



MEMORIA FINAL DEL PROYECTO
VIDZ.TO

CICLO FORMATIVO DE GRADO SUPERIOR
Desarrollo de Aplicaciones Web

AUTORES

Franco Ariel Rocha Vallejos

TUTOR / COORDINADOR

Manuel Retamosa Tejero

CURSO

2020/2021

I.E.S. CLARA DEL REY

C/Padre Claret, 8. 28002 Madrid. Tel. 91 519 52 57. Fax 91 519 53 63
correo@iesclaradelrey.es www.iesclaradelrey.es

Índice

1. Título

El título del proyecto es vidz.to , se trata de una aplicación copia de la famosa plataforma de google y la segunda página web más visitada Youtube. Básicamente se trata de una página web donde puedes ver tus vídeos favoritos o incluso subir tus propios vídeos.

2. Introducción

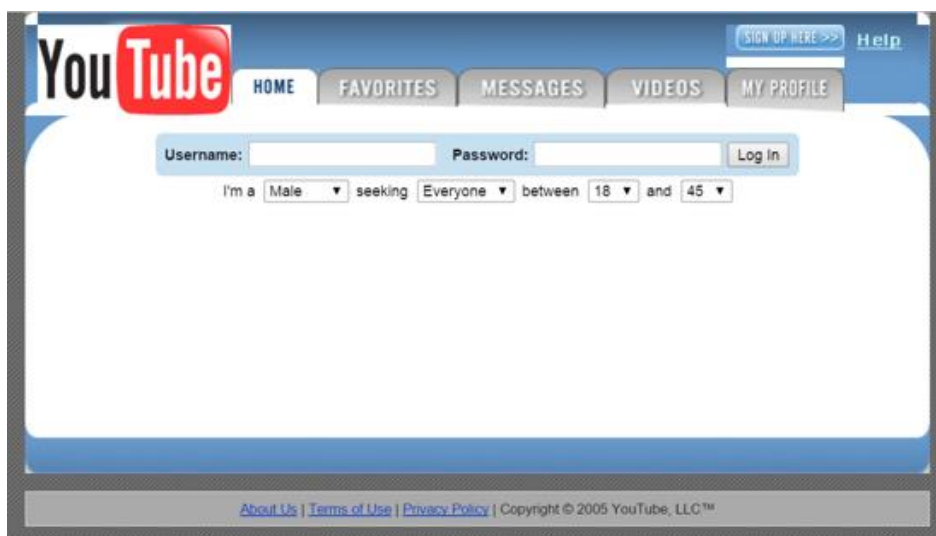
(Español)

Esta es la documentación correspondiente al proyecto de fin de Grado Superior de Desarrollo de Aplicaciones Web (2019-2021), que ha sido realizado entre los meses de Abril-Junio de 2021.

La aplicación surge a modo de curiosidad para intentar entender el interior de la segunda aplicación con más visitas en el mundo, Youtube.

Youtube en sus inicios utilizó el lenguaje de programación php, en la actualidad utiliza Python, C, C++ para el backend. A modo de curiosidad Youtube contiene más de 1 millón de líneas de código en Python.

Toda aplicación comienza con un diseño no muy estético, Youtube es un claro ejemplo, aquí dejo algunas imágenes de sus inicios.

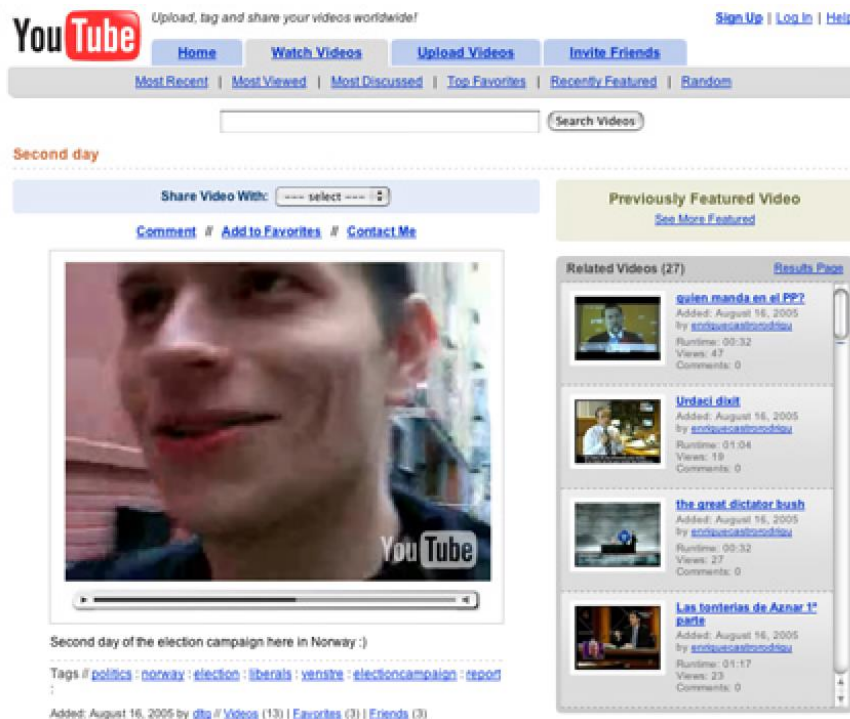


(English)

This is the documentation corresponding to the final project of Web Application Development Course (2019-2021), which has been carried out between the months of April-June 2021.

The application arises as a curiosity to try to understand the interior of the second most visited application in the world, Youtube.

Youtube in its beginnings used the php programming language, currently it uses Python, C, C ++ for the backend. By way of curiosity, YouTube contains more than 1 million lines of code in Python. Every application begins with a not very aesthetic design, YouTube is a clear example, here are some images of its beginnings.



3. Alcance del proyecto y análisis previo

El principal objetivo de este proyecto es la creación de una aplicación en la que las personas puedan ver sus vídeos favoritos o de interés, además la interacción del usuario con la aplicación es clave para este proyecto.

Se implementarán las siguientes funcionalidades en el proyecto:

- Suscribirse a un canal
- Dar like o dislike a un canal
- Número de visualizaciones
- Número de suscripciones
- Fecha de subida del vídeo
- Información del canal e imagen del canal
- Título del canal
- Visualizar vídeo
- Comentarios
- Editar vídeo como propietario del canal
- Editar información del canal como propietario

6. Diseño de la solución escogida

6.1 Herramientas y tecnologías utilizadas

Servidor

PHP

PHP es un lenguaje de código abierto muy popular, adecuado para desarrollo web y que puede ser incrustado en HTML. Es popular porque un gran número de páginas y portales web están creadas con PHP. Incrustado en HTML significa que en un mismo archivo vamos a poder combinar código PHP con código HTML, siguiendo unas reglas.



MySQL

MySQL es un sistema de gestión de bases de datos relacionales de código abierto (RDBMS, por sus siglas en inglés) con un modelo cliente-servidor. RDBMS es un software o servicio utilizado para crear y administrar bases de datos basadas en un modelo relacional.



Laravel (Framework)

Laravel es un framework de PHP del tipo MVC (Modelo-Vista-Controlador).



Cliente

Bootstrap

Bootstrap es un [framework](#) CSS desarrollado por Twitter en 2010, para estandarizar las herramientas de la compañía.

El framework combina [CSS](#) y JavaScript para estilizar los elementos de una página HTML.

El principal objetivo de bootstrap es permitir la construcción de sitios web responsive para dispositivos móviles.



Javascript

[JavaScript](#) es un lenguaje de secuencias de comandos que te permite crear contenido de actualización dinámica, controlar multimedia, animar imágenes y prácticamente todo lo demás.



Laravel Livewire

Livewire es un framework para Laravel que te permite crear interfaces dinámicas de forma simple, sin dejar de lado la comodidad de Blade.



6.2. Diseño previo de la página

En este apartado se incluyen los wireframes utilizados para la creación de la interfaz visual de la aplicación, así como una pequeña descripción de los colores y logos utilizados.

Logotipo

El logo usado será uno de creación propia.



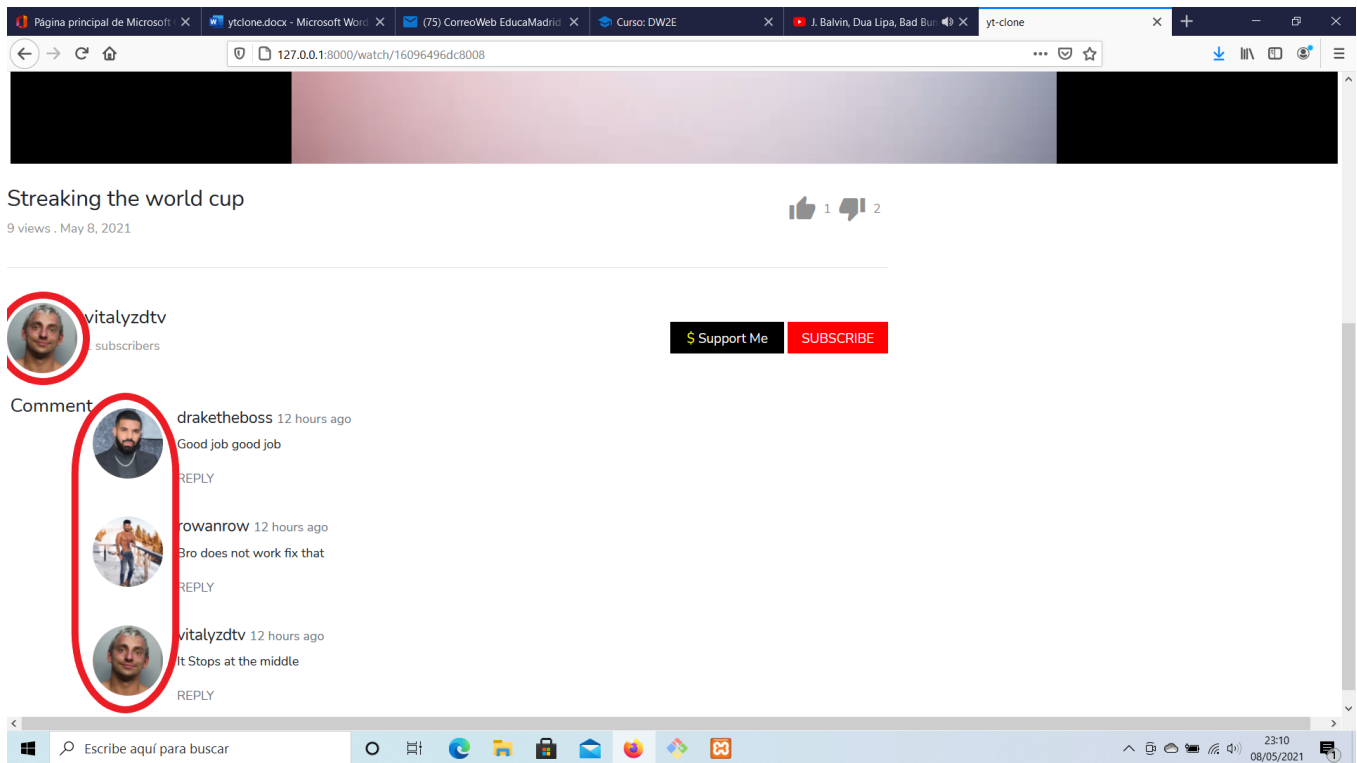
Logo de YTCLONE

Uso y proporción de imágenes

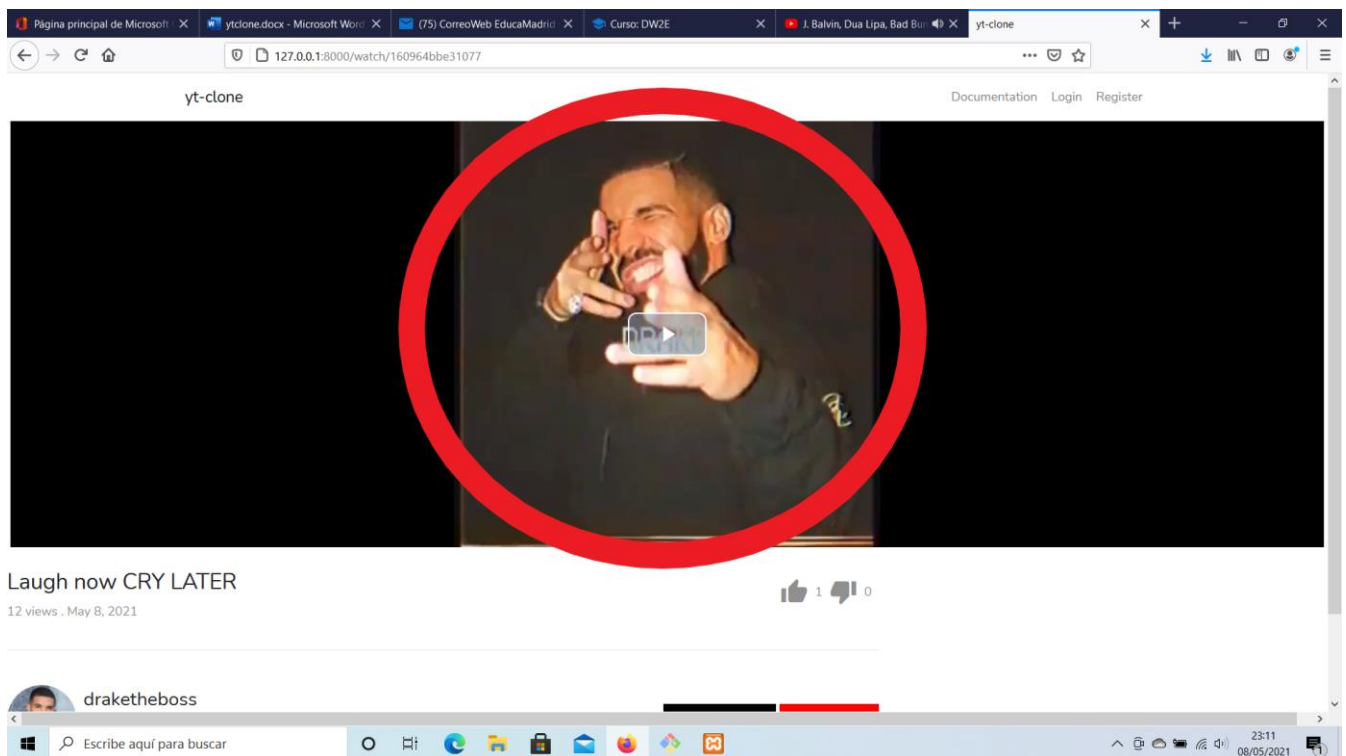
Todas las imagenes son formateadas en .png

En la aplicación utilizamos dos tipos de imágenes:

- La imagen del canal que se guarda en la carpeta "images"



- La imagen del vídeo a modo de “thumbnail” que se guarda en la carpeta “videos”

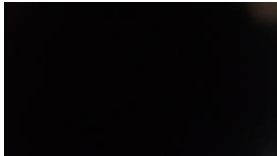


Colores

Los colores utilizados para los estilos de la página web han sido:

#000000

R: 0 B: 0 G: 0



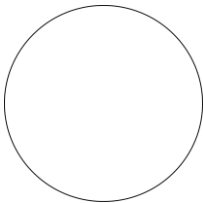
#ff0000

R: 255 B: 0 G: 0



#FFFFFF

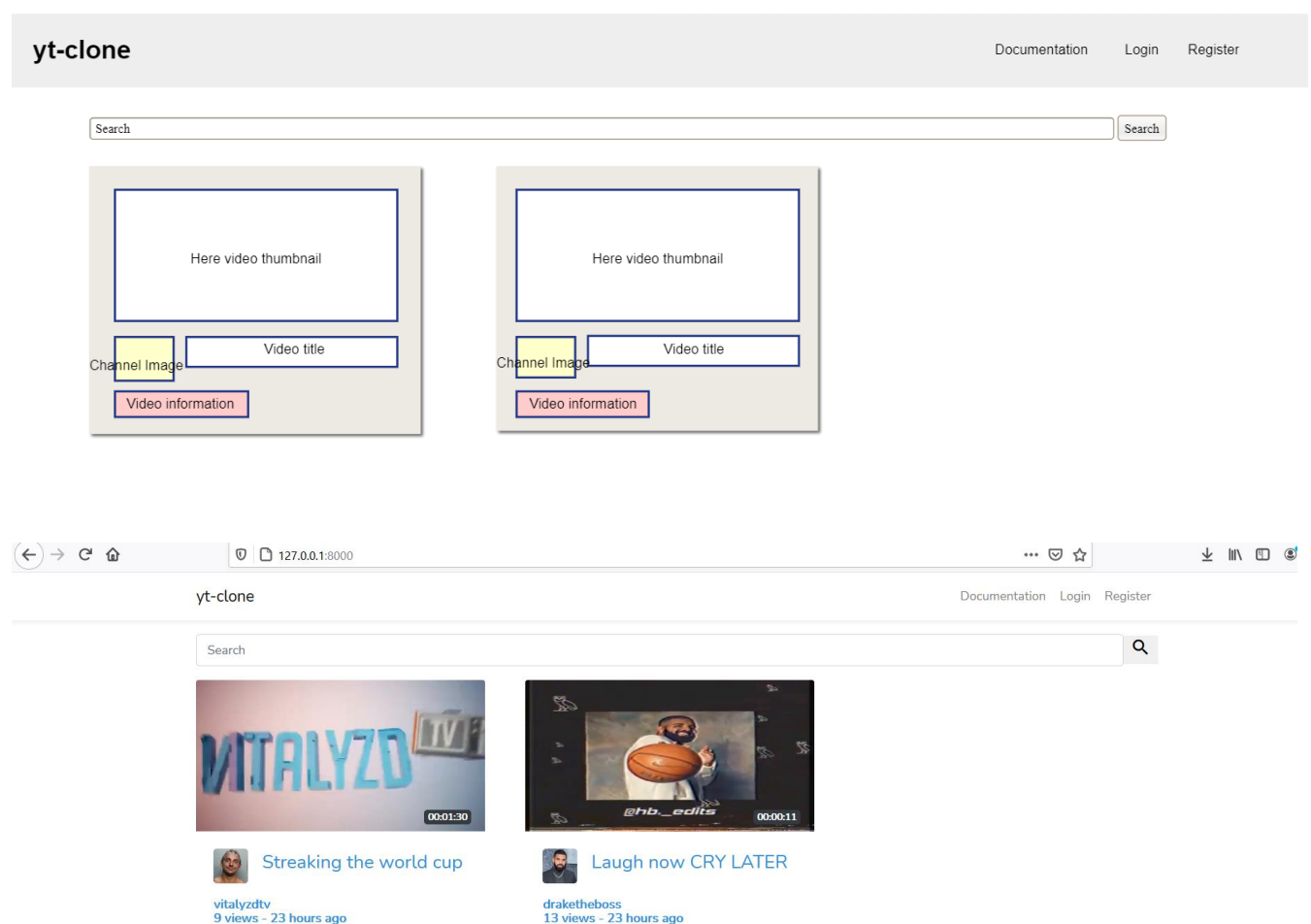
R: 255 B: 255 G: 255



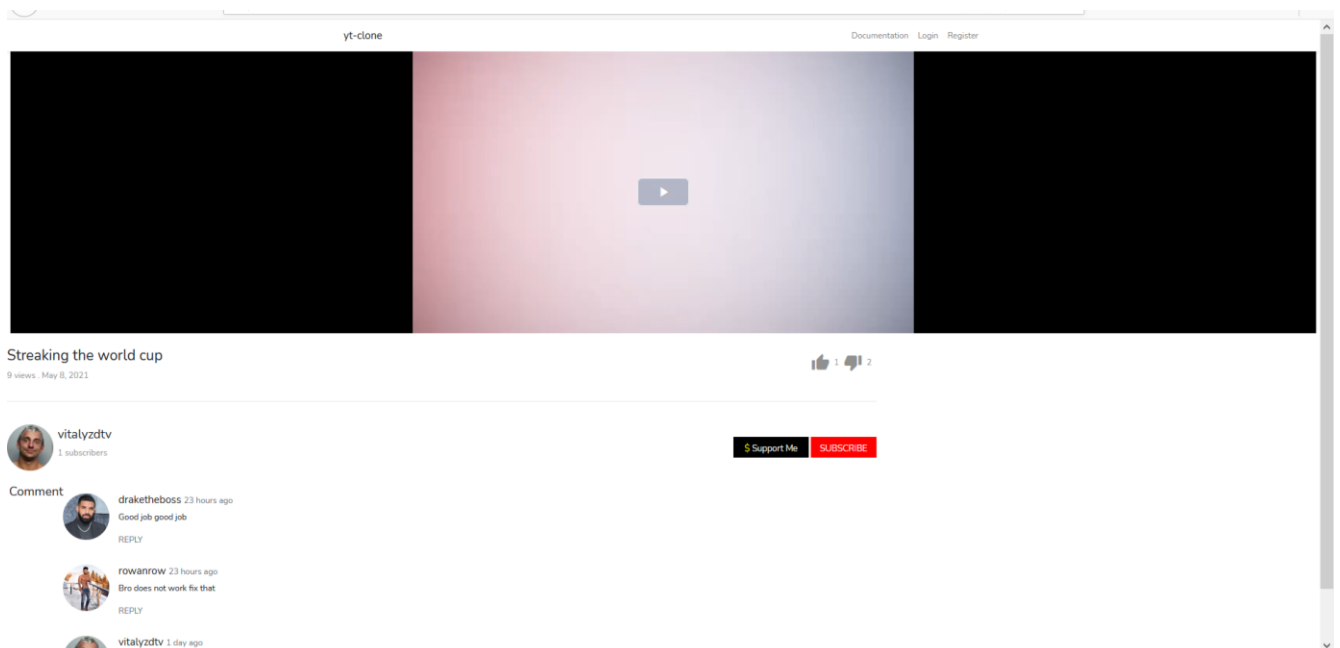
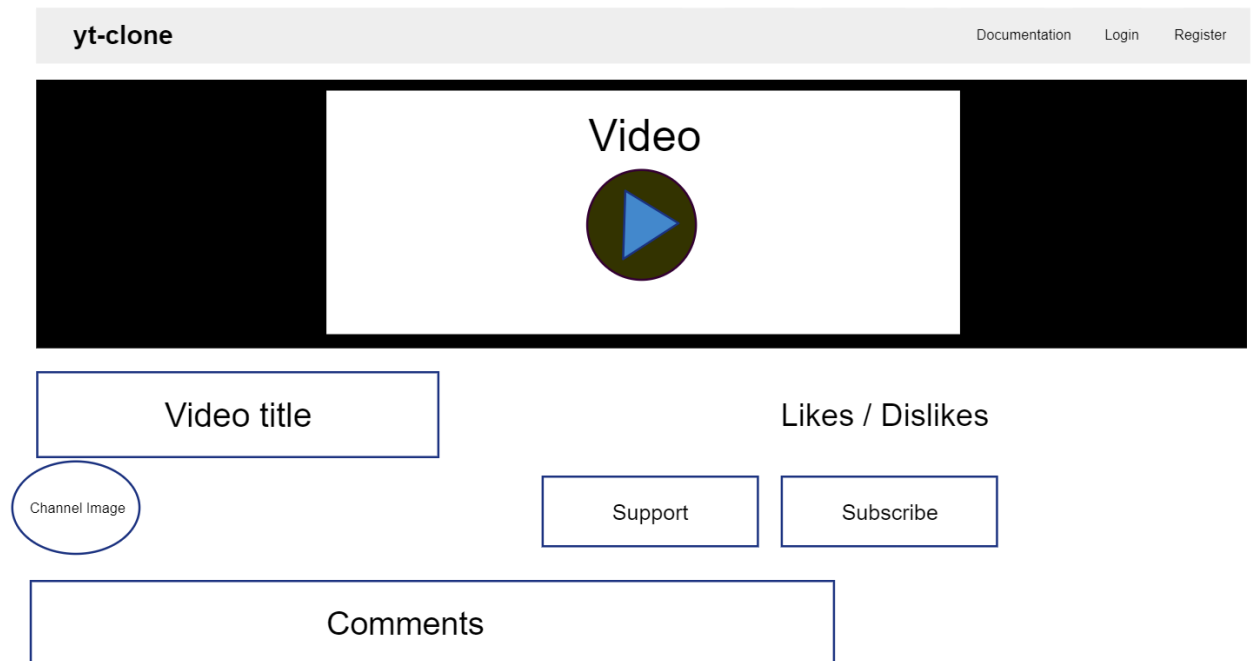
Wireframes: Diseño de la web

Antes de ponernos a codificar el diseño, nos pusimos a diseñar como queríamos que quedara la parte visual. Adjunto a este documento, se incluyen todos los wireframes de los diferentes elementos de la web. A continuación se presentan ejemplos de Wireframe y los aspectos finales de la aplicación.

Página de inicio:



Página de ver vídeo:



9. Conclusiones

9.1 Mejoras

Para una versión posterior de la aplicación, sería ideal poder contar con las siguientes mejoras:

1. Una manera para que el usuario pueda crear una lista de reproducción y compartirla.
2. Botón que se integre con alguna red social, whastapp o sms para compartir el vídeo.
3. Añadir algún procedimiento de ingreso de dinero. Una empresa deberá tener ingresos para poder continuar ofreciendo sus servicios.

9.2 Comentarios

Como comentario final, me gustaría destacar que este proyecto ha conllevado un gran esfuerzo tanto de investigación y aprendizaje , como de organización de trabajo y análisis de los objetivos. He elegido los frameworks de desarrollo Laravel y Livewire, que no he tenido la oportunidad de ver durante el curso. Me ha supuesto un esfuerzo aprender dichos lenguajes, pero me ha resultado muy positivo tanto personalmente como para mi futuro profesional.

De hecho este proyecto me ha guiado en mi decisión de especializarme en PHP así como en los frameworks de Symfony y Laravel. Jamás hubiera pensado como te facilitan la vida dichos frameworks especialmente tras haber desarrollado previamente en el curso una aplicación sin frameworks sino con PHP puro.

Enfrentarme a este proyecto fue bastante frustrante, aunque al final las horas echadas buscando errores y maldiciendo al ordenador fueron más que recompensadas.

Tras haber desarrollado una aplicación de este calibre ya no temo a crear aplicaciones más avanzadas, con la base aprendida en el ciclo y con ayuda de internet es posible hacer aplicaciones muy competentes.

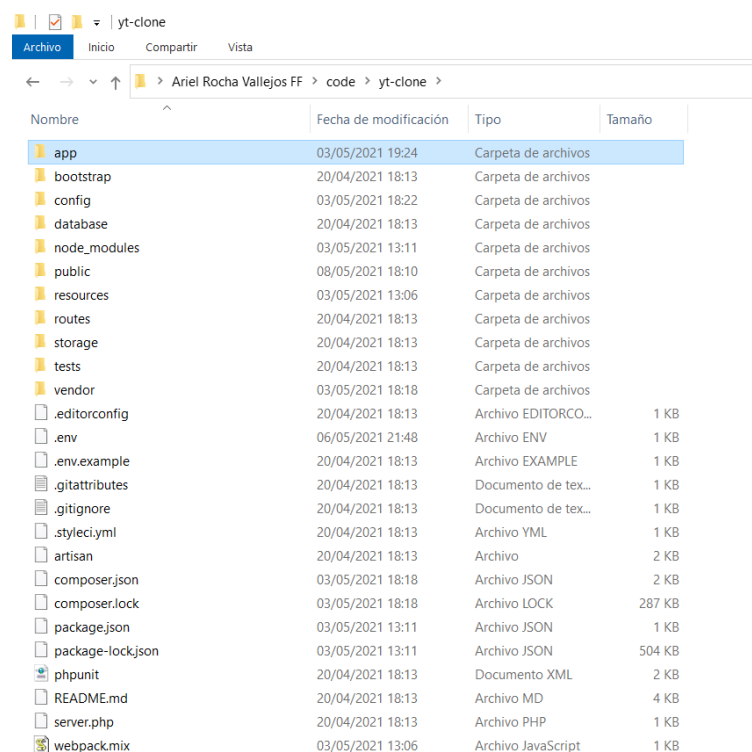
10. Bibliografía y referencias

World Wide Web Consortium	https://www.w3c.es/
Bootstrap	https://www.getbootstrap.com
MySQL	https://dev.mysql.com/doc/
Laravel Livewire	https://laravel-livewire.com/
Laravel Framework	https://laravel.com/docs/8.x
Javascript MDN Mozilla	https://developer.mozilla.org/docs/Web/JavaScript

PildorasInformaticas	https://www.youtube.com/pildorasinformaticas
StackOverflow	https://es.stackoverflow.com/
Wikipedia	https://es.wikipedia.org/
Laracast	https://laracasts.com/discuss
Laravel Freecodecamp	https://www.youtube.com/watch?v=ImtZ5yENzgE
YouTube	https://www.youtube.com/
Patreon	https://www.patreon.com

11. Anexos

Anexo 1. Estructura de carpetas

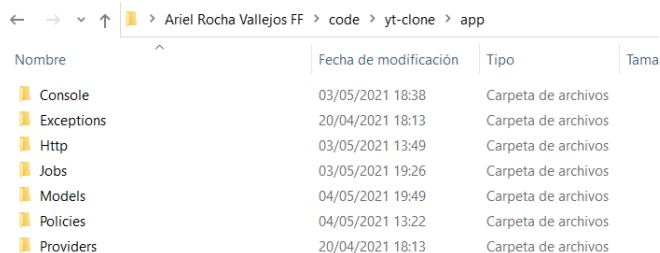


Nombre	Fecha de modificación	Tipo	Tamaño
app	03/05/2021 19:24	Carpeta de archivos	
bootstrap	20/04/2021 18:13	Carpeta de archivos	
config	03/05/2021 18:22	Carpeta de archivos	
database	20/04/2021 18:13	Carpeta de archivos	
node_modules	03/05/2021 13:11	Carpeta de archivos	
public	08/05/2021 18:10	Carpeta de archivos	
resources	03/05/2021 13:06	Carpeta de archivos	
routes	20/04/2021 18:13	Carpeta de archivos	
storage	20/04/2021 18:13	Carpeta de archivos	
tests	20/04/2021 18:13	Carpeta de archivos	
vendor	03/05/2021 18:18	Carpeta de archivos	
.editorconfig	20/04/2021 18:13	Archivo EDITORCO...	1 KB
.env	06/05/2021 21:48	Archivo ENV	1 KB
.env.example	20/04/2021 18:13	Archivo EXAMPLE	1 KB
.gitattributes	20/04/2021 18:13	Documento de tex...	1 KB
.gitignore	20/04/2021 18:13	Documento de tex...	1 KB
.styleci.yml	20/04/2021 18:13	Archivo YML	1 KB
artisan	20/04/2021 18:13	Archivo	2 KB
composer.json	03/05/2021 18:18	Archivo JSON	2 KB
composer.lock	03/05/2021 18:18	Archivo LOCK	287 KB
package.json	03/05/2021 13:11	Archivo JSON	1 KB
package-lock.json	03/05/2021 13:11	Archivo JSON	504 KB
phpunit	20/04/2021 18:13	Documento XML	2 KB
README.md	20/04/2021 18:13	Archivo MD	4 KB
server.php	20/04/2021 18:13	Archivo PHP	1 KB
webpack.mix	03/05/2021 13:06	Archivo JavaScript	1 KB

Se tratan de carpetas creadas por Laravel.

- App

Dentro de app tenemos los diferentes directorios que contienen:



Nombre	Fecha de modificación	Tipo	Tamaño
Console	03/05/2021 18:38	Carpeta de archivos	
Exceptions	20/04/2021 18:13	Carpeta de archivos	
Http	03/05/2021 13:49	Carpeta de archivos	
Jobs	03/05/2021 19:26	Carpeta de archivos	
Models	04/05/2021 19:49	Carpeta de archivos	
Policies	04/05/2021 13:22	Carpeta de archivos	
Providers	20/04/2021 18:13	Carpeta de archivos	

Console: Contiene todos los comandos de Php Artisan. Con laravel puedes crear tu propio comando que se guardará en este directorio.

Exceptions: El directorio de excepciones contiene el controlador de excepciones de su aplicación y también es un buen lugar para colocar cualquier excepción lanzada por su aplicación.

Http: El directorio Http contiene sus controladores, middleware y solicitudes de formulario. Casi toda la lógica para manejar las solicitudes que ingresan a su aplicación se colocará en este directorio.

Jobs: Este directorio no existe de forma predeterminada, pero se creará automáticamente si ejecuta el comando `make: job` Artisan. El directorio jobs contiene los trabajos que se pueden poner en cola para su aplicación. Su aplicación puede poner los trabajos en cola o ejecutarse sincrónicamente dentro del ciclo de vida de la solicitud actual. Los trabajos que se ejecutan sincrónicamente durante la solicitud actual a veces se denominan "comandos", ya que son una implementación del patrón de comando.

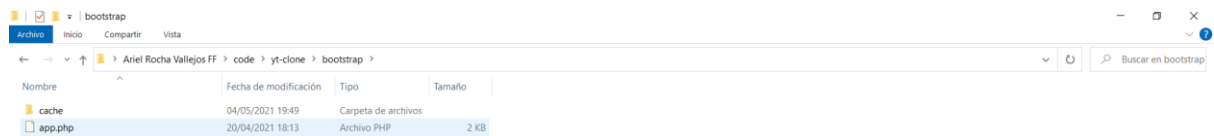
Models: El directorio Models contiene todas sus clases de modelo Eloquent. El ORM de Eloquent incluido con Laravel proporciona una implementación de ActiveRecord para trabajar con su base de datos. Cada tabla de la base de datos tiene un "Modelo" correspondiente que se utiliza para interactuar con esa tabla. Los modelos le permiten consultar datos en sus tablas, así como insertar nuevos registros en la tabla. Básicamente facilita la gestión con la base de datos omitiendo el uso de comandos sql.

Policies: Este directorio no existe de forma predeterminada, pero se creará automáticamente si ejecuta el comando `make: policy`. El directorio de políticas contiene las clases de políticas de autorización para su aplicación. Las políticas se utilizan para determinar si un usuario puede realizar una acción determinada contra un recurso.

Providers: El directorio Providers contiene todos los proveedores de servicios para su aplicación. Los proveedores de servicios inician su aplicación vinculando servicios en el contenedor de servicios, registrando eventos o realizando cualquier otra tarea para preparar su aplicación para las solicitudes entrantes. En una nueva aplicación de Laravel, este directorio ya contendrá varios proveedores. Puede agregar sus propios proveedores a este directorio según sea necesario.

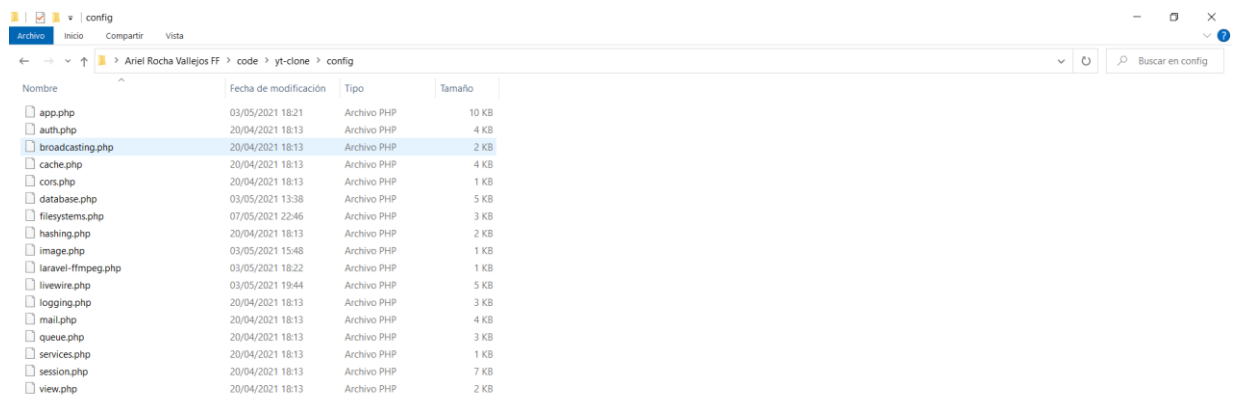
- Bootstrap

El directorio bootstrap como su nombre indica contiene todos los ficheros necesarios para el correcto funcionamiento de Bootstrap.



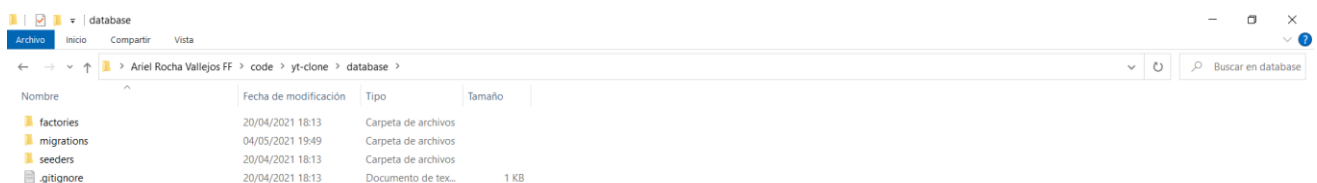
- Config

El directorio de configuración, como su nombre lo indica, contiene todos los archivos de configuración de su aplicación.



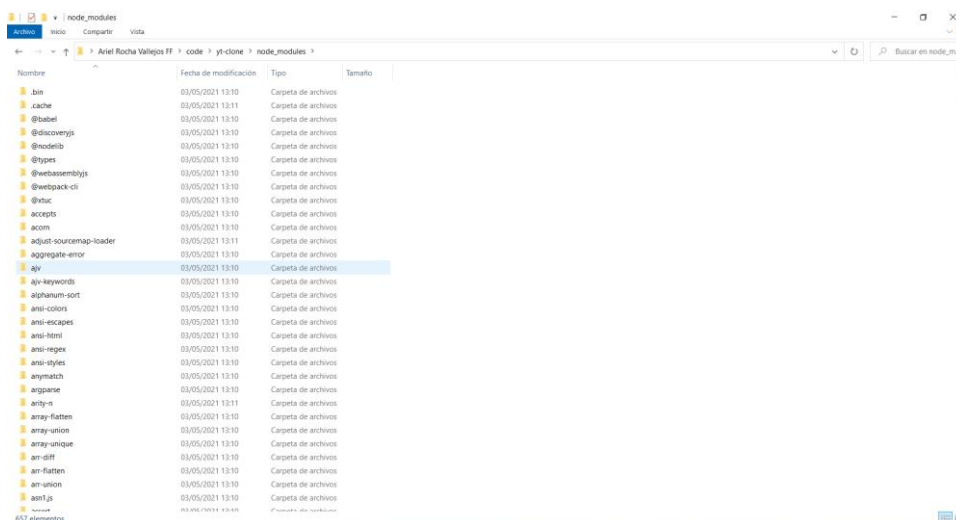
- Database

El directorio de la base de datos contiene las migraciones de la base de datos, las fábricas de modelos y las semillas.



- Node_modules

Contiene todos los paquetes de Node.js que hemos instalado para el desarrollo de nuestro proyecto.



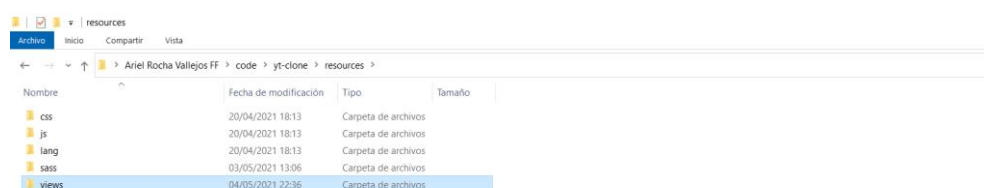
- Public

El directorio public contiene el archivo index.php, que es el punto de entrada para todas las solicitudes que ingresan a su aplicación y configura la carga automática. Este directorio también alberga activos, como imágenes, JavaScript, CSS y vídeos de la aplicación.



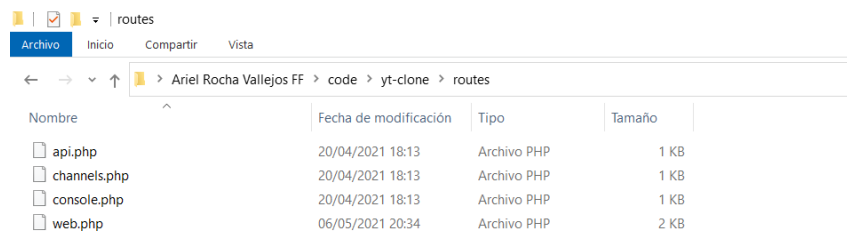
- Resources

El directorio Resources contiene sus vistas, así como sus activos sin compilar, como CSS o JavaScript. Este directorio también aloja todos los archivos de idioma.



- Routes

El directorio Routes contiene todas las definiciones de ruta para su aplicación. De forma predeterminada, se incluyen varios archivos de ruta con Laravel: web.php, api.php, console.php y channels.php.



Nombre	Fecha de modificación	Tipo	Tamaño
api.php	20/04/2021 18:13	Archivo PHP	1 KB
channels.php	20/04/2021 18:13	Archivo PHP	1 KB
console.php	20/04/2021 18:13	Archivo PHP	1 KB
web.php	06/05/2021 20:34	Archivo PHP	2 KB

El archivo web.php contiene rutas que RouteServiceProvider coloca en el grupo de middleware web, que proporciona el estado de la sesión, la protección CSRF y el cifrado de cookies. Si la aplicación no ofrece una API RESTful sin estado, es probable que todas las rutas estén definidas en el archivo web.php.

El archivo api.php contiene rutas que RouteServiceProvider coloca en el grupo de middleware api. Estas rutas están destinadas a estar sin estado, por lo que las solicitudes que ingresan a la aplicación a través de estas rutas deben autenticarse mediante tokens y no tendrán acceso al estado de la sesión.

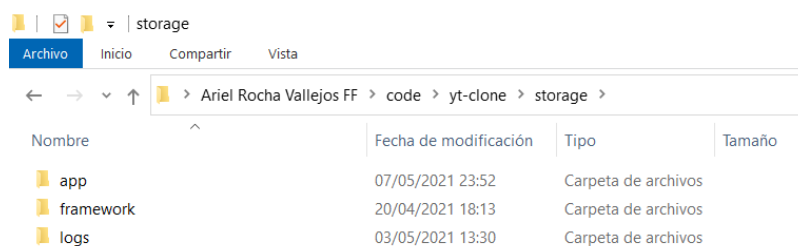
El archivo console.php es donde puedes definir todos tus comandos de consola basados en el cierre. Cada cierre está vinculado a una instancia de comando, lo que permite un enfoque simple para interactuar con los métodos de E / S de cada comando. Aunque este archivo no define rutas HTTP, define puntos de entrada (rutas) basados en la consola en su aplicación.

El archivo channels.php es donde puede registrar todos los canales de transmisión de eventos que admite su aplicación.

- Storage

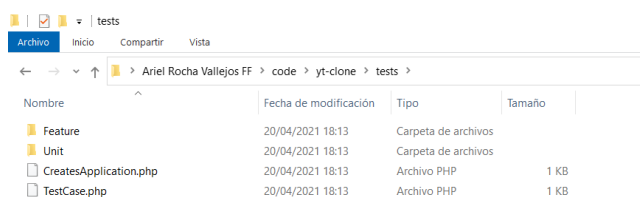
El directorio Storage contiene sus registros, plantillas Blade compiladas, sesiones basadas en archivos, cachés de archivos y otros archivos generados por el marco. Este directorio está segregado en directorios de aplicaciones, marcos y registros. El directorio “App” se puede utilizar para almacenar cualquier archivo generado por su aplicación. El directorio “framework” se utiliza para almacenar cachés y archivos generados por el marco. Finalmente, el directorio “logs” contiene los archivos de registro de su aplicación.

El directorio `storage / app / public` se puede utilizar para almacenar archivos generados por el usuario, como los avatares de perfil, que deberían ser de acceso público. Se debe crear un enlace simbólico en `public / storage` que apunte a este directorio. Puede crear el enlace usando el comando `php artisan storage:link` Artisan.



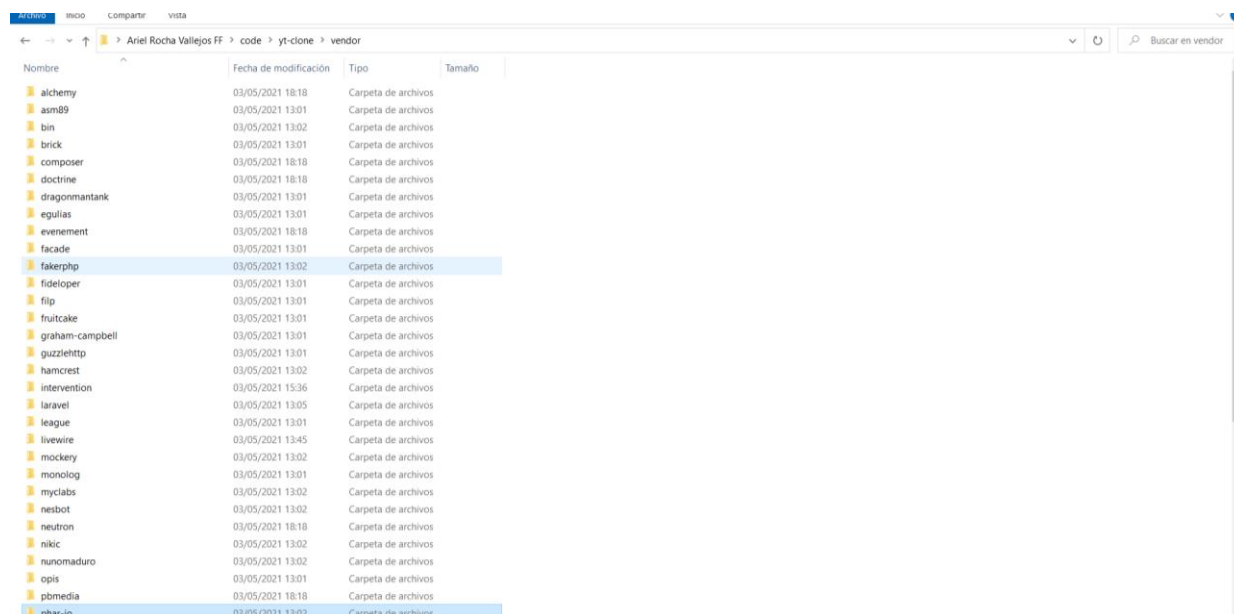
- Tests

El directorio “Tests” contiene sus pruebas automatizadas. Las pruebas unitarias de ejemplo PHPUnit y las pruebas de funciones se proporcionan listas para usar. Cada clase de prueba debe tener como sufijo la palabra Prueba. Puede ejecutar sus pruebas usando los comandos `phpunit` o `php vendor / bin / phpunit`. O, si desea una representación más detallada y hermosa de los resultados de su prueba, puede ejecutar sus pruebas usando el comando `php artisan test` Artisan.



- Vendor

El directorio “vendor” contiene sus dependencias de Composer.



Un fichero a tener en cuenta que se encuentra en el directorio raíz es el fichero llamado “.env”. El archivo .env predeterminado de Laravel contiene algunos valores de configuración comunes que pueden diferir en función de si su aplicación se está ejecutando localmente o en un servidor web de producción. Estos valores luego se recuperan de varios archivos de configuración de Laravel dentro del directorio de configuración usando la función env de Laravel como la conexión a la base de datos.

Anexo 2. Guía para implementar el proyecto en tu equipo

Se deberán descargar las siguiente aplicaciones en el equipo:

Vagrant

Instalar vagrant. Vagrant es una herramienta que nos ayuda a crear y manejar máquinas virtuales con un mismo entorno de trabajo. Nos permite definir los servicios a instalar así como también sus configuraciones. Está pensado para trabajar en entornos locales y lo podemos utilizar con shell scripts, Chef, Puppet o Ansible.

Cabe destacar que vagrant no tiene la capacidad para correr una maquina virtual sino que simplemente se encarga de las características con las que debe crearse esa VM y los complementos a instalar. Para poder trabajar con las máquinas virtuales es necesario que también instalamos [VirtualBox](#).

Virtualbox

VirtualBox es un software para virtualización, también conocido como hipervisor de tipo 2, que se utilizar para virtualizar sistemas operativos dentro de nuestro ordenador existente, creando lo que se conoce como máquina virtual. Un hipervisor de tipo 2 se diferencia con los de tipo 1 en que necesita un sistema operativo para funcionar, a

diferencia de los de tipo 1 en los que el propio hipervisor funciona sobre el hardware, o máquina host.

Composer

Composer es un sistema de gestión de paquetes para programar en PHP el cual provee los formatos estándar necesarios para manejar dependencias y librerías de PHP. Composer hará la tarea de implementar librerías mucho más fácil ya que simplemente se utilizara un comando para agregar las librerías al proyecto.

Git

Git es un software de control de versiones. Lo utilizaremos para descargar la última de versión de una aplicación que necesitaremos llamada Laravel Homestead.

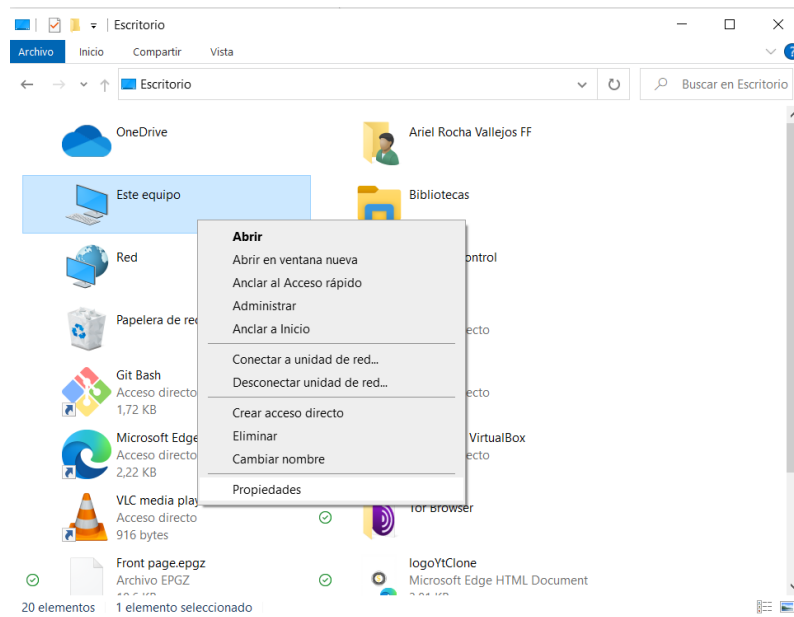
Laravel Homestead

Laravel Homestead es el box de Vagrant pre-empaquetado oficial que brinda un maravilloso entorno de desarrollo sin la necesidad de instalar PHP, un servidor web, ni ningún otro software de servidor en tu máquina local. Los boxes de Vagrant son completamente desechables. Si algo sale mal, simplemente puedes destruir y volver a crear el box en cuestión de minutos. Laravel Homestead trae todas las herramientas necesarias para poder empezar a desarrollar sin necesidad de hacer un setup del equipo.

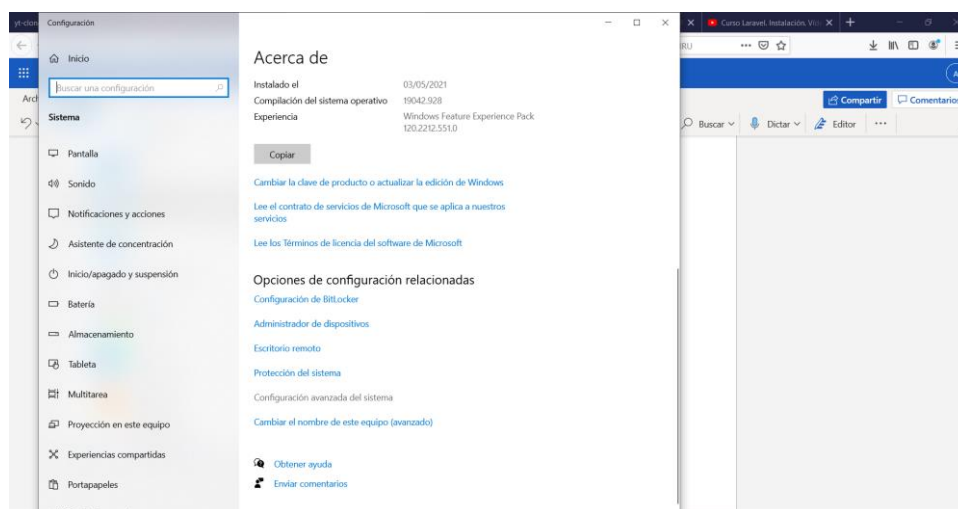
Una vez descargadas las aplicaciones se deberá seguir los siguientes pasos:

- Se deberá crear un path a la aplicación de virtualbox de tal manera que sea accesible desde cualquier lugar de nuestro equipo. Para ello simplemente accedemos a las variables de entorno del equipo:

Click derecho en “Este equipo”.

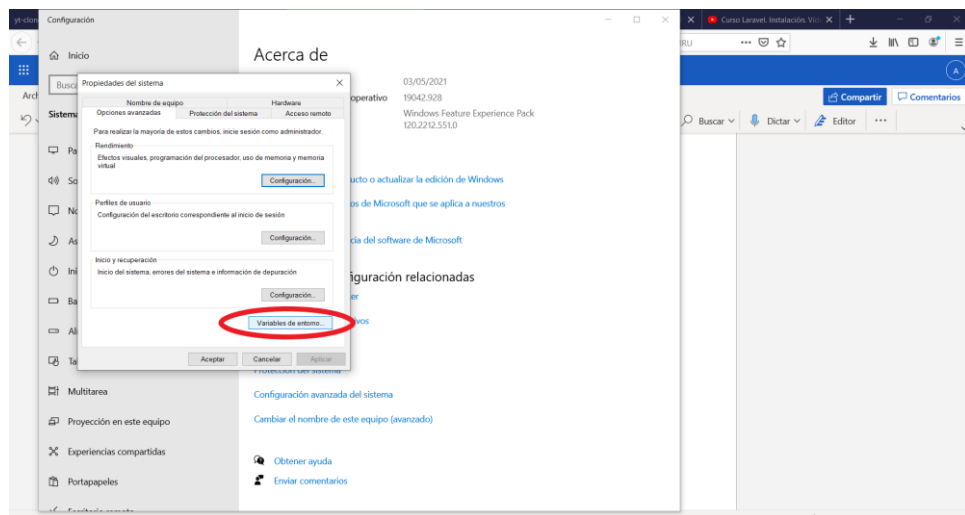


Accedemos a propiedades.

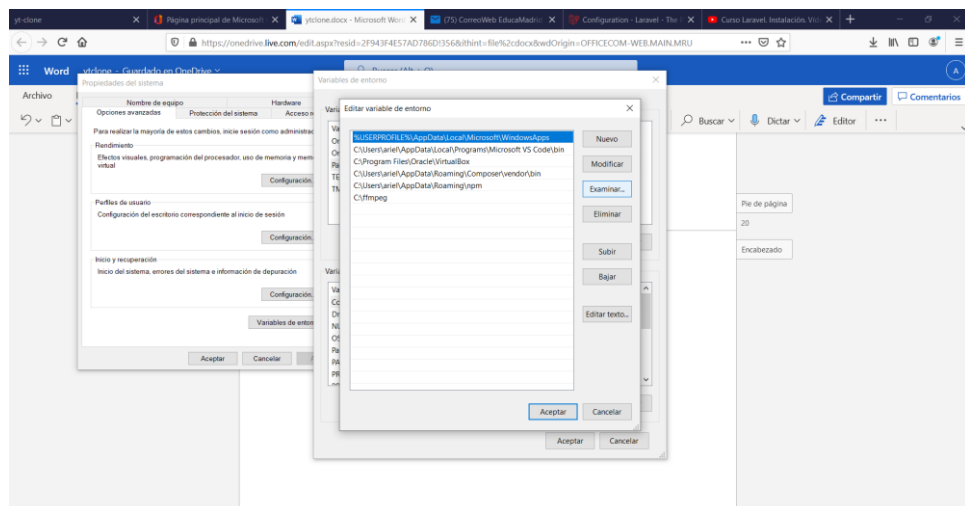


Una vez en propiedades accedemos a configuración avanzada del sistema.

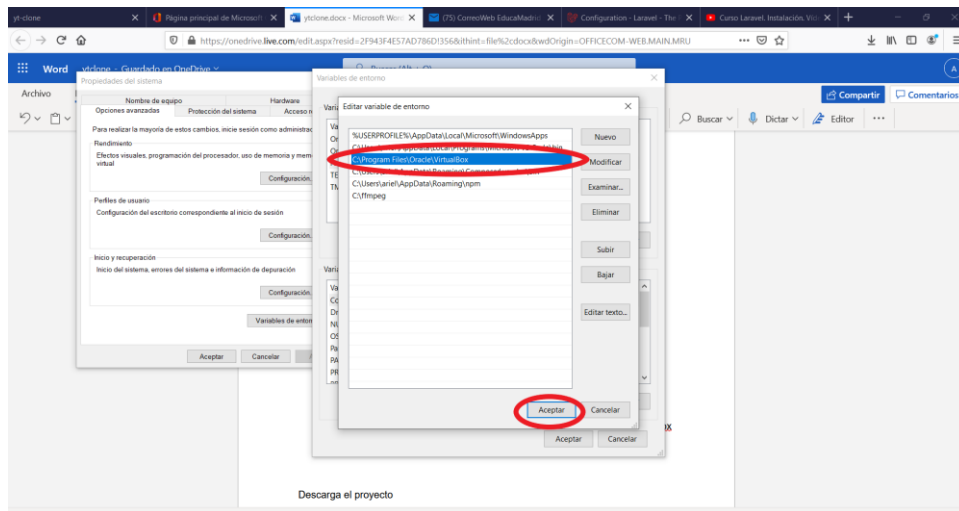
Se resalta en la imagen siguiente al botón para acceder a las variables de entorno.



Dentro de las variables de entorno seleccionamos la casilla donde pone “PATH”, una vez seleccionado clicamos el botón de editar nos saldrá la siguiente pestaña:




Clicamos el botón “Nuevo” y escribimos la ruta donde se encuentra virtualbox en mi caso:



Clicamos el botón “Aceptar” en todas las pestañas que nos vayamos encontrando. Y ya estaría configurado, de todas maneras podemos hacer la prueba ejecutando el siguiente comando:

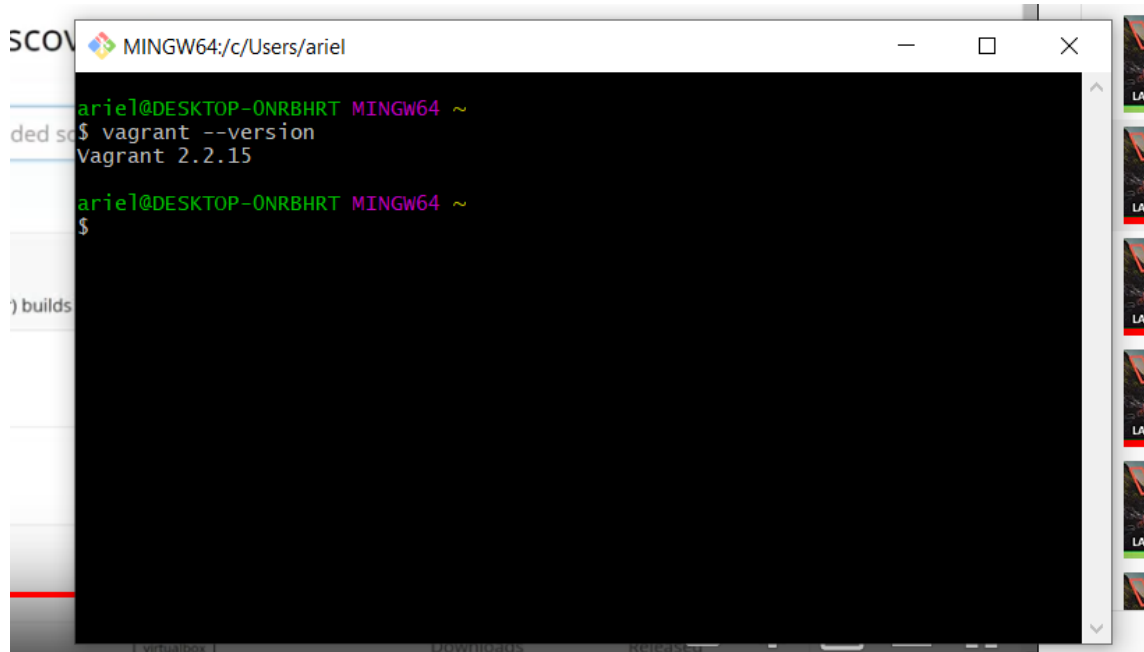
`vboxmanage --version`

 `MINGW64:/c/Users/ariel/code/yt-clone`

```
ariel@DESKTOP-0NRBHRT MINGW64 ~/code/yt-clone
$ vboxmanage --version
6.1.22r144080

ariel@DESKTOP-0NRBHRT MINGW64 ~/code/yt-clone
$
```

Comprobamos que a la aplicación vagrant también se puede acceder por consola desde cualquier lugar dentro del equipo usando el siguiente comando:



```
MINGW64:/c/Users/ariel
ariel@DESKTOP-0NRBHRT MINGW64 ~
$ vagrant --version
Vagrant 2.2.15
ariel@DESKTOP-0NRBHRT MINGW64 ~
$
```

En caso de que no se pudiera obtener la versión repetiremos el mismo proceso que con Virtualbox solo que esta vez añadiremos la aplicación vagrant a Path dentro de las variables de entorno.

El siguiente paso será pasar el laravel homestead dentro de la máquina virtual, esto lo haremos utilizando el siguiente comando en la terminal/console:



```
ariel@DESKTOP-0NRBHRT MINGW64 ~
$ vagrant box add laravel/homestead
```

Una vez ejecutemos el comando nos aparecerán las siguientes opciones:

```
MINGW32:/c/Users/Juan
$ VBoxManage --version
5.1.30r118389

Juan@Juan-PC MINGW32 ~
$ vagrant --version
Vagrant 2.0.0

Juan@Juan-PC MINGW32 ~
$ vagrant box add laravel/homestead
==> box: Loading metadata for box 'laravel/homestead'
    box: URL: https://vagrantcloud.com/laravel/homestead
This box can work with multiple providers! The providers that it
can work with are listed below. Please review the list and choose
the provider you will be working with.

1) parallels
2) virtualbox
3) vmware_desktop

Enter your choice: 2
==> box: Adding box 'laravel/homestead' (v4.0.0) for provider: virtualbox
    box: Downloading: https://vagrantcloud.com/laravel/boxes/homestead/versions/
4.0.0/providers/virtualbox.box
```

Seleccionamos la opción de Virtualbox en este caso el número 2.

Una vez finalizado el proceso tocará hacer uso de la aplicación git.

Mediante el comando :

- Git clone <https://github.com/laravel/homestead.git> Homestead

```
MINGW32:/c/Users/Juan
    box: URL: https://vagrantcloud.com/laravel/homestead
This box can work with multiple providers! The providers that it
can work with are listed below. Please review the list and choose
the provider you will be working with.

1) parallels
2) virtualbox
3) vmware_desktop

Enter your choice: 2
==> box: Adding box 'laravel/homestead' (v4.0.0) for provider: virtualbox
    box: Downloading: https://vagrantcloud.com/laravel/boxes/homestead/versions/
4.0.0/providers/virtualbox.box
    box:
==> box: Successfully added box 'laravel/homestead' (v4.0.0) for 'virtualbox'!

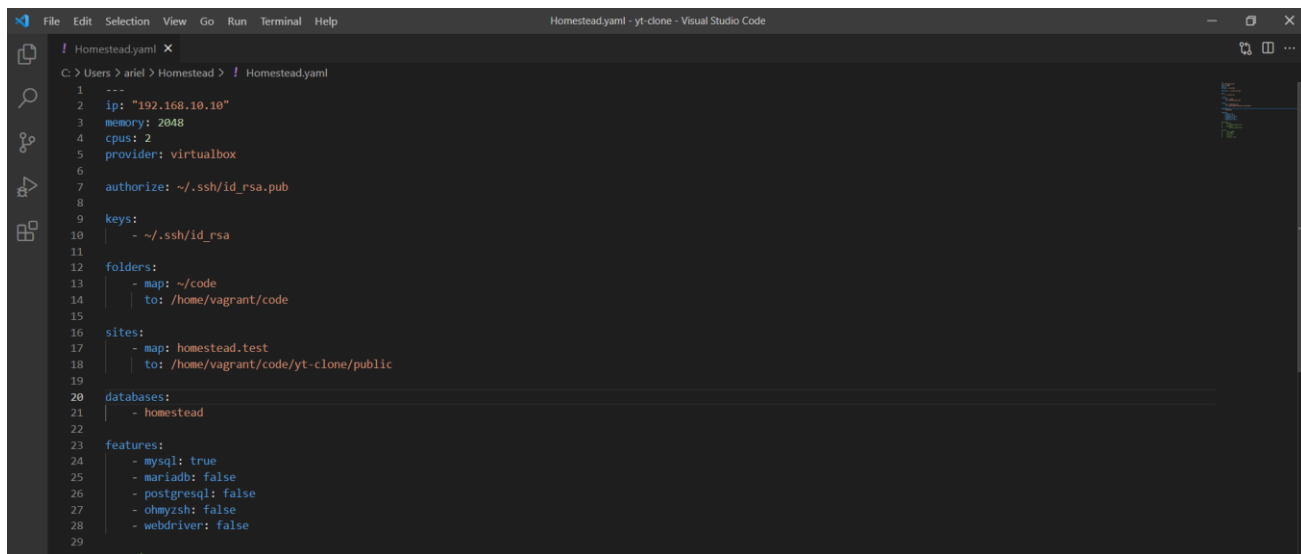
Juan@Juan-PC MINGW32 ~
$ git clone https://github.com/laravel/homestead.git Homestead
Cloning into 'Homestead'...
fatal: unable to access 'https://github.com/laravel/homestead.git/': Could not r
esolve host: github.com

Juan@Juan-PC MINGW32 ~
$
```


Una vez hecho esto ya habríamos instalado todo lo único que haría falta sería instalar composer, el gestor de librerías de PHP.

Una vez instaladas todas las aplicaciones anteriores y configurado las variables de entorno pasaremos a editar el fichero Homestead.yaml que se encuentra en el directorio de Homestead.

Cuando abrimos el fichero Homestead.yaml nos encontramos:



```
1 ---
2 ip: "192.168.10.10"
3 memory: 2048
4 cpus: 2
5 provider: virtualbox
6
7 authorize: ~/.ssh/id_rsa.pub
8
9 keys:
10 | - ~/.ssh/id_rsa
11
12 folders:
13 | - map: ~/code
14 |   to: /home/vagrant/code
15
16 sites:
17 | - map: homestead.test
18 |   to: /home/vagrant/code/yt-clone/public
19
20 databases:
21 | - homestead
22
23 features:
24 | - mysql: true
25 | - mariadb: false
26 | - postgresql: false
27 | - ohmyzsh: false
28 | - webdriver: false
29
```

En el documento por partes tenemos:

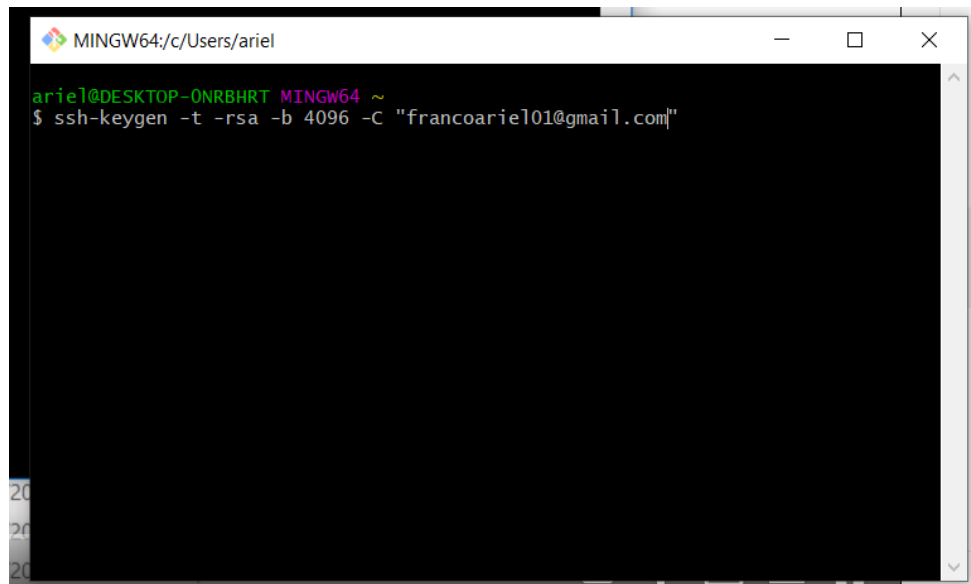
- El número ip
- La memoria ram dedicada a la máquina virtual
- La cantidad de cpus dedicadas a la máquina virtual
- El proveedor que en este caso es VirtualBox
- A continuación encontramos 2 claves: Tenemos una clave pública que se denomina ~/.ssh/id_rsa.pub y tenemos la clave privada como ~/.ssh/id_rsa esta última no tendrá ninguna terminación.

Al ser la primera vez no tendremos creada la carpeta ssh que contiene ambas claves pública y privada.

Para crear nuestra carpeta ssh y crear nuestras dos claves utilizaremos el siguiente comando:

```
ssh-keygen -t rsa -b 4096 -C "Escribir email aquí"
```

Un ejemplo:



```
MINGW64/c/Users/ariel
ariel@DESKTOP-ONRBHRT MINGW64 ~
$ ssh-keygen -t -rsa -b 4096 -C "francoariel01@gmail.com"
```

- El siguiente apartado que encontramos se trata de “Folders” es decir las carpetas:

Bajo este apartado está “map” y “to”. “Map” hace referencia a la carpeta en tu ordenador local y “to” hace referencia a la carpeta que creará en tu máquina virtual. Son las carpetas compartidas entre la máquina virtual y la máquina local.

Como el apartado “map” hace referencia a una carpeta denominada code dentro del directorio raíz del usuario lo que se deberá hacer es crear una carpeta llamada “code” dentro del directorio raíz del usuario.

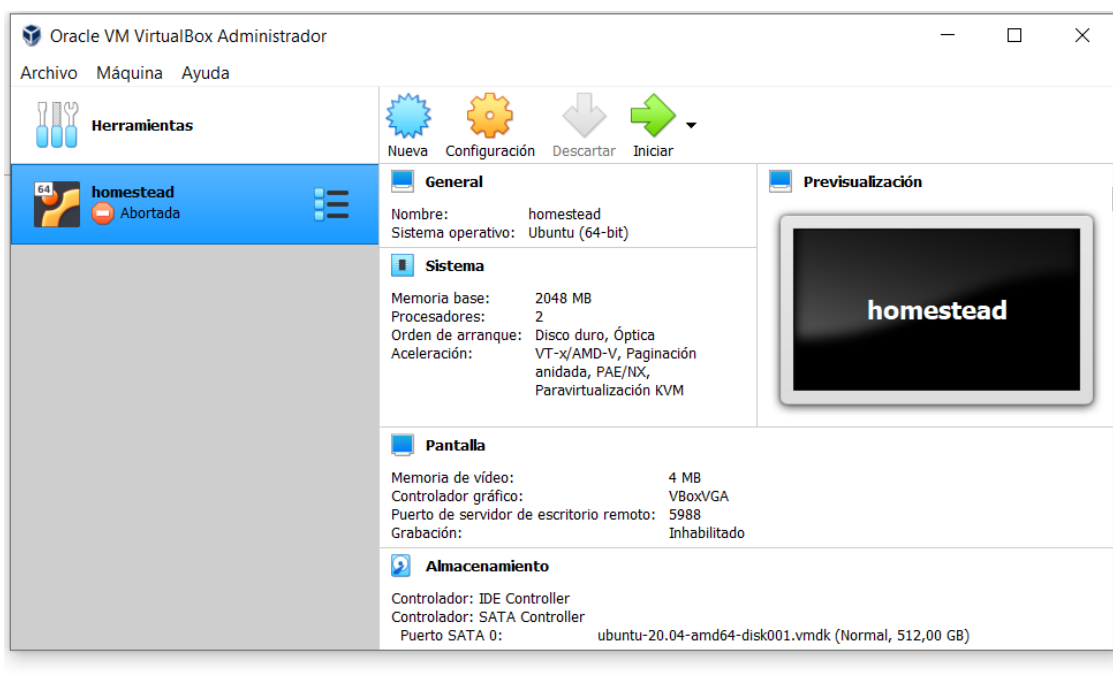
- En el apartado sites encontraremos el enlace que introduciremos en nuestra máquina local para abrir el sitio web del proyecto.
- En el apartado databases se encuentra la base de datos que se creará con el nombre indicado unavez se ejecute el comando “php artisan migrate”.

Una vez hechos los cambios deberemos dirigirnos a la carpeta llamada Homestead y mediante la terminal ejecutaremos el comando :

vagrant up

```
MINGW64:/c/Users/ariel/Homestead
ariel@DESKTOP-0NRBHRT MINGW64 ~/Homestead (main)
$ vagrant up
```

Tras ejecutar se pedirán permisos de administrador que deberemos aceptar. Una vez se haya completado el proceso de ejecutar este comando se habrá creado una nueva máquina virtual en virtualbox.



Tras esto ya tendremos nuestro equipo en orden para poder obtener el proyecto de GitHub

Y ejecutarlo en la máquina local.

Descarga el proyecto

A continuación, descargamos el proyecto disponible a través del nuestro repositorio en Github.

La manera de hacerlo es antes que nada ir al fichero Homestead.yaml dentro del directorio code. Y editar la configuración en el apartado sites:

```
authorize: ~/.ssh/id_rsa.pub
keys:
  - ~/.ssh/id_rsa
folders:
  - map: ~/code
    to: /home/vagrant/code
sites:
  - map: homestead.test
    to: /home/vagrant/code/Laravel/public
databases:
  - homestead

# blackfire:
#   - id: foo
#     token: bar
#     client-id: foo
#     client-token: bar

# ports:
#   - send: 50000
#     to: 5000
```

Deberemos añadir un nombre de la aplicación en la localización del círculo resaltado, en mi caso llamaré a la aplicación Laravel. Este comando es un indicador para el ordenador para poder localizar nuestra carpeta que contiene el proyecto.

El siguiente paso es crear el proyecto con el nombre que anteriormente hemos introducido en el lugar del círculo rojo. En mi caso Laravel por lo tanto crearemos una aplicación ejecutando el siguiente comando dentro del directorio code:

```
composer create-project laravel/laravel Laravel
```

En el comando la última palabra Laravel deberá sustituirse por el nombre que daremos a nuestra carpeta donde se guardará el proyecto. En mi caso la carpeta donde guardaré mi proyecto se llamará Laravel.

```
vagrant@homestead:~/code$ composer create-project laravel/laravel Laravel
```

Una vez creada la carpeta nos dirigiremos a ella e iniciaremos un git, es decir utilizaremos el comando **git init** dentro de la carpeta que contiene nuestro proyecto en mi caso la carpeta se llama Laravel.

```
$ git init
```

Una vez iniciada el directorio de git ya podemos extraer el proyecto desde el repositorio de GitHub a través del siguiente enlace: