MDIS

Programación Web

Amin Kasrou Aouam

UNIVERSIDAD
DE GRANADA

2020-07-16

Índice

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

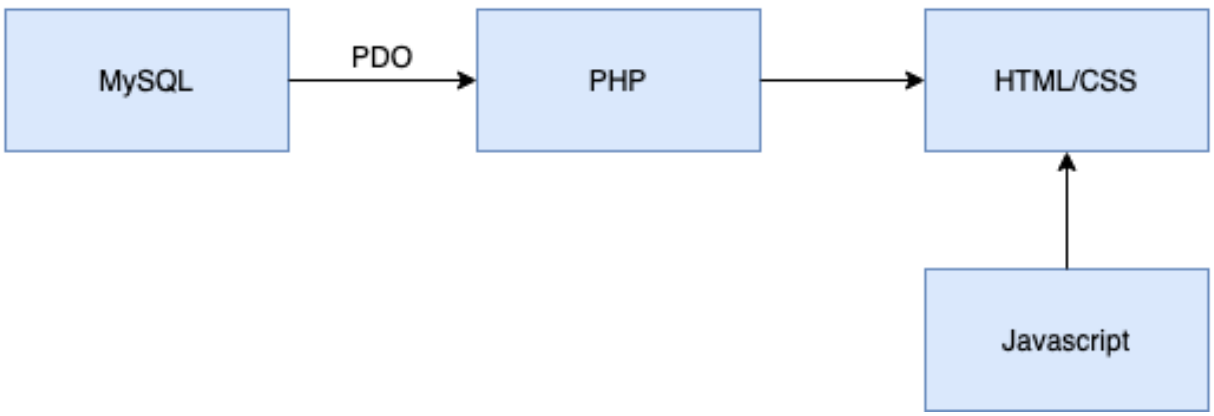

Figura 1: Arquitectura del sistema

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 ésto, no hacemos ninguna consulta *SQL* fuera del archivo **database.php**, lo que nos permite separar la lógica interna de la presentación.

Estructura del proyecto

La segmentación de los diferentes archivos de un proyecto es un aspecto que facilita mucho la búsqueda en un proyecto. La estructura de directorios es la siguiente:

- database: archivos SQL para la creación de la base de datos.
- docs: memoria del proyecto.
- src: código fuente (PHP y Javascript).
- src/static: bibliotecas de javascript, CSS y fotos.

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.

La impresión de los informes en formato PDF es posible gracias a jsPDF, biblioteca simple y que produce documentos con un diseño cuidado.

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:

```
{ pkgs ? import <nixpkgs> { } }:

with pkgs;

mkShell {
  # Definición de los paquetes
  buildInputs = [ php74 php74Extensions.pdo_mysql mysql57 ];

  # Comandos que se ejecutan al entrar en la nix-shell
  shellHook = ''
    pkill mysql
    rm -rf .mysql && mkdir .mysql

    mysqld --datadir="$(pwd)/.mysql" --socket="$(pwd)/.mysql/mysql.sock" --initialize-insecure
    mysqld --datadir="$(pwd)/.mysql" --socket="$(pwd)/.mysql/mysql.sock" --skip-networking &
    sleep 1

    mysql --socket="$(pwd)/.mysql/mysql.sock" -u root < $(pwd)/database/db.sql

    alias mysql='mysql --socket="$(pwd)/.mysql/mysql.sock" -u root'

    php -S localhost:8000
  '';
}
```

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:

```
nix-shell
```

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