ARQUITECTURA C4

PRESENTADO POR:

DAVID ALEJANDRO PEÑALOSA VAZQUEZ

BRIAN STEVEN CUBILLOS CUBILLOS

KEVIN STEVEN ORTIZ VALLEJO

DANIELA MARTIN ORTIZ

INGENIERIA DE SISTEMAS Y COMPUTACIÓN

UNIVERSIDAD DE CUNDINAMARCA

OCTAVO SEMESTRE

FUSAGASUGÁ

2024

Introducción

Este documento proporciona un plan completo para implementar el proyecto de Fisioterapia Online, detallando la arquitectura, el modelo C4 y la implementación en Django. Se busca una solución modular monolítica con la posibilidad de escalar a microservicios en el futuro.

2. Arquitectura de Software

La arquitectura elegida es una **arquitectura modular monolítica** que divide el sistema en módulos independientes pero que forman parte de un único despliegue. Esta estructura permite un desarrollo y mantenimiento más eficiente, asegurando la sostenibilidad del proyecto.

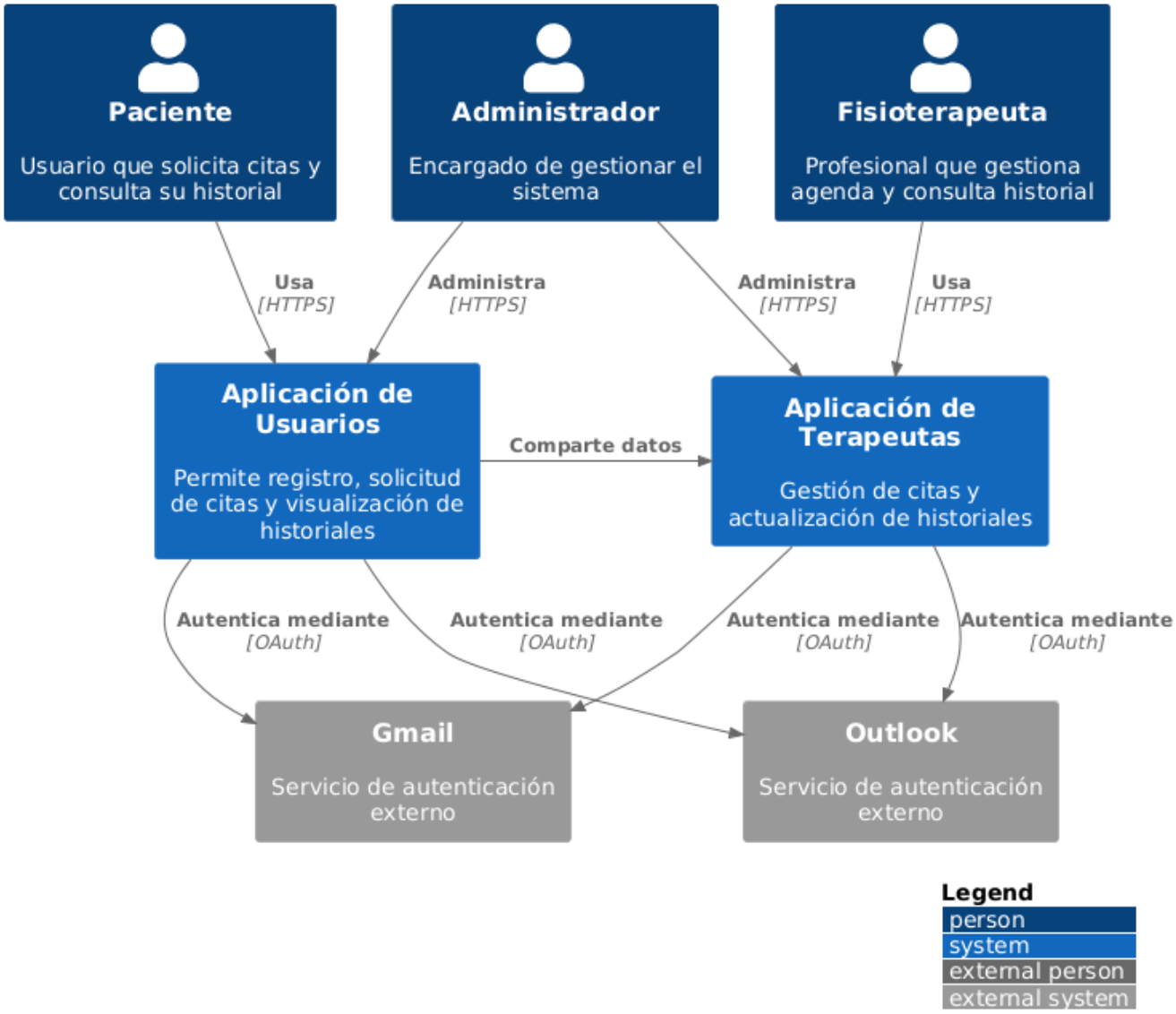
2.1 Módulos Identificados

- **Módulo de Usuarios**: Registros, autenticación con dos factores, inicio de sesión con Gmail/Outlook y recuperación de contraseñas.
- **Módulo de Citas**: Solicitud y gestión de citas por parte de los pacientes y consulta de agendas por parte de los fisioterapeutas.
- **Módulo de Historial de Pacientes**: Consultas y actualizaciones del historial de pacientes por los fisioterapeutas.
- **Módulo de Administración**: Gestión y supervisión de las actividades de la aplicación (opcional).

3. Modelo C4 Aplicado al Proyecto

El modelo C4 permite representar la arquitectura a diferentes niveles de abstracción. A continuación, se detalla cada nivel para el proyecto de Fisioterapia Online con la inclusión de dos aplicaciones independientes para usuarios y terapeutas:

3.1 Nivel 1: Diagrama de Contexto



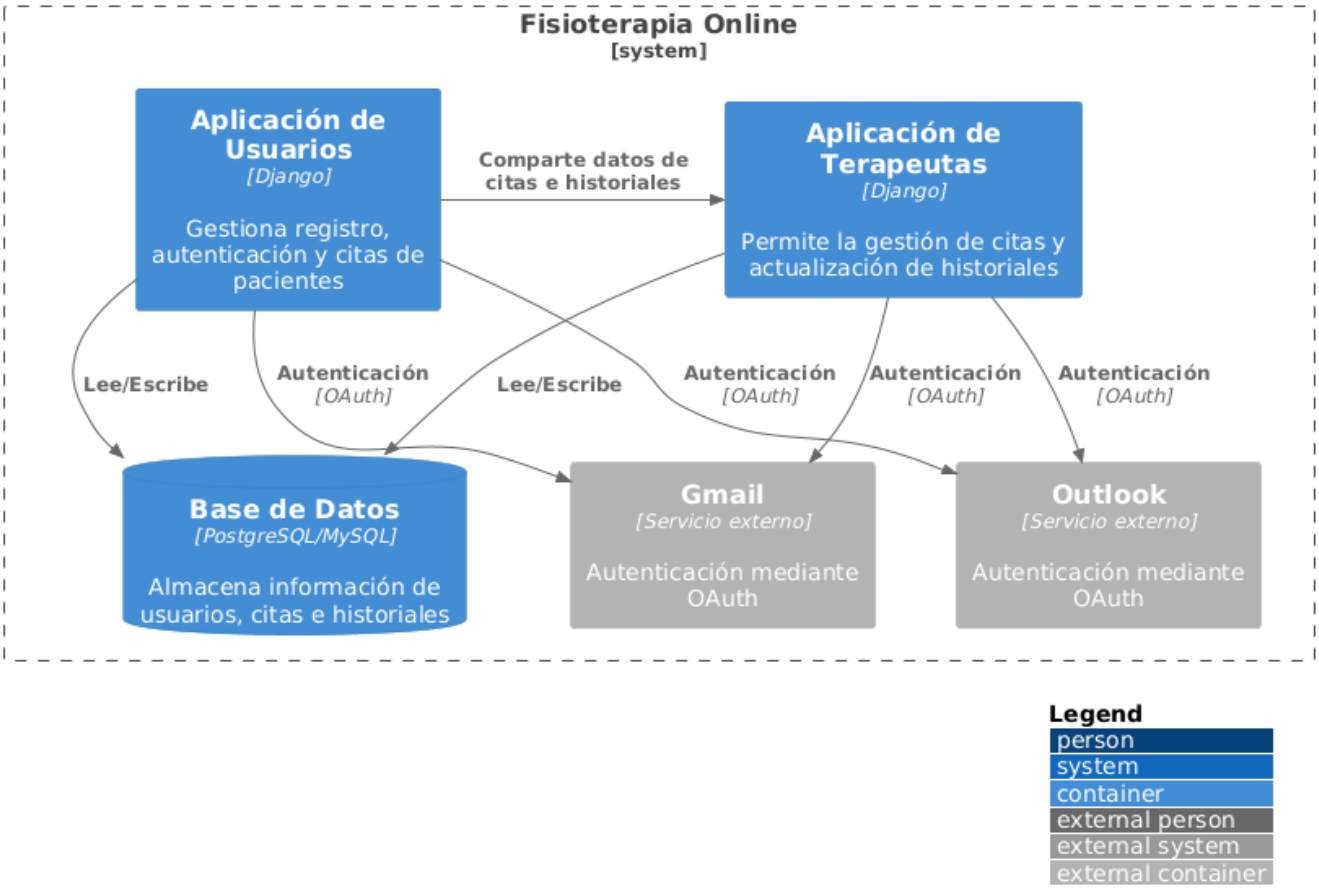
Objetivo: Mostrar una vista general del sistema y su interacción con los usuarios y sistemas externos.

- **Actores Principales**:
    - **Pacientes**: Usuarios que solicitan citas y consultan su historial.
    - **Fisioterapeutas**: Profesionales que gestionan la agenda y consultan el historial de los pacientes.
    - **Administradores**: Encargados de gestionar el sistema y supervisar las actividades.
    - **Sistemas externos**: Servicios de autenticación como Gmail y Outlook.
- **Sistema Central**:
    - **Aplicación de Fisioterapia Online** que permite la interacción de pacientes y fisioterapeutas con las funcionalidades principales.

- **Aplicación de Usuarios**: Utilizada por los pacientes para el registro, solicitud de citas y visualización de historiales.
- **Aplicación de Terapeutas**: Utilizada por los fisioterapeutas para gestionar citas y actualizar historiales.

Descripción: La aplicación interactúa con los pacientes y fisioterapeutas para gestionar citas y el historial de los pacientes, mientras que se integra con servicios de autenticación externos para mejorar la seguridad y la facilidad de uso.

3.2 Nivel 2: Diagrama de Contenedores



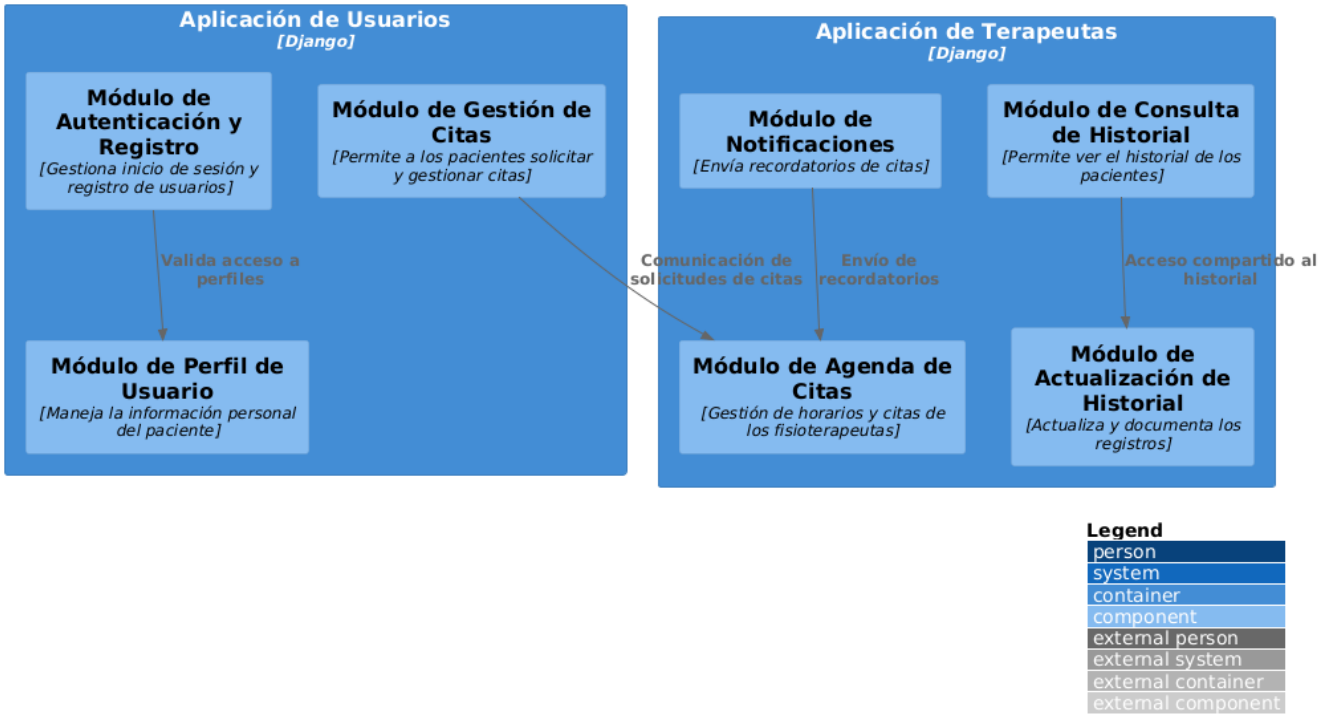
Objetivo: Mostrar los contenedores que componen el sistema y sus interacciones.

- **Aplicación Web de Usuarios (Django)**:
  - Contiene la lógica de negocio para registros, autenticación y gestión de citas de los pacientes.
  - Proporciona vistas y API RESTful para la interacción de los pacientes.
- **Aplicación Web de Terapeutas (Django)**:
  - Contiene la lógica de negocio para la gestión de citas, consulta y actualización de historiales de los pacientes.
  - Proporciona un panel para visualizar agendas y detalles de los pacientes.

- **Base de Datos**:
  - **PostgreSQL** o **MySQL**, utilizada como una base de datos central que es compartida por ambas aplicaciones para almacenar información de usuarios, citas y registros de historial.
- **Servicios de Autenticación Externa**:
  - Integración con APIs de Gmail/Outlook para el inicio de sesión seguro.
  - Servicios de doble factor de autenticación para mejorar la seguridad.
- **Servidor de Archivos** (opcional): Para almacenar documentos y archivos relacionados con los pacientes.

Descripción de la Interacción: Las aplicaciones de usuarios y terapeutas se comunican con la base de datos central para almacenar y recuperar datos, y con los servicios de autenticación para gestionar los inicios de sesión. Ambas aplicaciones funcionan de forma independiente pero comparten los mismos datos.

3.3 Nivel 3: Diagrama de Componentes



Objetivo: Desglosar los contenedores en sus componentes y mostrar sus interacciones.

Aplicación de Usuarios:

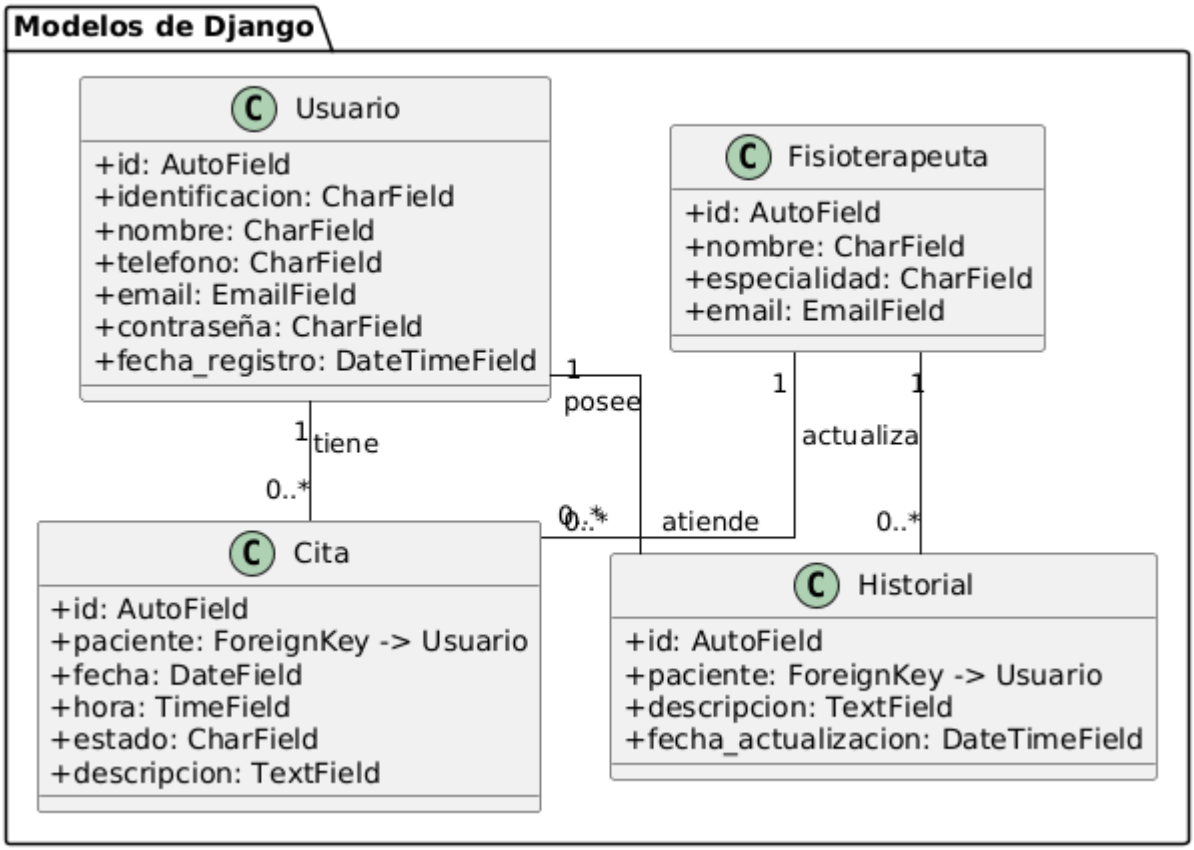
- **Módulo de Autenticación y Registro**: Maneja el inicio de sesión, el registro y la autenticación de dos factores.
- **Módulo de Gestión de Citas**: Permite a los pacientes solicitar, modificar y cancelar citas.
- **Módulo de Perfil de Usuario**: Gestiona la información personal y las preferencias del paciente.

Aplicación de Terapeutas:

- **Módulo de Agenda de Citas**: Permite a los fisioterapeutas ver y gestionar sus horarios y citas asignadas.
- **Módulo de Consulta de Historial**: Proporciona acceso a los historiales de los pacientes.
- **Módulo de Actualización de Historial**: Los fisioterapeutas pueden registrar y actualizar la información después de las sesiones de terapia.
- **Módulo de Notificaciones**: Envía recordatorios de citas y notificaciones de actualizaciones relevantes.

Descripción de la Interacción: Cada módulo de las aplicaciones se comunica con el componente central de la aplicación Django que actúa como intermediario entre la base de datos y la lógica de negocio. Las notificaciones se envían a través de servicios de mensajería integrados para mantener a los usuarios y terapeutas informados.

3.4 Nivel 4: Diagrama de Código



Objetivo: Mostrar el diseño de las clases y códigos clave.

Modelo de Usuario (usuarios/models.py):

from django.contrib.auth.models import AbstractUser

from django.db import models

```
class Usuario(AbstractUser):
    identificacion = models.CharField(max_length=20, unique=True)
    telefono = models.CharField(max_length=15)
    # Otros campos personalizados
```

Modelo de Cita (citas/models.py):

```
from django.db import models
from usuarios.models import Usuario

class Cita(models.Model):
    paciente = models.ForeignKey(Usuario, on_delete=models.CASCADE)
    fecha = models.DateField()
    hora = models.TimeField()
        estado = models.CharField(max_length=10, choices=[('Pendiente', 'Pendiente'), ('Completada', 'Completada')])
    # Otros campos relacionados
```

Modelo de Historial (historial/models.py):

```
from django.db import models
from usuarios.models import Usuario

class Historial(models.Model):
    paciente = models.ForeignKey(Usuario, on_delete=models.CASCADE)
    descripcion = models.TextField()
    fecha_actualizacion = models.DateTimeField(auto_now=True)
    # Otros campos si es necesario
```

4. Implementación de Django

4.1 Configuración Inicial

- **Crea el proyecto y los módulos**:

```
django-admin startproject fisioterapia_online
```

```
cd fisioterapia_online
python manage.py startapp usuarios
python manage.py startapp citas
python manage.py startapp historial
```

- **Configura la base de datos en `settings.py`**:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',  # o 'mysql', 'sqlite3', etc.
        'NAME': 'nombre_de_la_base_de_datos',
        'USER': 'usuario',
        'PASSWORD': 'contraseña',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

4.2 Desarrollo de Funcionalidades

- **Registro y Autenticación de Usuarios**: Usa `django.contrib.auth` y personaliza el modelo de usuario para incluir autenticación de dos factores con bibliotecas como `django-otp`.
- **Solicitud y Gestión de Citas**: Implementa vistas y formularios en `citas/views.py` para que los pacientes puedan solicitar citas y los fisioterapeutas vean su agenda.
- **Historial de Pacientes**: Crea vistas en `historial/views.py` para que los fisioterapeutas puedan consultar y actualizar los registros.

5. Plan de Desarrollo y Documentación

- **Gestión de Proyecto en Azure DevOps**: Planifica los sprints y divide las tareas en historias de usuario. Usa repositorios en GitHub o Azure DevOps con control de versiones.
- **Documentación C4**: Asegúrate de documentar cada nivel de arquitectura para mantener la transparencia y la comunicación con los desarrolladores.

6. Pruebas y Calidad

- **Pruebas Unitarias y de Integración**: Implementa pruebas unitarias con `pytest` o `unittest` para validar las funcionalidades principales de cada módulo. Realiza pruebas de

integración para verificar que los componentes trabajen juntos correctamente y que las interacciones entre la aplicación de usuarios, la aplicación de terapeutas y la base de datos funcionen sin problemas.

- **Pruebas de Usuario y Aceptación**: Realiza pruebas con usuarios reales y fisioterapeutas para evaluar la experiencia de uso, asegurando que la interfaz sea intuitiva y que las funcionalidades cumplan con los requisitos establecidos.
- **Revisiones de Código**: Implementa revisiones de código colaborativas para mantener la calidad del proyecto, identificando posibles mejoras en la estructura y corrección de errores antes de integrarlos en la rama principal.
- **Auditorías de Seguridad**: Realiza auditorías periódicas de seguridad, enfocándote en la protección de datos sensibles, autenticación y autorización. Asegúrate de proteger contra vulnerabilidades comunes como inyecciones SQL y ataques CSRF.
- **Herramientas de Automatización**: Utiliza herramientas como Jenkins o GitHub Actions para integrar la ejecución de pruebas automáticas en el flujo de trabajo de desarrollo, garantizando que cada cambio pase por un control de calidad.

7. Despliegue y Mantenimiento

- **Contenerización con Docker**: Configura Docker para contenerizar ambas aplicaciones y la base de datos, facilitando la replicación del entorno de desarrollo en producción.
- **Configuración de Servidores**: Utiliza servidores como Nginx para servir la aplicación y un servidor de aplicaciones como Gunicorn para ejecutar la aplicación Django en producción.
- **Monitoreo y Logs**: Implementa sistemas de monitoreo como Prometheus y herramientas de visualización como Grafana para supervisar el rendimiento de la aplicación. Usa logging estructurado con herramientas como ELK Stack (Elasticsearch, Logstash y Kibana) para analizar eventos y resolver problemas rápidamente.
- **Copia de Seguridad y Recuperación de Desastres**: Configura un plan de respaldo regular de la base de datos y archivos esenciales para garantizar la continuidad del servicio en caso de fallos.
- **Escalabilidad**: Planifica la posibilidad de escalar la aplicación horizontalmente, dividiendo los módulos en microservicios si la carga aumenta significativamente o si se necesitan nuevas funcionalidades complejas.

Conclusión

La solución arquitectónica propuesta para el proyecto de Fisioterapia Online proporciona una base sólida y sostenible para el desarrollo de la aplicación. La arquitectura modular monolítica permite un desarrollo ágil y una gestión clara, mientras que las medidas de calidad y seguridad garantizan la confiabilidad del sistema. Con la documentación detallada y la implementación de pruebas, el proyecto está preparado para adaptarse a futuras expansiones y escalabilidad. La estructura planteada permite la colaboración efectiva entre desarrolladores y facilita la comunicación con las partes interesadas, asegurando un producto de alta calidad que puede evolucionar con las necesidades de los usuarios y del mercado.