Programación III

# PHP

* para desarrollo web
* puede ser incrustado en HTML
* es multiplataforma
* es “invisible” al navegador
* cualquier variable es una posición de memoria y comienza con el símbolo $

Api Doc 🡪 se usa para documentar las funciones de PHP (hay un ejemplo en el github de Octavio)

## Arrays

Un array en PHP es un mapa ordenado. Un mapa es un tipo de datos que asocia valores con claves.

### ¿Cómo crear un vector?

$vec= array(1,2,3);

## Arrays asociativos:

$vec= array(“Juan”=>22, “Mercedes”=>12);

var\_dump($vec);

Salida:

array(2)((“Juan”)=> int(22) (“Mercedes”)=> int(12));

## Recomendaciones estándares:

<https://www.php-fig.org/psr/psr-1/>

Clave de classroom: i6hdt55

## include y require

Los hacen lo mismo: cuando tengo dos archivos distintos pegan el código de uno en otro, la diferencia es que require tira error mientras que include tira warning.

Para que el código se incluya una sola vez se utiliza:

require\_once

include\_once

Ejemplo:

uno.php

<?php

$dato= "hola";

require\_once "dos.php";

?>

dos.php

<?php

echo $dato;

?>

Registrarse en un servidor gratuito

Por ejemplo: 00webhost

Ver los ejercicios ahí

# FUNCIONES Y ARCHIVOS

## Funciones propias en PHP

**function** NombreFuncion($param\_1, $param\_2= “valor”){

}

## Incluir/ requerir archivos

Copia todo el código existente del archivo especificado dentro del archivo que posee dicha declaración.

La declaración include y requireson idénticas, excepto en caso de falla

* Require producirá un error fatal (E\_COMPILE\_ERROR) y frenará el script.
* Include solo producirá una advertencia (E\_WARNING) y el script continuará.

### Ejercicio

1. PRODUCTO – CONTEINER  
   Los conteiner pueden ser (chico, mediano o grande con una capacidad de 1000 kg, 2500 kg o 9000 kg.  
   Los productos tienen un identificador único de producto, el nombre, el importador, el país de origen y los kilos.  
   Si el producto ya existe se suman los kilos.
   1. Agregar()
   2. BorrarProducto()
   3. SacarKilos()
   4. Modificar() 🡪 algo
   5. Mostrar()

## Operador de Resolución de Ámbito (::)

El Operador de Resolución de Ámbito (también denominado Paamayim Nekudotayim) o en términos simples, el doble dos-puntos, es un token que permite acceder a elementos [estáticos](https://www.php.net/manual/es/language.oop5.static.php), [constantes](https://www.php.net/manual/es/language.oop5.constants.php), y sobrescribir propiedades o métodos de una clase.

Cuando se hace referencia a estos ítems desde el exterior de la definición de la clase, se utiliza el nombre de la clase.

A partir de PHP 5.3.0, es posible hacer referencia a una clase usando una variable. El valor de la variable no puede ser una palabra clave (por ej., ***self***, *parent* y *static*).

Paamayim Nekudotayim significa doble dos-puntos - en Hebreo!

### Ejemplo #1 :: desde el exterior de la definición de la clase

<?php  
class MyClass {  
    const CONST\_VALUE = 'Un valor constante';  
}  
  
$classname = 'MyClass';  
echo $classname::CONST\_VALUE; // A partir de PHP 5.3.0  
  
echo MyClass::CONST\_VALUE;  
?>

Las tres palabras claves especiales ***self*, *parent* y *static*** son utilizadas para acceder a propiedades y métodos desde el interior de la definición de la clase.

### Ejemplo #2 :: desde el interior de la definición de la clase

<?php  
class OtherClass extends MyClass  
{  
    public static $my\_static = 'variable estática';  
  
    public static function doubleColon() {  
        echo parent::CONST\_VALUE . "\n";  
        echo self::$my\_static . "\n";  
    }  
}  
$classname = 'OtherClass';  
$classname::doubleColon(); // A partir de PHP 5.3.0  
  
OtherClass::doubleColon();  
?>

Cuando una clase extendida sobrescribe la definición parent de un método, PHP no invocará al método parent. Depende de la clase extendida el hecho de llamar o no al método parent. Esto también se aplica a definiciones de métodos [Constructores y Destructores](https://www.php.net/manual/es/language.oop5.decon.php), [Sobrecarga](https://www.php.net/manual/es/language.oop5.overloading.php), y [Mágicos](https://www.php.net/manual/es/language.oop5.magic.php).

### Ejemplo #3 Invocando a un método parent

<?php  
class MyClass  
{  
    protected function myFunc() {  
        echo "MyClass::myFunc()\n";  
    }  
}  
  
class OtherClass extends MyClass  
{  
    // Sobrescritura de definición parent  
    public function myFunc()  
    {  
        // Pero todavía se puede llamar a la función parent  
        parent::myFunc();  
        echo "OtherClass::myFunc()\n";  
    }  
}  
  
$class = new OtherClass();  
$class->myFunc();  
?>

## API REST

POST index.php CargarProducto.php Conteiner.php

$id= $POST[]

Conteiner.Cargar($id);

Cargar($id);

id

include CargarProducto.php

## POSTMAN

### MÉTODO POST

<?php

$numero1=0;

$numero2=0;

$numero1= $\_POST['num1'];

$numero2= $\_POST['num2'];

$operacion= $\_POST['operacion'];

switch ($operacion) {

case 1:

echo "El resultado de la suma es: " . ($numero1 + $numero2);

break;

case 2:

echo "El resultado de la resta es: " . ($numero1 - $numero2);

break;

case 3:

echo "El resultado de la multiplicacion es: " . ($numero1 \* $numero2);

break;

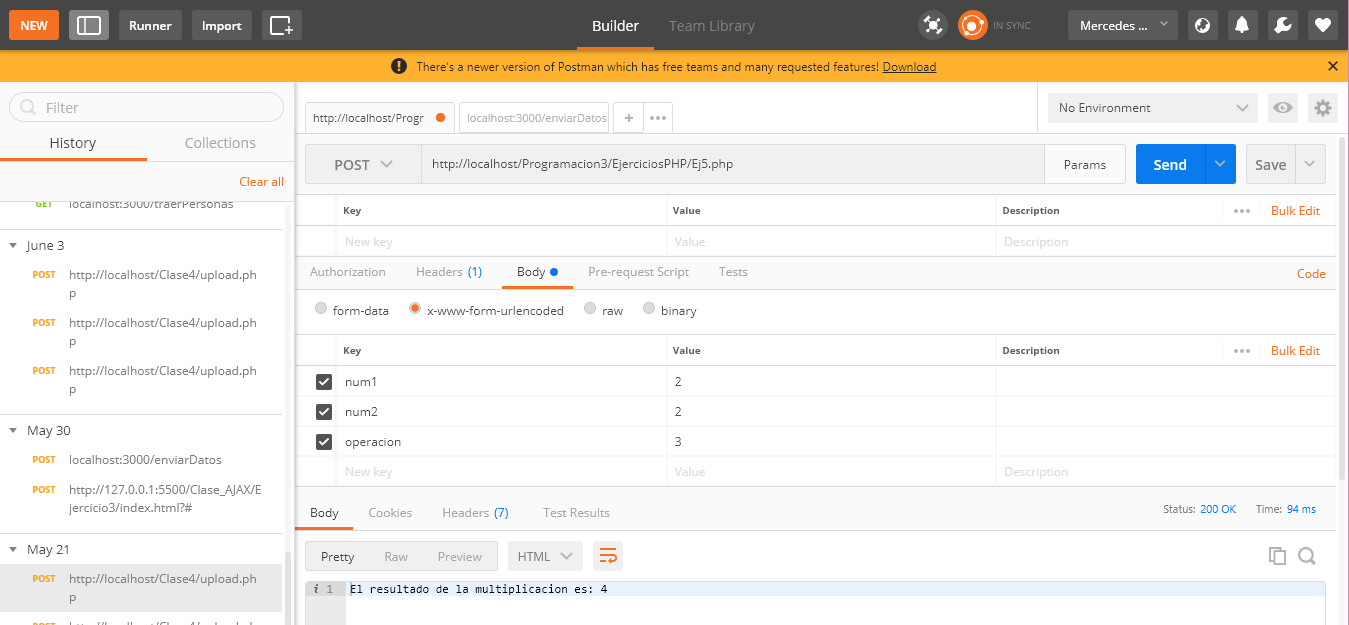
case 4:

echo "El resultado de la division es: " . ($numero1 / $numero2);

break;

}

?>



## ARCHIVOS

E/S con Archivos

1. fopen (abrir)
2. fclose (cerrar)
3. fread/fgets (leer)
4. fwrite/fputs (escribir)
5. copy
6. unlink (elimina)
7. upload file

int fopen devuelve un entero

<?php

int $ar;

$ar= fopen(archivo, modo);

fclose($ar)

?>

|  |  |
| --- | --- |
| Modo | Descripción |
| r | Abre un archivo para sólo lectura. El cursor comienza al principio del archivo. |
| w | Abre un archivo para sólo escritura. Si no existe, crea uno nuevo. Si existe, borra el contenido. |
| a | Abre un archivo para sólo escritura. Si no existe, crea uno nuevo. Si existe, mantiene el contenido. El cursor comienza en el final del archivo. |
| x | Crea un nuevo archivo para sólo lectura. Retorna FALSE y un error si el archivo existe. |
| r+ | Abre un archivo para lectura/escritura. Ídem r. |
| w+ | Abre un archivo para lectura/escritura. Ídem w. |
| a+ | Abre un archivo para lectura/escritura. Ídem a. |
| x+ | Crea un archivo para lectura/escritura. Ídem x. |

### Envío de datos

¿Cómo proteger las páginas? .htacces

A la única página que le podemos pegar con post es index y nos va a devolver datos

#### HTTP

#### Método GET

El par de nombres/ valores es enviado en la dirección URL.

Protocolo http x GET

#### Método POST

POST 🡪encriptado

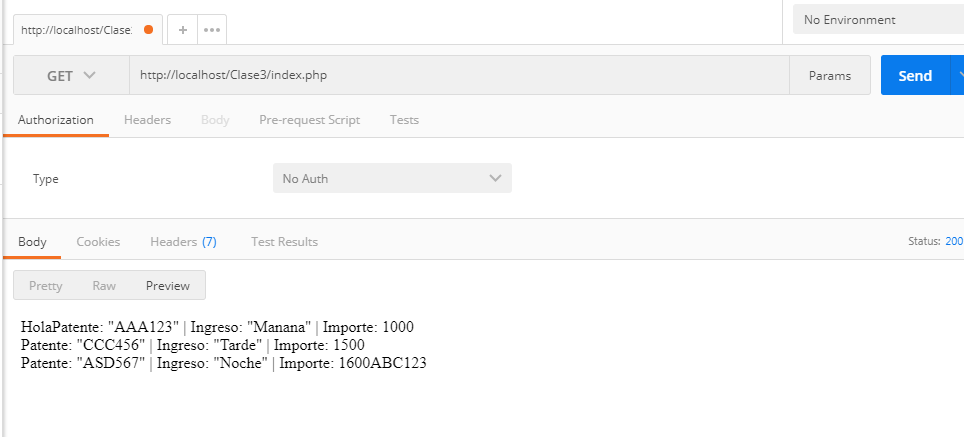
Manda los datos en el cuerpo del mensaje

#### Manejo de Formularios

Tanto GET como POST crean un array asociativo.

Para guardar en archivos:

$arraydeDatos= implode(',', $unVehiculo->toArray()); o JSON encode



Ejemplo🡪 <https://github.com/octaviovillegas/estacionamiento>

##### Para saber por qué método le está pegando Postman:

$metodo= $\_SERVER['REQUEST\_METHOD'];

echo $metodo;

Ruteo cuando le pegamos a una API 🡪 le pegamos a la misma URL de distinta manera

* GET 🡪devuelve
* POST 🡪 ALTA
* DELETE🡪 ELIMINAR
* PUT🡪 MODIFICAR

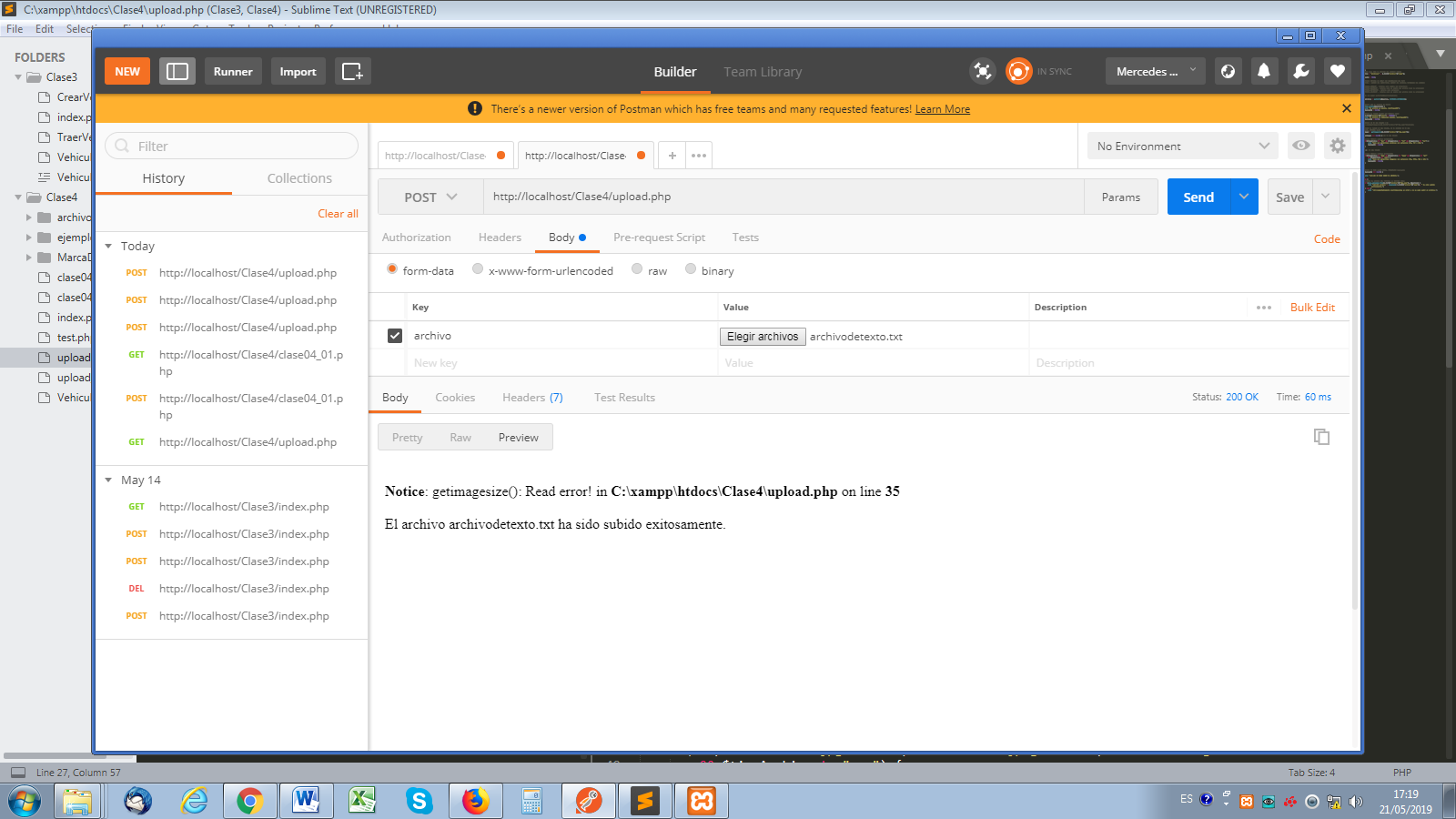
Se usa un switch para identificar cada método

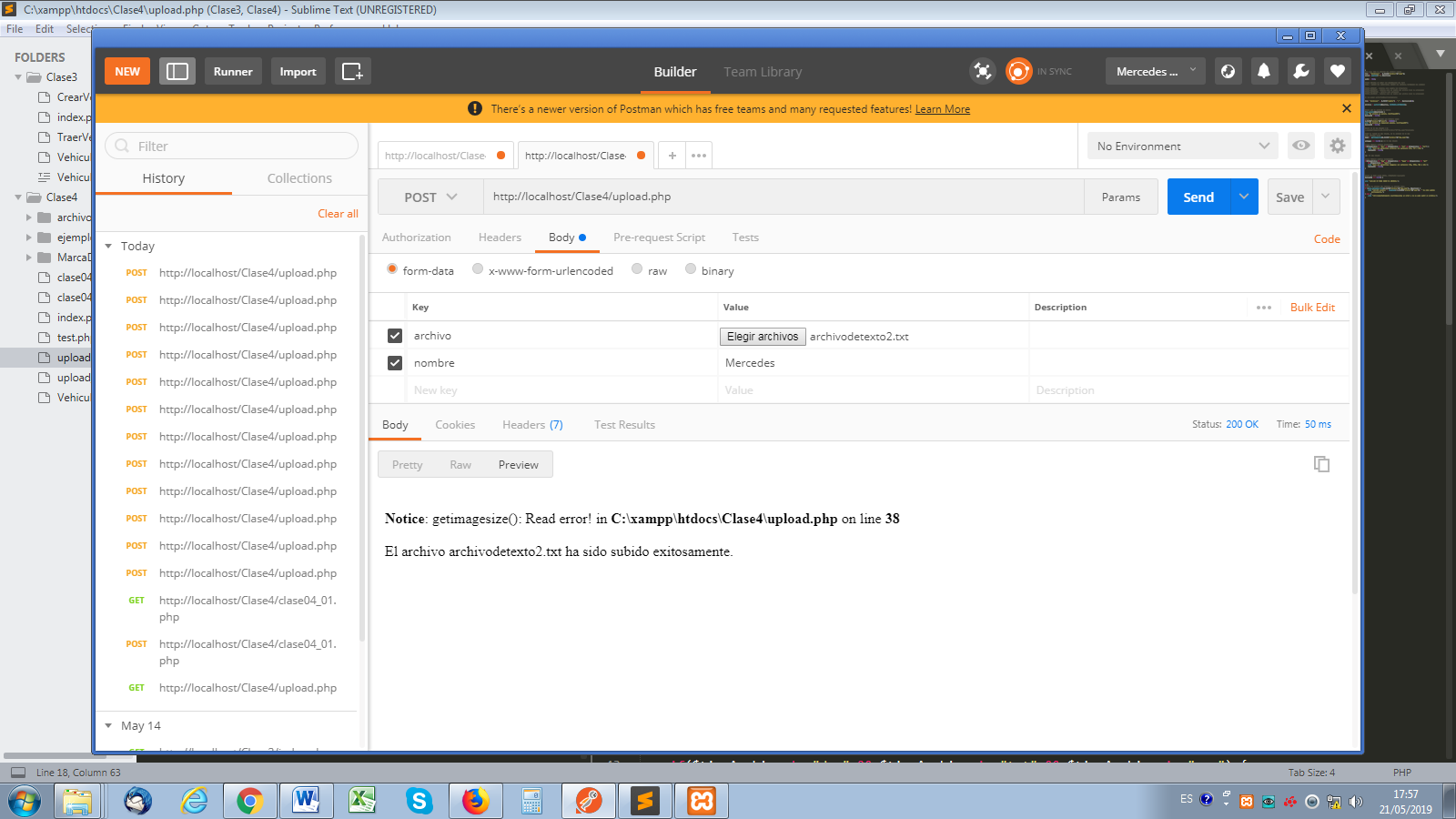
En el mismo ejercicio agregar una foto

Se le puede poner marca de agua

Cada vez que la persona cambia su foto se guarda en /backup con la fecha

Subir un archivo usando postman





# En el Parcial

Hacer el estacionamiento!!!!!!!

1. ABM listado, filtrado (ej: productos más vendidos) y facturación
2. Fotos con nombre… seleccionar una foto o habilitar web cam(està en el github de octavius)
   1. Backup
   2. Poner marca de agua
3. Base de datos my SQL

# TP LA COMANDA

En Github

# SQL

¿Por qué se crearon las bases de datos?

Son posiciones de memoria de datos. Es como un archivo, cambia la arquitectura que tiene para obtener los datos.

A🡪 instrucción INSERT

B🡪 DELETE (no olvidar el where)

M🡪 UPDATE (no olvidar el where)

Select/ listar 🡪 Relación entre las tablas

Una tabla puede representar:

Un objeto

Más de un objeto

Una parte de un objeto

Referencia 🡪 tabla –clase

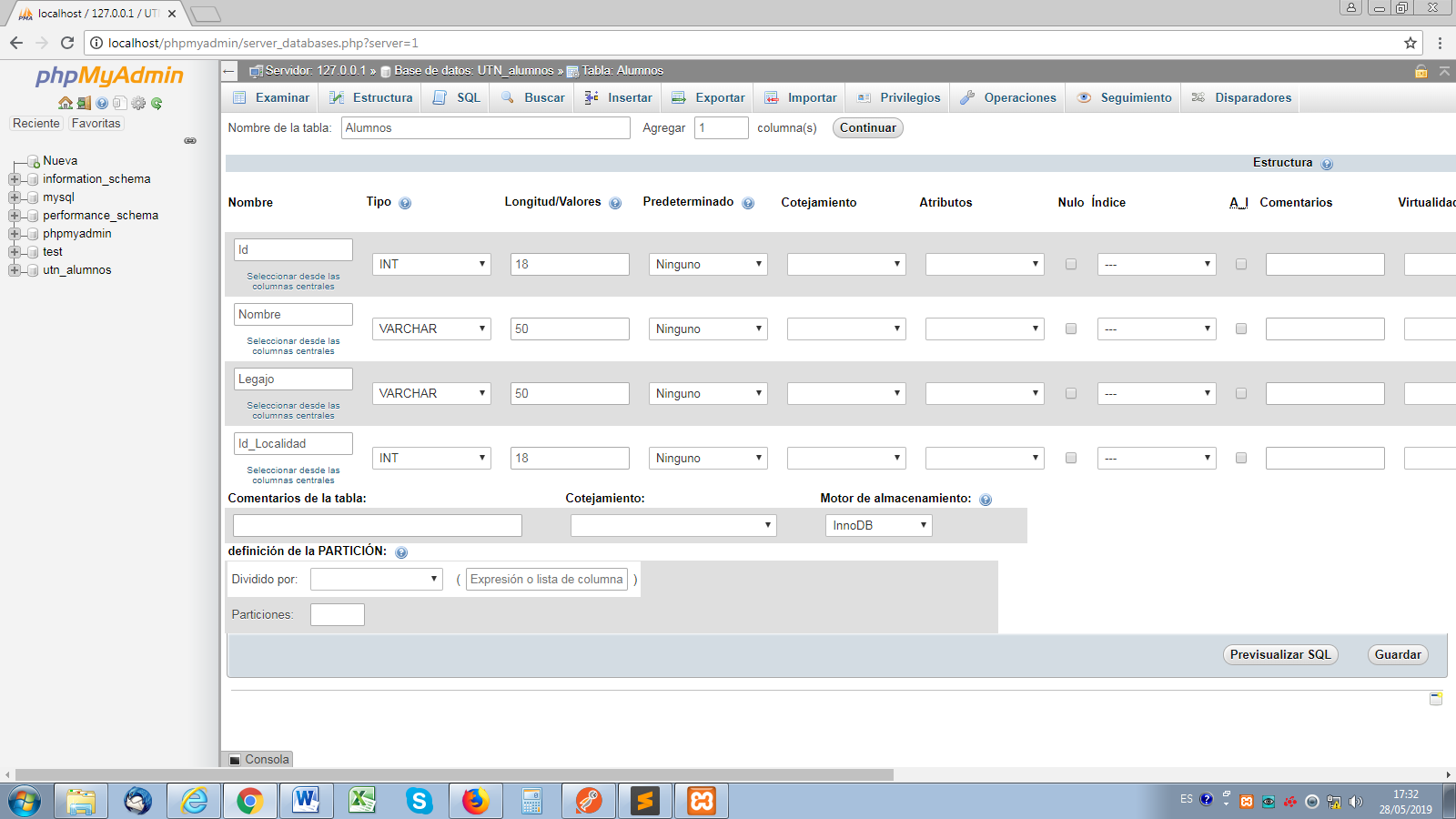
Las vistas de la tabla son lo que le devolvemos al cliente.

La vista es un select que permanece en la base de datos. Qué datos, como me los va a traer y con qué filtro.

Formas normales🡪 evitan que los datos se repitan en las bases de datos

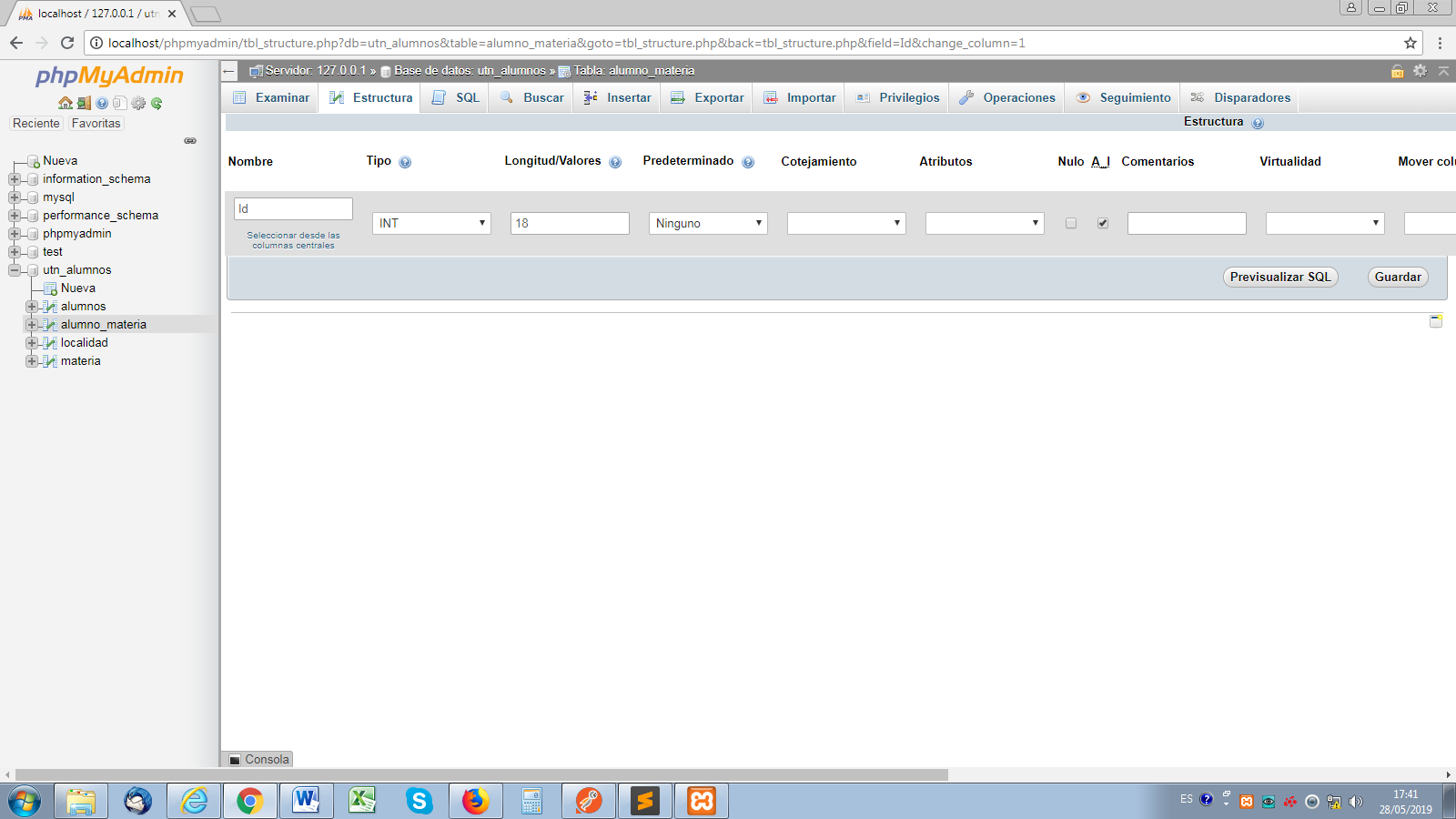
Tabla intermedia que asocia los campos. Primary key🡪 1 sola

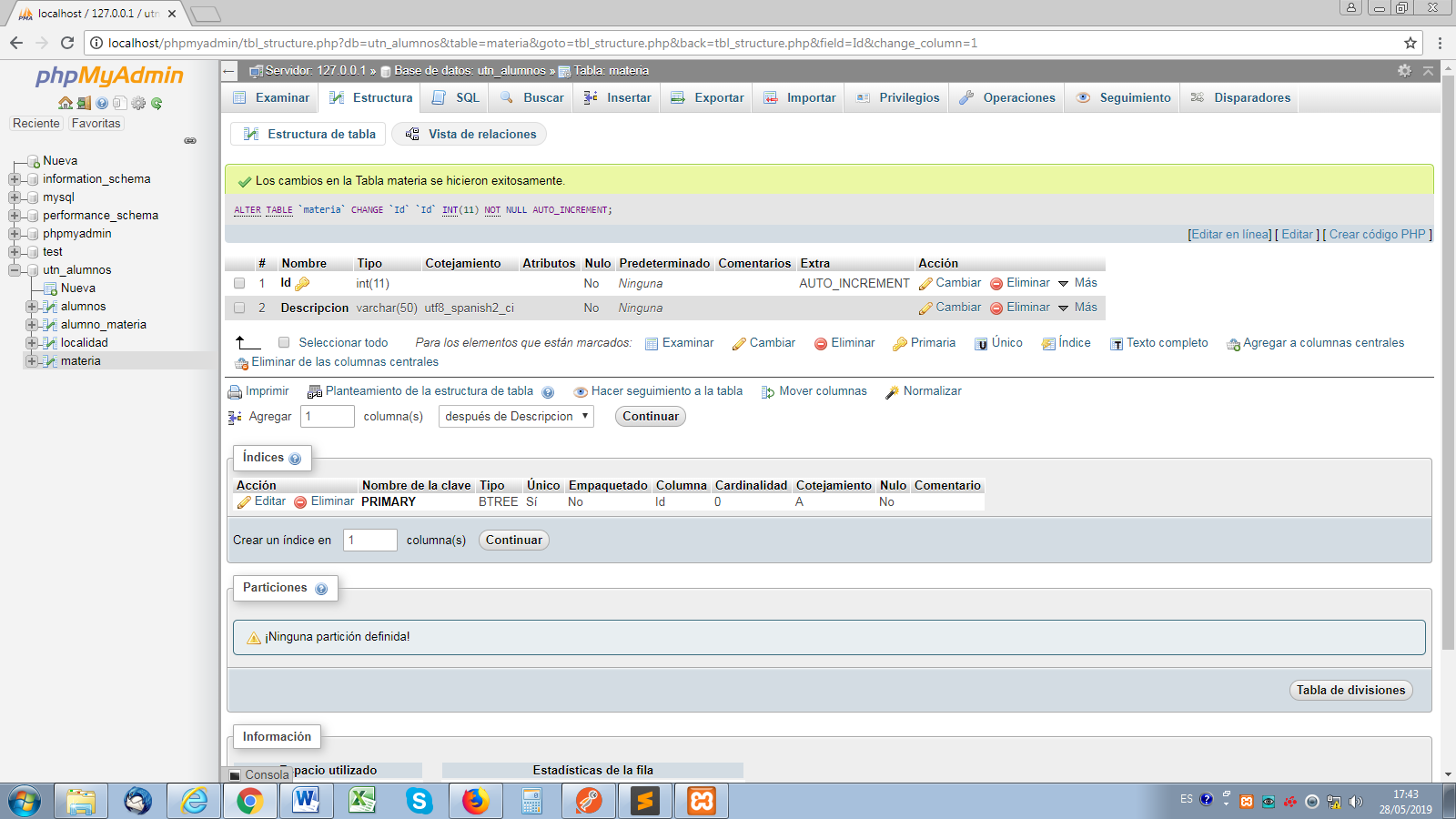
Entonces hay que crear una tabla



Fecha🡪 tymestam y como valor predeterminado el current time spam

Autoincremental



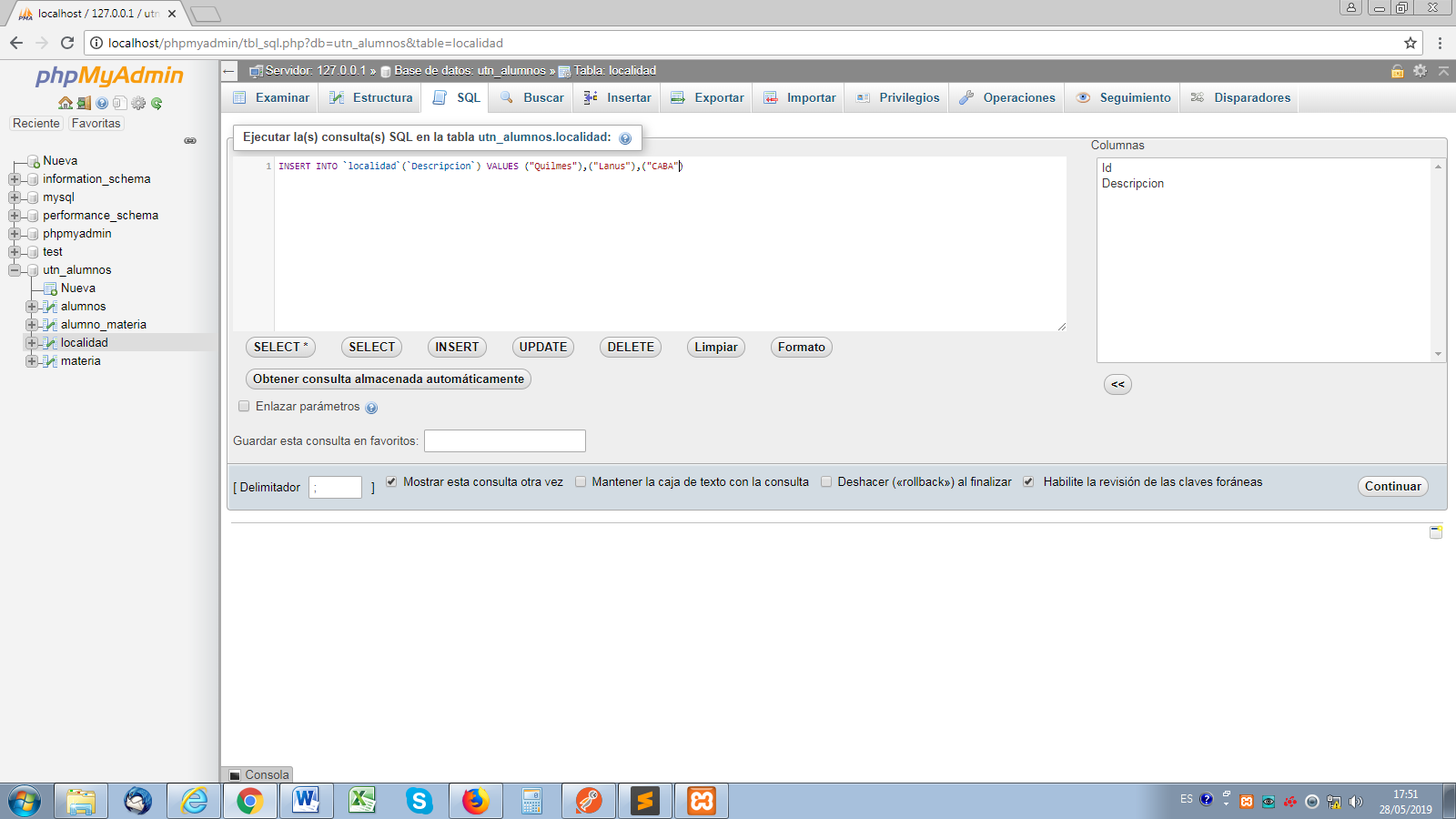


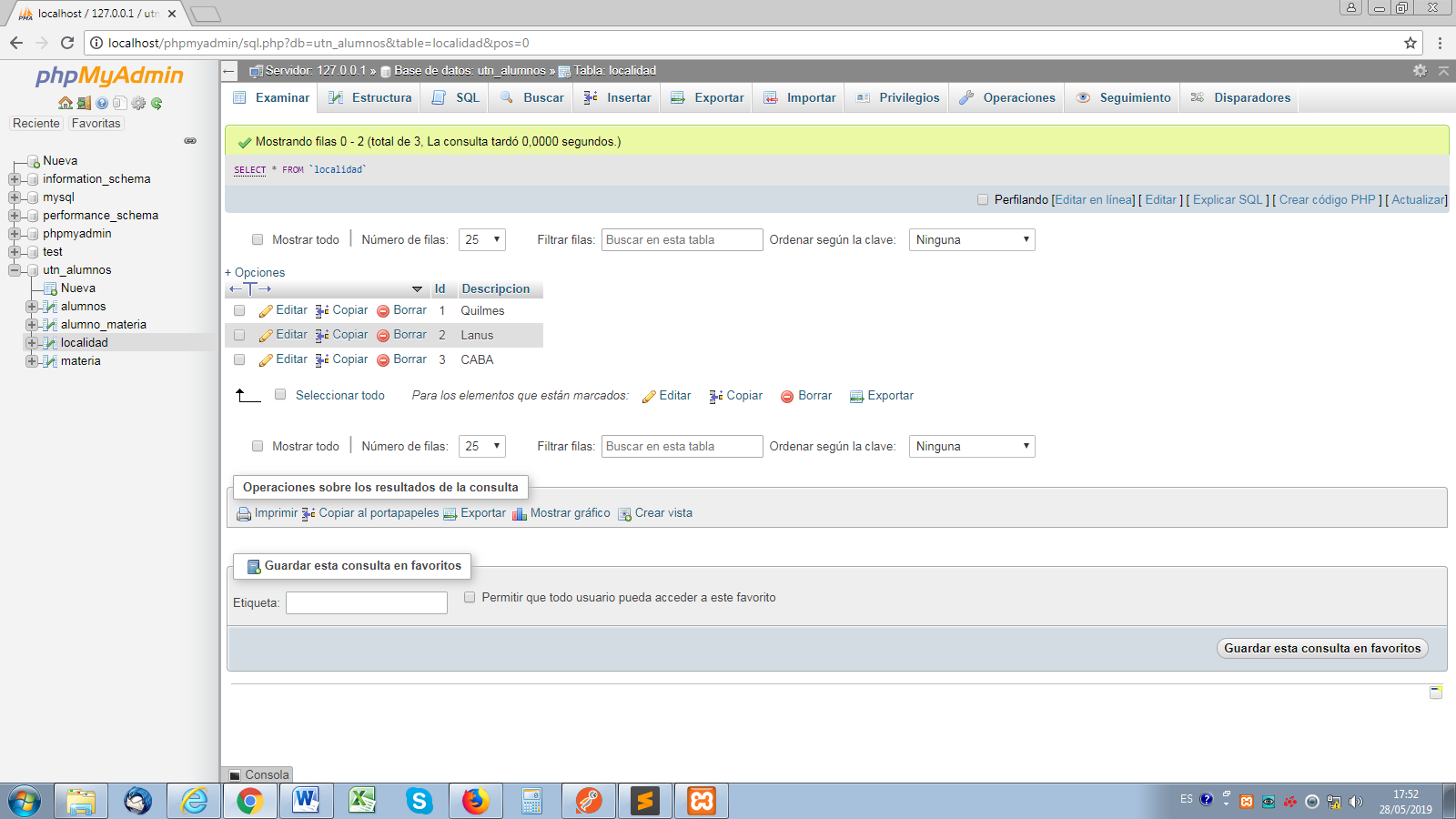
## Insert

Define a través de los paréntesis que le vamos a insertar y en qué orden

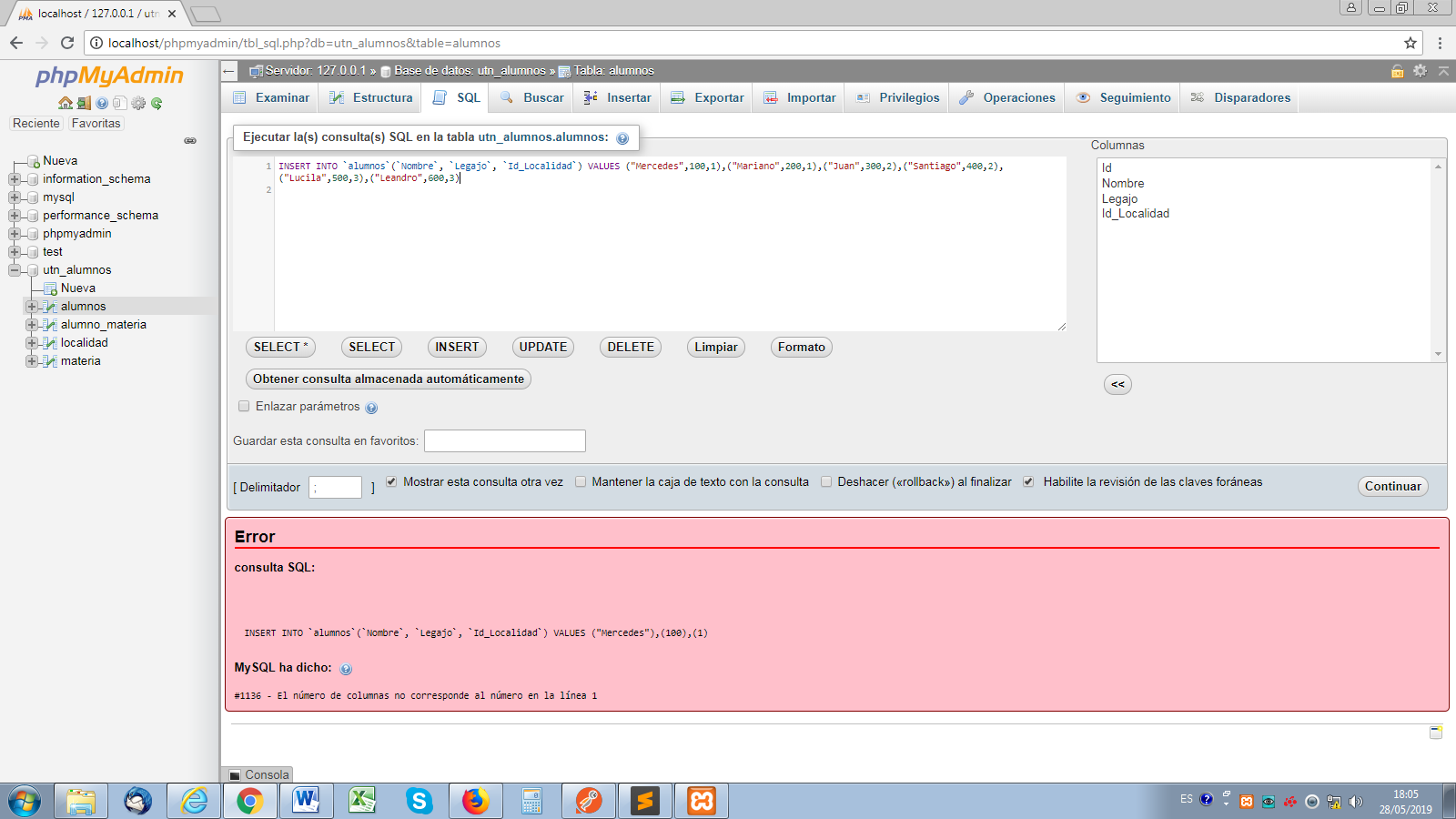
Insert into()

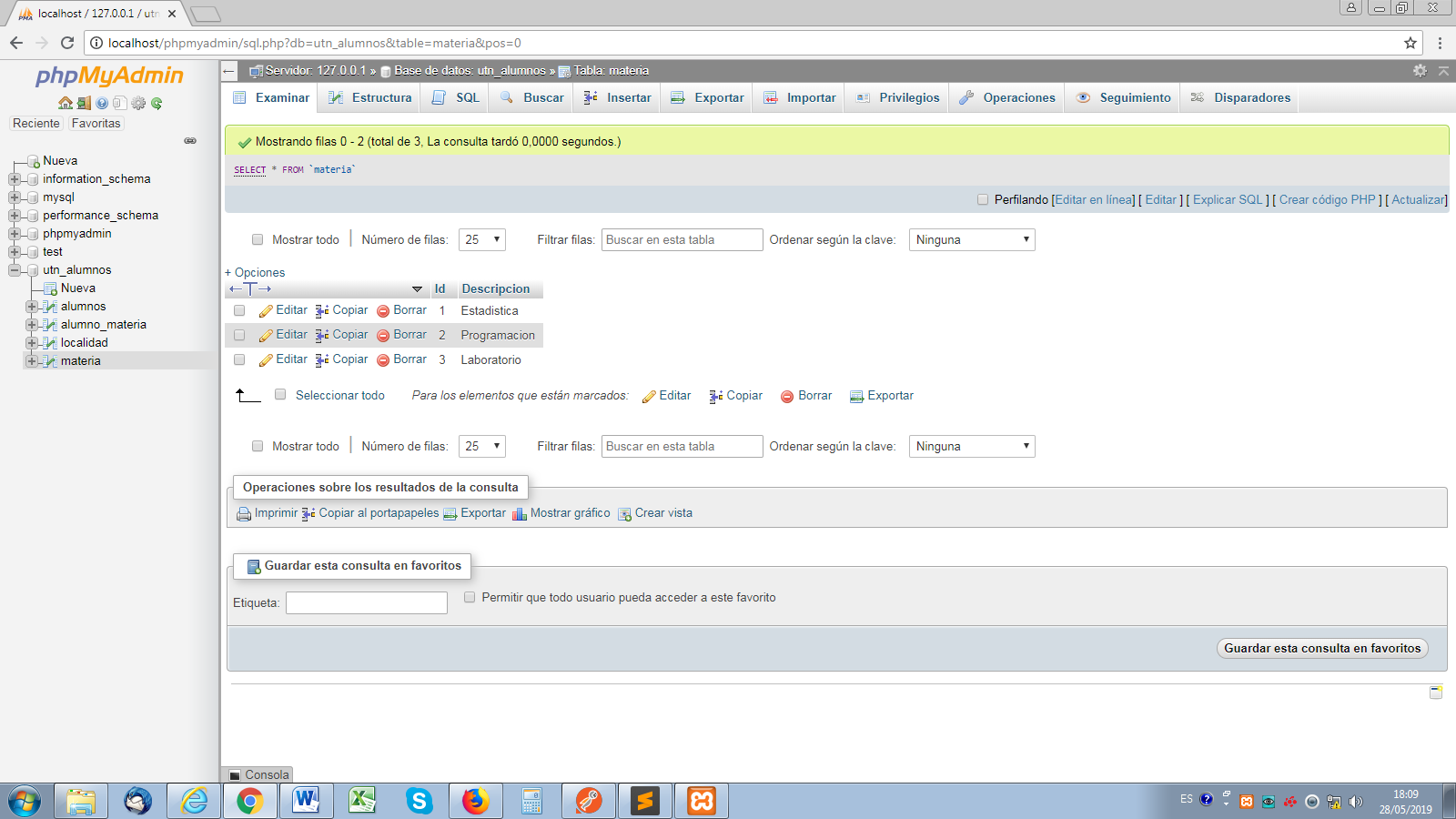
values(),(),()



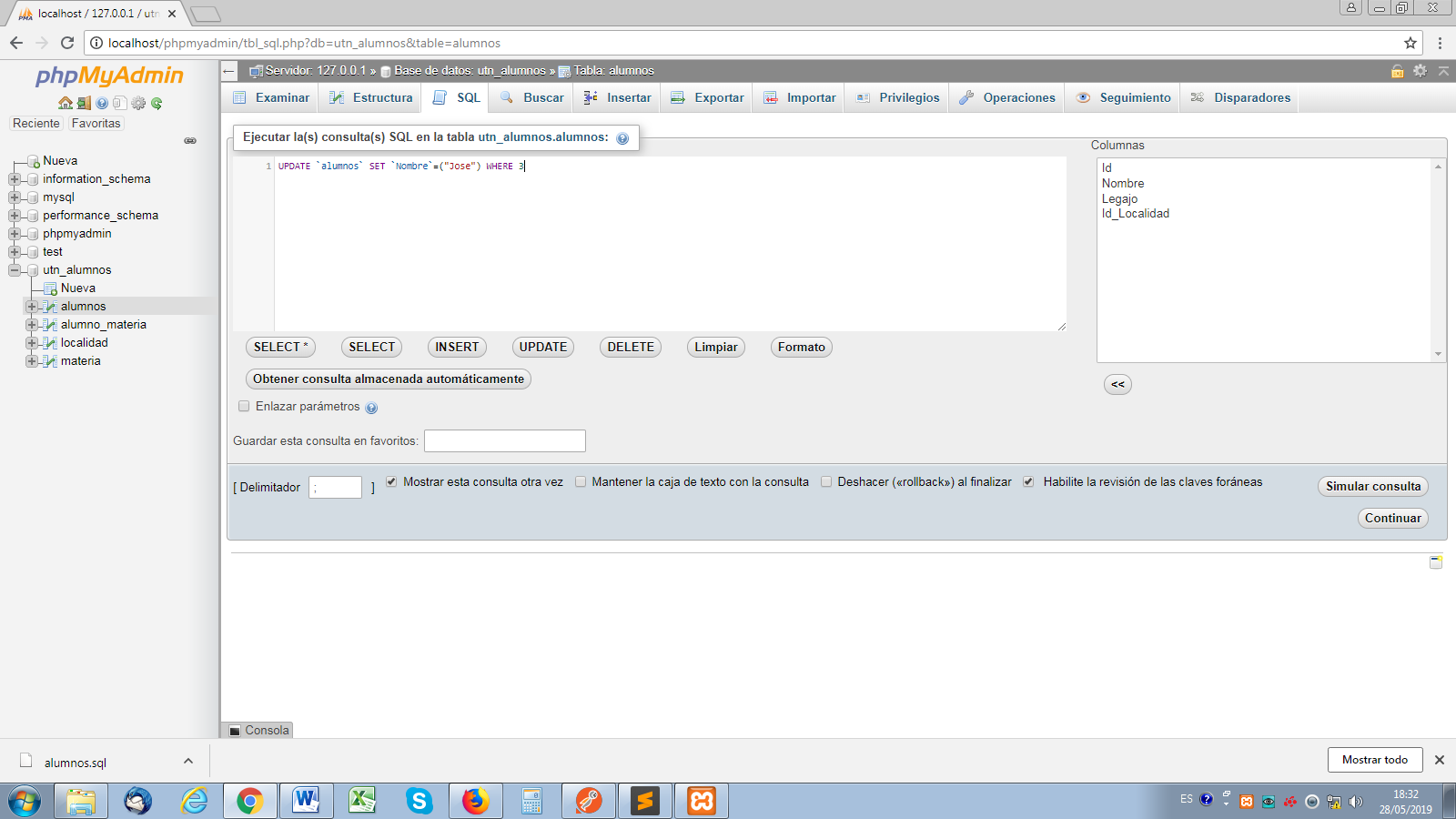


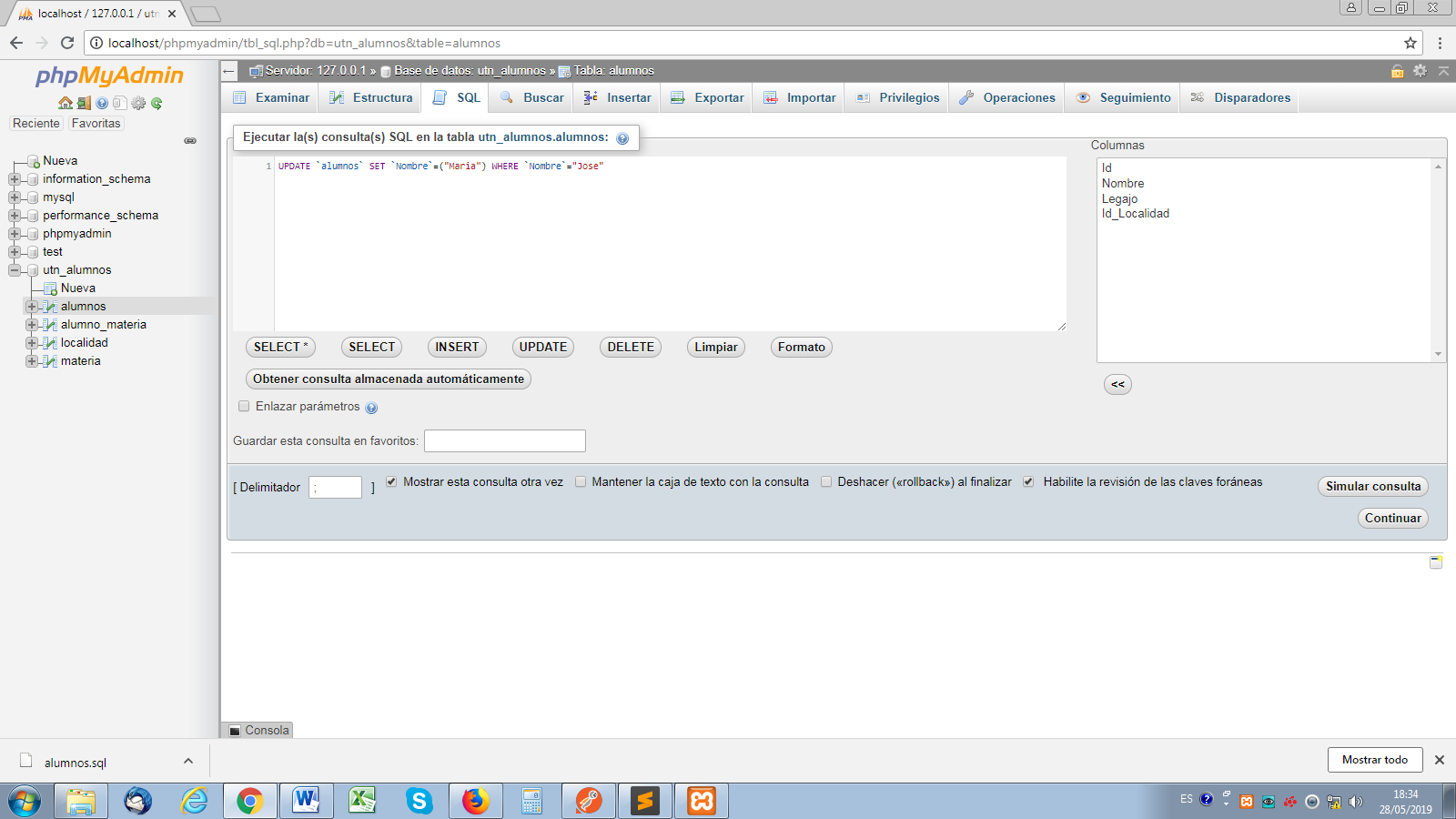
Insert varios valores a la vez





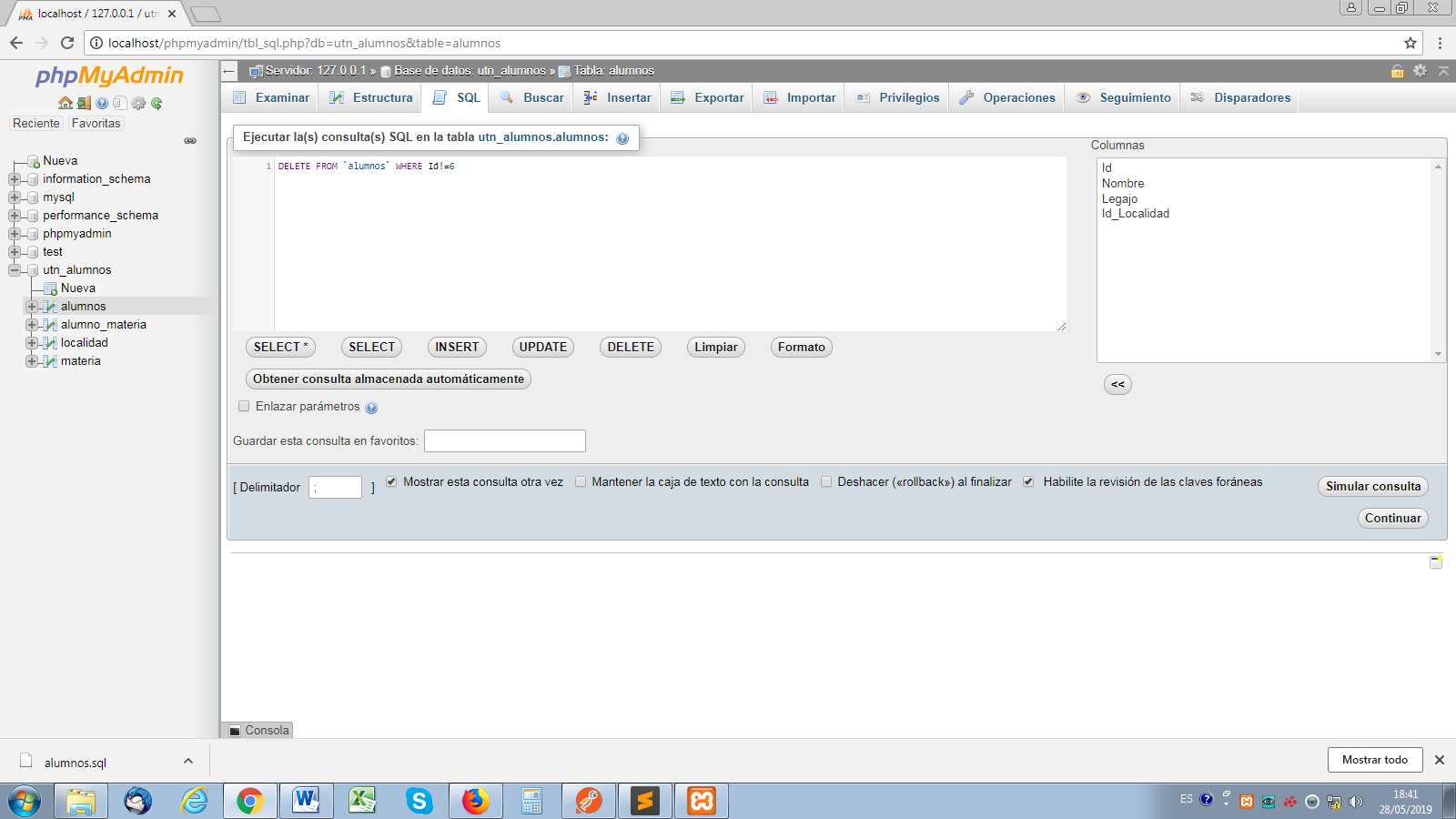
Cambiar el nombre de todos los alumnos:





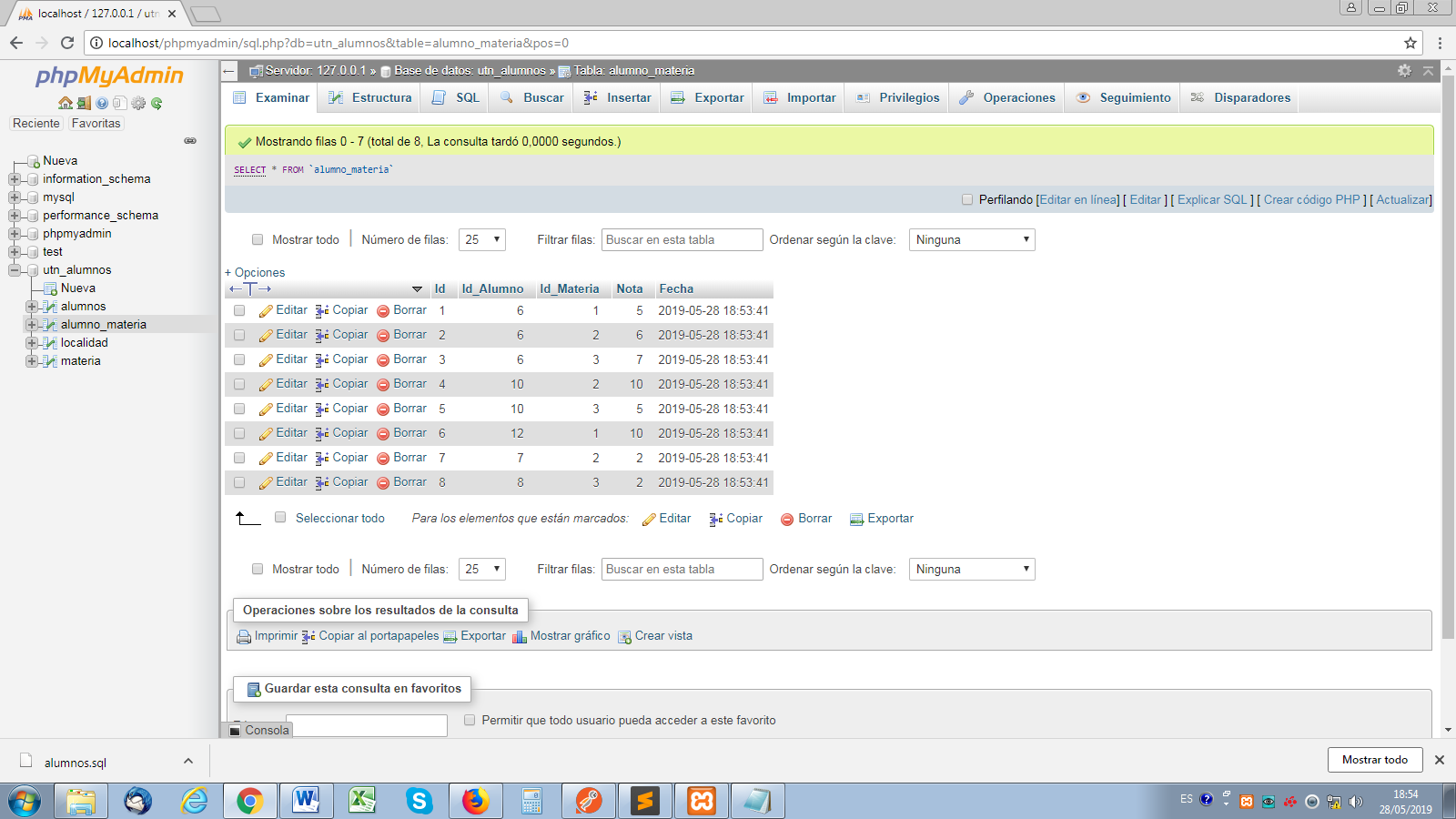
En el where puede haber otra tabla

El signo != tamb puede ser <>

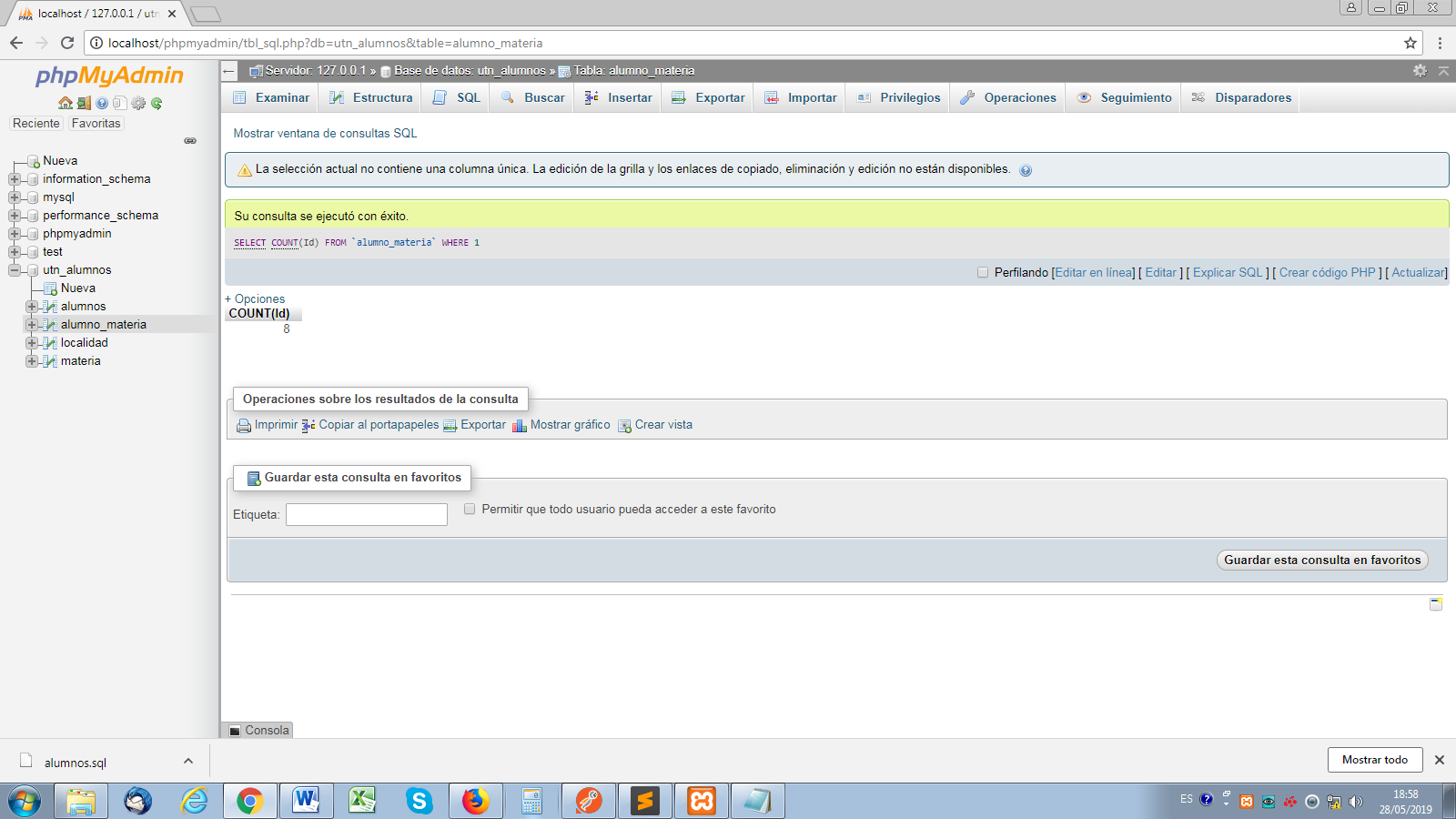


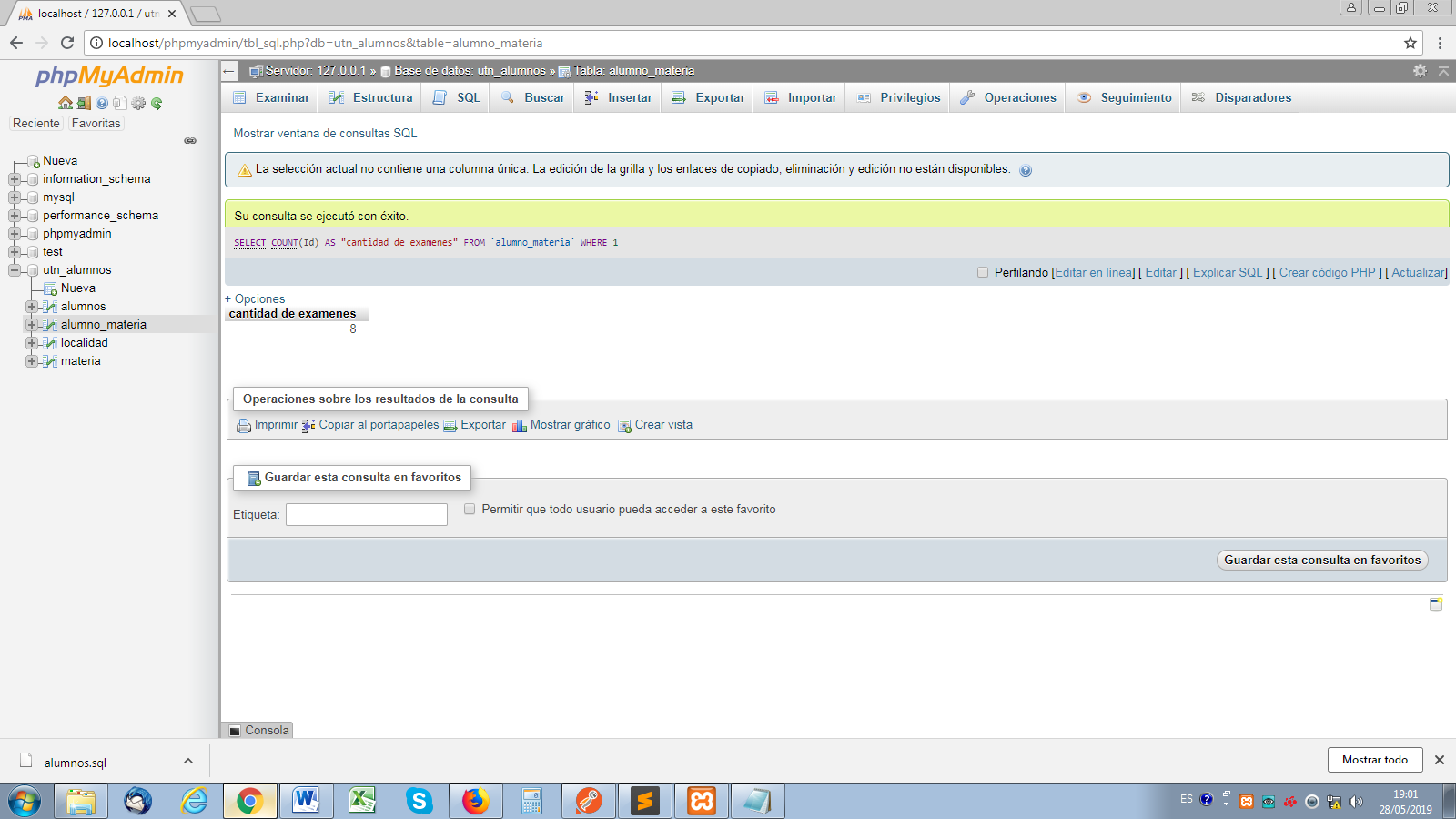
Agregamos más alumnos con INSERT

INSERT INTO `alumno\_materia`(`Id\_Alumno`, `Id\_Materia`, `Nota`) VALUES (6,1,5),(6,2,6),(6,3,7),(10,2,10),(10,3,5),(12,1,10),(7,2,2),(8,3,2)

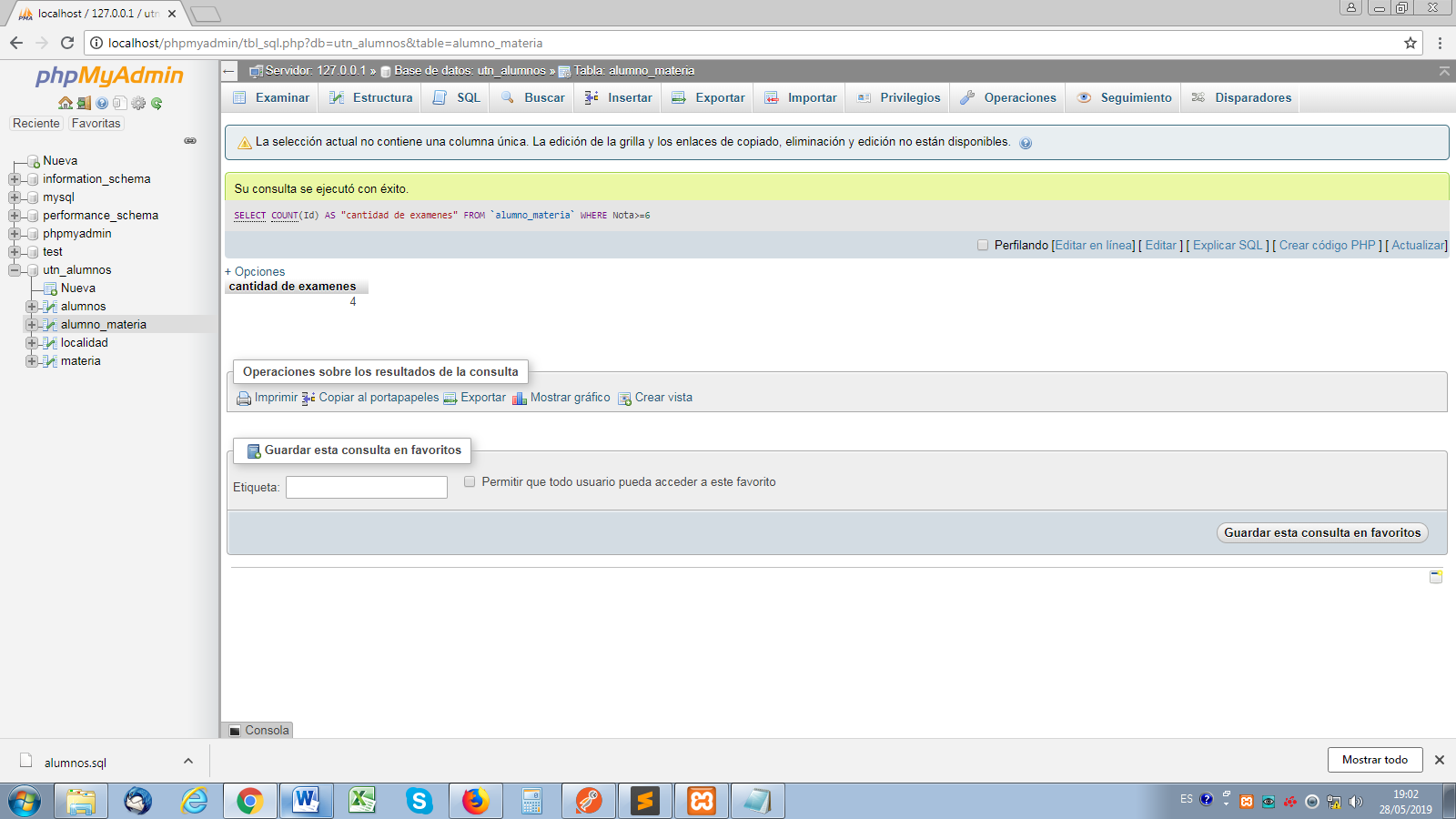


## CONSULTAS 🡪 SELECT

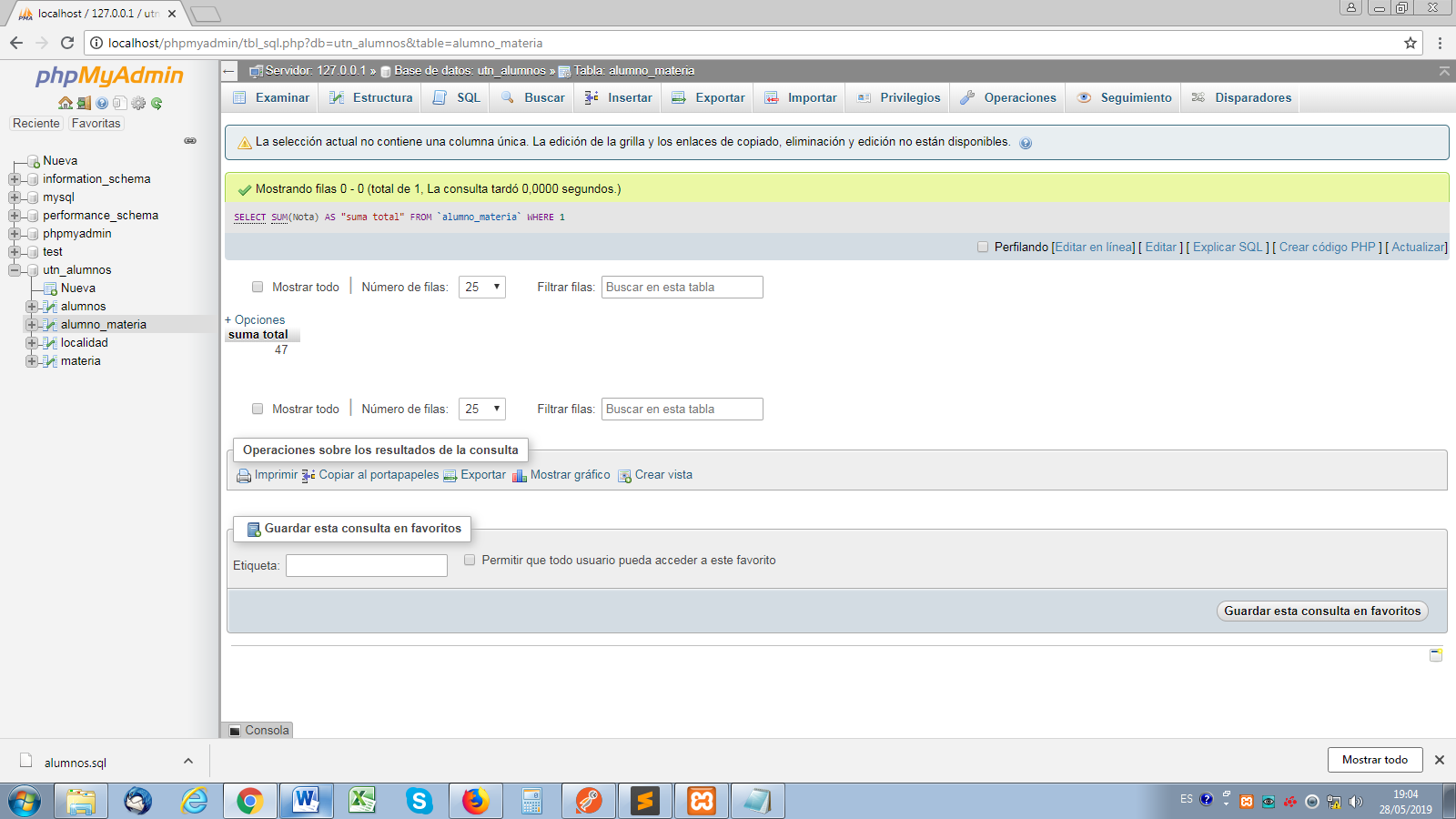




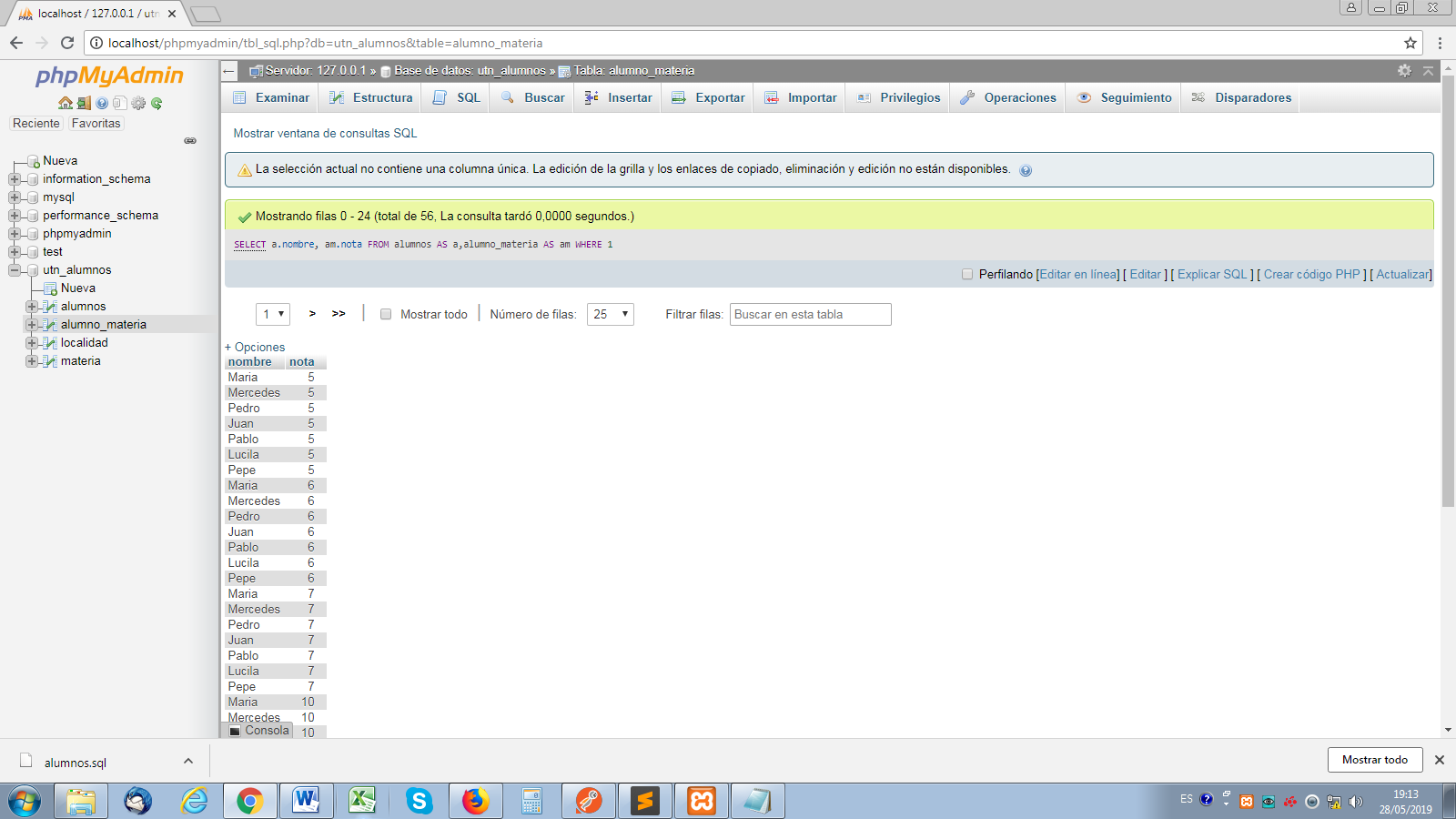
Traer los aprobados



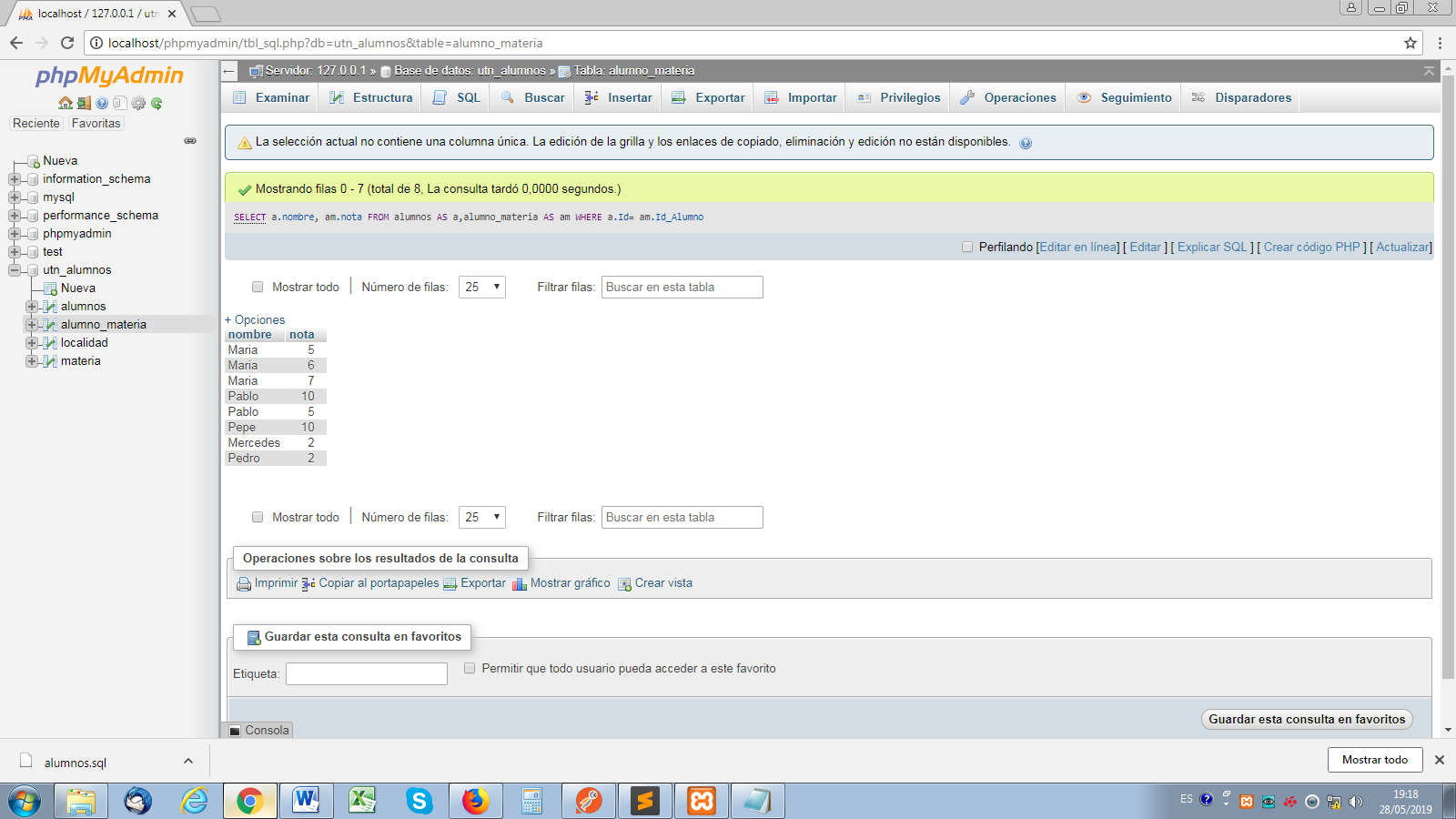
Para saber la suma de notas, el promedio es AVG



Traer el nombre y la nota:

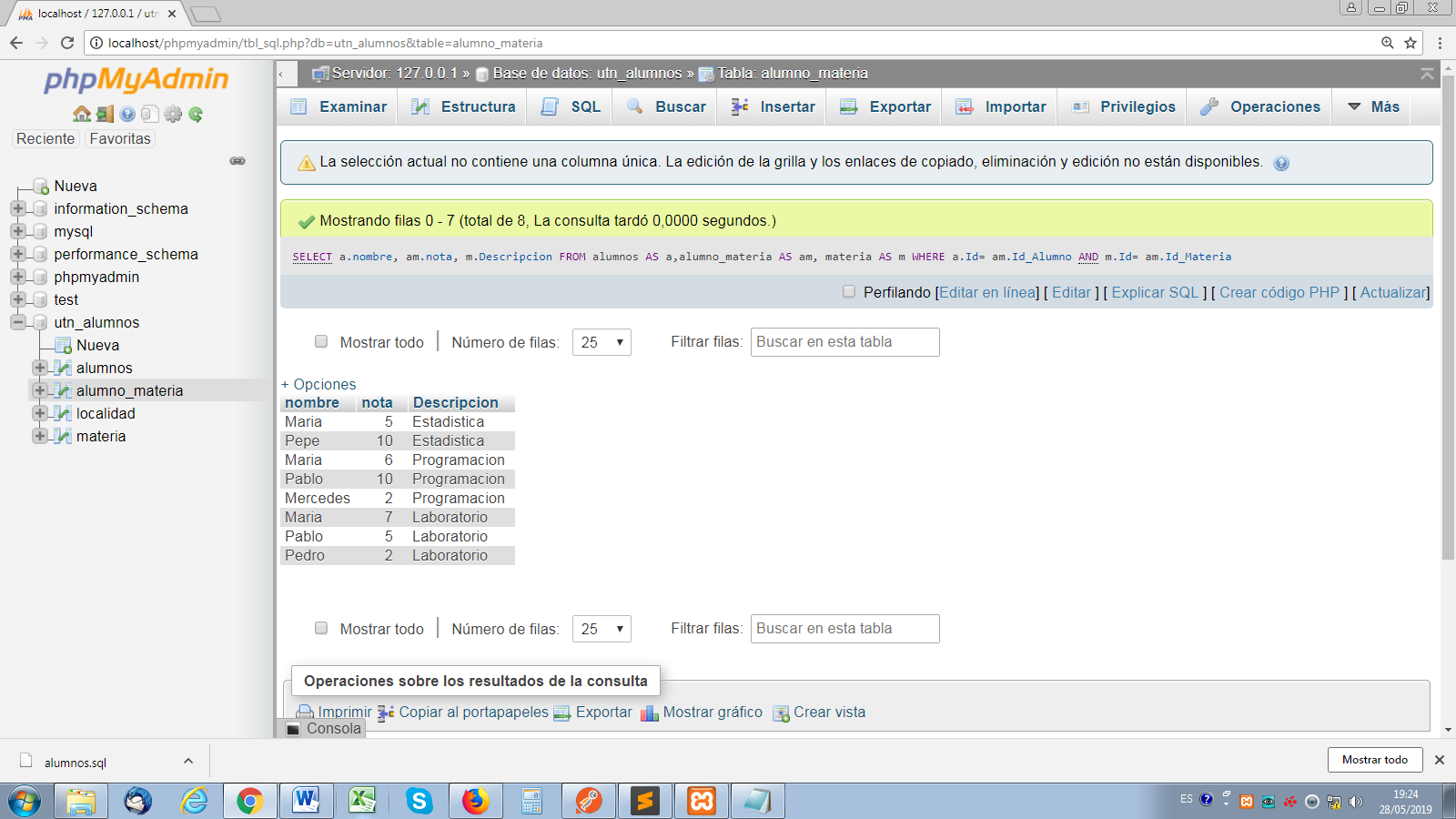


Mal! Me trae el producto cartesiano

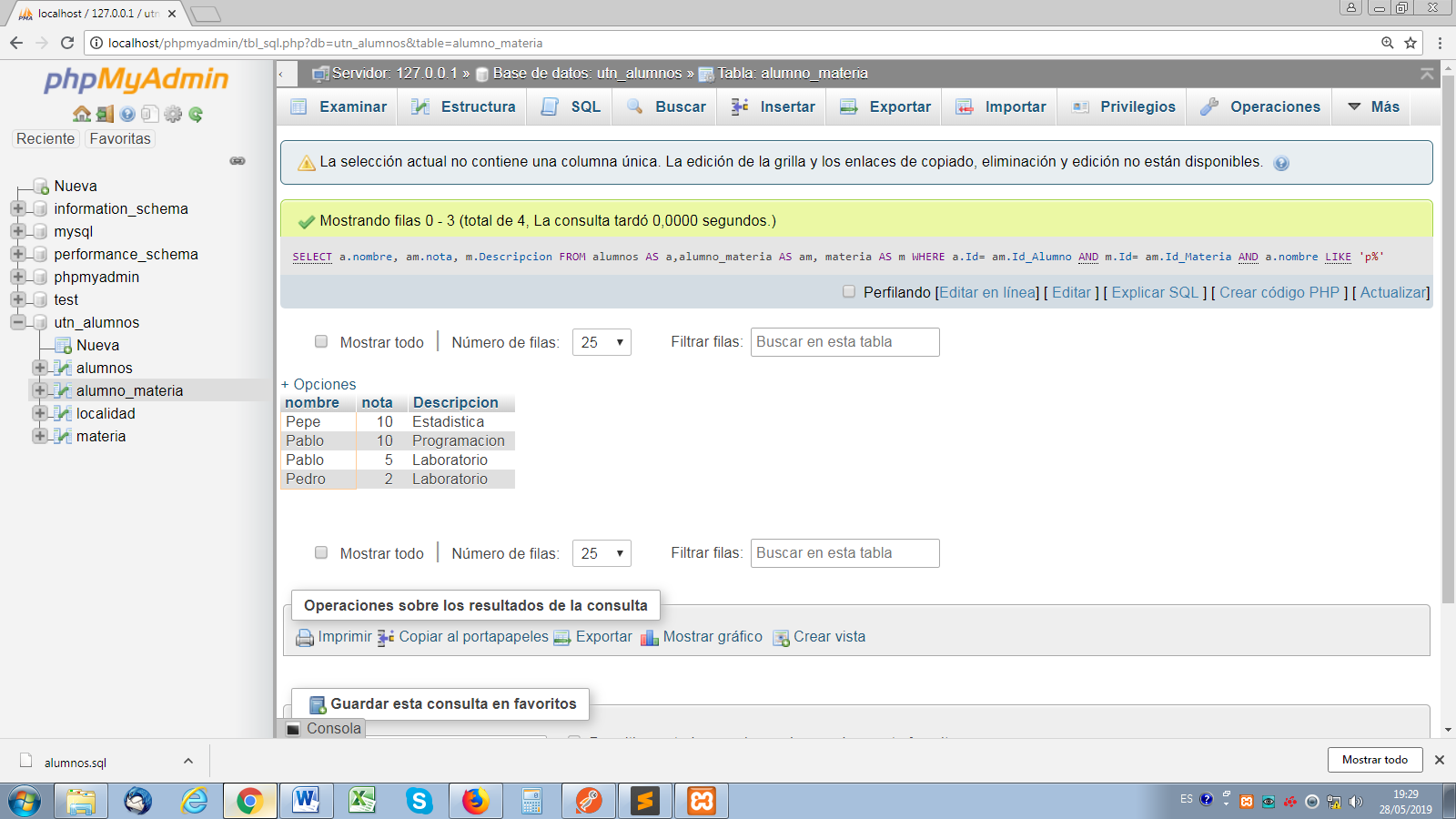


Bien!!!

Para traer nombre, nota y materia

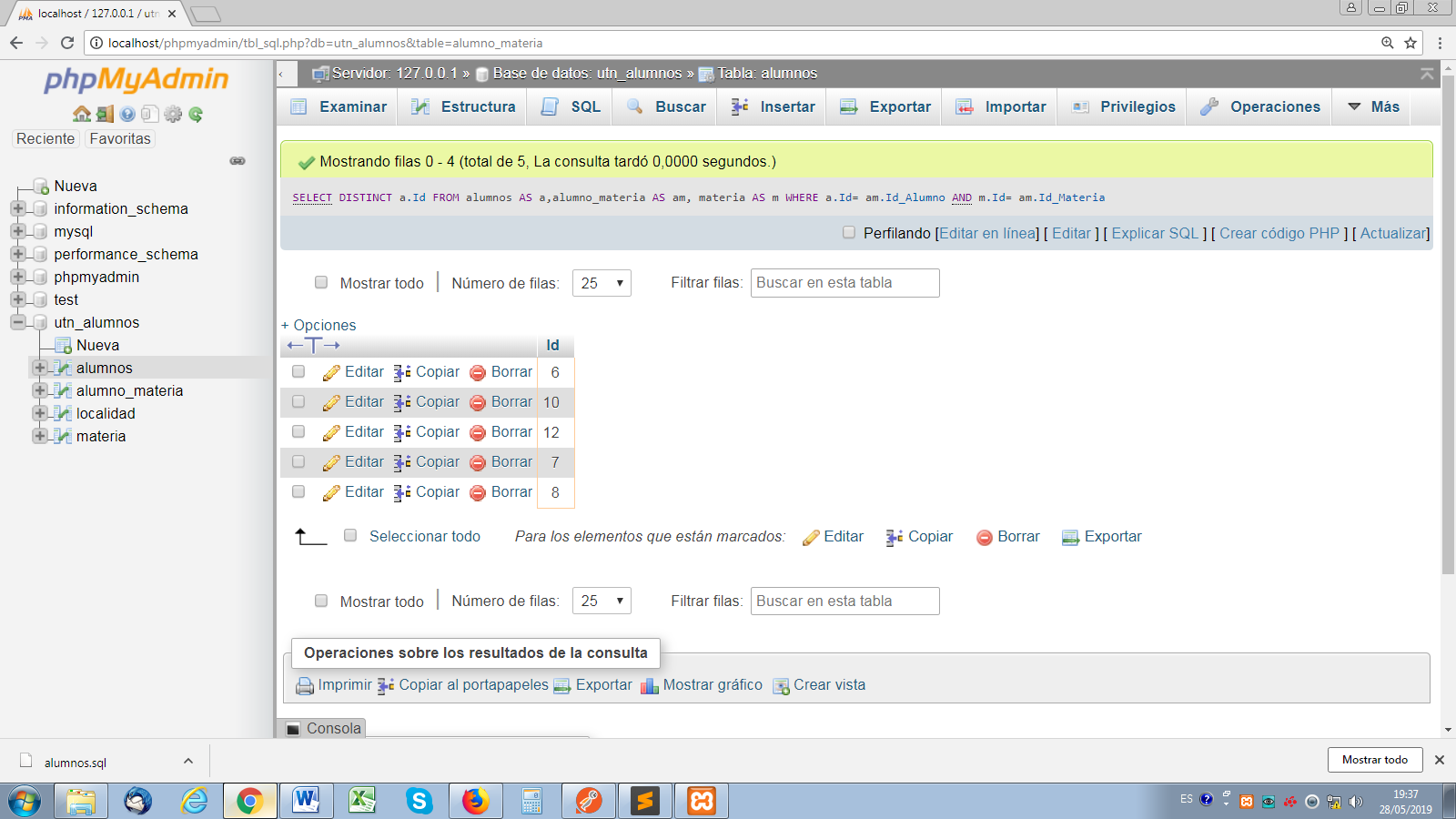


Todos los nombres que empiezan con P

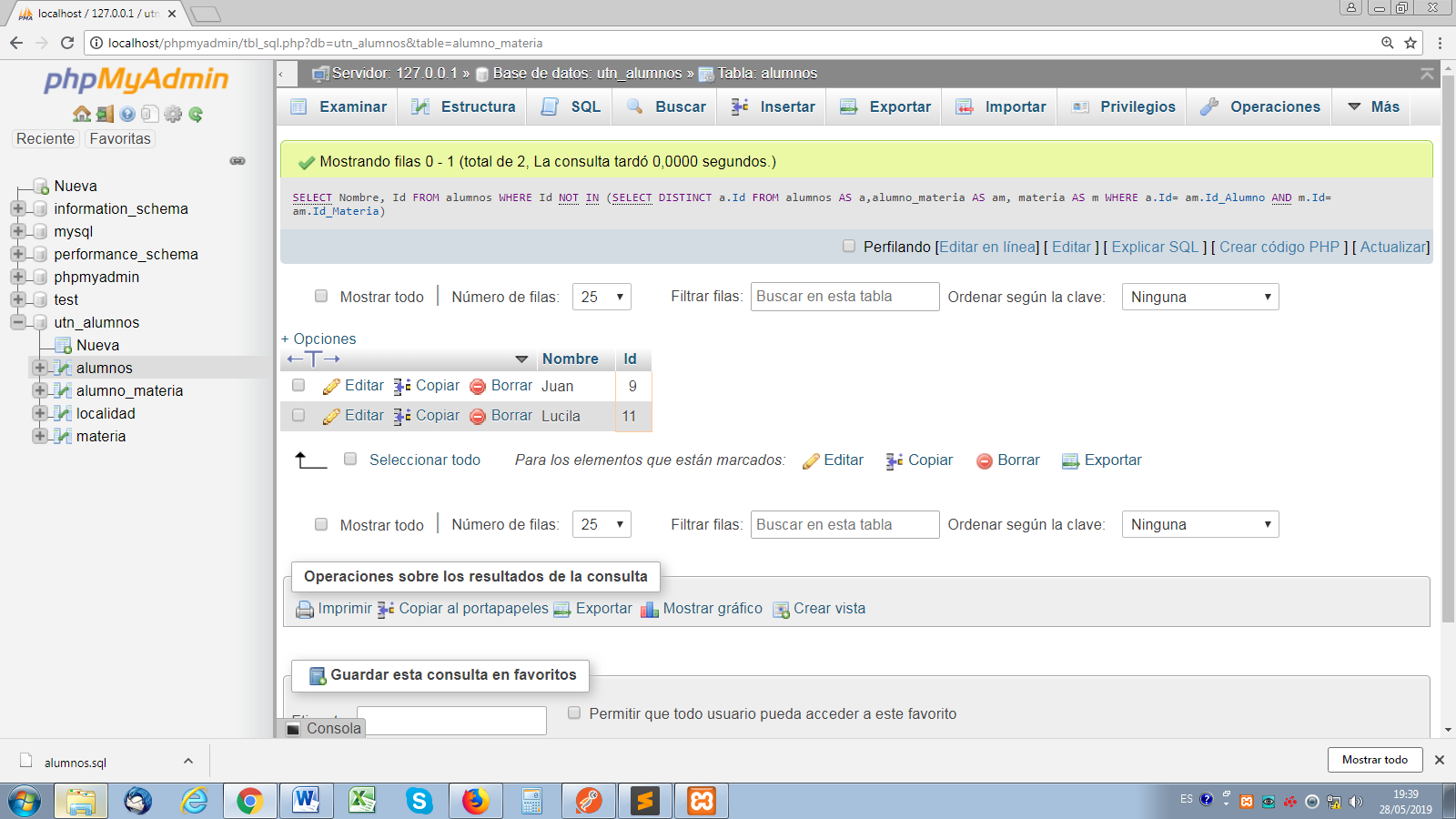


SQL es declarativo, puede devolver cosas distintas en distintas consultas

Id’s que no están



Los que no rindieron… (que no estan en la tabla anterior)

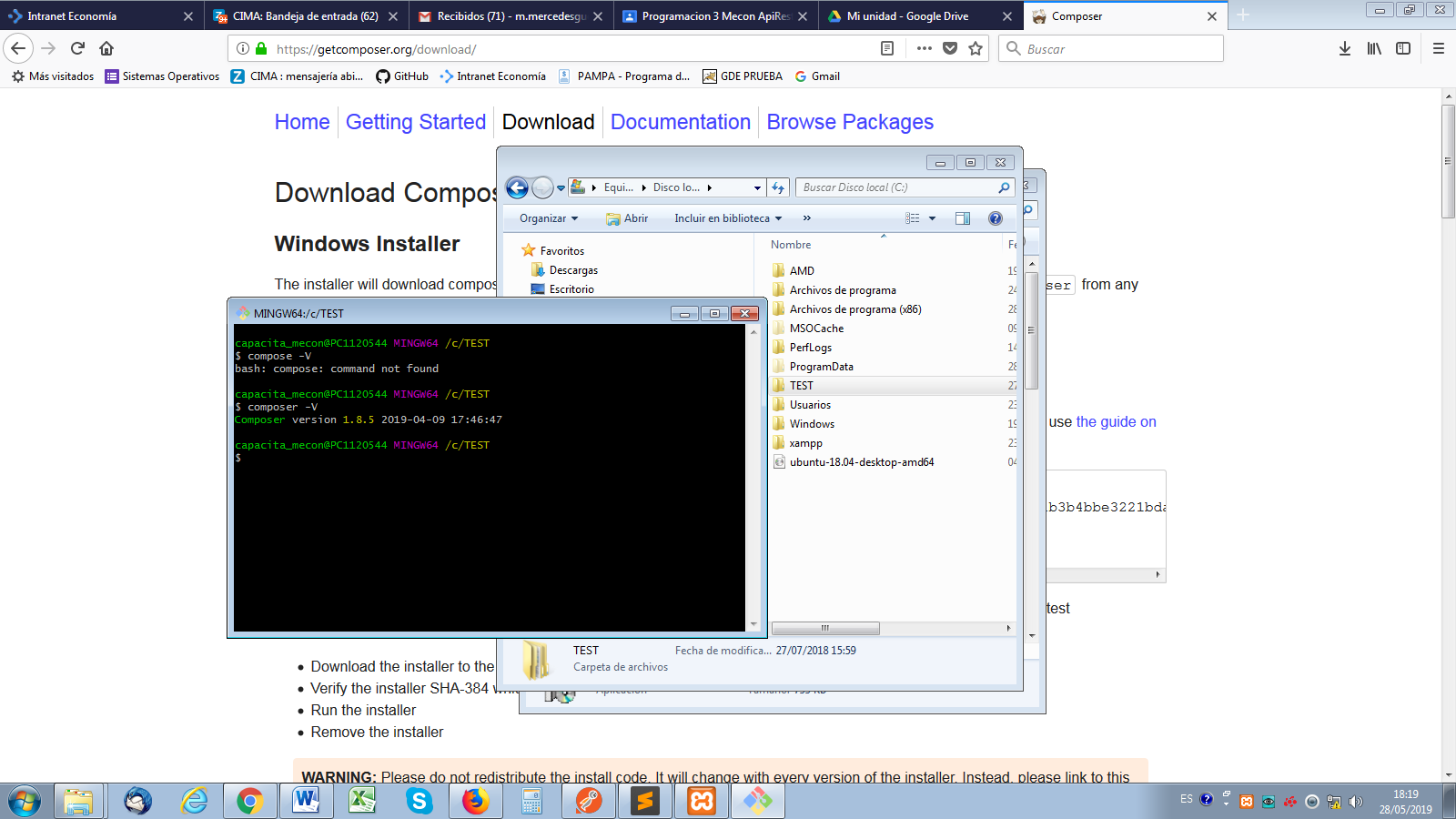


# Composer

PHP 7.3.4 Instalar

Composer 1.6

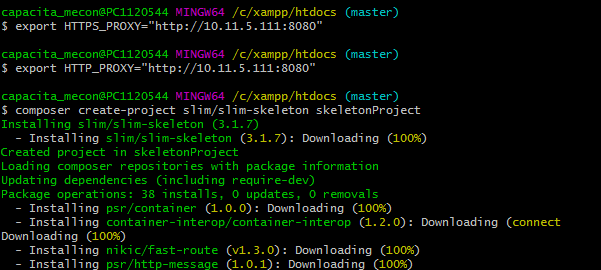
<https://getcomposer.org/download/>

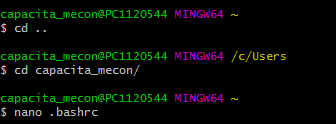




Herramienta que nos permite hacer distintas cosas con PHP. Agrupar bibliotecas.

Crea una carpeta que se llama vendor donde va descargando cosas.





En ese archivo agregar los export 🡪 CTRL+X para guardar

## Composer create-project

Instrucción que va a crear un proyecta respetando la estructura y los patrones de diseño de ese proyecto.

Composer require

Devuelve referencias a instancias

Lo subimos a un servidor y ya tenemos una apirest

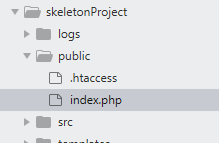
Esto es un patrón de diseño

Archivo .htacces:

Es lo primero que lee cuando entra al servidor

Para poner contraseñas o para redireccionar a index.php

Abrimos el archivo index.php



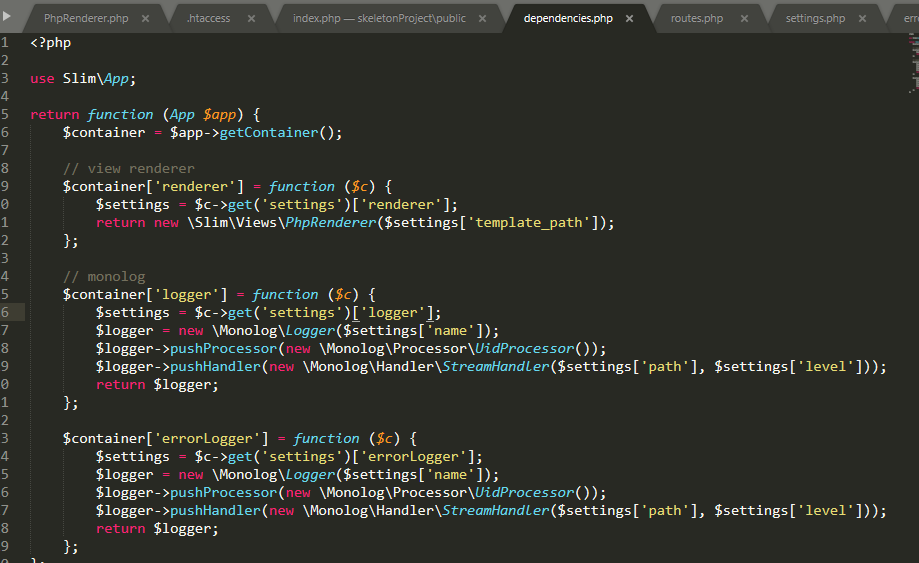
 Lo que hace es cargar las bibliotecas

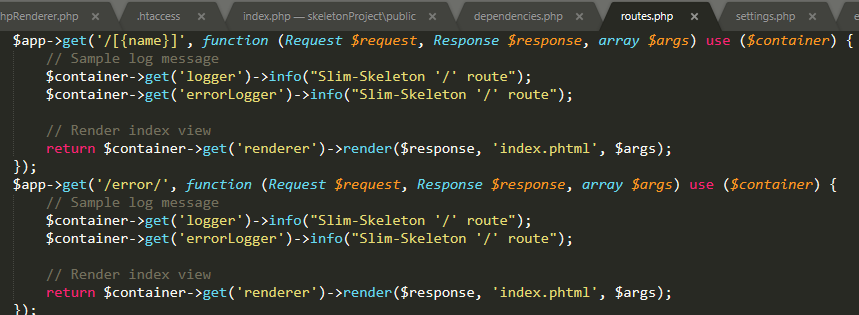
 Asigna a la variable

* Para hacer andar algo hay que configurarlo en settings.php
* Después en dependencias.php creo los objetos dentro del Conteiner del app
* GET container -> Base de datos objeto
* El uso se puede hacer dentro de las rutas o en objetos.

Cosas que cambiamos del skeleton:





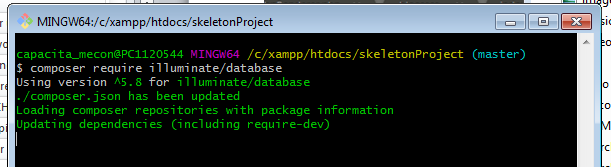


En el ejemplo de Octavio Full-API-REST-Skeleton-master está el POST PUT DELETE terminado

## ORM

Vamos a usar un ORM

Es un objeto que viene de la base de datos como un array de string (SELECT) hace una acción, traer una tabla y por cada elemento de la tabla crea un objeto con sus respectivos atributos

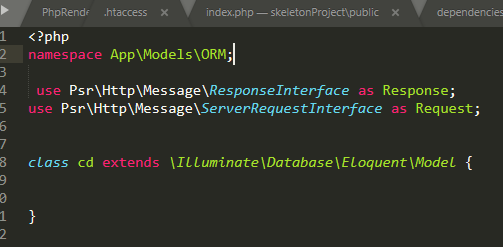


Instrucciones:

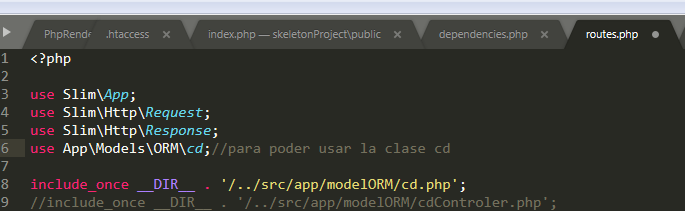
<https://github.com/octaviovillegas/APIREST-PHP-POO-JWT-MIDDLEWARE-Documentar/blob/master/apirestV7-ORM/README.md>

Use 🡪 utiliza un namespace, es una agrupación lógica

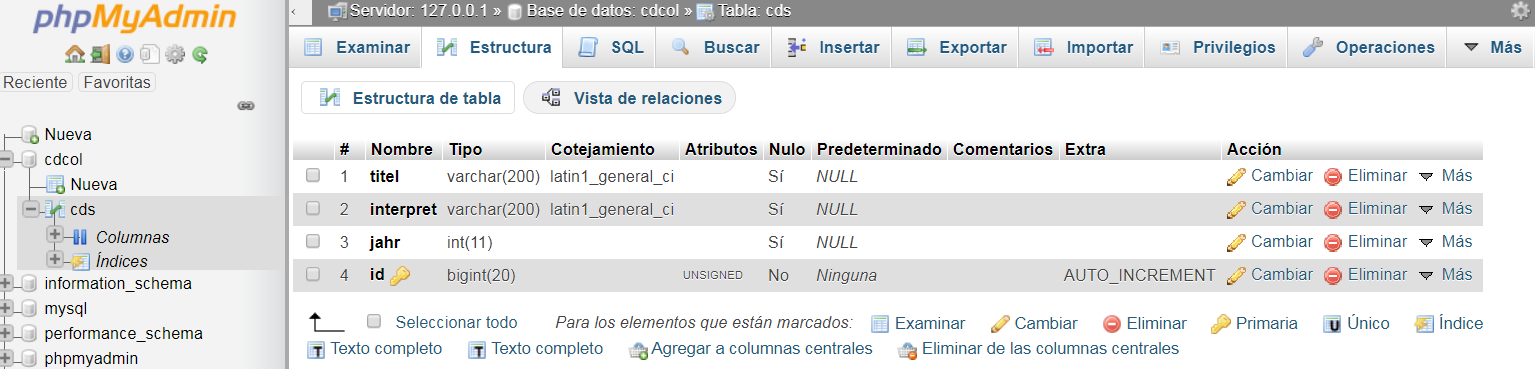
Las clases las defino dentro de un namespace



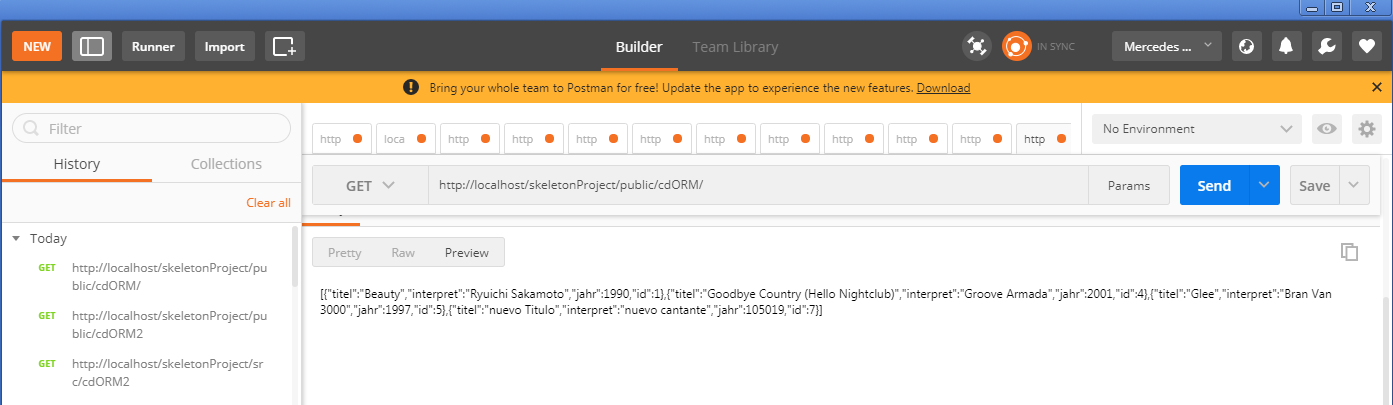
En routes uso ese namespace para que se entere que esa clase existe



Base de datos



Traer los cd’s



INSERT, INSERT INTO, WHERE ya está todo programado  
<https://github.com/octaviovillegas/APIREST-PHP-POO-JWT-MIDDLEWARE-Documentar/blob/master/apirestV7-ORM/index.php>

|  |
| --- |
| echo "<br><br><br>"; |
|  | echo "Agregar un cd<br>"; |
|  | $cd = new \App\Models\cd(); |
|  | $cd->titel="Album negri"; |
|  | $cd->interpret="Matalico"; |
|  | $cd->jahr=2020; |
|  |  |
|  | $cd->save(); |
|  |  |
|  | echo $cd->toJson(); |
|  | echo "<br><br><br>"; |
|  | echo "Traer un cd<br>"; |
|  |  |
|  | $otroCd =$cd->find(5); |
|  | echo $otroCd->toJson(); |
|  |  |
|  | echo "<br><br><br>"; |
|  | echo 'Modificar un cd<br>'; |
|  | $otroCd->titel="cambiado"; |
|  | $otroCd->interpret="Matalico"; |
|  | $otroCd->jahr=2666; |
|  |  |
|  | $otroCd->save(); |
|  | echo $otroCd->toJson(); |
|  | echo "<br><br><br>"; |
|  | echo 'buscar los cds<br>'; |
|  | $respuesta= $cd->where('titel', "=","Album negri")->get(); |
|  | echo $respuesta->toJson(); |
|  |  |
|  | echo "<br><br><br>"; |
|  | echo 'buscar el primero un cd<br>'; |
|  | $respuesta= $cd->where('titel', "=","Album negri")->first(); |
|  | echo $respuesta->toJson(); |
|  |  |
|  | echo "<br><br><br>"; |
|  | echo 'buscar el primero un cd $cd->where(titel,Album negri)->first() <br>'; |
|  | $respuesta= $cd->where('titel',"Album negri")->first(); |
|  | echo $respuesta->toJson(); |
|  |  |
|  | echo "<br><br><br>"; |
|  | echo 'buscar el wheretitel <br>'; |
|  | $respuesta= $cd->wheretitel("Album negri")->first(); |
|  | echo $respuesta->toJson(); |
|  |  |
|  |  |
|  |  |
|  | echo "<br><br><br>"; |
|  | echo 'borrar un cd<br>'; |
|  |  |
|  | $respuesta->delete(); |
|  | echo $respuesta->toJson(); |
|  |  |

wheretitel es un atributo de la clase

Solo hay que hacer la tabla y la clase🡪 la tabla tiene que tener una s y la clase no. Ejemplo: cds y cd.php