

# Trabajo de fin de grado Clínica Privada

Desarrollo de Aplicaciones Web

Autor: Alejandro Vlad Orive

Tutor: Gustavo Millán García

Fecha: 14-6-2025

Centro : Instituto les el Cañaveral , Móstoles.

# Índice

<b>Introducción.....</b>	<b>4</b>
Descripción y contexto del proyecto.....	4
<b>1.2.- Motivación del proyecto.....</b>	<b>6</b>
<b>1.3.- Beneficios esperados.....</b>	<b>7</b>
<b>2.- Objetivo/s generales del proyecto.....</b>	<b>9</b>
<b>3.- Objetivos específicos.....</b>	<b>9</b>
<b>4.- Contexto actual.....</b>	<b>10</b>
<b>5.- Análisis de requisitos.....</b>	<b>11</b>
<b>5.1.- Diagrama de casos de uso.....</b>	<b>11</b>
<b>5.2.- Requisitos funcionales principales: funcionalidades que debe tener la aplicación.....</b>	<b>12</b>
Administrador.....	12
Médico.....	12
Paciente.....	12
Recepcionista.....	13
<b>5.3.- Requisitos no funcionales: rendimiento, seguridad, usabilidad, etc.....</b>	<b>13</b>
1. Rendimiento.....	13
2. Seguridad.....	13
3. Disponibilidad.....	14
4. Usabilidad.....	14
5. Trazabilidad.....	14
<b>5.4.- Descripción de los usuarios y sus necesidades.....</b>	<b>15</b>
<b>6.- Diseño de la aplicación.....</b>	<b>16</b>
<b>6.1.-Interfaz gráfica de usuario.....</b>	<b>17</b>
SIN CUENTA.....	17
Página principal (sin inicio de sesión).....	17
Nutrición.....	18
Deportes.....	18
Ayuda.....	19
Crear cuenta.....	19
Inicio de sesión.....	20
ADMIN.....	21
Crear usuario.....	21
Crear rol y visualización de roles.....	23
Editar roles usuarios.....	23
Datos estadísticos.....	24
RECEPCIONISTA.....	25
Ver pacientes.....	25
Modificar citas.....	26
MÉDICO.....	27

Ver pacientes.....	27
Administrar tratamiento.....	27
Ver citas.....	28
PACIENTE.....	29
Pedir cita.....	29
Historial de citas.....	30
<b>6.2.- Arquitectura del sistema.....</b>	<b>31</b>
<b>6.3.- Diagramas de clases.....</b>	<b>32</b>
<b>6.4.- Diseño de la base de datos: esquemas y tablas.....</b>	<b>32</b>
Relaciones clave.....	34
<b>7.- Desarrollo de la aplicación.....</b>	<b>35</b>
<b>7.1.- Tecnologías y herramientas utilizadas (lenguajes de programación, frameworks, bases de datos, etc.).....</b>	<b>35</b>
Symfony (PHP).....	35
Angular (TypeScript).....	35
Docker.....	35
MySQL.....	35
DBeaver.....	35
Doctrine ORM.....	36
GitHub.....	36
Cursor.....	36
<b>7.2.- Descripción de las principales funcionalidades implementadas ilustrándolo con fragmentos de código relevantes.....</b>	<b>37</b>
Seguridad.....	37
Interceptor JWT en Angular.....	38
Administrador.....	40
Doctor.....	40
Recepcionista.....	41
Paciente.....	43
<b>8.- Planificación del proyecto.....</b>	<b>46</b>
<b>8.1.- Acciones.....</b>	<b>46</b>
<b>8.2.- Temporalización y secuenciación.....</b>	<b>47</b>
<b>9.- Pruebas y validación.....</b>	<b>47</b>
<b>10.- Relación del proyecto con los módulos del ciclo.....</b>	<b>51</b>
<b>11.- Conclusiones.....</b>	<b>52</b>
<b>12.- Proyectos futuros.....</b>	<b>53</b>
Ampliaciones y mejoras.....	53
Nuevos proyectos futuros.....	53
<b>13.- Bibliografía/Webgrafía.....</b>	<b>54</b>

# Introducción

Este proyecto de fin de ciclo del Grado Superior en Desarrollo de Aplicaciones Web consiste en el diseño y desarrollo de una página web para una clínica privada.

La motivación principal que impulsó esta elección fue la creciente necesidad de adaptación tecnológica en el sector sanitario. Con el paso de los años, es cada vez más habitual que clínicas y centros médicos incorporen plataformas digitales que faciliten la relación con sus pacientes. Entre estas herramientas destacan las webs corporativas que permiten la gestión de citas médicas a distancia, de forma lógica, rápida, sencilla y accesible a todos los pacientes.

El objetivo principal de este proyecto es ofrecer una solución web que facilite la interacción entre pacientes y clínica, promoviendo una experiencia más ágil, segura y alineada con las exigencias del entorno digital actual.

## Descripción y contexto del proyecto

Este proyecto se enmarca en el desarrollo de un sistema organizativo integral para una clínica privada, con el objetivo de digitalizar la gestión de citas y facilitar la interacción entre los distintos perfiles que intervienen en el funcionamiento del centro médico. Se ha diseñado una interfaz web completa y adaptada a cuatro tipos principales de usuario: **pacientes, médicos, recepcionistas y administradores.**

Para garantizar un acceso seguro y personalizado, se ha optado por un sistema de autenticación basado en **DNI y contraseña**. Los pacientes pueden registrarse utilizando su DNI y, una vez autenticados, tienen la posibilidad de solicitar citas médicas, consultar su historial de citas, cancelarlas en caso necesario, visualizar los servicios disponibles junto al equipo médico y sanitario y poder acceder tanto a información sobre las calorías ganadas y perdidas al ingerir alimentos o hacer ciertos deportes por un tiempo según su peso.

El perfil de **médico** permite crear y gestionar los servicios médicos y consultar citas programadas.

Los **encargados de Recepción** tienen acceso a los datos esenciales de los pacientes, lo cual les permite confirmar o cancelar citas de forma eficiente, sin comprometer la privacidad ni acceder a información médica sensible, al mismo tiempo, permite acceder a los presupuestos y gestionar el cobro y el envío.

Por último, el perfil de **administrador** dispone de permisos avanzados para crear cuentas de usuario para médicos, personal sanitario y administrativo. Esto permite por ejemplo, otorgar acceso temporal a determinadas funcionalidades en situaciones específicas, manteniendo siempre la seguridad y el control del sistema.

En conjunto, este proyecto simula un entorno profesional realista que integra la lógica fundamental de una plataforma moderna de gestión clínica.

## 1.2.- Motivación del proyecto

La elección de desarrollar una página web para una clínica privada responde tanto a criterios de **viabilidad técnica** como de **interés personal**. Opté por un enfoque más realista y manejable, centrado en una clínica privada, que permite aplicar los conocimientos adquiridos de forma efectiva y organizada.

El proyecto busca **resolver un problema real y actual**: la falta de digitalización efectiva a nivel usuario de algunos centros médicos del ámbito privado. Especialmente lo he dirigido a la **gestión de citas y la interacción con el paciente**. Hoy en día, la sociedad demanda cada vez más que las herramientas digitales estén adaptadas de forma predictiva a cualquier usuario sin que tenga que tener conocimientos informáticos y resulte fácil acceder a una plataforma médica de forma autónoma para el paciente de cualquier edad. Permite solicitar cita médica online, sin tener que acudir presencialmente o pasar largos periodos de espera para ser atendido. Esta es una **comodidad esencial** para muchos usuarios, y un estándar en el entorno tecnológico actual.

Además, el uso de soluciones web que integren bases de datos eficientes permite mejorar la atención, reducir errores y centralizar la información de forma ordenada y accesible, lo cual es especialmente relevante en un sector tan sensible y dinámico como el sanitario.

Por otro lado, esta elección está estrechamente vinculada a mis motivaciones personales. Me interesa profundamente el ámbito de la salud y, a largo plazo, me gustaría **combinar la informática con sectores como la genética, la biología y la medicina**, donde el tratamiento y análisis de grandes volúmenes de datos es clave para su evolución. Estos campos, aunque tradicionalmente más alejados del desarrollo web, pueden beneficiarse enormemente de herramientas tecnológicas adaptadas a sus necesidades.

En resumen, este proyecto no solo representa una solución **realista y técnicamente viable**, sino también una forma de **responder a las nuevas demandas sociales y tecnológicas** del sector sanitario, al tiempo que se alinea con mis inquietudes profesionales y personales.

## 1.3.- Beneficios esperados

La finalización con éxito de este proyecto conlleva una serie de beneficios tanto a nivel técnico como práctico, aplicables al entorno sanitario y a mi desarrollo profesional:

- **Mejora de la eficiencia en la gestión clínica:** La digitalización de procesos clave —como la solicitud, consulta y cancelación de citas— permite reducir la carga administrativa del personal, minimizar errores humanos y optimizar el tiempo de atención al paciente.
- **Facilidad de acceso para los usuarios:** Los pacientes podrán gestionar sus citas de forma autónoma y remota, evitando desplazamientos innecesarios y largas esperas telefónicas. Esto mejora significativamente la experiencia del usuario, una característica cada vez más valorada en el contexto actual.
- **Segmentación y control de roles de usuario:** El sistema establece de forma clara los permisos y accesos según cada perfil (paciente, médico y administrativo), garantizando una gestión segura, organizada y conforme a principios de privacidad y eficiencia operativa.
- **Aplicación e integración de conocimientos técnicos:** El desarrollo del proyecto ha sido una oportunidad para poner en práctica los conocimientos adquiridos durante el ciclo formativo: diseño de bases de datos, desarrollo web, seguridad, diseño de interfaces, gestión de usuarios y despliegue de aplicaciones.
- **Proyección hacia el mundo profesional:** Al simular un entorno clínico funcional, he aprendido a traducir la lógica de negocio del sector sanitario a soluciones informáticas reales. Este proyecto formará parte de mi portafolio técnico, lo cual resulta especialmente útil para futuras entrevistas o procesos de selección.
- **Ampliación del conocimiento en el ámbito sanitario:** La investigación necesaria para desarrollar este sistema me ha permitido comprender mejor la organización interna de una clínica, los flujos de trabajo sanitarios y la interacción entre los diferentes perfiles profesionales. Este acercamiento práctico complementa mi interés personal por áreas como la biología, la genética y la medicina, donde la informática puede aportar un gran valor.
- **Potencial de escalabilidad y mejora:** Aunque inicialmente está pensado para una clínica privada de tamaño medio, su diseño modular permite escalar el

sistema, adaptarlo a nuevas necesidades o integrarlo con otros servicios externos sin comprometer su estructura base.

En conjunto, este proyecto no solo cumple una función académica, sino que ofrece una solución realista, útil y escalable. Al mismo tiempo, refuerza mis competencias profesionales y consolida mi motivación por aplicar la informática en sectores clave como el sanitario



## 2.- Objetivo/s generales del proyecto

Desarrollar una aplicación web funcional y segura que permita gestionar de forma eficiente los procesos internos de una clínica privada, facilitando la interacción entre los distintos perfiles de usuario (pacientes, médicos , administradores y recepcionistas), y contemplando todas las etapas necesarias del desarrollo: desde el análisis de requisitos hasta su diseño, despliegue ,implementación, validación y documentación.

## 3.- Objetivos específicos

- Identificar y analizar los requisitos funcionales y no funcionales de la aplicación web para la clínica privada.
- Diseñar modelos que representen las distintas perspectivas y funcionalidades del sistema, tales como diagramas de casos de uso, diagramas de clases y esquemas de base de datos.
- Seleccionar una arquitectura software adecuada que garantice la seguridad, escalabilidad y eficiencia del sistema.
- Implementar la interfaz gráfica y las funcionalidades de backend para los distintos perfiles de usuario: pacientes, médicos, recepcionistas y administradores.
- Planificar y organizar las actividades necesarias para el desarrollo, pruebas y despliegue del proyecto, asegurando una correcta gestión del tiempo y los recursos.
- Validar y testear la aplicación para garantizar su correcto funcionamiento y cumplimiento de los requisitos establecidos.

## 4.- Contexto actual

### **Estado del arte**

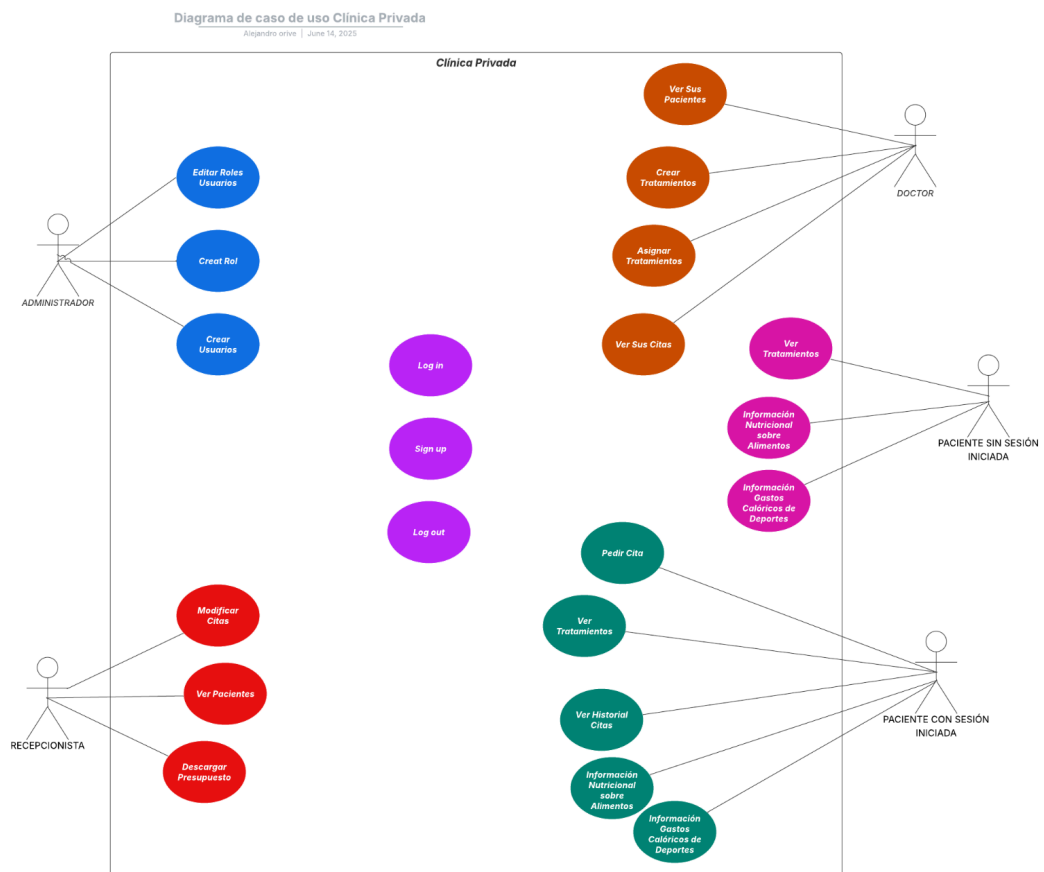
Actualmente, muchos sistemas ofrecen plataformas digitales para la reserva y gestión de citas médicas, pero en algunos casos presentan deficiencias como interfaces poco intuitivas, procesos complicados para los usuarios o falta de integración entre perfiles (pacientes, médicos, administración). Estas limitaciones motivan la necesidad de desarrollar soluciones más eficientes, accesibles y adaptadas a las demandas actuales, en las que la rapidez, seguridad y facilidad para pedir citas sin esperas prolongadas sean una prioridad.

### **Conceptos clave**

Se definirán y aplicarán los términos fundamentales relacionados con la gestión clínica digital, tales como “cita médica online”, “perfil de usuario”, “autenticación mediante DNI”, “gestión de roles y permisos”, “base de datos clínica” y “seguridad en aplicaciones web”. Estos conceptos son esenciales para comprender el alcance y los retos técnicos del proyecto, además de establecer una base teórica sólida que guíe el diseño y desarrollo del sistema.

## 5.- Análisis de requisitos

### 5.1.- Diagrama de casos de uso.



## 5.2.- Requisitos funcionales principales: funcionalidades que debe tener la aplicación.

### Administrador

- Crear cuentas de usuario para médicos y otros administradores.
- Crear nuevos roles personalizados en caso de necesidad.
- Editar o modificar los roles asignados a los distintos usuarios, permitiendo así una gestión flexible y segura de los permisos.

### Médico

- Visualizar la lista de pacientes que tienen citas asignadas con él.
- Crear nuevos tratamientos que ofrece dentro de la clínica.
- Asignar tratamientos (servicios) a médicos.
- Visualizar el historial de citas en las que ha participado como profesional.

### Paciente

- Solicitar una cita con un médico disponible en la clínica.
- Consultar los tratamientos que ofrece la clínica y aquellos que le han sido asignados.
- Visualizar su historial de citas anteriores, con fecha, profesional asignado y estado de la cita , además de su cancelación.
- Acceso a datos detallados sobre el contenido nutricional de diversos alimentos.
- Acceso a información sobre el gasto calórico generado al practicar diferentes deportes, considerando la duración de la actividad y el peso corporal.

## Recepcionista

- Modificar, confirmar o cancelar citas según las necesidades de la clínica o del paciente.
- Acceder a la información básica de los pacientes para la correcta gestión de citas sin vulnerar la confidencialidad.
- Descargar y gestionar los presupuestos asociadas a las citas médicas y tratamientos realizados.

## 5.3.- Requisitos no funcionales: rendimiento, seguridad, usabilidad, etc.

### 1. Rendimiento

El sistema está optimizado para garantizar en todo momento un tiempo de respuesta medio inferior a un segundo para las operaciones más comunes.

### 2. Seguridad

En el proyecto se añadió la autenticación basada en **tokens JWT (JSON Web Tokens)** para garantizar la seguridad en las comunicaciones entre el cliente y el servidor. Esto permite validar la identidad del usuario en cada petición de manera eficiente y segura.

Se implementaron rutas específicas para el **administrador**, protegidas mediante un **guard (guardia)** que verifica el estado de autenticación y los permisos del usuario antes de permitir el acceso. Este guard actúa como un filtro para restringir el acceso a funcionalidades sensibles.

Además, se desarrollaron **interceptores** que interceptan las solicitudes HTTP para agregar automáticamente el token JWT en los encabezados y gestionar respuestas de error relacionadas con la autenticación, como la expiración del token.

Finalmente, se establecieron **restricciones de roles** en el backend para asegurar que solo usuarios con permisos adecuados puedan ejecutar ciertas acciones, asegurando así un control de acceso robusto y personalizado según el tipo de usuario.

### **3. Disponibilidad**

Para garantizar la disponibilidad del sistema, se implementaron estrategias que aseguran que el servicio esté operativo y accesible para los usuarios en todo momento. Entre ellas, se incluyeron mecanismos de manejo de errores, así como el uso de sesiones persistentes y autenticación segura para evitar interrupciones inesperadas. Además, la arquitectura modular permite mantener y actualizar componentes sin afectar el servicio global, lo que mejora la continuidad operativa.

### **4. Usabilidad**

El diseño de la interfaz se enfocó en la simplicidad y facilidad de uso, teniendo en cuenta que muchos usuarios pueden ser de edad avanzada y no estar familiarizados con tecnologías complejas. Por ello, se prioriza una experiencia intuitiva, con elementos visuales claros, botones grandes y textos legibles. Se evitó la sobrecarga de información en pantalla para reducir la confusión y facilitar la navegación. Además, se consideraron contrastes adecuados y una estructura lógica para que los usuarios puedan completar tareas con el menor esfuerzo posible, mejorando así la accesibilidad y satisfacción general.

### **5. Trazabilidad**

Se implementó trazabilidad para registrar y seguir el historial de acciones realizadas dentro del sistema. Esto incluye el registro de eventos importantes como cambios en datos, accesos de usuarios, y operaciones críticas, permitiendo así auditar y rastrear cualquier modificación o incidencia.

## 5.4.- Descripción de los usuarios y sus necesidades.

- **Administrador:**  
Responsable de la configuración general del sistema, la gestión de usuarios y la supervisión de las actividades realizadas. Requiere acceso completo a la sección administrativa destinada a la gestión de usuarios y demás funcionalidades administrativas.
- **Doctor:**  
Utiliza el sistema para gestionar citas, consultar el historial médico de los pacientes y registrar tratamientos. Necesita un acceso rápido, seguro y eficiente a la información clínica para poder desempeñar sus funciones adecuadamente.
- **Recepcionista:**  
Encargado de registrar pacientes, gestionar citas y atender solicitudes administrativas, incluyendo la gestión de presupuesto en formato PDF. Requiere una interfaz sencilla y eficiente que facilite la realización de estas tareas cotidianas.
- **Paciente:**  
Usuario final que puede acceder a su información personal, historial médico, citas, información sobre deportes y gasto calórico según tiempo y peso, así como descripción nutricional de alimentos. Necesita una experiencia intuitiva y segura para consultar sus datos y llevar a cabo gestiones básicas de manera autónoma.

Además, el diseño del sistema permite asignar múltiples roles a un mismo usuario, así como añadir o eliminar roles de forma dinámica. Esta flexibilidad facilita que ciertos usuarios puedan desempeñar funciones adicionales temporalmente, adaptándose a diferentes necesidades operativas sin la necesidad de crear nuevas cuentas de usuario.

## 6.- Diseño de la aplicación

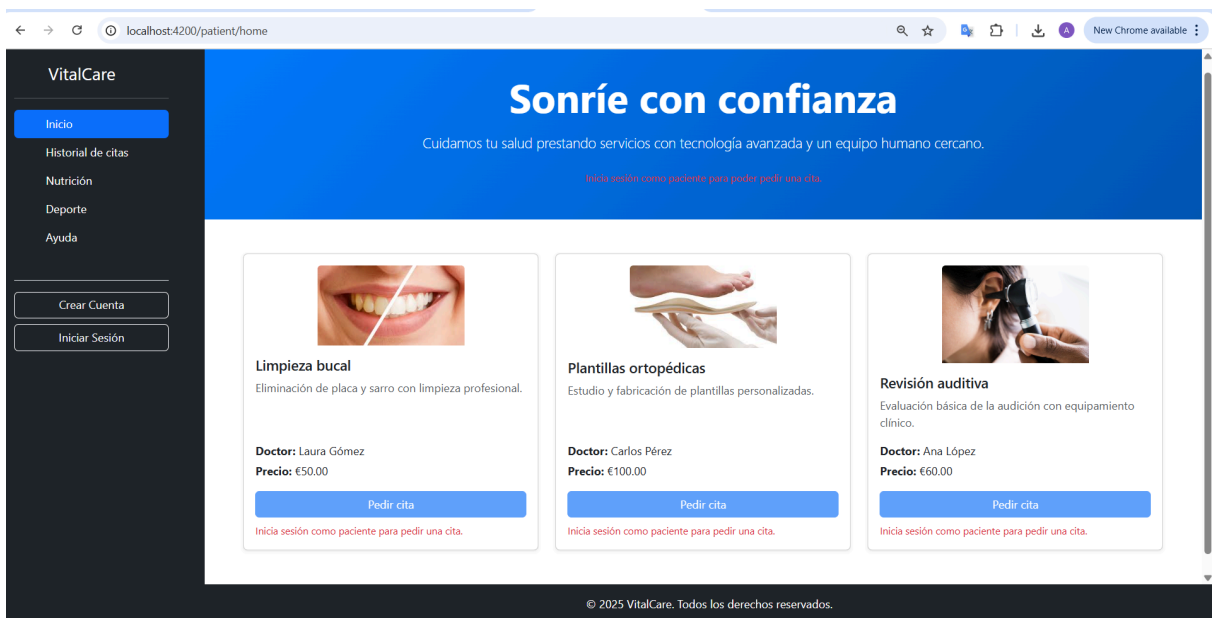
- **Angular:** Se utilizó Angular como framework frontend por su capacidad para crear interfaces dinámicas, reactivas y escalables. Angular facilita el desarrollo de aplicaciones SPA (Single Page Application) que ofrecen una experiencia fluida y rápida al usuario, además de contar con una arquitectura modular y herramientas integradas para el manejo eficiente de componentes y rutas.
- **Symfony:** En el backend se seleccionó Symfony debido a su robustez, flexibilidad y amplia comunidad. Symfony permite construir APIs RESTful de manera segura y eficiente, facilita la integración con bases de datos y ofrece una estructura organizada que mejora la mantenibilidad del código. Además, su sistema de bundles permite reutilizar funcionalidades y acelerar el desarrollo.
- **Diseño de Base de Datos:** Se optó por un modelo relacional bien normalizado para asegurar la integridad y consistencia de los datos. La base de datos está diseñada para manejar relaciones entre usuarios, roles, citas y presupuestos , facilitando consultas eficientes y evitando redundancia de información. Se aplicaron técnicas de indexación para optimizar el rendimiento en las operaciones más frecuentes.



## 6.1.-Interfaz gráfica de usuario.

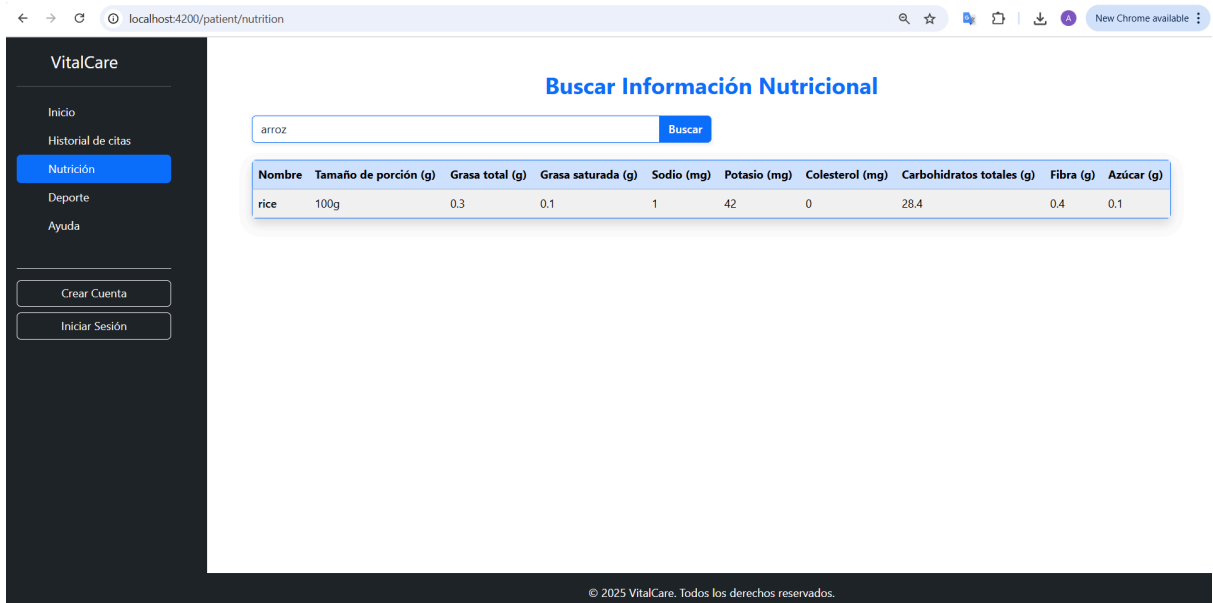
### SIN CUENTA

#### Página principal (sin inicio de sesión)



Sin iniciar sesión el usuario tiene acceso a las siguientes funcionalidades : visualización de tratamientos, acceso a la sección de nutrición para consultar datos nutricionales de alimentos, una sección de ayuda, un módulo para obtener información sobre gastos calóricos según el deporte practicado, el peso y la duración , crear cuenta (crear paciente) y inicio de sesión.

## Nutrición

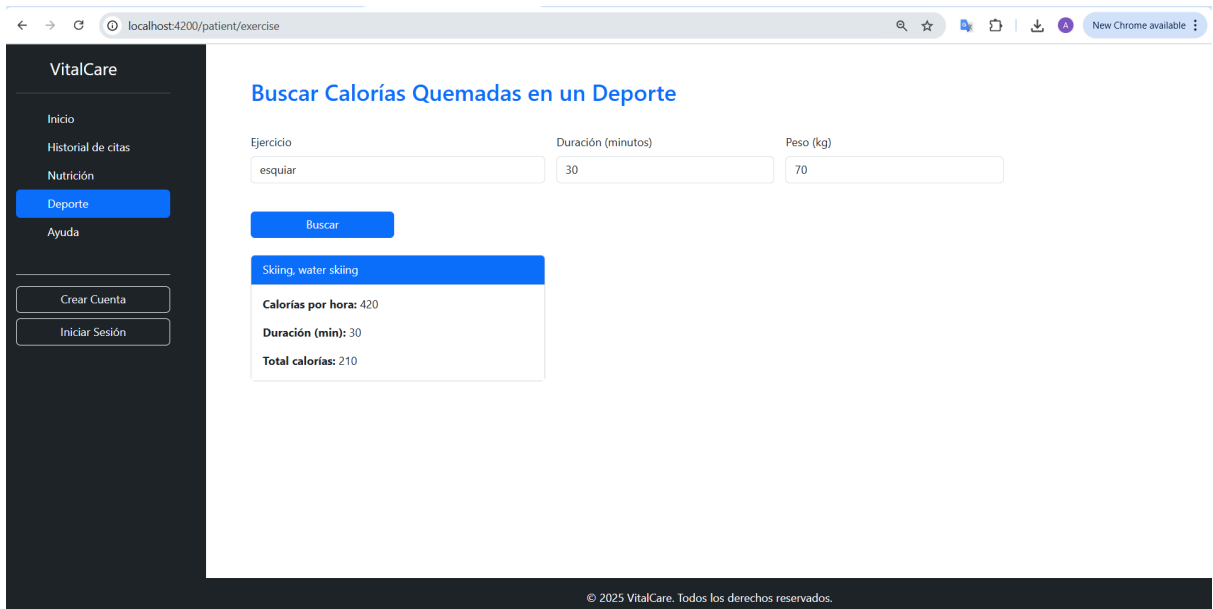


Nombre	Tamaño de porción (g)	Grasa total (g)	Grasa saturada (g)	Sodio (mg)	Potasio (mg)	Colesterol (mg)	Carbohidratos totales (g)	Fibra (g)	Azúcar (g)
rice	100g	0.3	0.1	1	42	0	28.4	0.4	0.1

© 2025 VitalCare. Todos los derechos reservados.

En este apartado puedes escribir un alimento en la barra de búsqueda , que se conectará mediante una api a un servicio que traerá información nutricional sobre esa comida en proporciones de 100 g al darle al botón de buscar.

## Deportes



Buscar Calorías Quemadas en un Deporte

Ejercicio: esquiar      Duración (minutos): 30      Peso (kg): 70

Buscar

Skiing, water skiing

Calorías por hora: 420

Duración (min): 30

Total calorías: 210

© 2025 VitalCare. Todos los derechos reservados.

En este apartado te permite introducir un deporte , un peso y duración ,al darle al botón buscar se conectará mediante una api a un servicio que traerá información de la cantidad de calorías quemadas según los parámetros introducidos.

## Ayuda

VitalCare

Inicio  
Historial de citas  
Nutrición  
Deporte  
**Ayuda**

Crear Cuenta  
Iniciar Sesión

### Preguntas frecuentes

¿Qué debo llevar a mi cita?

Solo necesitas tu DNI y llegar 10 minutos antes.

¿Puedo cancelar una cita?

Sí, puedes hacerlo desde tu historial de citas.

¿Cómo se paga?

El pago se realiza en recepción, en efectivo o tarjeta.

¿Cuándo recibo la factura?

Inmediatamente después de pagar se le entregará una copia física, además de mandarse a su correo digitalmente en el caso de tener el correo registrado.

© 2025 VitalCare. Todos los derechos reservados.

En esta sección encontrarás respuestas a las preguntas más frecuentes y orientación para el uso del sistema. Aquí podrás aprender a:

## Crear cuenta

VitalCare

Inicio  
Historial de citas  
Nutrición  
Deporte  
Ayuda

Crear Cuenta  
Iniciar Sesión

### Registro de Usuario

Correo electrónico

Email

Contraseña \*

Contraseña

DNI\*

DNI

Nombre \*

Nombre

Apellido \*

Apellido

Teléfono \*

Teléfono

Fecha de nacimiento \*

dd/mm/yyyy

Registrar

© 2025 VitalCare. Todos los derechos reservados.

Aquí los pacientes pueden iniciar sesión. El correo electrónico no es obligatorio, pero si desean recibir el presupuesto por correo, deberán proporcionar. De lo contrario, el presupuesto se entregará en mano al momento de confirmar la cita con recepción.

## Inicio de sesión

The screenshot displays the VitalCare login interface. On the left, a dark sidebar contains the VitalCare logo, a list of navigation links (Inicio, Historial de citas, Nutrición, Deporte, Ayuda), and two buttons: 'Crear Cuenta' and 'Iniciar Sesión'. The main content area is white and features a centered login form titled 'Iniciar Sesión'. This form includes two input fields labeled 'DNI' and 'Contraseña', followed by a blue button labeled 'Iniciar Sesión'. The browser's address bar at the top shows the URL 'localhost:4200/patient/sign\_in'. At the bottom of the page, a footer contains the copyright notice '© 2025 VitalCare. Todos los derechos reservados.'

Cualquier usuario debe identificarse utilizando su DNI y la contraseña que haya registrado previamente.

# ADMIN

## Crear usuario

Bienvenido Admin

Crear Usuario

Crear Rol

Editar Usuarios

Estadísticas

Sign out

**Crear Usuario**

Tipo de Usuario

Selecciona un tipo

DNI

Contraseña

Roles

☐ admin ☐ receptionist ☐ doctor ☐ patient

Crear Usuario Cancelar

© 2025 VitalCare. Todos los derechos reservados.

La interfaz para crear usuarios adapta el formulario según el tipo de usuario seleccionado. Es decir, se mostrarán los campos necesarios para registrar correctamente a cada tipo de usuario (recepcionista, administrador o médico), además de asignar los roles correspondientes.

–Administrador:

**Crear Usuario**

Tipo de Usuario

Administrador

DNI

12345635Q

Contraseña

.....

Roles

☒ admin ☐ receptionist ☐ doctor ☐ patient

Crear Usuario Cancelar

–Recepcionista:

### Crear Usuario

Tipo de Usuario

Recepcionista

DNI

12345635Q

Contraseña

.....

Nombre

nombre

Apellido

apellido

Teléfono

123456789

Roles

☐ admin ☒ receptionist ☐ doctor ☐ patient

Crear Usuario

Cancelar

–Doctor:

### Crear Usuario

Tipo de Usuario

Doctor

DNI

12345635Q

Contraseña

.....

Nombre

nombre

Apellido

apellido

Teléfono

123456789

Especialidad

odontólogo

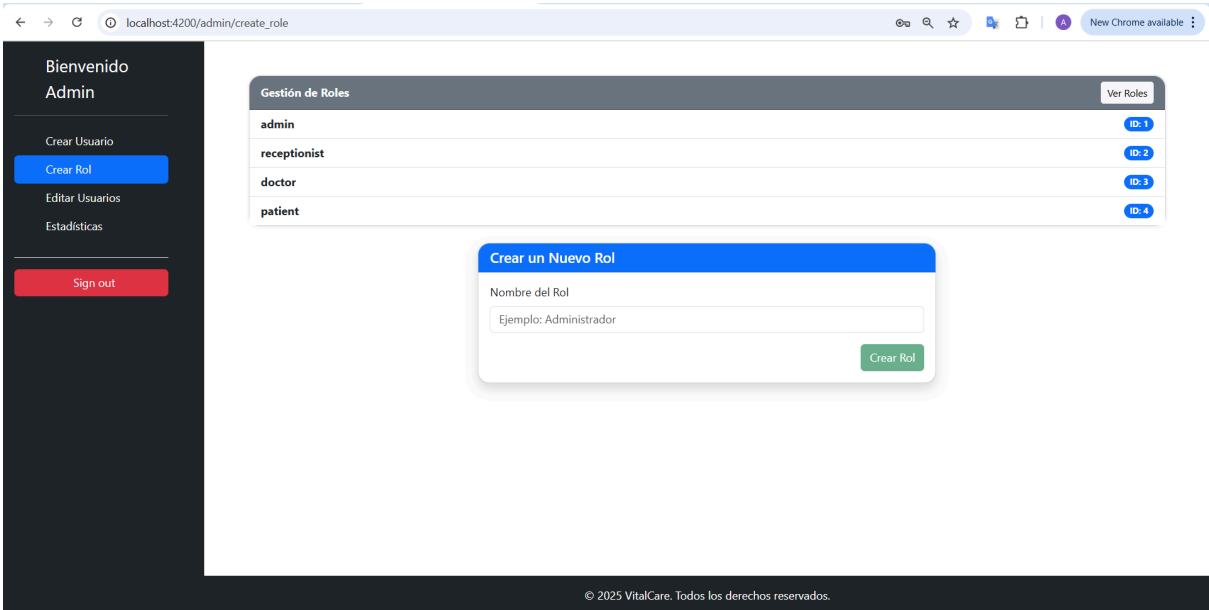
Roles

☐ admin ☐ receptionist ☒ doctor ☐ patient

Crear Usuario

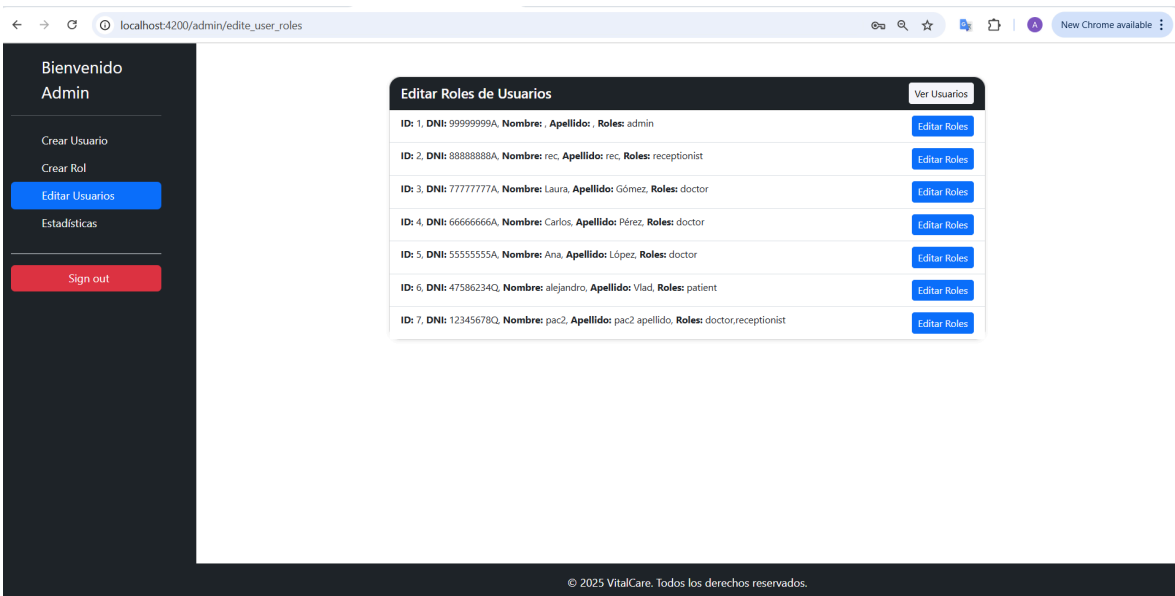
Cancelar

## Crear rol y visualización de roles



Esta sección te permite visualizar los roles existentes además de crearlos

## Editar roles usuarios



Esta sección permite editar los roles asignados a un usuario. Se ha implementado un diseño multirol por cuenta, con el objetivo de cubrir posibles necesidades futuras. Por ejemplo, en caso de que un usuario con un rol determinado no tenga acceso a ciertas funcionalidades, se le pueden otorgar privilegios adicionales para acceder a múltiples interfaces durante un tiempo limitado.

## Editar Roles de Usuario

☐ admin

☒ receptionist

☒ doctor

☐ patient

Cancelar

Guardar Cambios

## Datos estadísticos



Datos estadísticos de interés.



# RECEPCIONISTA

## Ver pacientes

Bienvenido  
Recepcionista

Ver Pacientes

Modificar Citas

Sign out

Buscar paciente por DNI.

Buscar

Refrescar

Lista de Pacientes

Nombre y apellidos	DNI	Teléfono
alejandro Vlad	47586234Q	722604393
pac2 pac2 apellido	12345678Q	123456789

© 2025 VitalCare. Todos los derechos reservados.

Esta sección te permite ver todos los pacientes , además de poder buscarlos por DNI.

47586234Q

Buscar

Refrescar

### Lista de Pacientes

Nombre y apellidos	DNI	Teléfono
alejandro Vlad	47586234Q	722604393

## Modificar citas

← → ↻ localhost:4200/receptionist/show\_appointments 🔍 ☆ 📄 🗑️ ⓘ New Chrome available

Bienvenido  
Recepcionista

Ver Pacientes

Modificar Citas

Sign out

Buscar paciente por DNI...

Buscar

Refrescar

Historial de Citas

Fecha	DNI Paciente	Email	Doctor	Tratamiento	Paciente	Teléfono	Estado
2025-06-19 11:00:00	47586234Q	abladorive@gmail.com	Laura Gómez	limpieza_bucal	alejandro Vlad	722604393	<div>confirmed</div> <div>Editar</div> <div>PDF</div>
2025-06-27 10:00:00	47586234Q	abladorive@gmail.com	Ana López	revisión_auditiva	alejandro Vlad	722604393	<div>cancelled</div> <div>Editar</div> <div></div>
2025-06-19 09:00:00	12345678Q	avladorive@gmail.com	Laura Gómez	limpieza_bucal	pac2 pac2 apellido	123456789	<div>confirmed</div> <div>Editar</div> <div>PDF</div>

© 2025 VitalCare. Todos los derechos reservados.

← → ↻ localhost:4200/receptionist/show\_appointments 🔍 ☆ 📄 🗑️ ⓘ New Chrome available

Bienvenido  
Recepcionista

Ver Pacientes

Modificar Citas

Sign out

47586234Q

Buscar

Refrescar

Historial de Citas

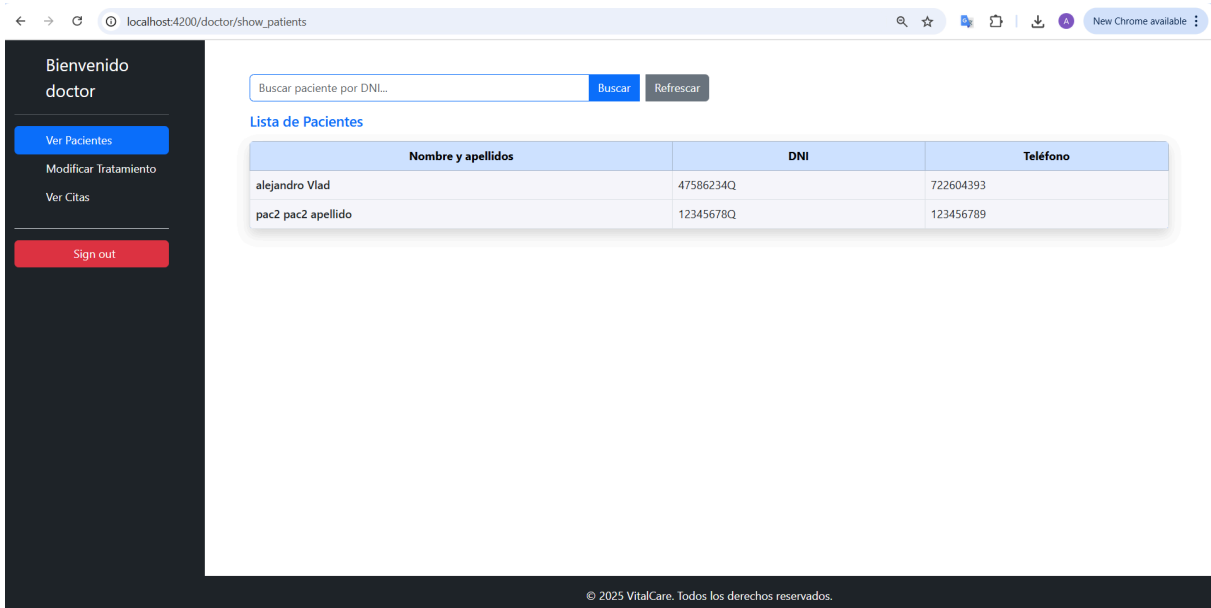
Fecha	DNI Paciente	Email	Doctor	Tratamiento	Paciente	Teléfono	Estado
2025-06-19 11:00:00	47586234Q	abladorive@gmail.com	Laura Gómez	limpieza_bucal	alejandro Vlad	722604393	<div>confirmed</div> <div>Editar</div> <div>PDF</div>
2025-06-27 10:00:00	47586234Q	abladorive@gmail.com	Ana López	revisión_auditiva	alejandro Vlad	722604393	<div>cancelled</div> <div>Editar</div> <div></div>

© 2025 VitalCare. Todos los derechos reservados.

El recepcionista tiene la capacidad de aceptar o cancelar una cita médica y puedes hacer una búsqueda por el dni del paciente que perdió la cita.. En caso de ser aceptada, se genera automáticamente una presupuesto que puede ser descargada directamente desde el sistema y, adicionalmente, enviada al correo electrónico del paciente si se ha proporcionado.

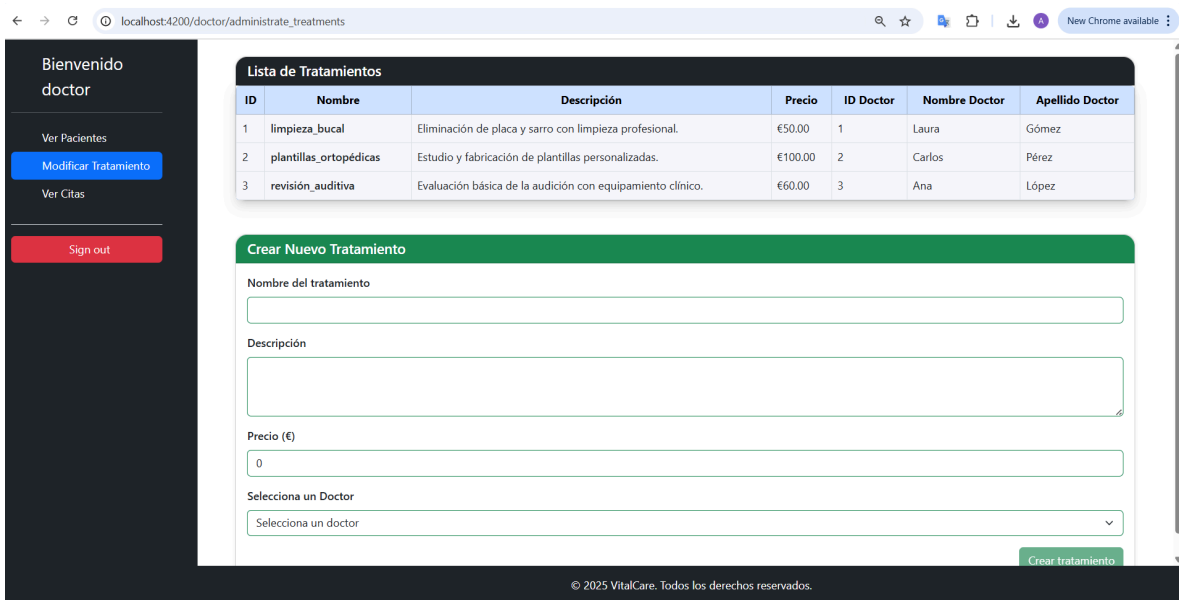
# MÉDICO

## Ver pacientes



Esta sección te permite ver todos los pacientes , además de poder buscarlos por DNI.

## Administrar tratamiento



Esta sección permite ver los tratamientos (servicios) asociados a cada médico. Como he diseñado el sistema para que un tratamiento solo pueda ser realizado por un único médico a la vez, al crear uno nuevo se selecciona exclusivamente al profesional que lo llevará a cabo.

# Ver citas

Bienvenido doctor

Ver Pacientes

Modificar Tratamiento

Ver Citas

Sign out

Buscar paciente por DNI...

Buscar

Refrescar

Historial de Citas

Fecha	DNI Paciente	Nombre Doctor	Apellido Doctor	Tratamiento	Nombre Paciente	Apellido Paciente	Teléfono	Estado
2025-06-19 11:00:00	47586234Q	Laura	Gómez	limpieza_bucal	alejandro	Vlad	722604393	confirmed
2025-06-19 09:00:00	12345678Q	Laura	Gómez	limpieza_bucal	pac2	pac2 apellido	123456789	confirmed

© 2025 VitalCare. Todos los derechos reservados.

En esta sección se muestran únicamente las citas asignadas al doctor que ha iniciado sesión, puedes hacer una búsqueda por el dni del paciente que pedio la cita.

Bienvenido doctor

Ver Pacientes

Modificar Tratamiento

Ver Citas

Sign out

47586234Q

Buscar

Refrescar

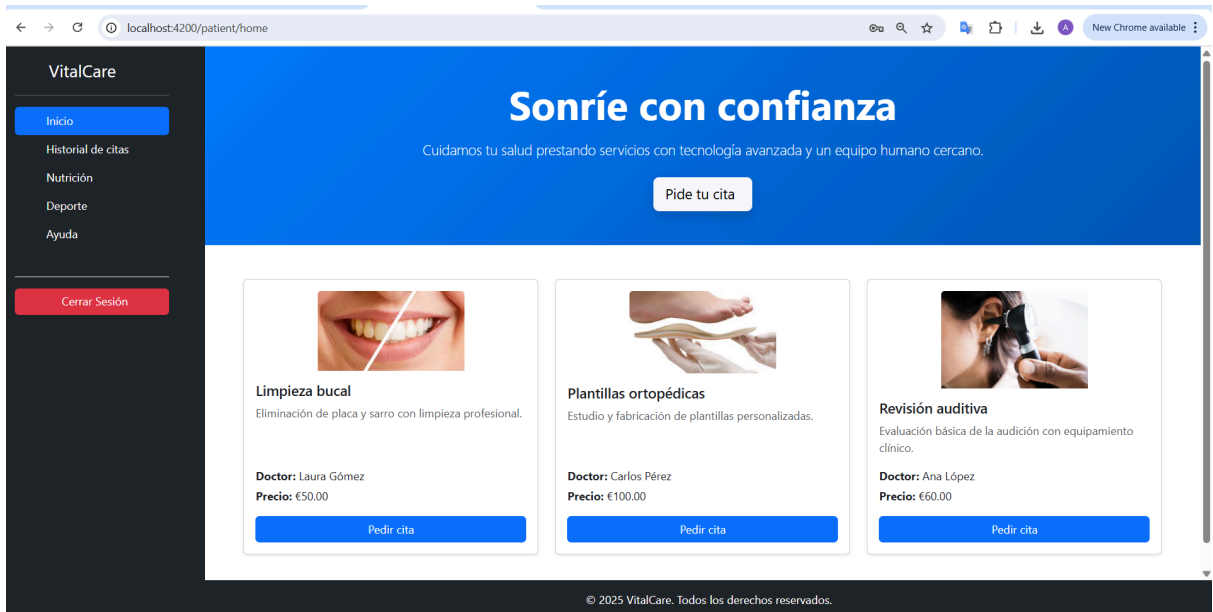
Historial de Citas

Fecha	DNI Paciente	Nombre Doctor	Apellido Doctor	Tratamiento	Nombre Paciente	Apellido Paciente	Teléfono	Estado
2025-06-19 11:00:00	47586234Q	Laura	Gómez	limpieza_bucal	alejandro	Vlad	722604393	confirmed

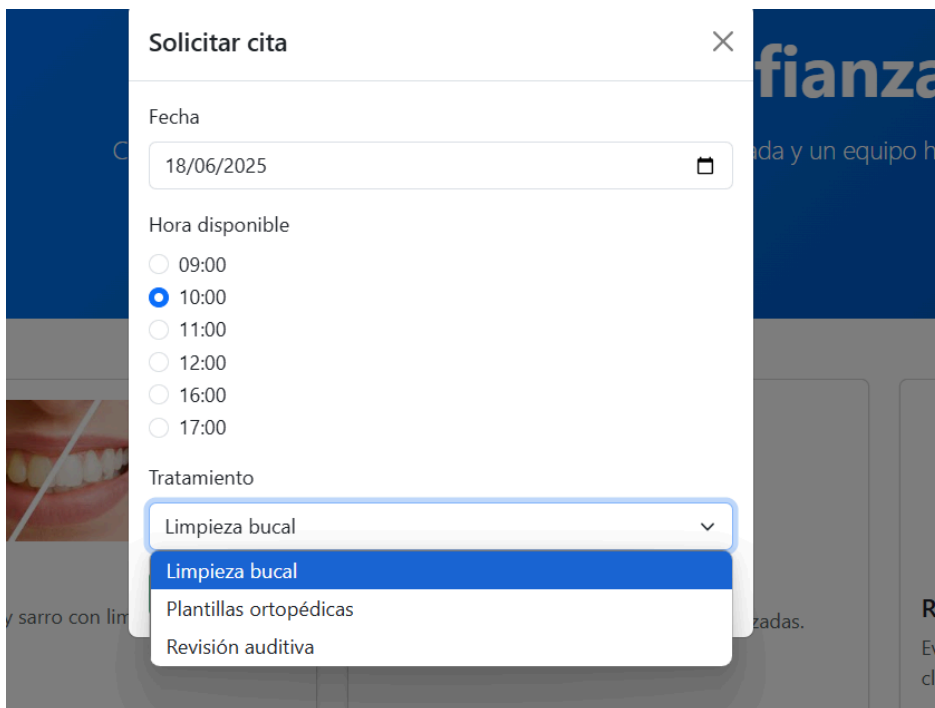
© 2025 VitalCare. Todos los derechos reservados.

## PACIENTE

### Pedir cita



Al solicitar una cita, si haces clic en 'Pide tu cita', se abrirá una ventana emergente (modal) que te permitirá seleccionar la hora, la fecha y el tratamiento. En cambio, si eliges pedir cita desde un tratamiento específico, se abrirá el mismo modal, pero con ese tratamiento ya preseleccionado.



# Historial de citas

VitalCare

Inicio

Historial de citas

Nutrición

Deporte

Ayuda

Cerrar Sesión

Historial de Citas

Fecha	Nombre del Doctor	Apellido del Doctor	Tratamiento	Estado de la Cita	Cancelar Cita
2025-06-19 11:00:00	Laura	Gómez	limpieza_bucal	CONFIRMADA	<div>Cancelar Cita</div>
2025-06-27 10:00:00	Ana	López	revisión_auditiva	CANCELADA	Cita cancelada

© 2025 VitalCare. Todos los derechos reservados.

En esta sección puedes ver el estado de tus citas y cancelarlas. (pendiente ,cancelada o confirmada).

## 6.2.- Arquitectura del sistema.

Este proyecto está basado en una arquitectura **cliente-servidor** organizada bajo el patrón **MVC (Modelo-Vista-Controlador)**, lo cual permite una separación clara de responsabilidades y una mejor mantenibilidad del código.

- **Arquitectura Cliente-Servidor:**

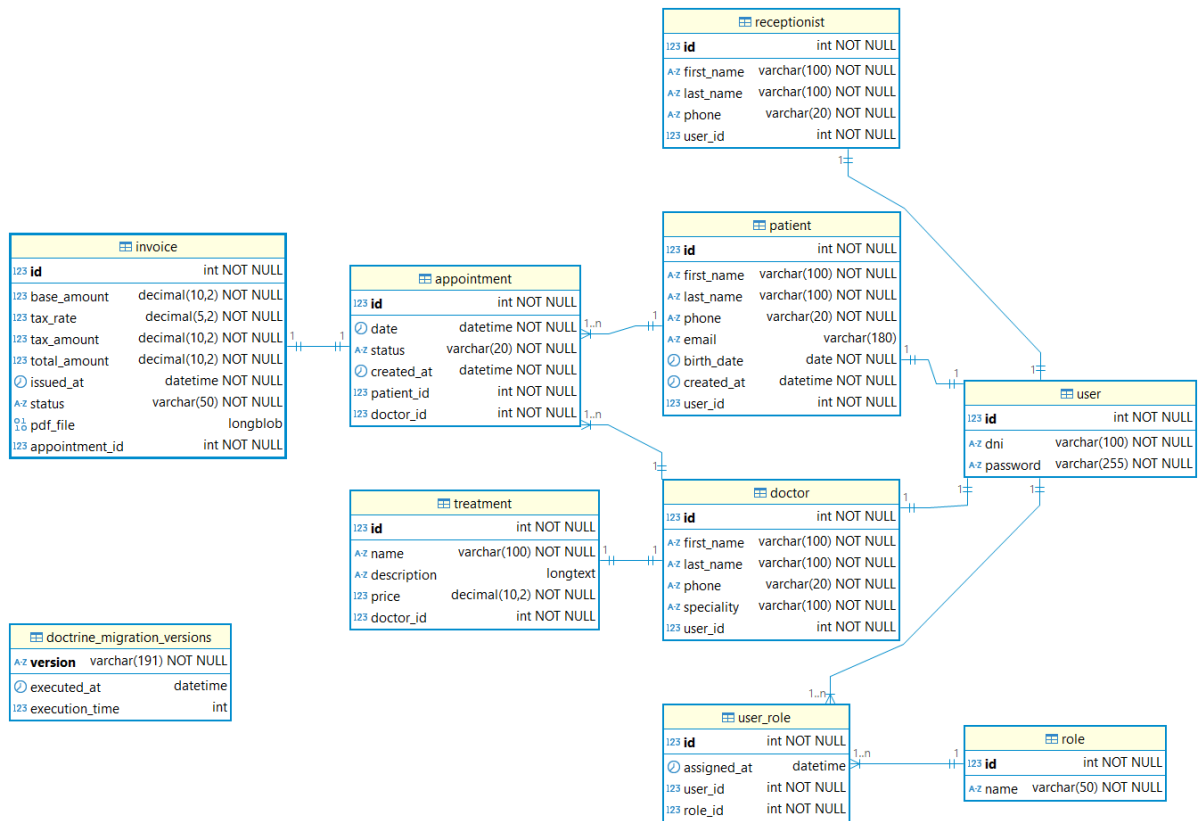
La aplicación se divide en dos partes principales:

- **Cliente (Frontend):** desarrollado con Angular, se encarga de la interfaz de usuario y la interacción directa con el usuario final. Este cliente envía solicitudes HTTP al servidor para acceder o modificar los datos.
- **Servidor (Backend):** implementado con Symfony, expone una API RESTful que procesa las peticiones del cliente, gestiona la lógica de negocio y se comunica con la base de datos.

- **Patrón MVC:**

- **Modelo:** representa los datos del sistema y su lógica. En este caso, se implementa mediante entidades Doctrine que reflejan las tablas de la base de datos.
- **Vista:** en el contexto del backend, no hay vistas tradicionales, ya que la salida del servidor es JSON; sin embargo, en el cliente Angular, se construyen vistas dinámicas que presentan los datos al usuario.
- **Controlador:** gestiona las peticiones entrantes, llama a los servicios necesarios, actualiza los modelos y responde al cliente

## 6.3.- Diagramas de clases.



## 6.4.- Diseño de la base de datos: esquemas y tablas.

- user:**  
 Tabla central que representa a todos los usuarios del sistema. Su clave primaria es **id**.  
 Cada usuario puede tener uno o más roles asociados.
- role:**  
 Contiene los distintos roles del sistema (doctor, paciente, recepcionista, admin, etc.).



La clave primaria es `id`.

- **`user_role`:**  
Es una tabla intermedia que relaciona usuarios con roles (relación N:M).  
Tiene claves foráneas hacia `user(id)` y `role(id)`.
- **`patient`:**  
Almacena la información específica de los pacientes.  
La clave foránea `user_id` establece una relación 1:1 con `user`.
- **`doctor`:**  
Contiene los datos de los doctores.  
También está relacionada 1:1 con la tabla `user` mediante la clave foránea `user_id`.
- **`receptionist`:**  
Representa al personal de recepción.  
Tiene una relación 1:1 con `user` a través de la clave foránea `user_id`.
- **`appointment`:**  
Registra las citas médicas.  
Contiene claves foráneas hacia `patient(id)` y `doctor(id)`, estableciendo así una relación de muchos a uno con ambas entidades.
- **`invoice`:**  
Representa los presupuestos generados a partir de una cita.  
La clave foránea `appointment_id` enlaza con la tabla `appointment`, estableciendo una relación 1:1.
- **`treatment`:**  
Define los tratamientos que ofrece la clínica.  
Cada tratamiento está asociado a un único doctor mediante la clave foránea `doctor_id`.

---

## Relaciones clave

- **1 usuario puede tener múltiples roles**, y cada rol puede estar asignado a **múltiples usuarios** (`user_role` como relación N:M).
- Un **paciente, doctor o recepcionista corresponde a un único usuario**, pero cada uno pertenece a su propia entidad especializada.
- Un **doctor puede tener un tratamiento**, y cada tratamiento pertenece a un **único doctor**.
- Un **paciente puede tener múltiples citas**, al igual que un **doctor puede participar en múltiples citas**, pero **cada cita se asocia a un único paciente y un único doctor**.
- Cada **cita puede generar una** presupuesto , estableciendo una relación 1:1 entre `appointment` e `invoice`.

## 7.- Desarrollo de la aplicación

### 7.1.- Tecnologías y herramientas utilizadas (lenguajes de programación, frameworks, bases de datos, etc.).

#### Symfony (PHP)

Se ha utilizado el framework **Symfony** en el lado del servidor por su robustez, escalabilidad y adherencia al patrón de arquitectura MVC (Modelo-Vista-Controlador). Symfony permite estructurar el backend de forma limpia, reutilizable y testable. Además, su integración con herramientas como **Doctrine ORM** y su sistema de seguridad han facilitado la gestión de roles, autenticación y bases de datos.

---

#### Angular (TypeScript)

Para el desarrollo del frontend se ha empleado **Angular**, un framework moderno de desarrollo web que permite crear interfaces dinámicas y reactivas. Angular ha aportado una estructura clara basada en componentes, así como un sistema eficaz de enrutamiento, servicios y formularios. Su estrecha integración con TypeScript ha mejorado la escalabilidad y mantenibilidad del código.

---

#### Docker

**Docker** ha sido clave para garantizar un entorno de desarrollo coherente y replicable. Se ha utilizado para levantar contenedores de servicios como el backend en Symfony, la base de datos MySQL y otros recursos necesarios. Gracias a Docker, se ha podido desplegar el sistema sin preocupaciones por diferencias de entorno entre máquinas.

---

#### MySQL

Se ha utilizado **MySQL** como sistema gestor de bases de datos relacional por su fiabilidad, rendimiento y amplia compatibilidad con herramientas modernas. Su estructura basada en esquemas y claves foráneas ha permitido modelar correctamente las relaciones entre entidades del dominio de la clínica médica.

---

#### DBeaver

La herramienta **DBeaver** se ha usado como cliente de administración de bases de datos. Ha sido útil para visualizar, modificar y depurar datos de forma visual, así como para realizar pruebas rápidas sobre la estructura de la base de datos.

---

## Doctrine ORM

Dentro de Symfony, se ha empleado **Doctrine** como herramienta ORM (Object-Relational Mapping). Esto ha permitido mapear las clases PHP a las tablas de la base de datos, facilitando la gestión de datos de forma orientada a objetos y reduciendo el uso de consultas SQL explícitas.

---

## GitHub

**GitHub** ha funcionado como sistema de control de versiones y repositorio remoto del proyecto. Gracias a Git, se ha podido gestionar el historial de cambios, trabajar en ramas separadas y mantener un control riguroso del progreso del proyecto. GitHub también permite compartir y presentar el proyecto de forma profesional.

---

## Cursor

Se ha utilizado **Cursor** como entorno de desarrollo (IDE) para facilitar la edición y ejecución del código, así como la organización general del proyecto. Este IDE ofrece una interfaz intuitiva y herramientas integradas que han permitido agilizar el desarrollo, gestionar múltiples archivos y conectarse fácilmente con servicios externos como bases de datos o contenedores. Su uso ha contribuido a una experiencia de programación más fluida y eficiente.

## 7.2.- Descripción de las principales funcionalidades implementadas ilustrándolo con fragmentos de código relevantes.

### Seguridad

**Los guards en Angular** protegen rutas y controlan el acceso según el rol del usuario. Por ejemplo, el **AuthGuard** evita que usuarios no autenticados accedan a zonas privadas, y otros guards como **AdminGuard**, **DoctorGuard** o **ReceptionistGuard** aseguran que solo usuarios con esos roles específicos accedan a sus respectivas secciones. Así se refuerza la seguridad del frontend.

```
Angular > src > app > guards > TS admin.guard.ts > ...
1  import { inject } from '@angular/core';
2  import { CanActivateFn, CanActivateChildFn, Router } from '@angular/router';
3  import { jwtDecode } from 'jwt-decode';
4
5  export const AdminGuard: CanActivateFn & CanActivateChildFn = () => {
6    const router = inject(Router);
7    const token = localStorage.getItem('token');
8
9    if (!token) {
10     router.navigate(['/unauthorized']);
11     return false;
12   }
13
14   const decoded: any = jwtDecode(token);
15   if (decoded.roles?.includes('ROLE_ADMIN')) {
16     return true;
17   }
18
19   router.navigate(['/unauthorized']);
20   return false;
21 };
~
```

```

export const routes: Routes = [
  {
    path: 'admin',
    component: AdminComponent,
    canActivate: [AdminGuard],
    canActivateChild: [AdminGuard],
    children: [
      { path: '', redirectTo: 'create_user', pathMatch: 'full' },
      { path: 'create_user', component: CreateUserComponent },
      { path: 'create_role', component: CreateroleComponent },
      { path: 'edite_user_roles', component: EditeUserRolesComponent },
      { path: 'sign_up', component: SignupComponent },
      { path: 'sign_in', component: SigninComponent },
      { path: 'statistics', component: StatisticsComponent },
    ],
  },
],

```

## Interceptor JWT en Angular

El interceptor se activa automáticamente cada vez que el cliente realiza una petición HTTP. Su función principal es recuperar el token JWT almacenado en `localStorage` y adjuntarlo en la cabecera `Authorization` de la solicitud, utilizando el formato `Bearer`. Esto permite que el backend (Symfony) identifique al usuario autenticado y valide su rol. En caso de no existir token, la solicitud se envía sin cabecera, pero será rechazada por el backend, ya que se ha configurado para requerir obligatoriamente un token válido y con los permisos adecuados (rol autorizado) para acceder a los recursos protegidos.

```

Angular > src > app > interceptors > TS jwt.interceptor.ts > ...
1  import { HttpInterceptorFn } from '@angular/common/http';
2
3  export const JwtInterceptor: HttpInterceptorFn = (req, next) => {
4    const token = localStorage.getItem('token');
5    //console.log(' Interceptor activo, token:', token);
6
7    if (token) {
8      const cloned = req.clone({
9        headers: req.headers.set('Authorization', `Bearer ${token}`),
10      });
11      return next(cloned);
12    }
13
14    return next(req);

```

```

Angular > src > app > TS app.config.ts > appConfig > providers
1  // src/app/app.config.ts
2  import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';
3  import { provideRouter } from '@angular/router';
4  import { provideHttpClient, withInterceptors } from '@angular/common/http';
5  import { routes } from './app.routes';
6  import { JwtInterceptor } from './interceptors/jwt.interceptor';
7
8  export const appConfig: ApplicationConfig = {
9    providers: [
10     provideZoneChangeDetection({ eventCoalescing: true }),
11     provideRouter(routes),
12     provideHttpClient(withInterceptors([JwtInterceptor]))
13   ]
14 };
15

```

La configuración en `lexik_jwt_authentication.yaml` define los parámetros para generar y validar tokens JWT, utilizando claves RSA privadas y públicas junto con una contraseña (`pass_phrase`) que protege la clave privada. Esta configuración permite crear tokens seguros que contienen la información de autenticación del usuario.

En el archivo `security.yaml`, el firewall configurado para las rutas que comienzan con `/api` está marcado como `stateless`, lo que indica que no se mantiene sesión en el servidor y que la autenticación se realiza exclusivamente mediante el token JWT enviado en el encabezado `Authorization` con el esquema Bearer. Esto asegura que cada solicitud a las rutas protegidas requiere un token válido, garantizando la autenticación sin estado.

```

Symfony > config > packages > ! lexik_jwt_authentication.yaml
1  lexik_jwt_authentication:
2    secret_key: '%kernel.project_dir%/config/jwt/private.pem' # Ruta de la clave privada
3    public_key: '%kernel.project_dir%/config/jwt/public.pem' # Ruta de la clave pública
4    pass_phrase: '%env(JWT_PASSPHRASE)%'
5    token_ttl: 3600 # Tiempo de expiración del token en segundos (1 hora)
6

```

```
Symfony > config > packages > ! security.yaml
1  security:
2      password_hashers:
3          App\Entity\User: 'auto'
4
5      providers:
6          app_user_provider:
7              entity:
8                  class: App\Entity\User
9                  property: dni
10
11     firewalls:
12         dev:
13             pattern: ^/(_(profiler|wdt)|css|images|js)/
14             security: false
15
16
17     api:
18         pattern: ^/api
19         stateless: true
20         provider: app_user_provider
21         jwt: ~
22
```

## Administrador

- Crear nuevos usuarios (doctores, pacientes, recepcionistas, admins,etc.)
- Asignar uno o varios roles a cada usuario.
- Gestionar los roles de los usuarios existentes.
- Estadísticas.

---

## Doctor



- Consultar solo sus citas asignadas.
  - Ver los datos del paciente de cada cita.
  - Asociarse a un único tratamiento a la vez.
  - Visualizar información del tratamiento que realiza.
- 

## Recepcionista

- Aceptar o cancelar citas médicas.
- Generar presupuestos tras confirmar una cita.
- Descargar y enviar los presupuestos por correo al paciente.
- Controlar el estado de las citas (pendiente, aceptada, cancelada).

Para la parte de recepción añadí varias librerías nuevas:

**Dompdf** es una biblioteca PHP que permite generar archivos PDF directamente desde código HTML y CSS. Su principal función es **convertir páginas web o plantillas HTML en documentos PDF**, lo que lo hace muy útil para crear informes, presupuestos, tickets, certificados, etc., de forma dinámica en aplicaciones web.

```
private function generatePdf(Invoice $invoice, string $patientName, string $doctorName, string $treatmentName):  
{  
    $options = new Options();  
    $options->set('defaultFont', 'Arial');  
    $dompdf = new Dompdf($options);  
  
    $html = '  
    <html>  
    <head>  
        <style>  
            body {
```

```

$dompdf->loadHtml($html);
$dompdf->setPaper('A4', 'portrait');
$dompdf->render();

return $dompdf->output();

```

**MailerInterface** es una interfaz que ofrece **Symfony** para enviar correos electrónicos de forma sencilla y flexible utilizando el componente **Mailer**. (Usado para mandar el presupuesto al correo del cliente)

Funciona con distintos **transportes de correo** (SMTP, Gmail, SendGrid, etc.).

Se inyecta automáticamente como dependencia para que puedas usarlo desde tus controladores o servicios.

```

$email = (new Email())
    ->from('epmticnotifications@gmail.com')
    ->to($patientMail)
    ->subject('Cita confirmada - Presupuesto adjunta')
    ->text('Hola, tu cita ha sido confirmada. La Presupuesto está adjunta a este correo.')
    ->attach($pdfContent, 'presupuesto.pdf', 'application/pdf');

try {
    $mailer->send($email);
} catch (\Throwable $e) {
    return $this->json([
        'error' => 'No se pudo enviar el correo: ' . $e->getMessage()
    ], 500);
}

```

Symfony > config > packages > ! mailer.yaml

```

1  framework:
2      mailer:
3          dsn: '%env(MAILER_DSN)%'
4          message_bus: false

```



epmticnotifications@gmail.com

Hola, tu cita ha sido confirmada. La factura está adjunta a este correo.



epmticnotifications@gmail.com

para mí ▼



Hola, tu cita ha sido confirmada. La factura está adjunta a este correo.

1 archivo adjunto • Analizado por Gmail ⓘ



## Paciente

- Registrarse en la plataforma.
- Solicitar citas médicas indicando fecha, hora y tratamiento.
- Ver el historial de sus citas.
- Recibir el presupuesto por correo electrónico cuando sea emitida.

### Integración con APIs externas en la sección de paciente

Dentro de la sección destinada al paciente, se implementaron dos integraciones con servicios ofrecidos por la plataforma **APIs Ninja**, conocida por proporcionar una amplia variedad de APIs útiles.

En concreto, se utilizaron dos de ellas:

- Una API que permite **calcular las calorías quemadas** durante la práctica de un deporte, en función del tipo de actividad, el peso del usuario y la duración del ejercicio.

- Otra API que ofrece **información nutricional detallada** de distintos alimentos, permitiendo al paciente conocer el aporte calórico y los macronutrientes de lo que consume.

Cree dos servicios y los llame en un mismo controlador

:

```
public function getNutrition(string $query): array
{
    $response = $this->client->request('GET', 'https://api.api-ninjas.com/v1/nutrition', [
        'headers' => [
            'X-API-Key' => $this->apiKey
        ],
        'query' => [
            'query' => $query
        ]
    ]);

    return $response->toArray();
}
```

```

public function getCaloriesBurned(array $params): array
{
    if (empty($params['activity'])) {
        throw new \InvalidArgumentException('El parámetro "activity" es requerido y no puede estar vacío.');
```

---

```

    }

    $query = [
        'activity' => $params['activity'],
    ];

    if (!empty($params['weight_kg'])) {
        $query['weight'] = $params['weight_kg'] * 2.20462;
    }

    if (!empty($params['duration_minutes'])) {
        $query['duration'] = $params['duration_minutes'];
    }

    $response = $this->client->request('GET', 'https://api.api-ninjas.com/v1/caloriesburned', [
        'headers' => [
            'X-API-Key' => $this->apiKey,
        ],
        'query' => $query,
    ]);

    return $response->toArray();
}

```

```

#[Route('/api/nutrition/{food}', name: 'app_nutrition', methods: "GET")]
public function getData(string $food, NutritionApiService $nutritionApiService, TranslationService $translation): Response
{
    $translatedFood = $translation->translateToEnglish($food);

    $data = $nutritionApiService->getNutrition($translatedFood);

    return $this->json($data);
}

#[Route('/api/exercise', name: 'api_exercise', methods: ['POST'])]
public function getExerciseData(Request $request, ExerciseApiService $exerciseApi, TranslationService $translation): Response
{
    $data = json_decode($request->getContent(), true);

    if (
        !isset($data['activity']) || trim($data['activity']) === '' ||
        !isset($data['duration_minutes']) ||
        !isset($data['weight_kg'])
    ) {
        return $this->json(['error' => 'Faltan parámetros requeridos o están vacíos'], 400);
    }

    $data['activity'] = $translation->translateToEnglish($data['activity']);

    $calories = $exerciseApi->getCaloriesBurned($data);

    return $this->json([
        'calories' => $calories,
    ]);
}

```

## 8.- Planificación del proyecto

### 8.1.- Acciones

#### 1. Toma de requisitos

Se identificaron las necesidades funcionales y no funcionales del sistema, incluyendo los diferentes roles (paciente, doctor, recepcionista, administrador) y las funcionalidades específicas requeridas para cada uno.

#### 2. Análisis de requisitos

Se evaluaron y priorizaron los requisitos obtenidos, estableciendo las relaciones entre entidades, los flujos de navegación y los permisos asociados a cada perfil.

#### 3. Diseño

- **Base de datos:** Se diseñó un modelo relacional adaptado al sistema de gestión de una clínica médica.
- **Interfaz de usuario (frontend):** Se definieron las vistas principales para cada rol utilizando Angular.
- **Arquitectura backend:** Se estructuró el servidor con Symfony siguiendo el patrón MVC, integrando JWT para la autenticación.

#### 4. Implementación (codificación)

- Desarrollo de las entidades y relaciones en Symfony.
- Creación del sistema de autenticación basado en JWT.
- Programación del frontend en Angular, con guards e interceptores.
- Integración con APIs externas para funciones avanzadas como el cálculo de calorías o información nutricional.

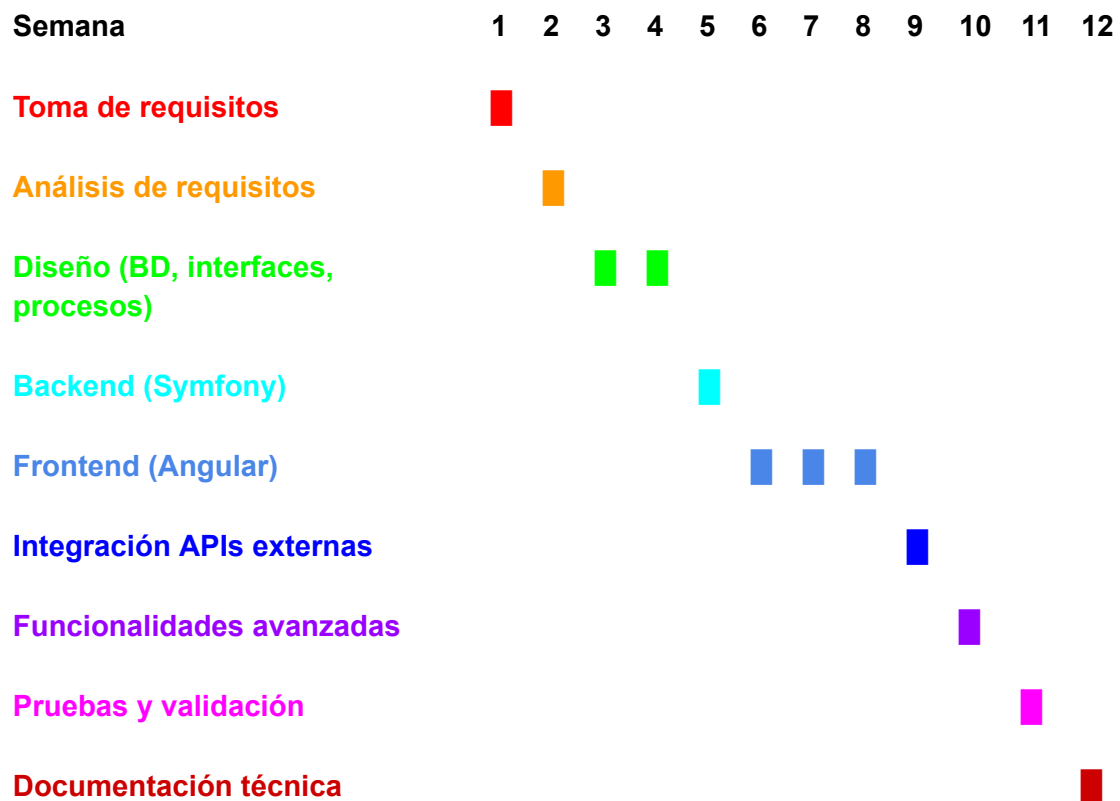
#### 5. Pruebas

Se realizaron pruebas funcionales de cada módulo, asegurando que los flujos de usuario (registro, login, gestión de citas, presupuesto , etc.) funcionaran correctamente, y validando el control de acceso para cada rol con Selenium y JMeter.

## 6. Documentación

Se elaboró la memoria del proyecto, incluyendo la justificación técnica de decisiones, diagramas, ejemplos de uso, planificación temporal y conclusiones finales.

## 8.2.- Temporalización y secuenciación



## 9.- Pruebas y validación

**PHPUnit con WebTestCase (Symfony):** Es una herramienta para pruebas de backend o integración, enfocada en probar la lógica del servidor y APIs.

```

namespace App\Tests\Controller;

use Symfony\Bundle\FrameworkBundle\Test\WebTestCase;
use Symfony\Bundle\FrameworkBundle\KernelBrowser;

class Test1Test extends WebTestCase
{
    private array $adminCredentials = [
        'dni' => '99999999A',
        'password' => '1234567',
    ];

    private function loginAndGetToken(KernelBrowser $client, array $credentials): string
    {
        $client->request(
            'POST',
            '/api/signIn',
            [],
            [],
            ['CONTENT_TYPE' => 'application/json'],
            json_encode($credentials)
        );

        $response = $client->getResponse();
        $data = json_decode($response->getContent(), true);

        return $data['token'] ?? '';
    }

    public function testListUsers(): void
    {
        /** @var KernelBrowser $client */
        $client = static::createClient();
        $token = $this->loginAndGetToken($client, $this->adminCredentials);

        $client->request(
            'GET',
            '/api/getAllUsers',
            server: ['HTTP_Authorization' => "Bearer $token"]
        );

        $this->assertResponseIsSuccessful();
        $this->assertResponseStatusCodeSame(200);
    }
}

```



```
8 class Test1Test extends WebTestCase
49
50 public function testCreateAdmin(): void
51 {
52     /** @var KernelBrowser $client */
53     $client = static::createClient();
54     $token = $this->loginAndGetToken($client, $this->adminCredentials);
55
56     $data = [
57         'dni' => '99999249Z',
58         'password' => 'testpassword',
59         'roles' => [1],
60     ];
61
62     $client->request(
63         'POST',
64         '/api/admin',
65         [],
66         [],
67         [
68             'CONTENT_TYPE' => 'application/json',
69             'HTTP_Authorization' => "Bearer $token"
70         ],
71         json_encode($data)
72     );
73
74     $this->assertResponseStatusCodeSame(201);
75 }
76
```

```

public function testEditRoleUser(): void
{
    /** @var KernelBrowser $client */
    $client = static::createClient();
    $token = $this->loginAndGetToken($client, $this->adminCredentials);

    $data = [
        'id' => 2,
        'role' => ['ROLE_RECEPTIONIST'],
    ];

    $client->request(
        'PATCH',
        '/api/editRoleUser',
        [],
        [],
        [
            'CONTENT_TYPE' => 'application/json',
            'HTTP_Authorization' => "Bearer $token"
        ],
        json_encode($data)
    );

    $this->assertResponseStatusCodeSame(200);
}

```

```

alejandro@Alejandro: ~/proje X alejandro@Alejandro: ~/proje X alejandro@Alejandro: ~/proje X + v
root@946f5a8b0ea6:/var/www/symfony/Symfony# php bin/phpunit
PHPUnit 11.5.23 by Sebastian Bergmann and contributors.

Runtime:      PHP 8.2.28
Configuration: /var/www/symfony/Symfony/phpunit.dist.xml

....                                                4 / 4 (100%)

Time: 00:01.665, Memory: 26.00 MB

OK (4 tests, 5 assertions)
root@946f5a8b0ea6:/var/www/symfony/Symfony#

```

## 10.- Relación del proyecto con los módulos del ciclo

Este proyecto integra y aplica conocimientos adquiridos a lo largo de los distintos módulos del ciclo formativo de grado superior, así como experiencias obtenidas durante el periodo de Formación en Centros de Trabajo (FCT). La relación es la siguiente:

- **Desarrollo Web en Entorno Servidor** : Uso de **Symfony (PHP)** como framework principal para la lógica del backend, implementación de controladores, rutas, servicios, seguridad y conexión con la base de datos mediante Doctrine.
- **Desarrollo Web en Entorno Cliente**: Uso de **Angular** para el desarrollo del frontend, incluyendo componentes, servicios, rutas, formularios reactivos e interceptores.
- **Despliegue de Aplicaciones Web** : Uso de **Docker** para contenerizar el backend y la base de datos (**MySQL**), facilitando el despliegue y pruebas en distintos entornos.
- **Diseño de Interfaces Web** : Diseño de interfaces de usuario con Angular y CSS, siguiendo criterios de usabilidad y organización visual adecuada.
- **Bases de Datos** : Modelado de la base de datos según el dominio de una clínica médica, uso de **MySQL** como sistema gestor y herramientas como **DBeaver** para su gestión y mantenimiento.
- **Formación y Orientación Laboral / Empresa e Iniciativa Emprendedora** : El proyecto simula un entorno profesional real, promoviendo competencias como el trabajo autónomo, la iniciativa, la organización y la documentación técnica.
- **FCT (Formación en Centros de Trabajo)**: La experiencia previa realizando una **intranet corporativa** durante las prácticas me permitió afianzar habilidades como la autenticación de usuarios, estructuración del backend y gestión de roles, que han sido fundamentales en el desarrollo del presente proyecto.

## 11.- Conclusiones

Este proyecto ha sido una experiencia clave en mi proceso de aprendizaje. He podido aplicar y consolidar los conocimientos adquiridos durante el ciclo formativo, tanto a nivel técnico como organizativo. Gracias a él, he aprendido no solo a desarrollar una aplicación funcional con tecnologías modernas como Symfony, Angular, Docker y JWT, sino también a estructurar un sistema completo, con roles de usuario, autenticación, seguridad, y generación de PDFs, además de integrar APIs externas para aportar funcionalidades útiles.

Durante el proceso he descubierto lo crucial que es planificar antes de codificar. A veces he querido avanzar deprisa en la parte técnica sin haber madurado bien los requisitos, y eso me ha hecho retroceder en varias ocasiones. Esta experiencia me ha hecho valorar aún más la fase de análisis y diseño, y me ha dejado claro que una buena planificación es tan importante como el propio código. En el futuro, quiero enfrentarme a nuevos proyectos con una visión más estratégica desde el principio.

En particular, me gustaría abordar proyectos a mayor escala en un futuro, como podría ser el sistema de un hospital. Habiendo trabajado en una pequeña clínica privada, he aprendido muchísimo y me doy cuenta del enorme desafío y aprendizaje que supondría construir algo más grande y complejo. Este proyecto me ha dado mucha motivación para seguir aprendiendo y mejorando.

También he identificado áreas en las que quiero seguir creciendo, como el análisis, automatizado, la integración continua, el conocimiento de librerías, APIs que me permitan reducir tareas manuales, bases de datos y el analizar datos. Cuanto más automatizado y profesional sea el desarrollo, más tiempo tendré para centrarme en la lógica real del negocio.

En definitiva, estoy muy contento con el resultado, pero sobre todo con todo lo que he aprendido por el camino. Este proyecto me ha confirmado que me apasiona el desarrollo backend, y me ha dado confianza para enfrentarme a retos aún mayores.

## 12.- Proyectos futuros

### Ampliaciones y mejoras

Aunque el proyecto está funcional, especialmente en la parte backend, considero que la interfaz frontend es un área con margen de mejora significativa. A pesar de haber invertido bastante tiempo en ella, no estoy seguro de que el resultado haya sido tan bueno como esperaba. El desarrollo frontend me ha resultado más complejo y me ha requerido más esfuerzo comparado con el backend. En futuras versiones, planeo dedicar más tiempo a mejorar la usabilidad, el diseño visual y la experiencia del usuario para que la aplicación sea más intuitiva y atractiva.

Además, sería interesante integrar funcionalidades adicionales, como un sistema de notificaciones en tiempo real para citas, o incorporar análisis de datos clínicos que permitan obtener informes avanzados para los usuarios. También podría ampliarse la gestión de roles y permisos para hacer el sistema aún más seguro y flexible.

---

### Nuevos proyectos futuros

Este proyecto me ha abierto la puerta a nuevas áreas de interés profesional. En particular, quiero ampliar mis conocimientos en análisis de datos, estadísticas y bases de datos, con la intención de especializarme en el ámbito de la ciencia de datos. Por ello, desarrollar un sistema para una clínica donde se manejan grandes volúmenes de datos clínicos y se pueden realizar análisis avanzados me parece un desafío muy atractivo y útil para el futuro. A partir de esta base, puedo abordar proyectos más complejos relacionados con Big Data y Business Intelligence en el sector sanitario u otros sectores que requieran análisis de grandes cantidades de información.

## 13.- Bibliografía/Webgrafía

En la elaboración de este proyecto se han consultado diversas fuentes y documentación técnica que han sido fundamentales para el desarrollo y la implementación de funcionalidades. Es imprescindible citar todas las fuentes utilizadas para dar crédito a sus autores y para permitir la consulta posterior de la información, siguiendo la normativa APA.

**API Ninja:** para la integración de servicios externos relacionados con cálculo de calorías y valores nutricionales.

<https://api-ninjas.com/>

**Bootstrap:** framework CSS para el diseño responsivo y estilizado de la interfaz frontend.

<https://getbootstrap.com/>

**Docker:** para la creación y gestión de contenedores, facilitando la configuración del entorno de desarrollo y despliegue.

<https://www.docker.com/>

**MySQL:** como sistema gestor de base de datos relacional para almacenar la información clínica.

<https://www.mysql.com/>

img:

-limpiezabucal:<https://clinicadodental.com/limpieza-dental-profunda-antes-y-despu>  
[es/](https://clinicadodental.com/limpieza-dental-profunda-antes-y-despu)

-plantillas ortopédicas:<https://www.ortosport.net/productos/pie-calzado/cuidado-pie>

-revisión auditiva:<https://drmiguelmayo.com/osteoma-en-el-oido/>