* Clone your project
* Go to the folder application using cd command on your cmd or terminal
* Run composer install on your cmd or terminal
* Copy .env.example file to .env on the root folder. You can type copy .env.example .env if using command prompt Windows or cp .env.example .env if using terminal, Ubuntu
* Open your .env file and change the database name (DB\_DATABASE) to whatever you have, username (DB\_USERNAME) and password (DB\_PASSWORD) field correspond to your configuration.
* Run php artisan key:generate
* Run php artisan migrate
* Run php artisan serve
* Go to http://localhost:8000/

**Подключение**

-Устанавливаем Composer

-Через командную строку устанавливаем Laravel(как указано на сайте Laravel)

-Устанавливаем наш новый проект Laravel в C:\OSPanel\domains(если установлен Open Server)

-В настройках Open Server и в разделе "Домены" настроим новый домен для нашего нового проекта. Домашний каталог у Laravel (там где лежит файл index.php) находится в каталоге "public", и именно его мы должны указать в качестве папки домена. Имя домена "mysite.loc" может быть любое



***Функции(хэлперы)***

**protected $table='имя таблицы';** – в Model указывать имя то таблицы с которой связана Model

**dd(переменная)** – показывает значение и останавливает выполнение кода

**dump(переменная)** – показывает значение без остановки выполнения кода

***Eloquent ORM***

Laravel содержит [ORM-библиотеку](https://ru.wikipedia.org/wiki/ORM) Eloquent, которая работает с базой данных. Eloquent использует связь Model со своей БД.

**find(id)** – возвращает элемент из БД соответствующий id

$user = User::find(1);

**create([])** –добавляет элементов в БД(в Model необходимо вставить protected $guarded=[] или protected $guarded=false;)

$user = User::create([

‘title’=>’some’,

…]);

**update([])** –изменяет существующий элементов в БД

$user = User::find(…)->update([

‘title’=>’update some’,

…]);

**delete()** –удаляет существующий элементов в БД(В Миграции(создание таблицы) указать $table->SoftDeletes()-данные останутся в таблице и их можно восстановить и в Model указать use SoftDeletes;)

$user = User::find(…)->delete();

**restore()** –восстанавливает удаленный элементов в БД(если он был удален при помощи SoftDeletes)

$user = User:: withTrashed() ->where('airline\_id', 1) ->restore()

**firstOrCreate([проверка], [новый элемент])** –возвращает элемент БД соответствующий условиям(например значение определенного атрибута), а если такого нет то создаст *Новый* и возвращает его. Можно использовать, чтобы избежать дублирования

**updateOrCreate([проверка], [обновление элемента или новый элемент])** – тоже что и firstOrCreate(), только если находится элемент соответствующий условию в него вносятся изменения и возвращает измененный элемент

***Методы Laravel***

**all()** –возвращает массив из Коллекция(collect). Позволяет получить все элементы БД

$user = User::all();

**where(‘ключ’, ‘значение’)** –фильтрует Коллекция(collect) по “ключ” / “значение”

$posts=User::where('is\_published', 1)->get();

-or-

$posts=User:: all()->where('is\_published', 1);

**first()** –возвращает первый элемент Коллекция(collect). Позволяет получить первый элемент БД

$user = User::first();

***Controller***

**Создание Controller**

**-**в командной строке php artisan make:controller [Name(CamelCase)Conroller]

***Migration***

**Создание Migration(создание таблицы) c Model**

**-**в командной строке php artisan make:model [Name] -m

-подключение к базе: файл .env заменить DB\_DATABASE на имя вашей базы

**Накатить(выполнить функцию up) Migration**

**-**в командной строке php artisan migrate

**Oткатить(выполнить функцию down) Migration**

**-**в командной строке php artisan migrate:rollback

**Обновить Migration**

**-**в командной строке php artisan migrate:fresh

**Добавить колонку в Migration**

**-**в командной строке php artisan make:migration [add\_colomun\_’имя атрибута’\_to\_’имя миграции где создавалась таблица(2023\_01\_24\_084927\_create\_posts\_table.php)’]

public function up() {

Schema::table('posts', function (Blueprint $table) {

$table->text('description')->nullable();

}); }

public function down() {

Schema::table('posts', function (Blueprint $table) {

$table->dropColumn('description');

}); }

**Удалить колонку в Migration**

**-**в командной строке php artisan make:migration [delete\_colomun\_’имя атрибута’\_to\_’имя миграции где создавалась таблица(2023\_01\_24\_084927\_create\_posts\_table.php)’]

public function up() {

Schema::table('posts', function (Blueprint $table) {

$table->dropColumn('description');

}); }

public function down() {

Schema::table('posts', function (Blueprint $table) {

$table->text('description')->nullable();

}); }

**Редактировать колонку(имя,тип данных..) в Migration**

**Если будет ошибка при php artisan migrate, то выполнить composer require doctrine/dbal**

**-**в командной строке php artisan make:migration [edit\_colomun\_’имя атрибута’\_to\_’имя миграции где создавалась таблица(2023\_01\_24\_084927\_create\_posts\_table.php)’]

public function up() {

Schema::table('posts', function (Blueprint $table) {

!name!

$table->renameColumn('content', ‘post\_content’);

!type!

$table->text('column\_name')->change();

}); }

public function down() {

Schema::table('posts', function (Blueprint $table) {

!name!

$table->renameColumn(' post\_content ', ‘content’);

!type!

$table->string('column\_name')->change();

}); }

**Удалить таблицу в Migration**

**-**в командной строке php artisan make:migration [delete\_’имя миграции где создавалась таблица(2023\_01\_24\_084927\_create\_posts\_table.php)’]

public function up() {

Schema::dropIfExists('posts')

public function down() {

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

$table->id();

$table->text('description')->nullable();

…

}); }

***View***

**В Contoller в функции указываем – return view(‘file in folder view’, compact(таблица)), данные из compact(таблица) передаются в файл в папке view**

**Bootstrap – composer require Laravel/ui** after **php artisan ui bootstrap** after **npm run dev**

**<link rel="stylesheet" href="{{asset('css/app.css')}}"> - подключаем на странице стили bootstrap, с помощью asset путь всегда будет прописан правильно, даже если имя файла blade.php поменяется или его расположение**

**@foreach(array as item){{item}}endforeach** –перебор массива в файлах blade.php, при этом **item** выводится на html странице

**@yield(‘name’)** –создается шаблон на html странице с именем ‘name’

**@extends (‘folder/file’)** –на другой странице blade.php вставляется этот шаблон, с указанием пути к файлу

**@section (‘name’) ‘text’ endsection** – в этой же странице где @extends вставляется @section с тем же именем ‘name’ который подставляем свое значение ‘text’ вместо @yield

**->name(‘name’)** –задает любое имя страницы в Route и далее в проекте можно использовать это name когда надо указать ссылку на страницу…

<a href="{{route('main.index')}}">Main</a>