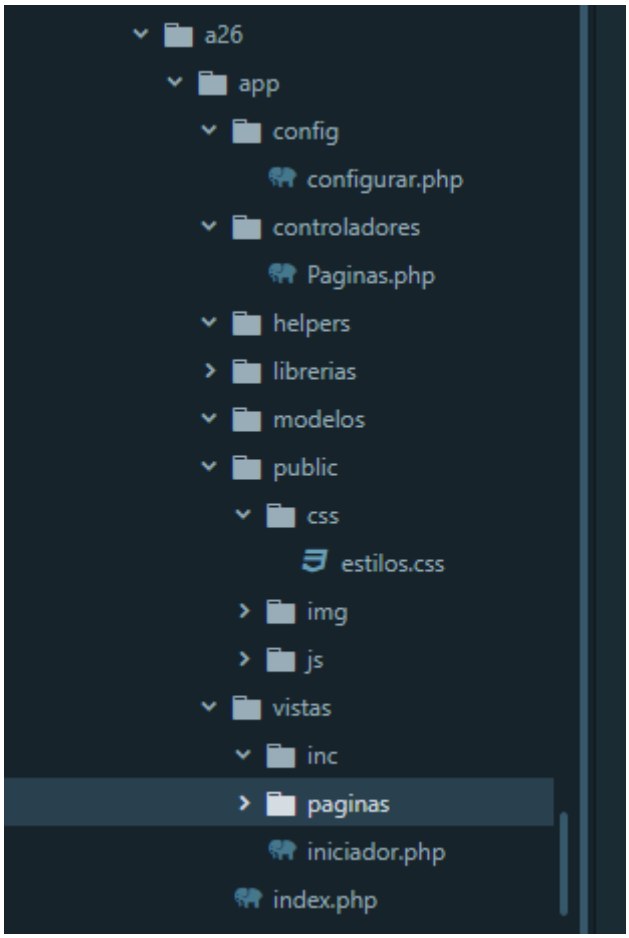
Vamos a crear un framework propio desde cero. Esto nos permitirá agilizar el tiempo de desarrollo ya que podremos reutilizar toda la estructura de carpetas y código, además de reducir los posibles bugs, ya que será un código optimizado.

Ejercicio 1. Estructura de carpetas

Existen muchas posibilidades para crear la estructura de carpetas de nuestro framework. Una opción es dividir en dos carpetas el proyecto: una app, con los ficheros de nuestra aplicación, y otra public, que tendrá todos los ficheros estáticos:

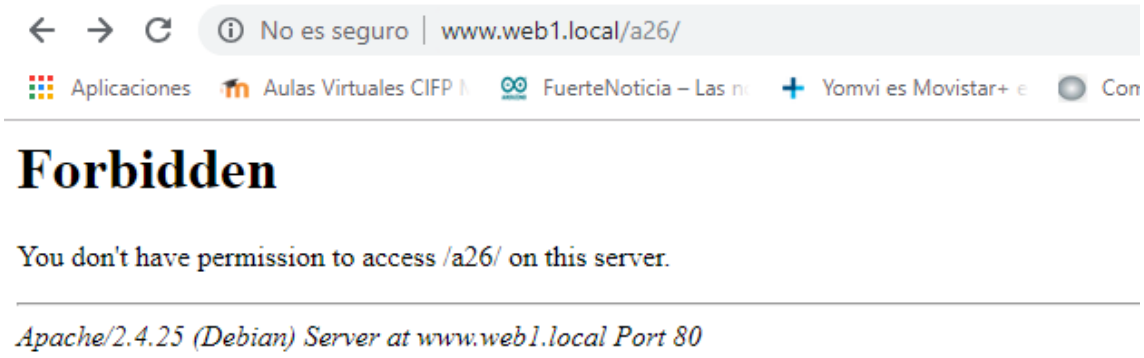


Su utilidad es permitir una mayor organización del proyecto siguiendo un patrón muy utilizado como es el MVC, básicamente la estructura cuenta con la carpeta **principal (App)**, dentro de ella tenemos la carpeta **"config"** para configuraciones, la carpeta **controladores,** para guardar todos los controladores de la aplicación, **helpers** una carpeta de **ayuda, librerías** para almacenar las que usamos, la carpeta **public** donde están nuestros ficheros estáticos de la aplicación. En el se almacena el **css, las imágenes, los archivos js.**

Por último, la **carpeta vista**s que contiene las vistas de la aplicación

Ejercicio 2. Configuraciones iniciales

Tal y como está configurado Apache nos permite mostrar las carpetas debajo de app en el navegador. Esta configuración no nos interesa. Averigua cómo desactivarla para que no se vean las carpetas. No utilices .htaccess en este apartado.



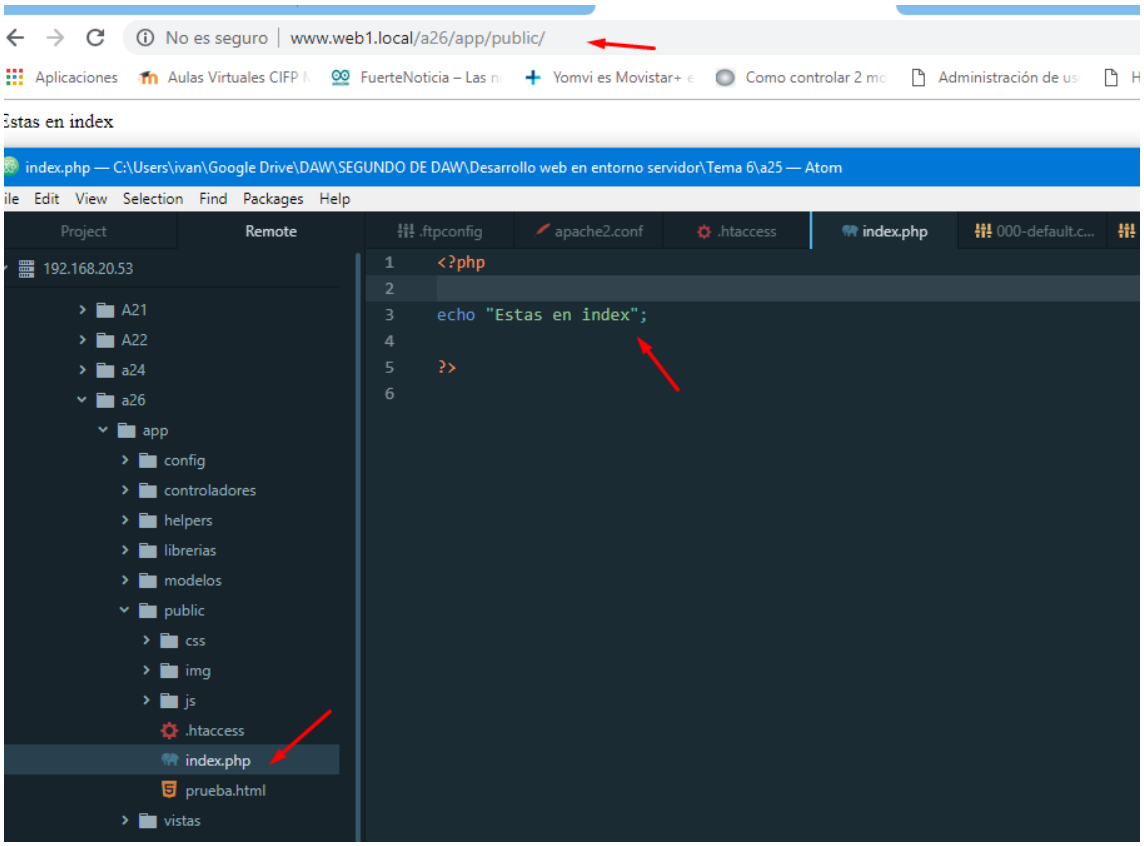
Entramos en apache.conf y escribimos la siguiente directiva

```
<Directory /var/www/html/a26>
  Options -Indexes
  AllowOverride All
</Directory>
```

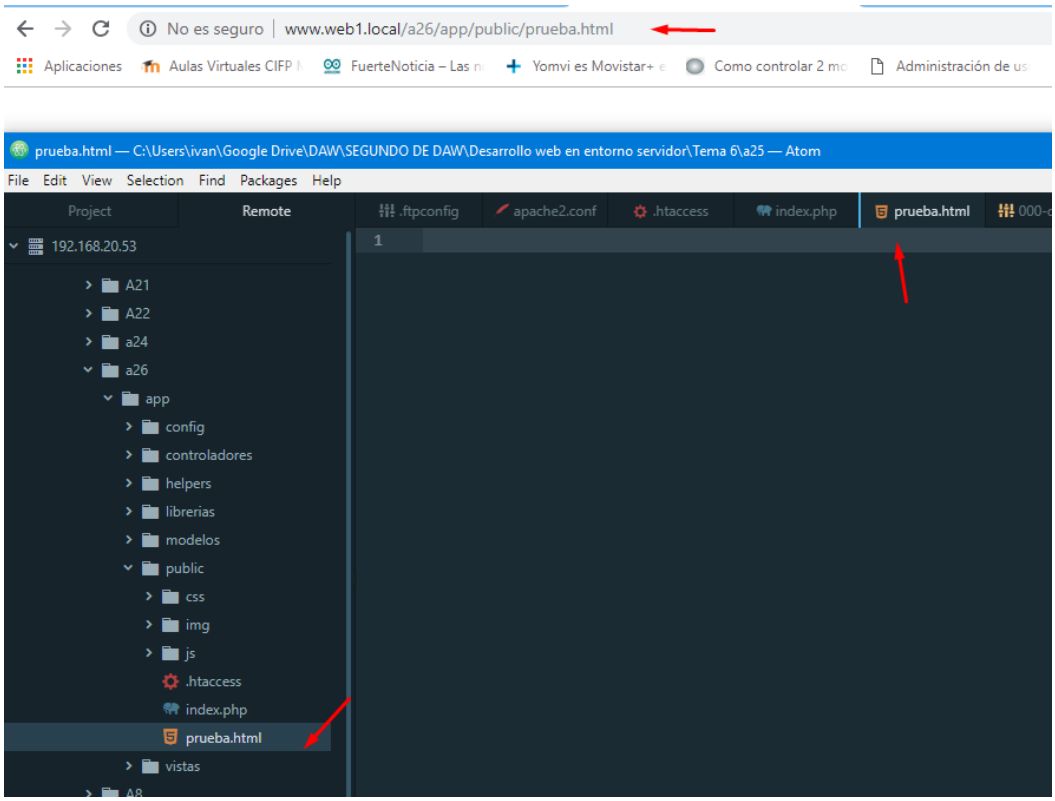
Ejercicio 3. Rutas amigables y redirección a index.php

Queremos que nuestra aplicación trabaje con rutas amigables. Así, si escribimos http://www.misitio.com/articulos/actualizar/2 estaremos indicando que queremos actualizar el artículo con id=2 (controlador/método/parámetros)

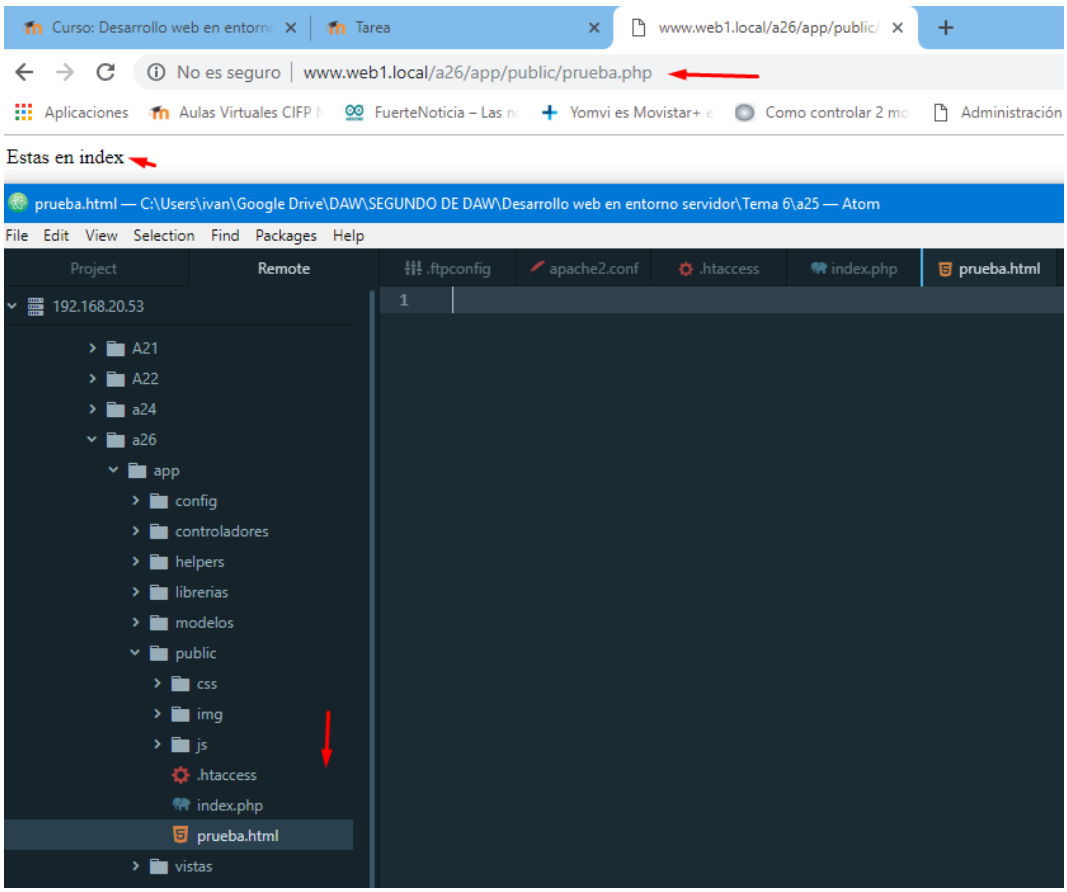
Además nos interesa que si se accede a un fichero que no existe se redireccione a index.php. Si el fichero existe se mostrará.



Comprobamos entrando en el archivo “prueba.html” que si existe en la carpeta

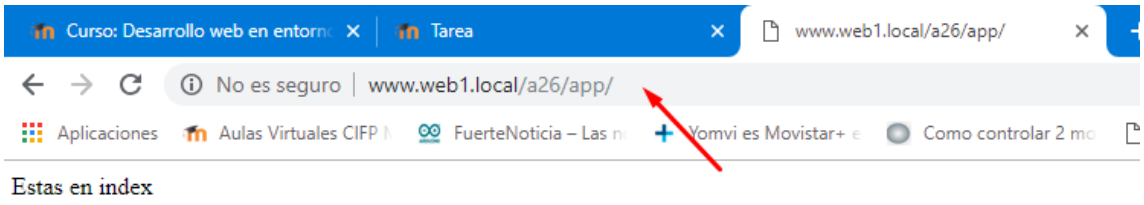


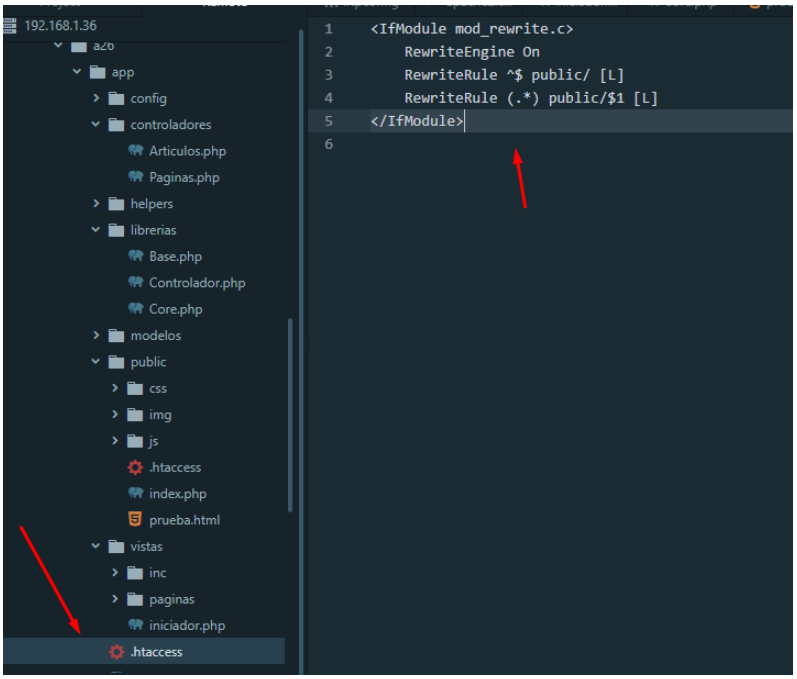
Probamos con el archivo prueba.php que no existe en la aplicación, vemos como redirige a Index



Ejercicio 4

Vamos a crear el último .htaccess para poder acceder a nuestros ficheros en el servidor sin necesidad de mostrar la carpeta public en la URL. Este fichero se ubicará en la carpeta raíz del proyecto, a la misma altura que app y public:

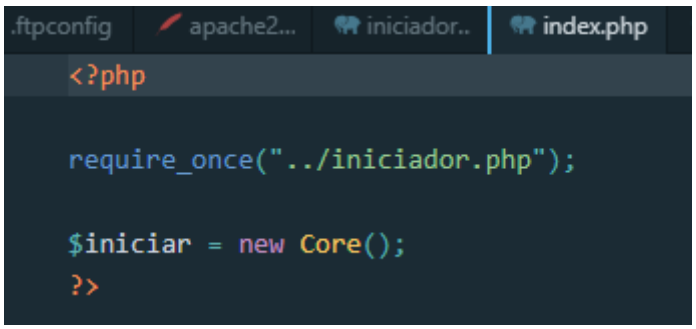




Ejercicio 5

El fichero **index.php** va a cargar el iniciador e instanciar la clase Core.php que ahora codificaremos:

Index.php



Iniciador.php

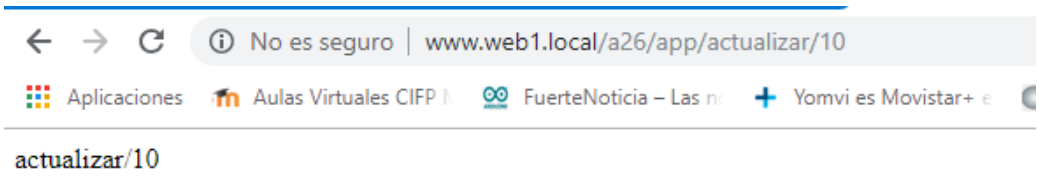
```php
<?php

require_once('librerias/Base.php');
require_once('librerias/Controlador.php');
require_once('librerias/Core.php');

 ?>

```

Ejercicio 6. La clase Core.php

La clase **Core.php** es la principal de nuestro programa. De momento va a analizar la URL e imprimirla por pantalla.

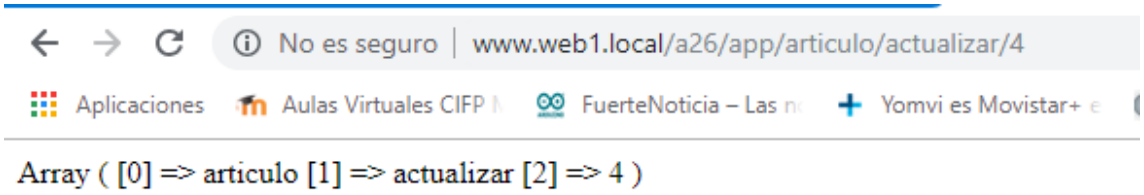
```php
<?php
class Core {

    protected $controladorActual ="Paginas"; //Controlador por defecto
    protected $metodoActual = "index"; // Método por defecto
    protected $parametros = []; // Por defecto no hay parámetros

//constructor donde se pasa a la variable url la funcion getUrl()
    public function __construct() {
        $url = $this->getUrl();
    }

//funcion que obtiene el get
    public function getURL(){
        echo $_GET["url"];
    }
}

 ?>
```

Obtiene los datos que se han introducido por Get



actualizar/10

Ejercicio 7. Parsear la URL



Código comentado

```php
    funcion que obtiene el get
    public function getURL(){

        if (isset($_GET["url"])) { //si existe parametro en la url
            $url = rtrim($_GET["url"], "/"); //rtrim quita los espacios en blanco o / del final del string
            $url = filter_var($url, FILTER_SANITIZE_URL);//elimina todos los caracteres excepto letras y digitos
            $url = explode("/", $url);//divide el string en varios

            if($url[0] == "Articulos"){ //cogemos la posicion 0 del array y detectamos si es articulos en ese caso el
                                        //controaldor pasara a sera Articulos
              $this->controladorActual = "Articulos";

            }
            return print_r($url);
        }

        echo $_GET["url"];

    }
```

Ejercicio 8. Cargar el controlador por la URL

Vamos a seguir modificando el fichero Core.php para que en lugar de cargar un controlador por defecto (en nuestro caso Paginas.php) cargue el controlador que se pase por la URL.

Constructor

```php
public function __construct() {

        $url = $this->getUrl();

        // Busca en controladores si el controlador existe
        if (file_exists("../controladores/" . ucwords($url[0]) . ".php")) {
            $this->controladorActual = ucwords($url[0]);

            //unset Paginas que es el controlador por defecto
            unset($url[0]);
        }

        //Importamos el controlador nuevo
        require_once "../controladores/" . $this->controladorActual . ".php";
        $this->controladorActual = new $this->controladorActual;
}
```

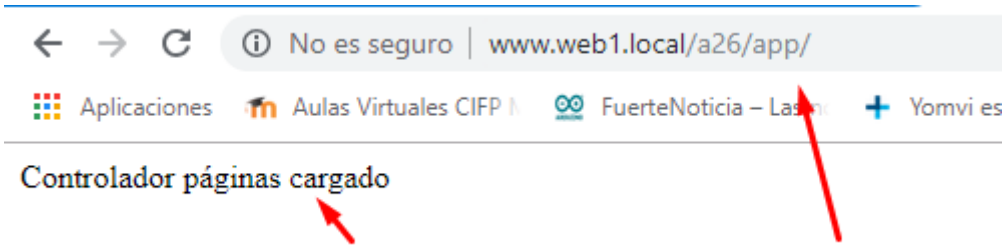
El código busca si existe ese archivo en la carpeta controladores y le asigna a la variable “controladorActual” el nombre del controlador con el primer carácter en mayúscula.

Vacía con un unset la variable url.

Por último, importa el controlador según el valor en la variable controladorActual.

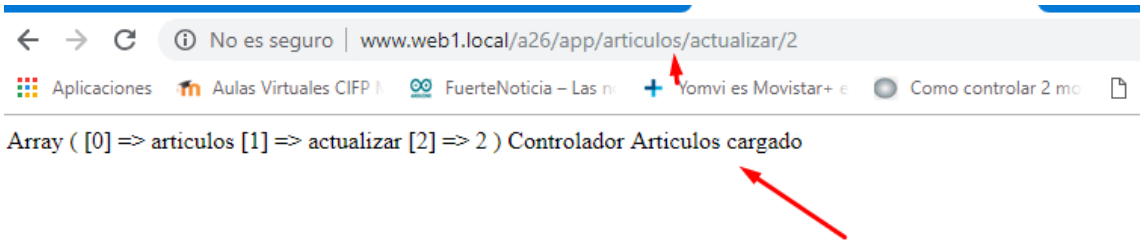
Tienes que testear que se cargue el controlador por defecto.

El controlador se carga por defecto



Con un código similar puedes crear otro controlador (llámale como quieras) y probar que eres capaz de cargarlo desde la URL. Por ejemplo, si llamas a /articulos tendrás que crear un fichero en la carpeta app/controladores que se llame Articulos.php y con un código similar al anterior.

Comprobamos llamando en la url /articulos si nos carga el controlador de artículos.



Realizamos la comprobación

```php
public function getURL(){

    if (isset($_GET["url"])) { //si existe parametro en la url
        $url = rtrim($_GET["url"], "/"); //rtrim quita los espacios en blanco o / del final del string
        $url = filter_var($url, FILTER_SANITIZE_URL);//elimina todos los caracteres excepto letras y digitos
        $url = explode("/", $url);//divide el string en varios

        if($url[0] == "articulos"){ //cogemos la posicion 0 del array y detectamos si es articulos en ese caso el
                                    //controaldor pasara a ser Articulos
          $this->controladorActual = "Articulos";

        }
        return print_r($url);
    }

    echo $_GET["url"];

}
```