



47GIIN PROYECTO DE INGENIERÍA DEL SOFTWARE

Proyecto – Presentación final

Moisés Sevilla Corrales
PROF. Doctor Horacio Daniel Kuna

28 de febrero de 2025

Índice

Proyecto – Presentación final	4
Introducción.....	5
1. Detallamos los módulos funcionales y no funcionales.	6
1.1 Especificación de módulos.....	6
1.1.1 Funcionales	6
1.1.2 No funcionales.....	6
1.2 Descripción de los módulos	7
1.2.1 Funcionales	7
1.2.2 No funcionales.....	8
2. Estimación de tamaño por módulo.....	9
2.1 Módulos a programar (debe ser alrededor del 20% del proyecto):	10
2.2 Entorno tecnológico y metodológico.....	10
2.3 Planificación de actividades	11
2.3.1 Planificación SCRUM	12
3. Definición de Backlog en Jira.....	12
4. Definición de nuestra Base de Datos.	13
5. Definimos nuestra estructura en backend con django.....	14
6. Estructurando código y pruebas con Django REST framework.....	17
6.1 Realización de pruebas documentadas.....	18
6.1.1 Pacientes.....	23
6.1.2 Agendar cita médica	23
6.1.3 Consultar disponibilidad de horarios	26
6.1.4 Cancelar o reprogramar citas	27

7.	Definimos los módulos del frontend con Vue.js.....	28
8.	Vistas generales de los módulos	29
8.1	Módulo Pacientes	29
8.2	Módulo Médicos.....	32
8.3	Módulo para Agendar Citas Médicas	33
8.4	Módulo para vista de Citas del Paciente	36
8.5	Módulo Tratamientos	37
8.6	Módulo Gestión del Historial Clínico.....	40
8.7	Módulo Derivaciones.....	42
8.8	Módulo Gestión de Facturas	43
8.9	Módulo Usuarios/Roles	45
8.10	Módulo de Login	46
8.11	Módulo Seguridad y Auditoria	49
8.12	Módulo Seguridad y Auditoria 2	52
8.13	Módulo Proveedores de Material de Laboratorio	53
8.14	Módulo Proveedores de Material Médico	54
9.	Análisis técnico	55
9.1	Análisis de performance con Google Chrome LightHouse	55
9.2	Análisis de accesibilidad con Axe DevTools.....	60
10.	Evaluación heurística.....	61
11.	Mejoras aplicables	62
	Conclusión	63
	Bibliografía	64
	Se aporta firma digital certificada.....	64

Proyecto – Presentación final

Consultorio Médico

Clínica Rehabilita tu alma SA

Proyecto – Presentación final

Introducción

El objetivo de este proyecto es desarrollar un sistema de gestión de reservas específicamente adaptado a consultorios médicos, donde se pueda registrar, gestionar y organizar las citas de los pacientes. El sistema permitirá a los médicos la gestión de citas, en esta entrega final recapitularemos a modo resumen, todo lo visto en las entregas anteriores a fin de plasmar los conceptos más relevantes del proyecto.

En un principio se había planteado realizar el proyecto en un entorno como Java, donde todo es más estático, automatizado y sencillo de realizar e ir a por lo seguro.

Pero se optó por desarrollar el proyecto en un entorno ambicioso desconocido totalmente para mí, la curva de aprendizaje en tan poco tiempo ha sido un gran desafío, se ha usado lenguajes de programación en Python, Javascript, lenguajes de marcado HTML, lenguaje de estilo css, base de datos con Postgre SQL, administrada por pgAdmin, framework de backend Django REST framework para construir APIs RESTful, framework de frontend VUE.js respaldado por Node.js, más una cantidad de librerías añadidas, necesarias para el desarrollo, que iremos nombraremos más adelante.

Algunas de ellas para iniciar el proyecto son:

Asgiref.....	3.8.1
Django.....	5.1.4
django-cors-headers.....	4.6.0
django-filter.....	24.3
django-rest-framework.....	3.15.2
pip.....	24.3.1
psycopg.....	3.2.3
sqlparse.....	0.5.3
tzdata.....	2024.2

1. Detallamos los módulos funcionales y no funcionales.

1.1 Especificación de módulos

1.1.1 Funcionales

Módulo de pacientes.

1. Gestión de pacientes.
2. Registro de historia clínica.

Módulo de citas.

3. Gestión de citas médicas.

Módulo de tratamiento y Derivaciones.

4. Gestión de tratamientos.
5. Derivaciones a especialistas.

Módulos de pagos y facturación.

6. Registro de pagos.
7. Gestión de facturación.

Módulos de usuarios y perfiles.

8. Gestión de usuarios del sistema.
9. Gestión de roles y permisos.

Módulos de proveedores.

10. Gestión de proveedores de laboratorio.
11. Gestión de proveedores de material médico.
12. Gestión de órdenes de compra de proveedores.

Módulo de especialistas.

13. Gestión de especialistas.

1.1.2 No funcionales

14. Reprogramación de citas automática.
15. Seguridad y auditoría.
16. Datos generales

1.2 Descripción de los módulos

1.2.1 Funcionales

1. Gestión de pacientes

Facilitará la creación, consulta, actualización y eliminación de registros de pacientes, incluyendo información personal y de salud.

2. Registro de historia clínica

Facilitará la gestión de la historia clínica de los pacientes, abarcando antecedentes médicos y tratamientos previos.

3. Gestión de citas médicas

Permitirá programar, consultar, modificar y cancelar citas médicas para una mejor organización del tiempo del personal y atención al paciente.

4. Gestión de tratamientos

Ofrecerá funcionalidades para registrar y gestionar tratamientos asignados a los pacientes, incluyendo su seguimiento y resultados.

5. Derivaciones a especialistas

Gestionará las derivaciones de pacientes a especialistas, registrando motivos y seguimiento de estas.

6. Registro de pagos

Facilitará el registro y gestión de pagos de los pacientes, asegurando un control financiero adecuado.

7. Gestión de facturación

Permitirá crear y gestionar facturas relacionadas con consultas y tratamientos, asegurando un proceso de facturación efectivo.

8. Gestión de usuarios del sistema

Facilitará la creación y gestión de usuarios del sistema, asegurando un acceso adecuado al personal médico y administrativo.

9. Gestión de roles y permisos

Definirá y gestionará roles y permisos para los usuarios, asegurando un control de acceso efectivo.

10. Gestión de proveedores de laboratorio

Gestionará información sobre proveedores de laboratorio, incluyendo datos de contacto y productos ofrecidos.

11. Gestión de proveedores de material médico

Permitirá gestionar información sobre proveedores de materiales médicos y sus condiciones de entrega.

12. Gestión de órdenes de compra de proveedores

Facilitará la creación y gestión de órdenes de compra, registrando materiales y fechas de entrega.

13. Gestión de especialistas

Permitirá gestionar información sobre especialistas en el consultorio, incluyendo su especialidad y horarios.

1.2.2 No funcionales

14. Reprogramación de Citas Automática

Facilitará la cancelación y reprogramación de citas médicas, sugiriendo nuevas fechas automáticamente.

15. Seguridad y Auditoría

Gestionará la seguridad del sistema y la auditoría general, asegurando un acceso controlado y registro de eventos.

16. Datos generales

Permitirá a los pacientes poder contactar, localizar y encontrar la ubicación del centro, así como recibir la información solicitada.

2. Estimación de tamaño por módulo

Módulo	Porcentaje de participación / Producto
Gestión de pacientes	6%
Registro de historia clínica	9%
Gestión de citas médicas	7%
Gestión de tratamientos	7%
Derivaciones a especialistas	7%
Registro de pago	5%
Gestión de facturación	8%
Gestión de usuarios del sistema	8%
Gestión de roles y permisos	9%
Gestión de proveedores de laboratorio	5%
Gestión de proveedores de material médico	5%
Gestión de órdenes de compra de proveedores	6%
Gestión de especialistas	8%
Reprogramación de citas automática	4%
Seguridad y auditoría	5%
Datos generales	1%
Total	100%

2.1 Módulos a programar (debe ser alrededor del 20% del proyecto):

Módulos programados (32%)
Módulo de pacientes 6%
Módulo de especialistas 8%
Módulo de gestión de citas médicas 10%
Módulo de gestión de usuarios del sistema 8%

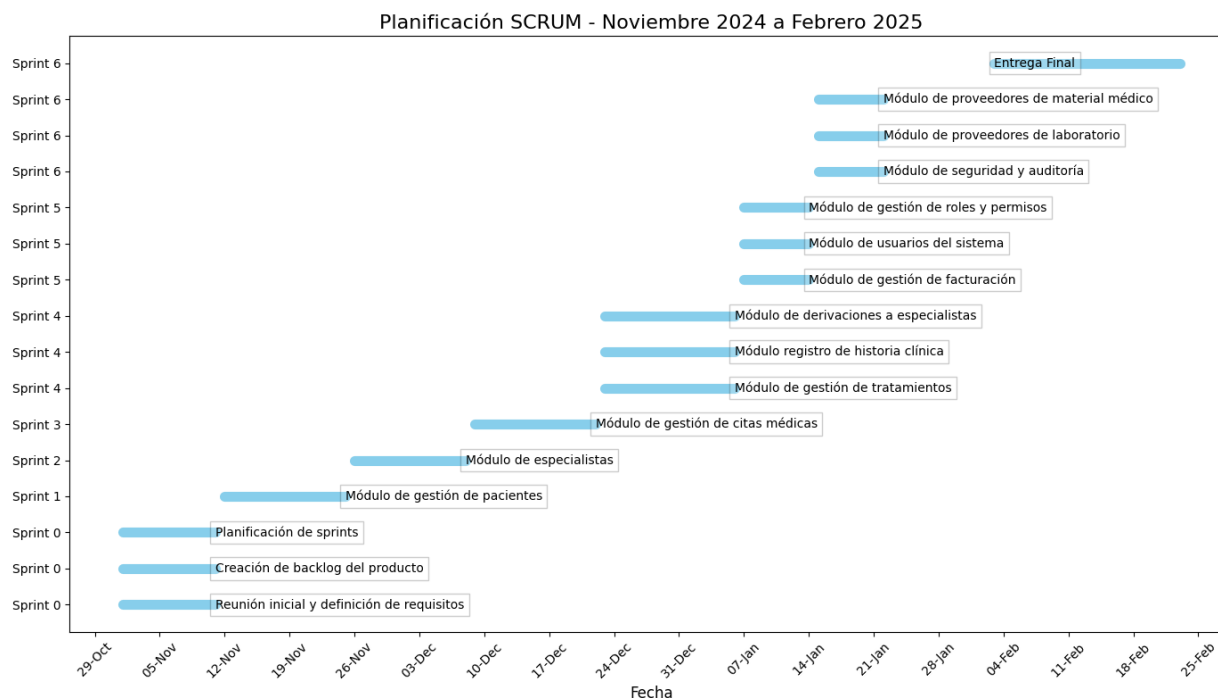
2.2 Entorno tecnológico y metodológico

Lenguaje de programación backend:	Python
Lenguaje de programación frontend:	JavaScript
Lenguaje de marcado:	HTML
Lenguaje de estilo:	CSS
Framework Backend:	Django
Framework Frontend:	Vue.js
Gestión de versionado de código:	GitHub
Gestión de incidencias:	Jira
Arquitectura:	Cliente-servidor
Motor de Base de Datos:	PostgreSQL
Metodología seleccionada:	Scrum
Tipo de proyecto:	Sin cliente final
Sistema operativo usado:	Windows 11 Pro
Entorno de desarrollo:	Visual Studio Code
Navegadores web probados:	Chrome, Edge, Firefox, Opera GX.

2.3 Planificación de actividades

Actividad	Fecha de inicio	Fecha de finalización
Sprint 0: Planificación y preparación		
Reunión inicial y definición de requisitos	01/11/2024	11/11/2024
Creación de backlog del producto	01/11/2024	11/11/2024
Planificación de sprints	01/11/2024	11/11/2024
Sprint 1: Desarrollo inicial		
Módulo de gestión de pacientes	12/11/2024	25/11/2024
Sprint 2: Expansión de funcionalidades		
Módulo de especialistas	26/11/2024	08/12/2024
Sprint 3: Expansión de funcionalidades		
Módulo de gestión de citas médicas	09/12/2024	22/12/2024
Sprint 4: Expansión de funcionalidades		
Módulo de gestión de tratamientos	23/12/2024	06/01/2025
Módulo registro de historia clínica	23/12/2024	06/01/2025
Módulo de derivaciones a especialistas	23/12/2024	06/01/2025
Sprint 5: Reportes y pruebas		
Módulo de gestión de facturación	07/01/2025	14/01/2025
Módulo de usuarios del sistema	07/01/2025	14/01/2025
Módulo de gestión de roles y permisos	07/01/2025	14/01/2025
Sprint 6: Seguridad, ajustes y entrega		
Módulo de seguridad y auditoría	15/01/2025	22/01/2025
Módulos de proveedores de laboratorio	15/01/2025	22/01/2025
Módulo de proveedores de material médico	15/01/2025	22/01/2025
Entrega Final	23/02/2025	03/02/2025

2.3.1 Planificación SCRUM



3. Definición de Backlog en Jira

Empezamos por definir nuestro proyecto con un Backlog en Jira donde seguir con el propósito de nuestros Springs, definimos historias de usuario, incidencias, clasificación de versiones para llevarlo a término, todo ello lo podemos consultar en nuestro enlace a Jira ya que hay infinidad de contenido que mostrar y llenaríamos está memoria, entraremos a través de:

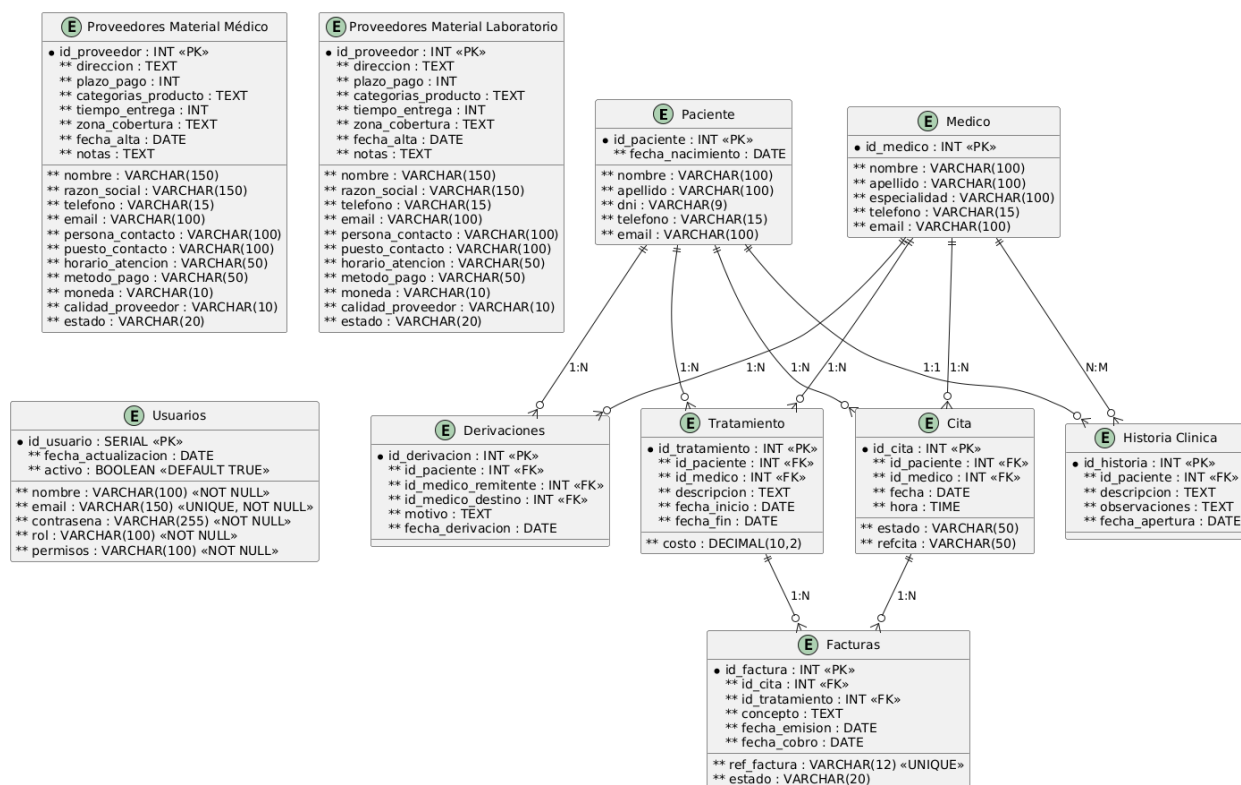
<https://moisessevilla.atlassian.net/jira/software/projects/SGRCM/boards/1>

4. Definición de nuestra Base de Datos.

Comenzamos por definir nuestra base de datos y crearla en postgresql, aquí mostramos el diagrama entidad relación, pero está todo el detalle de las tablas creadas en nuestro archivo bd_consultorio.sql adjunto en nuestro Github:

https://github.com/moisessevilla/47GIIN_Proyecto_Consultorio

- BD: Consultorio
- Tablas: Paciente, Medico, Cita, Derivaciones, Tratamiento, Historia_Clinica, Facturas, Usuarios, Proveedores Material Médico, Proveedores Material Laboratorio.



5. Definimos nuestra estructura en backend con django.

- Archivos en Github: https://github.com/moisessevilla/47GIIN_Proyecto_Consultorio
 - Del entorno Django en backend los más significativos.
 - **models.py** (Estructuras de datos y las tablas)
 - **opciones.html** (Menú básico para tener accesos rápidos)
 - **serializers.py** (Convierte modelos django a JSON y viceversa)
 - **settings.py** (Configuración principal, incluye configuración de la BD)
 - **urls.py** (Mapea las rutas URL)
 - **views.py** (Define la lógica de negocio y proceso de rutas HTTP)
 - Pondremos unos ejemplos de lo que pueden contener los archivos que terminamos de listar:
 - CRUD completo para el módulo de pacientes en **views.py**.
(El mismo ejemplo se aplica para el resto de módulos.):

```
# CRUD Paciente
class PacienteViewSet(viewsets.ModelViewSet):
    queryset = Paciente.objects.all().order_by("id_paciente")
    serializer_class = PacienteSerializer

    def create(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        self.perform_create(serializer)
        return Response(serializer.data, status=status.HTTP_201_CREATED)

    def update(self, request, *args, **kwargs):
        instance = self.get_object()
        serializer = self.get_serializer(instance, data=request.data)
        serializer.is_valid(raise_exception=True)
        self.perform_update(serializer)
        return Response(serializer.data, status=status.HTTP_200_OK)

    def destroy(self, request, *args, **kwargs):
        instance = self.get_object()
        self.perform_destroy(instance)
        return Response(
            {"Mensaje": "Paciente eliminado exitosamente."}, status=status.HTTP_200_OK
        )
```

- En **serializers.py** podemos encontrar validaciones del backend con los datos recibidos del formulario y validarlos, devolviendo respuestas de verificación.

```
# Serializer para el modelo Paciente
class PacienteSerializer(serializers.ModelSerializer):

    class Meta:
        model = Paciente # Modelo asociado al serializer
        fields = "__all__" # Incluye todos los campos del modelo

    def validate_dni(self, value):
        if self.instance:
            # Validación al actualizar
            if (
                Paciente.objects.filter(dni=value)
                .exclude(id_paciente=self.instance.id_paciente)
                .exists()
            ):
                raise serializers.ValidationError(
                    "El DNI ya está registrado en otro paciente."
                )
            else:
                # Validación al crear
                if Paciente.objects.filter(dni=value).exists():
                    raise serializers.ValidationError("El DNI ya está registrado.")
        return value

    def validate_email(self, value):
        if self.instance:
            # Validación al actualizar
            if (
                Paciente.objects.filter(email=value)
                .exclude(id_paciente=self.instance.id_paciente)
                .exists()
            ):
                raise serializers.ValidationError(
                    "El email ya está registrado en otro paciente."
                )
            else:
                # Validación al crear
                if Paciente.objects.filter(email=value).exists():
                    raise serializers.ValidationError("El email ya está registrado.")
        return value
```

- En **models.py** definimos nuestro modelo de base de datos para que se comunique con PostgreSQL.

```
# Modelo para los pacientes
class Paciente(models.Model):
    id_paciente = models.AutoField(primary_key=True) # Identificador único del paciente
    dni = models.CharField(unique=True, max_length=9) # DNI del paciente
    nombre = models.CharField(max_length=100) # Nombre del paciente
    apellido = models.CharField(max_length=100) # Apellido del paciente
    email = models.CharField(unique=True, max_length=100) # Correo único del paciente
    telefono = models.CharField(
        max_length=15, blank=True, null=True
    ) # Teléfono del paciente (opcional)
    contrasena = models.CharField(max_length=100) # Contraseña del paciente

    class Meta:
        managed = False
        db_table = "paciente"
```

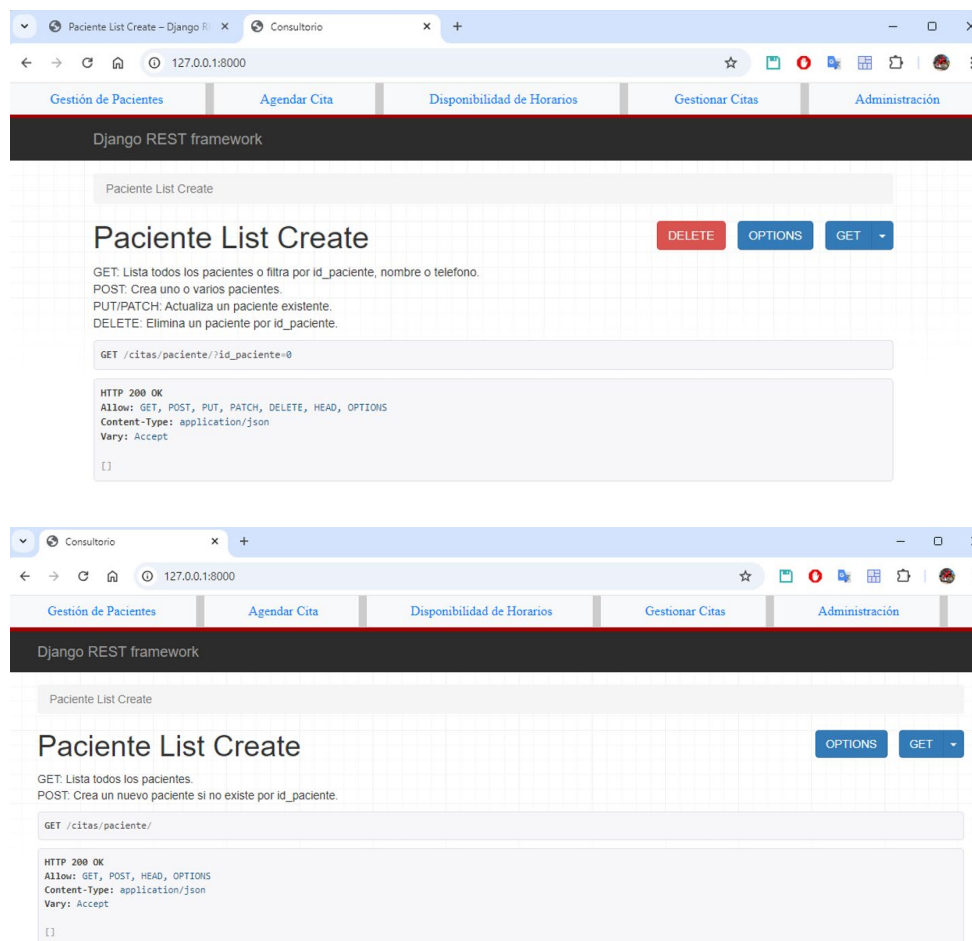
- Muy importante es que los datos de conexión con nuestra base de datos PostgreSQL estén bien configurados para el buen funcionamiento en **settings.py**.

```
# Configuración de la base de datos
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql', # Motor de base de datos PostgreSQL
        'NAME': 'consultorio', # Nombre de la base de datos
        'USER': 'postgres', # Usuario de la base de datos
        'PASSWORD': '1234', # Contraseña del usuario
        'HOST': 'localhost', # Dirección del servidor
        'PORT': '5432', # Puerto del servidor
    }
}
```

- Esto es solo unos ejemplos, todo el código se proporciona en los archivos GitHub que hemos mencionado antes y que también se reflejaron en la bibliografía.

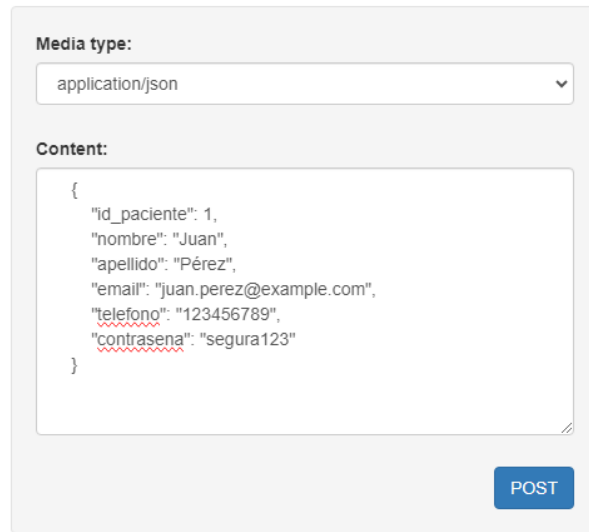
6. Estructurando código y pruebas con Django REST framework

- Para el backend vamos a usar **Django** y una serie de “**endpoints**” cargados en nuestro fichero “**views.py**” como crear, leer, actualizar y eliminar pacientes, agendar citas médicas, consultar disponibilidad de horarios y cancelar o reprogramar citas.
- Esto nos darán feedback durante las pruebas CRUD con **Django REST framework**, para ello hemos diseñado un pequeño menú sencillo “**opciones.html**”, para manejarnos entre las distintas opciones y facilitarnos la tarea durante el proceso, tomando este aspecto amigable.



6.1 Realización de pruebas documentadas

- Como hemos comentado en el apartado anterior para las pruebas usaremos el **Django REST framework**.
- Empezamos las pruebas creando pacientes, cargamos el JSON de ejemplo.



The screenshot shows a REST client interface. At the top, there is a 'Media type' dropdown menu set to 'application/json'. Below it, the 'Content' field contains a JSON object representing a patient:

```
{  "id_paciente": 1,  "nombre": "Juan",  "apellido": "Pérez",  "email": "Juan.perez@example.com",  "telefono": "123456789",  "contrasena": "segura123"}
```

. At the bottom right of the content area is a blue button labeled 'POST'.

- Nos devuelve que ha sido creado correctamente.

POST: Crea un nuevo paciente si no existe por id_paciente.

POST /citas/paciente/

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "id_paciente": 1,
  "nombre": "Juan",
  "apellido": "Pérez",
  "email": "Juan.perez@example.com",
  "telefono": "123456789",
  "contrasena": "segura123"
}
```

- Intentamos crear de nuevo el mismo paciente para ver si lo duplica y comprobar que nuestra condición de que valide, si existe el usuario nos avise.

The screenshot shows a REST client interface. On the left, a 'Media type' dropdown is set to 'application/json'. Below it, a 'Content' text area contains a JSON object for a patient with id 1, name Juan, and other details. A 'POST' button is at the bottom right. On the right, the response is shown: 'POST: Crea un nuevo paciente si no existe por id_paciente.' followed by 'POST /citas/paciente/' and an 'HTTP 400 Bad Request' status. The response headers include 'Allow: GET, POST, HEAD, OPTIONS', 'Content-Type: application/json', and 'Vary: Accept'. The response body is a JSON object with an error message: '{ "error": "El paciente ya existe." }'.

- Cargamos 10 pacientes y realizamos un GET para continuar con las pruebas, comprobamos que los datos han sido introducidos correctamente.

```
{
  "id_paciente": 1,
  "nombre": "Juan",
  "apellido": "Pérez",
  "email": "juan.perez@example.com",
  "telefono": "123456789",
  "contrasena": "segura123"
},
{
  "id_paciente": 2,
  "nombre": "María",
  "apellido": "Gómez",
  "email": "maria.gomez@example.com",
  "telefono": "987654321",
  "contrasena": "password456"
},
{
  "id_paciente": 3,
  "nombre": "Carlos",
  "apellido": "Hernández",
  "email": "carlos.hernandez@example.com",
  "telefono": "555666777",
  "contrasena": "qwerty789"
},
{
  "id_paciente": 4,
  "nombre": "Ana",
  "apellido": "López",
  "email": "ana.lopez@example.com",
  "telefono": "111222333",
  "contrasena": "abc123"
},
{
  "id_paciente": 5,
  "nombre": "Luis",
  "apellido": "Martínez",
  "email": "luis.martinez@example.com",
  "telefono": "444555666",
  "contrasena": "contraseña"
},
{
  "id_paciente": 6,
  "nombre": "Sofía",
  "apellido": "Rodríguez",
  "email": "sofia.rodriguez@example.com",
  "telefono": "777888999",
  "contrasena": "segura456"
},
{
  "id_paciente": 7,
  "nombre": "Miguel",
  "apellido": "García",
  "email": "miguel.garcia@example.com",
  "telefono": "333444555",
  "contrasena": "clave123"
},
{
  "id_paciente": 8,
  "nombre": "Laura",
  "apellido": "Moreno",
  "email": "laura.moreno@example.com",
  "telefono": "888999000",
  "contrasena": "pass456"
},
{
  "id_paciente": 9,
  "nombre": "Pedro",
  "apellido": "Ruiz",
  "email": "pedro.ruiz@example.com",
  "telefono": "666777888",
  "contrasena": "123seguro"
},
{
  "id_paciente": 10,
  "nombre": "Elena",
  "apellido": "Castro",
  "email": "elena.castro@example.com",
  "telefono": "999000111",
  "contrasena": "elena123"
}
```

- Ahora hacemos pruebas para actualizar datos por **PUT** y por **PATCH**.
 - Utilizaremos **PUT** para actualizar todos los campos.
 - Utilizamos **PATCH** para actualizar campos concretos.

Probamos con **PUT** a realizar una actualización de teléfono del id_paciente=1

Media type:
application/json

Content:

```
{
  "id_paciente": 1,
  "nombre": "Juan",
  "apellido": "Pérez",
  "email": "juan.perez@example.com",
  "telefono": "111222333",
  "contrasena": "segura123"
}
```

PUT PATCH

Antes

```
{
  "id_paciente": 1,
  "nombre": "Juan",
  "apellido": "Pérez",
  "email": "juan.perez@example.com",
  "telefono": "123456789",
  "contrasena": "segura123"
},
```

Después

PUT /citas/paciente/?id_paciente=1

HTTP 200 OK
Allow: GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "id_paciente": 1,
  "nombre": "Juan",
  "apellido": "Pérez",
  "email": "juan.perez@example.com",
  "telefono": "111222333",
  "contrasena": "segura123"
}
```

- Por ejemplo con **PUT** nos pide que ingresemos todos los campos aunque no se edite el contenido, de lo contrario nos indica que es un campo requerido.

Media type: application/json

Content:

```
{
  "id_paciente": 1,
  "nombre": "Juan",
  "apellido": "Pérez",
  "email": "juan.perez@example.com",
  "telefono": "111222333",
  "contrasena": "segura123"
}
```

PUT /citas/paciente/?id_paciente=1

HTTP 400 Bad Request
Allow: GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "contrasena": [
    "Este campo es requerido."
  ]
}
```

- Pero si utilizamos **PATCH**, podemos elegir un único campo a modificar.

Media type: application/json

Content:

```
{
  "telefono": "444555666"
}
```

PUT PATCH

Antes

Después

PUT /citas/paciente/?id_paciente=1

```
HTTP 200 OK
Allow: GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

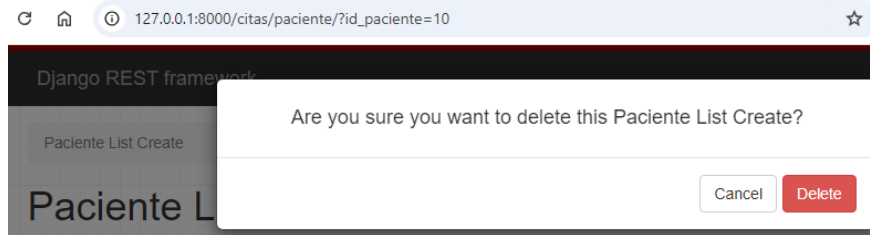
{
  "id_paciente": 1,
  "nombre": "Juan",
  "apellido": "Pérez",
  "email": "juan.perez@example.com",
  "telefono": "111222333",
  "contrasena": "segura123"
}
```

PATCH /citas/paciente/?id_paciente=1

```
HTTP 200 OK
Allow: GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "id_paciente": 1,
  "nombre": "Juan",
  "apellido": "Pérez",
  "email": "juan.perez@example.com",
  "telefono": "444555666",
  "contrasena": "segura123"
}
```

- Para ir finalizando el CRUD de pacientes usaremos **DELETE**.
 - Probamos eliminando el paciente 10, nos pedirá confirmación, la aceptamos.



DELETE: Elimina un paciente por id_paciente.

```
DELETE /citas/paciente/?id_paciente=10
```

```
HTTP 200 OK
Allow: GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "mensaje": "Paciente eliminado exitosamente."
}
```

- Si intentamos eliminar de nuevo el paciente, nos dirá que no lo encuentra, lo mismo pasaría si intenta buscar un id_paciente que no esté registrado.

DELETE: Elimina un paciente por id_paciente.

```
DELETE /citas/paciente/?id_paciente=10
```

```
HTTP 404 Not Found
Allow: GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "error": "Paciente no encontrado."
}
```

- Aplicamos la misma metodología para los siguientes 3 casos de uso, repitiendo las mismas pruebas, pero con los casos específicos para cada uno de ellos, ampliando el código de los endpoints en “**views.py**”.

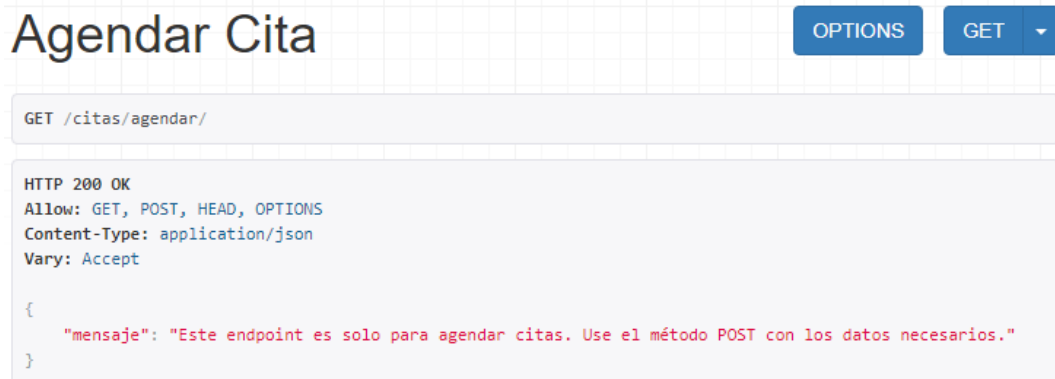
- Desglose de verificaciones usadas en cada caso:

6.1.1 Pacientes

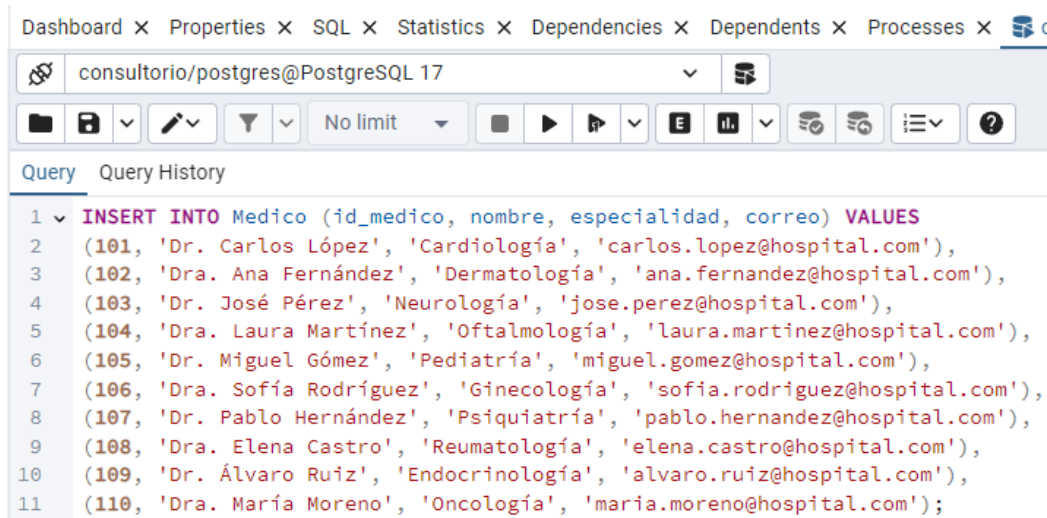
- GET – Muestra todo o filtra la búsqueda por id_paciente, nombre o telefono.
- POST – Comprueba si el paciente ya existe, si no lo crea.
- PUT – Actualiza todos los datos del id_paciente, verifica si falta alguna columna, primero verifica que el id_paciente exista.
- PATCH – Actualiza parcialmente columnas de una tabla buscada por id_paciente, primero verifica que el id_paciente exista.
- DELETE – Verifica primero que el id_paciente exista, si lo encuentra pide consentimiento de eliminación, si aprobamos se elimina.

6.1.2 Agendar cita médica

- En este caso limitamos solo el endpoint para agendar citas, mostramos un mensaje si intentamos leer datos con GET.



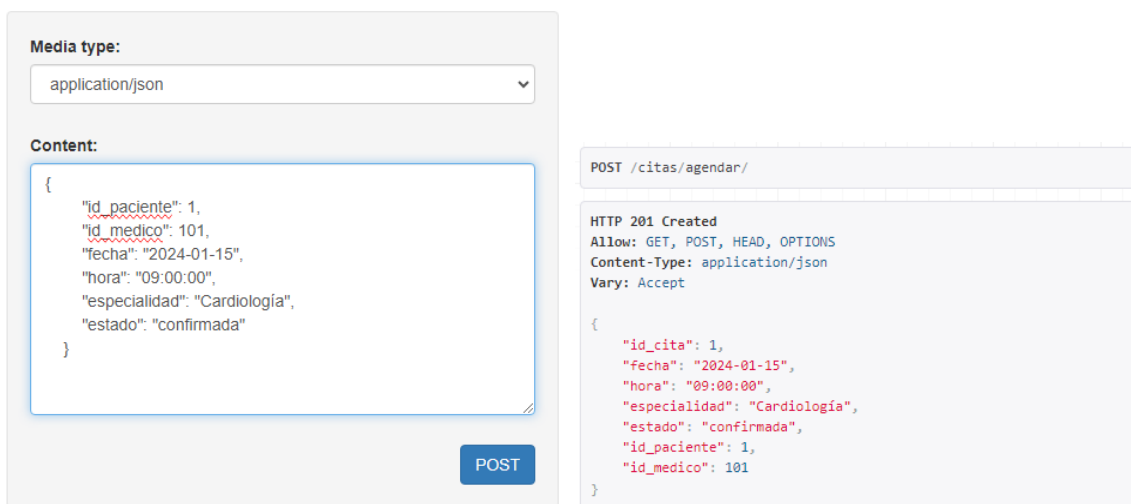
- Como no tenemos médicos dados de alta, de forma rápida le cargo un script SQL al “PgAdmin 4”, para seguir con las pruebas.



The screenshot shows the PgAdmin 4 interface with a query editor. The query is an INSERT statement into the 'Medico' table, adding 10 records with IDs 101 through 110, each with a name, specialty, and email address.

```
1 INSERT INTO Medico (id_medico, nombre, especialidad, correo) VALUES
2 (101, 'Dr. Carlos López', 'Cardiología', 'carlos.lopez@hospital.com'),
3 (102, 'Dra. Ana Fernández', 'Dermatología', 'ana.fernandez@hospital.com'),
4 (103, 'Dr. José Pérez', 'Neurología', 'jose.perez@hospital.com'),
5 (104, 'Dra. Laura Martínez', 'Oftalmología', 'laura.martinez@hospital.com'),
6 (105, 'Dr. Miguel Gómez', 'Pediatría', 'miguel.gomez@hospital.com'),
7 (106, 'Dra. Sofía Rodríguez', 'Ginecología', 'sofia.rodriguez@hospital.com'),
8 (107, 'Dr. Pablo Hernández', 'Psiquiatría', 'pablo.hernandez@hospital.com'),
9 (108, 'Dra. Elena Castro', 'Reumatología', 'elena.castro@hospital.com'),
10 (109, 'Dr. Álvaro Ruiz', 'Endocrinología', 'alvaro.ruiz@hospital.com'),
11 (110, 'Dra. María Moreno', 'Oncología', 'maria.moreno@hospital.com');
```

- POST – Verificamos si existe el id_paciente, el id_medico, fecha y hora, si existe no se asigna una cita, si no se registra una nueva.



The screenshot shows a REST client interface. On the left, a POST request is configured with the URL '/citas/agendar/' and a JSON body containing patient and doctor information. On the right, the response is shown as a 201 Created status with a JSON body containing the assigned appointment ID and details.

Media type: application/json

Content:

```
{
  "id_paciente": 1,
  "id_medico": 101,
  "fecha": "2024-01-15",
  "hora": "09:00:00",
  "especialidad": "Cardiología",
  "estado": "confirmada"
}
```

POST /citas/agendar/

HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "id_cita": 1,
  "fecha": "2024-01-15",
  "hora": "09:00:00",
  "especialidad": "Cardiología",
  "estado": "confirmada",
  "id_paciente": 1,
  "id_medico": 101
}
```


- Verificamos para no duplicar citas y generamos 10 citas por JSON o SQL

Media type:
application/json

Content:

```
{
  "id_paciente": 1,
  "id_medico": 101,
  "fecha": "2024-01-15",
  "hora": "09:00:00",
  "especialidad": "Cardiología",
  "estado": "confirmada"
}
```

POST

POST /citas/agendar/

HTTP 400 Bad Request
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "error": "Ya existe una cita en este horario."
}
```

Query Query History

```
1 1 INSERT INTO cita (id_cita, id_paciente, id_medico, fecha, hora, especialidad, estado)
2 (2, 2, 102, '2024-01-15', '10:00:00', 'Dermatología', 'confirmada'),
3 (3, 3, 103, '2024-01-15', '11:00:00', 'Neurología', 'confirmada'),
4 (4, 4, 104, '2024-01-15', '13:00:00', 'Oftalmología', 'confirmada'),
5 (5, 5, 105, '2024-01-15', '14:00:00', 'Pediatría', 'confirmada'),
6 (6, 6, 106, '2024-01-15', '15:00:00', 'Ginecología', 'confirmada'),
7 (7, 7, 107, '2024-01-16', '09:00:00', 'Psiquiatría', 'confirmada'),
8 (8, 8, 108, '2024-01-16', '10:00:00', 'Reumatología', 'confirmada'),
9 (9, 9, 109, '2024-01-16', '11:00:00', 'Endocrinología', 'confirmada'),
10 (10, 10, 110, '2024-01-16', '13:00:00', 'Oncología', 'confirmada');
```

6.1.3 Consultar disponibilidad de horarios

- Aquí hacemos solo una lectura de datos a modo de consulta con GET, filtramos por especialidad y fecha para ver los horarios disponibles.

Verificamos que existen datos

```
GET / citas/gestionarcita/

HTTP 200 OK
Allow: GET, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
  {
    "id_cita": 1,
    "fecha": "2024-01-15",
    "hora": "09:00:00",
    "especialidad": "Cardiología",
    "estado": "confirmada",
    "id_paciente": 1,
    "id_medico": 101
  },
]
```

Filtramos búsqueda

Disponibilidad Horarios

OPTIONS GET

```
GET / citas/disponibilidad/?especialidad=Cardiolog%C3%ADa&fecha=2024-01-15

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "disponibles": [
    "10:00:00",
    "11:00:00",
    "13:00:00",
    "14:00:00",
    "15:00:00"
  ]
}
```

- Si no tenemos citas disponibles nos marca todos los horarios disponibles para esa especialidad:

```
GET / citas/disponibilidad/?especialidad=Cardiolog%C3%ADa&fecha=2024-01-14

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "disponibles": [
    "09:00:00",
    "10:00:00",
    "11:00:00",
    "13:00:00",
    "14:00:00",
    "15:00:00"
  ]
}
```

6.1.4 Cancelar o reprogramar citas

- Aquí no se permite el método POST, solo GET, PATCH y DELETE, para la lectura de citas, reprogramarla o cancelarlas, no introducir nuevas.
 - Aplicamos un filtro para la búsqueda de citas por id_cita, id_paciente o id_medico.
- Para realizar el patch es imprescindible indicar el id_cita, luego modificamos fecha y hora.

Filtramos por id_medico	Reagendamos la cita	Comprobamos el estado
<pre>GET /citas/gestionarcita/?id_medico=110 HTTP 200 OK Allow: GET, PATCH, DELETE, HEAD, OPTIONS Content-Type: application/json Vary: Accept [{ "id_cita": 10, "fecha": "2024-01-16", "hora": "13:00:00", "especialidad": "Oncología", "estado": "confirmada", "id_paciente": 10, "id_medico": 110 }]</pre>	<pre>PATCH /citas/gestionarcita/?id_medico=110 HTTP 200 OK Allow: GET, PATCH, DELETE, HEAD, OPTIONS Content-Type: application/json Vary: Accept { "mensaje": "Cita reprogramada exitosamente." }</pre> <p>Content:</p> <pre>{ "id_cita": 10, "fecha": "2024-01-16", "hora": "13:00:00", }</pre>	<pre>GET /citas/gestionarcita/?id_medico=110 HTTP 200 OK Allow: GET, PATCH, DELETE, HEAD, OPTIONS Content-Type: application/json Vary: Accept [{ "id_cita": 10, "fecha": "2024-01-20", "hora": "10:00:00", "especialidad": "Oncología", "estado": "confirmada", "id_paciente": 10, "id_medico": 110 }]</pre>

- Usamos DELETE para eliminarla, se solicita el id_cita para su búsqueda.

Cita agendada	Correcta cancelación	Ya no hay cita programada
<pre>GET /citas/gestionarcita/?id_cita=10 HTTP 200 OK Allow: GET, PATCH, DELETE, HEAD, OPTIONS Content-Type: application/json Vary: Accept [{ "id_cita": 10, "fecha": "2024-01-20", "hora": "10:00:00", "especialidad": "Oncología", "estado": "confirmada", "id_paciente": 10, "id_medico": 110 }]</pre>	<pre>DELETE /citas/gestionarcita/?id_cita=10 HTTP 200 OK Allow: GET, PATCH, DELETE, HEAD, OPTIONS Content-Type: application/json Vary: Accept { "mensaje": "Cita cancelada exitosamente." }</pre>	<pre>GET /citas/gestionarcita/?id_cita=10 HTTP 200 OK Allow: GET, PATCH, DELETE, HEAD, OPTIONS Content-Type: application/json Vary: Accept []</pre>

7. Definimos los módulos del frontend con Vue.js.

- Los ficheros que se adjuntan en GitHub son:
 - Del entorno Vue.js en frontend los más significativos.
 - **main.js** (Se importan los assets de forma global y la conexión con django)
 - **index.js** (Se vinculan las vistas de los componentes con el nombre de url)
 - **App.vue** (Se define la ubicación de la cabecera de la página principal)
 - **AppHeader.vue** (Se define la cabecera y las rutas de las vistas)
 - **style-header.css** (Contiene el código con los estilos de la cabecera)
 - **style.css** (Contiene el código con los estilos de las vistas)
 - Listado de los 14 módulos implicados en el proyecto:
 - **PacientesView.vue**
 - **MedicosView.vue**
 - **AgendarView.vue**
 - **TratamientosView.vue**
 - **HistorialClínicoView.vue**
 - **FacturasView.vue**
 - **CitasView.vue**
 - **EstadisticasView.vue**
 - **ProveedoresLaboratorioView.vue**
 - **ProveedoresMedicoView.vue**
 - **SeguridadAuditoriaView.vue**
 - **SeguridadAuditoria2View.vue**
 - **UsuariosView.vue**
 - **LoginView.vue**.

8. Vistas generales de los módulos

8.1 Módulo Pacientes

Conectado con el usuario: admin

Clínica Rehabilita tu alma

Pacientes | Médicos | Agendar |

Gestión de Pacientes

DNI:*
Formato: 8 dígitos + 1 letra

Nombre:*
Mínimo 3 caracteres

Apellido:*
Mínimo 3 caracteres

Email:*
Ingrese un email válido

Teléfono:*
Ingrese un teléfono de 9 dígitos

Contraseña:*
Ingrese un contraseña 6 caracteres o más

☐ Mostrar Contraseña

* Campos requeridos.

Crear Paciente

Filtro: Nombre

Ingrese criterio de búsqueda

ID	DNI	Nombre	Apellido	Email	Teléfono	Acciones
1	63573831V	Juan	Pérez	juan.perez@paciente.com	123456789	<div>Editar</div> <div>Eliminar</div>
2	17417394S	María	Gómez	maria.gomez@paciente.com	987654321	<div>Editar</div> <div>Eliminar</div>
3	16011283C	Carlos	Hernández	carlos.hernandez@paciente.com	555666777	<div>Editar</div> <div>Eliminar</div>
4	67427492I	Ana	López	ana.lopez@paciente.com	111222333	<div>Editar</div> <div>Eliminar</div>

1

2

- Controlamos todos los campos del formulario para validarlos correctamente antes de crear el paciente, mostramos un CRUD completo del paciente en vista del modo de desarrollador.

- Se han indicado los campos que son obligatorios rellenar en el formulario.
 - Se indica con un “*” y además de agrega un pie con “* Campos requeridos”.
 - En el código de Vue.js, en la declaración del input ponemos el valor “required”.

Gestión de Pacientes

DNI:* Formato: 8 dígitos + 1 letra

Nombre:* Mínimo 3 caracteres

Apellido:* Mínimo 3 caracteres

Email:* Ingrese un email válido

Teléfono:* Ingrese un teléfono de 9 dígitos

Contraseña:* Ingrese un contraseña 6 caracteres o más

☐ Mostrar Contraseña

* Campos requeridos.

- Se incluye un gestor de paginación en el listado, se reduce la altura del contenedor de 7 a 5 registros, el modo de funcionamiento lo hemos determinado para:
 - Si tiene menos de 5 registros solo hay una página sin scroll lateral vertical.
 - Más de 5 registros listados se activa el scroll “overflow-y: auto;” en styles.css.
 - Cada 10 registros listados, se genera una nueva página.

ID	DNI	Nombre	Apellido	Email	Teléfono	Acciones
6	44521873E	Sofía	Rodríguez	sofia.rodriguez@paciente.com	777888999	Editar Eliminar
7	33599935C	Miguel	García	miguel.garcia@paciente.com	333444555	Editar Eliminar
8	17321695Z	Laura	Moreno	laura.moreno@paciente.com	888999000	Editar Eliminar
9	34322836E	Pedro	Ruiz	pedro.ruiz@paciente.com	666777999	Editar Eliminar
10	04433870C	Marco	Rodriguez	marco.rodriguez@paciente.com	753951852	Editar Eliminar

1

- Cuando agregamos el registro 11, se genera nueva página

ID	DNI	Nombre	Apellido	Email	Teléfono	Acciones
11	53543655H	Jose	Fernandez	jose.fernandez@paciente.com	605123678	Editar Eliminar

1 2

- El DNI es único para cada paciente, no se puede repetir, nos mostrara una alerta de que ya está registrado.

localhost:8080 dice
Error al guardar el paciente:
dni: Ya existe paciente con este dni.

Aceptar

Gestión de Pacientes

DNI:* 63573831V
Nombre:* Lucas
Apellido:* Gil
Email:* lucas.gil@paciente.com
Teléfono:* 624849445
Contraseña:* *****
☐ Mostrar Contraseña
* Campos requeridos.

Crear Paciente

Filtro: Nombre ▼ Ingrese criterio de búsqueda

ID	DNI	Nombre	Apellido	Email	Tel
1	63573831V	Juan	Pérez	juan.perez@paciente.com	123

- El email es único para cada paciente, no se puede repetir, nos mostrara una alerta de que ya está registrado.

localhost:8080 dice
Error al guardar el paciente:
email: Ya existe paciente con este email.

Aceptar

Gestión de Pacientes

DNI:* 63573365U
Nombre:* Lucas
Apellido:* Gil
Email:* juan.perez@paciente.com
Teléfono:* 624849445
Contraseña:* *****
☐ Mostrar Contraseña
* Campos requeridos.

Crear Paciente

Filtro: Nombre ▼ Ingrese criterio de búsqueda

ID	DNI	Nombre	Apellido	Email	Tel
1	63573831V	Juan	Pérez	juan.perez@paciente.com	123

8.2 Módulo Médicos

Conectado con el usuario: admin

Clínica Rehabilita tu alma

Pacientes | Médicos | Agendar

Gestión de Médicos

Nº Colegiado:*

Nombre:*

Especialidad:*

Email:*

* Campos requeridos.

Filtro: Nombre

ID	Nº Colegiado	Nombre	Especialidad	Email	Acciones
1	TEMP101	Dra. Ana Fernández	Dermatología	ana.fernandez@hospital.com	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
2	TEMP102	Dr. José Pérez	Neurología	jose.perez@hospital.com	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
3	TEMP103	Dra. Laura Martínez	Oftalmología	laura.martinez@hospital.com	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
4	TEMP104	Dr. Miguel Gómez	Pediatría	miguel.gomez@hospital.com	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>

1 2

- Como en el apartado de pacientes, en los médicos especialistas se pone como campos únicos el número de colegiado y el email, el cuál en caso de existir nos arroja un error.
 - Ejemplo agregamos o editamos un nuevo especialista, editamos el registro id=1, con el número de colegiado del ncolegiado=2 y el email=4

localhost:8080 dice

Error al guardar el especialista:
 ncolegiado: Ya existe medico con este ncolegiado.
 email: Ya existe medico con este email.

Gestión de Médicos

Nº Colegiado:*

Nombre:*

Especialidad:*

Email:*

* Campos requeridos.

8.3 Módulo para Agendar Citas Médicas

Conectado con el usuario: admin

Clínica Rehabilita tu alma

Pacientes | Médicos | Agendar

Gestión de Citas Médicas

Paciente:*

Especialidad:*

Fecha:*

Hora:*

Confirmar Cita: ☒

Cita automática: ☐

Referencia Cita: Se generará automáticamente
 * Campos requeridos.

Filtro:

ID Cita	Ref. Cita	Paciente	Especialidad	Médico	Fecha	Hora	Estado	Acciones
1	202501070101	Maria Gómez	Neurología	Dr. José Pérez	12-01-2025	11:00:00	pendiente	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
2	202501070102	Maria Gómez	Pediatría	Dr. Miguel Gómez	15-01-2024	10:00:00	confirmada	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
3	202501070103	Carlos Hernández	Oftalmología	Dra. Laura Martínez	15-01-2024	11:00:00	confirmada	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
4	202501070104	Ana López	Pediatría	Dr. Miguel Gómez	15-01-2024	13:00:00	confirmada	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>

1 2 3

- Para esta demostración se ha cargado la lista de pacientes activos de la tabla paciente.

Gestión de Citas Médicas

Paciente:*

Especialidad:*

Fecha:*

Hora:*

Confirmar Cita: ☒

Cita automática: ☐

Referencia Cita: Se generará automáticamente
 * Campos requeridos.

Filtro:

ID Cita	Ref. Cita	Paciente	Especialidad	Médico	Fecha	Hora	Estado	Acciones
1	202501070101	Maria Gómez	Neurología	Dr. José Pérez	12-01-2025	11:00:00	pendiente	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
2	202501070102	Maria Gómez	Pediatría	Dr. Miguel Gómez	15-01-2024	10:00:00	confirmada	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
3	202501070103	Carlos Hernández	Oftalmología	Dra. Laura Martínez	15-01-2024	11:00:00	confirmada	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
4	202501070104	Ana López	Pediatría	Dr. Miguel Gómez	15-01-2024	13:00:00	confirmada	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>

1 2 3

- Y un listado de especialistas activos de la tabla medico.

Gestión de Citas Médicas

Paciente:*

Especialidad:*

Fecha:*

Hora:*

Confirmar Cita: ☒

Cita automática: ☐

Referencia Cita: Se generará automáticamente
 * Campos requeridos.

Filtro:

ID Cita	Ref. Cita	Paciente	Especialidad	Médico	Fecha	Hora	Estado	Acciones
1	202501070101	Maria Gómez	Neurología	Dr. José Pérez	12-01-2025	11:00:00	pendiente	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
2	202501070102	Maria Gómez	Pediatría	Dr. Miguel Gómez	15-01-2024	10:00:00	confirmada	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
3	202501070103	Carlos Hernández	Oftalmología	Dra. Laura Martínez	15-01-2024	11:00:00	confirmada	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
4	202501070104	Ana López	Pediatría	Dr. Miguel Gómez	15-01-2024	13:00:00	confirmada	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>

1 2 3

- Pero en un uso real se podría escribir por ejemplo el DNI del paciente y saldría su nombre completo.
- También se podría elegir la especialidad y que salga un listado de los médicos disponibles y asignarles uno, el que más convenga.

DNI Paciente:*	63573831V	▼
Nombre Paciente:	Juan Baeza	
Especialidad:*	Dermatología	▼
Nombre Médico:	Dra. Ana Fernández	
Fecha:*	15 / 01 / 2024	📅
Hora Disponible:*	09 : 00	🕒
Confirmar Cita:	<input checked="" type="checkbox"/>	

- Se ha estado trabajando en varios modelos, se podría depurar o elegir uno de ambos formatos o personalizar el que el cliente nos pidiese.
- Se agrega un calendario completo para elegir la fecha de citación.

The image displays two different calendar UI designs. The left design is a compact month view for 'enero de 2025', showing the first few letters of the months (ene., feb., mar., abr., may., jun., jul., ago., sep., oct., nov., dic.) and a list of years from 2025 to 2029. The right design is a more traditional full-month calendar grid for 'enero de 2025', showing days of the week (L, M, X, J, V, S, D) and dates from 1 to 31, with the 14th highlighted. Both designs include a date input field at the top and navigation icons.

- Tenemos un check en el ejemplo lo marcamos a mano, en un uso real se activaría forma automática, cuando se le enviase al paciente un SMS y confirmase la asistencia.
- El módulo de citas automáticas no se programa por falta de tiempo, lleva mucho código detrás, validar días de calendario disponibles, control de festivos, franja horaria disponible de apertura diaria, disponibilidad por especialidad y médico, etc.
- La Referencia de Cita se genera directamente en el Backend con un código que nos proporcionara un código exclusivo no repetible mediante validación, la primera vez que se asigne una cita se genera.

Confirmar Cita: ☒

Cita automática: ☐

Referencia Cita: Se generará automáticamente

- Las siguientes veces si queremos reagendar la cita podemos modificar fecha, hora y especialista, pero el código no es modificable.

Confirmar Cita: ☒

Cita automática: ☐

Referencia Cita: A6B465FA06F

8.4 Módulo para vista de Citas del Paciente

Clínica Rehabilita tu alma

Conectado con el usuario: Miguel López

Citas | Logout

Consulta de Citas

Detalles de la Cita

ID Cita:	18
Referencia Cita:	A085C531DAE
DNI:	64564576D
Paciente:	Roberto Gómez
Especialidad:	Dermatología
Médico:	Dra. Ana Fernández
Fecha:	14-01-2025
Hora:	15:00
Estado:	confirmada

Detalles de la Cita

ID Cita:	20
Referencia Cita:	EFCC60374AE
DNI:	64564576D
Paciente:	Roberto Gómez
Especialidad:	Psiquiatría
Médico:	Dr. Moises Sevilla
Fecha:	13-01-2025
Hora:	11:00
Estado:	confirmada

- Vista general de un paciente tras hacer login con el rol de paciente.
- El rol Paciente puede realizar consultas de las citaciones que tiene pendiente de realizar, en nuestro sistema tendremos un historial de citaciones de cada paciente, depende del cliente final que el paciente pueda tener acceso a más controles o vistas.

8.5 Módulo Tratamientos

Clínica Rehabilita tu alma

Pacientes | Médicos | Agendar | Tratamientos | Historial | Derivaciones

Gestión de Tratamientos

Paciente: Juan Pérez

Médico: Dr. José Pérez (Neurología)

Descripción: Tratamiento para diabetes tipo 2 con insulina

Fecha de Inicio: 01/01/2025

Fecha de Fin: 30/06/2025

Costo: 0,00

Actualizar Tratamiento

Cancelar

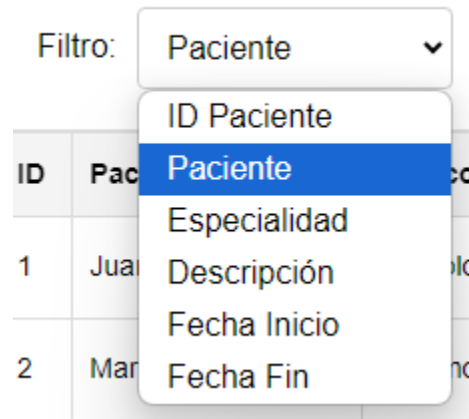
Filtro: Paciente

Ingrese criterio de búsqueda

ID	Paciente	Médico	Descripción	Fecha Inicio	Fecha Fin	Costo	Acciones
1	Juan Pérez	Neurología	Tratamiento para diabetes tipo 2 con insulina	01-01-2025	30-06-2025	0.00 €	<div>Editar</div> <div>Eliminar</div>
2	María Gómez	Oftalmología	Rehabilitación física post cirugía de rodilla	10-02-2025	10-05-2025	0.00 €	<div>Editar</div> <div>Eliminar</div>
3	Carlos Hernández	Pediatría	Terapia psicológica cognitivo-conductual	15-01-2025	Baja indefinida	0.00 €	<div>Editar</div> <div>Eliminar</div>
4	Ana López	Reumatología	Tratamiento con antibióticos para infección respiratoria	20-01-2025	10-02-2025	0.00 €	<div>Editar</div> <div>Eliminar</div>

- Encontramos el nombre de paciente, médico, descripción, fecha de inicio, fecha fin y costo.
 - Este módulo está reservado para los especialistas, tienen un CRUD completo donde poder gestionar de forma completa los tratamientos a los distintos pacientes.
 - Al igual que en el resto de módulos pueden ordenar de menor a mayor y viceversa por ID, Paciente, Médico, Descripción, Fecha Inicio, Fecha Fin, Costo.

- También podemos filtrar la búsqueda por:



- En un uso real se buscaría el paciente por DNI, en esta demostración recogemos los pacientes anidados de la tabla pacientes y los mostramos en el módulo de tratamientos.

Gestión de Tratamientos

Paciente:

Médico:

Descripción:

Fecha de Inicio:

Fecha de Fin:

Costo:

- Replicamos el comportamiento, pero ahora con la tabla de médico anidada, recogemos el nombre y la especialidad del médico y la mostramos, en un uso real se ajustaría a petición del cliente.

Gestión de Tratamientos

Paciente:

Médico:

Descripción:

Fecha de Inicio:

Fecha de Fin:

Costo:

Actualizar Tratamiento

Filtro:

Búsqueda

- También se respetan los calendarios globales.

Fecha de Fin:

Costo:

Actualizar Tratamiento

Filtro:

Búsqueda

ID Paciente

1 Juan Pérez

Fecha de Fin:

Costo:

Actualizar Tratamiento

Filtro:

Búsqueda

ID Paciente

1 Juan Pérez

8.6 Módulo Gestión del Historial Clínico

Clínica Rehabilita tu alma

[Pacientes](#) | [Médicos](#) | [Agendar](#) | [Tratamientos](#) | [Historial](#) | [Derivaciones](#)

Gestión del Historial Clínico

Editar Registro Clínico

Paciente:

Descripción:

Observaciones:

Fecha de Apertura:

Filtro:

ID	Paciente	Descripción	Observaciones	Fecha de Apertura	Acciones
1	Juan Pérez	Hipertensión arterial diagnosticada.	Controlar presión arterial cada semana.	15-01-2025	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
2	María Gómez	Fractura de tibia y peroné.	Sugerida fisioterapia tras retirar el yeso.	10-02-2025	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
3	Carlos Hernández	Diabetes tipo 2.	Recomendado ajuste en dieta y ejercicio regular.	20-03-2025	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
4	Ana López	Gripe estacional.	Tratamiento con antigripales y descanso.	05-04-2025	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>

- En este módulo observamos que conserva la estética acorde con el resto de módulos, se muestra el ID, Paciente, Descripción, Observaciones, Fecha de Apertura.
- En este módulo se incluye campos de texto con autoscroll vertical y redimensionables, por si quisiéramos ampliar la vista del contenido para leer el comentario de una vez.

Observaciones:

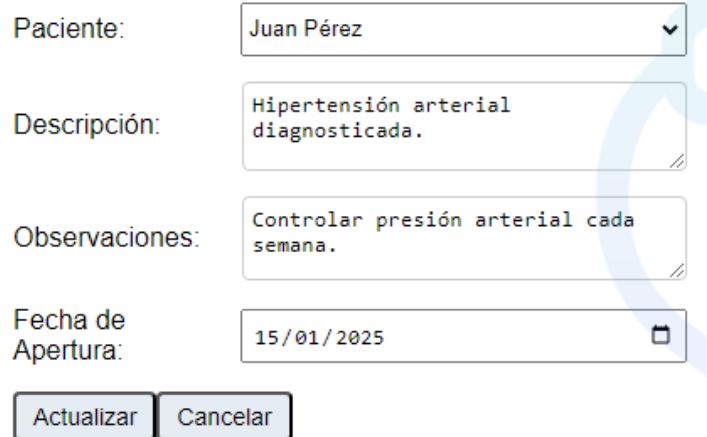
Recomendado ajuste en dieta y ejercicio regular.

Observaciones:

Recomendado ajuste en dieta y ejercicio regular.

- Se incluye una nueva funcionalidad que se escalará al resto de módulos en la entrega al cliente, es un texto indicativo arriba del formulario para saber si estamos en modo creación o modo edición.

Editar Registro Clínico



Paciente: Juan Pérez

Descripción: Hipertensión arterial diagnosticada.

Observaciones: Controlar presión arterial cada semana.

Fecha de Apertura: 15/01/2025

Actualizar Cancelar

- Ahora el formulario de Nuevos registros también lleva el botón cancelar para vaciar todos los campos, en caso de no querer hace un nuevo registro.

Nuevo Registro Clínico



Paciente:

Descripción:

Observaciones:

Fecha de Apertura: dd/mm/aaaa

Registrar Cancelar

8.7 Módulo Derivaciones

Clínica Rehabilita tu alma



Pacientes | Médicos | Agendar | Tratamientos | Historial | Derivaciones

Gestión de Derivaciones

Editar Derivación

Paciente:

Médico Remitente:

Médico Destino:

Motivo:

Fecha de Derivación:

Actualizar

Cancelar

Filtro:

ID	Paciente	Médico Remitente	Médico Destino	Motivo	Fecha de Derivación	Acciones
1	Juan Pérez	Dra. Ana Fernández (Dermatología)	Dr. José Pérez (Neurología)	Paciente requiere evaluación por especialidad de cardiología debido a arritmias frecuentes.	20-01-2025	<div>Editar</div> <div>Eliminar</div>
2	María Gómez	Dr. José Pérez (Neurología)	Dra. Laura Martínez (Oftalmología)	Dolores articulares persistentes, se deriva a reumatología para estudio.	05-01-2025	<div>Editar</div> <div>Eliminar</div>

- En ese módulo presentamos el ID, Paciente, Médico Remitente, Médico Destino, Motivo y Fecha de Derivación.
- Este módulo adopta todas las funcionalidades del módulo de historia clínica, aquí se ha implementado el CRUD completo sin verificar para hacer una demostración el video de presentación.

- Los estados de la factura seleccionables pueden ser a modo demostración, Pendiente, Pagada o Cancelada.

Estado:

- Pendiente
- Pagada
- Cancelada

- Se adjunta ejemplo (Personalizar en la entrega final según petición):

Clínica Rehabilita tu alma



Factura Detallada

Referencia Factura: FACT000001

Referencia Cita: undefined

Paciente: undefined

Fecha Emisión: 17-01-2025

Fecha Cobro: 20-01-2025

Estado: Pagada

Concepto	Precio sin IVA	IVA (21%)	Total con IVA
Consulta médica general	0.00 €	0.00 €	0.00 €

Total sin IVA: 0.00 €

IVA (21%): 0.00 €

Total con IVA: 0.00 €

8.9 Módulo Usuarios/Roles

Conectado con el usuario: admin

Clínica Rehabilita tu alma

Pacientes | Médicos | Agendar | Tratamientos | Historial | Derivaciones | Facturas | Usuarios/Roles | Logout

Gestión de Usuarios

Editar Usuario

Nombre:

Email:

Contraseña:*

Rol:

Estado:

ID	Nombre	Email	Rol	Estado	Acciones
1	Juan Pérez	juan.perez@usuario.com	Paciente	Activo	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
2	María López	maria.lopez@usuario.com	Recepcionista	Activo	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
3	Carlos Gómez	carlos.gomez@usuario.com	Médico	Activo	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
4	Laura Martínez	laura.martinez@usuario.com	Admin	Activo	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>

- Aquí tenemos el id_usuario, Nombre de usuario, email, contraseña encriptada y enmascarada, el ROL a definir entre, Admin, Médico, Paciente, Recepcionista o Usuario.

Rol:

Estado:

- También podemos activar o desactivar usuarios según su estado, para no permitir el uso en nuestro sistema de forma temporal o definitiva.

Estado:

8.10 Módulo de Login

The screenshot shows the login interface of a web application. At the top, there is a green header bar with the text 'Clínica Rehabilita tu alma' and a heart icon. Below this, the main content area has a light blue background with a large, faint watermark of a heart and a key. In the center, there is a white box titled 'Iniciar Sesión'. Inside this box, there are two input fields: 'Email:' with the value 'admin@hospital.com' and 'Contraseña:' with a masked password '.....'. Below the password field is a blue button labeled 'Entrar'.

- La vista de administrador, será con todos los accesos posibles, como hemos ido viendo a lo largo de los sprints anteriores y actual, pero por cada rol o permiso definido se accederá a determinadas áreas predefinidas de nuestro sistema.
- Vista general de un paciente tras hacer login con el rol de paciente.

The screenshot shows the 'Consulta de Citas' (Appointment Consultation) page. At the top, there is a green header bar with the text 'Clínica Rehabilita tu alma' and a heart icon. Below this, there is a blue navigation bar with the text 'Citas | Logout'. The main content area has a light blue background with a large, faint watermark of a heart and a key. In the center, there is a white box titled 'Consulta de Citas'. Inside this box, there are two side-by-side panels, each titled 'Detalles de la Cita'. Each panel contains a list of appointment details.

Detalles de la Cita	
ID Cita:	18
Referencia Cita:	A085C531DAE
DNI:	64564576D
Paciente:	Roberto Gómez
Especialidad:	Dermatología
Médico:	Dra. Ana Fernández
Fecha:	14-01-2025
Hora:	15:00
Estado:	confirmada

Detalles de la Cita	
ID Cita:	20
Referencia Cita:	EFCC60374AE
DNI:	64564576D
Paciente:	Roberto Gómez
Especialidad:	Psiquiatría
Médico:	Dr. Moises Sevilla
Fecha:	13-01-2025
Hora:	11:00
Estado:	confirmada

- El rol Paciente puede realizar consultas de las citas que tiene pendiente de realizar, en nuestro sistema tendremos un historial de citas de cada paciente, depende del cliente final que el paciente pueda tener acceso a más controles o vistas.
- Vista general de un paciente tras hacer login con el rol de Médico especialista.

Conectado con el usuario: Dra. Sara Gómez

Clínica Rehabilita tu alma

Agendar | Tratamientos | Derivaciones | Historial | Logout

Gestión del Historial Clínico

Nuevo Registro Clínico
Paciente:
Descripción:
Observaciones:
Fecha de Apertura:

Filtro:

ID	Paciente	Descripción	Observaciones	Fecha de Apertura	Acciones
1	Juan Pérez	Hipertensión arterial diagnosticada. Revisar semanalmente.	Controlar presión arterial cada semana.	24-01-2025	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>

- Tiene acceso a Agendar, Tratamientos, Derivaciones, Historial Clínico y cualquier módulo que se quisiera definir en un futuro que se pudiese asignar a los especialistas.

- Vista general de un usuario trabajador de la clínica perteneciente a un área específica con un rol distinto, por ejemplo, tendríamos esta vista de recepcionista:

Clínica Rehabilita tu alma



Conectado con el usuario: Recepcionista - Laura Sanz

Pacientes | Médicos | Facturas | Historial | Logout

Gestión de Facturas

Editar Factura

Cita:

Tratamiento:

Concepto:

Fecha de Emisión:

Fecha de Cobro:

Estado:

Actualizar

Cancelar

Filtro: Ingrese criterio de búsqueda

ID	Ref. Factura	Cita	Costo	Concepto	Fecha Emisión	Fecha Cobro	Estado	Acciones
1	FACT000001	Ref: 202501070101 - Paciente: N/A	€0.00	Consulta médica general	17-01-2025	20-01-2025	Pagada	<div>PDF</div> <div>Editar</div> <div>Eliminar</div>

- Las vistas de los roles son solo ejemplos, no tiene por qué tener acceso a esos módulos, o poder tener vista a otros distintos, todo esto lo pactaríamos con el cliente final.

8.11 Módulo Seguridad y Auditoría

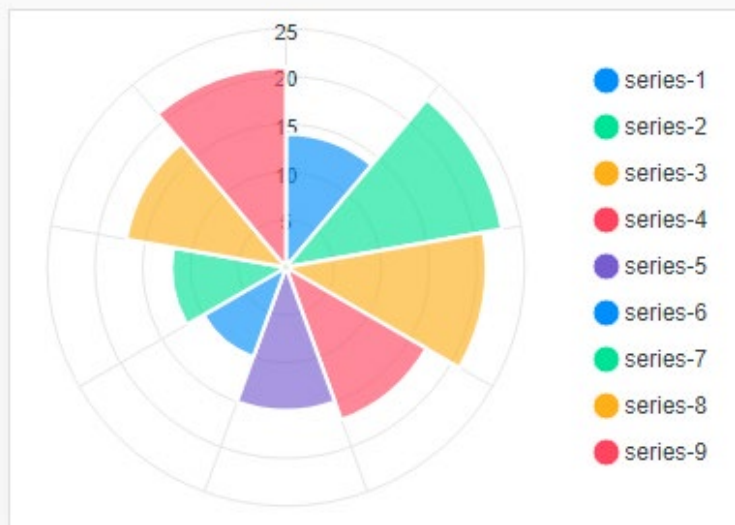


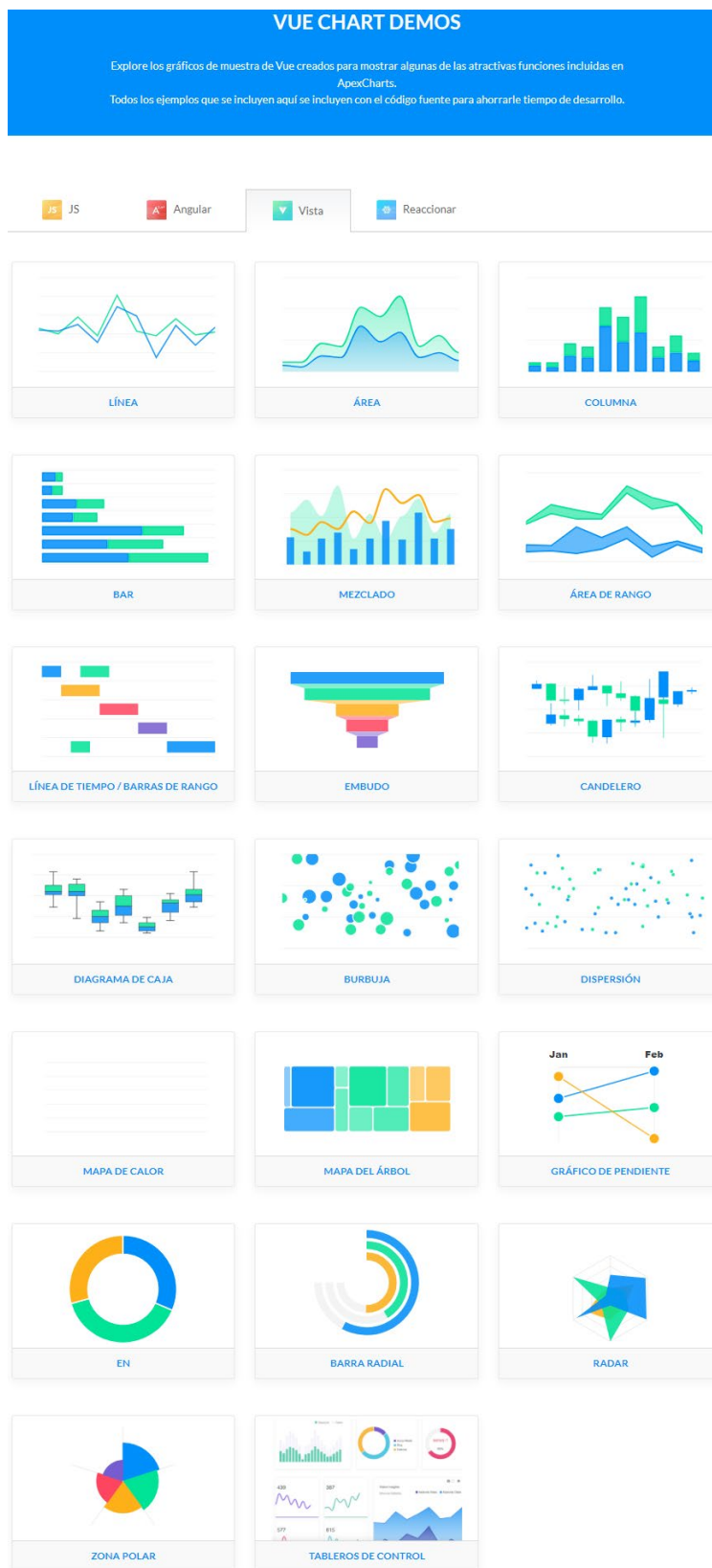
- A modo de ejemplo podríamos graficar:
 - Usuarios Activos vs Inactivos.
 - Tratamientos por Mes.
 - Derivaciones Mensuales.
 - Estado de Facturas.
 - Historial Clínico Mensual.
 - Proveedores Activos vs Inactivos.
 - Una pequeña tabla donde resumir los detalles de auditoría, donde también se podría exportar a PDF con la herramienta adquirida del Sprint 5 anterior.

- Como novedad, en este módulo se ha importado para nuestro frontend en VUE ApexCharts, que lo introducimos en nuestra vista con el código **import VueApexCharts from "vue3-apexcharts"**.
- Se recopila información de cómo manejar las diferentes vistas que nos ofrece y se crean dos vistas a modo de ejemplo gráfico de lo que vería el administrador o la persona encargada de gestionar el módulo, Auditoria y Auditoria2 se le presentaría una serie de opciones al cliente para que elija al gusto. Un ejemplo de implementación que nos ofrece ApexCharts y posibles módulos a elegir.

```
new Vue({
  el: '#app',
  components: {
    apexchart: VueApexCharts,
  },
  data: {
    series: [14, 23, 21, 17, 15, 10, 12, 17, 21],
    chartOptions: {
      chart: {
        type: 'polarArea',
      },
      stroke: {
        colors: ['#fff']
      },
      fill: {
        opacity: 0.8
      }
    }
  }
})
```

```
<div id="chart">
  <apexchart type="polarArea" :options="chartOptions" :series="series"></apexchart>
</div>
```

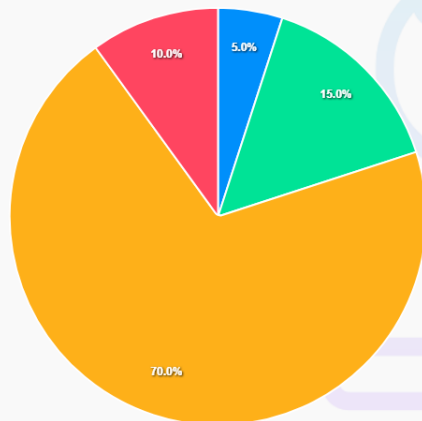




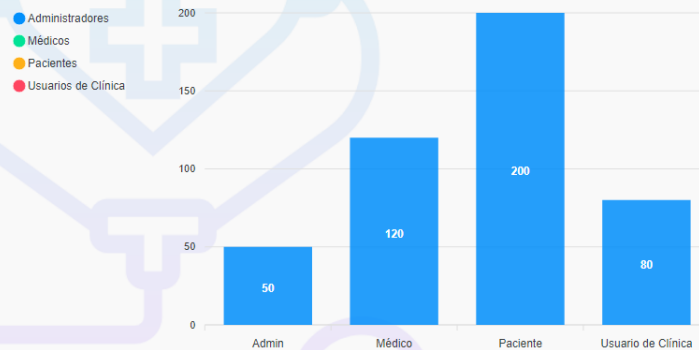
8.12 Módulo Seguridad y Auditoría 2

Seguridad y Auditoría

Distribución de Roles



Actividades por Usuario



Registros de Auditoría

ID	Usuario	Acción	Fecha	Estado
----	---------	--------	-------	--------

- A modo de ejemplo, en la de tipo quesito podríamos graficar:
 - Distribuciones de Roles, donde haríamos la siguiente lectura:
 - Administradores 5%.
 - Usuarios de Clínica 10%.
 - Médicos especialistas 15%.
 - Pacientes 70%.
- En la gráfica de barras verticales estaríamos viendo la actividad por usuario, donde quedaría reflejada la interacción con el sistema por cada rol tomado.
- Luego veríamos una pequeña tabla/contenedor donde recopilar los datos más relevantes, donde también podríamos exportarlo a un PDF.

8.13 Módulo Proveedores de Material de Laboratorio

Conectado con el usuario: admin

Clínica Rehabilita tu alma

Pacientes | Médicos | Agendar | Citas | Tratamientos | Derivaciones | Historial | Facturas | Material de Laboratorio | Material Médico | Auditoria | Auditoria2 | Usuarios/Roles | Logout

Proveedores de Material de Laboratorio

Nombre:

Razón Social:

Dirección:

Teléfono:

Email:

Persona de Contacto:

Puesto de Contacto:

Horario de Atención:

Método de Pago:

Moneda:

Plazo de Pago (días):

Categorías de Producto:

Tiempo de Entrega (días):

Zona de Cobertura:

Calidad del Proveedor:

Fecha de Alta:

Estado:

Notas:

ID	Nombre	Razón Social	Teléfono	Email	Persona de Contacto	Estado	Acciones
----	--------	--------------	----------	-------	---------------------	--------	----------

- Definimos la tabla para nuestra base de datos, nos vemos abrumados por la cantidad de datos que deberíamos manejar, por falta de tiempo no se implementa.

```
-- Tabla de Proveedores de Laboratorio
CREATE TABLE proveedores_laboratorio (
  id_proveedor SERIAL PRIMARY KEY,
  nombre VARCHAR(150) NOT NULL,
  razon_social VARCHAR(150),
  direccion TEXT,
  telefono VARCHAR(15),
  email VARCHAR(100),
  persona_contacto VARCHAR(100),
  puesto_contacto VARCHAR(100),
  horario_atencion VARCHAR(50),
  metodo_pago VARCHAR(50),
  moneda VARCHAR(10),
  plazo_pago INT,
  categorias_producto TEXT,
  tiempo_entrega INT,
  zona_cobertura TEXT,
  calidad_proveedor VARCHAR(10),
  fecha_alta DATE DEFAULT CURRENT_DATE,
  estado VARCHAR(20) DEFAULT 'Activo',
  notas TEXT
);
```

8.14 Módulo Proveedores de Material Médico

Conectado con el usuario: admin

Clínica Rehabilita tu alma

Pacientes | Médicos | Agendar | Citas | Tratamientos | Derivaciones | Historial | Facturas | Material de Laboratorio | Material Médico | Auditoria | Auditoria2 | Usuarios/Roles | Logout

Proveedores de Material Médico

Nombre:

Razón Social:

Dirección:

Teléfono:

Email:

Persona de Contacto:

Puesto de Contacto:

Horario de Atención:

Método de Pago:

Moneda:

Plazo de Pago (días):

Categorías de Producto:

Tiempo de Entrega (días):

Zona de Cobertura:

Calidad del Proveedor:

Fecha de Alta:

Estado:

Notas:

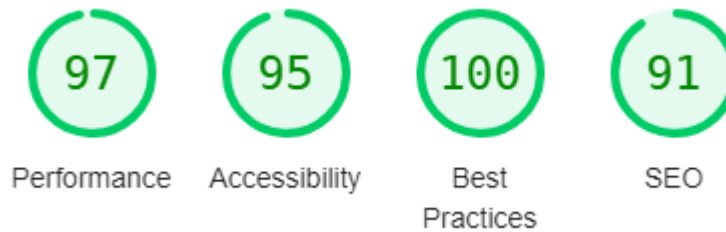
ID	Nombre	Razón Social	Teléfono	Email	Estado	Acciones
----	--------	--------------	----------	-------	--------	----------

- Definimos la tabla para nuestra base de datos, tras mucho deliberar, sería bueno unificar en un solo módulo de proveedores donde realizar los pedidos y dejando un listado con opciones donde elegir si se trata de laboratorio, medico, varios o alguno más que podamos añadir en un futuro. Esto nos ahorraría recursos de implementación de diseño y un filtrado por tipo de departamento de compras.

```
-- Tabla de Proveedores de Material Médico
CREATE TABLE proveedores_material_medico (
  id_proveedor SERIAL PRIMARY KEY,
  nombre VARCHAR(150) NOT NULL,
  razon_social VARCHAR(150),
  direccion TEXT,
  telefono VARCHAR(15),
  email VARCHAR(100),
  persona_contacto VARCHAR(100),
  puesto_contacto VARCHAR(100),
  horario_atencion VARCHAR(50),
  metodo_pago VARCHAR(50),
  moneda VARCHAR(10),
  plazo_pago INT,
  categorias_producto TEXT,
  tiempo_entrega INT,
  zona_cobertura TEXT,
  calidad_proveedor VARCHAR(10),
  fecha_alta DATE DEFAULT CURRENT_DATE,
  estado VARCHAR(20) DEFAULT 'Activo',
  notas TEXT
);
```

9. Análisis técnico

9.1 Análisis de performance con Google Chrome LightHouse



- **Performance (97):**
 - Indica que el rendimiento de la aplicación es excelente, con tiempos de carga rápidos y optimizaciones adecuadas para una experiencia fluida.
- **Accessibility (95):**
 - Refleja que la aplicación está altamente accesible para la mayoría de los usuarios, incluyendo aquellos con discapacidades. Sin embargo, existe una pequeña oportunidad de mejora en algunos aspectos.
- **Best Practices (100):**
 - Demuestra que la aplicación sigue todas las mejores prácticas recomendadas en términos de seguridad, accesibilidad, y desarrollo web, sin ningún problema detectado.
- **SEO (91):**
 - Indica que la aplicación está bien optimizada para los motores de búsqueda, aunque hay espacio para realizar ajustes menores que podrían mejorar su visibilidad y clasificación.

En el apartado de **Performance**, la descripción de la puntuación obtenida es:

METRICS		Expand view
● First Contentful Paint	0.2 s	■ Largest Contentful Paint
● Total Blocking Time	0 ms	● Cumulative Layout Shift
● Speed Index	0.6 s	0

1. **First Contentful Paint (FCP):**

- Mide el tiempo que tarda en aparecer el primer contenido visible (texto o imagen) en la página.

2. **Largest Contentful Paint (LCP):**

- Evalúa el tiempo necesario para que el contenido más grande visible (imagen o bloque de texto) se renderice por completo.

3. **Total Blocking Time (TBT):**

- Suma de los periodos en que la página queda bloqueada (más de 50 ms), afectando la interactividad.

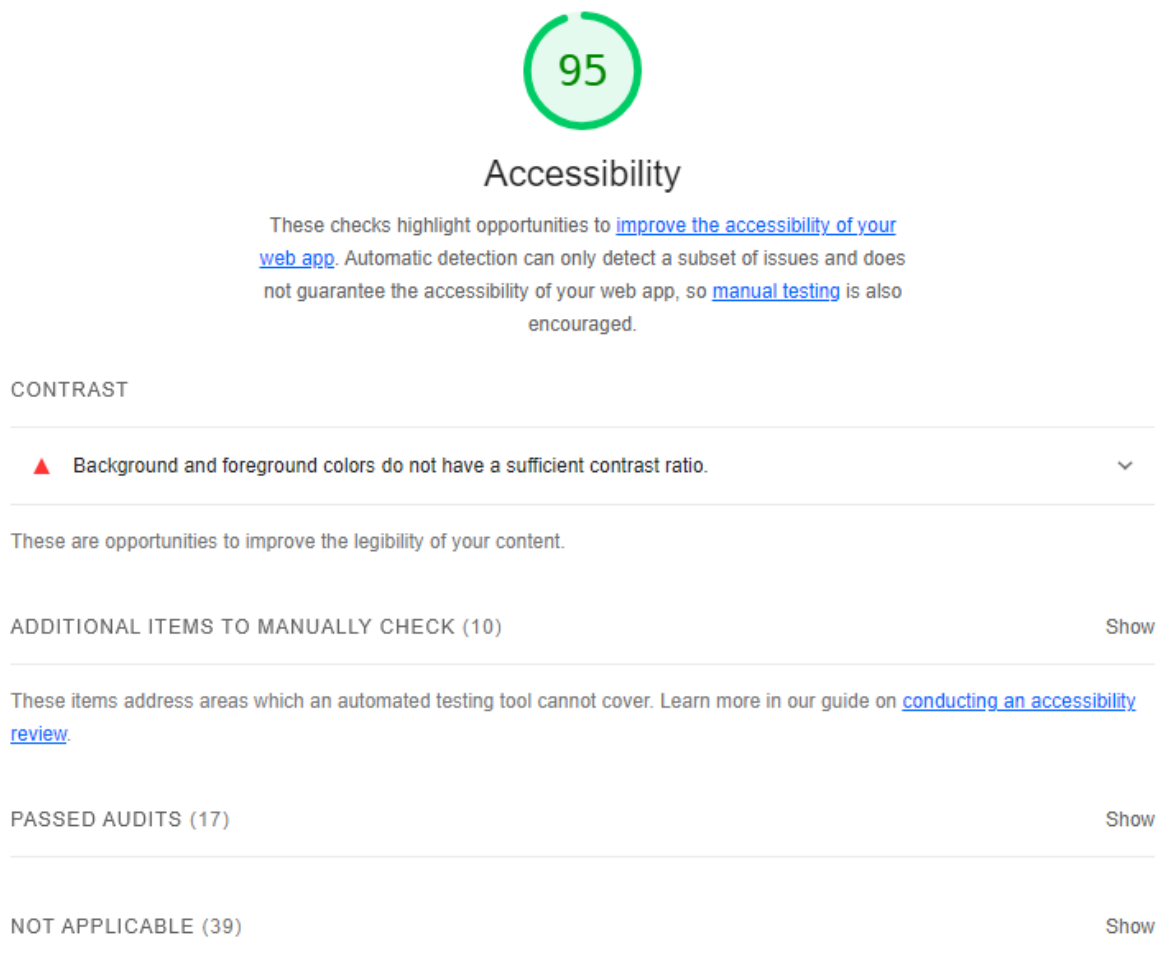
4. **Cumulative Layout Shift (CLS):**

- Mide la estabilidad visual de la página al cargar, evaluando cuánto se mueven los elementos en la pantalla.

5. **Speed Index:**

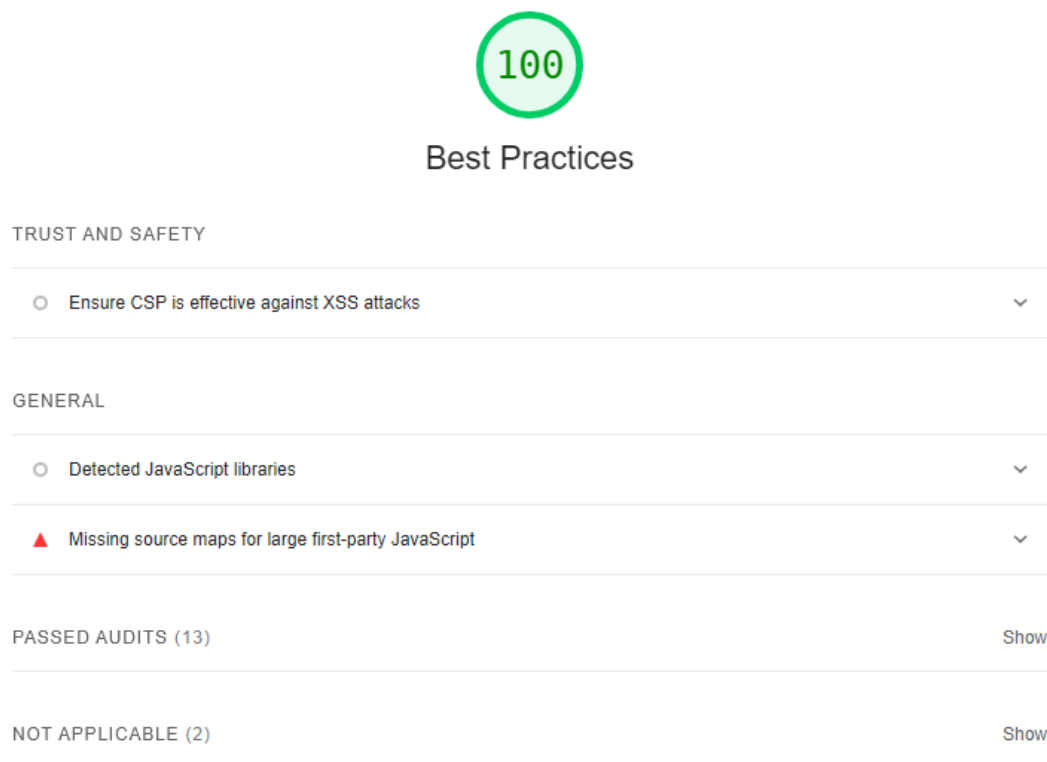
- Indica la rapidez con la que el contenido de la página es visualmente completo.

En el apartado de **Accesibility**, la descripción de la puntuación obtenida es:



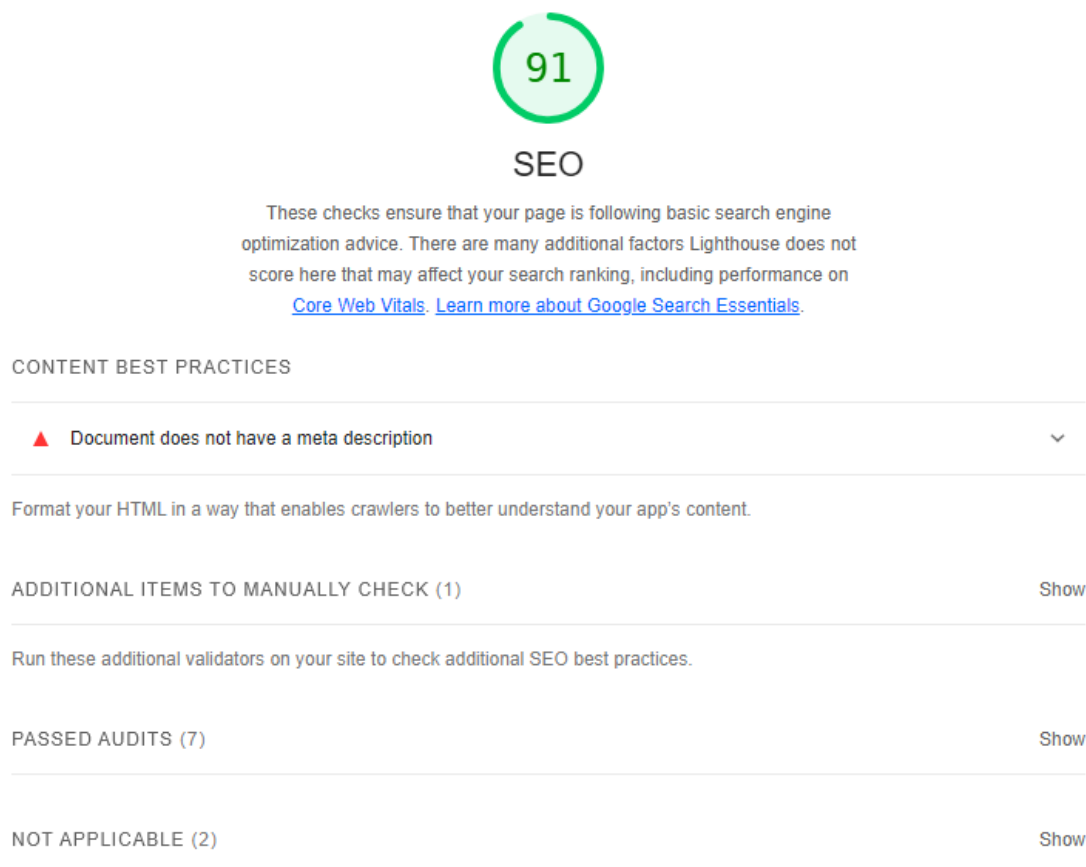
- El valor que nos muestra en Accesibility es bastante bueno, pero con un pequeño margen de mejora, nos indica que la mayoría de los elementos de accesibilidad cumplen con los estándares establecidos, pero que los contrastes de color de fondo y texto podrían no ser suficientes para aquellos con discapacidades visuales.

En el apartado de **Best Practices**, la descripción de la puntuación obtenida es:



- El análisis muestra una puntuación de 100, indicando un cumplimiento sólido en aspectos clave, pero se detectó un área de mejora relacionada con la ausencia de mapas de origen para grandes scripts JavaScript de primera parte, lo que podría dificultar la depuración. La auditoría también nos sugiere revisar la configuración de CSP para proteger contra ataques XSS y verificar las bibliotecas JavaScript utilizadas.

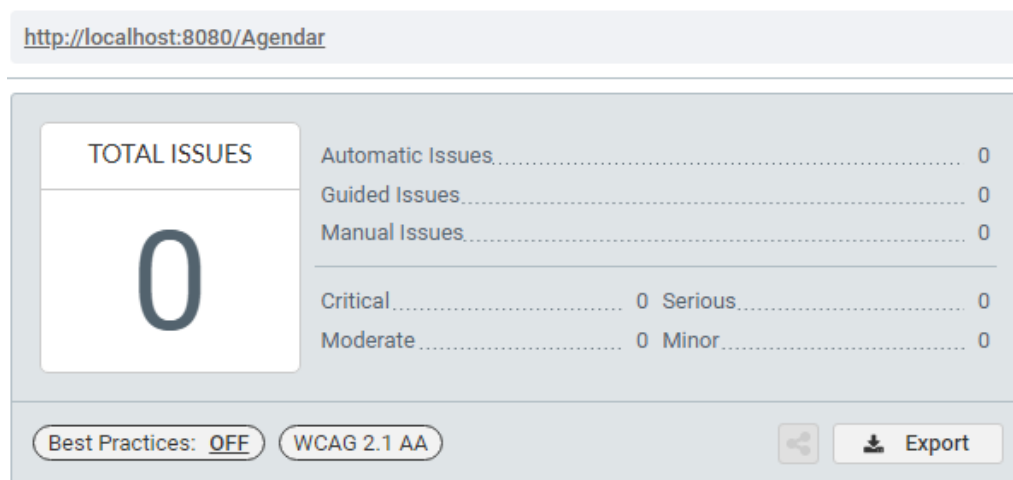
En el apartado de **Best Practices**, la descripción de la puntuación obtenida es:



- El análisis de SEO refleja un buen cumplimiento de las prácticas básicas de optimización para motores de búsqueda, aunque se aprecia que se detectó la falta de una meta descripción en el documento, lo que puede afectar la comprensión del contenido por parte de los motores de búsqueda. Lighthouse nos recomienda formatear el HTML de manera más adecuada para mejorar la indexación y revisar los validadores adicionales para optimizar aún más el rendimiento SEO.

9.2 Análisis de accesibilidad con Axe DevTools

Nuestro entorno está adaptado con la normativa **Axe DevTools**, tras realizar unas pruebas en nuestra aplicación, llegando a conseguir cero issues, en todas y cada una de las vistas que ha sido probado.



Para lograr estos resultados se han ido haciendo test y corrigiendo un par de anotaciones para contraste de texto con el color de fondo en el Header y en el título de nuestra aplicación, nada más que resaltar.

10. Evaluación heurística

Evaluación basada en las [heurísticas de Nielsen](#)

Heurística	Descripción	Ejemplo
1. Visibilidad del estado del sistema	El sistema informa a los usuarios sobre las acciones en curso a través de mensajes claros y oportunos.	Al enviar una cita desde el formulario, se muestra el mensaje "Cita creada exitosamente".
2. Relación entre el sistema y el mundo real	Utiliza términos comprensibles y familiares para el usuario, evitando lenguaje técnico.	La interfaz utiliza términos como "Paciente" y "Especialidad" en lugar de "Entidad de usuario" o "Categoría médica".
3. Control y libertad del usuario	Ofrece opciones para deshacer acciones o salir de procesos no deseados.	Botón "Cancelar" en los formularios de pacientes y citas para salir sin guardar cambios.
4. Consistencia y estándares	Sigue convenciones y utiliza el mismo diseño y terminología en toda la interfaz.	Los botones para "Editar" y "Eliminar" son consistentes en todas las tablas de datos.
5. Prevención de errores	Minimiza errores mediante controles y retroalimentación clara.	Deshabilita el botón de "Crear cita" si no se completan todos los campos obligatorios del formulario.
6. Reconocimiento en lugar de memorización	Reduce la carga cognitiva mostrando opciones disponibles en lugar de requerir memorización.	Menú desplegable para seleccionar pacientes y médicos al crear una cita, en lugar de ingresar manualmente los datos.

7. Flexibilidad y eficiencia de uso	Ofrece atajos o métodos rápidos sin comprometer la usabilidad para principiantes.	La funcionalidad de búsqueda rápida por nombre o DNI en la tabla de pacientes.
8. Estética y diseño minimalista	Mantiene una interfaz limpia con solo elementos relevantes visibles.	La página de inicio muestra solo las opciones esenciales del sistema: "Pacientes", "Médicos", "Agendar".
9. Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de errores	Muestra mensajes de error claros con explicaciones y soluciones sugeridas.	Mensaje: "Formato de fecha incorrecto. Por favor, utiliza el formato dd/mm/aaaa"
10. Ayuda y documentación	Proporciona documentación o herramientas de ayuda accesibles para tareas complejas.	Leer la Actividad 3, a modo de tutorial para guiar al usuario en el proceso de registro de una cita.

11. Mejoras aplicables

Se proponen diversas mejoras para optimizar la experiencia del usuario en la aplicación.

Como la incorporación de indicadores visuales de progreso, personalización de mensajes en función del contexto del usuario, y opciones de deshacer para acciones críticas. También, se sugiere unificar estilos tipográficos y colores para garantizar consistencia, así como validar campos en tiempo real para prevenir errores, se recomienda implementar sugerencias automáticas, atajos de teclado para usuarios avanzados, y reducir elementos innecesarios para un diseño más limpio, una cosa a destacar, es la necesidad de ofrecer documentación accesible y mensajes claros que faciliten la resolución de errores. Estas acciones fortalecerán la usabilidad, accesibilidad y eficiencia de la aplicación.

Conclusión

Para completar la memoria, el desarrollo de este proyecto ha logrado implementar un sistema completo y funcional para la gestión de un consultorio médico, abarcando tanto módulos funcionales como no funcionales. Hemos optado por utilizar tecnologías modernas como Django REST Framework para el backend y Vue.js para el frontend, se ha construido una solución robusta que incluye funcionalidades clave como la gestión de pacientes, citas, tratamientos, facturación, y roles de usuarios, entre otros. También se ha realizado un análisis técnico y de accesibilidad que valida su rendimiento, usabilidad y cumplimiento de estándares de diseño.

Hemos usado un SCRUM que nos ha permitido estructurar el desarrollo en 6 iteraciones organizadas, logrando así un avance gradual y consistente. Hemos realizado también pruebas documentadas y análisis heurísticos asegurando la calidad del sistema, destacando un rendimiento sobresaliente según Google Chrome Lighthouse y Axe DevTools, con espacio para pequeñas mejoras en accesibilidad y SEO.

En el proyecto hemos demostrado una sólida capacidad técnica y metodológica, enfrentándonos a desafíos como la curva de aprendizaje de nuevas tecnologías. Hay muchísimas horas de trabajo que no se ven reflejadas ya que han sido de aprendizaje, para llegar a un código entregable se han desestimado muchos por el camino, pero teníamos un compromiso y era entregar un producto de alta calidad.

En la generación de esta actividad se ha prestado atención a las normativas APA, generación de índice funcional, titulaciones, justificación del texto, numeración de apartados cronológicamente, control de viudas, etc. Toda la memoria ha sido revisada minuciosamente.



Bibliografía

Video entrega definición iteración #01 (Youtube): <https://youtu.be/UB5t81CrauE>

Video entrega Iteración #01 (Youtube): <https://youtu.be/t45tf4bSV-4>

Video entrega Iteración #02 (Youtube): <https://youtu.be/t45tf4bSV-4>

Video entrega Iteración #03 (Youtube): <https://youtu.be/eFSVOqkhCBI>

Video entrega Iteración #04 (Youtube): <https://youtu.be/hlJDualsr9k>

Video entrega Iteración #05 (Youtube): <https://youtu.be/daNdXw4VAX4>

Video entrega Iteración #06 (Youtube): <https://youtu.be/7NlFm7i209o>

Video entrega Proyecto Final (Youtube): <https://youtu.be/31G38rfDNOM>

Manual de uso de Vue.js: <https://es.vuejs.org/v2/guide/>

Manual de uso de django: <https://docs.djangoproject.com/es/5.1/>

Herramienta para creación de pdfs con javascript: <https://www.npmjs.com/package/jspdf>

Guía de creación de PDFs con js <https://libreriasjs.com/libreria-javascript-crear-pdf-jspdf/>

Herramienta para graficar en el módulo de Seguridad y auditoría: <https://apexcharts.com/>

Herramienta en línea para la creación de diagramas UML: <https://app.diagrams.net/>

Herramienta de gestión de incidencias (Jira):

<https://moisessevilla.atlassian.net/jira/software/projects/SGRCM/boards/1/backlog?selectedIssue=SGRCM-92>

Herramienta de gestión de versionado de código (GitHub):

<https://github.com/moisessevilla/47GIIN-Proyecto-Consultorio>

Se aporta firma digital certificada

Para que este documento tenga validez, se adjunta Firma Digital Certificada del desarrollador.