping.

ORM model provides CRUID operations i.e created by user using model.

Cmd : php artisan make:auth

Controller =>HomeController

Model =>User.php

View=>[home.blade.php](http://home.blade.php)

View=>auth=>register.blade.php

View=>auth=>login.blade.php

View=>auth=>verify.blade.php

$this->middleware(“auth”);

Middleware =>middleware is a default URI(uniform resource interface) i.e provide default url its means user can not hit a unwanted url.

Note : middleware is a mechanism i.e used to redirect one page to another page where user authenticated with right credentials or not.

Middleware provide a convenient mechanism for inspecting and filtering HTTP requests entering your application. For example, Laravel includes a middleware that verifies the user of your application is authenticated. If the user is not authenticated, the middleware will redirect the user to your application's login screen. However, if the user is authenticated, the middleware will allow the request to proceed further into the application.

Ex : $this->middleware(“auth”);

Namespace :

Another common use-case for route groups is assigning the same PHP namespace to a group of controllers using the namespace method:

Route::namespace('Admin')->group(function () {

// Controllers Within The "App\Http\Controllers\Admin" Namespace

});

Remember, by default, the RouteServiceProvider includes your route files within a namespace group, allowing you to register controller routes without specifying the full App\Http\Controllers namespace prefix. So, you only need to specify the portion of the namespace that comes after the base App\Http\Controllers namespace.

ORM : object relation mapping

Using Model we create our own cruid operations using orm model

Cmd : php artisan make:model Flight

Cmd : php artisan make:model Flight –migration

Cmd: php artisan make:model Flight -m

### [Eloquent Model Conventions](https://laravel.com/docs/5.8/eloquent#eloquent-model-conventions)

Now, let's look at an example Flight model, which we will use to retrieve and store information from our flights database table:

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Flight extends Model

{

//

}

**Note : eloquent model will used to retrive , store, update or delete data using model and controller in Laravel.**

**Events :**

Eloquent models fire several events, allowing you to hook into the following points in a model's lifecycle: retrieved, creating, created, updating, updated, saving, saved, deleting, deleted, restoring, restored. Events allow you to easily execute code each time a specific model class is saved or updated in the database. Each event receives the instance of the model through its constructor.

<?php

namespace App;

use App\Ev­ents\UserSaved;

use App\Events\UserDeleted;

use Illuminate\Notifications\Notifiable;

use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable

{

use Notifiable;

/\*\*

\* The event map for the model.

\*

\* @var array

\*/

protected $dispatchesEvents = [

'saved' => UserSaved::class,

'deleted' => UserDeleted::class,

];

Protected $table=’users’;

}

**CSRF :**

Csrf is a form token to send any data using form there will used csrf token i.e @csrf.

Note : CSRF is a token that is used to send securely data on server using form

Ex. <form method=”post”>

@csrf

</form>