

# Configuración de Laravel 6 en Docker

**Objetivo:** Configurar docker para nginx (última versión), php (última versión), mysql8 y laravel 6 (última versión)

**Autor:** Ambrosio Cardoso Jiménez

**Fecha última revisión:** 26-Feb-2020

## Requisitos:

- Tener instalado docker (para esta práctica se uso versión 19.03.5, build 633a0ea838)
- Tener instalado docker-compose
- Tener instalado composer (gestor de paquetes para programar en PHP el cual provee los formatos estándar necesarios para manejar dependencias y librerías)

**Paso 1:** Crear una carpeta **dockers/ugm**

**Paso 2:** Dentro de ella crear carpeta **nginx** y dentro de ella un archivo de texto **nginx.conf** cuyo contenido será la siguiente:

```
server {  
    listen 80;  
    root /var/www/html/proy_ugm/public;  
    index index.php index.html;  
    server_name localhost;  
  
    error_log /var/log/nginx/error.log;  
    access_log /var/log/nginx/access.log;
```

```

location / {
    try_files $uri $uri/ /index.php?$query_string;
}

location ~ \.php$ {
    try_files $uri =404;
    fastcgi_split_path_info ^(.+\.php)(/.+)$;
    fastcgi_pass ugm_php:9000;
    fastcgi_index index.php;
    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param PATH_INFO $fastcgi_path_info;
}
}

```

**Paso 3:** Crear la carpeta vacía **dockers/ugm/www**

**Paso 4:** Crear el archivo **docker-compose.yml** en el directorio **ugm** con el siguiente contenido:

```
version: '3.7'
```

```
networks:
```

```
    ugm:
```

```
services:
```

```
    ugm_nginx:
```

image: nginx:stable-alpine

container\_name: ugm\_nginx

ports:

- "8000:80"

volumes:

- "./www:/var/www/html"

- "./nginx/nginx.conf:/etc/nginx/conf.d/default.conf"

depends\_on:

- ugm\_php

- ugm\_mysql

networks:

- ugm

ugm\_mysql:

image: mysql:latest

container\_name: ugm\_mysql8

volumes:

- ugm\_mysql8\_data:/var/lib/mysql

command: ['--character-set-server = utf8mb4', '--collation-server = utf8mb4\_unicode\_ci', '--default-authentication-plugin = mysql\_native\_password']

tty: true

ports:

- "3333:3306"

environment:

MYSQL\_DATABASE: bdugm

MYSQL\_USER: ugmDev

```
MYSQL_PASSWORD: pwdUgmDev
MYSQL_ROOT_PASSWORD: t0pS3cr3t
SERVICE_NAME: ugm_mysql
```

networks:

- ugm

ugm\_php:

build:

context: .

dockerfile: Dockerfile

container\_name: ugm\_php

volumes:

- "./www:/var/www/html"

ports:

- "9999:9000"

networks:

- ugm

volumes:

ugm\_mysql8\_data:

**Paso 5:** Crear el archivo **Dockerfile** en el directorio **ugm** con el siguiente contenido:

```
#--- El repositorio docker de PHP
FROM php:7.2-fpm-alpine
RUN docker-php-ext-install pdo pdo_mysql mysqli
```

#--- Usar apk en lugar de apt-get para cuando se usa alpine

#--- nota de Cardoso 26-Feb-2020

RUN apk update && apk add \

curl \

wget \

zip \

unzip \

composer \

npm

**Paso 6:** Ejecutar la siguiente instrucción desde la carpeta **ugm**

**docker-compose build**

**NOTA:** En ocasiones aparece el siguiente mensaje de error:

**ERROR:** Couldn't connect to Docker daemon at http+docker://localunixsocket - is it running?

**SOLUCION:** iniciar docker (systemctl start docker para linux y simplemente ejecutarlo para Windows)

Volver a ejecutar la sentencia **docker-compose build**

**Paso 7:** Iniciar los contenedores ejecutando

**docker-compose up -d** (-d => para ejecutar en segundo plano)

**Paso 8:** Comprobar que los tres contenedores (nginx, mysql8 y php) estén corriendo

**docker ps**

```
[cardoso@asuboshy ugm]$ docker ps
CONTAINER ID   IMAGE               COMMAND                  CREATED        STATUS        PORTS                               NAMES
9f24d371f4dc   nginx:stable-alpine  "nginx -g 'daemon of..." 4 days ago    Up 14 minutes  0.0.0.0:8000->80/tcp                ugm_nginx
112cef5a1fc3   ugm_ugm_php         "docker-php-entrypoi..." 4 days ago    Up 14 minutes  0.0.0.0:9999->9000/tcp              ugm_php
8a9ac1b36dae   mysql:latest        "docker-entrypoint.s..." 4 days ago    Up 14 minutes  33060/tcp, 0.0.0.0:3333->3306/tcp    ugm_mysql8
[cardoso@asuboshy ugm]$
```

Figura 1. Contenedores en ejecución

**Paso 9:** Entrar al contenedor

```
docker exec -it php /bin/sh
```

**Paso 10:** Escriba composer. Debe mostrar la documentación de composer, si dice que no existe o no reconoce la sentencia entonces **ejecutar las siguientes sentencias**

```
# php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"  
# HASH="$(wget -q -O - https://composer.github.io/installer.sig)"  
# php composer-setup.php --install-dir=/usr/local/bin --filename=composer  
# mv /usr/local/bin/composer.phar /usr/bin/composer
```

**Paso 11:** Estando en el contenedor cambiar al directorio ***/var/www/html***

```
cd /var/www/html
```

**Paso 12:** Crear el proyecto (proy\_ugm) laravel estando en la ruta /var/www/html

```
# composer create-project --prefer-dist laravel/laravel proy_ugm
```

**esperar a que termine la descarga...**

**Paso 13:** Editar el archivo /var/www/html/proy\_ugm/.env cambiando los datos resaltados

```
APP_NAME = ugm
```

```
APP_ENV = local
```

```
APP_KEY = base64:82Z0EFT/ysqWDaaThBcTajemwOsVT1Pkov4pLkTJePw =
```

```
APP_DEBUG = true
```

```
APP_URL = http://localhost:8000
```

LOG\_CHANNEL = stack

DB\_CONNECTION = mysql

DB\_HOST = 172.21.0.3

DB\_PORT = 3306

DB\_DATABASE = ugm

DB\_USERNAME = root

DB\_PASSWORD = t0pS3cr3t

...

La **IP** se obtiene con la siguiente instrucción

```
docker inspect -f  
'{{range .NetworkSettings.Networks}}  
  {{.IPAddress}} {{end}}'  
ugm_mysql8
```

(En una sola línea, **ugm\_mysql8** es el nombre del contenedor como se aprecia en la figura 1)

**Paso 14:** Desde el host probar desde el navegador <http://localhost:8000>

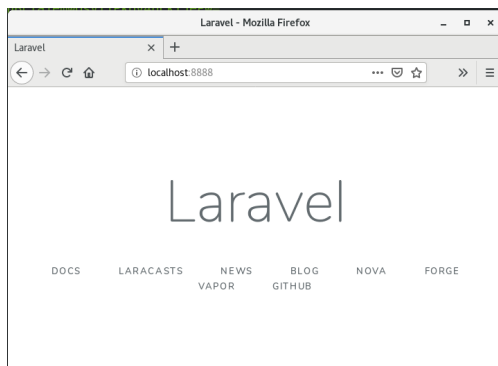


Figura 2. Laravel en ejecución

# CRUD

a) Entrar al contenedor de `ugm_mysql8`

```
docker exec -it ugm_mysql8 /bin/sh
```

b) Entrar a mysql con la cuenta de root

```
#mysql -u root -p
```

c) Crear la BD, las tablas y sus relaciones

```
CREATE DATABASE bdugm CHARACTER SET utf8mb4
```

```
COLLATE utf8mb4_unicode_ci;
```

```
use bdugm;
```

```
show tables;
```

```
CREATE TABLE casilla
```

```
( id BIGINT AUTO_INCREMENT
```

```
, ubicacion VARCHAR (100) NOT NULL
```

```
, CONSTRAINT pkcasilla PRIMARY KEY(id)
```

```
);
```

```
CREATE TABLE candidato
```

```
( id BIGINT AUTO_INCREMENT
```

```
, nombrecompleto VARCHAR (200) NOT NULL
```

```
, foto VARCHAR (200)
```

```
, sexo CHAR
```

```
, perfil VARCHAR (200)
```

```
, CONSTRAINT pkcandidato PRIMARY KEY (id)
```

```
);
```



```
CREATE TABLE eleccion
( id BIGINT AUTO_INCREMENT
, periodo VARCHAR (100) NOT NULL
, fecha DATE
, fechaapertura DATE
, horaapertura TIME
, fechacierre DATE
, horacierre DATE
, observaciones TEXT
, CONSTRAINT pkeleccion PRIMARY KEY (id)
);
```

```
CREATE TABLE funcionario
( id BIGINT AUTO_INCREMENT
, nombrecompleto VARCHAR (200) NOT NULL
, sexo CHAR
, CONSTRAINT pkfuncionario PRIMARY KEY (id)
);
```

```
CREATE TABLE rol
( id BIGINT AUTO_INCREMENT
, descripcion VARCHAR (100) NOT NULL
, CONSTRAINT pkrol PRIMARY KEY(id)
);
```

```
CREATE TABLE eleccioncomite
( id BIGINT AUTO_INCREMENT
, eleccion_id BIGINT
, funcionario_id BIGINT
, rol_id BIGINT
, CONSTRAINT pkeleccioncomite PRIMARY KEY (id)
, CONSTRAINT ukeleccioncomite UNIQUE KEY (eleccion_id,funcionario_id)
, CONSTRAINT fkfuncionario_eleccioncomite
  FOREIGN KEY (funcionario_id)
  REFERENCES funcionario (id)
, CONSTRAINT fkrol_eleccioncomite
  FOREIGN KEY (rol_id)
  REFERENCES rol (id)
, CONSTRAINT fkeleccion_elecomite
  FOREIGN KEY (eleccion_id)
  REFERENCES eleccion (id)
);
```

```
CREATE TABLE voto
( id BIGINT AUTO_INCREMENT
, eleccion_id BIGINT
, casilla_id BIGINT
, evidencia VARCHAR (200)
, CONSTRAINT pkvoto PRIMARY KEY (id)
, CONSTRAINT ukvoto UNIQUE
  (eleccion_id, casilla_id)
, CONSTRAINT fkeleccion_voto
```

```
    FOREIGN KEY (eleccion_id)
    REFERENCES eleccion (id)
, CONSTRAINT fkcasilla_id
    FOREIGN KEY (casilla_id)
    REFERENCES casilla (id)
);
```

```
CREATE TABLE votocandidato
( voto_id BIGINT
, candidato_id BIGINT
, votos INT
, CONSTRAINT pkvotocandidato
    PRIMARY KEY(voto_id, candidato_id)
, CONSTRAINT fkvoto_vc FOREIGN KEY (voto_id)
    REFERENCES voto (id)
, CONSTRAINT fkcandidato_vc
    FOREIGN KEY (candidato_id)
    REFERENCES candidato (id)
);
```

```
CREATE TABLE funcionariocasilla
( id BIGINT AUTO_INCREMENT
, funcionario_id BIGINT
, casilla_id BIGINT
, rol_id BIGINT
, eleccion_id BIGINT
, CONSTRAINT pkfc PRIMARY KEY (id)
```

```

, CONSTRAINT ukfc
    UNIQUE KEY (funcionario_id, eleccion_id)
, CONSTRAINT fkfunc_fc FOREIGN KEY (funcionario_id)
    REFERENCES funcionario(id)
, CONSTRAINT casilla_fc FOREIGN KEY (casilla_id)
    REFERENCES casilla (id)
, CONSTRAINT fkrol_fc FOREIGN KEY (rol_id)
    REFERENCES rol (id)
, CONSTRAINT fkeleccion_fc FOREIGN KEY (eleccion_id)
    REFERENCES eleccion (id)
);

```

```

CREATE TABLE imeiautorizado
( id BIGINT AUTO_INCREMENT
, funcionario_id BIGINT
, eleccion_id BIGINT
, casilla_id BIGINT
, imei VARCHAR(20) NOT NULL
, CONSTRAINT pkimei PRIMARY KEY (id)
, CONSTRAINT ukimei UNIQUE KEY (funcionario_id, eleccion_id)
, CONSTRAINT fkfun_imei FOREIGN KEY (funcionario_id)
    REFERENCES funcionario (id)
, CONSTRAINT fkeleccion_imei FOREIGN KEY (eleccion_id)
    REFERENCES eleccion (id)
, CONSTRAINT fkcasilla_imei FOREIGN KEY (casilla_id)
    REFERENCES casilla (id)
);

```

```
show tables;
```

```
INSERT INTO casilla (ubicacion )
```

```
VALUES ('Rectoria'),('Facultad de medicina'),('Facultad de arquitectura');
```

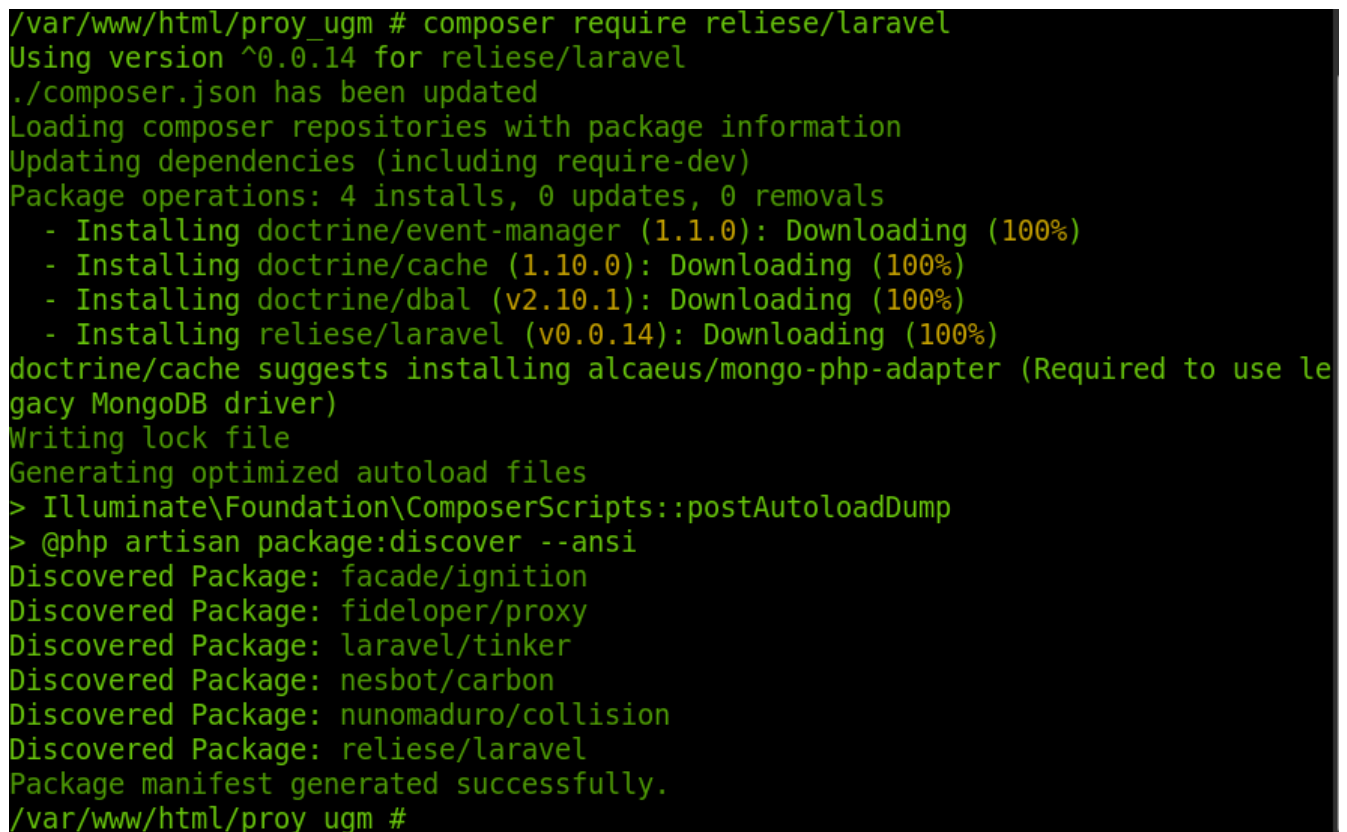
d) Generar los modelos que representen las tablas y sus relaciones, ejecutando la siguiente instrucción desde la carpeta **dockers/ugm**

```
docker exec -it ugm_php /bin/sh
```

y estando dentro del contenedor ejecutar los siguientes comandos

```
# cd /var/www/html/proy_ugm
```

```
# composer require reliese/laravel
```



```
/var/www/html/proy_ugm # composer require reliese/laravel
Using version ^0.0.14 for reliese/laravel
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 4 installs, 0 updates, 0 removals
  - Installing doctrine/event-manager (1.1.0): Downloading (100%)
  - Installing doctrine/cache (1.10.0): Downloading (100%)
  - Installing doctrine/dbal (v2.10.1): Downloading (100%)
  - Installing reliese/laravel (v0.0.14): Downloading (100%)
doctrine/cache suggests installing alcaeus/mongo-php-adapter (Required to use le
gacy MongoDB driver)
Writing lock file
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: laravel/tinker
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Discovered Package: reliese/laravel
Package manifest generated successfully.
/var/www/html/proy_ugm #
```

Figura 3. Proceso de descarga de plugin reliese laravel

e) Editar el archivo **config/app.php** y agregar en la seccion de providers

```
// ...
```

```
'providers' => [
```

```

/*
 * Package Service Providers...
 */

Reliese\Coders\CodersServiceProvider::class,
],

```

f) Generar las clases modelo de todas las tablas de la BD, ejecutando la siguiente instrucción:

```
/var/www/html/proy_ugm # php artisan vendor:publish --tag=reliese-models
```

```
/var/www/html/proy_ugm # php artisan config:clear
```

```
/var/www/html/proy_ugm # php artisan code:models
```

El resultado de esta última instrucción son los modelos como se aprecia en la figura 4

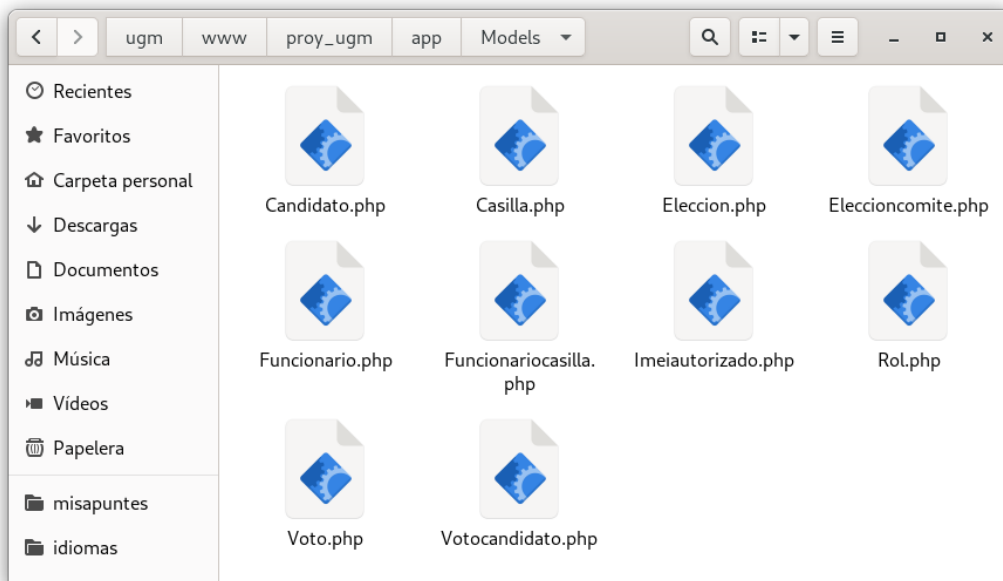


Figura 4. Clases que representa cada una de las tablas de la BD

g) Crear un controlador, comenzamos con la casilla

```
/var/www/html/proy_ugm # php artisan make:controller CasillaController --resource
```

la sentencia anterior genera un archivo en la subcarpeta **app/Http/Controllers/** con el siguiente contenido:

<?php

```
namespace App\Http\Controllers;
```

```
use Illuminate\Http\Request;
```

```
use App\Models\Casilla;
```

```
class CasillaController extends Controller
```

```
{
```

```
    /**
```

```
     * Display a listing of the resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
     */
```

```
    public function index()
```

```
    {
```

```
        $casillas = Casilla::all();
```

```
        return view('casilla/list', compact('casillas'));
```

```
    }
```

```
    /**
```

```
     * Show the form for creating a new resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
     */
```

```
    public function create()
```

```
    {
```

```
        return view('casilla/create');
```

```
    }
```

```

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $validacion = $request->validate([
        'ubicacion' => 'required|max:100',
    ]);

    $casilla = Casilla::create($validacion);

    return redirect('casilla')->with('success',
        $casilla->ubicacion . ' guardado satisfactoriamente ...');
}

```

```

/**
 * Display the specified resource.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    //

```



```
}
```

```
/**
```

```
 * Show the form for editing the specified resource.
```

```
 *
```

```
 * @param int $id
```

```
 * @return \Illuminate\Http\Response
```

```
 */
```

```
public function edit($id)
```

```
{
```

```
    $casilla = Casilla::find($id);
```

```
    return view('casilla/edit',
```

```
        compact('casilla'));
```

```
}
```

```
/**
```

```
 * Update the specified resource in storage.
```

```
 *
```

```
 * @param \Illuminate\Http\Request $request
```

```
 * @param int $id
```

```
 * @return \Illuminate\Http\Response
```

```
 */
```

```
public function update(Request $request, $id)
```

```
{
```

```
    $validacion = $request->validate([
```

```
        'ubicacion' => 'required|max:100',
```

```
    ]);
```

```

        Casilla::whereId($id)->update($validacion);
        return redirect('casilla')
            ->with('success', 'Actualizado correctamente...');
    }

```

```

/**
 * Remove the specified resource from storage.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function destroy($id)
{
    $casilla = Casilla::find($id);
    $casilla->delete();
    return redirect('casilla');
}
} //--- end class

```

**Se pueden agregar o quitar métodos a la clase anterior y en cada uno de ellos se indicará la lógica de la aplicación**

h) En la carpeta resources/views crear un archivo con el nombre de **plantilla.blade.php** y pegar el siguiente código:

```

<!DOCTYPE html>
<html lang="es">
<head>

```

```

<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>Elecciones</title>
<link href="{{ asset('css/app.css') }}" rel="stylesheet" type="text/css" />
</head>
<body>
    <div class="container">
        @yield('content')
    </div>
    <script src="{{ asset('js/app.js') }}" type="text/js"> </script>
</body>
</html>

```

i) Crear una carpeta **casilla** dentro de **resources/views/** y dentro de ella crear tres archivos: **create.blade.php**, **edit.blade.php** e **list.blade.php**, a continuación se pone el contenido de cada uno:

#### **create.blade.php**

```

@extends('plantilla')
@section('content')
<style>
    .uper {
        margin-top: 40px;
    }
</style>
<div class="card uper">
    <div class="card-header">

```

Agregar Casillas

</div>

<div class="card-body">

@if (\$errors->any())

<div class="alert alert-danger">

<ul>

@foreach (\$errors->all() as \$error)

<li>{{ \$error }}</li>

@endforeach

</ul>

</div> <br />

@endif

<form method="post" action="{{ route('casilla.store') }}"

enctype="multipart/form-data">

{{ csrf\_field() }}

<div class="form-group">

@csrf

<label for="ubicacion">Ubicación:</label>

<input type="text" class="form-control" name="ubicacion"/>

</div>

<button type="submit" class="btn btn-primary">Guardar</button>

</form>

</div>

</div>

@endsection

## edit.blade.php

```
@extends('plantilla')

@section('content')

<style>

    .uper {

        margin-top: 40px;

    }

</style>

<div class="card uper">

    <div class="card-header">

        Editar casilla

    </div>

    <div class="card-body">

        @if ($errors->any())

            <div class="alert alert-danger">

                <ul>

                    @foreach ($errors->all() as $error)

                        <li>{{ $error }}</li>

                    @endforeach

                </ul>

            </div> <br />

        @endif

        <form method="POST"

            action="{{ route('casilla.update', $casilla->id) }}"

            enctype="multipart/form-data">

            {{ csrf_field() }}
```

```
@method('PUT')
```

```
<div class="form-group">
```

```
    @csrf
```

```
    <label for="id">ID:</label>
```

```
    <input type="text"
```

```
    class="form-control"
```

```
    readonly="true"
```

```
    value="{{ $casilla->id }}"
```

```
    name="id"/>
```

```
</div>
```

```
<div class="form-group">
```

```
    @csrf
```

```
    <label for="descripcion">Ubicación:</label>
```

```
    <input type="text"
```

```
    value="{{ $casilla->ubicacion }}"
```

```
    class="form-control"
```

```
    name="ubicacion"/>
```

```
</div>
```

```
<button type="submit" class="btn btn-primary">Guardar</button>
```

```
</form>
```

```
</div>
```

```
</div>
```

```
@endsection
```

## list.blade.php

```
@extends('plantilla')
```

```
@section('content')
```

```
< style >
```

```
    .uper {
```

```
        margin-top: 40px;
```

```
    }
```

```
< /style >
```

```
< div class = "uper" >
```

```
    @if(session()->get('success'))
```

```
        < div class = "alert alert-success" >
```

```
            {{ session()->get('success') }}
```

```
        < /div > < br / >
```

```
@endif
```

```
< table class = "table table-striped" >
```

```
    < thead >
```

```
        < tr >
```

```
            < td > ID < /td >
```

```
            < td > UBICACION < /td >
```

```
            < td colspan = "2" > Action < /td >
```

```
        < /tr >
```

```
    < /thead >
```

```
    < tbody >
```

```
        @foreach($casillas as $casilla)
```

```
            < tr >
```

```

<td> {{ $casilla->id }} </td>
<td> {{ $casilla->ubicacion }} </td>
<td> <a href="{{ route('casilla.edit', $casilla->id) }}"
class="btn btn-primary"> Edit </a> </td>
<td>
    <form action="{{ route('casilla.destroy', $casilla->id) }}"
method="post">
        @csrf
        @method('DELETE')
        <button class="btn btn-danger" type="submit"
onclick="return confirm('Esta seguro de borrar {{ $casilla-
>ubicacion }}') "> Del </button>
    </form>
</td>
</tr>
@endforeach
</tbody>
</table>
<div>
@endsection

```

j) Crear las rutas para poder acceder a los recursos, para ello editamos el archivo **routes/web.php** y agregar al final del mismo la siguiente instrucción

```
Route::resource('casilla', 'CasillaController');
```



k) Para dar una apariencia bonita a las vista podemos usar bootstrap, instalando los paquetes correspondientes ejecutando la siguiente sentencia:

```
/var/www/html/proy_ugm # composer require laravel/ui
```

```
/var/www/html/proy_ugm # composer require laravel/ui --dev
Using version ^2.0 for laravel/ui
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Your requirements could not be resolved to an installable set of packages.

Problem 1
- Conclusion: remove laravel/framework v6.17.1
- Conclusion: don't install laravel/framework v6.17.1
- laravel/ui 2.x-dev requires illuminate/filesystem ^7.0 -> satisfiable by illuminate/filesystem[v7.0.0, v7.0.1, v7.0.2, v7.0.3, v7.0.4, v7.0.5, v7.0.6, 7.x-dev]
- laravel/ui v2.0.0 requires illuminate/filesystem ^7.0 -> satisfiable by illuminate/filesystem[v7.0.0, v7.0.1, v7.0.2, v7.0.3, v7.0.4, v7.0.5, v7.0.6, 7.x-dev]
- laravel/ui v2.0.1 requires illuminate/filesystem ^7.0 -> satisfiable by illuminate/filesystem[v7.0.0, v7.0.1, v7.0.2, v7.0.3, v7.0.4, v7.0.5, v7.0.6, 7.x-dev]
- don't install illuminate/filesystem 7.x-dev|don't install laravel/framework v6.17.1
- don't install illuminate/filesystem v7.0.0|don't install laravel/framework v6.17.1
- don't install illuminate/filesystem v7.0.1|don't install laravel/framework v6.17.1
- don't install illuminate/filesystem v7.0.2|don't install laravel/framework v6.17.1
- don't install illuminate/filesystem v7.0.3|don't install laravel/framework v6.17.1
- don't install illuminate/filesystem v7.0.4|don't install laravel/framework v6.17.1
- don't install illuminate/filesystem v7.0.5|don't install laravel/framework v6.17.1
- don't install illuminate/filesystem v7.0.6|don't install laravel/framework v6.17.1
- Installation request for laravel/framework (locked at v6.17.1, required as ^6.2) ->
- Installation request for laravel/ui ^2.0 -> satisfiable by laravel/ui[2.x-dev, v2.0.0, v2.0.1]

Installation failed, reverting ./composer.json to its original content.
```

Figura 5. Error al instalar laravel/ui

Resolver cambiando de versión

```
/var/www/html/proy_ugm # composer require laravel/ui "^1.2" --dev
```

vea la siguiente figura

```
/var/www/html/proy_ugm # composer require laravel/ui "^1.2" --dev
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Installing laravel/ui (v1.2.0): Downloading (100%)
Writing lock file
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: laravel/tinker
Discovered Package: laravel/ui
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Discovered Package: reliese/laravel
Package manifest generated successfully.
/var/www/html/proy_ugm #
```

Figura 6. Resultado de composer require laravel/ui "^1.2" --dev

Para terminar de configurar ejecute

```
/var/www/html/proy_ugm # php artisan ui vue
```

```
/var/www/html/proy_ugm # php artisan ui bootstrap
```

```
/var/www/html/proy_ugm # npm install && npm run dev
```

Si esta última sentencia marca error de npm no disponible, realice los siguientes pasos:

### Windows

<https://bertofern.wordpress.com/2019/01/08/solucion-node-js-npm-no-reconocido-como-comando-interno-o-externo/>

### Linux

<https://comoinstalar.info/nodejs-en-linux/>

## API RESTFULL

Para este ejemplo vamos a crear un controlador para la tabla candidato

```
/var/www/html/proy_ugm # php artisan make:controller -r Api/CandidatoController
```

Al ejecutar la sentencia anterior se crea la carpeta Api dentro de Http/Controllers y dentro de ella el archivo CandidatoController.php, abra el archivo y reemplace el contenido por el siguiente código:

```
<?php
```

```
namespace App\Http\Controllers\Api;
```

```
use App\Http\Controllers\Api\GenericController as GenericController;
```

```
use Illuminate\Http\Request;
```

```
use Illuminate\Support\Facades\Validator;
```

```
use App\Models\Candidato;
```

```
class CandidatoController extends GenericController
```

```
{
```

```
    /**
```

```
     * Display a listing of the resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
     */
```

```
    public function index()
```

```
    {
```

```
        $candidatos = Candidato::all();
```

```
        $resp = $this->sendResponse($candidatos, "Listado de candidatos");
```

```
        return ($resp);
```

```
    }
```

```

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    //
}

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{

    $validacion = Validator::make($request->all(), [
        'nombrecompleto' => 'unique:candidato|required|max:200',
        'sexo' => 'required'
    ]);

    if ($validacion->fails())

```

```
return $this->sendError("Error de validacion", $validacion->errors());
```

```
$fotocandidato = ""; $perfilcandidato = "";
```

```
if ($request->hasFile('foto')){
```

```
    $foto          = $request->file('foto');
```

```
    $fotocandidato = $foto->getClientOriginalName();
```

```
}
```

```
if ($request->hasFile('perfil')){
```

```
    $perfil          = $request->file('perfil');
```

```
    $perfilcandidato = $perfil->getClientOriginalName();
```

```
}
```

```
$campos          = array(
```

```
    'nombrecompleto' => $request->nombrecompleto,
```

```
    'sexo'           => $request->sexo,
```

```
    'foto'           => $fotocandidato,
```

```
    'perfil'         => $perfilcandidato,
```

```
);
```

```
if ($request->hasFile('foto')) $foto->move(public_path('img'),
```

```
$fotocandidato);
```

```
if ($request->hasFile('perfil')) $perfil->move(public_path('img'),
```

```
$perfilcandidato);
```

```
$candidato = Candidato::create($campos);
```

```
$resp = $this->sendResponse($candidato,  
    "Guardado...");  
return($resp);
```

```
} //-- End store
```

```
/* *
```

```
 * Display the specified resource.
```

```
 *
```

```
 * @param int $id
```

```
 * @return \Illuminate\Http\Response
```

```
 */
```

```
public function show($id)
```

```
{
```

```
    //
```

```
}
```

```
/* *
```

```
 * Show the form for editing the specified resource.
```

```
 *
```

```
 * @param int $id
```

```
 * @return \Illuminate\Http\Response
```

```
 */
```

```
public function edit($id)
```

```
{
```

```

        //
    }

    /**
     * Update the specified resource in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @param int $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        //
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param int $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        //
    }
}

```

Hemos agregado código solo a dos métodos el listado (index) y para guardar (store).

Es importante hacer notar que la clase anterior usa una clase de nombre GenericController. Sin embargo dicho archivo no se crea de manera automática. Por lo tanto, hay que crearlo manualmente en la misma carpeta con el nombre de GenericController.php y cuyo contenido es el siguiente fragmento de código:

```
<?php
```

```
namespace App\Http\Controllers\Api;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller as Controller;
class GenericController extends Controller
{
    /**
     * success response method.
     *
     * @return \Illuminate\Http\Response
     */
    public function sendResponse($result, $message)
    {
        $response = [
            'success' => true,
            'data'     => $result,
            'message'  => $message,
        ];
        return response()->json($response, 200);
    }
    /**
```



```

* return error response.
*
* @return \Illuminate\Http\Response
*/
public function sendError($error, $errorMessagees = [], $code = 404)
{
    $response = [
        'success' => false,
        'message' => $error,
        'data' => [],
    ];
    if(!empty($errorMessagees)){
        $response['data'] = $errorMessagees;
    }
    return response()->json($response, $code);
}
}

```

Esta última clase dispone de dos métodos que serán usados en cualquier subclase que ha de exponer sus métodos a través de la API.

Vamos a probar nuestra api con dos clientes distintos (curl y Postman)

Primero insertamos dos registros a la tabla candidatos

INSERT INTO candidato (nombrecompleto, foto, sexo, perfil)

VALUES

('Ambrosio Cardoso Jimenez','cardoso.png','M','cardoso.pdf'),

('Adolfo Angel Cardoso Vasquez','', 'M','');

curl

curl <http://localhost:8000/api/candidato>

```
{
  • "success":true,
  • "data":[
    1. {
      • "id":1,
      • "nombrecompleto":"Ambrosio Cardoso Jiménez",
      • "foto":"cardoso.png",
      • "sexo":"M",
      • "perfil":"cardoso.pdf"
    },
    2. {
      • "id":2,
      • "nombrecompleto":"Adolfo Angel Cardoso Vasquez",
      • "foto":"",
      • "sexo":"M",
      • "perfil":""
    }
  ],
  • "message":"Listado de candidatos"
}
```

## Postman

The image shows two Postman API request examples. The left panel shows a GET request to `http://localhost:8000/api/candidato` with a status of 200 OK. The response is a JSON object with `"success": true` and a `"data"` array containing two candidate objects. The right panel shows a POST request to the same endpoint with a status of 200 OK. The response is a JSON object with `"success": true` and a `"message": "Guardado..."`.

**GET Request:**

```
GET http://localhost:8000/api/candidato
```

**Response:**

```
{
  "success": true,
  "data": [
    {
      "id": 1,
      "nombrecompleto": "Ambrosio Cardoso Jiménez",
      "foto": "cardoso.png",
      "sexo": "M",
      "perfil": "cardoso.pdf"
    },
    {
      "id": 2,
      "nombrecompleto": "Adolfo Angel Cardoso Vasquez",
      "foto": "",
      "sexo": "M",
      "perfil": ""
    }
  ],
  "message": "Listado de candidatos"
}
```

**POST Request:**

```
POST http://localhost:8000/api/candidato
```

**Body (form-data):**

| KEY  | VALUE                   | DESCRIPTION |
|--|-------------------------|-------------|
| <input checked="" type="checkbox"/> nombrecompleto | Rosa Blanca Pérez López |             |
| <input checked="" type="checkbox"/> sexo           | F                       |             |
| <input checked="" type="checkbox"/> foto           | rosa.jpg                |             |
| <input checked="" type="checkbox"/> perfil         | Select Files            |             |

**Response:**

```
{
  "success": true,
  "data": {
    "nombrecompleto": "Rosa Blanca Pérez López",
    "sexo": "F",
    "foto": "rosa.jpg",
    "perfil": "",
    "id": 3
  },
  "message": "Guardado..."
}
```

Figura 6. Listado de candidatos y agregar nuevo candidato