**CREAR UN CONTROLADOR**

Php artisan make:controller TaskController

**CREAR UN MODELO**

Php artisan make:model Task

1. -**Crear un proyecto con laravel**

Composer create-project laravel/laravel –prefer-dist nombreproyecto “5.1”

1. **-Modificar el archivo ENV para acceso a la base datos**
2. **-Crear virtualhost para el proyecto**
3. **–Modificación del archivo Windows/system32/driver/etc/hosts**
4. **--instalación de laravelcollective.**

* En Archivo composer .json

"require": {

"laravelcollective/html": "~5.0"

}

* En config/app.php

'providers' => [

// ...

'Collective\Html\HtmlServiceProvider',

// ...

],

'aliases' => [

// ...

'Form' => 'Collective\Html\FormFacade',

'Html' => 'Collective\Html\HtmlFacade',

// ...

],

1. **Crear migraciones**

php artisan make:migration create\_nomtabla\_table –create=nomtable

1. **Crear relaciones con las tablas en las migraciones**

ej: paises con departamentos

$table->integer(‘pais\_id’)->unsigned();

$table->foreign(‘pais\_id’)->references(‘id’)->on(‘paises’);

1. **Ejecutar migraciones**

php artisan migrate

**nota**:se deben crear las migraciones para las tablas que no tienen dependencia primero.

1. **crear seeders**

* en el archivo database/factories/ModelFactory.php

Ej:

define(App\User::class,function($faker){

return[

’nombre’=>$faker->firstName,

‘direccion’=>$faker->address

];

});

//nota más fakers <https://github.com/fzaninotto/Faker>

En el archive dabase\seed\DataBaseSeeder.php en el método run llamar al Factory.

Ej: Factory(‘App\User’,10)->create()

1. **definir el flujo de pantallas**

programa balsamiq

1. **crear clases y relaciones**

php artisan make:model Client

class Client extends model

{

Protected $primarykey =’id’;

Protected $table =’clients’;

Protected $fillable=[‘nombre’,’ape’,’persona\_id’];

Protected $guarded =[‘id’];

//se establece la relación

public function person(){

Return $this->hasOne(‘\App\Person’,’id’,’persona\_id’);

}

}

1. **Insertar en modelos relacionados**

**Ej:**

Class ClientController extends Controller

{

Public function store(Request $request)

{

if($request->ajax())

{

$person = new Person;

$person->nombrecompleto = $request->nombrecompleto;

$client = new Client;

$client->persona\_id = $person->id

$person->client()->save($client);

}

}

}

**13. Crear rutas.**

Dentro del archive App\http\routes.php

Tipos de rutas:

-Utilizando un controlador

Route::post(‘client/store’,’ClientController@store’);

-utilizando get para devolver un vista directamente

Route::get(‘’,function(){

Return view(‘welcome’);

});

**13.Crear un layout para las vistas.**

**app.blade.php**

En la ruta resources/views/ colocamos las plantillas.

Ej:plantillas

<!DOCTYPE html>

<html lanng=”en”>

<head>

<meta charset=”utf-8”>

<meta name=”viewport” initial-scale=”1” content=”witdh=device-width”>

<meta name=”\_token” content=”{!!csrf\_token() !!}”/>

<tittle>nombre del proyecto</tittle>

</head>

<body>

@yield(‘content’);

@yield(‘scripts’);

</body>

</html>

**14.Crear views**

Cliente.app.php

Crear vista en resources/views

@extends(‘layout.app’) // de cual plantilla extiende

@section(‘content’) // sección que se incluirá en la parte de la plantilla

@endsection

@section(‘scripts’) // zona para colocar scripts

{!! Html::script(‘js/ajax.js’) !!}

@endsection

**15. Creación de Forms**

{!!Form::open(array(“url”=>’/’,’method’=>’post’,’class’=>’form-horizontal’,’id’=>’formjobs’))!!}

**16.Creación de etiquetas y cajas de texto,combobox.**

{!!Form::label(‘actividadcampo’,’Actividad’) !!}

{!!Form::text(‘actiname’,null,[“class”=>’form-control’,”id”=>”actividad-name”,”placeholder”=>”Nombre de la actividad”])!!}

{!!Form::select(‘Departamento’,$arrayOption,’Seleccione’,[‘classs’=>’form-control’,’id’=>’departamento\_id’])!!}

**17.Función ajax.**

$.ajax({

url:{{route(’select-ajax’)}},

method:’POST’,

data:{var\_id:valor,var\_id2:valor},

headers:{‘X-CSRF-Token’ : $(‘meta[name=\_token]’).attr(‘content’)},

success:function(data){

$(“select[‘name’=’dpto’]”).html()

$(“select[‘name’=’dpto’]”).html(data.options)

}

})