línea horizontal

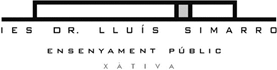


Quico Blesa Penadés

2º DAW 24/25

IES Doctor Lluis Simarro la Cabra

Jose Castillo





*ÍNDICE*

[**INTRODUCCION 3**](#_kotfr1wox850)

[**OBJETIVOS DE LA APP 4**](#_kmc04x9i799v)

[**TECNOLOGÍAS UTILIZADAS 5**](#_l09jy7t0x8l7)

[Frontend (Entorno Cliente) 5](#_sc6an5c70ww)

[Backend (Entorno Servidor) 5](#_nb5qbw5yy7vm)

[React (Framework de Javascript) 6](#_1qzbov5dv7fd)

[Flujo de datos entre frontend y backend 6](#_vm9dk97r0smd)

[->solicitud desde react 6](#_3316gwswnfgo)

[->Recepción desde php 7](#_4j1ncvgqsxx2)

[**Desarrollo del proyecto 8**](#_uhni6wj6ft9k)

[Diagrama de entidades 8](#_psul72dno1nd)

[Diagrama de clases 9](#_2giup06upe5s)

[Casos de uso 10](#_bpkvnc9u3xjs)

[-> Crear una cita entre médico y cliente 10](#_9ntg9ltkm2tq)

[-> Completar una cita y crear una receta médica 13](#_k1245hkctsoe)

[-> Registrar un usuario y iniciar sesion de usuario 16](#_ftl77l6q5zha)

[-> Otros casos de uso… 18](#_90ynv64hms9y)

# INTRODUCCION

Durante una pequeña investigación sobre el funcionamiento interno de las clínicas médicas y su presencia en internet, observamos que muchas aún operan de forma obsoleta. Por ejemplo, los productos se registran en hojas de Excel o incluso en papel, y los médicos anotan los datos de sus pacientes sin información detallada ni un sistema organizado. Las citas, además, se gestionan de forma ineficiente.

A raíz de esta problemática, una clínica nos contactó con la necesidad de desarrollar una aplicación web que les permitiera mostrar sus servicios y productos en internet, al mismo tiempo que facilitara la gestión interna. Esta solución debía permitir al administrador gestionar productos, categorías y personal médico de forma ágil. También debía contar con una interfaz para que los médicos pudieran registrar información de los pacientes, visualizar estadísticas de productos más recomendados y asignar citas de manera centralizada, lo que permitiría mantener todo organizado y disponible para futuros análisis.

En este contexto nace **MediConnect**, una solución digital diseñada para mejorar la gestión interna de una clínica médica y aumentar su visibilidad en internet.

Los beneficios de esta aplicación son múltiples: permitirá al administrador gestionar todos los recursos de la clínica de forma sencilla y dinámica, con acceso a estadísticas y gráficos sobre productos, categorías y rendimiento general, mejorando así el control y la toma de decisiones.

# OBJETIVOS DE LA APP

**MediConnect** tiene como objetivo desarrollar una aplicación web integral para la gestión de clínicas médicas, con los siguientes fines:

1. **Mejorar la gestión interna** de productos, categorías y personal médico mediante un panel de administración intuitivo.
2. **Registrar y gestionar pacientes** con información detallada, accesible para el equipo médico.
3. **Asignar y organizar citas médicas** de manera centralizada, eficiente y visual.
4. **Ofrecer estadísticas y gráficos interactivos** sobre productos más recomendados y rendimiento de la clínica.
5. **Aumentar la presencia online** de la clínica mediante una interfaz visible para los clientes con los servicios y productos ofrecidos.
6. **Facilitar la toma de decisiones** gracias a una plataforma basada en datos y registros actualizados.
7. **Ofrecer una experiencia unificada** tanto para el administrador como para los médicos.

# 

# 

# TECNOLOGÍAS UTILIZADAS

Para el desarrollo de la aplicación **MediConnect**, se ha optado por una arquitectura basada en el **Modelo-Vista-Controlador (MVC)**, separando el frontend del backend. Esta elección permite una mayor escalabilidad, mantenibilidad y organización del proyecto.

## Frontend (Entorno Cliente)

El **frontend** es la parte de la aplicación con la que interactúan los usuarios. Para su desarrollo se ha utilizado **React**, una biblioteca moderna y eficiente de JavaScript para la construcción de interfaces de usuario. React permite ofrecer una experiencia dinámica e intuitiva al usuario.

* **Librería principal:** React (framework de JavaScript).
* **Herramienta de compilación:** Vite.
* **Estilado de la app:** Bootstrap 5 y CSS utilizando Flexbox y Grid.

## Backend (Entorno Servidor)

El **backend** se encarga de procesar las solicitudes de los clientes, proporcionando los datos requeridos y gestionando la lógica del sistema.

* **Lenguaje:** PHP nativo, con el cual se ha desarrollado una API que suministra los recursos a la aplicación y gestiona las peticiones de los usuarios.
* **Gestor de base de datos:** MySQL, utilizado para almacenar y administrar los datos de forma eficiente y segura.
* **Herramienta de administración de bases de datos:** HeidiSQL, empleada para conectarse, crear, modificar y consultar bases de datos de forma visual, sin necesidad de escribir todas las consultas SQL manualmente.

## React (Framework de Javascript)

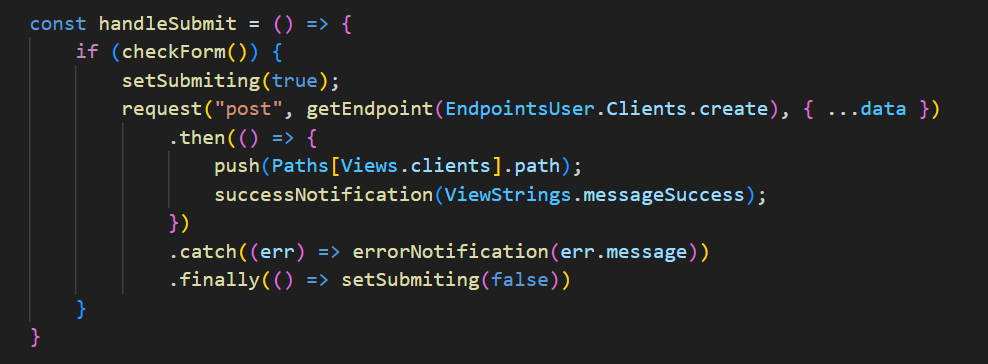
React es un framework de JavaScript que está diseñado para construir la interfaz de usuario a partir de pequeñas piezas reutilizables llamadas componentes que gestionan su propio estado interno. También react utiliza un enfoque donde los datos sólo fluyen en la misma dirección que es de componentes padre a componentes hijos, esto se hace pasando los datos como propiedades del componente también conocidas como "props".

* **Componentes**: Piezas pequeñas de código que forman la interfaz de usuario.
* **Props**: Propiedades que se pasan de un componente padre a un componente hijo.

## Flujo de datos entre frontend y backend

La comunicación entre el frontend y el backend se realiza mediante solicitudes **HTTP** a través de la api construida en **PHP**. Principalmente las solicitudes son de tipo **POST/GET** aunque no siempre son iguales ya que cuando se pasan imágenes o diferentes tipos de archivos cambia la forma de transmitir los datos.

### **->solicitud desde react**

Desde el front construido en react enviamos de esta manera la solicitud al backend, la función request es un **custom hook** que hace uso de **axios** para enviar la solicitud **Post** al servidor.

### 

### **->Recepción desde php**

Desde el back en php se recibe la información con la función **postInput()**, a continuación se crea la transición de la conexión con la base de datos y se comprueba si necesitas autenticación con las funciones **checkUser()** o **checkAdmin()**.

Se valida la información recibida por el **postInput()** con la función validate, si la validación es correcta se realizan las acciones necesarias y se devuelven los datos al cliente usando **Response::sendResponse()**.



# DESARROLLO DEL PROYECTO

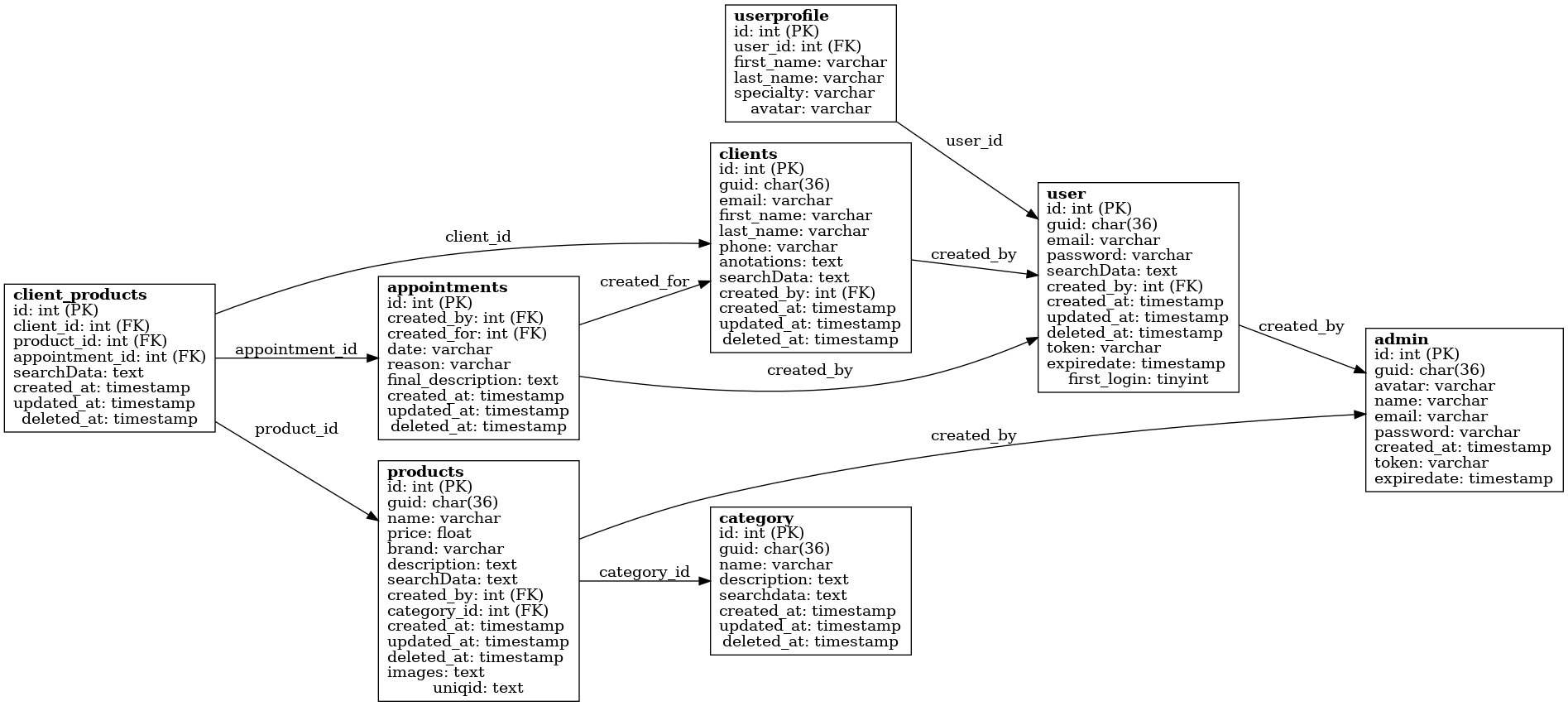
## Diagrama de entidades

En esta imagen se muestra las **relaciones** con sus **cardinalidades** respectivas entre entidades.

Un administrador crea productos, una categoría contiene productos, un administrador crea usuarios, un usuario tiene un perfil, un usuario crea citas, un usuario crea clientes, un cliente puede tener más de una cita, muchos productos pueden estar recomendados a muchos clientes.

## 

## Diagrama de clases



## 

## 

## 

## 

## 

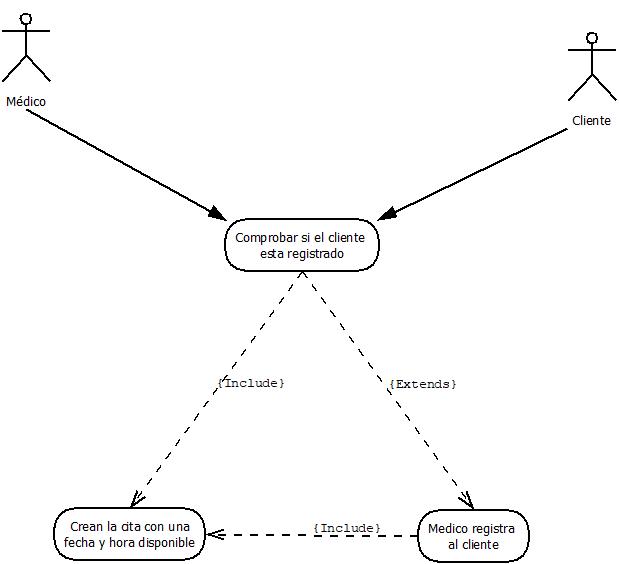
## 

## 

## 

## Casos de uso

### **-> Crear una cita entre médico y cliente**



**Nombre de la acción:** Comprobar si el usuario está registrado

| **Actores:** Médico y cliente. |
| --- |
| **Condiciones anteriores a la acción:** El médico debe estar anteriormente registrado por el administrador de la clínica. |
| **Flujo de la acción:**   1. El cliente entrega sus datos a su médico. 2. El médico comprueba que el cliente está registrado. (Haciendo uso de los filtros en el apartado “Clientes”). |
| **Condiciones posteriores a la acción:** Una vez comprobado seguir con el caso de uso. |
| **Alternativas:** Sin alternativas. |

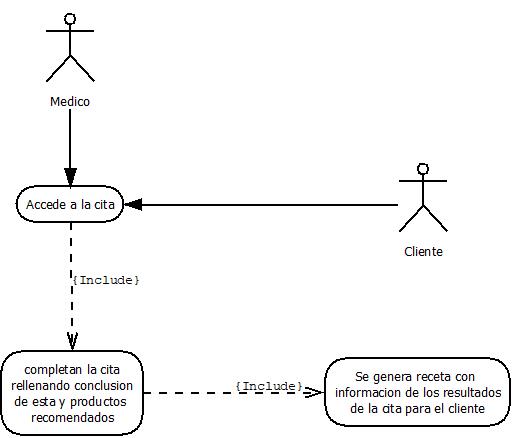
**Nombre de la acción:** Registrar al cliente

| **Actores:** Médico y cliente. |
| --- |
| **Condiciones anteriores a la acción:** El médico debe haber iniciado sesión anteriormente. |
| **Flujo de la acción:**   1. El médico pregunta los datos necesarios para el registro al cliente y el cliente le entrega sus datos. 2. El médico le realiza un examen genérico para guardar en la base de datos el estado del cliente. |
| **Condiciones posteriores a la acción:** Una vez registrado ya se pueden crear citas. |
| **Alternativas:** Sin alternativas. |

**Nombre de la acción:** Crear cita para cliente

| **Actores:** Médico y cliente |
| --- |
| **Condiciones anteriores a la acción:** El cliente debe estar registrado anteriormente por su médico. |
| **Flujo de la acción:**   1. El médico entra al apartado de calendario para poder visualizar días y horas que tiene libres y así comunicárselo al cliente. 2. El cliente elige un día y hora para la cita. 3. El médico crea la cita para que se guarde en la base de datos. |
| **Condiciones posteriores a la acción:** Una vez creada la cita termina el caso de uso. |
| **Alternativas:**   1. Entre el tiempo que se creó la cita y el día de la cita podría ser cancelada esta y se terminaría el caso de uso. |

### **-> Completar una cita y crear una receta médica**



**Nombre de la acción:** Acceder a la cita

| **Actores:** Médico y cliente |
| --- |
| **Condiciones anteriores a la acción:** El médico debe estar anteriormente registrado por el administrador de la clínica y el cliente debe asistir y estar registrado. |
| **Flujo de la acción:**   1. El cliente llega a la clínica por el motivo de su visita. 2. El médico busca la cita en su apartado de trabajo diario y accede a ella. |
| **Condiciones posteriores a la acción:** Una vez dentro de la cita seguir con el caso de uso. |
| **Alternativas:** El cliente no se presenta a la cita. |

**Nombre de la acción:** Completar cita

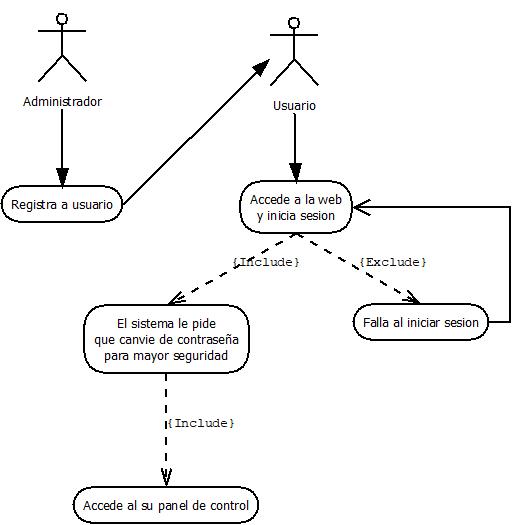
| **Actores:** Médico y cliente |
| --- |
| **Condiciones anteriores a la acción:** El médico debe haber accedido a la cita y el cliente estar presente. |
| **Flujo de la acción:**   1. El médico examina la razón de la cita del cliente. 2. El médico examina al cliente y realiza los exámenes médicos necesarios. 3. El médico rellena la receta con la conclusión y los productos recomendados para el tratamiento del cliente. |
| **Condiciones posteriores a la acción:** Una vez completada la cita sigue el caso de uso. |
| **Alternativas:** Sin alternativas. |

**Nombre de la acción:** Crear receta médica

| **Actores:** Médico y cliente |
| --- |
| **Condiciones anteriores a la acción:** El médico debe haber completado la cita propuesta por el cliente. |
| **Flujo de la acción:**   1. El médico descarga el pdf con la receta. 2. El cliente recibe la receta para poder dirigirse al mostrador y comprar los productos. |
| **Condiciones posteriores a la acción:** Una vez entregada la receta se termina el caso de uso. |
| **Alternativas:** El cliente puede no comprar ningún producto. |

### **-> Registrar un usuario y iniciar sesion de usuario**

### 



**Nombre de la acción:** Registrar usuario

| **Actores:** Administrador y usuario |
| --- |
| **Condiciones anteriores a la acción:** Sólo un administrador puede hacer la acción de registrar a un usuario para que este pueda acceder a su panel de control. |
| **Flujo de la acción:**   1. El administrador contrata a un médico. 2. El administrador se dirige al apartado “usuario” para rellenar el cuestionario con la información del usuario. |
| **Condiciones posteriores a la acción:** Una vez creado el usuario las próximas acciones son por parte del usuario. |
| **Alternativas:** Sin alternativas. |

**Nombre de la acción:** Acceder al sitio web y iniciar sesión

| **Actores:** usuario y administrador |
| --- |
| **Condiciones anteriores a la acción:** El administrador de la clínica debe haber creado al usuario desde su panel de control. |
| **Flujo de la acción:**   1. El usuario accede a la app y se sitúa en el apartado de inicio de sesión. 2. Introduce su email y contraseña genérica que le proporciona el administrador. 3. El usuario completa el formulario de cambio de contraseña. 4. El usuario accede a su panel de control. |
| **Condiciones posteriores a la acción:** Una vez el usuario acceda a su panel de control y se complete el formulario de cambio de contraseña termina el caso de uso. |
| **Alternativas:** Sin alternativas. |

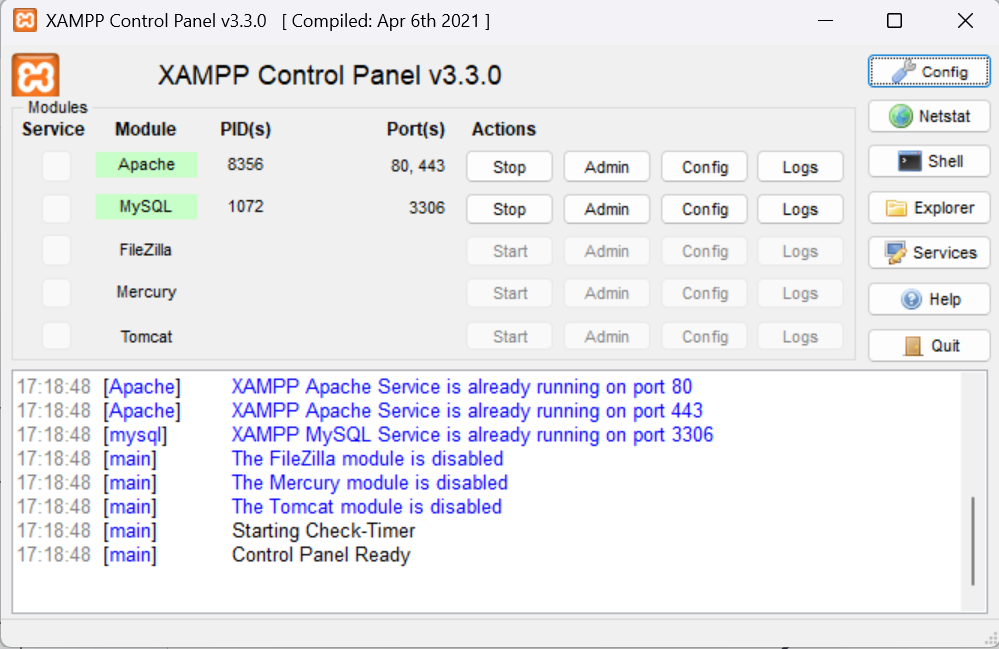
### **-> Otros casos de uso…**

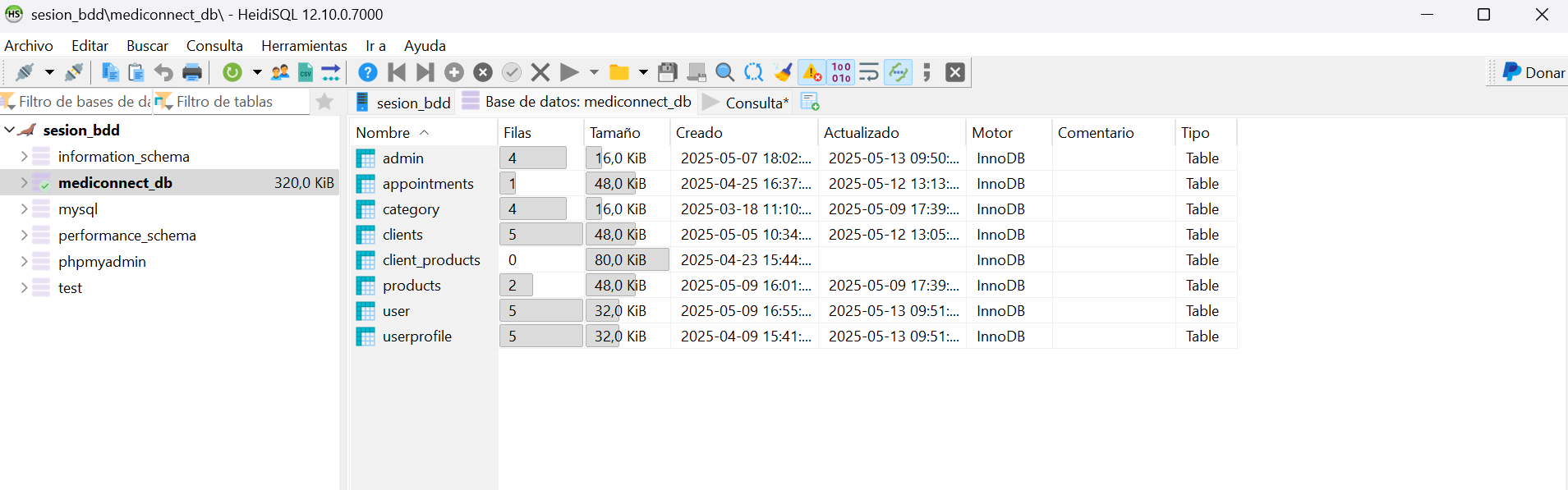
1. Iniciar Sesión
   1. Actores: Administrador y usuario
   2. Descripción: Tanto el administrador como el usuario ingresan sus credenciales en sus respectivos inicios de sesión para acceder a la aplicación.
2. Editar Perfil
   1. Actores: Administrador y usuario
   2. Descripción: Tanto el administrador como el usuario pueden modificar sus datos personales.
3. Editar cuenta
   1. Actores: Administrador y usuario
   2. Descripción: Tanto el administrador como el usuario pueden modificar sus datos de cuenta como email y contraseña.
4. Elegir Idioma
   1. Actores: Todo tipo de usuario
   2. Descripción: Todo tipo de usuario puede elegir el idioma de la interfaz de la aplicación..
5. Cerrar sesión
   1. Actores: Administrador y usuario
   2. Descripción: Tanto el administrador como el usuario cierra su sesión en la aplicación.
6. **Casos de uso del Administrador**
   1. Gestionar Categorías
      1. Crear categoría
         1. Actor: Administrador
         2. Descripción: El administrador puede crear nuevas categorías en la base de datos.
      2. Editar categoría
         1. Actor: Administrador
         2. Descripción: El administrador puede crear nuevas categorías en la base de datos.
      3. Borrar categoría
         1. Actor: Administrador
         2. Descripción: El administrador puede crear nuevas categorías en la base de datos.
      4. Visualizar todas las categorías
         1. Actor: Administrador
         2. Descripción: El administrador puede crear nuevas categorías en la base de datos.
   2. Gestionar Productos
      1. Crear producto
         1. Actor: Administrador
         2. Descripción: El administrador puede crear nuevos productos en la base de datos.
      2. Editar producto
         1. Actor: Administrador
         2. Descripción: El administrador puede editar productos en la base de datos.
      3. Borrar producto
         1. Actor: Administrador
         2. Descripción: El administrador puede borrar productos de la base de datos.
      4. Visualizar todos los productos
         1. Actor: Administrador
         2. Descripción: El administrador visualiza todos los productos de la base de datos.
7. Gestionar Usuarios
   1. Crear usuario
      1. Actor: Administrador
      2. Descripción: El administrador puede crear nuevos usuarios en la base de datos.
   2. Editar usuario
      1. Actor: Administrador
      2. Descripción: El administrador puede editar información del usuario en la base de datos.
   3. Borrar usuario
      1. Actor: Administrador
      2. Descripción: El administrador puede borrar usuarios de la base de datos.
   4. Visualizar todos los usuarios
      1. Actor: Administrador
      2. Descripción: El administrador visualizará todos los usuarios registrados en la clínica.
8. **Casos de uso del Usuario**
   1. Gestionar Clientes
      1. Crear Cliente
         1. Actor: Usuario
         2. Descripción: El administrador puede crear nuevas categorías en la base de datos.
      2. Editar Cliente
         1. Actor: Usuario
         2. Descripción: El administrador puede crear nuevas categorías en la base de datos.
      3. Borrar Cliente
         1. Actor: Usuario
         2. Descripción: El administrador puede crear nuevas categorías en la base de datos.
      4. Visualizar todos sus Clientes
         1. Actor: Usuario
         2. Descripción: El administrador puede crear nuevas categorías en la base de datos.
   2. Gestionar Citas
      1. Crear producto
         1. Actor: Administrador
         2. Descripción: El administrador puede crear nuevos productos en la base de datos.
      2. Editar producto
         1. Actor: Administrador
         2. Descripción: El administrador puede editar productos en la base de datos.
      3. Borrar producto
         1. Actor: Administrador
         2. Descripción: El administrador puede borrar productos de la base de datos.
      4. Visualizar todos los productos
         1. Actor: Administrador
         2. Descripción: El administrador visualiza todos los productos de la base de datos.

# IMPLEMENTACIÓN DEL PROYECTO

En este apartado explicaremos como trabajamos desde un principio corriendo la aplicación en local hasta desplegar esta en un servidor y dominio propio.

## Primera fase del proyecto

Durante el inicio del proyecto se trabajó completamente en local por lo tanto con la herramienta XAMPP podíamos correr tanto apache como MYSQL.

También con la herramienta de diseño HeidiSQL se fueron diseñando las diferentes versiones de la base de datos ya que con el progreso del proyecto podían hacer falta más columnas o tipos de datos en esta, desde heidiSQL se exportó finalmente el sql de creación de la base de datos en MYSQL.

## Servidor en azure

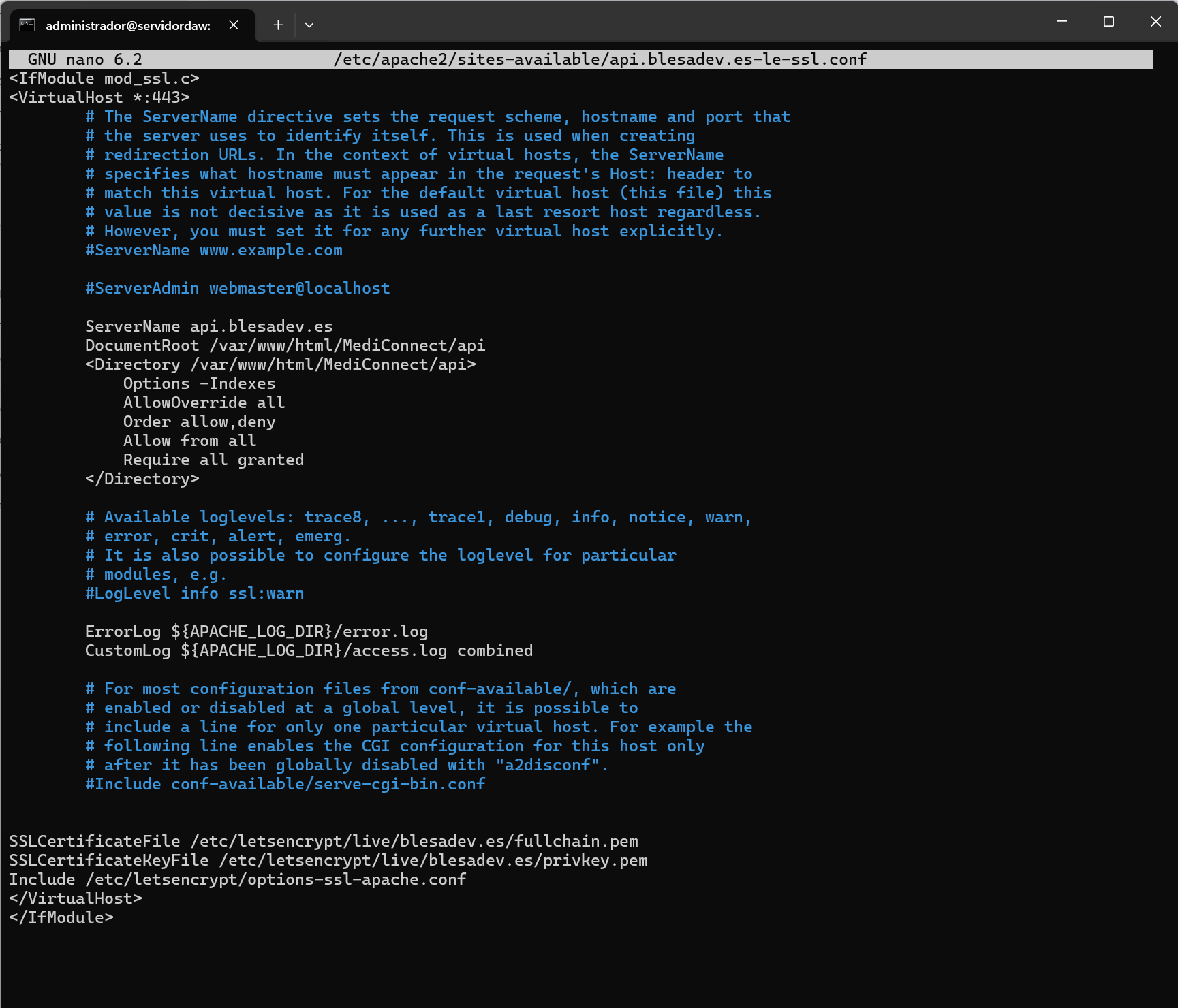
Una vez ya el proyecto terminado y sin errores en local se creó una máquina virtual en azure con el sistema operativo de ubuntu Server 22.24, a esta máquina se le instaló mySQL para poder más tarde crear la base de datos dentro del servidor.

También se le instaló apache que es un servidor web de código abierto utilizado para alojar aplicaciones en internet.

Se configuraron dos virtualHost para que no hubiera conflictos entre la api y la app.

1. VirtualHost para la api

Como se puede observar apunta a la carpeta /var/www/html/MediConnect/api.



1. VirtualHost para la app

Como se puede observar apunta a la carpeta /var/www/html/MediConnect/app.

