***APUNTES DWES***

Tabla de contenido

[PHP 2](#_Toc160198016)

[-Basics 2](#_Toc160198017)

[-Objetos 2](#_Toc160198018)

[-Autoload 2](#_Toc160198019)

[-Instalación SQLite 2](#_Toc160198020)

[-Instalación MariaDB (Windows) 2](#_Toc160198021)

[-Conexión con PHP a MariaDB 2](#_Toc160198022)

[-Sesiones 2](#_Toc160198023)

[-Cookies 2](#_Toc160198024)

[DJANGO 2](#_Toc160198025)

[-Creación de Proyecto e Inicialización 2](#_Toc160198026)

[-Shell 2](#_Toc160198027)

[-ORM 2](#_Toc160198028)

[-Paginación 2](#_Toc160198029)

[-Django-Rest-Framework 2](#_Toc160198030)

# PHP

## -Basics

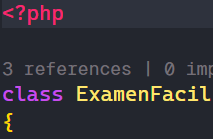
* Las variables se crearán usando $ y utilizando camelCase. Ej: $miVarible.
* Las constantes se definen usando define(‘NOMBRE’, valores) y usando SNAKE\_CASE.
* Para unir se usará un punto (.) en vez del más (+).
* Para ejecutar php dentro del HTML usaremos <?php ?> o <?= ?>, en el caso del segundo equivale a hacer <?php echo ?>.
* Para crear una función usaremos function nombreFunct(tipo $valor):tipoDevuelto{}.
* El foreach tendrá la siguiente estructura ($variable as $key => $value), donde el =>$value se podrá omitir.

## -Objetos

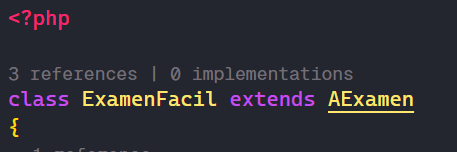
Las clases en PHP se escribirán en PascalCase.

En caso de querer crear una clase abstracta se pondrá abstract antes del class.

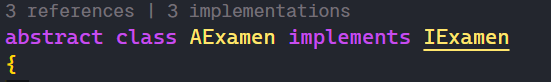
Las clases se crearán de la siguiente forma:



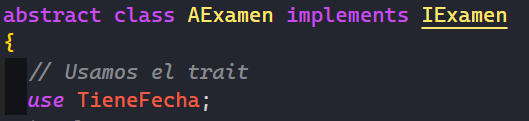
Las clases podrán extender de otras clases usando el extends:



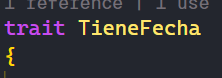
Las clases a su vez pueden implementar interfaces usando el implements:



En las clases de PHP podremos usar **traits**, se podrán añadir dentro de la clase que lo vaya a usar o en un archivo aparte. Para usarlo debemos añadir a la clase el: use {nombreDelTrait}



El trait se creará con trait {nombreDelTrait}:



## -Autoload

Para poder cargar todo de manera automáticamente

define('DOC\_ROOT',dirname(dirname(\_FILE\_)));

spl\_autoload\_register(function ($class) {

$classPath = DOC\_ROOT . "/src/"; // Ruta absoluta del directorio de clases

require\_once $classPath . $class . ".php";

});

## -Instalación SQLite

Descargamos la última versión de SQLite (binaries y tools) y lo añadimos los archivos del .zip en C:\Windows\System32 para poder usarlo desde la terminal.

Creamos el archivo en caso de que no exista siguiendo los siguientes pasos:

* En la terminal escribimos sqlite3 para iniciarlo.
* Escribimos el nombre de nuestra base de datos, con “.open NOMBRE\_BD.db”, en caso de que no exista SQLite creará el archivo.

## -Instalación MariaDB (Windows)

Instalamos la última versión de MariaDB, asignamos contraseña a root y añadimos el directorio que contenga C:\Program Files\MariaDB 11.2\bin en el PATH.

Para crear una copia de la base de datos deberemos hacer lo siguiente:

*mysqldump -u nombre\_usuario -p nombre\_base\_de\_datos > backup.sql*

Para cargar la copia usando PowerShell usaremos:

*Get-Content backup.sql | mysql -u nombre\_usuario -p ejercicios*

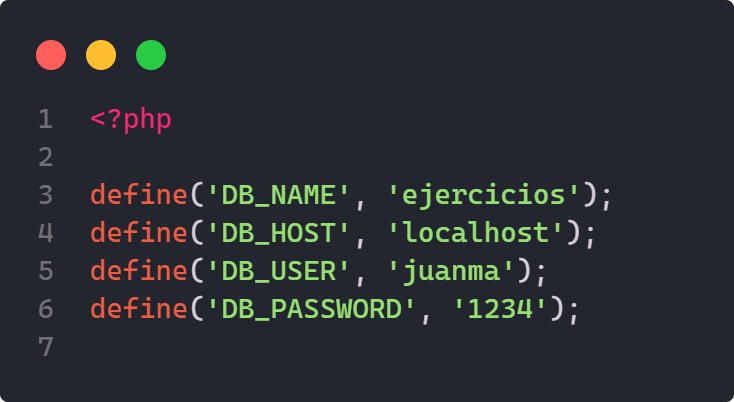
Para usar la base de datos que queramos, usaremos:

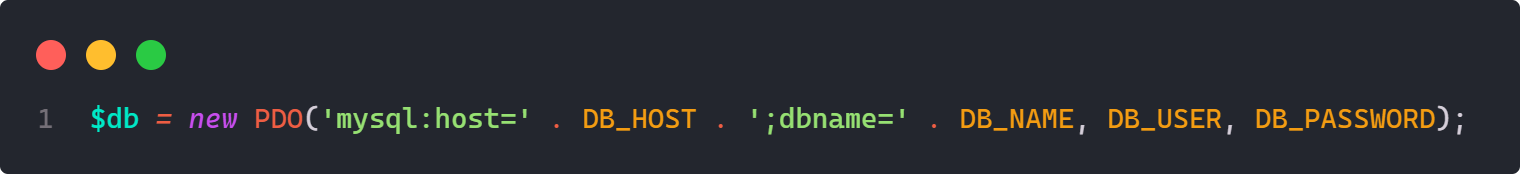
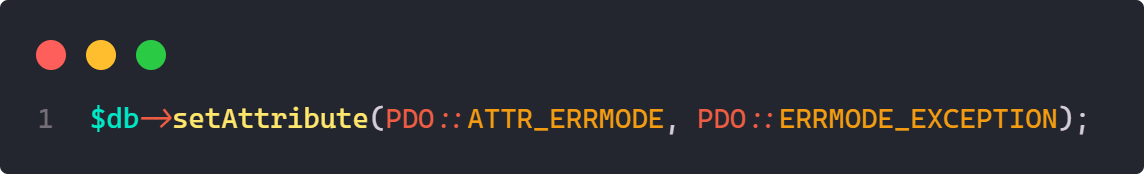
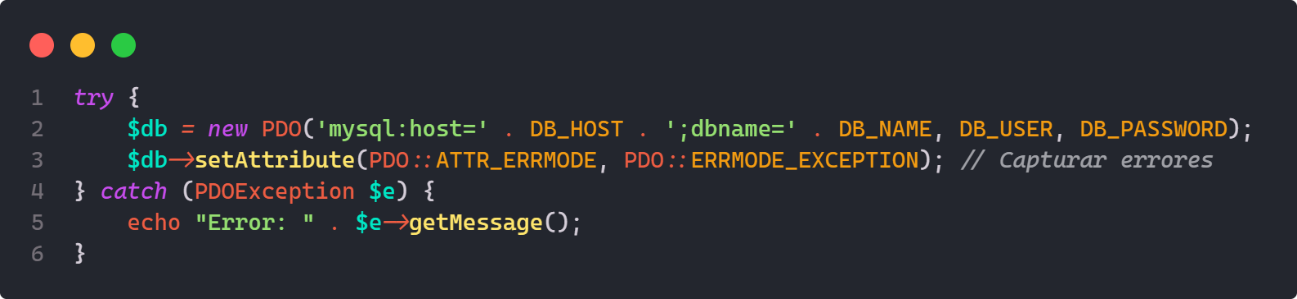
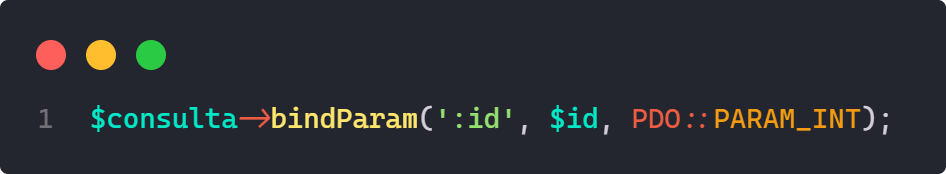
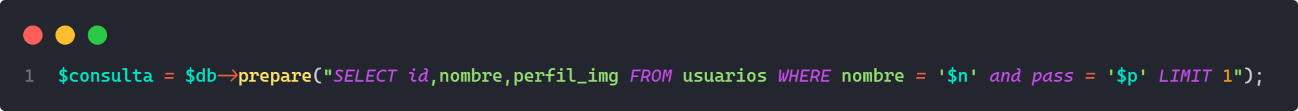
*USE NOMBRE\_DATABASE;*

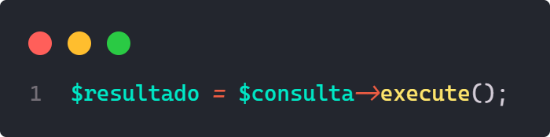
## -Conexión con PHP a MariaDB

Para conectarnos a la base de datos deberemos seguir los siguientes pasos:

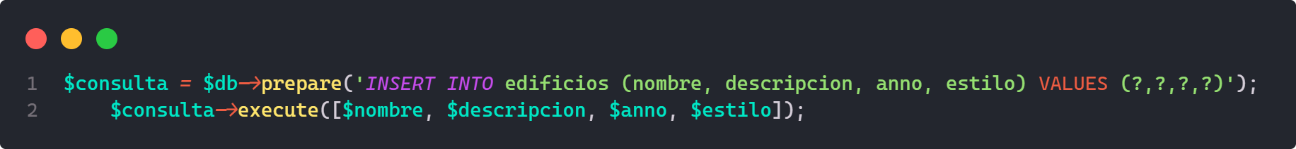
1. Tendremos un archivo data.php con los datos de la conexión.



1. Para conectarnos deberemos incluir el archivo anterior con *require\_once(‘data.php’)* o include(‘data.php’) usando el autoload.
2. Lo siguiente será crear un try-catch para manejar los errores que puedan llegar a surgir.
3. Dentro del try usaremos ***PDO*** para crear la conexión:
4. Podemos activar el manejo de errores mediante:
5. Ejemplo de como quedaría la creación de la conexión:
6. Para cerrar la conexión usaremos el nombre de la variable que contiene la conexión y la igualaremos a *null*.
7. Antes de ejecutar la consulta deberemos asignar a los parámetros (‘:id’) su valor para realizar la consulta mediante bindParam(‘:param’, $valor, tipo)
8. Para crear consultas debemos usar el método *prepare()* y guardaremos la consulta en una variable para más tarde poder ejecutarla con *execute()*.



Otra forma de hacer la consulta(usando un array con lo valores):



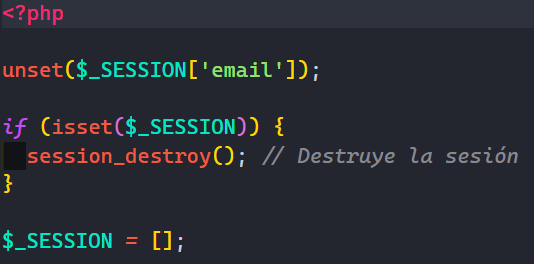
## -Sesiones

Cada vez que queramos iniciar una sesión debemos usar session\_start()

Para guardar información en la sesión usaremos $\_SESSION[‘info’], esto lo igualaremos a la info que deseamos guardar. Por ejemplo:



Para eliminar una sesión (ejemplo: logout), podremos usar una de las siguientes formas:



## -Cookies

Para crear una cookie debemos seguir el siguiente esquema:

setcookie(“{nombreCookie}”, {valorCookie}, tiempo de expiración)



Para acceder a los valores de las cookies, es igual que con las sesiones, deberemos usar $\_COOKIE[‘{valor}’]



Para eliminar la cookie le asignaremos un valor vacío y el tiempo de expiración será negativo (*time() - 3600*)

# DJANGO

## -Creación de Proyecto e Inicialización

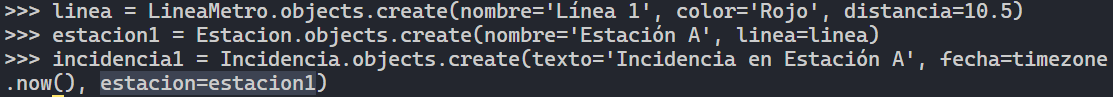
1. Creamos el virtualenv usando virtualenv venv
2. Instalamos Django con pip install django
3. Crearemos nuestro proyecto django usando django-admin startproject {nombreProyecto} . Añadimos el punto al final para que no nos cree dentro del proyecto otra carpeta más
4. Creamos la app usando django-admin startapp {nombreApp}
5. Añadimos la app al INSTALLED\_APPS del setting.py del proyecto
6. Creamos el archivo urls.py en la app
7. Añadimos al urls del proyecto las urls de la app
8. En el urls de la app añadimos el app\_name y el patterns

## -Shell

Para poder usar el shell\_plus se deberá instalar previamente django-extensions y añadirlo a APP\_INSTALED como ‘django\_extensions’

## -ORM

Para crear el objeto usamos el nombre del modelo + objects + create + (parámetros )

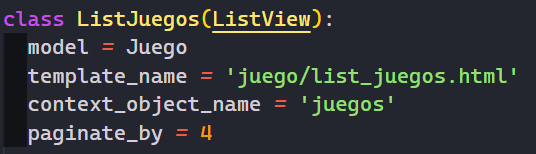


Para ver todos los objectos de un modelo usaremos ***{nombreDelModelo}.objects.all()***

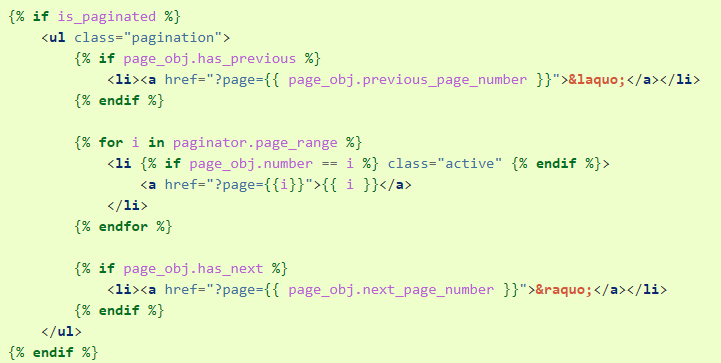
## -Paginación

Para realizar la paginación debemos seguir estos pasos:

1. En el views.py pondremos donde queramos hacer la paginación un paginate\_by = {numeroElementosPagina}

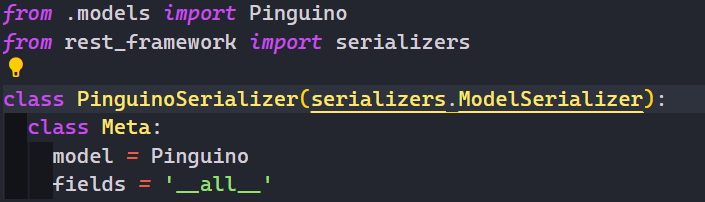


1. En el template ya recibimos los objetos necesarios y para crear la navegación entre páginas solo deberemos hacer lo siguiente:

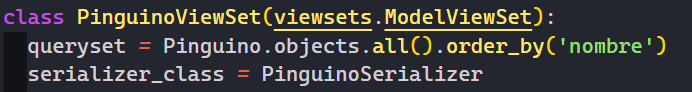


## -Django-Rest-Framework

1. Primero deberemos instalarlo mediante pip install djangorestframework
2. Deberemos tener creados los modelos y en base a estos crearemos los serializers en un archivo serializers.py e importaremos serializers de rest\_framework



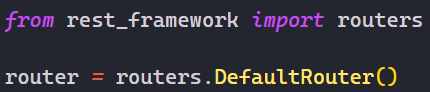
1. En nuestro archivo views.py crearemos una clase NombreViewSet, donde añadiremos el query\_set y el serializer\_class



Se le podrán añadir restricciones para que solo puedan acceder los usuarios que nosotros elijamos.



1. En el urls.py de nuestra app deberemos importar routers y crear uno y registrar la vista.





Añadimos al urlpatterns el path de la api con las urls del router.

