

---

# MDIS

Programación Web

Amin Kasrou Aouam



**UNIVERSIDAD  
DE GRANADA**

2020-07-16

## Índice

<b>MDIS</b>	<b>3</b>
Funcionalidades . . . . .	3
Tecnologías . . . . .	3
Arquitectura . . . . .	3
Dependencias . . . . .	4
Despliegue . . . . .	4

## MDIS

MDIS es un sistema de información que permite la gestión de una consulta médica.

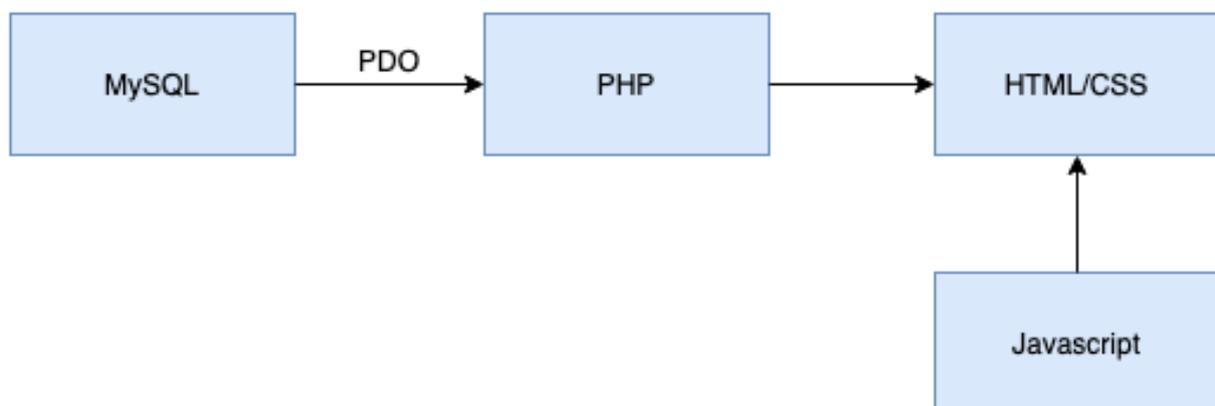
### Funcionalidades

- Gestión de Usuarios
- Gestión de Pacientes
- Gestión de Calendario
- Gestión de Citas
- Gestión de Vacaciones
- Gestión de Informes

### Tecnologías

- PHP
- MySQL
- Javascript
- Fullcalendar
- JQuery
- JQueryUI
- Nix

### Arquitectura



**Figura 1:** Arquitectura del sistema

Es un sistema web clásico, con la característica de que las consultas a la base de datos se realizan mediante *PDO*, para evitar vulnerabilidades del tipo *inyección de SQL*.

Las operaciones que conllevan una inserción o transformación de datos se realizan en los archivos que terminan en **\_management.php**, además de éste, no hacemos ninguna consulta *SQL* fuera del archivo **database.php**, lo que nos permite separar la lógica interna de la presentación.

## Dependencias

El apartado de la gestión de citas se ha realizado utilizando la biblioteca FullCalendar de Javascript. La conexión entre el *backend* y el *frontend* se realiza mediante intercambio de JSON, la implementación se encuentra en los archivos que terminan en **\_feed.php**.

A partir de los elementos de la base de datos, formateados y transformados, obtenemos las citas, los festivos y la configuración del calendario de cada doctor. Finalmente, personalizamos el comportamiento del calendario según estos datos.

Los elementos del calendario también requieren de JQueryUI, para darle un toque más moderno a los distintos componentes.

Por último, hacemos uso de la función `$.ajax()` de JQuery para realizar peticiones *GET* síncronas, dado que ciertos componentes de FullCalendar no pueden ser ejecutados como funciones asíncronas.

## Despliegue

El desarrollo y despliegue del sistema se han hecho gracias a *Nix*, un gestor de paquetes que permite entornos de desarrollo y despliegue reproducibles.

A continuación mostramos el código que define el entorno de desarrollo:

```
1 { pkgs ? import <nixpkgs> { } }:  
2  
3 with pkgs;  
4  
5 mkShell {  
6   # Definición de los paquetes  
7   buildInputs = [ php74 php74Extensions.pdo_mysql mysql57 ];  
8  
9   # Comandos que se ejecutan al entrar en la nix-shell  
10  shellHook = ''  
11    pkill mysql  
12    rm -rf .mysql && mkdir .mysql  
13  
14    mysqld --datadir="$(pwd)/.mysql" --socket="$(pwd)/.mysql/mysql.sock  
15    " --initialize-insecure  
16    mysqld --datadir="$(pwd)/.mysql" --socket="$(pwd)/.mysql/mysql.sock  
17    " --skip-networking &  
18    sleep 1  
19  
20    mysql --socket="$(pwd)/.mysql/mysql.sock" -u root < $(pwd)/database  
21    /db.sql  
22  
23    alias mysql='mysql --socket="$(pwd)/.mysql/mysql.sock" -u root'  
24  '';  
25 }
```

Como podemos ver, *Nix* nos permite:

- Instalar las dependencias necesarias, en el ámbito de una shell
- Ejecutar una base de datos temporal
- Inicializar la base de datos
- Lanzar el servidor web interno de PHP (se podría reemplazar por Apache/Nginx/...)

El único comando que tenemos que ejecutar es:

```
1 nix-shell
```

Y ya dispondremos de un sistema funcional, y accesible en la URL **localhost:8000**.