

```
require_once('settings.php');

# Inicia la sesión del usuario
function login() {
    $user_valido = validar_user_y_pass();
    if($user_valido) {
        $_SESSION['user'] = $user_valido;
        $_SESSION['date'] = date('Y-m-d H:i:s');
    }
    got();
}

# Valida el usuario y la contraseña
function validar_user_y_pass() {
    $user = $_POST['user'];
    $pass = $_POST['pass'];
    if($user == 'admin' & $pass == '12345') {
        return true;
    }
}

# Traer el POST data y retornar el hash MD5
function md5_data($data) {
    return md5($data);
}
```



|  |    |
|--|----|
| 1. Php básico .....  | 5  |
| 1.1. Introducción a PHP .....                                      | 5  |
| 1.1.1. Que es PHP .....  | 5  |
| 1.1.2. Requisitos para Desarrollar Aplicaciones con PHP.....       | 5  |
| 1.2. Introducción a php .....                                      | 20 |
| 1.2.1 Estructura básica de un script PHP .....                     | 20 |
| 1.3. Funciones de salida en PHP.....                               | 22 |
| 1.4. Variables y constantes .....                                  | 22 |
| 1.5. Tipos de datos.....   | 23 |
| 1.6. Números y operadores.....                                     | 24 |
| 1.7. Arreglos, Arreglos asociativos y funciones para arreglos..... | 27 |
| 1.8. Isset() y Empty().....  | 31 |
| 1.9. Estructuras de control.....                                   | 34 |
| 1.9.1. Estructuras condicionales.....                              | 34 |
| 1.9.1.1. if.....   | 34 |
| 1.8.1.2. Switch .....  | 35 |
| 1.10. Estructuras repetitivas .....                                | 37 |
| 1.11 Funciones definidas por el usuario .....                      | 42 |
| 1.11.1. Funciones en PHP.....                                      | 42 |
| 1.12.2. Funciones que no retornan valor.....                       | 42 |
| 1.12.3. Funciones que retornan valor.....                          | 43 |
| 1.13 include, require, include_once, require_once .....            | 44 |
| 1.14. Json_encode y json_decode .....                              | 47 |
| 2. PHP Intermedio .....  | 51 |
| 2.1. Programación Orientada a objetos.....                         | 51 |
| 2.1.1. Modificadores de acceso en PHP .....                        | 52 |
| 2.2. Clases .....  | 53 |
| 2.2.1. Métodos estáticos.....                                      | 54 |
| 2.3. Herencia .....  | 56 |
| 2.4. Clases Abstractas .....                                       | 58 |
| 2.5. Interfaces.....   | 60 |
| 2.6. Polimorfismo .....  | 62 |
| 2.7. Autoload.....   | 63 |
| 2.8. Namespaces en PHP.....  | 64 |
| 2.9. Composer .....  | 65 |
| 2.9.1. Autoload con composer .....                                 | 66 |

|   |    |
|---|----|
| 3. PHP Avanzado .....   | 67 |
| 3.1. Integración de php con Bases de datos relacionales (Mysql) ..... | 67 |
| 3.1.1. Bases de datos relacionales.....                               | 67 |
| 3.1.2. Entidades y Atributos .....                                    | 69 |
| 3.1.3. Identificadores Únicos.....                                    | 70 |
| 3.1.4. Relaciones.....  | 71 |
| 3.1.6. Normalización de bases de datos .....                          | 73 |
| 3.2. Mysql.....   | 75 |
| 3.2.1. Características MySQL.....                                     | 76 |
| 3.2.2. Consola de MySQL .....   | 76 |
| 3.2.3.1 Tipos de datos numéricos.....                                 | 80 |
| 3.2.3.2. Tipos de datos de carácter .....                             | 81 |
| 3.2.3.3. Tipos de dato fecha .....                                    | 81 |
| 3.2.3.4. Modificadores o Constraints.....                             | 81 |
| 3.3. SQL Structured Query Language.....                               | 82 |
| 3.3.1. Que se puede hacer con SQL.....                                | 82 |
| 3.3.2. Comandos DDL.....  | 83 |
| 3.3.2.1 SHOW DATABASE .....   | 83 |
| 3.3.2.2 CREATE DATABASE .....   | 83 |
| 3.3.2.3. DROP DATABASE .....  | 83 |
| 3.3.2.4. USE.....   | 84 |
| 3.3.2.5. CREATE TABLE .....   | 84 |
| 3.3.2.6 ALTER TABLE .....   | 85 |
| 3.3.2.7. Agregar columnas.....  | 85 |
| 3.3.2.8. Eliminar columnas .....                                      | 86 |
| 3.3.2.9. Renombrar una columna .....                                  | 86 |
| 3.3.2.10. Modificar el tipo de dato .....                             | 86 |
| 3.3.2.11 CONSTRAINTS.....   | 86 |
| 3.3.2.12. PRIMARY KEY.....  | 87 |
| 3.3.3. Comandos DML.....  | 92 |
| 3.3.3.1. INSERT INTO .....  | 92 |
| 3.3.3.2. Update .....   | 93 |
| 3.3.4.2. WHERE .....  | 95 |
| 3.3.4.3. Operadores Lógicos .....                                     | 96 |
| 3.3.4.4. Operadores de comparación .....                              | 97 |
| 3.3.4.5 Operador Like.....  | 98 |

|   |     |
|---|-----|
| 3.3.4.6. Between.....                                     | 98  |
| 3.4. Introducción a PDO .....                             | 99  |
| 3.4.1. Excepciones y opciones con PDO.....                | 99  |
| 3.4.2. Conexión a bases de datos .....                    | 100 |
| 3.4.3. Registro de datos .....                            | 101 |
| 3.5 Creación de una aplicación usando PHP+Mysql+PDO ..... | 102 |
| 3.6 Api Rest .....  | 110 |
| 3.6.1. Métodos API .....                                  | 110 |
| 3.6.2. Códigos de estado en Response HTTP .....           | 111 |
| 3.6.2.1. Respuestas informativas.....                     | 111 |
| 3.5.2.2 Respuestas satisfactorias.....                    | 111 |
| 3.5.2.3. Redirecciones.....                               | 113 |
| 3.5.2.4. Errores de cliente .....                         | 114 |
| 3.5.2.5. Errores de servidor .....                        | 116 |
| 3.6.3. Tipos de autenticación .....                       | 118 |

## 1. Php básico

### 1.1. Introducción a PHP

#### 1.1.1. Que es PHP

PHP es un lenguaje de código abierto que corre del lado del servidor y puede ser incrustado en documentos HTML. Una de las grandes ventajas que se tiene el programar con PHP es que posee una curva de aprendizaje muy rápido lo cual facilita el proceso de creación de aplicaciones orientadas al servidor. a pesar de que PHP es un programa orientado servidor permite su implementación en diferentes entornos de ejecución.

#### 1.1.2. Requisitos para Desarrollar Aplicaciones con PHP.

Para poder desarrollar los diferentes scripts en PHP se requiere contar con una aplicación que permita el soporte ejecución a través de un servidor Local. En la industria del desarrollo con PHP hoy en día existen múltiples herramientas que permiten la ejecución en un entorno controlado, a continuación, se listan algunos programas que facilitan el proceso al momento de montar y configurar un entorno de servidor.

### INTALACIÓN EN WINDOWS

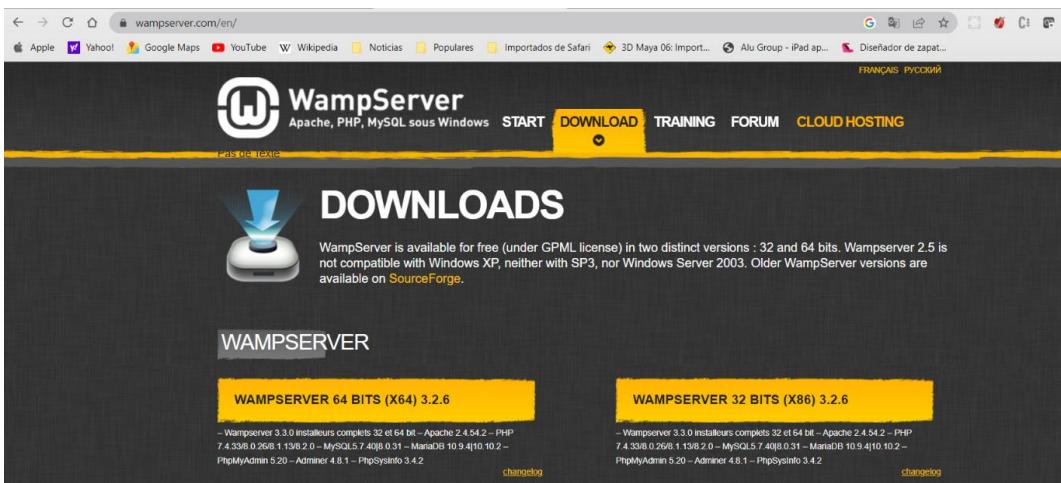
XAMPP Apache + MariaDB + PHP + Perl

The screenshot shows the Apache Friends website's download section for XAMPP. At the top, there are navigation links: Apache Friends, Descargar, Alojamiento, Comunidad, Acerca de, Buscar.., and ES. Below the header, the XAMPP logo (a stylized orange 'X' with a play button icon) is displayed next to the text "XAMPP Apache + MariaDB + PHP + Perl". To the left, there is a "¿Qué es XAMPP?" section with a brief description and a link to "Pulsa aquí para otras versiones". To the right, there is a large XAMPP logo graphic. Below these, there are four download links: "Descargar" (with a green arrow pointing to "XAMPP para Windows 8.2.0 (PHP 8.2.0)", which is highlighted with a red box), "XAMPP para Linux 8.2.0 (PHP 8.2.0)", and "XAMPP para OS X 8.2.0 (PHP 8.2.0)".

XAMPP es una de las herramientas que facilita el proceso de desarrollo de aplicaciones en PHP ya que es un proceso de instalación y configuración es muy intuitivo por medio de un asistente que es lo que hará paso a paso. XAMPP Es una distribución de Apache gratuita que posee todas las herramientas para base de datos como MaríaDB.

Url de descarga: <https://www.apachefriends.org/es/index.html>

**WampServer**



WampServer hola al igual que XAMPP es un entorno de desarrollo web que permite la creación de aplicaciones basadas en PHP ya que posee en su estructura interna un servidor Apache2, go soporte en scripting PHP y base de datos Mysql; WampServer posee un administrador gráfico de bases de datos llamado phpmyadmin el cual facilitará el proceso de creación y gestión de bases de datos en mysql.

Url de descarga: <https://www.wampserver.com/en/>

## Laragon

**Download**

Laragon is a universal development environment. It has many features to make you more productive:

**Benefits of Laragon**

After downloading, You can add **git**, **phpmyadmin**, **Node.js/MongoDB**, **Python/Django/Flask/Postgres**, **Ruby**, **Java**, **Go** using "**Tools > Quick add**"

Note: **You can also download from GitHub**

**Edition**

**Download Laragon - Full (173 MB)**

- **Laragon Full (64-bit):** Apache 2.4, Nginx, MySQL 8, PHP 8, Redis, Memcached, Node.js 18, npm, git

**Download Laragon - Portable (38 MB)**

- **Laragon Portable:** PHP 5.4, MySQL 5.1, bitman - Good for getting started with PHP, then you can add newer versions of PHP/MySQL easily later using "**Tools > Quick add**"

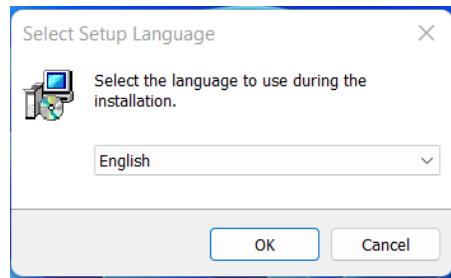
Laragon es un entorno de desarrollo universal para aplicaciones del lado del servidor utilizando Apache, PHP y mysql

Url descarga: <https://laragon.org/download/index.html>

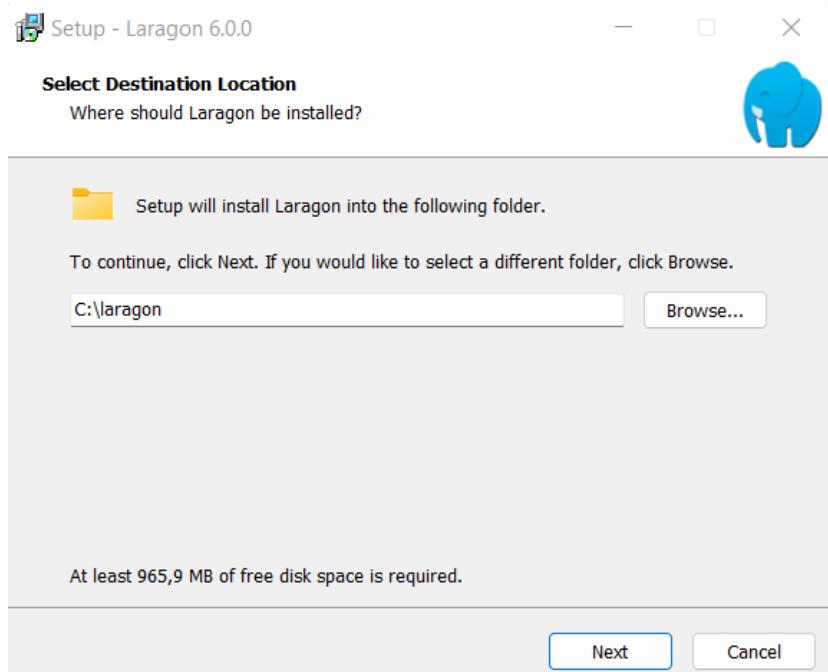
## Configuración e instalación de laragon

Para realizar el proceso de instalación del Laragon siga los siguientes pasos:

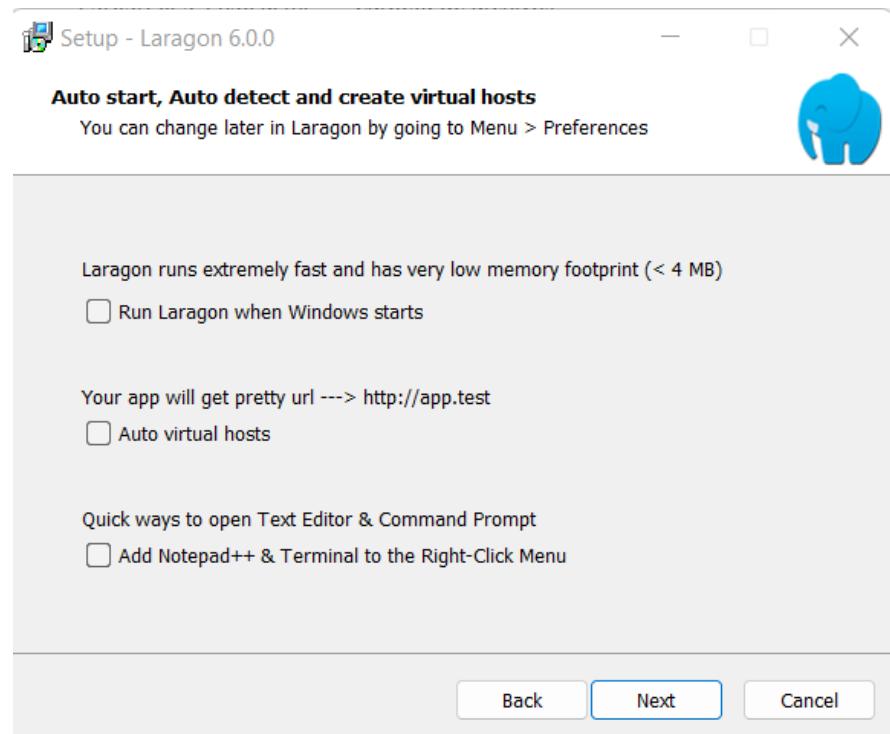
1. Descargue la versión del Laragon teniendo en cuenta el sistema operativo que tenga a disposición ya sea de 64 bits o de 32 bits.
2. Haga doble clic sobre el archivo que se ha descargado y siga los pasos teniendo en cuenta las indicaciones del asistente de instalación.



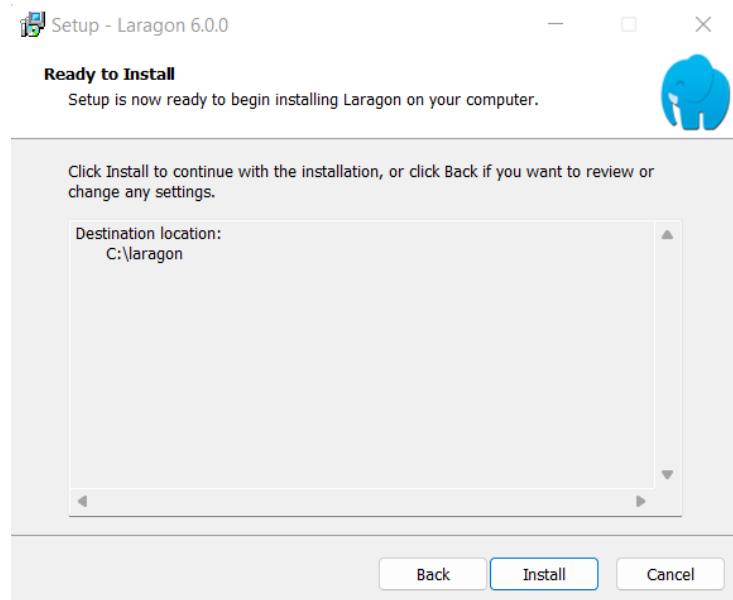
- a. **Selecione el idioma que desea configurar para la interfaz de usuario. Se recomienda conservar inglés por defecto.**
- b. Haga clic en el botón ok.
- c. Seleccione la carpeta de instalación y haga clic en el botón next.



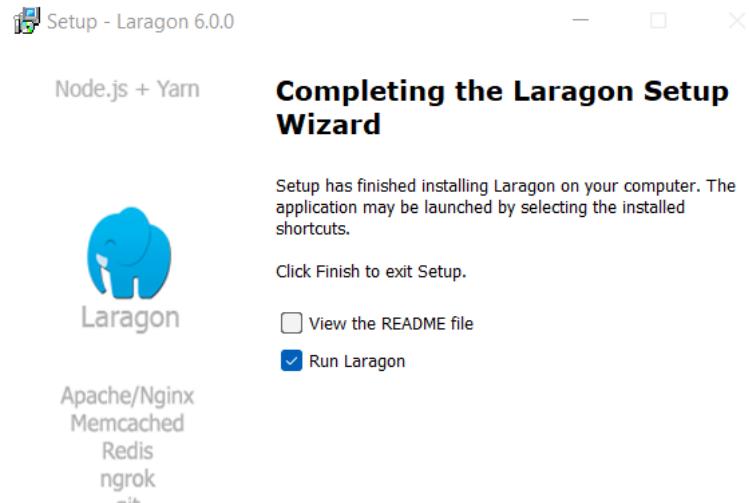
- d. En la siguiente ventana del asistente se debe especificar el comportamiento que va a tener la Laragon cuando se ejecute. En este apartado se especifica si se desean generar hosts virtuales, si se desea que Laragón arranque cuando el sistema operativo se reinicie y por último si se desea instalar el editor de note ++.



- e. Se recomienda desmarcar las opciones anteriores y a continuación hacer clic en el botón de next.



- f. en la ventana final se hace clic en el botón de instalar(Install).  
g. Y se espera que finalice el proceso de instalación.



- h. cuando finalice el proceso de instalación el asistente muestra una ventana que nos permitirá iniciar Laragon. si deseamos iniciar Laragón se deja activar la opción run Laragon.

Cuando se ejecuta el programa automáticamente nos aparece una ventana en la cual podremos observar una serie de opciones que nos permitirán personalizar el comportamiento de ejecución del servidor local.



A year from now you may wish you had started today.

Start All

Web

Database

Terminal

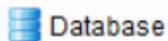
Root

Start All

esta opción permite iniciar todos los servicios del servidor local tanto el servidor web Apache 2.0 como el servidor de bases de datos MySQL.

Web

Esta opción permite abrir el servidor local en el navegador web por defecto.



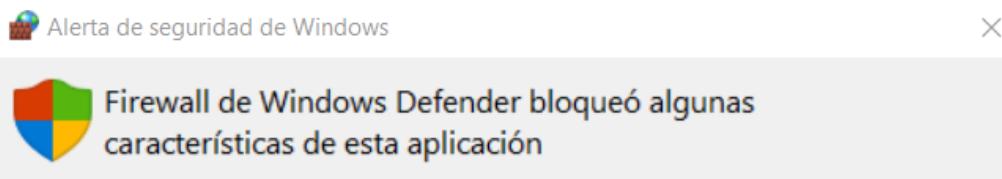
**Database** En esa opción se puede visualizar una herramienta de gestión y administración de mysql.



**Terminal** esta opción permite abrir la ventana de administración en consola.



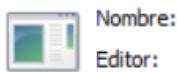
**Root** esta opción permite abrir la carpeta principal del servidor local. La carpeta principal el servidor se conoce con el nombre de documentroot.



Firewall de Windows Defender bloqueó algunas características de esta aplicación

Firewall de Windows Defender bloqueó algunas características de mysqld en todas las redes

públicas y privadas.



Nombre: \mysqld

Editor: Desconocido

Ruta de  
acceso:

C:\laragon\bin\mysql\mysql-8.0.30-winx64\bin\mysqld.exe

Permitir que mysqld se comunique en estas redes:

Redes privadas, como las domésticas o del trabajo

Redes públicas, como las de aeropuertos y cafeterías (no se recomienda porque  
estas redes públicas suelen tener poca seguridad o carecer de ella)

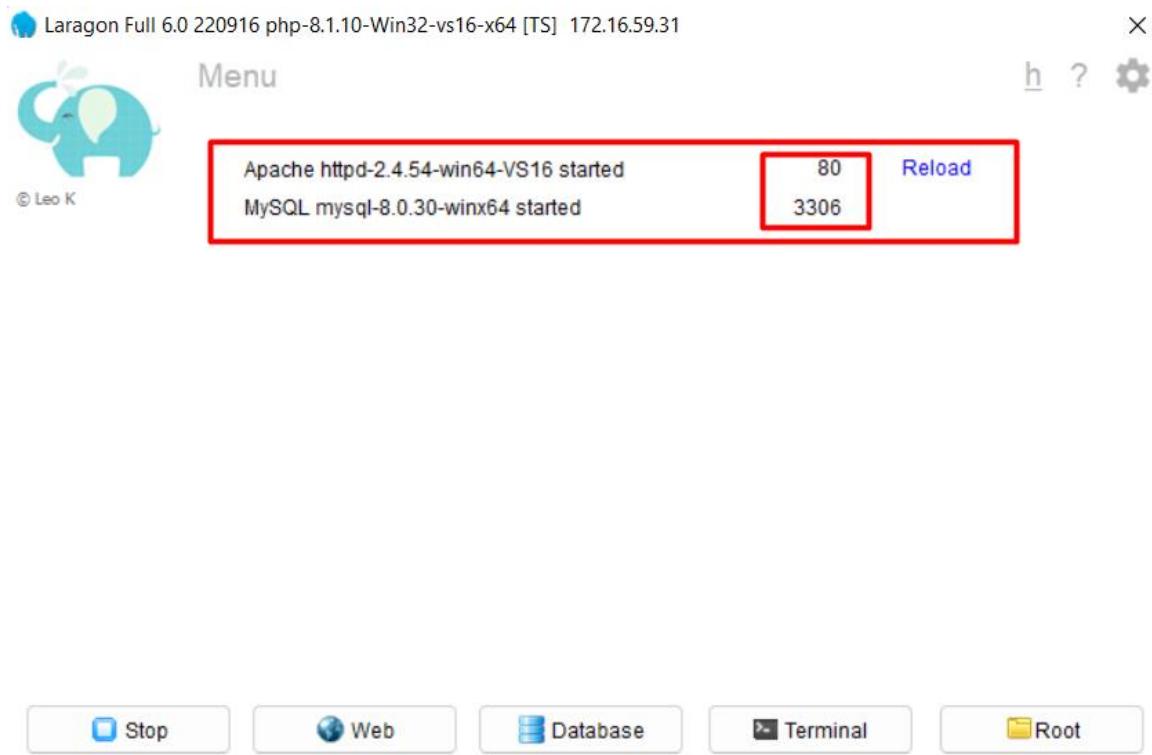
[¿Cuál es el riesgo de permitir que una aplicación pase a través de un firewall?](#)



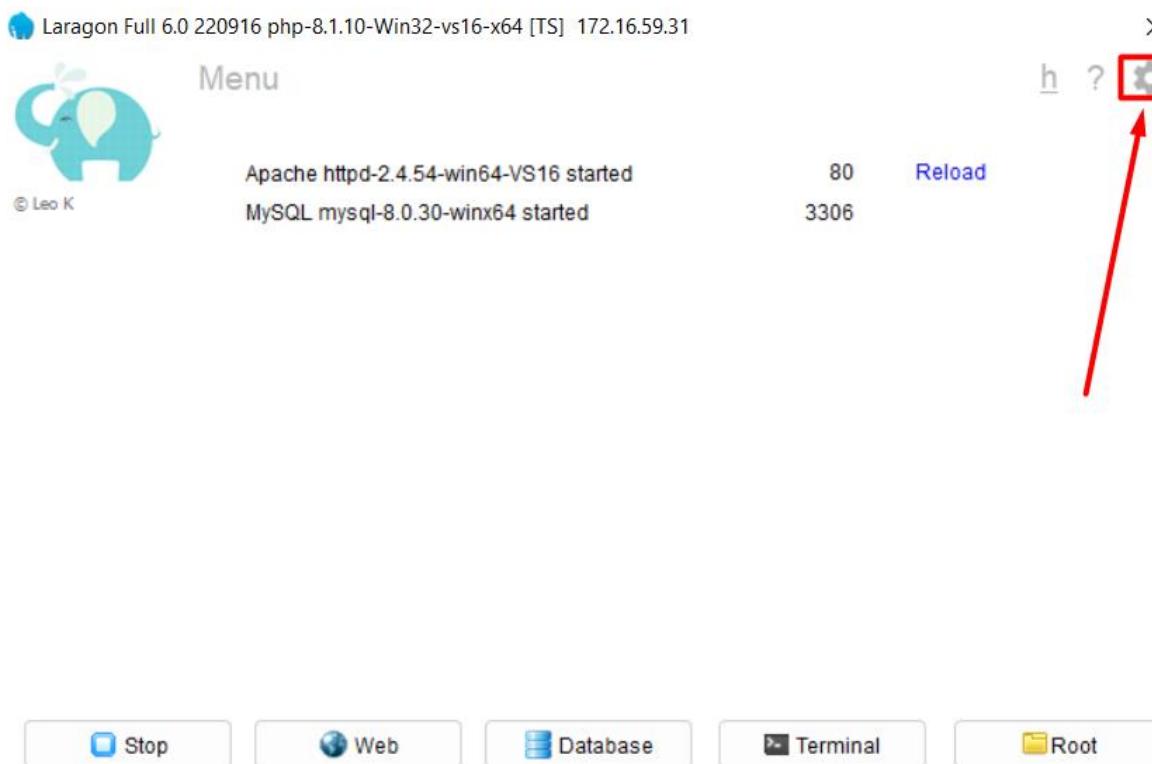
Permitir acceso

[Cancelar](#)

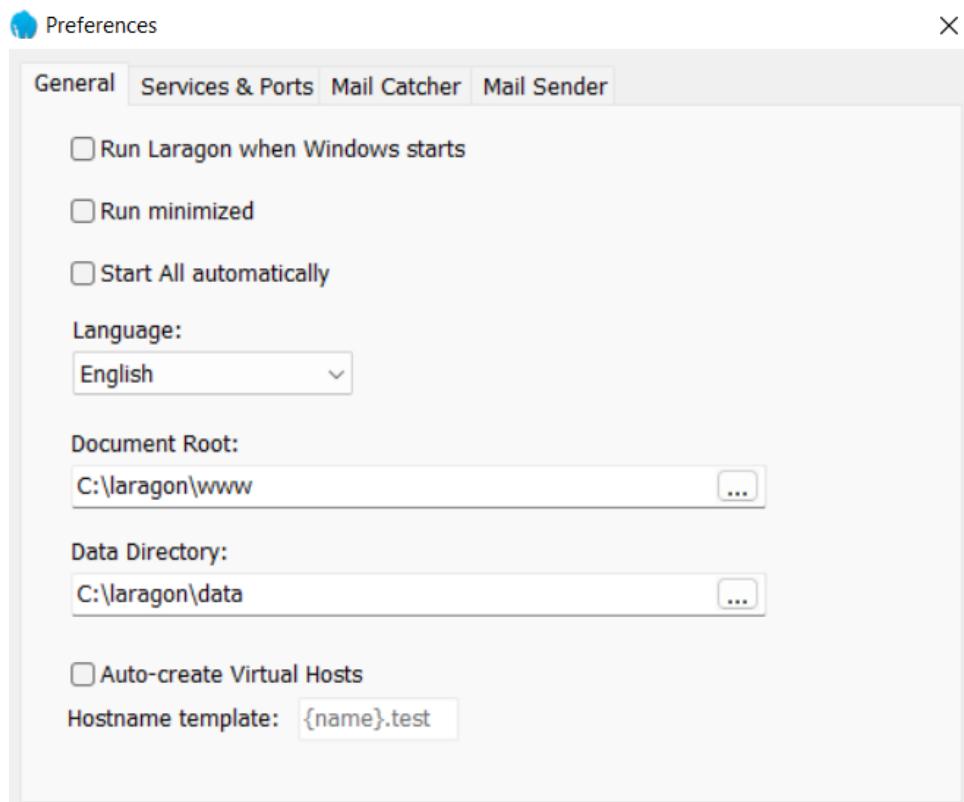
cuando se activan los permisos del firewall se puede observar que en la ventana del Laragón se muestran los servicios de Apache y MySQL.



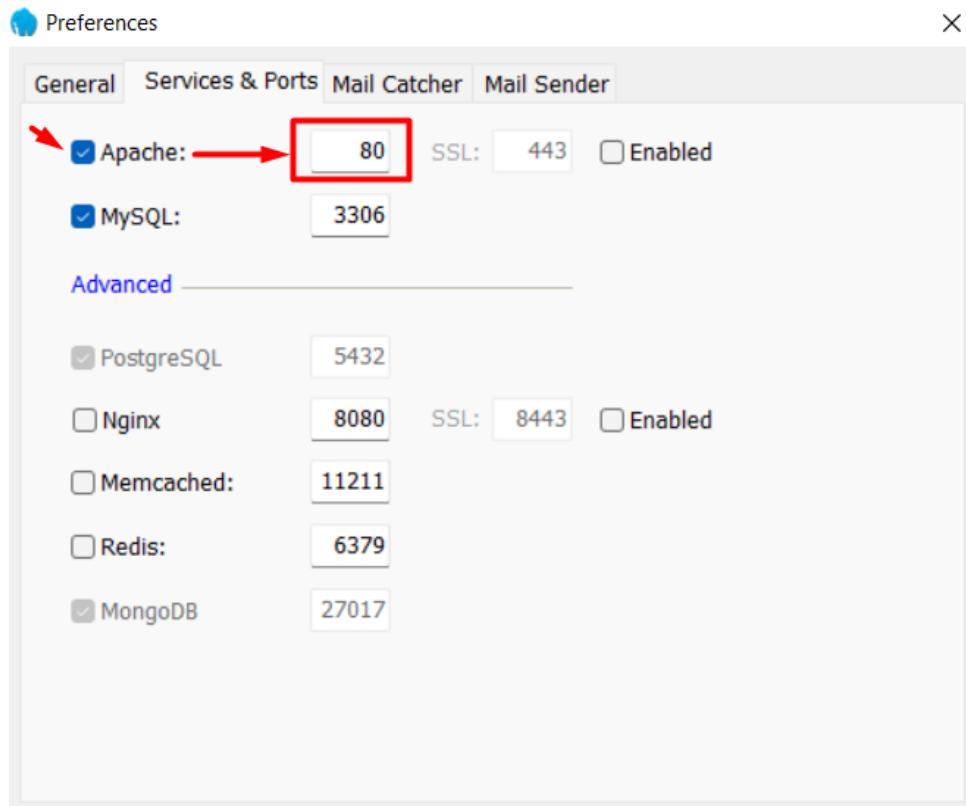
Cuando se trabaja con un servidor local por defecto el servidor se configura con el puerto 80 ya que es el puerto que tradicionalmente se utiliza en la web. En ocasiones pueden surgir problemas de conflicto con el puerto 80 ya que algunos programas en Windows pueden estar utilizando este puerto, en este caso existen dos posibilidades la primera es asignar un nuevo puerto al servidor local y esto lo podemos llevar a cabo haciendo uso del botón de configuración el cual podremos encontrar en la parte superior derecha de la ventana.



Cuando damos clic en esta opción nos aparece la siguiente ventana la cual nos va a permitir modificar la configuración original de los puertos que están debidamente configurados en el programa.



Para configurar el puerto se hace clic en la pestaña service and ports. Se recomienda los siguientes valores para el puerto de Apache: 81, 8081, 8080. Es importante tener en cuenta si se han hecho modificaciones del puerto por defecto al momento de llamar el servidor local desde el navegador se debe especificar el puerto asignado, por ejemplo: localhost: 8080



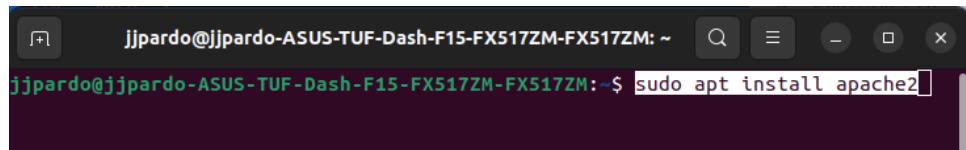
## Linux

En el siguiente apartado veremos cómo se realiza la instalación y configuración del servidor Apache y MySQL haciendo uso de las diferentes herramientas de instalación De Linux.

1. Abre la ventana terminal del sistema operativo. Recuerde que para acceder a la terminal debe presionar las teclas Ctrl+Alt+T. Es muy importante tener en cuenta que para realizar este proceso de instalación y configuración debemos contar con permisos de administrador.
2. Lo primero que debemos hacer es hacer una actualización del repositorio de paquetes del sistema operativo, para llevar a cabo ese proceso ingrese el comando **sudo apt update**.

```
jjpardo@jjpardo-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM:~$ sudo apt update
Hit:1 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://co.archive.ubuntu.com/ubuntu jammy InRelease
Get:4 http://download.opensuse.org/repositories/home:/jstaf/xUbuntu_22.04 InRelease [1.515 B]
Hit:5 http://co.archive.ubuntu.com/ubuntu jammy-updates InRelease
Err:4 http://download.opensuse.org/repositories/home:/jstaf/xUbuntu_22.04 InRelease
```

3. Para poder hacer la instalación de Apache después de haber actualizado el repositorio de paquetes del sistema operativo Linux se debe ingresar el siguiente comando: **sudo apt install apache2**.



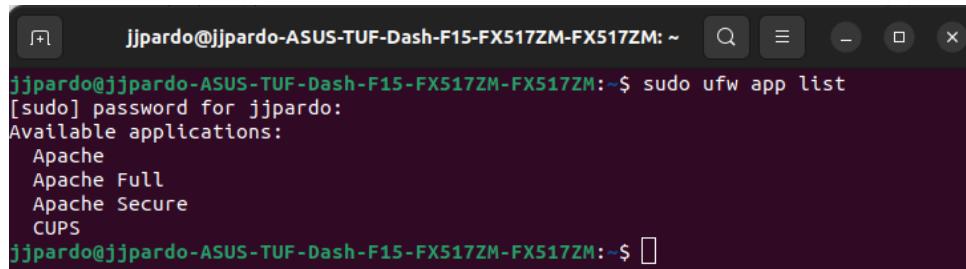
```
Jjpardo@jjpardo-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM: ~
jjpardo@jjpardo-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM:~$ sudo apt install apache2
```

durante el proceso de instalación el asistente solicitará una serie de confirmaciones para poder continuar con la instalación. Se recomienda que a todas las solicitudes que realice el asistente se responda de forma afirmativa cómo se puede visualizar en las siguientes imágenes:

```
apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
libaprutil1-dbd-sqlite3 libaprutil1-ldap
0 upgraded, 8 newly installed, 0 to remove and 31 not upgraded.
Need to get 1.918 kB of archives.
After this operation, 7.706 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

es muy importante tener en cuenta aplicar las mayúsculas de acuerdo a la respuesta

4. Cuando finalice la instalación de Apache se debe configurar el firewall de Linux para poder tener acceso al puerto 80. Para realizar la instalación y configuración del firewall vamos a utilizar la aplicación UFW a continuación se describen los comandos necesarios para llevar a cabo la configuración del firewall.
  - `sudo ufw app list` este comando se utiliza para visualizar las aplicaciones que se encuentran disponibles para ser configuradas.

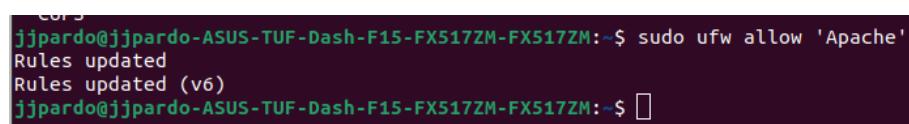


```
Jjpardo@jjpardo-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM: ~
[jjpardo] password for jjpardo:
Available applications:
 Apache
 Apache Full
 Apache Secure
 CUPS
jjpardo@jjpardo-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM:~$
```

El perfil de Apache permite definir únicamente el puerto 80

el perfil de Apache full permite definir el puerto 80 y el puerto 443 el cual es Utilizado Por la capa segura SSL.

- `sudo ufw allow 'Apache'` ese comando permite especificar una regla en el firewall y así poder utilizar el puerto 80 sin ningún problema.



```
Rules updated
Rules updated (v6)
jjpardo@jjpardo-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM:~$
```

- `sudo ufw status` este comando permite verificar el estado de firewall en el caso que el firewall se encuentre inactivo se puede ingresar el comando `sudo ufw enable`. Por defecto el comando `sudo ufw status` permite visualizar el siguiente resultado:

```
jjpardo@jjpardo-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM:~$ sudo ufw status
Status: active

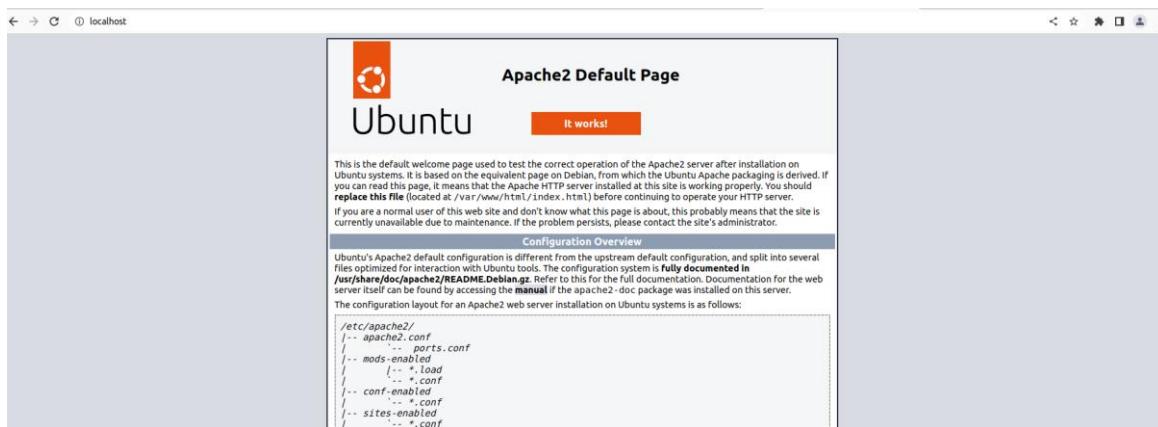
To                         Action      From
--                         --         --
Apache                      ALLOW       Anywhere
Apache (v6)                  ALLOW       Anywhere (v6)
```

- `sudo systemctl status apache2` este comando permite verificar el estado del servidor web Apache. los ejecuta el comando se obtiene el siguiente resultado:

```
jjpardo@jjpardo-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2023-04-06 09:47:13 -05; 5h 26min ago
     Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 1337 (apache2)
      Tasks: 55 (limit: 18652)
    Memory: 7.9M
      CPU: 117ms
     CGroup: /system.slice/apache2.service
             ├─1337 /usr/sbin/apache2 -k start
             ├─1338 /usr/sbin/apache2 -k start
             └─1339 /usr/sbin/apache2 -k start

abr 06 09:47:13 jjpardo-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM systemd[1]: Starting >
abr 06 09:47:13 jjpardo-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM apachectl[1265]: AH00074: >
abr 06 09:47:13 jjpardo-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM systemd[1]: Started T...
lines 1-16/16 (END)
```

para finalizar la comprobación si la instalación del servidor web Apache quedó correctamente se puede acceder a la dirección IP del equipo haciendo uso del navegador web o simplemente ingresando la palabra localhost de igual manera en el navegador web. a continuación, se visualiza una imagen de ejemplo.



A continuación, vamos a estudiar unos cuántos comandos que se utilizan en Linux para llevar a cabo algunas tareas sobre el servidor web.

**sudo systemctl stop apache2** : Este comando permite de tener el servidor web cuando se está ejecutando.

**sudo systemctl start apache2**: este comando permite arrancar el servidor web luego de haberse detenido ya sea haciendo uso del comando stop o por algún problema de configuración.

**sudo systemctl restart apache2**: este comando permite detener e iniciar el servidor sin necesidad de ejecutar los dos comandos anteriores.

**sudo systemctl reload apache2**: este comando permite recargar los cambios o actualizaciones realizadas en la configuración de Apache.

**sudo systemctl disable apache2**: este comando permite deshabilitar el inicio del servidor web cuando se reinicie el servidor.

**sudo systemctl enable apache2**: este comando permite habilitar el inicio automático del servidor web cuando se reinicia el servidor.

Cuando se inicia con el trabajo hoy en un servidor web se recomienda que se utilice hosts virtuales, en el servidor por defecto al momento de instalar Apache se va a habilitar un host virtual el cual almacenará todos los proyectos que usted cree durante su proceso de desarrollo. Por defecto esta carpeta se denomina document root y está ubicada en la siguiente carpeta `/var/www/html`.

Para poder crear un host virtual lo primero que se debe realizar es crear una carpeta dentro de la carpeta www qué se encuentra ubicada en la carpeta var. Puede acceder al siguiente video en el cual se detalla el proceso que se debe llevar a cabo para la configuración del Host.(CrearHostVirtualApache.mp4).

### instalación de PHP en Linux

para el correcto funcionamiento programación utilizando el lenguaje PHP es necesario realizar el proceso de instalación y configuración en el sistema operativo Linux haciendo uso de la terminal es importante tener en cuenta que para realizar este proceso es necesario contar con permisos de administración.

- `sudo apt update`
- `sudo apt upgrade`
- `sudo apt install php libapache2-mod-php php-mysql`
- `php -v`

```
[sudo] password for jjpardo: 
jjpardo@jjpardo-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM:~$ php -v
PHP 8.1.2-1ubuntu2.11 (cli) (built: Feb 22 2023 22:56:18) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.1.2, Copyright (c) Zend Technologies
    with Zend OPcache v8.1.2-1ubuntu2.11, Copyright (c), by Zend Technologies
jjpardo@jjpardo-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM:~$ 
```

- `sudo nano /etc/apache2/mods-enabled/dir.conf`

```
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm
</IfModule>
```

| PHP Version 8.1.2-1ubuntu2.11                  |  |
|--|--|
| <b>System</b>                                  | Linux jjpardo-ASUS-TUF-Dash-F15-FX5172M-FX5172M 5.19.0-38-generic #39--22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 17 21:16:15 UTC 2023 x86_64  |
| <b>Build Date</b>                              | Feb 22 2023 22:56:18   |
| <b>Build System</b>                            | Linux  |
| <b>Server API</b>                              | Apache 2.0 Handler   |
| <b>Virtual Directory Support</b>               | disabled   |
| <b>Configuration File (php.ini) Path</b>       | /etc/php/8.1/apache2   |
| <b>Loaded Configuration File</b>               | /etc/php/8.1/apache2/php.ini   |
| <b>Scan this dir for additional .ini files</b> | /etc/php/8.1/apache2/conf.d  |
| <b>Additional .ini files parsed</b>            | /etc/php/8.1/apache2/conf.d/10-mysqli.ini, /etc/php/8.1/apache2/conf.d/10-opcache.ini, /etc/php/8.1/apache2/conf.d/10-pdo.ini, /etc/php/8.1/apache2/conf.d/20-calendar.ini, /etc/php/8.1/apache2/conf.d/20-ctype.ini, /etc/php/8.1/apache2/conf.d/20-exif.ini, /etc/php/8.1/apache2/conf.d/20-fil.ini, /etc/php/8.1/apache2/conf.d/20-finfo.ini, /etc/php/8.1/apache2/conf.d/20-ftp.ini, /etc/php/8.1/apache2/conf.d/20-gettext.ini, /etc/php/8.1/apache2/conf.d/20-iconv.ini, /etc/php/8.1/apache2/conf.d/20-mysqli.ini, /etc/php/8.1/apache2/conf.d/20-pdo_mysql.ini, /etc/php/8.1/apache2/conf.d/20-readline.ini, /etc/php/8.1/apache2/conf.d/20-shmop.ini, /etc/php/8.1/apache2/conf.d/20-sockets.ini, /etc/php/8.1/apache2/conf.d/20-sysvmsg.ini, /etc/php/8.1/apache2/conf.d/20-system.ini, /etc/php/8.1/apache2/conf.d/20-tokenizer.ini |
| <b>PHP API</b>                                 | 20210902   |
| <b>PHP Extension</b>                           | 20210902   |
| <b>Zend Extension</b>                          | 420210902  |
| <b>Zend Extension Build</b>                    | API420210902.NTS   |
| <b>PHP Extension Build</b>                     | API20210902.NTS  |
| <b>Debug Build</b>                             | no   |
| <b>Thread Safety</b>                           | disabled   |
| <b>Zend Signal Handling</b>                    | enabled  |
| <b>Zend Memory Manager</b>                     | enabled  |
| <b>Zend Multibyte Support</b>                  | disabled   |
| <b>IPv6 Support</b>                            | enabled  |
| <b>DTrace Support</b>                          | available, disabled  |

## Configuración HostVirtual

En el siguiente capítulo vamos a estudiar la forma de configurar un virtual host en PHP para así poder trabajar de una forma más cómoda y eficiente a la hora de desarrollar bajo un entorno de servidor.

- Crear la carpeta que va a ser utilizada como repositorio de todos los proyectos que vamos a desarrollar. Recuerde que el comando para crear carpetas desde la consola es mkdir.
- en este segundo paso se va a cambiar de propietarios a la carpeta que se definió anteriormente para el caso práctico la carpeta se llamó campuslands. el comando utilizado para cambiar el propietario de una carpeta en Linux es chown. es importante recordar que este comando se debe ejecutar con permisos de super usuario es decir se debe anteponer la palabra sudo. En la siguiente imagen podrá observar el uso de dichos comandos.

```
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:/var/www$ sudo chown -R trainertest:trainertest campuslands/
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:/var/www$ ls -l
total 8
drwxr-xr-x 2 trainertest trainertest 4096 abr 21 15:33 campuslands
drwxr-xr-x 2 root         root        4096 abr 21 15:18 html
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:/var/www$
```

- El siguiente paso es otorgar permisos de tipo 755 a la carpeta que vamos a utilizar en la configuración del host virtual.
- A continuación, creamos El archivo index.html el cual será el archivo principal de nuestro host virtual.
- El siguiente paso es ingresar a la carpeta de sitios disponibles. Y crear un nuevo archivo de configuración a partir del archivo de sites por defecto que existe en la carpeta. Ver imágenes a continuación.

```
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:/var/www/campuslands$ cd /
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:$ ls
access.log  boot  dev   etc   html  lib32  libx32   media  opt   root  sbin  srv  tmp  var
bin        cdrom error.log home  lib   lib64  lost+found  mnt   proc  run   snap  sys  usr
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:$ cd etc
```

```
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:/etc$ cd apache2
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:/etc/apache2$ ls
conf-available  conf-enabled  magic      mods-available  sites-available
envvars          mods-available ports.conf  sites-enabled
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:/etc/apache2$ cd sites-available/
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:/etc/apache2/sites-available$
```

```
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:/etc/apache2$ cd sites-available
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:/etc/apache2/sites-available$ ls
000-default.conf  default-ssl.conf
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:/etc/apache2/sites-available$
```

```
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:/etc/apache2/sites-available$ sudo cp 000-default.conf campusland.conf
[sudo] password for trainertest:
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:/etc/apache2/sites-available$ ls -l
total 16
-rw-r--r-- 1 root root 1332 sep  7 2022 000-default.conf
-rw-r--r-- 1 root root 1332 abr 21 18:17 campusland.conf
-rw-r--r-- 1 root root 6338 sep 29 2022 default-ssl.conf
trainerexpert@trainertest-HP-Laptop-15-ef2xxx:/etc/apache2/sites-available$
```

- Ingresar a el archivo campusland y configurar los parámetros de acuerdo a las siguientes imágenes.

```
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
#ServerName www.example.com

ServerAdmin webmaster@localhost
DocumentRoot /var/www/campusland
```

- Especificar el nombre del servidor

```
<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    ServerName campusland.buc.co
```



```
trainereexpert@trainereexpert-HP-Laptop-15-ef2xxx:/etc/apache2/sites-available$ cd ..
trainereexpert@trainereexpert-HP-Laptop-15-ef2xxx:/etc/apache2$ sudo a2ensite campusland.conf
[sudo] password for trainereexpert:
Enabling site campusland.
To activate the new configuration, you need to run:
    systemctl reload apache2
trainereexpert@trainereexpert-HP-Laptop-15-ef2xxx:/etc/apache2$
```

```
trainereexpert@trainereexpert-HP-Laptop-15-ef2xxx:/etc/apache2$ systemctl reload apache2.service
trainereexpert@trainereexpert-HP-Laptop-15-ef2xxx:/etc/apache2$ systemctl restart apache2.service
trainereexpert@trainereexpert-HP-Laptop-15-ef2xxx:/etc/apache2$
```

```
GNU nano 6.2
/etc/hosts *
127.0.0.1      localhost
127.0.1.1      trainereexpert-HP-Laptop-15-ef2xxx
::0      campusland.buc.co
# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
q
y
```



## Página principal host virtual

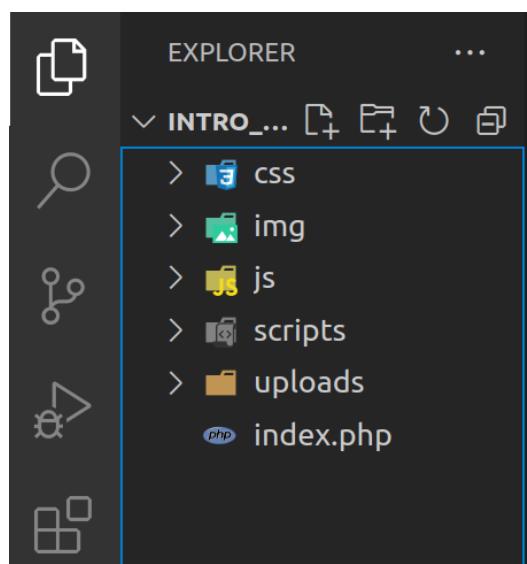
## 1.2. Introducción a php

En el siguiente capítulo se estudiarán los diferentes aspectos del lenguaje de programación PHP, se abordarán tópicos como variables, constantes, estructuras de control estructuras repetitivas, funciones, programación orientada a objetos en conexión con bases de datos.

### 1.2.1 Estructura básica de un script PHP

Al igual que en HTML en el lenguaje de servidor PHP se deben definir etiquetas quién dicen el inicio y el final de un script. Antes de iniciar la codificación vamos a entender cuál es la estructura básica de un proyecto que soporte desarrollo de scripting con PHP. La siguiente imagen se podrá visualizar una estructura básica recomendada para mantener el orden de los proyectos web usando PHP.

Es importante tener en cuenta que los proyectos que utilicen la tecnología PHP requieren ser almacenados en el documentRoot del servidor web, el documentRoot es la carpeta principal del servidor Apache donde se deben alojar cada uno de los proyectos que se han creado. En el siguiente video podrá observar dónde se ubica el document root. ([DocumentRoot.mp4](#)).



Estructura Basica Proyecto en Web/PHP

Muy bien ahora que conocemos y entendemos cómo estructurar un proyecto web en PHP vamos a analizar la estructura básica de un Script.

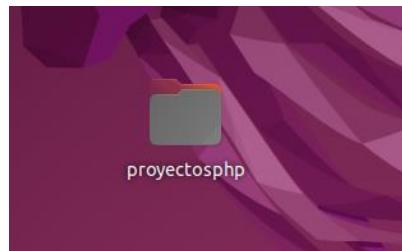
```
index.php
1 <?php
2 //Aqui va todo el codigo en PHP
3 ?>
```

Cómo podemos observar en el gráfico anterior tenemos lo siguiente:

en la línea número 1 tenemos la etiqueta de apertura del script en PHP, en la línea número dos que es el cuerpo del espíritu contendrá toda la lógica y programación que necesitemos incorporar en el script y en la línea número 3 se muestra una etiqueta de cierre del Script en PHP.

En la actualidad algunos desarrolladores omiten la etiqueta de cierre la cual se puede observar en la línea de código número 3, esto no es una práctica recomendada ya que puede generar confusión en la terminación del script e incluso llevar a errores al momento de ser ejecutado.

Cuando desarrollamos con PHP. Normalmente. Los archivos y proyectos se alojan. En la carpeta Principal. PHP. Da la flexibilidad que podamos ejecutar. Los archivos nativos de PHP. Haciendo uso. De Comando Php -S. A continuación. Se hace una breve demostración. Haciendo uso de este comando. El cual podrá ser utilizado. Al momento de desarrollo. Para comprobar. El funcionamiento de este comando. Vamos a crear una carpeta. En el escritorio. La cual llamaremos proyectos PHP.



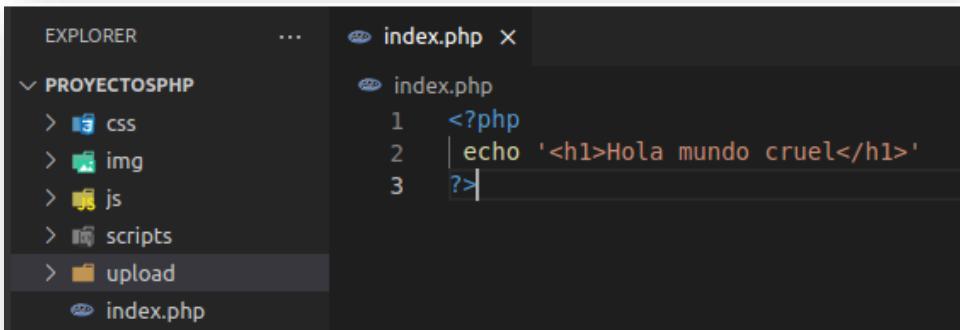
Posteriormente Crearemos un archivo de prueba. Para verificar. Que se ejecute correctamente. Abriremos la carpeta en el editor visual studio code pone el editor de su preferencia.

Para ejecutar el servidor temporal con PHP ingresamos el siguiente comando:

```
trainexpert@trainexpert-HP-Laptop-15-ef2xxx:~/Desktop/proyectosphp$ php -S localhost:3000
```

para verificar el funcionamiento se debe abrir un navegador web de su preferencia ingresar la siguiente instrucción en la barra de direcciones URL: localhost:3000 es importante tener en cuenta que este comando debe tener especificado el puerto que se le indicó al momento de ejecutarlo.

Para comprobar el funcionamiento y carga se crea un archivo en este caso lo llamaremos index.php ya que todo proyecto web debe tener un archivo principal. En la siguiente imagen se puede visualizar el proyecto en visual studio code con el archivo index.



```
index.php
<?php
echo '<h1>Hola mundo cruel</h1>';
?>
```

En la línea número uno podemos observar la etiqueta de apertura PHP, en la línea número 2 podremos observar la función echo la cual es utilizada en PHP para renderizar elementos en el documento web.

Antes de comenzar el estudio con PHP es muy importante tener en cuenta una estructura o base donde vamos a almacenar los proyectos.

### 1.3. Funciones de salida en PHP

En lenguajes de programación las funciones de salida son aquellos que permiten mostrar información al usuario cuando finaliza la ejecución de un proceso determinado o cuando finaliza el proceso una petición realizada por el usuario final.

En PHP existen varias funciones de impresión que permiten mostrar información en la salida estándar del servidor web. Las funciones más comunes son:

- echo(): Esta es la función más común para imprimir en PHP. Se utiliza para mostrar una o varias cadenas de texto en la salida del servidor web. La sintaxis básica es la siguiente:

```
echo "Texto a imprimir";
```

- print(): Esta función es similar a echo(), pero solo puede imprimir una cadena de texto a la vez. La sintaxis básica es la siguiente:

```
$texto = "Mundo";
printf("Hola %s", $texto);
```

En este ejemplo, el valor de la variable \$texto se incrusta en la cadena usando el marcador %s, que indica que se trata de una cadena de texto.

- sprintf(): Esta función es similar a printf(), pero en lugar de imprimir la cadena formateada en la salida estándar, devuelve la cadena formateada como resultado. La sintaxis básica es la siguiente:

```
$texto = "Mundo";
$mensaje = sprintf("Hola %s", $texto);
echo $mensaje;
```

### 1.4. Variables y constantes

Las variables en PHP se definen usando el símbolo \$ seguido del nombre de la variable. La asignación de un valor a la variable se realiza utilizando el operador =. A continuación, se muestran algunos ejemplos de cómo se declaran variables en PHP:

```
index.php
1  <?php
2  // Declarar una variable numérica
3  $edad = 25;
4
5  // Declarar una variable de texto
6  $nombre = "Juan";
7
8  // Declarar una variable booleana
9  $es_valido = true;
10 echo var_dump($nombre);
11 ?>
```

Como se puede observar en la imagen anterior que la línea número 10 se está imprimiendo el tipo de dato y el contenido de la variable haciendo uso de la función especial var\_dump.

las constantes PHP a igual que en cualquier lenguaje de programación son espacios reservados en memoria que no cambian durante la ejecución de un programa. En la siguiente imagen se puede visualizar ejemplos de definición de constantes en php.

```
// Declarar una constante numérica
define("PI", 3.1416);

// Declarar una constante de texto
define("SALUDO", "Hola Mundo!");

// Declarar una constante booleana
define("ES_VALIDO", true);
```

## 1.5. Tipos de datos

En PHP, existen varios tipos de datos que se pueden utilizar para almacenar diferentes tipos de información. Los tipos de datos más comunes son los siguientes:

- **Enteros (int)**: se utilizan para almacenar números enteros sin decimales.
- **Punto flotante (float)**: se utilizan para almacenar números con decimales.
- **Cadenas de texto (string)**: se utilizan para almacenar texto y caracteres.
- **Booleanos (bool)**: se utilizan para almacenar valores de verdad o falsedad, que se representan por true o false.
- **Arreglos (array)**: se utilizan para almacenar una colección de datos, que pueden ser de diferentes tipos.
- **Objetos (object)**: se utilizan para almacenar instancias de clases, que son definiciones de objetos.

- Recursos (resource): se utilizan para almacenar referencias a recursos externos, como conexiones a bases de datos o archivos abiertos.
- Nulos (null): se utilizan para representar una variable sin valor o sin definir.

Además de estos tipos de datos básicos, PHP también admite otros tipos de datos, como las constantes, que se mencionaron anteriormente, y los tipos de datos compuestos, como las estructuras de datos y las clases.

En resumen, en PHP existen varios tipos de datos que se pueden utilizar para almacenar diferentes tipos de información, como enteros, cadenas de texto, booleanos, arreglos, objetos y recursos. Es importante elegir el tipo de dato adecuado para cada situación, según el tipo de información que se desee almacenar y manipular.

En nuevo la siguiente imagen se podrá observar cómo se define y asignar las variables de acuerdo a un tipo de dato.

```

1  <?php
2
3  // Boolean
4  $logueado = true;
5  var_dump($logueado);
6
7  // Enteros
8  $numero = 200;
9  var_dump($numero);
10
11 // Floats
12 $float = 200.5;
13 var_dump($float);
14
15 // Strings
16 $nombre = "Juan";
17 var_dump($nombre);
18
19 $array = [];
20 var_dump($array);
21
22 ?>
```

## 1.6. Números y operadores

Al igual que en todos los lenguajes de programación PHP posee una serie de operadores lógicos, de comparación que nos permiten realizar operaciones básicas y complejas a continuación podrá observar un breve ejemplo en el cual se muestra cómo se aplica algunos de dichos operadores. 7. Estructuras en php.

|  |  |
|--|--|
| <p><b>1.Arithmetic Operator</b></p> <pre>+ = Addition - = Subtraction * = Multiplication / = Division % = Modulo ** = Exponentiation</pre>   | <p><b>2.Assignment Operator</b></p> <pre>= "equal to"</pre> <p><b>3.Array Operator</b></p> <pre>+ = Union == = Equality ==== = Identity != = Inequality &lt;&gt; = Inequality !== = Non-identity</pre>   |
| <p><b>4.Bitwise Operator</b></p> <pre>&amp; = and ^ = xor   = not &lt;&lt; = shift left &gt;&gt; = shift right</pre>   | <p><b>5.Comparison Operator</b></p> <pre>== = equal === = identical != = not equal !== = not identical &lt;&gt; = not equal &lt; = less than &lt;= less than or equal &gt; = greater than &gt;= = greater than or equal &lt;=&gt; = spaceship operator</pre> |
| <p><b>6.Execution Operator</b></p> <pre>`` = backticks</pre> <p><b>7.Error Control Operator</b></p> <pre>@ = at sign</pre> <p><b>8.Incrementing/Decrementing Operator</b></p> <pre>++\$a = PreIncrement \$a++ = PostIncrement --\$a = PreDecrement \$a-- = Postdecrement</pre> | <p><b>9.Logical Operator</b></p> <pre>&amp;&amp; = And    = Or ! = Not and = And xor = Xor or = Or</pre>   |

Fuente: <https://www.php.net/manual/es/language.operators.php>

```
1  <?php
2
3  $numero1 = 20;
4  $numero2 = 30;
5  $numero3 = 30;
6  $numero4 = "30";
7
8  var_dump($numero1 > $numero2);
9  echo "<br/>";
10
11 var_dump($numero1 < $numero2);
12 echo "<br/>";
13
14 var_dump($numero1 >= $numero2);
15 echo "<br/>";
16
17 var_dump($numero1 <= $numero2);
18 echo "<br/>";
19
20 var_dump($numero2 == $numero3);
21 echo "<br/>";
22
23 var_dump($numero2 == $numero4);
24 echo "<br/>";
25
26 var_dump($numero2 === $numero4);
27 echo "<br/>";
28
```

```
28 // -1 Si Izquierda es menor,
29 // 0 Si es igual,
30 // 1 Si izquierda es mayor
31 var_dump($numero1 <=> $numero2);
32 echo "<br/>";
33
34 var_dump($numero2 <=> $numero3);
35 echo "<br/>";
36
37 var_dump($numero2 <=> $numero1);
38 echo "<br/>";
39
```

Así como podemos utilizar los diferentes operadores de comparación para verificar los datos que se encuentran almacenados en las variables podemos manipular cadenas a continuación en la siguiente imagen se puede visualizar algunos métodos utilizados en php para la manipulación de cadenas de caracteres.

```

1  <?php include
2
3  $nombreCliente = "Campers Campuslands";
4
5  // Conocer extension de un string
6  echo strlen($nombreCliente);
7  var_dump($nombreCliente);
8
9  // Eliminar espacios en blanco
10 $texto = trim($nombreCliente);
11 echo strlen($texto);
12
13 //Convertirlo a mayusculas
14 echo strtoupper($nombreCliente);
15
16 // Convertirlo en minusculas
17 echo strtolower($nombreCliente);
18
19 $mail1 = "correo@correo.com";
20 $mail2 = "Correo@correo.com";
21
22 var_dump(strtolower($mail1) === strtolower($mail2));
23 echo str_replace('Juan', 'J', $nombreCliente);
24
25 // Revisar si un string existe o no
26 echo strpos($nombreCliente, 'Pedro');
27
28 $tipoCliente = "Premium - Empresarial";
29
30 echo "<br>";

```

```

29
30 echo "<br>";
31
32 echo "El Cliente " . $nombreCliente . " es " . $tipoCliente;
33
34 echo "El Cliente {$nombreCliente} es ${tipoCliente} ";

```

```

19string(19) "Campers Campuslands" 19CAMPERS CAMPUSLANDScampers campuslandsbool(true) Campers Campuslands
El Cliente Campers Campuslands es Premium - EmpresarialEl Cliente Campers Campuslands es Premium - Empresarial

```

## 1.7. Arreglos, Arreglos asociativos y funciones para arreglos

Los arreglos en PHP son estructuras de datos que permiten almacenar múltiples valores en una sola variable. Un arreglo puede contener cualquier tipo de valor, como números, cadenas, objetos, funciones y otros arreglos. Los arreglos en PHP pueden ser indexados numéricamente o asociativamente.

Para crear un arreglo indexado numéricamente en PHP, se puede utilizar la siguiente sintaxis:

```
$miArreglo = array("valor1", "valor2", "valor3");
```

```
1 <?php  
2  
3 $carrito = ['Tablet', 'Televisión', 'Computadora'];  
4  
5 // Util para ver los contenidos de un array  
6 echo "<pre>";  
7 var_dump($carrito);  
8 echo "</pre>";  
9  
10 // Acceder a un elemento del array  
11 echo $carrito[1];  
12  
13 // Añade un elemento en el indice 3 del arreglo  
14 $carrito[3] = 'Nuevo Producto...';  
15  
16 // Añadir un elemento nuevo al final...  
17 array_push($carrito, 'Audífonos');
```

```
19 // Añadir al inicio  
20 array_unshift($carrito, 'Smartwatch');  
21  
22  
23  
24 // Util para ver los contenidos de un array  
25 echo "<pre>";  
26 var_dump($carrito);  
27 echo "</pre>";  
28  
29  
30 $clientes = array('Cliente 1', 'Cliente 2', 'Cliente 3');  
31 echo "<pre>";  
32 var_dump($clientes);  
33 echo "</pre>";  
34  
35 echo $clientes[1];  
36  
37 ?>
```

PHP cuenta con muchas funciones integradas para trabajar con arreglos, como count, sort, array\_push, array\_pop, array\_merge, array\_key\_exists, entre otras.

Los arrays asociativos en PHP son un tipo de estructura de datos que permiten asociar claves con valores. A diferencia de los arrays indexados numéricamente que utilizan números enteros para acceder a sus valores, los arrays asociativos usan una clave única para acceder a cada uno de sus valores.

En PHP, los arrays asociativos se pueden crear utilizando la siguiente sintaxis:

```
$miArrayAsociativo = array(  
    "clave1" => "valor1",  
    "clave2" => "valor2",  
    "clave3" => "valor3"  
)
```

En este ejemplo, el array asociativo \$miArrayAsociativo contiene tres pares clave-valor. La clave "clave1" está asociada con el valor "valor1", la clave "clave2" está asociada con el valor "valor2", y la clave "clave3" está asociada con el valor "valor3".

Para acceder a un valor de un array asociativo, se utiliza su clave correspondiente. Por ejemplo:

```
echo $miArrayAsociativo["clave1"]; // Imprime "valor1"  
echo $miArrayAsociativo["clave2"]; // Imprime "valor2"  
echo $miArrayAsociativo["clave3"]; // Imprime "valor3"
```

Los arrays asociativos utilizando un bucle foreach de la siguiente manera:

```
foreach ($miArrayAsociativo as $clave => $valor) {  
    echo "Clave: " . $clave . ", Valor: " . $valor;  
}
```

Ejemplo:

```

1  <?php
2  $cliente = [
3      'nombre' => 'Juan',
4      'saldo' => 200,
5      'informacion' => [
6          'tipo' => 'premium',
7          'disponible' => 100
8      ]
9  ];
10
11 echo "<pre>";
12 var_dump($cliente['informacion']);
13 echo "</pre>";
14
15 // echo $cliente['nombre'];
16 // echo $cliente['informacion']['disponible'];
17
18 $cliente['codigo'] = 1209192012;
19
20 echo "<pre>";
21 var_dump($cliente);
22 echo "</pre>";
23 ?>

```

## Funciones adicionales en arreglos

```

1  <?php
2  // in_array - buscar elementos en un arreglo
3  $carrito = ['Tablet', 'Computadora', 'Televisión'];
4
5  var_dump( in_array('Tablet', $carrito) );
6  var_dump( in_array('Audífonos', $carrito) );
7
8  // Ordenar elementos de un arreglo
9  $numeros = array(1,3,4,5,1,2);
10 sort($numeros); // de menor a mayor
11 rsort($numeros); // de mayor a menor
12
13 echo "<pre>";
14 var_dump($numeros);
15 echo "</pre>";
16
17 // Ordenar arreglo asociativo
18 $cliente = array(
19     'saldo' => 200,
20     'tipo' => 'Premium',
21     'nombre' => 'Juan'
22 );

```

```
23 echo "<pre>";
24 var_dump($cliente);
25 echo "</pre>";
26
27 asort($cliente); // Ordena por valores (orden alfabetico)
28 arsort($cliente); // Ordena por valores (Z primero)
29 ksort($cliente); // ordena por llaves (orden alfabetico);
30 krsort($cliente); // ordena por llaves (orden alfabetico, DE LA Z a la A);
31
32 echo "<pre>";
33 var_dump($cliente);
34 echo "</pre>";
35
36
37 ?>
```

## 1.8. Isset() y Empty()

En PHP, isset() y empty() son dos funciones utilizadas para verificar si una variable o un elemento de un array tiene un valor definido o no. A pesar de que ambas funciones tienen objetivos similares, hay algunas diferencias clave entre ellas.

La función isset() comprueba si una variable o un elemento de un array está definido y no es null. Esta función devuelve true si la variable o el elemento de array existe y tiene un valor, y false en caso contrario. Por ejemplo:

```
$miVariable = "Hola";
if (isset($miVariable)) {
    echo "La variable está definida y tiene un valor";
} else {
    echo "La variable no está definida o no tiene valor";
}
```

En este ejemplo, la función isset() devuelve true porque \$miVariable está definida y tiene un valor.

Por otro lado, la función empty() comprueba si una variable o un elemento de un array está vacío. Esta función devuelve true si la variable o el elemento de array no tiene un valor definido, o si tiene un valor que se considera vacío (por ejemplo, una cadena vacía, 0, false, null, un array vacío, entre otros), y false en caso contrario. Por ejemplo:

```
$miVariable = "";
if (empty($miVariable)) {
    echo "La variable está vacía o no está definida";
} else {
    echo "La variable tiene un valor";
}
```

```
1  <?php ;
2
3  $clientes = [];
4  $clientes2 = array();
5  $clientes3 = array('Pedro', 'Juan', 'Karen');
6  $cliente = [
7      'nombre' => 'Juan',
8      'saldo' => 200
9  ];
10
11 // Empty - Revisa si un arreglo esta vacio
12 var_dump( empty($clientes) );
13 var_dump( empty($clientes3) );
14 var_dump( empty($clientes2) );
```

```
15
16
17 /* Isset - Revisar si un arreglo esta creado o una
18 propiedad esta definida*/
19 echo "<br>";
20 var_dump( isset($clientes4) );
21 var_dump( isset($clientes) );
22 var_dump( isset($clientes2) );
23 var_dump( isset($clientes3) );
24
25 /* Isset - permite revisar si una propiedad de un arreglo
26 asociativo, existe!*/
27 var_dump( isset($cliente['nombre']) );
28 var_dump( isset($cliente['codigo']) );
29 ?>
```

En PHP además de contar con las funciones para verificar si existe una variable o si tiene algún dato almacenado también podemos buscar elementos en un arreglo con funciones especiales a continuación en la siguiente imagen se puede observar algunas funciones que buscan elementos en un arreglo de ejemplos.

```
// in_array - buscar elementos en un arreglo
Dumps information about a variable
var_dump( mixed $expression [, mixed $... ]): string
var_dump( in_array('Audifonos', $carrito) );
```

PHP es un lenguaje de programación muy versátil que nos ofrece una gran diversidad de funciones para la manipulación de datos que se encuentran almacenados en arreglos a continuación mostraremos en unas pequeñas líneas de código unos ejemplos.

```
// Ordenar elementos de un arreglo
$numeros = array(1,3,4,5,1,2);
sort($numeros); // de menor a mayor
rsort($numeros); // de mayor a menor

echo "<pre>";
var_dump($numeros);
echo "</pre>";
```

```
// Ordenar arreglo asociativo
$cliente = array(
    'saldo' => 200,
    'tipo' => 'Premium',
    'nombre' => 'Juan'
);

echo "<pre>";
var_dump($cliente);
echo "</pre>";
```

```
asort($cliente); // Ordena por valores (orden alfabetico)
arsort($cliente); // Ordena por valores (Z primero)
ksort($cliente); // ordena por llaves (orden alfabetico);
krsort($cliente); // ordena por llaves (orden alfabetico, DE LA Z a la A);

echo "<pre>";
var_dump($cliente);
echo "</pre>";
```

## 1.9. Estructuras de control

Las estructuras de control en programación son herramientas que se utilizan para controlar el flujo de ejecución de un programa. Estas estructuras permiten que el programa tome decisiones y realice diferentes acciones en función de ciertas condiciones.

Las estructuras de control más comunes en programación son:

- Estructuras de control condicionales: como el condicional "if" que mencionamos anteriormente, y también el "switch". Estas estructuras permiten que el programa tome decisiones en función de si se cumple o no una determinada condición.
- Estructuras de control de repetición: también conocidas como bucles o ciclos, permiten que el programa repita una determinada acción un número de veces determinado o mientras se cumpla una determinada condición. Entre los bucles más comunes se encuentran el "for", el "while" y el "do-while".
- Estructuras de control de excepciones: permiten manejar errores o situaciones inesperadas que puedan ocurrir durante la ejecución del programa. Por ejemplo, las sentencias "try" y "catch" en Java son estructuras de control de excepciones que permiten manejar errores en tiempo de ejecución.

Estas estructuras de control son fundamentales en programación, ya que permiten que el programa tome decisiones y realice acciones en función de diferentes condiciones y situaciones. Esto permite que los programas sean más flexibles, más robustos y más capaces de manejar situaciones imprevistas.

### 1.9.1. Estructuras condicionales

Las estructuras condicionales son una estructura de control de flujo en programación que permite que el programa tome decisiones basadas en si se cumple o no una condición. En otras palabras, una estructura condicional permite que el programa ejecute diferentes bloques de código en función de si se cumple o no una determinada condición.

#### 1.9.1.1. if

En programación, el condicional "if" es una estructura de control que permite tomar decisiones en función de si se cumple o no una determinada condición. Se utiliza para definir una acción que se ejecutará si se cumple la condición especificada y otra acción que se ejecutará si no se cumple.

```
if (10 > 3){  
    //Instrucciones  
}  
  
//if else if else  
if (10 > 3){  
    //Instrucciones  
}else if(10<20){  
    //Instrucciones  
}else{  
    //Instrucciones  
}
```

```
//if else  
if (10 > 3){  
    //Instrucciones  
}else{  
    //Instrucciones  
}
```

#### 1.8.1.2. Switch

En PHP, el "switch" es una estructura de control que permite ejecutar diferentes bloques de código dependiendo del valor de una variable.

La sintaxis básica del switch en PHP es la siguiente:

```
switch (expresion) {  
  
    case valor1:  
        // bloque de código si la expresión es igual a valor1  
        break;  
  
    case valor2:  
        // bloque de código si la expresión es igual a valor2  
        break;  
  
    default:  
        // bloque de código si la expresión no coincide con ninguno de los valores anteriores  
        break;  
}
```

En este ejemplo, "expresión" es la variable que se evalúa en cada uno de los casos. Si la variable coincide con "valor1", se ejecuta el bloque de código correspondiente a ese caso. Si la variable coincide con "valor2", se ejecuta el bloque de código correspondiente a ese caso. Si la variable no coincide con ninguno de los valores anteriores, se ejecuta el bloque de código en el caso "default".

Es importante notar que después de cada bloque de código, se debe incluir la sentencia "**break**" para salir del switch y evitar que se ejecuten los demás casos.

```
1  <?php
2
3  $autenticado = true;
4  $admin = false;
5
6  ∵ if($autenticado && $admin ) {
7      echo "Usuario autenticado correctamente";
8  } else {
9      echo "Usuario no autenticado, inicia sesión";
10 }
11
12 // If anidados...
13 ∵ $cliente = [
14     'nombre' => 'Juan',
15     'saldo' => 0,
16     'informacion' => [
17         'tipo' => 'Regular'
18     ]
19 ];
```

```
20
21 echo "<br>";
22
23 if( !empty($cliente) ) {
24     echo "El Arreglo de cliente no esta vacio";
25
26     if($cliente['saldo'] > 0) {
27         echo "El Cliente tiene saldo disponible";
28     } else {
29         echo "No hay saldo";
30     }
31 }
32
33 echo "<br>";
```

```
35 // else if
36 if($cliente['saldo'] > 0 ) {
37     echo "El Cliente tiene saldo";
38 } else if ($cliente['informacion']['tipo'] === 'Premium') {
39     echo "El Cliente es Premium";
40 } else {
41     echo "No hay cliente definido o no tiene saldo o no es premium";
42 }
```

```
43
44 // Switch.
45
46 echo "<br>";
47
48 $tecnologia = 'HTML';
49
50 switch ($tecnologia) {
51     case 'PHP':
52         echo "PHP, un excelente lenguaje!";
53         break;
54     case 'JavaScript':
55         echo "Genial, el lenguaje de la web";
56         break;
57     case 'HTML':
58         echo 'Emmm...';
59         break;
60     default:
61         echo "Algún lenguaje que no se cual es";
62         break;
63 }
64
65 ?>
```

## 1.10. Estructuras repetitivas

Las estructuras repetitivas, también conocidas como estructuras de control de bucle, son herramientas fundamentales en la programación que permiten repetir la ejecución de un bloque de código varias veces. Estas estructuras son utilizadas cuando se desea realizar una tarea repetitiva sin tener que escribir el mismo código una y otra vez.

En la mayoría de los lenguajes de programación, existen tres tipos de estructuras repetitivas:

- **Bucle while:** permite repetir la ejecución de un bloque de código mientras se cumpla una condición.
- **Bucle do-while:** similar al bucle while, pero garantiza que el bloque de código se ejecuta al menos una vez, independientemente de si se cumple o no la condición.
- **Bucle for:** permite repetir la ejecución de un bloque de código un número fijo de veces, controlando el número de iteraciones mediante un contador.

A continuación, veremos cómo se declara las estructuras repetitivas en el lenguaje PHP:

#### While

```
while (condición) {
    // Código a ejecutar mientras la condición sea verdadera
}
```

#### Do-while

```
do {
    // Código a ejecutar al menos una vez
} while (condición);
```

#### For

```
for (inicialización; condición; incremento/decremento) {
    // Código a ejecutar en cada iteración
}
```

#### Ejemplo

```
1 <?php
2
3 // While
4
5 $i = 0; // Inicializador
6
7 while($i < 10) {
8
9     echo $i . "<br>";
10
11    $i++; // Incremento
12 }
13
14 echo "<br>";
15
16 // Do While
17 $i = 100;
18
19 do {
20     echo $i . "<br>";
21
22    $i++;
23 } while($i < 10);
```

```
47 // For Each
48 $clientes = array('Pedro', 'Juan', 'Karen');
49
50 foreach( $clientes as $cliente ):
51     echo $cliente . '<br/>';
52 endforeach;
53
54 $cliente = [
55     'nombre' => 'Juan',
56     'saldo' => 200,
57     'tipo' => 'Premium'
58 ];
59
60 foreach( $cliente as $key => $valor ):
61     echo $key . " - " . $valor . '<br/>';
62 endforeach;
63
64
65
66 ?>
```

Ejemplo foreach

```
$productos = [
    [
        'nombre' => 'Tablet',
        'precio' => 200,
        'disponible' => true
    ],
    [
        'nombre' => 'Televisión 24"',
        'precio' => 300,
        'disponible' => true
    ],
    [
        'nombre' => 'Monitor Curvo',
        'precio' => 400,
        'disponible' => false
    ]
];

foreach( $productos as $producto ) { ?>
    <li>
        <p>Producto: <?php echo $producto['nombre']; ?> </p>
        <p>Precio: <?php echo "$" . $producto['precio']; ?> </p>
        <p><?php echo ($producto['disponible']) ? 'Disponible' : 'No Disponible'; ?> </p>
    </li>
<?php
} 
```

## 1.11 Funciones definidas por el usuario

En programación, las funciones definidas por el usuario son bloques de código que se pueden llamar y ejecutar en cualquier parte de un programa para realizar una tarea específica. Estas funciones permiten que el código sea modular y reutilizable, lo que facilita la programación y el mantenimiento del código.

Al definir una función, se le da un nombre y se especifica qué hace la función cuando se la llama. La definición de la función incluye la lista de parámetros que la función espera recibir, y también puede incluir una declaración de tipo de retorno que indica qué tipo de valor se devolverá cuando se llame a la función.

Una vez definida la función, se puede llamar desde cualquier parte del programa simplemente escribiendo su nombre y proporcionando los argumentos necesarios. Cuando se llama a la función, se ejecuta el código definido en la función y, si se especificó un valor de retorno, se devuelve ese valor.

Las funciones definidas por el usuario son una herramienta muy útil para organizar y simplificar el código de un programa, ya que permiten descomponer tareas complejas en tareas más pequeñas y manejables. También pueden ayudar a mejorar la legibilidad del código, ya que se pueden dar nombres significativos a las funciones para describir lo que hacen.

### 1.11.1. Funciones en PHP

En PHP se pueden definir funciones siguiendo la siguiente estructura:

```
function Identificador ([p1],[p2]...){
```

```
    Expresiones.....
```

```
}
```

Cómo se visualiza en el ejemplo anterior en PHP se utiliza la palabra reservada `function` para indicar el inicio de la función a continuación el identificador que hace referencia al nombre que se le va a dar a la función, es importante recordar que los identificadores de funciones no deben contener espacios en blanco ni caracteres especiales; a continuación entre paréntesis van los parámetros no te que se definen `p1,p2` y están encerrados entre paréntesis cuadrados o corchetes cuando nosotros encontramos unos elementos entre corchetes esto indica que los parámetros son opcionales y va depender de El objetivo que tenga la función. En programación nos podemos encontrar con dos tipos de funciones las primeras son aquellas que no retornan ningún tipo de valor y la segunda son aquellas funciones que permiten retornar un valor el cual es generado a partir de un proceso interno que se lleva a cabo en la función.

### 1.12.2. Funciones que no retornan valor

En programación, una función que no retorna un valor se llama una función de "void". En PHP, se puede definir una función de void utilizando la palabra clave **void** en lugar de especificar un tipo de retorno. En el lenguaje de programación PHP se puede utilizar la palabra reservada `void` aunque no

es obligatoria en el siguiente ejemplo se podrá visualizar la misma función aplicando la palabra reservada void y no aplicándola.

```
1 <?php
2 declare(strict_types=1);
3 function sumar(int $numero1 = 0, array $numero2 ):void {
4     echo $numero1 + $numero2;
5 }
6 sumar(10, []);
7 ?>
```

```
1 <?php
2 declare(strict_types=1);
3 function sumar(int $numero1 = 0, array $numero2 ) {
4     echo $numero1 + $numero2;
5 }
6 sumar(10, []);
7 ?>
```

En el código visualizado en la imagen anterior podemos encontrar los siguientes: en la línea número dos se encuentra una declaración de tipos estrictos esta declaración se utiliza para forzar a la definición de un tipo de dato a una variable esto quiere decir que no dejaremos a la suerte El tipo de dato y su contenido, en la línea número 3 nos encontraremos con la definición de la función en este caso tiene como nombre sumar y podemos observar que recibe dos argumentos el primer argumento es un número entero el segundo argumento es un arreglo nótese que esos argumentos tienen definido un valor opcional en el caso de la variable número 1. y en la línea número 6 podemos observar cómo se hace el llamado a la función sumar y entre paréntesis se envían los valores que queremos procesar en la función sumar. Es importante tener en cuenta que el primer valor se asigna en el orden de los parámetros de la función esto quiere decir que el número 10 se almacenará en la variable número 1 y el segundo valor que son los paréntesis cuadrados que se utilizan para representar un arreglo se almacena en el segundo argumento.



Es de buena práctica utilizar la palabra reservada void a aquellas funciones que no retorna ningún tipo de valor esto me facilita la labor de identificar cuál función me retorna y cuál función no me retorna ningún valor.

#### 1.12.3. Funciones que retornan valor

En PHP al igual que en cualquier otro lenguaje de programación se pueden definir funciones que permiten recordar un valor al momento de finalizar la ejecución de todas las expresiones que se encuentran en su cuerpo principal. A continuación, se podrá visualizar un ejemplo en el cual se contará con una función de autenticación de usuario la cual recibirá como parámetro un valor de tipo booleano y retornará como respuesta un valor de tipo cadena con la respuesta final.

```

1  <?php
2  declare(strict_types=1);
3  //include 'includes/header.php';
4  function usuarioAutenticado(bool $autenticado) : ?string {
5      if($autenticado) {
6          return "El Usuario esta autenticado";
7      } else {
8          return null;
9      }
10 }
11 $usuario = usuarioAutenticado(false);
12 echo $usuario;
13 //include 'includes/footer.php';
14 ?>

```

En el ejemplo anterior podemos observar las siguientes características; en la línea número cuatro se declara de la función `usuarioAutenticado` la cual recibe como parámetro un valor booleano y retorna un string. Es importante tener en cuenta que para indicar el tipo de dato que va a retornar la función se debe utilizar los dos puntos ( : ) seguido de un símbolo de interrogante(?) y el tipo de dato a retornar. En la línea número 5 se encuentra definir una estructura condicional que permite validar si la variable `$autenticado` contiene verdadero o falso, si el valor es verdadero o true la función retornará el mensaje el usuario está autenticado en caso contrario retornará nulo. Es importante tener en cuenta que para que una función retorne el valor se debe utilizar la palabra reservada `return`.

### 1.13 include, require, include\_once, require\_once

En PHP, **include**, **require**, **include\_once** y **require\_once** son funciones que se utilizan para incluir archivos externos en un programa.

**include** y **include\_once** permiten incluir un archivo PHP en el programa. La diferencia entre ellas es que **include** puede incluir el mismo archivo varias veces mientras que **include\_once** asegura que el archivo solo se incluya una vez en el programa.

**require** y **require\_once** son similares a **include** y **include\_once**, pero en caso de que el archivo no pueda ser encontrado, se detendrá la ejecución del programa. Al utilizar **require\_once**, se asegura que el archivo solo se incluya una vez en el programa.

Estas funciones son útiles para incluir archivos de código externos en un programa PHP, lo que permite la reutilización de código y la separación de la lógica del programa en diferentes archivos.

A continuación, se dejan algunos enlaces de referencia que tienen información adicional acerca de `include`, `include_once`, `require` y `require_once`.

- `include`: <http://www.php.net/manual/es/function.include.php>
- `include_once`: <http://www.php.net/manual/es/function.include-once.php>
- `require`: <http://www.php.net/manual/es/function.require.php> y
- `require_once`: <http://www.php.net/manual/es/function.require-once.php>

En PHP existen grandes diferencias en la inclusión de archivos remotos y locales; los archivos que son incluidos de forma remota son aquellos que son interpretados previamente por el servidor origen y servidos al servidor destino es decir aquel que los está incluyendo, dichos archivos ya se encuentran interpretados; mientras que los archivos que son incluidos de forma local no serán previamente interpretados por el servidor ya que este proceso lo realizará el archivo que los incluyó.

Ejercicio:

En el siguiente ejercicio vamos a implementar la inclusión de un archivo que contenga un encabezado que diga desarrollo de aplicaciones con PHP y en la parte inferior un subtítulo que diga introducción a PHP. A continuación, se muestra una imagen de cómo debería quedar el sitio.

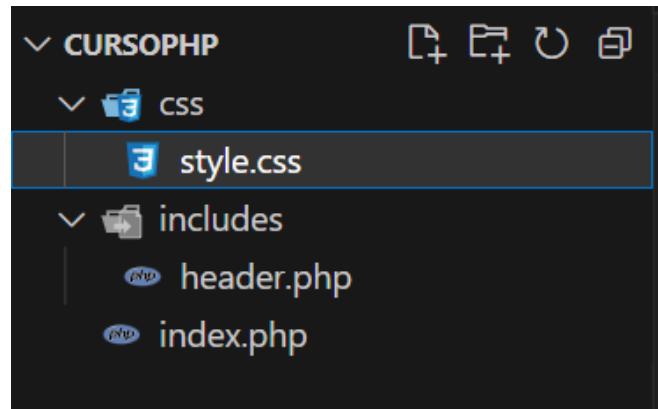


a continuación, se muestra la definición de las reglas CSS para el ejemplo dado:

```
1 body {
2     background: #4b6cb7; /* fallback for old browsers */
3     background: -webkit-linear-gradient(to right, #182848, #4b6cb7); /* Chrome 10-25, Safari 5.1-6 */
4     background: linear-gradient(to right, #182848, #4b6cb7); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
5 }
6
7 h1, p, a {
8     font-family: 'Carrois Gothic SC', sans-serif;
9     display: block;
10    text-align: center;
11    line-height:.8;
12    color: white;
13 }
14 .contenido {
15    max-width: 1200px;
16    margin: 5rem auto 0 auto;
17    color: white;
18    font-size: 3rem;
19    font-family: 'Carrois Gothic SC', sans-serif;
20 }
21 h1 {
22    font-size: 4rem;
23 }
24 a {
25    font-size: 1.4rem;
26 }
27 p {
28    font-size: 2.2rem;
29 }

30 .hashtag {
31     font-family: Arial, Helvetica, sans-serif;
32     margin-bottom: 2rem;
33 }
34 .resultado {
35     margin-top:10rem;
36     text-align: left;
37     font-size: 2rem;
38 }
```

A continuación, mostraremos la estructura del proyecto:



Como vemos en la estructura los archivos cambian un poco con respecto a la programación en HTML como en html los archivos suelen terminar en extensión html por buenas prácticas cuando desarrollamos en PHP debemos acostumbrarnos a nombrar los archivos con extensión PHP para no perder la estructura del lenguaje.

A continuación, se expone el código De El archivo header PHP y index. PHP

```
header.php
1 <div class="contenido">
2   <h1>Desarrollo de aplicaciones con PHP</h1>
3   <p>Introducción a PHP</p>
4 </div>
```

```
index.php
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <link rel="stylesheet" href="css/style.css">
8     <title>Document</title>
9   </head>
10  <body>
11    <header>
12      <?php include('includes/header.php') ?>
13    </header>
14  </body>
15 </html>
```

## 1.14. Json\_encode y json\_decode

La función **json\_encode()** en PHP se utiliza para convertir una estructura de datos en PHP en una cadena JSON.

En el siguiente ejemplo podremos observar el uso de la función json\_encode.

```
1  <?php
2      $productos = [
3          [
4              'nombre' => 'Tablet',
5              'precio' => 200,
6              'disponible' => true
7          ],
8          [
9              'nombre' => 'Televisión 24"',
10             'precio' => 300,
11             'disponible' => true
12         ],
13         [
14             'nombre' => 'Monitor Curvo',
15             'precio' => 400,
16             'disponible' => false
17         ]
18     ];
19 ?>
20 <!DOCTYPE html>
21 <html lang="en">
22 <head>
23     <meta charset="UTF-8">
24     <meta http-equiv="X-UA-Compatible" content="IE=edge">
25     <meta name="viewport" content="width=device-width, initial-scale=1.0">
26     <link rel="stylesheet" href="css/style.css">
27     <title>Document</title>
28 </head>
29
30 <body>
31     <header>
32         <?php include('includes/header.php') ?>
33     </header>
34     <h1>Uso de JSON Example</h1>
35     <nav>
36         <?php include('includes/enlaces.php') ?>
37     </nav>
38     <main>
39         <pre class="resultado">
40             <?php
41                 var_dump($productos);
42                 $json = json_encode($productos, JSON_UNESCAPED_UNICODE);
43             ?>
44             <br>
45             <?php
46                 var_dump($json);
47             ?>
48         </pre>
49     </main>
50 </body>
51 </html>
```

La función **json\_decode()** en PHP se utiliza para convertir una cadena JSON en una estructura de datos de PHP. Esta función toma una cadena JSON y la convierte en un objeto, un array asociativo o un valor escalar según corresponda.

Aquí un ejemplo básico de cómo usar **json\_decode()**:

```

1  <?php
2      $productos = [
3          [
4              'nombre' => 'Tablet',
5              'precio' => 200,
6              'disponible' => true
7          ],
8          [
9              'nombre' => 'Televisión 24"',
10             'precio' => 300,
11             'disponible' => true
12         ],
13         [
14             'nombre' => 'Monitor Curvo',
15             'precio' => 400,
16             'disponible' => false
17         ]
18     ];
19
20     $json = '{"nombre":"Jose Manuel","edad":16,"ciudad":"Nueva York"}';
21
22     $data = json_decode($json);
23 ?>

```

```

24     !DOCTYPE html
25     <html lang="en">
26     <head>
27         <meta charset="UTF-8">
28         <meta http-equiv="X-UA-Compatible" content="IE=edge">
29         <meta name="viewport" content="width=device-width, initial-scale=1.0">
30         <link rel="stylesheet" href="css/style.css">
31         <title>Document</title>
32     </head>
33     <body>
34         <header>
35             <?php include('includes/header.php') ?>
36         </header>
37         <h1>Uso de JSON Example</h1>
38         <nav>
39             <?php include('includes/enlaces.php') ?>
40         </nav>
41         <main>
42             <pre class="resultado">
43                 <?php
44                     var_dump($productos);
45                     $json = json_encode($productos, JSON_UNESCAPED_UNICODE);
46                 ?>
47                 <br>
48                 <?php
49                     var_dump($json);
50                 ?>
51             </pre>

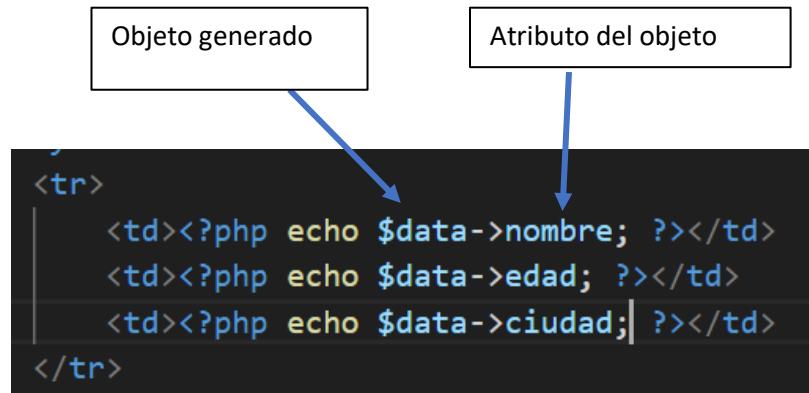
```

```

52         <table class="contenido">
53             <thead>
54                 <th>Nombre</th>
55                 <th>Edad</th>
56                 <th>Ciudad</th>
57             </thead>
58             <tbody>
59                 <tr>
60                     <td><?php echo $data->nombre ?></td>
61                     <td><?php echo $data->edad ?></td>
62                     <td><?php echo $data->ciudad ?></td>
63                 </tr>
64             </tbody>
65         </table>
66     </main>
67 </body>
68 </html>

```

En el ejercicio anterior pudimos observar que la función Json-decode permite convertir una cadena Jason y representarla en un objeto PHP tomado data al cual podemos acceder de forma directa a través del nombre objeto y sus propiedades. Es importante tener en cuenta que la forma de acceder a la información en este tipo de objetos es utilizando el nombre del objeto el símbolo flecha y la propiedad.



En caso de que se desee manipular el objeto de tipo PHP como un arreglo asociativo se puede agregar un segundo argumento en la función json-decode La cual es un valor booleano; en caso de que se le pase un valor true la función retornará un arreglo de tipo asociativo. A continuación, veremos un ejemplo.

```
1  <?php
2      $productos = [
3          [
4              'nombre' => 'Tablet',
5              'precio' => 200,
6              'disponible' => true
7          ],
8          [
9              'nombre' => 'Televisión 24"',
10             'precio' => 300,
11             'disponible' => true
12         ],
13         [
14             'nombre' => 'Monitor Curvo',
15             'precio' => 400,
16             'disponible' => false
17         ]
18     ];
19
20     $json = '{"nombre":"Jose Manuel","edad":16,"ciudad":"Nueva York"}';
21
22     $data = json_decode($json); //Retorno del objeto php
23     $dataAsocc = json_decode($json,true); // Retorno de un arreglo asociativo
24 ?>
```

```

25  <!DOCTYPE html>
26  <html lang="en">
27  <head>
28      <meta charset="UTF-8">
29      <meta http-equiv="X-UA-Compatible" content="IE=edge">
30      <meta name="viewport" content="width=device-width, initial-scale=1.0">
31      <link rel="stylesheet" href="css/style.css">
32      <title>Document</title>
33  </head>
34  <body>
35      <header>
36          <?php include('includes/header.php') ?>
37      </header>
38      <h1>Uso de JSON Example</h1>
39      <nav>
40          <?php include('includes/enlaces.php') ?>
41      </nav>
42      <main>
43          <pre class="resultado">
44              <?php
45                  var_dump($productos);
46                  $json = json_encode($productos, JSON_UNESCAPED_UNICODE);
47              ?>
48              <br>
49              <?php
50                  var_dump($json);
51              ?>
52          </pre>
53
54          <table class="contenido">
55              <thead>
56                  <th>Nombre</th>
57                  <th>Edad</th>
58                  <th>Ciudad</th>
59              </thead>
60              <tbody>
61                  <tr>
62                      <td><?php echo $dataAsocc['nombre']; ?></td>
63                      <td><?php echo $dataAsocc['edad']; ?></td>
64                      <td><?php echo $dataAsocc['ciudad']; ?></td>
65                  </tr>
66              </tbody>
67          </table>
68      </main>
69  </body>

```

Resumen Funciones más usadas en manipulación de data:

**array\_flip():** Intercambia las claves con sus valores correspondientes en un array.

**array\_fill():** Rellena un array con un valor especificado.

**array\_filter():** Filtra los elementos de un array utilizando una función de devolución de llamada.

**array\_map():** Aplica una función a cada elemento de un array y devuelve un nuevo array con los resultados.

**array\_reduce():** Reduce un array a un solo valor aplicando una función de devolución de llamada.

**array\_key\_exists():** Comprueba si una clave existe en un array.

**in\_array():** Comprueba si un valor existe en un array.

**array\_rand():** Devuelve una o varias claves aleatorias de un array.



`array_unique()`: Elimina los valores duplicados de un array.

`array_intersect()`: Devuelve un array con los valores comunes a todos los arrays dados.

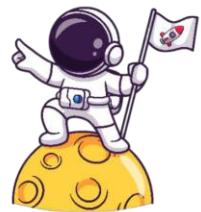
`array_diff()`: Devuelve un array con los valores del primer array que no están presentes en los arrays adicionales.

`array_push()`: Agrega uno o más elementos al final de un array.

`array_pop()`: Extrae y elimina el último elemento de un array.

`array_reverse()`: Revierte el orden de los elementos en un array.

`array_sum()`: Devuelve la suma de todos los valores de un array numérico.



## 2. PHP Intermedio

### 2.1. Programación Orientada a objetos



La programación orientada a objetos (POO) es un paradigma de programación que se basa en el concepto de "objetos". Los objetos son entidades que representan conceptos del mundo real y que pueden tener propiedades (atributos) y realizar acciones (métodos).

En la programación orientada a objetos, los objetos son la base fundamental y se crean a partir de clases. Una clase es una plantilla o un molde que define las propiedades y comportamientos que tendrán los objetos que se creen a partir de ella.

Los principales conceptos de la programación orientada a objetos son:

- **Clase:** Es una plantilla o definición que describe las características y comportamientos de los objetos que se pueden crear a partir de ella.
- **Objeto:** Es una instancia de una clase. Representa un individuo o entidad específica y tiene sus propias propiedades y comportamientos.
- **Atributos:** Son las propiedades o características de un objeto. Definen el estado de un objeto y se representan mediante variables en la clase.
- **Métodos:** Son las acciones o comportamientos que un objeto puede realizar. Representan las operaciones que pueden realizarse con un objeto y se definen como funciones en la clase.

- **Encapsulación:** Es el principio que establece que los atributos y métodos relacionados deben agruparse en una clase para ocultar los detalles internos y exponer solo una interfaz pública. Esto se logra mediante la especificación de niveles de acceso (público, privado, protegido) para los atributos y métodos.
- **Herencia:** Es un mecanismo que permite crear nuevas clases basadas en clases existentes. La clase que se utiliza como base se denomina "clase padre" o "superclase", y la clase que se deriva se llama "clase hija" o "subclase". La herencia permite la reutilización de código y la creación de jerarquías de clases.
- **Polimorfismo:** Es la capacidad de un objeto de tomar diferentes formas o comportarse de diferentes maneras según el contexto. Permite utilizar una interfaz común para objetos de diferentes clases y proporciona flexibilidad y extensibilidad en el diseño de programas.

La programación orientada a objetos permite organizar y modularizar el código de manera más clara y estructurada, lo que facilita el desarrollo, la mantenibilidad y la reutilización del código. Es ampliamente utilizado en diversos lenguajes de programación, como Java, C++, Python, PHP, entre otros.

#### 2.1.1. Modificadores de acceso en PHP

Los modificadores de acceso son palabras clave utilizadas en la programación orientada a objetos para controlar la visibilidad y el acceso a los miembros (atributos y métodos) de una clase. Estos modificadores permiten establecer qué partes del código pueden acceder y modificar dichos miembros.

En PHP, hay tres modificadores de acceso para controlar la visibilidad de propiedades y métodos en una clase:

- **public:** Los miembros declarados como **public** son accesibles desde cualquier lugar, ya sea desde dentro de la clase, desde las clases heredadas o desde fuera de la clase. Son visibles para todos.
- **private:** Los miembros declarados como **private** solo son accesibles desde dentro de la misma clase en la que se definen. No pueden ser accedidos desde fuera de la clase, ni siquiera por las clases heredadas.
- **protected:** Los miembros declarados como **protected** son accesibles desde dentro de la misma clase y desde las clases heredadas (subclases). Sin embargo, no pueden ser accedidos desde fuera de la clase directamente.



Es importante elegir adecuadamente los modificadores de acceso en función de las necesidades de diseño y la seguridad de la aplicación. Un buen diseño de clases utiliza el encapsulamiento y establece un acceso controlado a los miembros, evitando el acceso directo a los datos internos desde fuera de la clase y fomentando el uso de métodos para su manipulación.

## 2.2. Clases

Como vemos en el apartado introductorio a la programación orientada a objetos una clase es una plantilla que nos permite definir las características y comportamientos de los objetos que se pueden crear a partir de dicha clase a continuación veremos un ejercicio en el cual se evidencia como crear una clase y como instancia de una clase en PHP.

```
1 <?php
2     class Persona {
3         private $nombre;
4         protected $edad;
5
6         public function __construct($nombre, $edad) {
7             $this->nombre = $nombre;
8             $this->edad = $edad;
9         }
10        public function getNombre() {
11            return $this->nombre;
12        }
13        public function setNombre($nombre) {
14            $this->nombre = $nombre;
15        }
16        public function getEdad() {
17            return $this->edad;
18        }
19        public function setEdad($edad) {
20            $this->edad = $edad;
21        }
22        private function saludar() {
23            echo "Hola, mi nombre es " . $this->nombre;
24        }
25    }
26 ?>
```

Instanciar clases

```

1  <?php
2      require_once('clases/Persona.php');
3      $alumno = new Persona('Jose Manuel', 17)
4  ?>

```

Acceder a la información de la instancia de la clase

```

5  <!DOCTYPE html>
6  <html lang="en">
7  <head>
8      <meta charset="UTF-8">
9      <meta http-equiv="X-UA-Compatible" content="IE=edge">
10     <meta name="viewport" content="width=device-width, initial-scale=1.0">
11     <link rel="stylesheet" href="css/style.css">
12     <title>Document</title>
13 </head>
14 <body>
15     <header>
16         |     <?php include('includes/header.php') ?>
17     </header>
18     <h1>Programacion orientada a objetos en php</h1>
19     <nav>
20         |     <?php include('includes/enlaces.php') ?>
21     </nav>
22     <main>
23         <pre class="resultado">
24             <?php echo $alumno->getNombre(); ?>
25             <?php echo $alumno->getEdad(); ?>
26         </pre>
27     </main>
28 </body>
29 </html>

```

Atributos de clases en phpV8 o Sup

```

1  <?php
2      class Persona {
3
4          public function __construct(private string $nombre, protected int $edad) {
5              $this->nombre = $nombre;
6              $this->edad = $edad;
7          }
8          public function getNombre() {
9              return $this->nombre;
10         }
11         public function setNombre($nombre) {
12             $this->nombre = $nombre;
13         }
14         public function getEdad() {
15             return $this->edad;
16         }
17         public function setEdad($edad) {
18             $this->edad = $edad;
19         }
20         private function saludar() {
21             echo "Hola, mi nombre es " . $this->nombre;
22         }
23     }
24 ?>

```

### 2.2.1. Métodos estáticos

En programación, un método estático es un método que pertenece a la clase en sí y no a una instancia específica de la clase. A diferencia de los métodos de instancia, los métodos estáticos se pueden llamar directamente en la clase sin necesidad de crear un objeto o instancia de la misma.

Algunas características importantes de los métodos estáticos en PHP son:

- No requieren una instancia: Los métodos estáticos se pueden invocar directamente desde la clase, utilizando la sintaxis **Clase::metodoEstatico()**, sin necesidad de crear un objeto de la clase.
- No pueden acceder a propiedades de instancia: Los métodos estáticos no pueden acceder directamente a las propiedades de instancia de la clase, ya que no tienen una instancia específica asociada. Solo pueden acceder a propiedades estáticas (variables estáticas) que pertenezcan a la clase.
- No pueden utilizar **\$this**: En un método estático, no se puede utilizar la palabra clave **\$this** para hacer referencia a la instancia actual de la clase, ya que no hay una instancia asociada.
- Útiles para utilidades compartidas: Los métodos estáticos son útiles para definir funciones o utilidades que no dependen del estado de una instancia específica. Se pueden utilizar para operaciones globales, cálculos matemáticos, acceso a bases de datos, manipulación de archivos, etc.

### Ejemplo

```
<?php
class Persona {
    private string $nombre;
    protected int $edad;
    private static $nombreAux;
    public function __construct($nombre, $edad) {
        $this->nombre = $nombre;
        $this->edad = $edad;
        self::$nombreAux = $nombre;
    }
    public function getNombre() {
        return $this->nombre;
    }
    public function setNombre($nombre) {
        $this->nombre = $nombre;
    }
    public function getEdad() {
        return $this->edad;
    }
    public function setEdad($edad) {
        $this->edad = $edad;
    }
    public static function saludar() {
        return '<br>Hola como estas ' . self::$nombreAux;
    }
}
?>
```

Llamando método estático de la clase.

```

<?php

    include_once 'clases/Persona.php';
    $alumno = new Persona('Jose Pardo',23);

?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css/style.css">
    <title>Document</title>
</head>
<body>
    <header>
        <?php include('includes/header.php') ?>
    </header>
    <h1>Programacion orientada a objetos en php</h1>
    <nav>
        <?php include('includes/enlaces.php') ?>
    </nav>
    <main>
        <pre class="resultado">
            <?php
                echo $alumno->getNombre();
                echo $alumno->getEdad();
                echo Persona::saludar();
            ?>
        </pre>
    </main>
</body>
</html>

```

### 2.3. Herencia

La herencia en programación es un concepto que permite crear nuevas clases basadas en clases existentes, aprovechando y extendiendo su funcionalidad. La clase existente se conoce como clase base o clase padre, mientras que la nueva clase creada se llama clase derivada o clase hija.

Algunos conceptos importantes relacionados con la herencia son los siguientes:

- Clase base / Clase padre: Es la clase original de la cual se deriva una nueva clase. Define los atributos y métodos básicos que serán heredados por las clases derivadas.
- Clase derivada / Clase hija: Es la nueva clase creada que se basa en la clase base. Hereda los atributos y métodos de la clase base y puede agregar nuevos atributos y métodos, así como modificar o ampliar los existentes.

- **Herencia simple y herencia múltiple:** La herencia simple se refiere a la relación en la que una clase derivada hereda de una sola clase base. Por otro lado, la herencia múltiple se refiere a la relación en la que una clase derivada hereda de múltiples clases base. No todos los lenguajes de programación admiten la herencia múltiple.
- **Polimorfismo:** El polimorfismo es la capacidad de un objeto de una clase derivada para ser tratado como un objeto de su clase base. Esto permite utilizar una referencia de la clase base para manipular objetos de diferentes clases derivadas sin tener que conocer la clase concreta en tiempo de compilación.

### Ejemplo

```

class Transporte {
    public function __construct(protected int $ruedas, protected int $capacidad)
    {
    }

    public function getInfo() : string {
        return "El transporte tiene " . $this->ruedas . " ruedas y una capacidad de " . $this->capacidad . " personas ";
    }

    public function getRuedas() : int {
        return $this->ruedas;
    }
}

class Bicicleta extends Transporte {

    public function getInfo() : string {
        return "El transporte tiene " . $this->ruedas . " ruedas y una capacidad de " . $this->capacidad .
    " personas y NO GASTA GASOLINA ";
    }
}

class Automovil extends Transporte {
    public function __construct(protected int $ruedas, protected int $capacidad, protected string $transmision)
    {
    }

    public function getTransmision() : string {
        return $this->transmision;
    }
}

```

```

<?php
    include_once 'clases/herencia.php';
    $bicicleta = new Bicicleta(2, 1);
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css/style.css">
    <title>Document</title>
</head>
<body>
    <header>
        <?php include('includes/header.php') ?>
    </header>
    <h1>Programacion orientada a objetos en php</h1>
    <nav>
        <?php include('includes/enlaces.php') ?>
    </nav>
    <main>
        <pre class="resultado">
            <?php
                echo $bicicleta->getInfo();
                echo $bicicleta->getRuedas();
            ?>
            <hr>
            <?php
                $auto = new Automovil(4, 4, 'Manual');
                echo $auto->getInfo();
                echo $auto->getTransmision();
            ?>
        </pre>
    </main>
</body>
</html>

```

## 2.4. Clases Abstractas

En PHP, una clase abstracta es una clase que no se puede instanciar directamente, sino que sirve como una plantilla o base para otras clases. Se utiliza para definir la estructura común y los métodos que deben implementar las clases hijas.

Algunos aspectos importantes sobre las clases abstractas en PHP son los siguientes:

- Definición de una clase abstracta: Para definir una clase abstracta, se utiliza la palabra clave **abstract** antes de la declaración de la clase. Por ejemplo:

```

abstract class ClaseAbstracta {
    // Declaración de propiedades y métodos
}

```

- Métodos abstractos: Una clase abstracta puede contener métodos abstractos, que son métodos que no tienen implementación en la clase abstracta, sino que deben ser implementados en las clases hijas. La declaración de un método abstracto no incluye el cuerpo del método. Por ejemplo:

```
abstract class ClaseAbstracta {
    abstract public function metodoAbstracto();
}
```

- Herencia de una clase abstracta: Las clases hijas de una clase abstracta deben implementar todos los métodos abstractos definidos en la clase abstracta. Si una clase hija no implementa todos los métodos abstractos, también debe ser declarada como una clase abstracta. Una clase hija puede extender solo una clase abstracta a la vez.
- Instanciación de clases abstractas: No es posible crear una instancia directa de una clase abstracta utilizando el operador **new**. Sin embargo, se pueden utilizar las clases hijas para crear instancias.
- Implementación de métodos abstractos: Las clases hijas deben proporcionar una implementación concreta de los métodos abstractos definidos en la clase abstracta. La implementación debe tener la misma firma (nombre y parámetros) que el método abstracto en la clase abstracta.

Ejemplo:

```
<?php
abstract class Animal {
    abstract public function hacerSonido();
}

class Perro extends Animal {
    public function __constructor() {
    }
    public function hacerSonido() {
        echo "¡Guau!";
    }
}
class Gato extends Animal {
    public function __constructor() {
    }
    public function hacerSonido() {
        echo "Miiiauuu!";
    }
}
?>
```

```

<?php
    include_once 'clases/abstractas.php';
    $pluto = new Perro();
    $garfield = new Gato();
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css/style.css">
    <title>Document</title>
</head>
<body>
    <header>
        <?php include('includes/header.php') ?>
    </header>
    <h1>Programacion orientada a objetos en php - Clases Abstractas</h1>

    <main>
        <pre class="resultado">
            <h2>El perro hace</h2>
            <?php
                echo $pluto->hacerSonido();
            ?>
            <hr>
            <h2>El gato hace</h2>
            <?php
                echo $garfield->hacerSonido();
            ?>
        </pre>
    </main>
</body>
</html>

```

## 2.5. Interfaces

En la programación orientada a objetos, una interfaz es una estructura que define un conjunto de métodos que una clase debe implementar. Es un contrato que especifica qué métodos debe proporcionar una clase sin especificar cómo se implementan esos métodos.

Definición

```

interface MiInterfaz {
    public function metodo1();
    public function metodo2();
    // ...
}

```

Ejemplo básico

```
interface Figura {
    public function calcularArea();
}

class Circulo implements Figura {
    private $radio;

    public function __construct($radio) {
        $this->radio = $radio;
    }

    public function calcularArea() {
        return pi() * pow($this->radio, 2);
    }
}

$circulo = new Circulo(5);
echo $circulo->calcularArea(); // Imprimirá el área del círculo
```

En PHP, es posible lograr herencia entre interfaces mediante la utilización de la palabra clave **extends**. Esto permite extender una interfaz existente para agregar nuevos métodos o requerir la implementación de métodos adicionales.

Ejemplo herencia entre interfaces

```

<?php

interface Figura {
    public function calcularArea();
}

interface Figura3D extends Figura {
    public function calcularVolumen();
}

class Cubo implements Figura3D {
    private $lado;

    public function __construct($lado) {
        $this->lado = $lado;
    }

    public function calcularArea() {
        return 6 * pow($this->lado, 2);
    }

    public function calcularVolumen() {
        return pow($this->lado, 3);
    }
}

$cubo = new Cubo(5);
echo $cubo->calcularArea();      // Imprimirá el área del cubo
echo $cubo->calcularVolumen();

?>

```

A continuación, otro ejemplo de interfaces en php

```

<?php

interface TransporteInterfaz {
    public function getInfo() : string;
    public function getRuedas() : int;
}

class Transporte implements TransporteInterfaz {
    public function __construct(protected int $ruedas, protected int $capacidad)
    {

    }

    public function getInfo() : string {
        return "El transporte tiene " . $this->ruedas . " ruedas y una capacidad de " . $this->capacidad .
" personas ";
    }

    public function getRuedas() : int {
        return $this->ruedas;
    }
}

?>

```

## 2.6. Polimorfismo

El polimorfismo en la programación orientada a objetos es un concepto que permite tratar objetos de diferentes clases de manera uniforme, utilizando una interfaz común. Se basa en la capacidad de los objetos de una jerarquía de clases de responder de manera diferente a la misma llamada de método.

- Herencia: El polimorfismo se logra a través de la herencia, donde una clase hija hereda de una clase padre y puede redefinir o sobrescribir los métodos heredados de la clase padre.
- Sustitución: Los objetos de las clases hijas pueden ser tratados como objetos de la clase padre, lo que permite utilizarlos en lugar de objetos de la clase padre sin que se produzcan errores o comportamientos inesperados.

El polimorfismo se utiliza para crear código más flexible y modular, ya que permite escribir código que pueda manejar diferentes tipos de objetos de manera genérica sin preocuparse por los detalles específicos de cada clase.

```
<?php

interface TransporteInterfaz {
    public function getInfo() : string;
    public function getRuedas() : int;
}

class Transporte implements TransporteInterfaz {
    public function __construct(protected int $ruedas, protected int $capacidad)
    {

    }

    public function getInfo() : string {
        return "El transporte tiene " . $this->ruedas . " ruedas y una capacidad de " . $this->capacidad . " personas ";
    }

    public function getRuedas() : int {
        return $this->ruedas;
    }
}

class Automovil extends Transporte implements TransporteInterfaz {
    public function __construct(protected int $ruedas, protected int $capacidad, protected string $color)
    {

    }

    public function getInfo() : string {
        return "El transporte AUTO tiene " . $this->ruedas . " ruedas y una capacidad de " . $this->capacidad .
" personas y tiene el color" . $this->color;
    }

    public function getColor() : string {
        return "El color es " . $this->color;
    }
}

echo "<pre>";

var_dump($transporte = new Transporte(8, 20));
var_dump($auto = new Automovil(4, 4, 'Rojo'));

echo $transporte->getInfo();
echo "<br>";

echo $auto->getInfo();
echo "<br>";

echo $auto->getColor();

echo "</pre>";

?>
```

## 2.7. Autoload

En PHP, el autoloading (carga automática) es una técnica que permite cargar automáticamente las clases cuando son necesarias, sin tener que incluir manualmente los archivos de clase en cada punto del código. Esto facilita el desarrollo y el mantenimiento del código, ya que no es necesario preocuparse por incluir los archivos de clase de forma explícita.

El autoloading en PHP se basa en la función **spl\_autoload\_register()**, que permite registrar una o varias funciones de autoload. Estas funciones se ejecutan automáticamente cuando se intenta utilizar una clase que aún no ha sido cargada.

```
1 <?php
2     function my_autoload($clase){
3         require __DIR__.'/clases/'.$clase.'.php';
4     }
5     spl_autoload_register('my_autoload');
6
7     $detalles = new Detalles();
8     $clientes = new Clientes();
9 ?>
```

## 2.8. Namespaces en PHP

En PHP, la palabra clave **use** se utiliza en la definición de espacios de nombres (**namespace**) para importar clases, funciones y constantes desde otros espacios de nombres.

Cuando se utiliza **use** en la definición de un espacio de nombres, se está especificando una ruta corta (alias) para acceder a un elemento específico de otro espacio de nombres. Por ejemplo, si se tiene una clase **MiClase** definida en el espacio de nombres **MiEspacioDeNombres** y se quiere utilizar esa clase en otro espacio de nombres.

### Ejemplo

#### 1. Definir el namespace a las clases

```
1 <?php
2
3 namespace App;
4
5 class Clientes {
6     public function __construct()
7     {
8         echo "Desde la clase de Clientes.php<br>";
9     }
10 }
```

```

1  <?php
2
3  namespace App;
4
5  class Detalles {
6      public function __construct()
7      {
8          echo "Desde la clase de Detalles.php<br>";
9      }
10 }

```

## 2. Importar los namespace a la clase

```

1  <?php
2  use App\Clientes;
3  use App\Detalles;
4  function my_autoload($clase){
5      $fileClass = explode('\\', $clase);
6      require __DIR__ . '/clases/' . $fileClass[1] . '.php';
7  }
8  spl_autoload_register('my_autoload');
9
10 $detalles = new Detalles();
11 $clientes = new Clientes();
12 ?>

```

### Explicación

Este código PHP muestra un ejemplo de autoloading de clases utilizando el método **spl\_autoload\_register**. Aquí hay una explicación línea por línea:

- La línea 1 y 2 utilizan **use** para importar las clases **Clientes** y **Detalles** del namespace **App**. Esto permite utilizar esas clases en el código.
- La función **my\_autoload** se define como una función de carga automática personalizada. Recibe el nombre de la clase que se está intentando cargar.
- Dentro de la función **my\_autoload**, se divide el nombre de la clase en partes utilizando el carácter \ como separador. Esto se hace mediante la función **explode**.
- A continuación, se construye una ruta de archivo utilizando el directorio actual (**\_\_DIR\_\_**) y la carpeta "clases". Se agrega el nombre de la clase obtenido anteriormente (la segunda parte de **\$fileClass**).
- Finalmente, se utiliza la instrucción **require** para incluir el archivo de clase correspondiente.
- La función **spl\_autoload\_register** se utiliza para registrar la función **my\_autoload** como una función de carga automática. Esto significa que cuando se intente utilizar una clase que no se haya cargado previamente, PHP llamará a **my\_autoload** para intentar cargarla.
- Se crean objetos de las clases **Detalles** y **Clientes** utilizando el operador **new**. Estos objetos pueden utilizarse para acceder a los métodos y propiedades definidos en esas clases.

## 2.9. Composer

Composer es un administrador de dependencias para PHP que permite instalar librerías de terceros para facilitar el desarrollo con php. Composer es el equivalente a npm para node JS.

Iniciar Composer en el proyecto.

1. Abrir el proyecto
2. Ejecutar el comando composer init

```
Package name (<vendor>/<name>) [desarrollo/cursophp]:  
Description []: Inicializando composer en el proyecto  
Author [trainingLeader <131613955+trainingLeader@users.noreply.github.com>, n to skip]: Johlver <jjpardo2002@gmail.com>  
Minimum Stability []:  
Package Type (e.g. library, project, metapackage, composer-plugin) []:  
  
Define your dependencies.  
  
Would you like to define your dependencies (require) interactively [yes]? no  
Would you like to define your dev dependencies (require-dev) interactively [yes]? no  
Add PSR-4 autoload mapping? Maps namespace "Desarrollo\Cursophp" to the entered relative path. [src/, n to skip]: no
```

### 2.9.1. Autoload con composer

1. Despues de iniciar composer en el proyecto agregue el siguiente código a el archivo composer.json.

```
1  {  
2      "name": "desarrollo/cursophp",  
3      "description": "Curso php composer",  
4      "authors": [  
5          {  
6              "name": "Johlver",  
7              "email": "jjpardo2002@gmail.com"  
8          }  
9      ],  
10     "require": {},  
11     "autoload": {  
12         "psr-4": {  
13             "App\\": "./clases"  
14         }  
15     }  
16 }  
17 }
```

2. Ejecute el comando composer update

```
desarrollo@DESKTOP-L4V647N MINGW64 /c/apache/htdocs/cursophp
$ composer update ←
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating autoload files
No installed packages - skipping audit.

desarrollo@DESKTOP-L4V647N MINGW64 /c/apache/htdocs/cursophp
$
```

```
1 <?php
2     require 'vendor/autoload.php';
3     use App\Clientes;
4     use App\Detalles;
5
6     $cliente = new Clientes();
7     $detalle = new Detalles();
8 ?>
```

### 3. PHP Avanzado

#### 3.1. Integración de php con Bases de datos relacionales (Mysql)

La integración de bases de datos con PHP es una tarea común en el desarrollo web, ya que PHP es un lenguaje de programación ampliamente utilizado para crear aplicaciones dinámicas y sitios web interactivos. PHP proporciona una variedad de extensiones y funciones para trabajar con diferentes sistemas de gestión de bases de datos relacionales, como MySQL, PostgreSQL, Oracle, entre otros.

##### 3.1.1. Bases de datos relacionales

Las bases de datos relacionales son un tipo de sistema de gestión de bases de datos (SGBD) que organiza la información en tablas estructuradas y establece relaciones entre ellas. Estas bases de datos se basan en el modelo relacional propuesto por Edgar Codd en la década de 1970.

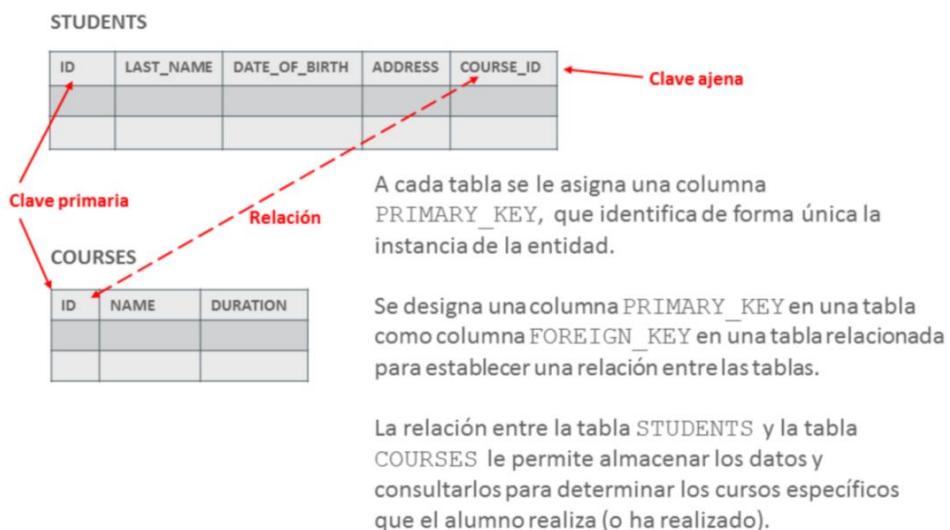
En una base de datos relacional, la información se almacena en tablas compuestas por filas y columnas. Cada tabla representa una entidad o concepto en el dominio del problema, y cada fila de la tabla corresponde a una instancia específica de esa entidad. Las columnas representan atributos o características de la entidad.

Las relaciones entre las tablas se establecen mediante claves primarias y claves foráneas. Una clave primaria es un atributo o conjunto de atributos que identifica de forma única cada fila en una tabla. Una clave foránea es un atributo en una tabla que hace referencia a la clave primaria de otra tabla, estableciendo así una relación entre ellas. Estas relaciones permiten conectar la información entre diferentes tablas y realizar consultas y operaciones complejas.

Las bases de datos relacionales se basan en el lenguaje de consulta estructurado (SQL, por sus siglas en inglés) para interactuar con los datos. SQL proporciona un conjunto de comandos y sentencias para realizar consultas, inserciones, actualizaciones y eliminaciones de datos en la base de datos.

Algunos ejemplos populares de sistemas de gestión de bases de datos relacionales son MySQL, Oracle Database, Microsoft SQL Server y PostgreSQL. Estas bases de datos se utilizan ampliamente en aplicaciones empresariales y sistemas de información, ya que ofrecen una estructura flexible y eficiente para almacenar y recuperar datos relacionados.

Un sistema de gestión de bases de datos relacionales (RDBMS) almacena los datos en tablas. Cada tabla recibe un nombre por parte del usuario que la crea. El usuario suele elegir un nombre relacionado con los datos que se almacenarán en la tabla; por ejemplo, STUDENTS, EMPLOYEES, LOCATIONS. Cuando se crea una tabla, el usuario también crea y nombra las columnas relacionadas con las características específicas que se almacenan para cada registro.

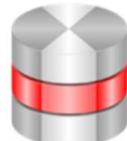


#### Ventajas RDBMS:

- **Menos redundancia:** En un sistema de archivos planos, hay mucha redundancia. Por ejemplo, los nombres de autores se almacenan varias veces.
- **Prevención de inconsistencias:** Si se almacena el mismo fragmento de información en más de un lugar, los cambios en los datos deben realizarse en todos los lugares en los que se almacenen los datos.
- **Eficacia:** Una base de datos suele ser más eficaz que un sistema de archivos planos, debido a que un fragmento de información se almacena en menos ubicaciones.
- **Integridad de los datos:** En un sistema de base de datos, es más fácil mantener la integridad de los datos porque a cada columna se asignan tipos de dato potentes.
- **Confidencialidad:** Es más fácil mantener la confidencialidad de la información si el almacenamiento de los datos está centralizado en una ubicación.

## Reglas para tablas de bases de datos relacionales

- Cada tabla tiene un nombre distinto.
- Cada tabla puede contener varias filas.
- Cada tabla tiene un valor para identificar de forma única las filas.
- Cada columna de una tabla tiene un nombre único.
- Las entradas en las columnas son valores únicos.
- Las entradas en las columnas son del mismo tipo.
- El orden de las filas y las columnas no es importante.



### 3.1.2. Entidades y Atributos

Las entidades son categorías de cosas que son importantes para un negocio y sobre las que se debe conservar información. Las entidades contienen datos e información que el negocio debe conocer y recordar. Estos son algunos ejemplos de entidades:

- **PERSON:** Agent, insured, employee, customer
- **PLACE:** State, country, municipality
- **THING:** Inventory item, vehicle, product
- **CONCEPT:** Policy, risk, coverage, job
- **ORGANIZATION:** Agency, department
- **EVENT:** claim, election

### Tipos de Entidades

| Nombre         | Descripción                               | Ejemplo                      |
|----------------|---|------------------------------|
| Principal      | Existe de forma independiente             | CUSTOMER, INSTRUCTOR         |
| Característica | Existe gracias a otra entidad (principal) | ORDER, CLASS OFFERING        |
| Intersección   | Existe gracias a dos o más entidades      | ORDER ITEM, CLASS ENROLLMENT |

### Entidades e instancias

Las entidades contienen instancias.

Una instancia de entidad es una única incidencia de una entidad.

Las entidades representan un juego de instancias que son de interés para un negocio concreto.

| Entidad      | Instancia                   |
|--------------|-----------------------------|
| PERSON       | John Smith                  |
| PRODUCT      | Uña de cobre de 2,5 x 35 mm |
| PRODUCT TYPE | Uña                         |
| JOB          | Violinista                  |

### 3.1.3. Identificadores Únicos

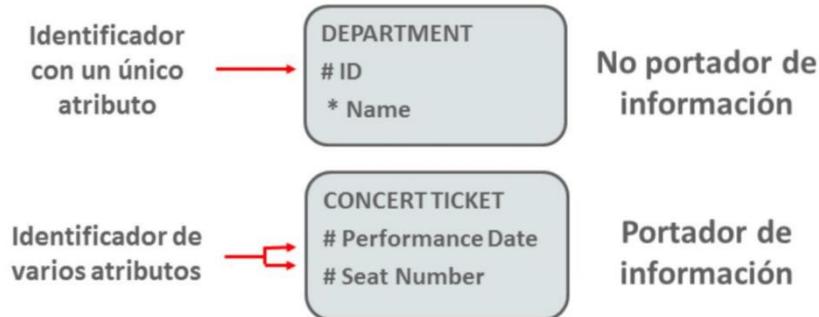
En las bases de datos, un identificador único es un atributo o conjunto de atributos que permite identificar de manera exclusiva cada registro en una tabla. También se le conoce como clave única o clave primaria. Su función principal es garantizar la integridad y la unicidad de los datos almacenados.

A continuación, se listan algunas características de los identificadores únicos:

- Unicidad: Cada valor de un identificador único debe ser único en la tabla. No puede haber duplicados en ese campo o conjunto de campos.
- No nulidad: Un identificador único no puede contener valores nulos. Debe tener un valor definido para cada registro en la tabla.
- Estabilidad: Los valores de los identificadores únicos deben ser estables y no cambiar con el tiempo. Se deben evitar los cambios en los valores de los identificadores únicos siempre que sea posible.
- Indexación: Los identificadores únicos suelen ser utilizados como base para crear índices en la tabla. Esto mejora el rendimiento de las consultas y facilita la búsqueda de registros específicos.
- Referencialidad: Los identificadores únicos pueden utilizarse como referencias en relaciones entre tablas, estableciendo claves foráneas para garantizar la integridad referencial.

Algunos ejemplos comunes de identificadores únicos son:

- Un campo de identificación autoincremental: Un número o valor generado automáticamente para cada registro nuevo en la tabla.
- Una combinación de campos: Dos o más campos que, cuando se combinan, forman un identificador único. Por ejemplo, una tabla de empleados podría tener un identificador único basado en la combinación del número de empleado y el departamento.
- Un valor único generado externamente: Un identificador único proporcionado por una fuente externa, como un número de seguro social o un número de identificación fiscal.



### 3.1.4. Relaciones

Una relación representa las reglas de negocio que enlazan entidades. Cada relación siempre tiene dos reglas de negocio. En el ejemplo de la diapositiva, las reglas de negocio son:

- Un DEPARTMENT puede contener uno o varios EMPLOYEES.
- Un EMPLOYEE debe asignarse a un único DEPARTMENT.

Las relaciones representan una asociación entre dos o más entidades.

La línea de relación del diagrama puede ser sólida (obligatoria) o discontinua (opcional).

Estas líneas terminan en una "única punta" (una instancia) o una "pata de gallo" (una o más instancias).

Las relaciones tienen nombres que ayudan a describir la conexión entre las entidades.

En el diagrama de relaciones, el nombre de la relación, desde cualquier perspectiva, se imprime cerca del punto de inicio de la línea de relación (consulte la diapositiva 5).

- DEPARTMENTS **contains** EMPLOYEES.
- EMPLOYEES **assigned to** DEPARTMENTS.

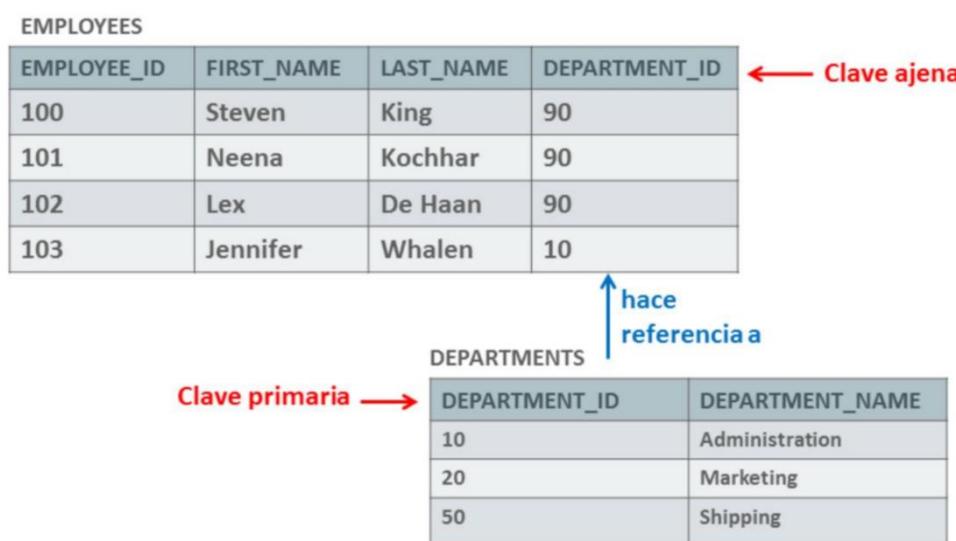
### 3.1.5. Clave ajena (Foránea)

Las claves foráneas, también conocidas como claves externas o claves ajenas (foreign keys en inglés), son atributos o conjuntos de atributos que se utilizan para establecer relaciones entre dos tablas en una base de datos relacional. Estas claves foráneas permiten mantener la integridad referencial y establecer la conexión entre los registros de diferentes tablas.

Cuando una tabla tiene una clave foránea, significa que contiene un atributo (o conjunto de atributos) que hace referencia a la clave primaria de otra tabla. La clave foránea en una tabla actúa como una referencia a una fila específica en otra tabla.

A continuación, se listan algunas características importantes de las claves foráneas:

- Relación entre tablas: Las claves foráneas se utilizan para establecer relaciones entre tablas relacionadas. Estas relaciones pueden ser de uno a uno, uno a muchos o muchos a muchos, y se definen mediante la correspondencia entre la clave foránea y la clave primaria de la tabla referenciada.
- Integridad referencial: Las claves foráneas garantizan la integridad referencial entre las tablas. Esto significa que no se pueden agregar, modificar o eliminar registros en la tabla relacionada de una manera que rompa la relación establecida por la clave foránea.
- Restricciones de integridad: Las claves foráneas pueden tener restricciones asociadas, como la restricción de clave externa (foreign key constraint). Estas restricciones pueden especificar acciones como restringir o eliminar en cascada, que definen cómo se deben manejar las modificaciones o eliminaciones en la tabla referenciada.
- Consultas y operaciones: Las claves foráneas permiten realizar consultas y operaciones que involucran múltiples tablas mediante la combinación de información relacionada. Se pueden utilizar para realizar JOINs y recuperar datos de manera eficiente.
- Mantenimiento de la consistencia: Las claves foráneas aseguran que los datos relacionados se mantengan consistentes en diferentes tablas. Si se actualiza la clave primaria en la tabla referenciada, las claves foráneas en otras tablas también se actualizarán automáticamente para mantener la integridad de los datos.



Componentes de una relación en bases de datos

**Nombre:** Etiqueta que aparece junto a la entidad a la que está asignada. Asegúrese de que todos los nombres de relación estén en minúsculas.

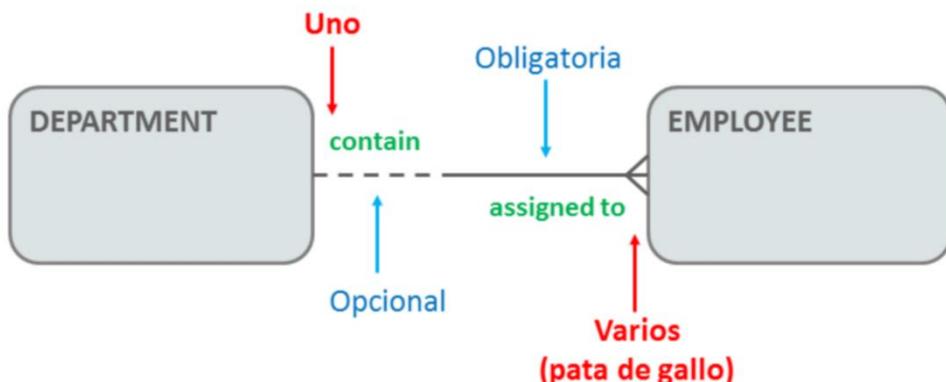
**Cardinalidad:** Número mínimo y máximo de los valores de la relación.

- Un único registro coincidente
- Uno o más registros coincidentes

**Opcionalidad:** Si la relación debe existir.

- Opcional (cero registros coincidentes)
- Obligatorio (al menos un registro coincidente en cada entidad)

- **Nombre**
- **Cardinalidad**
- **Opcionalidad**



### 3.1.6. Normalización de bases de datos

La normalización de bases de datos es un proceso de diseño que se utiliza para organizar y estructurar las tablas de una base de datos relacional de manera eficiente y libre de redundancias. El objetivo principal de la normalización es eliminar la duplicación de datos y garantizar la integridad y consistencia de la información almacenada.

El proceso de normalización se basa en una serie de reglas o formas normales (normal form) establecidas por Edgar Codd, el creador del modelo relacional. Estas formas normales se dividen en diferentes niveles, desde la primera forma normal (1NF) hasta la quinta forma normal (5NF), y cada nivel aborda un aspecto específico de la normalización.

A continuación, se resumen las tres formas normales más comunes:

- Primera Forma Normal (1NF):
  - Elimine los grupos repetidos de las tablas individuales.
  - Cree una tabla independiente para cada conjunto de datos relacionados.
  - Identifique cada conjunto de datos relacionados con una clave principal.
- Segunda Forma Normal (2NF):

- Cree tablas independientes para conjuntos de valores que se apliquen a varios registros.
- Relacione estas tablas con una clave externa.
- Tercera Forma Normal (3NF):
  - Elimine los campos que no dependan de la clave.

Normalización de una tabla de ejemplo: <https://learn.microsoft.com/es-es/office/troubleshoot/access/database-normalization-description>

1. Tabla sin normalizar:

| Nº alumno | Tutor  | Despacho-Tut | Clase1 | Clase2 | Clase3 |
|-----------|--------|--------------|--------|--------|--------|
| 1022      | García | 412          | 101-07 | 143-01 | 159-02 |
| 4123      | Díaz   | 216          | 101-07 | 143-01 | 179-04 |

2. Primera forma normal: sin grupos repetidos

Las tablas sólo deben tener dos dimensiones. Puesto que un alumno tiene varias clases, estas clases deben aparecer en una tabla independiente. Los campos Clase1, Clase2 y Clase3 de los registros anteriores son indicativos de un problema de diseño.

Las hojas de cálculo suelen usar la tercera dimensión, pero las tablas no deberían hacerlo. Otra forma de considerar ese problema es con una relación de uno a varios y poner el lado de uno y el lado de varios en tablas distintas. En su lugar, cree otra tabla en la primera forma normal eliminando el grupo repetido (Nº clase), según se muestra a continuación:

| Nº alumno | Tutor  | Despacho-Tut | Nº clase |
|-----------|--------|--------------|----------|
| 1022      | García | 412          | 101-07   |
| 1022      | García | 412          | 143-01   |
| 1022      | García | 412          | 159-02   |
| 4123      | Díaz   | 216          | 101-07   |
| 4123      | Díaz   | 216          | 143-01   |
| 4123      | Díaz   | 216          | 179-04   |

### 3. Segunda forma normal: eliminar los datos redundantes

Observe los diversos valores de N° clase para cada valor de N° alumno en la tabla anterior. N° clase no depende funcionalmente de N° alumno (la clave principal), de modo que la relación no cumple la segunda forma normal.

Las tablas siguientes demuestran la segunda forma normal:

Alumnos:

| Nº alumno | Tutor  | Despacho-Tut |
|-----------|--------|--------------|
| 1022      | García | 412          |
| 4123      | Díaz   | 216          |

Registro:

| Nº alumno | Nº clase |
|-----------|----------|
| 1022      | 101-07   |
| 1022      | 143-01   |
| 1022      | 159-02   |
| 4123      | 101-07   |
| 4123      | 143-01   |
| 4123      | 179-04   |

### 4. Tercera forma normal: eliminar los datos que no dependen de la clave

En el último ejemplo, Despacho-Tut (el número de despacho del tutor) es funcionalmente dependiente del atributo Tutor. La solución es pasar ese atributo de la tabla Alumnos a la tabla Personal, según se muestra a continuación:

Alumnos:

| Nº alumno | Tutor  |
|-----------|--------|
| 1022      | García |
| 4123      | Díaz   |

Personal:

| Nombre | Sala | Dept. |
|--------|------|-------|
| García | 412  | 42    |
| Díaz   | 216  | 42    |

## 3.2. Mysql



MySQL es un sistema manejador de bases de datos de libre uso y distribución bajo licencia GPL de los más utilizados y que está disponible para varios sistemas operativos (DUBOIS, 2009).

Su popularidad se debe principalmente a su licencia libre y a su facilidad de uso y administración. Por otra parte, ha sido integrada con otras herramientas libres como son Linux, Apache, PHP, entre otras. Esta combinación e integración de tecnologías dio nombre a la plataforma de desarrollo conocida como LAMP (Linux, Apache, MySQL y PHP).

### 3.2.1. Características MySQL



#### Velocidad

MySQL es veloz comparado con la mayoría de las bases libres.



#### Portabilidad

MySQL corre en muchos sistemas operativos entre ellos windows, linux, unix.



#### Facilidad de uso

MySQL es de alto desempeño pero a la vez fácil de usar.



#### Conectividad

MySQL soporta distintos esquemas de conectividad y las bases de datos pueden ser accedidas desde cualquier sitio de Internet.



#### Soporta el lenguaje SQL

MySQL soporta el lenguaje estructurado de consultas (SQL).



#### Seguridad

MySQL maneja esquemas de seguridad que permiten asignar permisos a nivel de usuario.

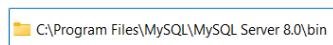


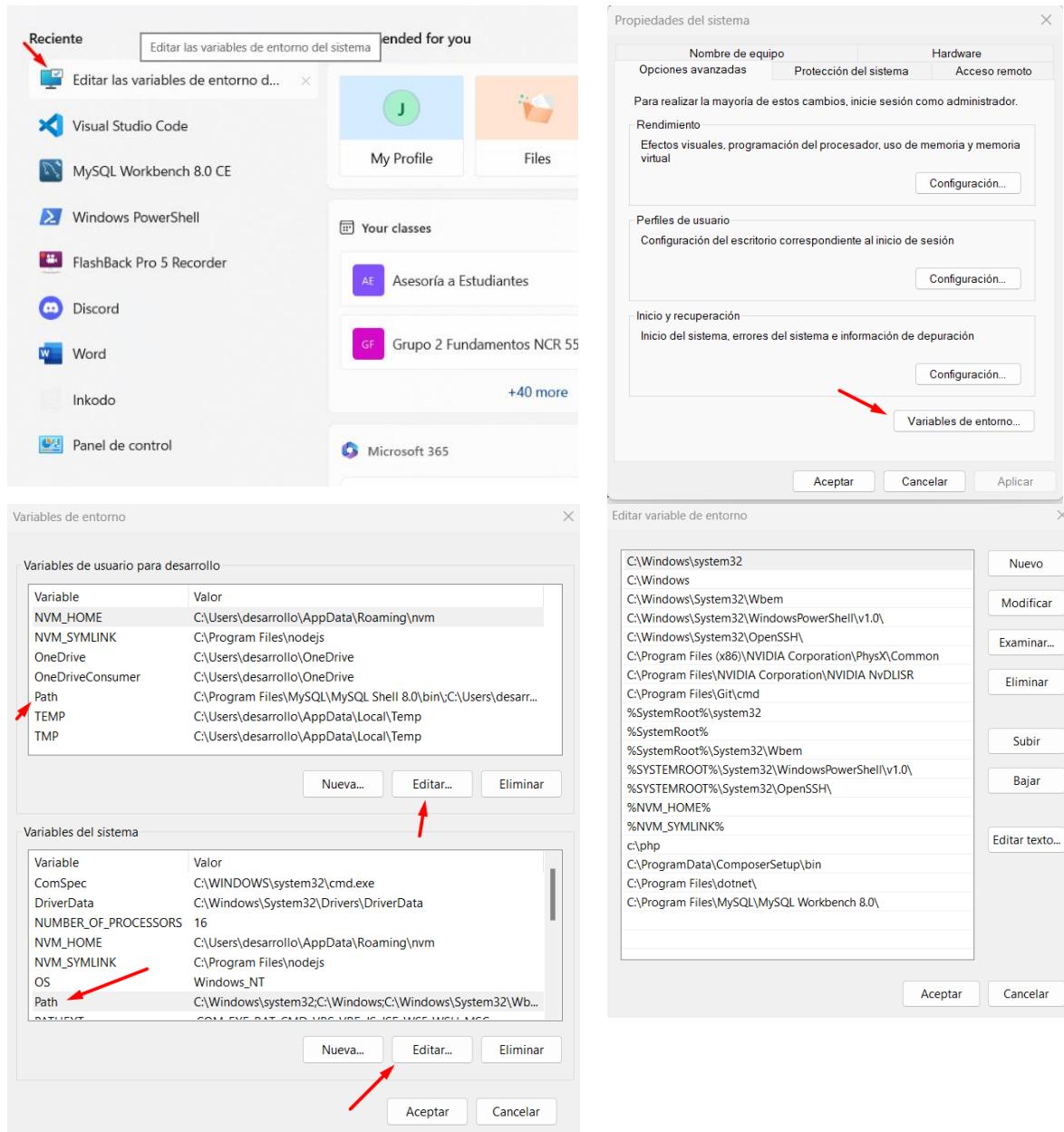
#### Robustez

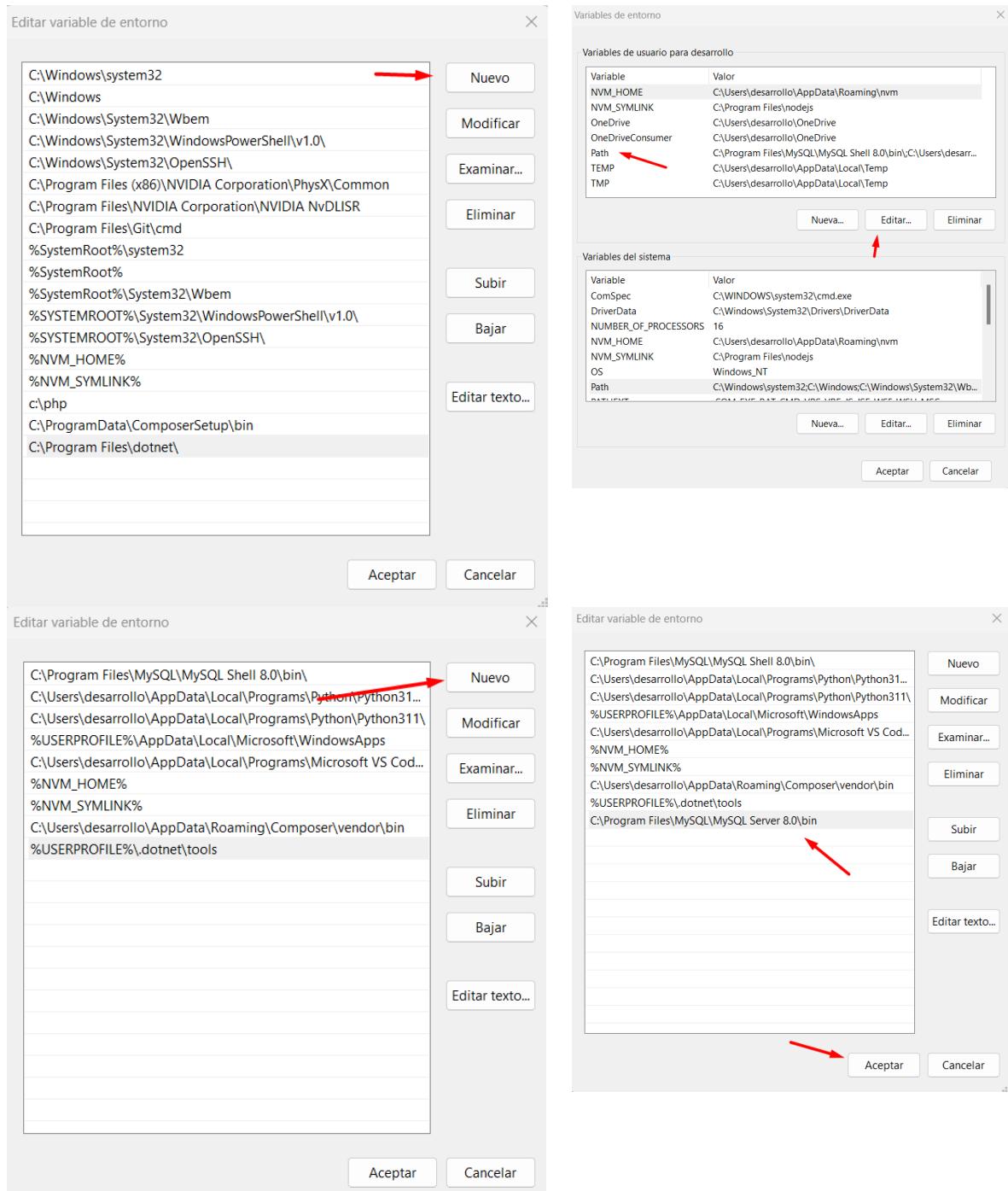
El servidor MySQL es multi-hilo y puede atender varios usuarios de manera simultánea.

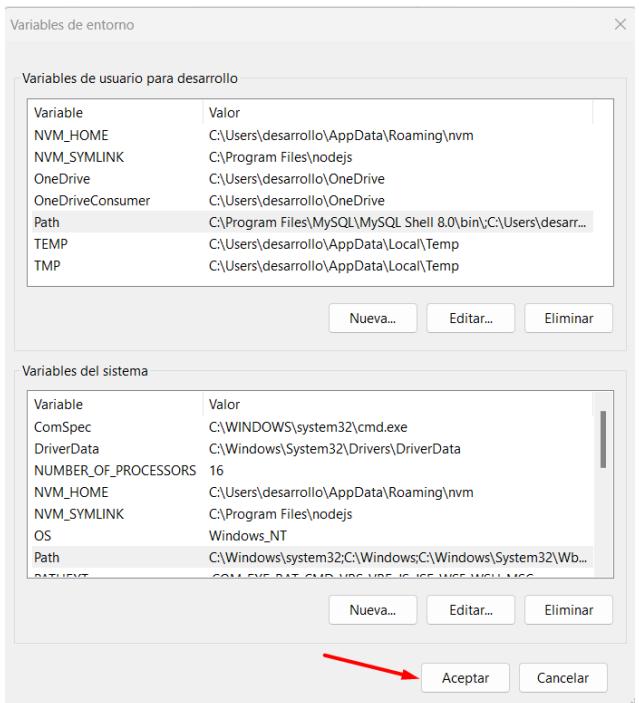
### 3.2.2. Consola de MySQL

Para acceder a la administración de bases de datos en MySQL desde la terminal del sistema operativo siga las siguientes instrucciones:

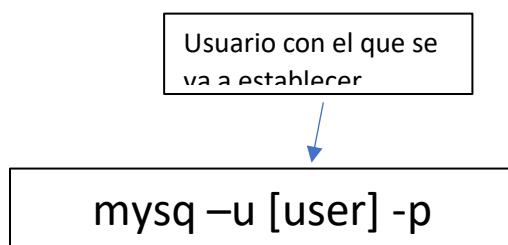
1. Abra el terminal de su sistema operativo
2. Para windows si se desea usar el powershell se debe configurar previamente el entorno de variables del sistema operativo y redireccionar a la carpeta donde se encuentra instalado MySQL Server. En mi caso la carpeta de instalación es: 







Ejecute el comando mysql -u [user] -p para iniciar sesión



Cuando ingrese las credenciales de acceso podrá observar la siguiente respuesta en la cual se puede visualizar el prompt de mysql.

A screenshot of a Windows PowerShell window titled 'Windows PowerShell'. The command entered is 'PS C:\Users\desarrollo> mysql -u root -p'. The response shows the MySQL monitor welcome message, including the connection id (27), server version (8.0.33), and copyright information. The prompt 'mysql>' is visible at the bottom, with a red arrow pointing to it.

```

PS C:\Users\desarrollo> mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 27
Server version: 8.0.33 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> |

```

### 3.2.3. Tipos de datos en MySQL

### 3.2.3.1 Tipos de datos numéricos

*Tipos de datos enteros.*

| TIPO                  | Bytes   | Valor Mínimo<br>(Con signo/Sin signo) | Valor Máximo<br>(Con signo/Sin signo) |
|-----------------------|---|---------------------------------------|---------------------------------------|
| TINYINT               | 1   | -128                                  | 127                                   |
|                       |   | 0                                     | 255                                   |
| BIT<br>(BOOL,BOOLEAN) | Número entero con valor 0 o 1. Sinónimo de TINYINT(1) |                                       |                                       |
| SMALLINT              | 2   | -32768                                | 32767                                 |
|                       |   | 0                                     | 65535                                 |
| MEDIUMINT             | 3   | -8388608                              | 8388607                               |
|                       |   | 0                                     | 16777215                              |
| INT                   | 4   | -2147483648                           | 2147483647                            |
|                       |   | 0                                     | 4294967295                            |
| BIGINT                | 8   | -9223372036854775808                  | 9223372036854775807                   |
|                       |   | 0                                     | 18446744073709551615                  |

*Tipos de datos en coma flotante.*

| Tipo             | Tamaño  |
|------------------|---|
| FLOAT (m,d)      | Contiene un número en coma flotante de precisión sencilla.<br>El valor M es la anchura a mostrar y D es el número de decimales.   |
| DOUBLE (m,d)     | Contiene un número en coma flotante de precisión doble. Igual que FLOAT la diferencia es el rango de valores posibles.  |
| DECIMAL (m [,d]) | Se usan para guardar valores para los que es importante preservar una precisión exacta, por ejemplo con datos monetarios.<br>Ejemplo: salario DECIMAL(5,2)<br>Si se omite D el valor por defecto es 0, los valores no tendrán punto decimal ni decimales. |

### 3.2.3.2. Tipos de datos de carácter

| Tipo                     | Tamaño   | Sintaxis                      |
|--------------------------|--|-------------------------------|
| CHAR (M)                 | Los valores válidos para M son de 0 a 255 caracteres.<br>Contiene una cadena de longitud constante.<br>Para mantener la longitud de la cadena, se rellena a la derecha con espacios. Estos espacios se eliminan al recuperar el valor. | PacIdentificacion<br>CHAR(10) |
| VARCHAR (M)              | Los valores válidos para M son de 0 a 255 caracteres.<br>Contiene una cadena de longitud variable.<br>Los espacios al final se eliminan.   | PacNombres<br>VARCHAR(50)     |
| BLOB                     | Una longitud máxima de 65.535 caracteres.<br>Válido para objetos binarios como imágenes, ficheros de texto, audio o video.   | PacImagenFoto BLOB            |
| TEXT                     | Una longitud máxima de 65.535 caracteres.<br>Sirve para almacenar texto plano sin formato.<br>Distingue entre minúsculas y mayúsculas.   | PacDescripcion TEXT           |
| TINYBLOB<br>TINYTEXT     | Longitud máxima de 255 caracteres.   |                               |
| MEDIUMBLOB<br>MEDIUMTEXT | Longitud máxima de 16777215 caracteres   |                               |
| LONGBLOB<br>LONGTEXT     | Longitud máxima de 4294967298 caracteres.  |                               |

### 3.2.3.3. Tipos de dato fecha

| TIPO      | RANGO  | FORMATO             |
|-----------|--|---------------------|
| DATE      | Válido para almacenar una fecha con año, mes y día.<br>Su rango oscila entre: '1000-01-01' y '9999-12-31'. | AAAA-MM-DD          |
| DATETIME  | Almacena una fecha y una hora.<br>Su rango oscila entre '1000-01-01 00:00:00' y '9999-12-31 23:59:59'.     | AAAA-MM-DD HH:MM:SS |
| TIME      | Una hora.<br>El rango está entre '-838:59:59' y '838:59:59'.   | HH:MM:SS            |
| TIMESTAMP | Almacena una fecha y hora UTC.<br>El rango está entre '1970-01-01 00:00:00' y algún momento del año 2037.  | AAAA-MM-DD HH:MM:SS |

### 3.2.3.4. Modificadores o Constraints

| MODIFICADOR    | USO  | TIPO DE CAMPO QUE APLICA  |
|----------------|--|---------------------------|
| AUTO_INCREMENT | El valor se va incrementando automáticamente en cada registro (1,2,3,etc).     | Enteros                   |
| DEFAULT        | Coloca un valor por defecto (el valor se coloca justo detrás de esta palabra). | Todos excepto TEXT y BLOB |
| NOT NULL       | Impide que un campo sea nulo.  | Todos                     |
| PRIMARY KEY    | Hace que el campo se considere llave primaria.                                 | Todos                     |
| UNIQUE         | 65535 bytes. Evita la repetición de valores.                                   | Todos                     |

### 3.3. SQL Structured Query Language

SQL (Structured Query Language) es un lenguaje de programación utilizado para administrar y manipular bases de datos relacionales. Fue desarrollado en la década de 1970 y se ha convertido en el estándar de facto para trabajar con bases de datos.

SQL permite a los usuarios crear, modificar y eliminar bases de datos, así como realizar consultas para recuperar información específica de una base de datos. Con SQL, puedes crear tablas para almacenar datos, definir relaciones entre las tablas, agregar, actualizar o eliminar registros y realizar consultas complejas para obtener información precisa.

El lenguaje SQL consta de varios comandos, entre los que se incluyen:

- DDL (Data Definition Language): Utilizado para definir y modificar la estructura de la base de datos. Incluye comandos como CREATE, ALTER y DROP para crear, modificar y eliminar tablas, índices, vistas, etc.
- DML (Data Manipulation Language): Utilizado para manipular los datos almacenados en la base de datos. Incluye comandos como INSERT, UPDATE y DELETE para agregar, actualizar y eliminar registros.
- DQL (Data Query Language): Utilizado para realizar consultas y recuperar información de la base de datos. El comando más común es SELECT, que permite especificar los criterios de búsqueda y los campos a recuperar.
- DCL (Data Control Language): Utilizado para controlar los privilegios de acceso a la base de datos. Incluye comandos como GRANT y REVOKE para otorgar y revocar permisos.

#### 3.3.1. Que se puede hacer con SQL

Con SQL, puedes realizar una amplia gama de tareas relacionadas con la administración y manipulación de bases de datos. Aquí hay algunas cosas que puedes hacer con SQL:

- Crear y administrar bases de datos: Puedes utilizar SQL para crear nuevas bases de datos y administrar su estructura. Esto incluye crear tablas, definir columnas, establecer restricciones de integridad, crear índices y definir relaciones entre tablas.
- Insertar, actualizar y eliminar datos: Puedes utilizar comandos SQL como INSERT, UPDATE y DELETE para agregar, modificar y eliminar registros en las tablas de la base de datos. Esto te permite mantener y actualizar los datos almacenados.
- Consultar datos: SQL es especialmente útil para realizar consultas complejas y recuperar datos específicos de una base de datos. Puedes utilizar el comando SELECT para especificar los criterios de búsqueda, filtrar los resultados, ordenar los datos y combinar información de varias tablas mediante joins.
- Filtrar y ordenar datos: SQL te permite filtrar los resultados de las consultas utilizando condiciones en la cláusula WHERE. También puedes ordenar los resultados utilizando la cláusula ORDER BY, lo que te permite obtener los datos en el orden deseado.
- Agregar funciones y cálculos: SQL ofrece varias funciones integradas, como SUM, AVG, COUNT, MAX, MIN, entre otras, que te permiten realizar cálculos y resúmenes de los datos almacenados en la base de datos. Puedes utilizar estas funciones en las consultas para obtener resultados agregados o realizar operaciones matemáticas.

- Crear vistas: Las vistas son consultas almacenadas que se pueden utilizar como tablas virtuales. Puedes crear vistas en SQL para simplificar consultas complejas, reutilizar consultas comunes o presentar una vista específica de los datos sin exponer toda la estructura de la base de datos.
- Establecer restricciones de integridad: SQL te permite establecer restricciones de integridad en las tablas para garantizar la consistencia y la calidad de los datos. Puedes definir restricciones como claves primarias, claves foráneas, restricciones de unicidad y restricciones de verificación.

### 3.3.2. Comandos DDL

Los comandos DDL (Data Definition Language) en SQL se utilizan para definir, modificar y eliminar la estructura de la base de datos. Estos comandos permiten crear tablas, definir restricciones, modificar la estructura de las tablas existentes y eliminar objetos de la base de datos.

#### 3.3.2.1 SHOW DATABASE

Comando permite visualizar las bases de datos que se encuentran creadas en el servidor de bases de datos.

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sakila |
| sys |
| world |
+-----+
6 rows in set (0.02 sec)
```

#### 3.3.2.2 CREATE DATABASE

Permite crear una base de datos en el servidor MySQL.

Sintaxis: **CREATE DATABASE [NOMBRE DE LA BD];**

```
mysql> CREATE DATABASE mitienda;
Query OK, 1 row affected (0.01 sec)

mysql> |
```

#### 3.3.2.3. DROP DATABASE

Permite eliminar una base de datos del servidor MySQL.

Sintaxis: **DROP DATABASE [Base de datos];**

```
mysql> DROP DATABASE mitienda; ←
Query OK, 0 rows affected (0.03 sec)

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sakila |
| sys |
| world |
+-----+
6 rows in set (0.00 sec)

mysql>
```

### 3.3.2.4. USE

Permite seleccionar una base de datos para administrar cada uno de los objetos de la base de datos.

Sintaxis: **USE [Base de datos];**

```
mysql> USE mitienda; ←
Database changed
mysql> |
```

### 3.3.2.5. CREATE TABLE

Permite crear una tabla en la base de datos seleccionada con el comando USE.

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
    ....
```

Sintaxis : );

Ejercicio:

Crear una tabla que permita almacenar la siguiente información de los clientes de la tienda:

- Identificación del cliente
- Primer nombre
- Segundo nombre
- Primer apellido
- Segundo apellido
- Email
- Fecha de nacimiento

```

mysql> CREATE TABLE cliente(
    -> idCliente      varchar(15),
    -> primerNombre   varchar(25),
    -> segundoNombre  varchar(25),
    -> primerApellido varchar(25),
    -> segundoApellido varchar(25)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> |

```

Para visualizar las tablas que se encuentran creadas en la base de datos puede utilizar el comando **SHOW TABLES;**

```

mysql> SHOW TABLES;
+-----+
| Tables_in_mitienda |
+-----+
| cliente           |
+-----+
1 row in set (0.00 sec)

mysql>

```

Para visualizar la estructura de una tabla creada en la base de datos puede utilizar el comando **DESCRIBE nombre tabla;**

```

mysql> DESCRIBE cliente;
+-----+-----+-----+-----+-----+
| Field        | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| idCliente    | varchar(15) | YES  |     | NULL    |       |
| primerNombre | varchar(25) | YES  |     | NULL    |       |
| segundoNombre| varchar(25) | YES  |     | NULL    |       |
| primerApellido| varchar(25) | YES  |     | NULL    |       |
| segundoApellido| varchar(25) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)

```

### 3.3.2.6 ALTER TABLE

Permite modificar la estructura de una tabla existente en la base de datos.

#### 3.3.2.7. Agregar columnas

`ALTER TABLE table_name`

Sintaxis. `ADD column_name datatype;`

Teniendo en cuenta el ejercicio anterior note que hicieron falta crear dos columnas una el email y otra la fecha de nacimiento. En este caso utilizaremos alter table para realizar la creación de estas columnas faltantes.

```

mysql> ALTER TABLE cliente ADD emailCliente varchar(100);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> |

```

Ahora que se ha agregado la columna email agregue la columna fecha nacimiento recuerde que es de tipo fecha. [Ver tipos de dato fecha](#).

```
mysql> ALTER TABLE cliente ADD fechaNacimiento date;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESCRIBE cliente;
+-----+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idCliente  | varchar(15) | YES  |     | NULL    |       |
| primerNombre | varchar(25) | YES  |     | NULL    |       |
| segundoNombre | varchar(25) | YES  |     | NULL    |       |
| primerApellido | varchar(25) | YES  |     | NULL    |       |
| segundoApellido | varchar(25) | YES  |     | NULL    |       |
| emailCliente | varchar(100) | YES  |     | NULL    |       |
| fechaNacimiento | date | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

### 3.3.2.8. Eliminar columnas

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Sintaxis:

### 3.3.2.9. Renombrar una columna

```
ALTER TABLE table_name
RENAME COLUMN old_name TO new_name;
```

Sintaxis:

### 3.3.2.10. Modificar el tipo de dato

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

Sintaxis:

### 3.3.2.11 CONSTRAINTS

En MySQL, los constraints (restricciones) son reglas que se aplican a las columnas de una tabla para mantener la integridad de los datos y asegurar que se cumplan ciertas condiciones. Estas restricciones definen reglas y limitaciones en los valores que se pueden insertar, actualizar o eliminar en una tabla. A continuación, se mencionan los tipos de constraints más comunes en MySQL:

- PRIMARY KEY: Es una restricción que se utiliza para identificar de manera única cada fila en una tabla. Solo puede haber una primary key por tabla y no puede tener valores nulos. Cuando se define una primary key, se crea automáticamente un índice para acelerar las búsquedas por clave primaria.
- FOREIGN KEY: Es una restricción que establece una relación entre dos tablas, utilizando una columna o un conjunto de columnas en una tabla (la foreign key) que hace referencia a la primary key de otra tabla (la tabla referenciada). Ayuda a mantener la integridad referencial y garantiza que los valores en la columna de la foreign key coincidan con los valores en la primary key de la tabla referenciada.

- UNIQUE: Es una restricción que garantiza que los valores en una columna o conjunto de columnas sean únicos en la tabla. A diferencia de la primary key, puede haber múltiples restricciones UNIQUE en una tabla y las columnas pueden tener valores nulos (pero no múltiples valores nulos).
- NOT NULL: Es una restricción que indica que una columna no puede contener valores nulos. Es decir, se requiere que se ingrese un valor en esa columna para cada fila.
- CHECK: Es una restricción que permite definir condiciones específicas para los valores de una columna. Solo se permite insertar o actualizar registros que cumplan con la condición definida.

### 3.3.2.12. PRIMARY KEY

Haga clic en [Ver definición](#) para ver el concepto.

```
mysql> DESCRIBE cliente;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| idCliente | varchar(15) | YES | | NULL |
| primerNombre | varchar(25) | YES | | NULL |
| segundoNombre | varchar(25) | YES | | NULL |
| primerApellido | varchar(25) | YES | | NULL |
| segundoApellido | varchar(25) | YES | | NULL |
| emailCliente | varchar(100) | YES | | NULL |
| fechaNacimiento | date | YES | | NULL |
+-----+-----+-----+-----+-----+
```

Como se puede observar en la imagen la tabla cliente no tiene una llave primaria definida lo cual atenta con la integridad de la información. Si ya tenemos una tabla creada podemos agregar este constraint de la siguiente forma:

Sintaxis:  
`ALTER TABLE Persons  
ADD CONSTRAINT PK_Person PRIMARY KEY (ID,LastName);`

```
mysql> ALTER TABLE cliente  
    -> ADD CONSTRAINT PK_Cliente PRIMARY KEY (idCliente);  
Query OK, 0 rows affected (0.04 sec)  
Records: 0  Duplicates: 0  Warnings: 0  
  
mysql>
```

```
mysql> DESCRIBE cliente;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| idCliente | varchar(15) | NO | PRI | NULL |  
| primerNombre | varchar(25) | YES | | NULL |  
| segundoNombre | varchar(25) | YES | | NULL |  
| primerApellido | varchar(25) | YES | | NULL |  
| segundoApellido | varchar(25) | YES | | NULL |  
| emailCliente | varchar(100) | YES | | NULL |  
| fechaNacimiento | date | YES | | NULL |
+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

Ejercicio.

La empresa mitienda desea expandir su negocio a otras ciudades del país y le solicita a usted como diseñador de bases de datos que realice los cambios pertinentes para cumplir este requerimiento de la gerencia.

### Tareas a realizar:

- Cree las tablas necesarias que permitan almacenar las ciudades, departamentos de Colombia. Considere una posible expansión internacional.
- Establezca relación entre los clientes y la ubicación geográfica.

En las tablas de bases de datos se pueden definir columnas autoincrem  ntales que permiten generar de forma autom  tica un consecutivo. A continuaci  n, veremos la sintaxis a tener en cuenta para la definici  n de campos autoincrem  ntales.

```
mysql> CREATE TABLE pais(
    -> idPais int NOT NULL AUTO_INCREMENT,
    -> nombrePais varchar(50) NOT NULL,
    -> CONSTRAINT pk_pais PRIMARY KEY (idPais)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE departamento(
    -> idDep int NOT NULL AUTO_INCREMENT,
    -> nombreDep varchar(50) NOT NULL,
    -> idPais int(11),
    -> CONSTRAINT pk_departamento PRIMARY KEY (idDep),
    ->           CONSTRAINT fk_PaisDep FOREIGN KEY (idPais) REFERENCES pais(idPais)
    -> );
Query OK, 0 rows affected, 1 warning (0.01 sec)

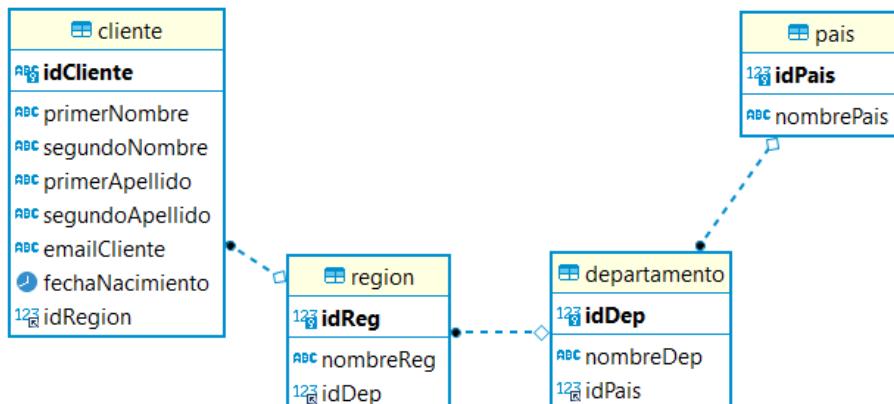
mysql> CREATE TABLE region(
    -> idReg int NOT NULL AUTO_INCREMENT,
    -> nombreReg varchar(50) NOT NULL,
    -> idDep int(11),
    -> CONSTRAINT pk_region PRIMARY KEY (idReg),
    ->           CONSTRAINT fk_DepRegion FOREIGN KEY (idDep) REFERENCES departamento(idDep)
    -> );
Query OK, 0 rows affected, 1 warning (0.01 sec)
```

```
mysql> ALTER TABLE cliente ADD idRegion int(11); ←
Query OK, 0 rows affected, 1 warning (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 1

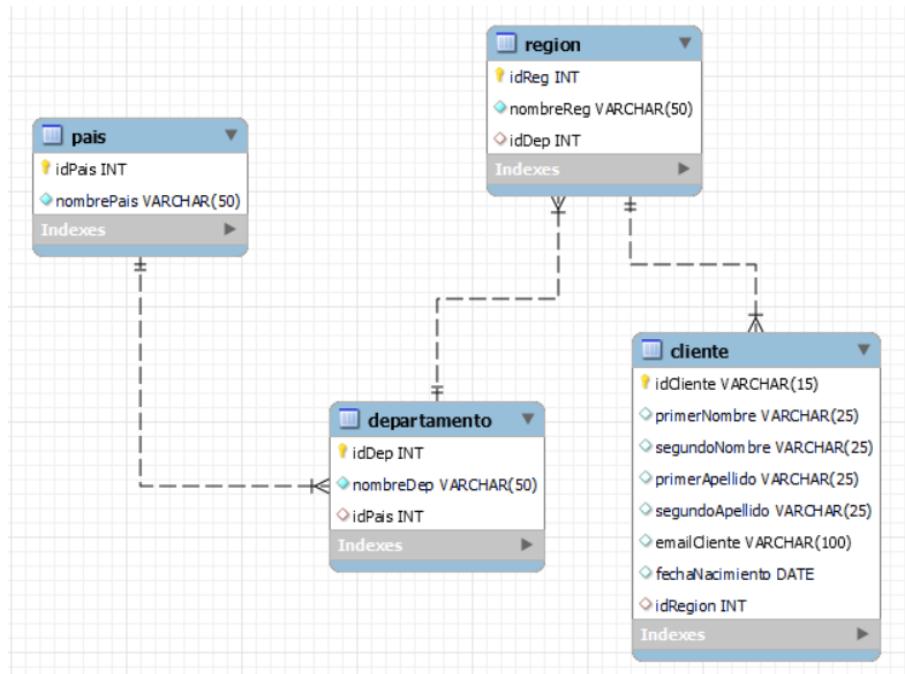
mysql> ALTER TABLE CLIENTE ADD CONSTRAINT FK_ClienteReg FOREIGN KEY (idRegion) REFERENCES region(idReg);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> |
```

### DER GENERADO POR DBeaver



### DER GENERADO POR Workbench

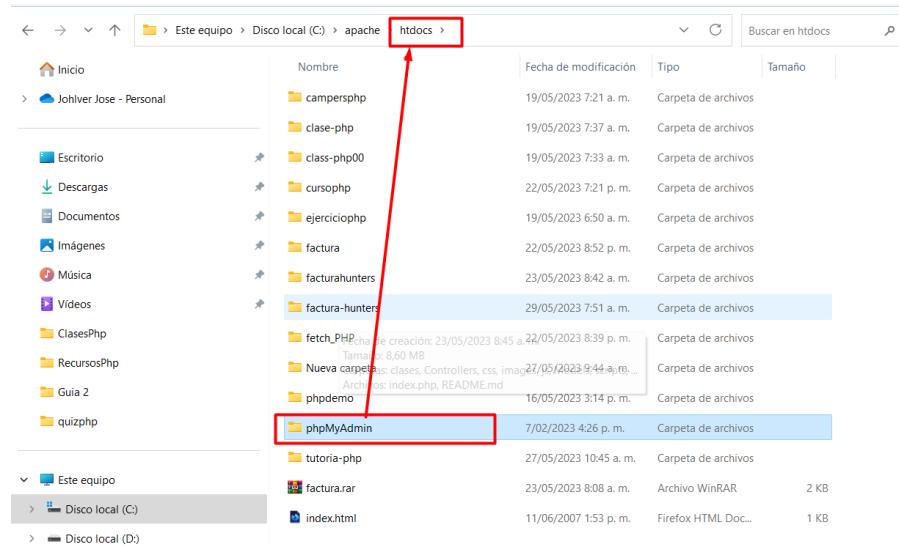


Existe otra herramienta muy poderosa y gratuita para trabajar con MySQL que se llama phpMyAdmin y a continuación veremos como configurarla. Siga los siguientes pasos:

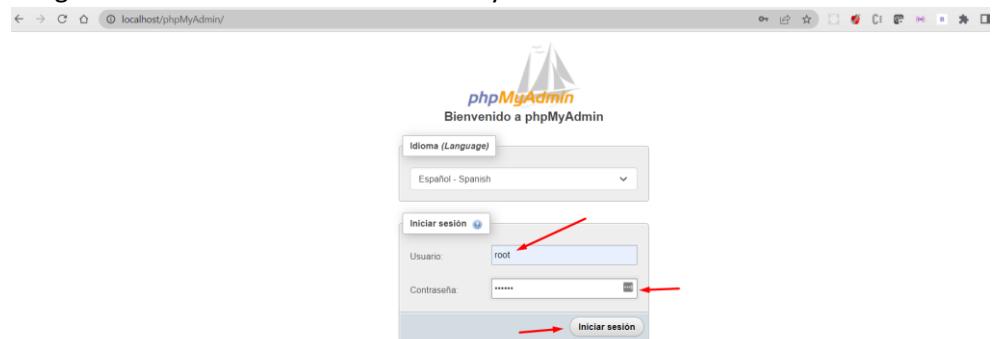
- Ingrese a la url <https://www.phpmyadmin.net/>

- Descomprima el archivo descargado y cámbiele el nombre a phpMyAdmin

- Copie la carpeta en el DocumentRoot del servidor apache. Recuerde que el DocumentRoot es la carpeta principal donde se desplegaran todos los proyectos a nivel local.



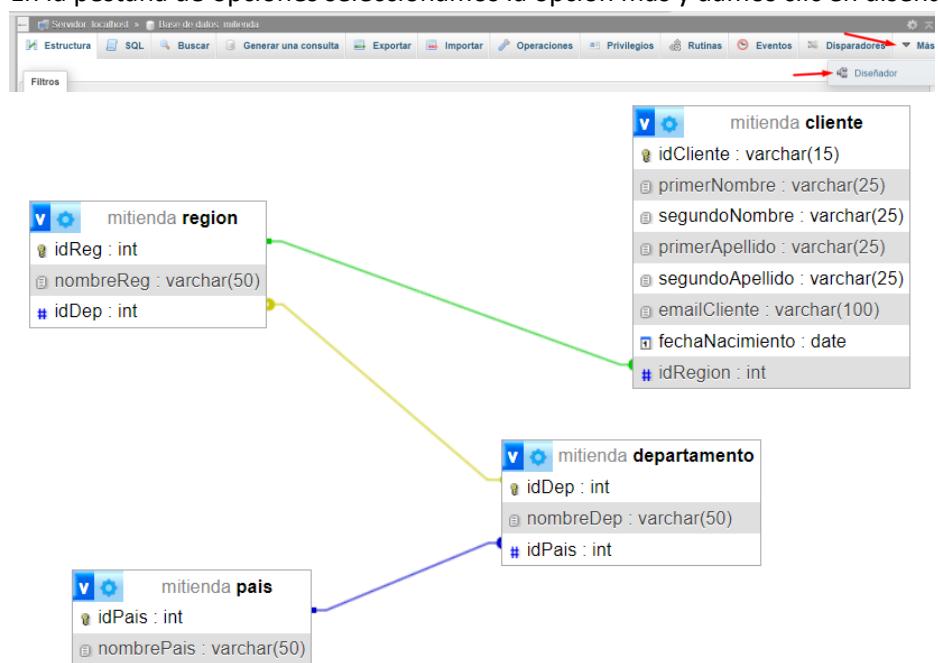
- Abra el navegador de su preferencia e ingrese la url <http://localhost/phpMyAdmin/>. E ingrese las credenciales de acceso a MySQL.



Si todo salió correctamente tendremos el siguiente resultado:

The screenshot shows the phpMyAdmin configuration page. It includes sections for 'Configuraciones generales' (General configurations), 'Servidor de base de datos' (Database server), 'Configuraciones de apariencia' (Appearance configurations), and 'phpMyAdmin' (phpMyAdmin version information). The 'Servidor de base de datos' section provides details about the MySQL server, including its host (localhost via TCP/IP), port (3306), and version (8.0.33). The 'phpMyAdmin' section shows the version (5.2.1), documentation, and download links.

En la pestaña de opciones seleccionamos la opción mas y damos clic en diseñador.



### Ejercicio.

Campus Bucaramanga desea realizar un programa que permita llevar el registro de todos sus colaboradores incluyendo los campers que se encuentran interesados en participar en el programa intensivo de entrenamiento en desarrollo de software. En la actualidad campus cuenta entre sus colaboradores profesionales en el área de marketing digital, diseñadores web, coordinadores académicos, trainers, personal del área administrativa y locativa (Personal de mantenimiento y personal de limpieza). El área de entrenamiento en campus cuenta con diferentes rutas de entrenamiento las cuales tienen asociados diferentes stack tecnológico como (Python, JavaScript, HTML, CSS, PHP, NodeJs, NetCore, Vue, React, Angular entre otros). Cada ruta de entrenamiento está compuesta por Unidades temáticas y las unidades temáticas contienen capítulo y los capítulos contienen ejes temáticos y los ejes temáticos posee módulos y los módulos poseen un contenido que los campers deben cumplir en su entrenamiento. El sistema debe permitir que el personal del área académica estructure las diferentes rutas de entrenamiento. Los campers que se registren en la plataforma y sean seleccionados para iniciar el programa deben configurar la ruta de entrenamiento

de acuerdo a los módulos que se han previamente por los administradores académicos; durante el proceso de selección los campers son ubicados en salones de acuerdo a su conocimiento; en apolo se ubicaran aquellos campers que poseen conocimientos básicos en programación, en artemis estarán los campers que cuentan con un conocimiento medio en programación y en sputnik estarán los campers que cuentan con altos conocimientos en programación. Los trainers que son los responsables de potenciar a cada uno de los campers estarán asignados a cada uno de los respectivos salones, tenga en cuenta que los trainers podrán estar en diferente salón dependiendo de la franja de entrenamiento asignada (6am a 9:30am, 10:15am 13:30, 14:00pm a 17:45, 18:00pm a 21:45 pm)

Tareas:

- Identifique las posibles entidades y sus atributos.
- Establezca la relación entre las diferentes entidades identificadas.
- Genere el DER de la base de datos diseñada
- 

### 3.3.3. Comandos DML

Los comandos DML en el lenguaje SQL se encarga de la manipulación de los registros en las bases de datos.

#### 3.3.3.1. INSERT INTO

El comando INSERT permite agregar nuevos registros a la tabla seleccionada en la consulta SQL.

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Sintaxis:

Ejercicio: En siguiente ejercicio se usará el comando insert into para registrar 5 países en la tabla país.

1. Revisar la estructura de la tabla para verificar el nombre de las columnas.

```
mysql> DESCRIBE pais;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| idPais | int | NO | PRI | NULL | auto_increment |
| nombrePais | varchar(50) | NO | | NULL | |
+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)
```

2. Construir la sentencia que permita ingresar un registro.

```
mysql> INSERT INTO pais(nombrePais) VALUES ('Colombia');
Query OK, 1 row affected (0.01 sec)
```

Si se desea agregar varios registro a la tabla se puede realizar de la siguiente forma:

```
mysql> INSERT INTO pais(nombrePais) VALUES ('Ecuador'),('Peru'),('Argentina'),('Estados Unidos');
Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

Nota. Cuando la tabla cuenta con una columna autoincremental no es necesario pasar como argumento el nombre de la columna ni el valor.

### 3.3.3.2. Update

El comando UPDATE en bases de datos se utiliza para modificar los datos existentes en una tabla. Es una de las operaciones fundamentales en SQL y permite actualizar uno o más registros en una tabla de acuerdo con ciertos criterios.

Es importante tener en cuenta que el comando UPDATE puede afectar múltiples filas en una tabla si no se especifica una cláusula WHERE adecuada. Por lo tanto, es recomendable tener precaución al utilizarlo y realizar pruebas exhaustivas para evitar cambios no deseados o errores en los datos.

```
UPDATE table_name  
SET column1 = value1, column2 = value2, .
```

Sintaxis: WHERE condition;

Ejercicio.

Teniendo en cuenta la siguiente información contenida en la tabla país modifique el nombre del país cuyo idPais es 3. y corresponde a Perú; modifíquelo por España usando el comando UPDATE.

| idPais | nombrePais     |
|--------|----------------|
| 1      | Colombia       |
| 2      | Ecuador        |
| 3      | Peru           |
| 4      | Argentina      |
| 5      | Estados Unidos |

```
mysql> UPDATE pais  
    -> SET nombrePais = 'España'  
    -> WHERE idPais = 3;  
Query OK, 1 row affected (0.01 sec)  
Rows matched: 1    Changed: 1    Warnings: 0
```

| idPais | nombrePais     |
|--------|----------------|
| 1      | Colombia       |
| 2      | Ecuador        |
| 3      | España         |
| 4      | Argentina      |
| 5      | Estados Unidos |

### 3.3.3.3 DELETE

El comando DELETE en bases de datos se utiliza para eliminar registros existentes de una tabla. Es una operación importante que permite eliminar datos no deseados, obsoletos o incorrectos de una base de datos.

`DELETE FROM table_name WHERE condition;`

Sintaxis:

Ejercicio.

El gerente de mitienda le genera una solicitud al departamento IT, el gerente le indica a el personal que se va a clausurar la sede de ecuador ya que no ha tenido suficiente acogida.

| idPaís | nombrePaís     |
|--------|----------------|
| 1      | Colombia       |
| 2      | Ecuador        |
| 3      | España         |
| 4      | Argentina      |
| 5      | Estados Unidos |

```
mysql> DELETE FROM pais
-> WHERE idPaís = 2;
Query OK, 1 row affected (0.00 sec)
```

| idPaís | nombrePaís     |
|--------|----------------|
| 1      | Colombia       |
| 3      | España         |
| 4      | Argentina      |
| 5      | Estados Unidos |

### 3.3.4. Comandos DQL

Los comandos DQL (Data Query Language) en bases de datos se utilizan para recuperar y consultar datos de una base de datos. Estos comandos permiten realizar consultas y obtener información específica de las tablas. Los comandos DQL más comunes son

- **SELECT:** El comando SELECT se utiliza para recuperar datos de una o varias tablas de la base de datos. Permite especificar qué columnas se desean recuperar, las tablas involucradas y las condiciones para filtrar los registros. El resultado de una consulta SELECT es un conjunto de filas que coinciden con los criterios especificados.
- **FROM:** La cláusula FROM se utiliza en conjunto con el comando SELECT para especificar las tablas de donde se deben recuperar los datos. Permite especificar una o varias tablas y establecer relaciones entre ellas mediante cláusulas de unión.
- **WHERE:** La cláusula WHERE se utiliza en conjunto con el comando SELECT para filtrar los registros que se desean recuperar. Permite establecer condiciones basadas en los valores de las columnas para seleccionar solo las filas que cumplen con esas condiciones.
- **ORDER BY:** La cláusula ORDER BY se utiliza para ordenar los resultados de una consulta SELECT en función de una o varias columnas. Puede especificar si el ordenamiento debe ser ascendente (ASC) o descendente (DESC).

- **GROUP BY:** La cláusula GROUP BY se utiliza para agrupar filas de una consulta SELECT en conjuntos basados en los valores de una o varias columnas. Se utiliza en conjunto con funciones de agregación, como SUM, COUNT, AVG, entre otras, para realizar cálculos en los conjuntos de datos agrupados.
- **HAVING:** La cláusula HAVING se utiliza en conjunto con la cláusula GROUP BY para filtrar los resultados de una consulta basada en las condiciones de las funciones de agregación aplicadas en la cláusula SELECT.

#### 3.3.4.1 SELECT

```
SELECT column1, column2, ...
FROM table_name;
```

Sintaxis:

Ejercicio:

El gerente comercial de mitienda desea ver los países en los cuales la empresa está ubicada y le solicita a Ud. como DBA le suministre dicha información.

```
mysql> SELECT idPaís, nombrePaís
-> FROM país;
+-----+-----+
| idPaís | nombrePaís |
+-----+-----+
|      1 | Colombia   |
|      3 | España      |
|      4 | Argentina   |
|      5 | Estados Unidos |
+-----+-----+
4 rows in set (0.00 sec)
```

#### 3.3.4.2. WHERE

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Sintaxis.

Ejercicio.

El gerente comercial de mitienda desea ver los países cuyo id sea 1.

```

2   SELECT idPais,nombrePais
3   FROM pais
4   WHERE idPais = 1;  4ms

```

The screenshot shows a MySQL Workbench interface. A red arrow points from the text "Haciendo uso del comando Insert agregue regiones y departamento en las tablas respectivas." down to the search results window. The results show one row: idPais 1 and nombrePais Colombia.

Ejercicio:

Haciendo uso del comando Insert agregue regiones y departamento en las tablas respectivas.

|   | * idDep<br>int | * nombreDep<br>varchar(50) | idPais<br>int |
|---|----------------|----------------------------|---------------|
| 1 | 1              | Santander                  | 1             |
| 2 | 2              | Amazonas                   | 1             |
| 3 | 3              | Norte de santander         | 1             |
| 4 | 4              | Cundinamarca               | 1             |

|   | * idDep<br>int | * nombreDep<br>varchar(50) | idPais<br>int |
|---|----------------|----------------------------|---------------|
| 1 | 5              | Nueva Inglaterra           | 5             |
| 2 | 6              | Atlántico Medio            | 5             |
| 3 | 7              | Atlántico Sur              | 5             |
| 4 | 8              | Centro Norte Oriental      | 5             |
| 5 | 9              | Centro Sur Oriental        | 5             |
| 6 | 10             | Centro Norte Occidental    | 5             |
| 7 | 11             | Centro Sur Occidental      | 5             |

Regiones

|   | * idReg<br>int | * nombreReg<br>varchar(50) | idDep<br>int |
|---|----------------|----------------------------|--------------|
| 1 | 1              | Maine                      | 5            |
| 2 | 2              | Nuevo Hampshire            | 5            |
| 3 | 3              | Vermont                    | 5            |
| 4 | 4              | Massachusetts              | 5            |
| 5 | 5              | Connecticut                | 5            |
| 6 | 6              | Rhode                      | 5            |
| 7 | 7              | Island                     | 5            |

|   | * idReg<br>int | * nombreReg<br>varchar(50) | idDep<br>int |
|---|----------------|----------------------------|--------------|
| 1 | 8              | Bucaramanga                | 1            |
| 2 | 9              | Floridablanca              | 1            |
| 3 | 10             | Barrancabermeja            | 1            |
| 4 | 11             | Girón                      | 1            |
| 5 | 12             | Piedecuesta                | 1            |
| 6 | 13             | Lebrija                    | 1            |

### 3.3.4.3. Operadores Lógicos

Los operadores lógicos son palabras clave utilizadas en SQL para filtrar los resultados de una consulta según condiciones lógicas específicas. Los operadores lógicos básicos en SQL son:

- **AND:** Este operador permite que se cumplan múltiples condiciones. Si tienes dos condiciones, ambas deben ser verdaderas para que la fila completa se incluya en los resultados de la consulta. Por ejemplo:

```
SELECT * FROM Employees WHERE Age > 30 AND Salary > 50000;
```

Esto seleccionará todos los registros de la tabla **Employees** donde la **Age** sea mayor que 30 y el **Salary** sea mayor que 50000.

- **OR:** Este operador también permite que se cumplan múltiples condiciones. Sin embargo, si tienes dos condiciones, solo una de ellas necesita ser verdadera para que la fila completa se incluya en los resultados de la consulta. Por ejemplo:

```
SELECT * FROM Employees WHERE Age > 30 OR Salary > 50000;
```

Esto seleccionará todos los registros de la tabla **Employees** donde la **Age** sea mayor que 30 o el **Salary** sea mayor que 50000.

- **NOT:** Este operador niega la condición que sigue. Si tienes una condición, la fila completa se incluirá en los resultados de la consulta solo si esa condición no es verdadera. Por ejemplo:

```
SELECT * FROM Employees WHERE NOT Salary > 50000;
```

Esto seleccionará todos los registros de la tabla **Employees** donde el **Salary** no sea mayor que 50000.

#### 3.3.4.4. Operadores de comparación

Los operadores de comparación en SQL se utilizan para comparar dos valores. Aquí están los más comunes:

- **= (Igual a):** Este operador se utiliza para verificar si dos valores son iguales. Por ejemplo:

```
SELECT * FROM Employees WHERE Salary = 50000;
```

Esta consulta seleccionará todos los registros de la tabla **Employees** donde el **Salary** es igual a 50000.

- **<> o != (No es igual a):** Estos operadores se utilizan para verificar si dos valores no son iguales. Por ejemplo:

```
SELECT * FROM Employees WHERE Salary <> 50000;
```

Esta consulta seleccionará todos los registros de la tabla **Employees** donde el **Salary** no es igual a 50000.

- **> (Mayor que):** Este operador se utiliza para verificar si un valor es mayor que otro. Por ejemplo:

```
SELECT * FROM Employees WHERE Salary > 50000;
```

- **< (Menor que):** Este operador se utiliza para verificar si un valor es menor que otro. Por ejemplo:

```
SELECT * FROM Employees WHERE Salary < 50000;
```

Esta consulta seleccionará todos los registros de la tabla **Employees** donde el **Salary** es menor que 50000.

- **>= (Mayor que o igual a):** Este operador se utiliza para verificar si un valor es mayor que o igual a otro. Por ejemplo:

```
SELECT * FROM Employees WHERE Salary >= 50000;
```

- **<= (Menor que o igual a)**: Este operador se utiliza para verificar si un valor es menor que o igual a otro. Por ejemplo:

#### 3.3.4.5 Operador Like

El operador **LIKE** en MySQL se utiliza en la cláusula **WHERE** para buscar un patrón específico en una columna. Es similar a cómo se utiliza **LIKE** en SQL en general.

Aquí hay dos caracteres especiales comúnmente usados con **LIKE**:

- **%**: El signo de porcentaje representa cero, uno o varios caracteres.
- **\_**: El guion bajo representa un solo carácter.

```
SELECT * FROM Employees WHERE FirstName LIKE 'J%';
```

Esta consulta seleccionará todos los registros de la tabla **Employees** donde **FirstName** comienza con "J". El % actúa como un comodín que coincide con cualquier secuencia de caracteres.

```
SELECT * FROM Employees WHERE PhoneNumber LIKE '123-456-____';
```

Esta consulta seleccionará todos los registros de la tabla **Employees** donde **PhoneNumber** tiene el formato '123-456-' seguido de cualquier cuatro caracteres.

#### 3.3.4.6. Between

El operador **BETWEEN** en MySQL se utiliza para seleccionar valores dentro de un rango determinado. Este rango puede ser numérico, de fecha o de texto.

La sintaxis básica de **BETWEEN** es:

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Ejemplos

- ```
SELECT *
FROM Orders
WHERE OrderPrice BETWEEN 50 AND 100;
```
- ```
SELECT *
FROM Orders
WHERE OrderDate BETWEEN '2022-01-01' AND '2022-12-31';
```

```
SELECT *
FROM Orders
WHERE OrderPrice NOT BETWEEN 50 AND 100;
```

### 3.4. Introducción a PDO

PDO (PHP Data Objects) es una interfaz de abstracción de bases de datos en PHP. Proporciona un conjunto de clases y métodos para preparar y ejecutar consultas SQL de una manera segura y eficiente. PDO no es una biblioteca de base de datos en sí, pero proporciona una forma unificada de trabajar con diferentes tipos de bases de datos.

Una de las principales ventajas de PDO es su capacidad para trabajar con parámetros vinculados y consultas preparadas. Esto puede ayudar a proteger tu aplicación contra ataques de inyección de SQL.

El primer argumento de la clase PDO es el DSN, DataSourceName, en el cual se han de especificar el tipo de base de datos.

```
try {
    $dsn = "mysql:host=localhost;dbname=$dbname";
    $dbh = new PDO($dsn, $user, $password);
} catch (PDOException $e){
    echo $e->getMessage();
}
```

Sintaxis.

#### 3.4.1. Excepciones y opciones con PDO

PDO maneja los errores en forma de excepciones, por lo que la conexión siempre ha de ir encerrada en un bloque try/catch. Se puede (y se debe) especificar el modo de error estableciendo el atributo error mode:

```
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_SILENT);
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

- **PDO::ERRMODE\_SILENT.** Es el **modo de error por defecto**. Si se deja así habrá que comprobar los errores de forma parecida a como se hace con **mysqli**. Se tendrían que emplear [PDO::errorCode\(\)](#) y [PDO::errorInfo\(\)](#) o su versión en PDOStatement [PDOStatement::errorCode\(\)](#) y [PDOStatement::errorInfo\(\)](#).
- **PDO::ERRMODE\_WARNING.** Además de establecer el código de error, PDO emitirá un mensaje E\_WARNING. Modo empleado para depurar o hacer pruebas para ver errores sin interrumpir el flujo de la aplicación.
- **PDO::ERRMODE\_EXCEPTION.** Además de establecer el código de error, PDO lanzará una excepción PDOException y establecerá sus propiedades para luego poder reflejar el error y su información. Este modo se emplea en la mayoría de situaciones, ya que permite manejar los errores y a la vez esconder datos que podrían ayudar a alguien a atacar tu aplicación.

### 3.4.2. Conexión a bases de datos

Para establecer conexión a una base de datos puede realizar de diferentes maneras, a continuación, estudiaremos una de las diferentes formas en la cual podemos realizar este tipo de tareas.

en este ejercicio lo primero que llevamos a cabo es la construcción de un archivo en PHP que permita configurar las diferentes opciones de conexión a dos tipos de motor de base de datos diferentes (Mysql y Postgres).

Archivo connectionString.php

```
<?php
$settings = Array(
    'db' => Array(
        'driver' => 'mysql',
        'host' => 'localhost',
        'username' => 'root',
        'database' => 'mitienda',
        'password' => '123456',
        'collation' => 'utf8mb4_unicode_ci',
        'flags' => [
            // Turn off persistent connections
            PDO::ATTR_PERSISTENT => false,
            // Enable exceptions
            PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
            // Emulate prepared statements
            PDO::ATTR_EMULATE_PREPARES => true,
            // Set default fetch mode to array
            PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
            // Set character set
            PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8mb4 COLLATE utf8mb4_unicode_ci'
        ]
    ),
    'db2' => Array(
        'driver' => 'pgsql',
        'host' => 'localhost',
        'username' => 'postgres',
        'database' => 'mitienda',
        'password' => '123456',
        'flags' => [
            // Turn off persistent connections
            PDO::ATTR_PERSISTENT => false,
            // Enable exceptions
            PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
            // Set default fetch mode to array
            PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
            // Set character set
            PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8'
        ]
    )
);
return $settings;
?>
```

Archivo Database.php : Este archivo contiene toda la información para establecer conexión con la base(s) de dato(s).

```

<?php
class Database{
    private $conn;
    private $settings;
    public function __construct() {
        // Requerir el archivo de configuración y asignarlo a $this->settings
        $this->settings = require_once('../config/connectionString.php');
    }

    public function getConnection($dbKey) {
        $dbConfig = $this->settings[$dbKey];
        $this->conn = null;

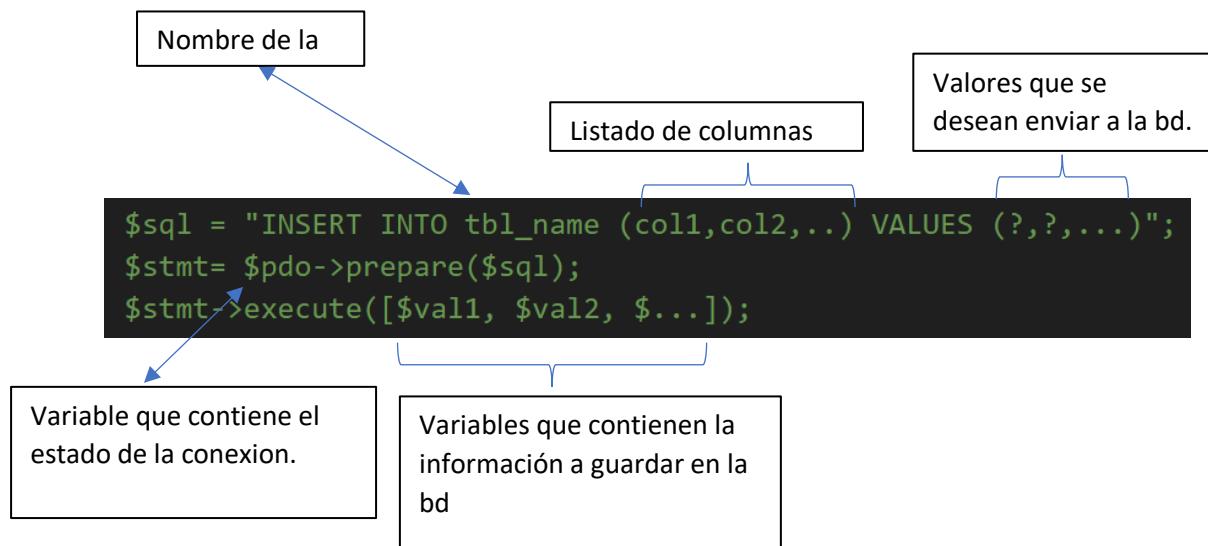
        // $dsn = "{$dbConfig['driver']}:host={$dbConfig['host']};dbname={$dbConfig['database']};charset={$dbConfig['ch
        $dsn = "{$dbConfig['driver']}:host={$dbConfig['host']};dbname={$dbConfig['database']}";
        try{
            $this->conn = new PDO($dsn, $dbConfig['username'], $dbConfig['password'], $dbConfig['flags']);
            echo 'ok';
        }catch(PDOException $exception){
            $error=[[
                'error' => $exception->getMessage(),
                'message' => 'Error al momento de establecer conexión'
            ]];
            return $error;
        }
        return $this->conn;
    }

}
?>

```

### 3.4.3. Registro de datos

La clase PDOStatement es la que trata las sentencias SQL. Una instancia de PDOStatement se crea cuando se llama a PDO->prepare(), y con ese objeto creado se llama a métodos como bindParam() para pasar valores o execute() para ejecutar sentencias. PDO facilita el uso de sentencias preparadas en PHP, que mejoran el rendimiento y la seguridad de la aplicación. Cuando se obtienen, insertan o actualizan datos, el esquema es: PREPARE -> [BIND] -> EXECUTE. Se pueden indicar los parámetros en la sentencia con un interrogante "?" o mediante un nombre específico.



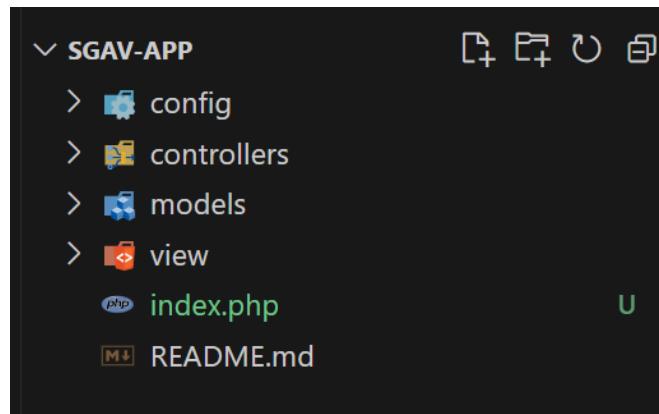
Ejemplo

```
$sql = "INSERT INTO pais (nombre_pais) VALUES (?)";
$stmt= $conn->prepare($sql);
$nombrePais='Mexico2';
$stmt->execute([$nombrePais]);
```

### 3.5 Creación de una aplicación usando PHP+Mysql+PDO

En este apartado vamos a crear un ejemplo de aplicación usando php, pdo, poo, mysql, postgres.

1. Cree un repositorio en git con el nombre sgav-app (Sistema de Gestión para el Alquiler de viviendas)
2. Clone el repositorio en la carpeta DocumentRoot del servidor web configurado previamente.
3. Cree la siguiente estructura de proyecto



4. Configura el autoload de clases usando composer(Previamente configurado). Siga los siguientes pasos.
  - a. Ingrese el comando en la terminal de vs Code. Recuerde que se debe encontrar en la carpeta del proyecto actual. `$ composer init`
  - b. Cuando se inicia el proceso de configuración se presenta la siguiente información:

```
Welcome to the Composer config generator

This command will guide you through creating your composer.json config.

Package name (<vendor>/<name>) [desarrollo/sgav-app]:
```

El asistente realizara una serie de preguntas. Sigas las indicaciones de la siguiente imagen:

```

This command will guide you through creating your composer.json config.
Enter
Description []: Sistema para la gestion y alquiler de viviendas Ingrese una descripción y presione Enter
Author [trainingLeader <131613955+trainingLeader@users.noreply.github.com>, n to skip]: Presione enter
Minimum Stability []: Presione enter
Package Type (e.g. library, project, metapackage, composer-plugin) []: project Escriba project
License []: Presione enter

Define your dependencies.

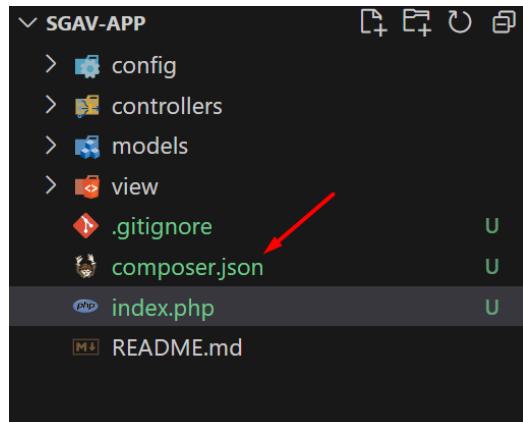
Would you like to define your dependencies (require) interactively [yes]? no
Would you like to define your dev dependencies (require-dev) interactively [yes]? no
Add PSR-4 autoload mapping? Maps namespace "Desarrollo\SgavApp" to the entered relative path. [src/, n to skip]: n
f

{
    "name": "desarrollo/sgav-app",
    "description": "Sistema para la gestion y alquiler de viviendas",
    "type": "project",
    "authors": [
        {
            "name": "trainingLeader",
            "email": "131613955+trainingLeader@users.noreply.github.com"
        }
    ],
    "require": {}
}

Do you confirm generation [yes]? yes
Would you like the vendor directory added to your .gitignore [yes]? yes

```

Cuando finalice la configuración en el proyecto se creará un archivo llamado composer.json el cual contendrá la información de paquetes y configuración realizada en el proyecto.



5. Configure las carpetas y alias para que composer realice de forma automática la carga de clases y las tengamos disponibles en cualquier instante del desarrollo.
6. Abra el archivo composer.json y agregue el código indicado en la siguiente imagen.

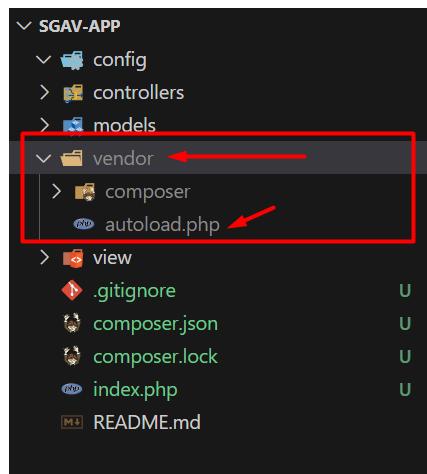
```
composer.json > ...
1  {
2      "name": "desarrollo/sgav-app",
3      "description": "Sistema para la gestion y alquiler de viviendas",
4      "type": "project",
5      "authors": [
6          {
7              "name": "trainingLeader",
8              "email": "131613955+trainingLeader@users.noreply.github.com"
9          }
10     ],
11     "require": {},
12     "autoload": {
13         "psr-4": {
14             "App\\": "./config",
15             "Models\\": "./models"
16         }
17     }
18 }
19 }
```

Cuando finalice de insertar el código indicado en la imagen anterior ingrese el comando composer update.

```
desarrollo@DESKTOP-L4V647N MINGW64 /c/apache/htdocs/sgav-app (main)
$ composer update
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating autoload files
No installed packages - skipping audit.

desarrollo@DESKTOP-L4V647N MINGW64 /c/apache/htdocs/sgav-app (main)
$
```

Como resultado de la actualización se genera la carpeta vendor y dentro de ella el archivo autoload.php



7. A continuación, se crearán los archivos que permitirán establecer conexión con la base de datos en este ejemplo se conectarán bases de datos MySQL y postgres.

Cree un archivo dentro de la carpeta config. Nombre este archivo Database.php. Agregue el siguiente código:

```

1  <?php
2      namespace App;
3      class Database{
4          private $conn;
5          protected static $settings=array(
6              "mysql"=> Array(
7                  'driver' => 'mysql',
8                  'host' => 'localhost',
9                  'username' => 'root',
10                 'database' => 'mitienda',
11                 'password' => '123456',
12                 'collation' => 'utf8mb4_unicode_ci',
13                 'flags' => [
14                     // Turn off persistent connections
15                     \PDO::ATTR_PERSISTENT => false,
16                     // Enable exceptions
17                     \PDO::ATTR_ERRMODE => \PDO::ERRMODE_EXCEPTION,
18                     // Emulate prepared statements
19                     \PDO::ATTR_EMULATE_PREPARES => true,
20                     // Set default fetch mode to array
21                     \PDO::ATTR_DEFAULT_FETCH_MODE => \PDO::FETCH_ASSOC,
22                     // Set character set
23                     \PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8mb4 COLLATE utf8mb4_unicode_ci'
24             ],
25             "pgsql"=> Array(
26                 'driver' => 'pgsql',
27                 'host' => 'localhost',
28                 'username' => 'postgres',
29                 'database' => 'mitienda',
30                 'password' => '123456',
31                 'flags' => [
32                     // Turn off persistent connections
33                     \PDO::ATTR_PERSISTENT => false,
34                     // Enable exceptions
35                     \PDO::ATTR_ERRMODE => \PDO::ERRMODE_EXCEPTION,
36                     // Set default fetch mode to array
37                     \PDO::ATTR_DEFAULT_FETCH_MODE => \PDO::FETCH_ASSOC,
38                     // Set character set
39                     \PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8'
40             ]
41         )
42     );
43     public function __construct($args = []) {
44         $this->conn = $args['conn'] ?? null;
45     }
46
47     public function getConnection($dbKey) {
48         $dbConfig = self::$settings[$dbKey];
49         $this->conn = null;
50         $dsn = "{$dbConfig['driver']}:host={$dbConfig['host']};dbname={$dbConfig['database']}";
51         try{
52             $this->conn = new \PDO($dsn, $dbConfig['username'], $dbConfig['password'], $dbConfig['flags']);
53             echo 'okkkkk';
54         }catch(\PDOException $exception){
55             $error=[[
56                 'error' => $exception->getMessage(),
57                 'message' => 'Error al momento de establecer conexión'
58             ]];
59             return $error;
60         }
61         return $this->conn;
62     }
63 }
64 ?>

```

El código PHP proporciona una clase llamada **Database** que se utiliza para establecer una conexión con una base de datos. Aquí está lo que hace el código en detalle:

- Define un espacio de nombres (**namespace**) llamado **App**. Esto permite organizar el código y evitar conflictos de nombres con otras clases o bibliotecas.
- Define la clase **Database** dentro del espacio de nombres **App**. La clase tiene una propiedad privada llamada **\$conn** que almacena la conexión a la base de datos.
- Declara una matriz estática llamada **\$settings**, que contiene la configuración para las conexiones a la base de datos. La matriz tiene dos elementos: uno para MySQL y otro para PostgreSQL. Cada elemento contiene los detalles de conexión específicos para el respectivo tipo de base de datos, como el controlador, el host, el nombre de usuario, la contraseña, etc.
- El constructor de la clase **Database** acepta un argumento opcional **\$args**, que puede contener una conexión a la base de datos existente. Si no se proporciona ningún argumento, se establece el valor predeterminado de **\$conn** como **null**.
- La función **getConnection** toma un parámetro **\$dbKey**, que especifica qué configuración de base de datos se debe utilizar (por ejemplo, "mysql" o "pgsql"). Utiliza esta clave para obtener la configuración correspondiente de la matriz **\$settings**.
- Dentro de la función **getConnection**, se establece **\$conn** en **null** para asegurarse de que no haya una conexión existente.
- Luego, se construye una cadena de conexión (**\$dsn**) utilizando los detalles de la configuración de la base de datos obtenidos de **\$settings**.
- Se intenta establecer una conexión a la base de datos utilizando la clase **PDO** de PHP. Si ocurre algún error durante la conexión, se captura la excepción **PDOException** y se devuelve un mensaje de error.
- Si la conexión se establece correctamente, se devuelve el objeto de conexión **\$conn**.

**En resumen, este código PHP proporciona una clase Database que se puede utilizar para conectarse a una base de datos utilizando PDO (PHP Data Objects) y configuraciones específicas para MySQL y PostgreSQL.**

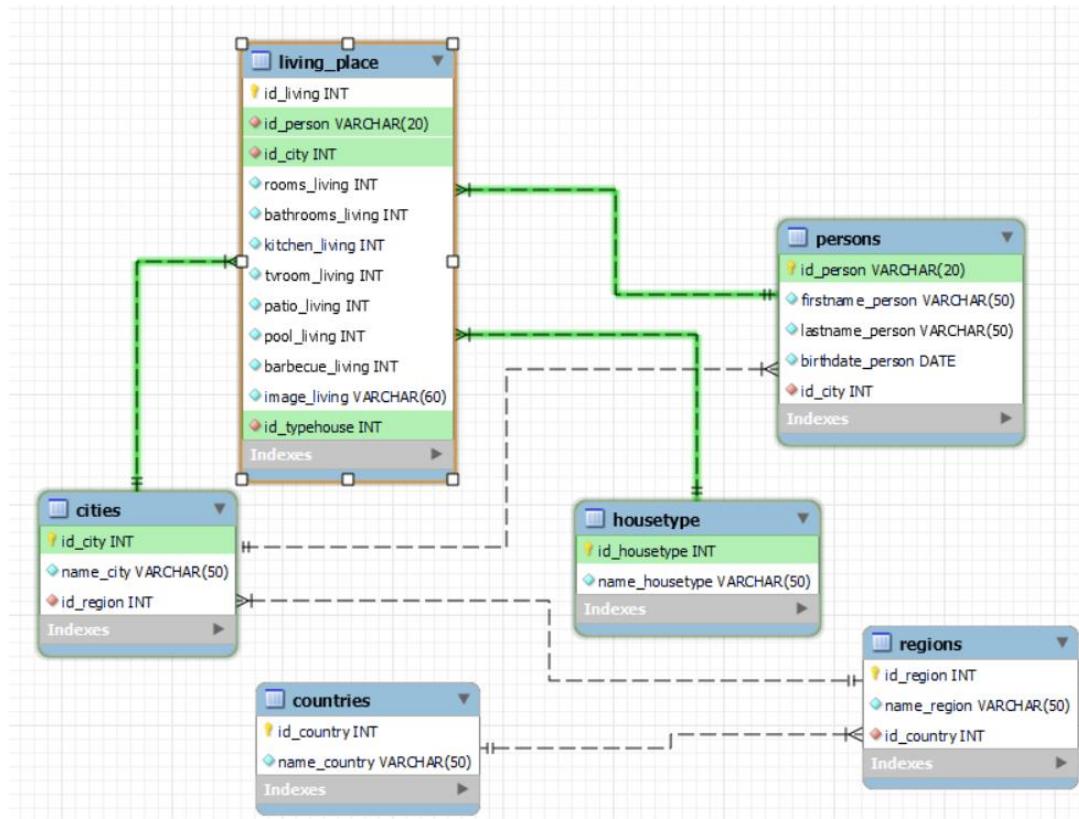
8. Cuando finalice con el archivo Database, cree un archivo en la raíz del proyecto y llámelo app.php. Agregue el siguiente código:

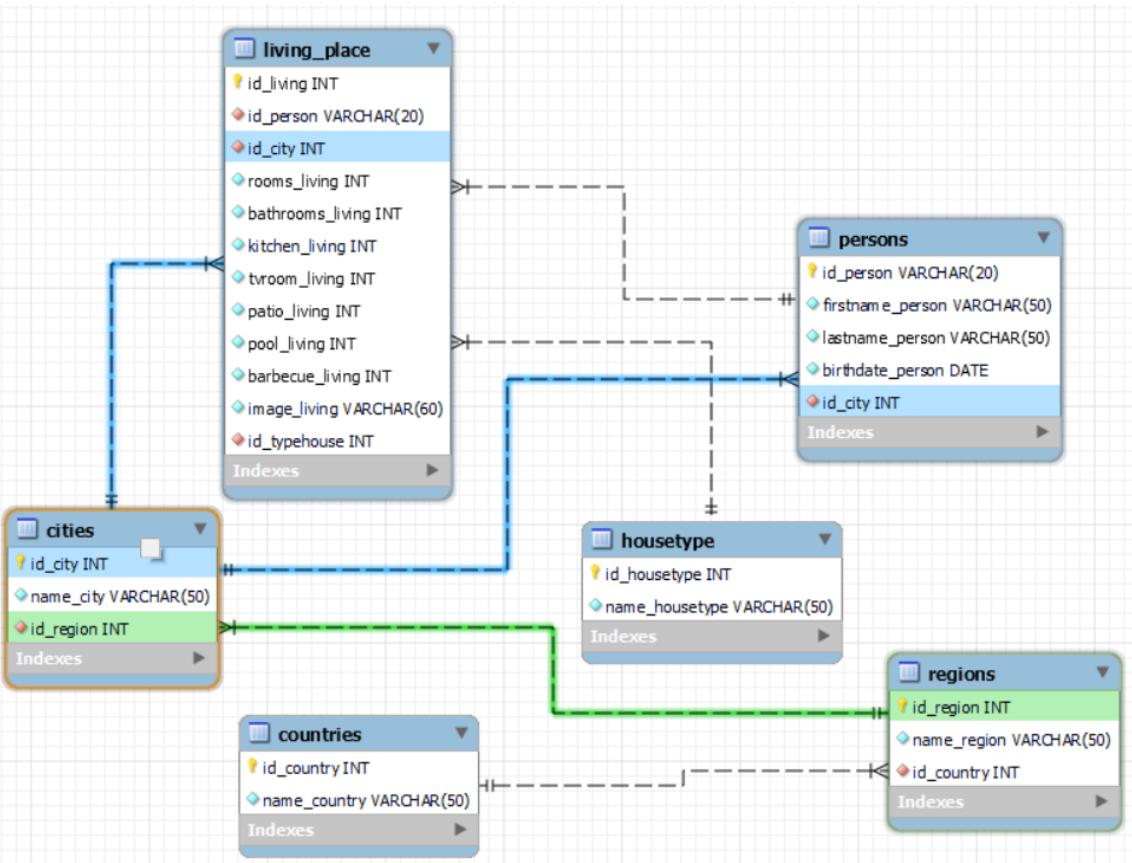
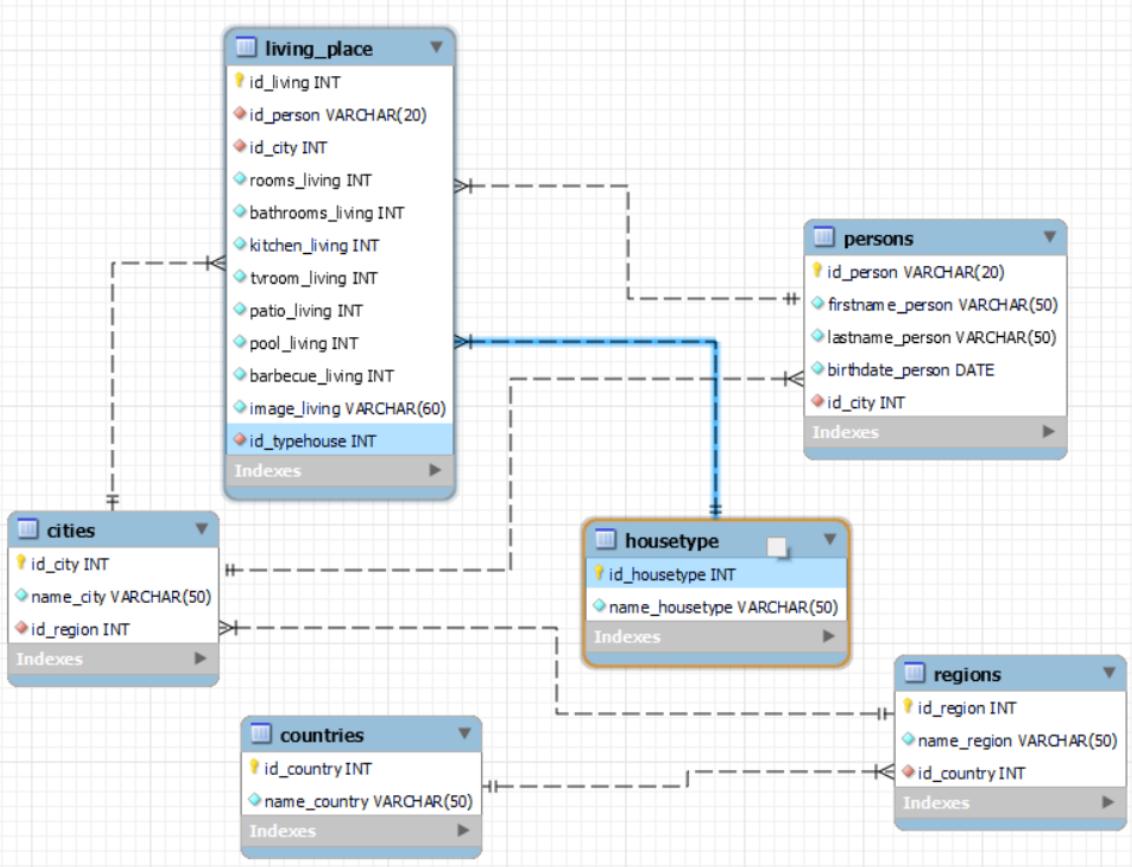
```
app.php > ...
1  <?php
2  require_once 'vendor/autoload.php';
3  use App\Database;
4  $db = new Database();
5  $conn = $db->getConnection('mysql');
6  ?>
```

- **require\_once 'vendor/autoload.php';** - Esta línea importa automáticamente las clases y archivos necesarios utilizando el autoloader de Composer. El archivo **autoload.php** se genera después de instalar las dependencias del proyecto a través de Composer. Esto asegura que la clase **Database** y todas sus dependencias se carguen correctamente.

- `use App\Database;` - Esta línea importa la clase **Database** del espacio de nombres **App**. Ahora podemos usar directamente el nombre **Database** en lugar de **App\Database**.
- `$db = new Database();` - Se crea una instancia de la clase **Database** utilizando el constructor sin argumentos. Esto crea un objeto **\$db** que representa la conexión a la base de datos.
- `$conn = $db->getConnection('mysql');` - Se llama al método **getConnection** del objeto **\$db** y se pasa la clave '**mysql**' como argumento. Este método devuelve la conexión a la base de datos específica de MySQL según la configuración proporcionada en la clase **Database**. El objeto de conexión se asigna a la variable **\$conn**.
- En resumen, el código carga las dependencias, crea una instancia de la clase **Database** y luego obtiene una conexión a la base de datos específica de MySQL utilizando esa instancia. La conexión resultante se almacena en la variable **\$conn**, que puede usarse más adelante para realizar consultas y operaciones en la base de datos.

9. Cree una base de datos en Mysql teniendo en cuenta el siguiente DER.





10. Configure los parámetros de acceso a la base de datos en la variable \$settings que se encuentra en el archivo Database.php.

```

<?php
namespace App;
class Database{
    private $conn;
    protected static $settings=array(
        "mysql"=> Array(
            'driver' => 'mysql',
            'host' => 'localhost', ←
            'username' => 'root', ←
            'database' => 'sgavapp', ←
            'password' => '123456', ←
            'collation' => 'utf8mb4_unicode_ci',
            'flags' => [
                // Turn off persistent connections
                \PDO::ATTR_PERSISTENT => false,
                // Enable exceptions
                \PDO::ATTR_ERRMODE => \PDO::ERRMODE_EXCEPTION,
                // Emulate prepared statements
                \PDO::ATTR_EMULATE_PREPARES => true,
                // Set default fetch mode to array
                \PDO::ATTR_DEFAULT_FETCH_MODE => \PDO::FETCH_ASSOC,
                // Set character set
                \PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8mb4 COLLATE
    
```

Nota: Es importante configurar adecuadamente las credenciales de acceso del servidor de bases de datos ya que esto puede variar entre servidores. Si desconoce estos parámetros consúltelos con el administrador de base de datos o con el trainer a cargo.

11. Genere los models files. Los modelos son la representación lógica de las tablas que se encuentran en la base de datos.

12. Instale el complemento intervention para la gestión de carga de archivos.

<https://packagist.org/packages/datatables/datatables?query=intervention#1.10.21>

Desde el terminal ejecute el siguiente comando: composer require intervention/image

```

$ composer require intervention/image ←
Info from https://repo.packagist.org: #StandWithUkraine
./composer.json has been updated
Running composer update intervention/image
Loading composer repositories with package information
- Installing ralouphie/getallheaders (3.0.3): Extracting archive
- Installing psr/http-message (2.0): Extracting archive
- Installing psr/http-factory (1.0.2): Extracting archive
- Installing guzzlehttp/psr7 (2.5.0): Extracting archive
- Installing intervention/image (2.7.2): Extracting archive
3 package suggestions were added by new dependencies, use `composer suggest` to see details.
Generating autoload files
2 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
No security vulnerability advisories found
Using version ^2.7 for intervention/image

desarrollo@DESKTOP-L4V647N MINGW64 /c/apache/htdocs/sgav-app (main)
$ 

```

### 3.6 Api Rest

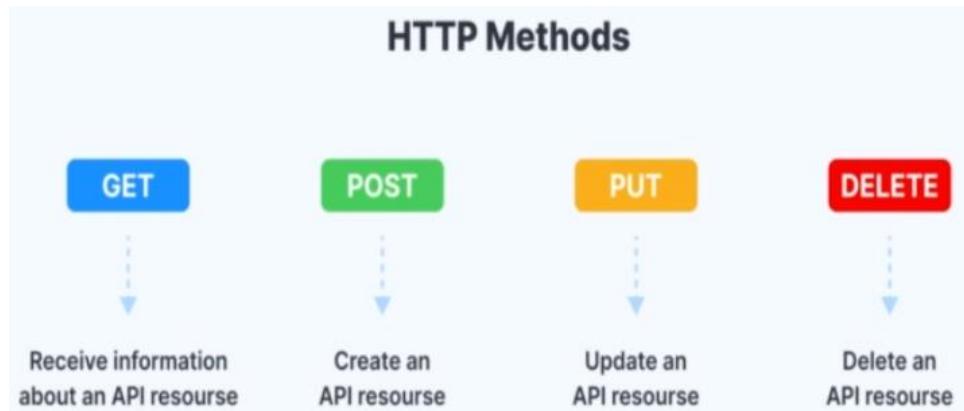
Un API (Application Programming Interface, por sus siglas en inglés) es un conjunto de reglas y protocolos que permite la comunicación entre distintos sistemas informáticos. En términos simples, es una interfaz que define cómo los diferentes componentes de software deben interactuar entre sí.

Un API proporciona una forma estandarizada y estructurada para que las aplicaciones y servicios se comuniquen y comparten datos y funcionalidades. A través de un API, una aplicación puede solicitar información o realizar acciones en otra aplicación o servicio, sin necesidad de conocer los detalles internos de cómo se implementa dicha funcionalidad.

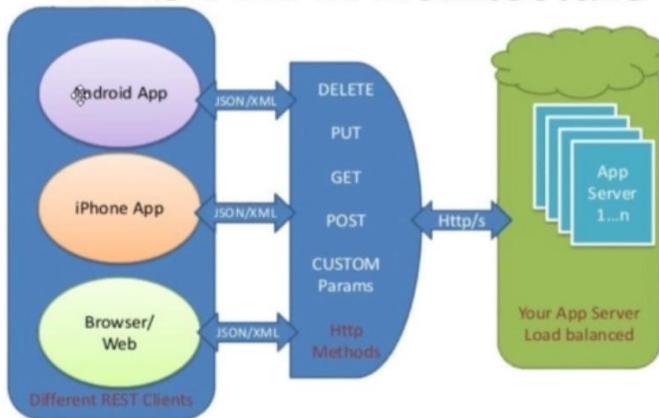
Existen diferentes tipos de API, como las API web, que permiten la comunicación entre aplicaciones a través de Internet utilizando protocolos como HTTP. También existen las API de sistema operativo, que proporcionan acceso a las funciones y recursos del sistema operativo subyacente, y las API de biblioteca, que ofrecen funciones y servicios específicos a los desarrolladores.

Las API han sido fundamentales en el desarrollo de aplicaciones modernas, ya que permiten la integración de diferentes sistemas, el intercambio de datos y la creación de servicios y aplicaciones más complejas a través de la reutilización de componentes existentes. También son ampliamente utilizadas en el desarrollo de interfaces de programación para servicios web (API web) y aplicaciones móviles, permitiendo la interacción con plataformas populares y servicios en la nube.

#### 3.6.1. Métodos API



# REST API Architecture



## 3.6.2. Códigos de estado en Response HTTP

### 3.6.2.1. Respuestas informativas

#### 100 Continue

Esta respuesta provisional indica que todo hasta ahora está bien y que el cliente debe continuar con la solicitud o ignorarla si ya está terminada.

#### 101 Switching Protocol

Este código se envía en respuesta a un encabezado de solicitud [Upgrade \(en-US\)](#) por el cliente e indica que el servidor acepta el cambio de protocolo propuesto por el agente de usuario.

#### 102 Processing (en-US) (WebDAV (en-US))

Este código indica que el servidor ha recibido la solicitud y aún se encuentra procesandola, por lo que no hay respuesta disponible.

#### 103 Early Hints (en-US)

Este código de estado está pensado principalmente para ser usado con el encabezado [Link](#), permitiendo que el agente de usuario empiece a [pre-cargar \(en-US\)](#) recursos mientras el servidor prepara una respuesta.

### 3.6.2.2 Respuestas satisfactorias

- GET: El recurso se ha obtenido y se transmite en el cuerpo del mensaje.
- HEAD: Los encabezados de entidad están en el cuerpo del mensaje.
- PUT o POST: El recurso que describe el resultado de la acción se transmite en el cuerpo del mensaje.

- TRACE: El cuerpo del mensaje contiene el mensaje de solicitud recibido por el servidor.

## [200 OK](#)

La solicitud ha tenido éxito. El significado de un éxito varía dependiendo del método HTTP:

### [201 Created](#)

La solicitud ha tenido éxito y se ha creado un nuevo recurso como resultado de ello. Ésta es típicamente la respuesta enviada después de una petición PUT.

### [202 Accepted](#)

La solicitud se ha recibido, pero aún no se ha actuado. Es una petición "sin compromiso", lo que significa que no hay manera en HTTP que permite enviar una respuesta asíncrona que indique el resultado del procesamiento de la solicitud. Está pensado para los casos en que otro proceso o servidor maneja la solicitud, o para el procesamiento por lotes.

### [203 Non-Authoritative Information](#)

La petición se ha completado con éxito, pero su contenido no se ha obtenido de la fuente originalmente solicitada, sino que se recoge de una copia local o de un tercero. Excepto esta condición, se debe preferir una respuesta de 200 OK en lugar de esta respuesta.

### [204 No Content \(en-US\)](#)

La petición se ha completado con éxito, pero su respuesta no tiene ningún contenido, aunque los encabezados pueden ser útiles. El agente de usuario puede actualizar sus encabezados en caché para este recurso con los nuevos valores.

### [205 Reset Content \(en-US\)](#)

La petición se ha completado con éxito, pero su respuesta no tiene contenidos y además, el agente de usuario tiene que inicializar la página desde la que se realizó la petición, este código es útil por ejemplo para páginas con formularios cuyo contenido debe borrarse después de que el usuario lo envíe.

### [206 Partial Content](#)

La petición servirá parcialmente el contenido solicitado. Esta característica es utilizada por herramientas de descarga como wget para continuar la transferencia de descargas anteriormente interrumpidas, o para dividir una descarga y procesar las partes simultáneamente.

### [207 Multi-Status \(en-US\) \(WebDAV \(en-US\)\)](#)

Una respuesta Multi-Estado transmite información sobre varios recursos en situaciones en las que varios códigos de estado podrían ser apropiados. El cuerpo de la petición es un mensaje XML.

### [208 Multi-Status \(en-US\) \(WebDAV \(en-US\)\)](#)

El listado de elementos DAV ya se notificó previamente, por lo que no se van a volver a listar.

### [226 IM Used \(en-US\)](#) ([HTTP Delta encoding](#))

El servidor ha cumplido una petición GET para el recurso y la respuesta es una representación del resultado de una o más manipulaciones de instancia aplicadas a la instancia actual.

#### 3.6.2.3. Redirecciones

##### [300 Multiple Choice \(en-US\)](#)

Esta solicitud tiene más de una posible respuesta. User-Agent o el usuario debe escoger uno de ellos. No hay forma estandarizada de seleccionar una de las respuestas.

##### [301 Moved Permanently \(en-US\)](#)

Este código de respuesta significa que la URI del recurso solicitado ha sido cambiado. Probablemente una nueva URI sea devuelta en la respuesta.

##### [302 Found](#)

Este código de respuesta significa que el recurso de la URI solicitada ha sido cambiado temporalmente. Nuevos cambios en la URI serán agregados en el futuro. Por lo tanto, la misma URI debe ser usada por el cliente en futuras solicitudes.

##### [303 See Other \(en-US\)](#)

El servidor envía esta respuesta para dirigir al cliente a un nuevo recurso solicitado a otra dirección usando una petición GET.

##### [304 Not Modified](#)

Esta es usada para propósitos de "caché". Le indica al cliente que la respuesta no ha sido modificada. Entonces, el cliente puede continuar usando la misma versión almacenada en su caché.

##### 305 Use Proxy Obsoleto

Fue definida en una versión previa de la especificación del protocolo HTTP para indicar que una respuesta solicitada debe ser accedida desde un proxy. Ha quedado obsoleta debido a preocupaciones de seguridad correspondientes a la configuración de un proxy.

##### 306 unused

Este código de respuesta ya no es usado más. Actualmente se encuentra reservado. Fue usado en previas versiones de la especificación HTTP1.1.

##### [307 Temporary Redirect \(en-US\)](#)

El servidor envía esta respuesta para dirigir al cliente a obtener el recurso solicitado a otra URI con el mismo método que se usó la petición anterior. Tiene la misma semántica que el código de respuesta HTTP 302 Found, con la excepción de que el agente usuario *no debe* cambiar el método HTTP usado: si un POST fue usado en la primera petición, otro POST debe ser usado en la segunda petición.

#### [308 Permanent Redirect \(en-US\)](#)

Significa que el recurso ahora se encuentra permanentemente en otra URI, especificada por la respuesta de encabezado HTTP Location:. Tiene la misma semántica que el código de respuesta HTTP 301 Moved Permanently, con la excepción de que el agente usuario *no debe* cambiar el método HTTP usado: si un POST fue usado en la primera petición, otro POST debe ser usado en la segunda petición.

#### 3.6.2.4. Errores de cliente

##### [400 Bad Request](#)

Esta respuesta significa que el servidor no pudo interpretar la solicitud dada una sintaxis inválida.

##### [401 Unauthorized](#)

Es necesario autenticar para obtener la respuesta solicitada. Esta es similar a 403, pero en este caso, la autenticación es posible.

##### 402 Payment Required

Este código de respuesta está reservado para futuros usos. El objetivo inicial de crear este código fue para ser utilizado en sistemas digitales de pagos. Sin embargo, no está siendo usado actualmente.

##### [403 Forbidden](#)

El cliente no posee los permisos necesarios para cierto contenido, por lo que el servidor está rechazando otorgar una respuesta apropiada.

##### [404 Not Found](#)

El servidor no pudo encontrar el contenido solicitado. Este código de respuesta es uno de los más famosos dada su alta ocurrencia en la web.

##### [405 Method Not Allowed \(en-US\)](#)

El método solicitado es conocido por el servidor pero ha sido deshabilitado y no puede ser utilizado. Los dos métodos obligatorios, GET y HEAD, nunca deben ser deshabilitados y no deberían retornar este código de error.

##### [406 Not Acceptable \(en-US\)](#)

Esta respuesta es enviada cuando el servidor, después de aplicar una [negociación de contenido servidor-impulsado \(en-US\)](#), no encuentra ningún contenido seguido por la criteria dada por el usuario.

##### [407 Proxy Authentication Required \(en-US\)](#)

Esto es similar al código 401, pero la autenticación debe estar hecha a partir de un proxy.

#### [408 Request Timeout](#)

Esta respuesta es enviada en una conexión inactiva en algunos servidores, incluso sin alguna petición previa por el cliente. Significa que el servidor quiere desconectar esta conexión sin usar. Esta respuesta es muy usada desde algunos navegadores, como Chrome, Firefox 27+, o IE9, usa mecanismos de pre-conexión HTTP para acelerar la navegación. También hay que tener en cuenta que algunos servidores simplemente desconectan la conexión sin enviar este mensaje.

#### [409 Conflict \(en-US\)](#)

Esta respuesta puede ser enviada cuando una petición tiene conflicto con el estado actual del servidor.

#### [410 Gone \(en-US\)](#)

Esta respuesta puede ser enviada cuando el contenido solicitado ha sido borrado del servidor.

#### [411 Length Required \(en-US\)](#)

El servidor rechaza la petición porque el campo de encabezado Content-Length no está definido y el servidor lo requiere.

#### [412 Precondition Failed \(en-US\)](#)

El cliente ha indicado pre-condiciones en sus encabezados la cual el servidor no cumple.

#### [413 Payload Too Large](#)

La entidad de petición es más larga que los límites definidos por el servidor; el servidor puede cerrar la conexión o retornar un campo de encabezado Retry-After.

#### [414 URI Too Long \(en-US\)](#)

La URI solicitada por el cliente es más larga de lo que el servidor está dispuesto a interpretar.

#### [415 Unsupported Media Type \(en-US\)](#)

El formato multimedia de los datos solicitados no está soportado por el servidor, por lo cual el servidor rechaza la solicitud.

#### [416 Requested Range Not Satisfiable \(en-US\)](#)

El rango especificado por el campo de encabezado Range en la solicitud no cumple; es posible que el rango está fuera del tamaño de los datos objetivo del URI.

#### [417 Expectation Failed \(en-US\)](#)

Significa que la expectativa indicada por el campo de encabezado Expect solicitada no puede ser cumplida por el servidor.

#### [418 I'm a teapot](#)

El servidor se rehúsa a intentar hacer café con una tetera.

#### [421 Misdirected Request \(en-US\)](#)

La petición fue dirigida a un servidor que no es capaz de producir una respuesta. Esto puede ser enviado por un servidor que no está configurado para producir respuestas por la combinación del esquema y la autoridad que están incluidos en la URI solicitada

#### [422 Unprocessable Entity \(en-US\) \(WebDAV \(en-US\)\)](#)

La petición estaba bien formada pero no se pudo seguir debido a errores de semántica.

#### [423 Locked \(en-US\) \(WebDAV \(en-US\)\)](#)

El recurso que está siendo accedido está bloqueado.

#### [424 Failed Dependency \(en-US\) \(WebDAV \(en-US\)\)](#)

La petición falló debido a una falla de una petición previa.

#### [426 Upgrade Required \(en-US\)](#)

El servidor se rehúsa a aplicar la solicitud usando el protocolo actual, pero puede estar dispuesto a hacerlo después que el cliente se actualice a un protocolo diferente. El servidor envía un encabezado Upgrade en una respuesta para indicar los protocolos requeridos.

#### [428 Precondition Required \(en-US\)](#)

El servidor origen requiere que la solicitud sea condicional. Tiene la intención de prevenir problemas de 'actualización perdida', donde un cliente OBTIENE un estado del recurso, lo modifica, y lo PONE devuelta al servidor, cuando mientras un tercero ha modificado el estado del servidor, llevando a un conflicto.

#### [429 Too Many Requests \(en-US\)](#)

El usuario ha enviado demasiadas solicitudes en un periodo de tiempo dado.

#### [431 Request Header Fields Too Large \(en-US\)](#)

El servidor no está dispuesto a procesar la solicitud porque los campos de encabezado son demasiado largos. La solicitud PUEDE volver a subirse después de reducir el tamaño de los campos de encabezado solicitados.

#### [451 Unavailable For Legal Reasons \(en-US\)](#)

El usuario solicita un recurso ilegal, como alguna página web censurada por algún gobierno.

### 3.6.2.5. Errores de servidor

#### [500 Internal Server Error](#)

El servidor ha encontrado una situación que no sabe cómo manejarla.

#### [501 Not Implemented \(en-US\)](#)

El método solicitado no está soportado por el servidor y no puede ser manejado. Los únicos métodos que los servidores requieren soporte (y por lo tanto no deben retornar este código) son GET y HEAD.

#### [502 Bad Gateway](#)

Esta respuesta de error significa que el servidor, mientras trabaja como una puerta de enlace para obtener una respuesta necesaria para manejar la petición, obtuvo una respuesta inválida.

#### [503 Service Unavailable](#)

El servidor no está listo para manejar la petición. Causas comunes puede ser que el servidor está caído por mantenimiento o está sobrecargado. Hay que tomar en cuenta que, junto con esta respuesta, una página usuario-amigable explicando el problema debe ser enviada. Estas respuestas deben ser usadas para condiciones temporales y el encabezado HTTP Retry-After: debería, si es posible, contener el tiempo estimado antes de la recuperación del servicio. El webmaster debe también cuidar los encabezados relacionados al caché que son enviados junto a esta respuesta, ya que estas respuestas de condición temporal deben usualmente no estar en el caché.

#### [504 Gateway Timeout](#)

Esta respuesta de error es dada cuando el servidor está actuando como una puerta de enlace y no puede obtener una respuesta a tiempo.

#### [505 HTTP Version Not Supported](#)

La versión de HTTP usada en la petición no está soportada por el servidor.

#### [506 Variant Also Negotiates \(en-US\)](#)

El servidor tiene un error de configuración interna: negociación de contenido transparente para la petición resulta en una referencia circular.

#### [507 Insufficient Storage \(en-US\)](#)

El servidor tiene un error de configuración interna: la variable de recurso escogida está configurada para acoplar la negociación de contenido transparente misma, y no es por lo tanto un punto final adecuado para el proceso de negociación.

#### [508 Loop Detected \(en-US\) \(WebDAV \(en-US\)\)](#)

El servidor detectó un ciclo infinito mientras procesaba la solicitud.

#### [510 Not Extended \(en-US\)](#)

Extensiones adicionales para la solicitud son requeridas para que el servidor las cumpla.

### [511 Network Authentication Required \(en-US\)](#)

El código de estado 511 indica que el cliente necesita autenticar para obtener acceso a la red.

Fuente: <https://developer.mozilla.org/es/docs/Web/HTTP>Status>

#### 3.6.3. Tipos de autenticación

En API REST, existen varios métodos de autenticación que se utilizan para garantizar la seguridad y proteger los recursos. Aquí tienes algunos ejemplos comunes de tipos de autenticación en API REST:

- Autenticación basada en token (Token-Based Authentication): En este enfoque, el cliente envía un token de acceso con cada solicitud al servidor. El token es emitido por el servidor después de que el cliente proporciona sus credenciales de autenticación (como nombre de usuario y contraseña) de manera segura. El servidor verifica la validez del token antes de procesar la solicitud. Ejemplos de implementaciones populares de este tipo de autenticación son JSON Web Tokens (JWT) y OAuth 2.0.
- Autenticación básica (Basic Authentication): En este método, el cliente envía las credenciales (nombre de usuario y contraseña) codificadas en Base64 en el encabezado de la solicitud. Aunque es fácil de implementar, la autenticación básica no cifra las credenciales y, por lo tanto, no se considera segura para entornos no HTTPS.
- Autenticación mediante API key (API Key Authentication): En este enfoque, el cliente envía una clave de API junto con la solicitud al servidor. La clave de API es un identificador único y secreto que se emite al cliente para autenticarse en cada solicitud. El servidor verifica la validez de la clave de API antes de procesar la solicitud.
- Autenticación mediante OAuth: OAuth es un protocolo de autorización que permite a una aplicación solicitar acceso a los recursos protegidos en nombre de un usuario. El cliente obtiene un token de acceso válido después de que el usuario autoriza la aplicación. Este token se utiliza posteriormente para autenticar las solicitudes al servidor.

#### 3.6.4. Implementando ApiRest (Slim)

Slim Framework es un framework de desarrollo de aplicaciones web minimalista y ligero para PHP. Está diseñado para ser simple y rápido, permitiendo a los desarrolladores crear aplicaciones web poderosas y eficientes con una curva de aprendizaje mínima.

Slim Framework se destaca por su enfoque minimalista y su capacidad para crear API y aplicaciones web rápidas y sencillas. Proporciona una base sólida para construir aplicaciones, sin agregar una cantidad abrumadora de características y complejidad innecesaria.

Algunas características clave de Slim Framework incluyen:

- Enrutamiento: Permite definir rutas para responder a diferentes solicitudes HTTP (GET, POST, PUT, DELETE, etc.) y asociarlas con controladores y acciones específicas.
- Manejo de solicitudes y respuestas: Proporciona una interfaz simple y flexible para manejar y manipular solicitudes HTTP entrantes y generar respuestas HTTP salientes.

- Contenedores de dependencias: Incorpora un contenedor de dependencias para gestionar la creación y resolución de objetos y sus dependencias.
- Middleware: Permite agregar capas adicionales de funcionalidad en el proceso de solicitud y respuesta, como la autenticación, el registro de solicitudes, el almacenamiento en caché, entre otros.
- Plantillas: Ofrece soporte para diferentes motores de plantillas, lo que facilita la creación de vistas y la separación de la lógica de presentación del resto de la aplicación.

Slim Framework es conocido por su flexibilidad y su énfasis en la simplicidad. Es una opción popular para aquellos que buscan un marco de trabajo ágil y fácil de usar para desarrollar aplicaciones web y API con PHP.

Para crear un proyecto usando ese framework siga los siguientes pasos:

1. Cree una carpeta en la cual desea almacenar el nuevo proyecto. Se recomienda que esta carpeta se encuentre ubicada en la carpeta principal del servidor web.
2. Haciendo uso del gestor de extensiones composer ingrese el siguiente comando desde la carpeta del proyecto: `composer require slim/slim:"4.*"`
3. Configure el sistema PSR 7 y el Server request. Para esto ejecute el comando `composer require slim/psr7`
4. Cree el archivo `index.php` en la raíz del proyecto.