Documentación de Sistema

Contenido

[Introducción 3](#_Toc181738867)

[Documentación Técnica: 4](#_Toc181738868)

[Configuración e instalación 7](#_Toc181738869)

[Documentación de Administración 10](#_Toc181738870)

[Gestión del Sistema 11](#_Toc181738871)

[Seguridad 11](#_Toc181738872)

[4. Mantenimiento 11](#_Toc181738873)

[5. Integraciones y Compatibilidad 12](#_Toc181738874)

[6. Resolución de Problemas Avanzados 12](#_Toc181738875)

Introducción

Este informe proporciona una documentación detallada del sistema, abarcando aspectos clave de su administración, así como los pasos necesarios para su configuración e instalación. Se describen los módulos principales, los requisitos técnicos y el proceso de implementación, con el objetivo de ofrecer una guía completa para el manejo eficiente y el mantenimiento del sistema.

# Documentación Técnica:

Este documento proporciona información detallada sobre la arquitectura, diseño y funcionamiento interno del sistema desarrollado con Django y React. Está dirigido a desarrolladores y técnicos que necesiten entender y mantener el sistema.

* **Contenido típico:**
  + Diagramas de arquitectura.

Diagrama

Descripción generada automáticamente

**Especificaciones Técnicas**

El sistema utiliza una arquitectura basada en Django y React, implementando una API REST que permite la comunicación entre el backend y el frontend. Estas son algunas especificaciones técnicas clave:

* **Backend**: Python 3.13 y Django 5.1.2, con una base de datos MySQL para persistencia de datos.
* **Frontend**: React 18.3.1, que permite crear una interfaz de usuario dinámica y modular.
* **Dependencias Adicionales**: Herramientas como djangorestframework y mysqlclient para el backend, y axios y react-router-dom para el frontend.
* **Entorno de Desarrollo**: Recomendado con Visual Studio Code y un entorno virtual para gestionar dependencias de Python.

El sistema está optimizado para funcionar en servidores con especificaciones básicas, con capacidad de escalar en caso de que se necesite un manejo de datos mayor.

**Descripciones de Módulos y Componentes**

1. **Módulo de Autenticación**:
   * Gestiona el inicio de sesión de usuarios utilizando RUT y contraseña.
   * Implementa seguridad básica y manejo de sesiones en Django, con opciones de autenticación en React.
2. **Módulo de Gestión de Citas Médicas**:
   * Permite la creación, visualización, modificación y cancelación de citas por parte de los médicos.
   * Incluye un sistema de notificaciones automáticas para avisar sobre cambios de estado de citas.
3. **Módulo Administrativo**:
   * Permite a los administradores generar reportes de citas, especialmente enfocándose en estadísticas de cancelación.
   * Incluye acceso a la configuración de usuarios y asignación de roles dentro del sistema.
4. **Interfaz de Usuario (Frontend)**:
   * Construida con React, organiza componentes reutilizables para cada sección de la aplicación.
   * Incluye un enrutamiento efectivo y formularios interactivos usando react-hook-form.

**Detalles de Algoritmos y Estructuras de Datos**

1. **Algoritmos de Filtrado y Ordenación de Citas**:
   * Utiliza consultas de Django ORM para obtener citas ordenadas y filtradas por fecha y estado.
   * Aplica paginación para evitar la sobrecarga en el frontend, especialmente en el módulo de reportes.
2. **Estructuras de Datos**:
   * **Base de Datos**: MySQL, con tablas principales para usuarios, citas, roles y notificaciones.
   * **Estructuras JSON**: Utilizadas para el intercambio de datos entre el backend y el frontend mediante API REST.

**Documentación de Código Fuente**

El código fuente está organizado en carpetas principales para Django y React, siguiendo convenciones estándar para facilitar su mantenimiento:

1. **Backend (Django)**:
   * **views.py**: Define las vistas para cada endpoint de la API.
   * **models.py**: Contiene los modelos de datos (ej. Citas, Usuarios) que se almacenan en MySQL.
   * **serializers.py**: Convierte datos de Django ORM a JSON y viceversa, optimizando la transmisión de datos.
2. **Frontend (React)**:
   * **components/**: Incluye todos los componentes reutilizables como formularios de ingreso y listas de citas.
   * **services/**: Contiene servicios de API que usan axios para comunicarse con el backend.
   * **context/**: Maneja el estado global de la aplicación, especialmente para la autenticación y el manejo de citas.

Configuración e instalación  
  
Requisitos del Sistema  
A continuación, se detallan los requisitos de hardware y software necesarios para ejecutar el sistema:

* **Hardware**: Se requiere un servidor con base de datos MySQL.
* **Software**:
  + Python 3.13
  + Django 5.1.2
  + React 18.3.1

Dependencias del Proyecto de Django  
Las siguientes dependencias deben ser instaladas para que el proyecto funcione correctamente. En el siguiente paso se explicará cómo instalarlas.

* makefile
* Copiar código
* asgiref==3.8.1
* attrs==24.2.0
* certifi==2024.8.30
* charset-normalizer==3.4.0
* coreapi==2.3.3
* coreschema==0.0.4
* Django==5.1.2
* django-cors-headers==4.5.0
* djangorestframework==3.15.2
* drf-spectacular==0.27.2
* idna==3.10
* inflection==0.5.1
* itypes==1.2.0
* Jinja2==3.1.4
* jsonschema==4.23.0
* jsonschema-specifications==2024.10.1
* MarkupSafe==3.0.2
* mysqlclient==2.2.5
* PyMySQL==1.1.1
* PyYAML==6.0.2
* referencing==0.35.1
* requests==2.32.3
* rpds-py==0.20.0
* sqlparse==0.5.1
* tzdata==2024.2
* uritemplate==4.1.1
* urllib3==2.2.3

Dependencias del Proyecto de React

* "axios": "^1.7.7",
* "react-hook-form": "^7.53.1",
* "react-hot-toast": "^2.4.1",
* "react-router-dom": "^6.27.0"

Procedimiento de Instalación

**Instalación de Herramientas Necesarias**  
Antes de proceder con la instalación y configuración del sistema en diferentes entornos, asegúrese de tener las siguientes herramientas instaladas:

* **Python** (se recomienda la versión 3.13) para el manejo del backend.
* **Node.js** para el manejo del frontend.

**Configuración del Entorno Virtual para Django (Backend)**

Abra una terminal y cree una instancia virtual en el directorio del proyecto con el siguiente comando:

* py venv nombre\_instancia

Luego, active el entorno virtual:

* nombre\_instancia/Scripts/actívate

Con el entorno virtual activado, instale Django ejecutando:

* python -m pip install django

**Instalación de Dependencias del Proyecto**

Después de instalar Django, debe instalar las dependencias necesarias listadas en el archivo requirements.txt. Para hacerlo, ejecute el siguiente comando:

* pip install -r requirements.txt

**Ejecutar el Servidor de Django**  
Una vez instaladas las dependencias, inicie el servidor de Django con el siguiente comando:

* py manage.py runserver

**Configuración de la Conexión a la Base de Datos**

* En el archivo settings.py dentro de la carpeta de su proyecto, configure la conexión con la base de datos que se utilizará. En este caso, si la base de datos está conectada localmente, deberá cambiar los parámetros según sea necesario.

**Instalación de React (Frontend)**  
Para configurar el frontend con React, siga estos pasos en una nueva terminal:

Primero, active la misma instancia virtual que se utilizó para el backend.

Luego, diríjase a la carpeta donde se encuentra el código del frontend (por ejemplo, la carpeta pantallas).

Instale React y sus dependencias con el siguiente comando:

* npm install

**Ejecutar el Frontend**  
Para iniciar el servidor de desarrollo de React, ejecute el siguiente comando:

* npm run dev

Si necesita instalar dependencias adicionales, puede hacerlo con los siguientes comandos:

* npm install react-router-dom react-hot-toast axios react-hook-form

# Documentación de Administración

**Objetivos**

El propósito de la administración del sistema es gestionar usuarios, asignar permisos y revisar el panel de control (dashboard) para monitorear métricas.

**Público objetivo**

El sistema está dirigido a administradores, médicos y personal autorizado que necesitan gestionar usuarios, realizar ajustes en las configuraciones, monitorear el rendimiento mediante el dashboard y anular horas de citas según sea necesario.

# Gestión del Sistema

**Administración de usuarios y roles**

La administración de usuarios permite gestionar y asignar diferentes roles en el sistema, asegurando que cada usuario tenga los permisos adecuados según su rol (administrador, médico, paciente).

Configuración de permisos

La configuración de permisos se establece a través de un inicio de sesión principal. Según el RUT ingresado, el sistema mostrará diferentes vistas y funcionalidades en función de los privilegios asignados al usuario, ya sea médico, administrador u otro tipo de usuario.

**Monitorización del sistema**

La supervisión y el mantenimiento del sistema incluyen el monitoreo continuo del rendimiento, la revisión de logs de actividad, y la detección de posibles fallos o incidentes. Esto asegura que el sistema funcione de manera óptima y permita intervenir rápidamente en caso de problemas.

# Seguridad

**Políticas de seguridad**

Las políticas de seguridad del sistema incluyen medidas de protección como el cifrado de datos, la autenticación mediante credenciales seguras, y el control de accesos basado en roles. Además, se implementan políticas de contraseña robustas y se restringe el acceso a funciones sensibles solo a usuarios autorizados, garantizando la integridad y confidencialidad de la información.

**Configuración de cortafuegos**

El sistema debe contar con una configuración de cortafuegos que permita filtrar el tráfico y proteger el acceso no autorizado a datos y servicios críticos.

**Gestión de certificados**

Se gestionan certificados digitales para asegurar las conexiones y la autenticidad de los datos en transmisión, aumentando así la seguridad de las operaciones.

# 4. Mantenimiento

**Tareas programadas**

Se deben programar tareas automáticas para la supervisión del sistema, incluyendo limpieza de logs antiguos y actualizaciones de rendimiento.

**Respaldo y recuperación de datos**

Es fundamental realizar respaldos periódicos de los datos y contar con un plan de recuperación para garantizar la disponibilidad de la información en caso de fallos o pérdida de datos.

**Actualizaciones y parches**

El sistema debe mantenerse actualizado mediante la aplicación de parches de seguridad y mejoras de funcionalidades según las recomendaciones del proveedor o desarrollador.

# 5. Integraciones y Compatibilidad

**Conexión con otros sistemas**

El sistema permite integrarse con otros sistemas utilizados por RedSalud, facilitando el flujo de información y evitando redundancia de datos.

**Compatibilidad con versiones anteriores**

El sistema es compatible con versiones anteriores para asegurar la continuidad y facilitar las actualizaciones sin pérdida de funcionalidades existentes.

# 6. Resolución de Problemas Avanzados

**Logs y diagnósticos**

La revisión de logs y diagnósticos permite identificar rápidamente incidentes, facilitando el análisis y resolución de problemas en el sistema.

**Contacto con soporte técnico**

Para problemas avanzados o no resueltos con la guía, se recomienda contactar al soporte técnico, quienes podrán brindar asistencia detallada.