Documentación del Sistema de Procesamiento de Imágenes Médicas DICOM

1. Punto de Entrada Principal

Implementación de Carga de Archivos

- **Interfaz Web:** Se ha desarrollado una interfaz web utilizando Django que permite a los usuarios cargar archivos DICOM. Esto se realiza mediante un formulario en el modelo DICOMFile, que incluye validaciones para asegurar que solo se acepten archivos con la extensión. dcm.
- **Validación de Archivos:** El proceso de carga valida tanto el tamaño del archivo como su tipo para asegurar que se trata de un archivo DICOM válido. Los archivos se almacenan en una ubicación específica del servidor (dicom_file/).

Cola de Procesamiento

- **Celery:** Se ha configurado Celery para manejar la carga y procesamiento asíncrono de archivos grandes. Los archivos cargados se encolan para su procesamiento sin bloquear el servidor, lo que permite una gestión eficiente de grandes volúmenes de datos.

2. Sistema de Procesamiento Distribuido

Grupo de Servidores (Trabajadores)

- **Configuración de Trabajadores:** El sistema utiliza múltiples servidores como trabajadores, cada uno procesando imágenes DICOM en cola mediante Celery.
- **Equilibrio de Carga:** Se ha implementado un equilibrio de carga básico para distribuir las tareas entre los servidores trabajadores, optimizando el uso de recursos y mejorando la eficiencia del procesamiento.

Escalabilidad

- **Escalado Horizontal:** El sistema está diseñado para escalar horizontalmente, permitiendo la adición o eliminación de servidores según la demanda de procesamiento. Celery facilita la incorporación de nuevos trabajadores para manejar la carga creciente.

3. Límites de Uso y Gestión de Recursos

Monitoreo y Gestión

- **Supervisión de Recursos:** Aunque la implementación de monitoreo avanzado no está completa, se ha considerado integrar herramientas como Prometheus o Grafana para supervisar el uso de recursos y evitar la sobrecarga de los servidores.
- **Algoritmo de Asignación de Tareas:** Actualmente, las tareas se asignan de manera básica. Se pueden implementar algoritmos más avanzados que consideren la carga de CPU y memoria para una distribución más equitativa de las tareas.

4. Integración de Bases de Datos

Base de Datos Distribuida

- **Diseño de Base de Datos:** Se utiliza PostgreSQL para almacenar los resultados del procesamiento en la base de datos bddicomimage. Se han creado dos tablas principales: DICOMFile y MedicalImageResult, que manejan los archivos cargados y los resultados del procesamiento.
- **Registro de Actividades:** La base de datos registra todas las actividades de procesamiento, incluidos los resultados exitosos y los fallos, para permitir un seguimiento detallado del rendimiento del sistema.

5. Sistema de Información para Usuarios

Interfaz de Usuario

- **Portal Web:** Se ha desarrollado un portal web en Django donde los usuarios pueden iniciar sesión para ver el estado de las imágenes enviadas y recuperar los resultados procesados.
- **Interacción con la Base de Datos:** El portal consulta la base de datos distribuida en tiempo real para mostrar los datos procesados a los usuarios. El sistema de autenticación y autorización garantiza que los usuarios solo puedan acceder a sus propios datos.

Autenticación y Control de Acceso

- **Implementación de Autenticación:** Se ha implementado la autenticación de usuarios utilizando el sistema de autenticación incorporado de Django. Se utilizan roles y permisos para controlar el acceso a diferentes partes del sistema.

6. Consideraciones Adicionales

Seguridad y Privacidad

- **Cumplimiento de Regulaciones:** Aunque la implementación completa de cumplimiento con HIPAA o GDPR está en desarrollo, se han implementado medidas de seguridad como el cifrado de datos. Los datos sensibles se cifran durante la transmisión utilizando TLS/SSL y se almacenan de forma segura.
- **Cifrado:** El cifrado de datos en reposo y en tránsito se asegura utilizando AES para los datos en reposo y TLS/SSL para los datos en tránsito, protegiendo la privacidad de los datos médicos.

Registro y Manejo de Errores

- **Mecanismo de Registro:** Se ha implementado un sistema básico de registro para supervisar el estado del sistema y rastrear errores. Los registros incluyen detalles sobre el procesamiento de tareas, fallos y éxitos.
- **Manejo de Errores:** El sistema maneja errores y tiempos de inactividad mediante mecanismos de recuperación automática y redundancia. Los errores en servidores individuales se gestionan mediante reinicios automáticos y redistribución de tareas.

7. Descripción del Proyecto

Este proyecto Django está diseñado para el procesamiento de imágenes médicas DICOM. Incluye una interfaz de administración para gestionar archivos DICOM y resultados de procesamiento.

8. Estructura de la Base de Datos

Tablas Generales de Django

1. **auth_group**
   - **Descripción:** Almacena grupos de usuarios.
   - **Campos:** id, name.
2. **auth_group_permissions**
   - **Descripción:** Relación entre grupos y permisos.
   - **Campos:** id, group_id, permission_id.
3. **auth_permission**
   - **Descripción:** Almacena permisos.
   - **Campos:** id, name, content_type_id, codename.
4. **auth_user**
   - **Descripción:** Almacena usuarios.
   - **Campos:** id, password, last_login, is_superuser, username, first_name, last_name, email, is_staff, is_active, date_joined.
5. **auth_user_groups**
   - **Descripción:** Relación entre usuarios y grupos.
   - **Campos:** id, user_id, group_id.
6. **auth_user_user_permissions**
   - **Descripción:** Relación entre usuarios y permisos.
   - **Campos:** id, user_id, permission_id.
7. **django_admin_log**
   - **Descripción:** Registro de acciones en el panel de administración.
   - **Campos:** id, action_time, user_id, content_type_id, object_id, object_repr, action_flag, change_message.
8. **django_content_type**
   - **Descripción:** Almacena tipos de contenido.
   - **Campos:** id, app_label, model.
9. **django_migrations**
   - **Descripción:** Registro de las migraciones aplicadas.
   - **Campos:** id, app, name, applied.
10. **django_session**
    - **Descripción:** Almacena sesiones de usuarios.
    - **Campos:** session_key, session_data, expire_date.
    -

Tablas Específicas del Proyecto

1. **upload_dicomfile**
   - **Descripción:** Almacena los archivos DICOM cargados.
   - **Campos:** id, file, uploaded_at.
2. **upload_dicomresult**
   - **Descripción:** Almacena los resultados del procesamiento de DICOM.

- **Campos:** id, device_name, raw_data, average_before_normalization, average_after_normalization, data_size.

9. Panel de Administración de Django

Configuración del Panel de Administración

- **Encabezado del Sitio:** "Sistema de procesamiento DICOM"
- **Título del Sitio:** "Sistema de procesamiento DICOM"
- **Título de Índice:** "Bienvenido al panel de administración"

Modelos y Administradores

- **DICOMFile**
  - **Campos en la Vista de Lista:** file, uploaded_at
  - **Campos Buscables:** file
- **MedicalImageResult**
  - **Campos en la Vista de Lista:** file_name, processed_at
  - **Campos Buscables:** file_name

10. Permisos de los Usuarios

Los permisos en Django están gestionados a través de grupos y permisos específicos. Los usuarios pueden tener diferentes niveles de acceso según sus roles:

- **Grupos:** Los usuarios se pueden asignar a grupos para administrar permisos de manera colectiva.
- **Permisos:** Los permisos se pueden asignar directamente a usuarios o grupos para controlar el acceso a diferentes partes del sistema.
-

11. Gestión de la Base de Datos

Consultas Básicas

- **Consultar Todas las Tablas:** \dt
- **Consultar Datos en una Tabla:** SELECT * FROM <nombre_de_la_tabla>;
- **Ver Estructura de una Tabla:** \d <nombre_de_la_tabla>

Ejemplos de Consultas

- **Consultar Todos los Archivos DICOM:** SELECT * FROM upload_dicomfile;
- **Consultar Todos los Resultados Médicos:** SELECT * FROM upload_dicomresult;
- **Ver Estructura de la Tabla upload_dicomfile:** \d upload_dicomfile
- **Ver Estructura de la Tabla upload_dicomresult:** \d upload_dicomresult