Diseño Completo de Endpoints de la API (RESTful)
La API del backend será el puente entre la aplicación de escritorio y la infraestructura en la nube, permitiendo la gestión de toda la información del consultorio. Se diseñará siguiendo los principios de RESTfulness para garantizar la coherencia, la escalabilidad y la facilidad de uso.
Para Gestión de Pacientes
El módulo de Gestión Integral del Expediente del Paciente es crucial para centralizar toda la información crítica y permitir un acceso inmediato y un seguimiento eficiente. Los siguientes endpoints facilitarán estas operaciones:

- 1. GET /patients
  - Método HTTP: GET
  - Ruta exacta: /api/v1/patients
  - Descripción: Permite obtener una lista de todos los pacientes registrados en el sistema. Soporta parámetros para búsqueda, filtrado, paginación y ordenamiento, facilitando la "Búsqueda Avanzada". Los filtros pueden incluir fecha, estado o tipo de consulta.
- 2. GET /patients/{id}
  - Método HTTP: GET
  - Ruta exacta: /api/v1/patients/{id} (donde {id} es el identificador único del paciente)
  - Descripción: Recupera los detalles completos de un paciente específico, incluyendo su perfil y la información general de su historial. Esto es útil para la función "Ver Detalles" o para abrir la ficha completa del paciente.
- 3. POST /patients
  - Método HTTP: POST
  - Ruta exacta: /api/v1/patients
  - Descripción: Crea un nuevo registro de paciente en el sistema con todos sus datos demográficos y de contacto.
- 4. PUT /patients/{id}
  - Método HTTP: PUT
  - Ruta exacta: /api/v1/patients/{id}
  - Descripción: Actualiza los datos de un paciente existente identificado por su ID. Permite modificar cualquier campo del perfil del paciente.
- 5. DELETE /patients/{id}
  - Método HTTP: DELETE
  - Ruta exacta: /api/v1/patients/{id}
  - Descripción: Realiza un "borrado suave" (soft delete) de un paciente. Esto significa que el registro no se elimina permanentemente de la base de datos, sino que se marca como inactivo o archivado (ej., is_deleted = true). El sistema está diseñado para que el registro se marque como inactivo, no se borre permanentemente.
- 6. POST /patients/{id}/archive
  - Método HTTP: POST
  - Ruta exacta: /api/v1/patients/{id}/archive
  - Descripción: Archiva un paciente específico por su ID. Esta acción complementa el borrado suave, marcando el paciente como archivado o inactivo. Esta funcionalidad puede aplicarse a múltiples pacientes seleccionados para acciones en lote.

Documento: Arq. Backend y APIs
Diseño Completo de Endpoints de la API (RESTful) (Continuación)
La API del backend continúa su diseño siguiendo los principios de RESTfulness, garantizando la coherencia, la escalabilidad y la facilidad de uso para todas las operaciones del "Ecosistema Digital Inteligente para Cirugía Especial". Los datos estructurados se almacenan en la Base de Datos Relacional (Cloud SQL), mientras que los archivos digitales residen en Cloud Storage, ambos con cifrado en reposo para asegurar la confidencialidad de la información del paciente. Todas las interacciones con la API se realizarán de forma segura a través de HTTPS (TLS 1.3).
Para Gestión de Historial Clínico
El registro cronológico y detallado del historial clínico es una funcionalidad central para el seguimiento eficiente del paciente. Los siguientes endpoints permiten la interacción con esta información:

- 1. GET /patients/{id}/history
  - Método HTTP: GET

    ◦ Ruta exacta: /api/v1/patients/{id}/history (donde {id} es el identificador único del paciente)
    ◦ Descripción: Permite obtener el historial clínico completo de un paciente específico, recuperando todas las entradas cronológicas de consultas, diagnósticos, notas de evolución, procedimientos y tratamientos asociados.
• 2. POST /patients/{id}/history
    ◦ Método HTTP: POST
    ◦ Ruta exacta: /api/v1/patients/{id}/history
    ◦ Descripción: Permite agregar una nueva entrada al historial clínico de un paciente específico. Esta acción es el equivalente digital a registrar una nueva consulta o procedimiento, incluyendo campos como fecha de consulta, diagnóstico y notas.
• 3. PUT /history/{id}
    ◦ Método HTTP: PUT
    ◦ Ruta exacta: /api/v1/history/{id} (donde {id} es el identificador único de una entrada específica del historial)
    ◦ Descripción: Actualiza los datos de una entrada existente en el historial clínico, permitiendo la corrección o modificación de detalles como diagnósticos o notas.
• 4. DELETE /history/{id}
    ◦ Método HTTP: DELETE
    ◦ Ruta exacta: /api/v1/history/{id}
    ◦ Descripción: Realiza un "borrado suave" de una entrada específica del historial clínico por su ID. Esto significa que el registro se marca como inactivo o archivado en la base de datos, en lugar de ser eliminado permanentemente, preservando la integridad histórica de los datos.

Para Gestión de Citas

El módulo de agenda y citas es fundamental para optimizar la programación y el control del flujo de pacientes, reduciendo los procesos manuales. Los endpoints asociados son:

• 1. GET /appointments
    ◦ Método HTTP: GET
    ◦ Ruta exacta: /api/v1/appointments
    ◦ Descripción: Obtiene la lista de todas las citas programadas en el consultorio. Soporta parámetros para filtrar por fecha (ej., "Citas del Día"), por paciente o por estado (ej., "Confirmada", "Cancelada", "Atendida"), así como paginación y ordenamiento.
• 2. GET /appointments/{id}
    ◦ Método HTTP: GET
    ◦ Ruta exacta: /api/v1/appointments/{id} (donde {id} es el identificador único de la cita)
    ◦ Descripción: Recupera los detalles completos de una cita específica por su ID, incluyendo el paciente asociado, fecha, hora, tipo y estado.
• 3. POST /appointments
    ◦ Método HTTP: POST
    ◦ Ruta exacta: /api/v1/appointments
    ◦ Descripción: Crea una nueva cita en el calendario del consultorio. Requiere la información necesaria como el paciente, la fecha, la hora y el tipo de cita.
• 4. PUT /appointments/{id}
    ◦ Método HTTP: PUT
    ◦ Ruta exacta: /api/v1/appointments/{id}
    ◦ Descripción: Actualiza los datos de una cita existente identificada por su ID. Permite modificar su fecha, hora, tipo, o cambiar su estado (ej., de "Programada" a "Confirmada" o "Cancelada").
• 5. DELETE /appointments/{id}
    ◦ Método HTTP: DELETE
    ◦ Ruta exacta: /api/v1/appointments/{id}
    ◦ Descripción: Realiza un "borrado suave" de una cita por su ID. El registro de la cita se marca como inactivo en la base de datos, lo que permite mantener un historial completo de las citas incluso si son canceladas o no se realizan.

Para Gestión Documental

La gestión documental avanzada centraliza y asegura el almacenamiento de todos los archivos digitales del paciente, eliminando la dispersión y mejorando la seguridad. Los archivos se almacenan en Cloud Storage con cifrado en reposo.

• 1. POST /patients/{id}/documents
    ◦ Método HTTP: POST
    ◦ Ruta exacta: /api/v1/patients/{id}/documents

    ◦ Descripción: Permite subir un nuevo archivo digital (PDF, imagen, video corto, etc.) y asociarlo directamente al expediente de un paciente específico. El archivo se cargará a Cloud Storage y sus metadatos se registrarán en la base de datos.
• 2. GET /patients/{id}/documents
    ◦ Método HTTP: GET
    ◦ Ruta exacta: /api/v1/patients/{id}/documents
    ◦ Descripción: Obtiene una lista de todos los documentos digitales (metadatos como nombre, tipo, URL) asociados a un paciente específico.
• 3. GET /documents/{id}
    ◦ Método HTTP: GET
    ◦ Ruta exacta: /api/v1/documents/{id} (donde {id} es el identificador único del documento)
    ◦ Descripción: Recupera un documento específico por su ID. Esta acción puede devolver una URL pre-firmada o un stream de datos que permita la descarga o visualización directa del archivo desde Cloud Storage.
• 4. DELETE /documents/{id}
    ◦ Método HTTP: DELETE
    ◦ Ruta exacta: /api/v1/documents/{id}
    ◦ Descripción: Realiza un "borrado suave" de un documento digital por su ID. Solo se marca el registro del documento como inactivo en la base de datos, sin eliminar el archivo físico de Cloud Storage de forma inmediata, manteniendo así un registro auditable.

Documento: Arq. Backend y APIs
Diseño Completo de Endpoints de la API (RESTful) (Continuación)
La definición de los endpoints RESTful es crucial para la interacción segura y eficiente entre el Dashboard de Escritorio (Núcleo Administrativo Central) y el backend en la nube (Google Cloud Platform). Estos endpoints facilitan la autenticación de usuarios, la gestión de permisos, el registro de actividades para auditoría y la integración de capacidades de Inteligencia Artificial para automatización y asistencia. Todas las comunicaciones se realizarán a través de HTTPS (TLS 1.3), garantizando el cifrado de datos en tránsito.
Para Autenticación y Autorización
La autenticación y la autorización son la base de la seguridad del "Ecosistema Digital Inteligente para Cirugía Especial", asegurando que solo el personal autorizado acceda a la información y funcionalidades.
• 1. POST /login
    ◦ Método HTTP: POST
    ◦ Ruta exacta: /api/v1/auth/login
    ◦ Descripción: Este endpoint es el punto de entrada para que un usuario inicie sesión en el sistema. Recibe las credenciales (nombre de usuario y contraseña), las valida contra la base de datos, y si son correctas, devuelve un token de autenticación que será utilizado para futuras solicitudes. Es fundamental para aplicar el control de acceso basado en roles. La comunicación se realiza cifrada vía HTTPS (TLS 1.3).
• 2. Endpoints para Gestión de Usuarios
    ◦ La gestión de usuarios es una funcionalidad clave del "Núcleo Administrativo Central", accesible para roles como "Administrador" y "Admin Principal".
    ◦ GET /users
        ▪ Método HTTP: GET
        ▪ Ruta exacta: /api/v1/users
        ▪ Descripción: Obtiene una lista de todos los usuarios registrados en el sistema. Este endpoint permite a los administradores visualizar el personal del consultorio y sus roles asociados.
    ◦ GET /users/{id}
        ▪ Método HTTP: GET
        ▪ Ruta exacta: /api/v1/users/{id} (donde {id} es el identificador único del usuario)
        ▪ Descripción: Recupera los detalles completos del perfil de un usuario específico, incluyendo su información de contacto y el rol asignado.
    ◦ POST /users
        ▪ Método HTTP: POST
        ▪ Ruta exacta: /api/v1/users
        ▪ Descripción: Permite crear una nueva cuenta de usuario en el sistema. Esta operación incluye la asignación de un rol inicial (ej., "Asistente", "Admin Secundario").

    - PUT /users/{id}
        - Método HTTP: PUT
        - Ruta exacta: /api/v1/users/{id}
        - Descripción: Actualiza los datos de un usuario existente, como su nombre, información de contacto o, fundamentalmente, su rol en el sistema.
    - DELETE /users/{id}
        - Método HTTP: DELETE
        - Ruta exacta: /api/v1/users/{id}
        - Descripción: Realiza un "borrado suave" de una cuenta de usuario por su ID. El registro del usuario se marca como inactivo en la base de datos, manteniendo la integridad histórica y facilitando posibles recuperaciones.
- 3. Endpoints para Gestión de Roles
    - El sistema cuenta con un robusto "Sistema de Roles", con roles como "Admin Principal", "Admin Secundario", "Asistente", "Invitado (Solo Lectura)". La información de roles se almacena en la tabla ROLES.
    - GET /roles
        - Método HTTP: GET
        - Ruta exacta: /api/v1/roles
        - Descripción: Obtiene una lista de todos los roles de usuario definidos en el sistema, junto con sus nombres y las capacidades generales asociadas.
    - GET /roles/{id}
        - Método HTTP: GET
        - Ruta exacta: /api/v1/roles/{id} (donde {id} es el identificador único del rol)
        - Descripción: Recupera los detalles específicos y el conjunto de permisos detallados asociados a un rol particular por su ID.
    - POST /roles
        - Método HTTP: POST
        - Ruta exacta: /api/v1/roles
        - Descripción: Permite la creación de una nueva definición de rol en el sistema, especificando su nombre y los permisos que este tendrá sobre las diferentes funcionalidades y datos.
    - PUT /roles/{id}
        - Método HTTP: PUT
        - Ruta exacta: /api/v1/roles/{id}
        - Descripción: Actualiza el nombre o los permisos de un rol existente. Esta es una operación crítica que debe ser manejada con extrema precaución.
    - DELETE /roles/{id}
        - Método HTTP: DELETE
        - Ruta exacta: /api/v1/roles/{id}
        - Descripción: Realiza un "borrado suave" de un rol, marcándolo como inactivo. Los roles inactivos no pueden ser asignados a nuevos usuarios, pero su registro se mantiene para auditoría y referencias históricas.
- 4. Verificación de Permisos por Rol
    - Descripción: La verificación de permisos basados en el rol del usuario es una lógica interna del backend que se ejecuta en el Servidor de Aplicación. Después de que un usuario es autenticado y su rol se consulta de la base de datos ROLES, el backend determina si el usuario tiene permiso para acceder a una funcionalidad o dato específico antes de procesar la solicitud. No se expone como un endpoint RESTful dedicado.

Para Logs de Auditoría

El "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)" es fundamental para la seguridad, registrando cronológicamente toda actividad y cambios en el sistema. Estos logs son inmutables (Write Once, Read Many - WORM).

- 1. POST /logs
    - Método HTTP: POST
    - Ruta exacta: /api/v1/logs
    - Descripción: Este endpoint es utilizado internamente por el backend para registrar de forma automática y asíncrona todas las acciones clave realizadas en el sistema. Esto incluye intentos de login (exitosos o fallidos), modificaciones de datos de pacientes, creación/actualización de citas, subida de documentos, y cualquier otra operación crítica que requiera un rastro auditable.
- 2. GET /logs
    - Método HTTP: GET
    - Ruta exacta: /api/v1/logs

    ◦ Descripción: Obtiene un listado cronológico de los registros de auditoría del sistema. Este endpoint es accesible solo para roles con permisos elevados (como el "Admin Principal"), permitiéndoles revisar el historial de actividades, monitorear la seguridad y rastrear operaciones específicas.

Para Integración con IA

La "Inteligencia y Automatización Integrada" utiliza el modelo Gemma 3 (o similar a Gemini) para potenciar el flujo de trabajo del consultorio. La IA se integra directamente en el backend mediante APIs.

• 1. POST /ai/generate-email
    ◦ Método HTTP: POST
    ◦ Ruta exacta: /api/v1/ai/generate-email
    ◦ Descripción: Envía una solicitud al modelo de IA para generar un borrador de correo electrónico. Se proporcionarían parámetros como el propósito del correo, el destinatario y puntos clave a incluir, y la IA devolvería el texto del borrador.
• 2. POST /ai/summarize
    ◦ Método HTTP: POST
    ◦ Ruta exacta: /api/v1/ai/summarize
    ◦ Descripción: Recibe un bloque de texto (ej., notas de consulta, informes) y utiliza el modelo de IA para generar un resumen conciso de dicho contenido.
• 3. POST /ai/chat
    ◦ Método HTTP: POST
    ◦ Ruta exacta: /api/v1/ai/chat
    ◦ Descripción: Permite enviar mensajes de texto al chat interno del sistema, donde el modelo de IA procesa la entrada y devuelve una respuesta textual. Es utilizado para consultas rápidas o para asistir en la redacción de informes. Cabe destacar que este chat funciona únicamente con texto y no genera archivos.
• 4. POST /ai/generate-template
    ◦ Método HTTP: POST
    ◦ Ruta exacta: /api/v1/ai/generate-template
    ◦ Descripción: Envía una solicitud a la IA para generar diversas plantillas, como hojas membretadas, basándose en la información o el formato requerido.
• 5. POST /ai/create-reminder
    ◦ Método HTTP: POST
    ◦ Ruta exacta: /api/v1/ai/create-reminder
    ◦ Descripción: Utiliza la IA para interpretar una solicitud de usuario y generar un recordatorio dentro del sistema, posiblemente con detalles como fecha, hora y descripción.

Documento: Arq. Backend y APIs

Especificación de Solicitudes y Respuestas

Para asegurar la interoperabilidad y una comunicación fluida y predecible entre el Dashboard de Escritorio (frontend) y el Servidor de Aplicación (backend), es fundamental definir contratos claros para las APIs. Estos contratos especifican la estructura JSON esperada tanto en las solicitudes (Request Body) enviadas al backend como en las respuestas (Response Body) recibidas. Todas las comunicaciones de datos sensibles se realizarán de forma segura a través de HTTPS (TLS 1.3).

1. Estructura JSON para Solicitudes (Request Body)

Las solicitudes al backend generalmente contendrán objetos JSON con los datos necesarios para realizar una operación específica (creación, actualización, etc.). Los ejemplos a continuación ilustran estructuras genéricas:

• Para Autenticación (Ej: POST /api/v1/auth/login)
    ◦ Método: POST
    ◦ Descripción: Envía las credenciales del usuario para iniciar sesión.
    ◦ Estructura JSON:
    ◦ Consideraciones: Las contraseñas se envían como texto plano en el cuerpo de la solicitud, pero la seguridad está garantizada por el cifrado TLS 1.3 en tránsito.
• Para Creación de Recursos (Ej: POST /api/v1/patients)
    ◦ Método: POST
    ◦ Descripción: Crea un nuevo perfil de paciente en el sistema.
    ◦ Estructura JSON:
    ◦ Nota: datos_contacto se almacenaría como JSON en la base de datos.
• Para Actualización de Recursos (Ej: PUT /api/v1/patients/{id})
    ◦ Método: PUT
    ◦ Descripción: Actualiza los datos de un paciente existente.

    ◦ Estructura JSON:
    ◦ Consideraciones: Solo los campos que necesitan ser modificados deben ser incluidos en la solicitud.
• Para Agendamiento de Citas (Ej: POST /api/v1/appointments)
    ◦ Método: POST
    ◦ Descripción: Permite programar una nueva cita.
    ◦ Estructura JSON:
• Para Interacción con IA (Ej: POST /api/v1/ai/generate-email)
    ◦ Método: POST
    ◦ Descripción: Envía datos para que el modelo de IA (Gemma 3 o similar a Gemini) genere un borrador de correo electrónico.
    ◦ Estructura JSON:
    ◦ Nota: Otros endpoints de IA como /ai/summarize o /ai/chat esperarían campos como text_to_summarize o message respectivamente.

2. Estructura JSON para Respuestas (Response Body)

Las respuestas del backend indicarán el resultado de la operación solicitada. Las respuestas de éxito suelen contener los datos del recurso solicitado o modificado.

• Para Recuperación de un Recurso Único (Ej: GET /api/v1/patients/{id})
    ◦ Método: GET
    ◦ Descripción: Devuelve los detalles completos de un paciente específico.
    ◦ Estructura JSON (HTTP 200 OK):
• Para Listado de Recursos con Paginación (Ej: GET /api/v1/patients)
    ◦ Método: GET
    ◦ Descripción: Devuelve una lista de pacientes, con soporte para paginación.
    ◦ Estructura JSON (HTTP 200 OK):
• Para Creación Exitosa (Ej: POST /api/v1/patients)
    ◦ Método: POST
    ◦ Descripción: Confirma la creación exitosa del recurso, incluyendo su identificador único.
    ◦ Estructura JSON (HTTP 201 Created):
• Para Login Exitoso (Ej: POST /api/v1/auth/login)
    ◦ Método: POST
    ◦ Descripción: Devuelve un token de autenticación para futuras solicitudes, junto con información básica del usuario y su rol.
    ◦ Estructura JSON (HTTP 200 OK):
• Para Respuesta de IA (Ej: POST /api/v1/ai/generate-email)
    ◦ Método: POST
    ◦ Descripción: Devuelve el texto generado por el modelo de IA.
    ◦ Estructura JSON (HTTP 200 OK):

3. Manejo de Archivos Grandes (Cloud Storage y URLs)

Es importante destacar que los archivos grandes, como PDFs de historiales, imágenes (ej. RX) y videos cortos, no se enviarán directamente en el cuerpo de las solicitudes o respuestas JSON. En su lugar, se utilizará Google Cloud Storage (equivalente a S3/Blob) para el almacenamiento seguro de estos archivos. Los datos sensibles almacenados en Cloud Storage estarán cifrados en reposo.

• Para Subida de Archivos: La aplicación de escritorio primero enviará el archivo directamente a Cloud Storage. Una vez que el archivo esté en la nube, el backend recibirá una URL (o un identificador) del archivo, que luego se almacenará en la base de datos relacional (Cloud SQL) junto con los metadatos relevantes del paciente o del historial clínico.
• Para Acceso a Archivos: Cuando se necesite visualizar un archivo, el backend proporcionará una URL firmada o un enlace seguro de Cloud Storage en la respuesta JSON. El frontend utilizará esta URL para acceder y mostrar el archivo directamente desde Cloud Storage, evitando cargar el backend con transferencias de grandes volúmenes de datos.

Manejo de Errores

Un manejo de errores estandarizado es vital para la robustez y la facilidad de depuración de la API, proporcionando retroalimentación clara tanto al frontend como a los desarrolladores.

1. Códigos de Estado HTTP Estándar

El backend utilizará códigos de estado HTTP estándar para indicar el resultado de cada solicitud:

• 200 OK: La solicitud fue exitosa. Se utiliza para operaciones GET que devuelven datos, y

para PUT/DELETE que resultaron en una modificación/eliminación exitosa.
• 201 Created: La solicitud POST fue exitosa y resultó en la creación de un nuevo recurso. La respuesta incluirá el recurso recién creado o su identificador.
• 400 Bad Request: La solicitud del cliente es incorrecta, malformada o contiene datos inválidos. Esto incluye errores de validación de entrada.
• 401 Unauthorized: La solicitud no ha sido aplicada porque le falta credenciales de autenticación válidas para el recurso solicitado. Esto ocurre si el token de autenticación está ausente, es inválido o ha expirado. También se usa cuando las credenciales de login son incorrectas.
• 403 Forbidden: La solicitud es válida, pero el usuario autenticado no tiene los permisos necesarios para acceder al recurso o realizar la acción solicitada. El control de acceso basado en roles es fundamental aquí.
• 404 Not Found: El recurso solicitado no existe en el servidor. Esto se aplica si se busca un ID de paciente que no existe.
• 500 Internal Server Error: Un error genérico del servidor que impide completar la solicitud. Esto indica un problema inesperado en el backend, no relacionado directamente con la solicitud del cliente.
2. Estructura JSON Estándar para Mensajes de Error
Todos los errores, especialmente los de tipo 4xx y 5xx, se devolverán con una estructura JSON estandarizada para facilitar su procesamiento por parte del frontend:

```
{
  "error": "Mensaje conciso del error",
  "code": 400, // Código de estado HTTP
  "details": "Mensaje detallado para el desarrollador o información de validación"
}
```

• error: Una descripción breve y legible por el usuario del problema.
• code: El código de estado HTTP relevante.
• details: Información adicional que puede ser útil para la depuración o para mostrar validaciones específicas al usuario (ej. lista de campos inválidos).
3. Ejemplos de Uso de Errores
• Escenario: Solicitud con Campos Obligatorios Vacíos (400 Bad Request)
    ◦ Descripción: El usuario intenta crear un paciente sin proporcionar el nombre.
    ◦ Respuesta JSON:
    ◦ Citas:. Mensajes como "Campo 'Nombre' requerido" o "Formato de email inválido".
• Escenario: Credenciales de Login Incorrectas (401 Unauthorized)
    ◦ Descripción: Un usuario intenta iniciar sesión con un nombre de usuario o contraseña incorrectos.
    ◦ Respuesta JSON:
    ◦ Citas:.
• Escenario: Acceso Denegado por Permisos (403 Forbidden)
    ◦ Descripción: Un usuario con rol de "Asistente" intenta acceder a la funcionalidad de "Gestionar Usuarios".
    ◦ Respuesta JSON:
    ◦ Citas: El sistema impone un control estricto de acceso basado en roles.
• Escenario: Recurso No Encontrado (404 Not Found)
    ◦ Descripción: El frontend solicita los detalles de un paciente con un id que no existe en la base de datos.
    ◦ Respuesta JSON:
    ◦ Citas:. Mensajes como "No se encontraron pacientes".
• Escenario: Error Inesperado del Servidor (500 Internal Server Error)
    ◦ Descripción: Un fallo inesperado ocurre en la lógica del backend (ej. un problema con la base de datos, un error en el código no manejado).
    ◦ Respuesta JSON:
    ◦ Citas:. Mensaje sobre "problemas de conexión o indisponibilidad del servicio".
4. Registro de Errores (Logs de Auditoría)
Es crucial que todos los errores, especialmente los errores 4xx y 5xx, sean registrados en el "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)". Estos logs son inmutables (Write Once, Read Many - WORM), lo que garantiza su integridad y son vitales para la detección de anomalías, la respuesta a incidentes y el cumplimiento normativo. Esto permite un monitoreo proactivo de la seguridad y el rendimiento del sistema.

Implementación de Autenticación y Autorización

Para asegurar la seguridad y el control de acceso en la plataforma de Remex 8, se definirá una estrategia robusta de autenticación y autorización. Dado el modelo de negocio multinivel de Remex 8, que implica la gestión de socios y distribuidores, es fundamental proteger la información y las funcionalidades de la API.

- Estrategia de Autenticación de Usuarios:
  - La autenticación se implementará utilizando tokens web JSON (JWT) [información no proveniente de las fuentes]. Después de que un usuario (cliente o distribuidor) inicie sesión exitosamente proporcionando sus credenciales (nombre de usuario/correo electrónico y contraseña), el backend generará un token JWT. Este token encapsulará información relevante sobre el usuario, como su ID y roles, y será firmado digitalmente para garantizar su integridad y autenticidad [información no proveniente de las fuentes].
  - Este token será enviado al cliente (aplicación web o móvil) y deberá ser incluido en las cabeceras de cada solicitud subsiguiente a la API. Esto permitirá al servidor verificar la identidad del usuario sin necesidad de consultar una base de datos en cada petición, optimizando el rendimiento [información no proveniente de las fuentes].
  - La validez de los tokens se gestionará estableciendo un tiempo de expiración limitado, tras el cual el usuario deberá refrescar su sesión o volver a autenticarse [información no proveniente de las fuentes]. Se podría implementar un mecanismo de refresh token para mejorar la experiencia de usuario, permitiendo renovar el token de acceso sin requerir la reintroducción de credenciales, siempre que el refresh token sea válido [información no proveniente de las fuentes].
- Implementación de Autorización (Control de Acceso Basado en Roles - RBAC):
  - La autorización se basará en un modelo de control de acceso basado en roles (RBAC) [información no proveniente de las fuentes]. Cada usuario en el sistema de Remex 8 (por ejemplo, clientes, distribuidores, administradores) tendrá uno o más roles asignados, que definirán sus permisos dentro de la plataforma [información no proveniente de las fuentes].
  - Antes de procesar cualquier solicitud a un endpoint de la API, el backend verificará el token JWT para extraer los roles del usuario. Luego, se comprobará si el rol del usuario tiene los permisos necesarios para acceder a la funcionalidad o a los datos específicos solicitados [información no proveniente de las fuentes]. Por ejemplo, un distribuidor tendrá acceso a funcionalidades relacionadas con su red y ventas, mientras que un cliente solo podrá acceder a información de productos y realizar compras.
  - Esto garantiza que solo los usuarios con los permisos adecuados puedan realizar ciertas acciones (ej., gestionar la red de afiliados, ver reportes de comisiones, acceder a información de precios de distribuidor) [información no proveniente de las fuentes].

Tecnología del Backend

La elección de la tecnología para el backend es crucial para la escalabilidad, mantenibilidad y eficiencia de la plataforma digital de Remex 8. Considerando que dogma.black se especializa en desarrollo de software y programación, y que se ha mencionado el uso potencial de Modelos de Lenguaje (LLMs) y APIs para automatización, se sugieren las siguientes opciones, las cuales no están explícitamente detalladas en las fuentes de Remex 8:

- Elección del Lenguaje de Programación:
  - Se recomienda Python o Node.js [información no proveniente de las fuentes].
    - Python: Es una excelente opción por su legibilidad, su vasta comunidad y su robusto ecosistema de librerías, especialmente útil si en el futuro se planean integraciones más profundas con análisis de datos o inteligencia artificial (dado el contexto de LLMs mencionados por dogma.black) [4, información no proveniente de las fuentes]. Su versatilidad lo hace apto para el desarrollo rápido y para manejar diversas tareas del backend.
    - Node.js: Ideal para aplicaciones en tiempo real y microservicios, Node.js permite usar JavaScript tanto en el frontend como en el backend, lo que puede agilizar el desarrollo si el equipo ya está familiarizado con JavaScript. Su modelo de E/S no bloqueante es eficiente para aplicaciones con muchas conexiones concurrentes, como una plataforma de redes sociales con alta interacción [1, 2, 4, información no proveniente de las fuentes].
- Elección del Framework de Backend y Justificación:
  - Si se elige Python:
    - Django: Es un framework completo y robusto que incluye muchas funcionalidades "listas para usar", como un ORM (Object-Relational Mapper), un panel de administración y un sistema de autenticación, lo que acelera el desarrollo de aplicaciones complejas y seguras [información no proveniente de las fuentes]. Sería ideal para una plataforma que

gestiona tanto productos como una red de distribuidores con diferentes roles y datos.
        ▪ Flask: Si se prefiere una solución más ligera y modular, Flask es una excelente alternativa. Ofrece mayor flexibilidad y control sobre los componentes a utilizar, siendo ideal para construir APIs RESTful y microservicios, permitiendo escalar funcionalidades específicas según la demanda de Remex 8 [información no proveniente de las fuentes].
    ◦ Si se elige Node.js:
        ▪ Express.js: Es el framework más popular y flexible para Node.js, ideal para construir APIs RESTful de manera rápida y eficiente [información no proveniente de las fuentes]. Su naturaleza unopinionated permite diseñar la arquitectura del backend con gran libertad, integrándose fácilmente con bases de datos y otros servicios necesarios para una plataforma de marketing y ventas multinivel [1, 2, 3, información no proveniente de las fuentes].

Estas elecciones se alinean con la necesidad de construir una infraestructura digital sólida que soporte el crecimiento de Remex 8 y las capacidades de automatización exploradas por dogma.black.

Configuración de Servicios de GCP para el Backend

La infraestructura del backend de Remex 8 se apoyará en Google Cloud Platform (GCP) para proveer la escalabilidad, resiliencia y seguridad necesarias. A continuación, se detalla la configuración de los servicios clave:

• Servidor de Aplicación: Cloud Run
    ◦ Para el despliegue del servidor de aplicación, se recomienda el uso de Cloud Run [información no proveniente de las fuentes]. Esta elección se alinea con la flexibilidad y la capacidad de construir APIs RESTful y microservicios mencionadas en la sección de "Tecnología del Backend" [información no proveniente de las fuentes].
    ◦ Propósito en la Arquitectura: Cloud Run permitirá desplegar los servicios del backend como contenedores sin servidor, lo que significa que la infraestructura subyacente será gestionada automáticamente por GCP [información no proveniente de las fuentes]. Esto facilitará la escalabilidad automática basada en la demanda de tráfico, desde cero hasta miles de solicitudes concurrentes, y optimizará los costos al pagar solo por los recursos consumidos [información no proveniente de las fuentes]. Es ideal para alojar la lógica de negocio de la API, procesar las solicitudes de autenticación/autorización y servir los datos a las aplicaciones cliente.
• API Gateway / Load Balancer: Cloud Load Balancing con integración de Cloud Endpoints o API Gateway
    ◦ Para servir como el punto de entrada seguro y único para todas las solicitudes externas a la API, se configurará un Cloud Load Balancer [información no proveniente de las fuentes]. Este balanceador de carga externo distribuirá de manera eficiente el tráfico hacia las instancias del servidor de aplicación en Cloud Run [información no proveniente de las fuentes].
    ◦ Rol en la Arquitectura: Además de la distribución de tráfico, se podría integrar con Cloud Endpoints o API Gateway [información no proveniente de las fuentes]. Esto proporcionaría capacidades adicionales como la gestión de API, monitoreo, control de acceso basado en claves de API, y la imposición de límites de tasa (rate limiting) para proteger el backend de usos indebidos o ataques de denegación de servicio [información no proveniente de las fuentes]. Garantizará que solo las solicitudes válidas y autorizadas lleguen a los servicios internos.
• Almacenamiento de Archivos: Cloud Storage Buckets
    ◦ Los archivos estáticos y multimedia de la plataforma, como las imágenes y videos de alta calidad utilizados en la estrategia de contenido para redes sociales, y otros documentos necesarios, se almacenarán en Cloud Storage Buckets [información no proveniente de las fuentes].
    ◦ Configuración y Consideraciones: Se configurarán buckets dedicados para diferentes tipos de contenido (ej., uno para imágenes de productos, otro para videos de marketing, otro para documentos internos de distribuidores) [información no proveniente de las fuentes].
    ◦ Seguridad y Acceso: Los buckets se configurarán con permisos de acceso específicos, utilizando políticas de IAM (Identity and Access Management) [información no proveniente de las fuentes]. Los archivos públicos (como imágenes de productos para el sitio web o redes sociales) se harán accesibles públicamente con URLs generadas, mientras que los archivos sensibles (ej., documentos de negocio para distribuidores) tendrán acceso restringido y se servirán a través de URLs firmadas temporalmente o mediante el backend una vez que el usuario haya sido autorizado [información no proveniente de las fuentes].

Se implementará versionamiento para la recuperación de archivos y la protección contra eliminaciones accidentales [información no proveniente de las fuentes].
• Monitoreo y Registro: Cloud Logging y Cloud Monitoring
    ◦ Para la recopilación centralizada y el análisis de registros, se utilizará Cloud Logging [información no proveniente de las fuentes].
    ◦ Configuración: Cloud Logging se configurará automáticamente para recopilar los logs del servidor de aplicación (Cloud Run), así como los logs de otros servicios de GCP como el Load Balancer y Cloud SQL [información no proveniente de las fuentes]. Esto permitirá una visibilidad completa del comportamiento del sistema, facilitando la depuración, la identificación de problemas de seguridad o rendimiento, y la auditoría [información no proveniente de las fuentes].
    ◦ Complementariamente, Cloud Monitoring se utilizará para recopilar métricas de rendimiento (uso de CPU, memoria, latencia de solicitudes, errores) y configurar alertas proactivas sobre el estado y la salud de la aplicación y la infraestructura [información no proveniente de las fuentes].
• Conexión a Base de Datos: Cloud SQL
    ◦ El backend se conectará a la base de datos de la plataforma, que estará alojada en Cloud SQL [información no proveniente de las fuentes].
    ◦ Nota: La configuración detallada de Cloud SQL, incluyendo el tipo de base de datos (PostgreSQL, MySQL), la capacidad y las estrategias de respaldo, se abordará en el Documento 2, "Arq. Base de Datos y Gestión de Datos" [información proveniente de la estructura del documento].
• Consideraciones de Red y Seguridad: VPC y Firewall
    ◦ Para garantizar la comunicación segura y controlada entre los diferentes servicios de GCP, se configurará una Virtual Private Cloud (VPC) [información no proveniente de las fuentes].
    ◦ Firewall Rules: Se implementarán reglas de firewall dentro de la VPC para restringir el tráfico entre los servicios y solo permitir las comunicaciones necesarias [información no proveniente de las fuentes]. Por ejemplo, se permitirá que el servidor de aplicación se conecte a la instancia de Cloud SQL, pero se bloqueará el acceso directo a la base de datos desde internet [información no proveniente de las fuentes]. Esto crea una capa adicional de seguridad y aislamiento para la infraestructura del backend.

Consideraciones Adicionales del Backend

Además de la configuración de servicios fundamental, existen consideraciones críticas para el backend que garantizan la robustez, seguridad y confiabilidad de la plataforma de Remex 8.

• Validación de Datos en el Backend:
    ◦ Es imperativo implementar una rigurosa validación de datos en el backend para todas las entradas recibidas a través de las APIs [información no proveniente de las fuentes, sino una práctica recomendada en arquitectura de software]. Esta validación asegura la integridad de la información, previene el ingreso de datos corruptos o maliciosos, y mantiene la consistencia de la base de datos [información no proveniente de las fuentes]. Esto incluye la verificación de tipos de datos, rangos, formatos, obligatoriedad de campos y la aplicación de reglas de negocio específicas antes de procesar o almacenar cualquier dato. Una validación robusta es la primera línea de defensa contra vulnerabilidades de seguridad y errores de aplicación [información no proveniente de las fuentes].
• Registro de Actividades Clave (Logs de Auditoría):
    ◦ Reafirmando la importancia de lo mencionado previamente en la sección de "Configuración de Servicios de GCP para el Backend" sobre Cloud Logging [información no proveniente de las fuentes], es esencial que el backend genere y recopile logs de auditoría para las actividades clave [información no proveniente de las fuentes]. Estos logs deben registrar acciones significativas como inicios de sesión (exitosos y fallidos), modificaciones de datos de usuarios o productos, transacciones (ventas de productos, registros de nuevos socios), y cualquier intento de acceso no autorizado [información no proveniente de las fuentes]. La configuración de Cloud Logging para recopilar estos logs del servidor de aplicación (Cloud Run) y otros servicios de GCP es crucial [información no proveniente de las fuentes]. Los logs de auditoría son fundamentales para:
        ▪ Seguridad: Permitir la detección temprana de patrones de comportamiento sospechosos o ataques, facilitando la investigación forense en caso de una brecha de seguridad [información no proveniente de las fuentes].
        ▪ Seguimiento y Cumplimiento: Proporcionar un historial inmutable de las operaciones del sistema, lo que es vital para la resolución de problemas, la rendición de cuentas y el cumplimiento de posibles regulaciones futuras [información no proveniente de

las fuentes].

Documento de Diseño de la Base de Datos (Cloud SQL)

Este documento detalla el diseño de la base de datos relacional para el "Ecosistema Digital Inteligente para Cirugía Especial", la cual se alojará en Google Cloud Platform (GCP) utilizando Cloud SQL (MySQL/PostgreSQL). El objetivo es centralizar toda la información crítica del consultorio de manera segura, garantizando la integridad de los datos sensibles mediante cifrado en reposo.

1. Modelo Entidad-Relación (ERD)

El modelo Entidad-Relación (ERD) describe las principales entidades y sus interconexiones dentro del ecosistema digital, sirviendo como la columna vertebral para la gestión de pacientes y la operación diaria del consultorio. Este modelo asegura que la información estructurada, como perfiles de pacientes, historiales clínicos, citas y usuarios, se almacene de forma coherente y segura.

Las entidades principales y sus relaciones son las siguientes:

• PACIENTES y HISTORIAL_CLINICO: La entidad PACIENTES "tiene" una relación de uno a muchos con HISTORIAL_CLINICO, lo que significa que un paciente puede tener múltiples entradas en su historial clínico a lo largo del tiempo.

• HISTORIAL_CLINICO y DOCUMENTOS: La entidad HISTORIAL_CLINICO "contiene" DOCUMENTOS. Esto permite adjuntar archivos digitales como PDFs, imágenes o videos cortos a entradas específicas del historial clínico de un paciente.

• PACIENTES y CITAS: La entidad PACIENTES "agenda" CITAS. Un paciente puede tener múltiples citas programadas con el consultorio.

• CITAS y PROCEDIMIENTOS_CITAS: La entidad CITAS "se relaciona con" PROCEDIMIENTOS_CITAS. Esta tabla intermedia permite asociar una o más citas con procedimientos específicos.

• PROCEDIMIENTOS_CITAS y PROCEDIMIENTOS: La entidad PROCEDIMIENTOS_CITAS "detalla" PROCEDIMIENTOS. Esto vincula los procedimientos realizados con las citas correspondientes.

• USUARIOS y ROLES: La entidad ROLES "asigna" USUARIOS. Cada usuario del sistema tiene un rol definido que determina sus permisos de acceso a la información y funcionalidades.

• USUARIOS y LOGS_AUDITORIA: La entidad USUARIOS "realiza" operaciones que son registradas en LOGS_AUDITORIA. Este componente registra cronológicamente todas las actividades y cambios realizados en el sistema, siendo vital para la seguridad y el seguimiento de operaciones.

2. Especificación Completa de Tablas (Parte 1)

A continuación, se detalla la estructura y las características de las tablas PACIENTES e HISTORIAL_CLINICO, incluyendo sus columnas, tipos de datos, restricciones y relaciones.

Tabla: PACIENTES

Esta tabla centraliza la información demográfica y de contacto de todos los pacientes del consultorio.

Columna
Tipo de Dato
Restricciones
Descripción
id_paciente
INT
PK, NOT NULL, AUTO_INCREMENT
Identificador único y autoincremental para cada paciente.
nombre
VARCHAR(255)
NOT NULL
Nombre(s) del paciente.
apellido
VARCHAR(255)
NOT NULL
Apellido(s) del paciente.
fecha_nac
DATE
NULLABLE
Fecha de nacimiento del paciente.
datos_contacto
JSON
NULLABLE
Almacena información de contacto adicional (ej., teléfono, email, dirección) en formato JSON.

activo
BOOLEAN
NOT NULL, DEFAULT TRUE
Indica si el registro del paciente está activo (TRUE) o ha sido eliminado suavemente/archivado (FALSE).
Tabla: HISTORIAL_CLINICO
Esta tabla almacena el registro cronológico y detallado de cada interacción clínica con un paciente.
Columna
Tipo de Dato
Restricciones
Descripción
id_historial
INT
PK, NOT NULL, AUTO_INCREMENT
Identificador único y autoincremental para cada entrada del historial clínico.
id_paciente
INT
FK a PACIENTES.id_paciente, NOT NULL
Clave foránea que vincula la entrada del historial a un paciente específico.
fecha_consulta
DATE
NOT NULL
Fecha en la que se realizó la consulta o se registró la entrada.
diagnostico
TEXT
NULLABLE
Diagnóstico asociado a la consulta.
notas_medico
TEXT
NULLABLE
Notas detalladas del médico sobre la evolución, procedimientos o tratamientos.
activo
BOOLEAN
NOT NULL, DEFAULT TRUE
Indica si la entrada del historial está activa (TRUE) o ha sido marcada como inactiva/archivada (FALSE).

2. Especificación Completa de Tablas (Parte 2)
Se presenta la estructura detallada de las tablas USUARIOS, ROLES y LOGS_AUDITORIA, cruciales para la seguridad del sistema y el control de acceso basado en roles.
Tabla: USUARIOS
Esta tabla almacena la información de los usuarios que acceden al sistema, incluyendo sus credenciales seguras y el rol asignado, lo cual es fundamental para el control de acceso.
Columna
Tipo de Dato
Restricciones
Descripción
id_usuario
INT
PK
Identificador único para cada usuario del sistema.
nombre
VARCHAR
Nombre de usuario, utilizado para el inicio de sesión.
hash_password
VARCHAR
Contraseña del usuario almacenada de forma segura mediante una función de hash.
id_role
INT
FK a ROLES.id_role
Clave foránea que vincula al usuario con un rol específico, definiendo sus permisos.
activo

BOOLEAN
Indica si la cuenta del usuario está activa (TRUE) o bloqueada/inactiva (FALSE).
Tabla: ROLES
Esta tabla define los diferentes tipos de roles dentro del sistema y los permisos asociados a cada uno, implementando el control de acceso basado en roles.
Columna
Tipo de Dato
Restricciones
Descripción
id_role
INT
PK
Identificador único para cada rol.
nombre_rol
VARCHAR
Nombre descriptivo del rol (ej., "Admin Principal", "Asistente", "Invitado (Solo Lectura)").
permisos
JSON
Estructura JSON que detalla las capacidades y permisos asociados a este rol.
Tabla: LOGS_AUDITORIA
Esta tabla es esencial para la seguridad y el monitoreo, registrando cronológicamente todas las actividades y cambios realizados en el sistema. Los logs son inmutables (Write Once, Read Many - WORM).
Columna
Tipo de Dato
Restricciones
Descripción
id_log
INT
PK
Identificador único y autoincremental para cada entrada del log de auditoría.
tabla_afectada
VARCHAR
Nombre de la tabla de la base de datos que fue afectada por la operación.
id_registro_afectado
INT
ID del registro específico dentro de la tabla_afectada que fue modificado o accedido.
tipo_operacion
VARCHAR
Tipo de operación realizada (ej., 'login_exitoso', 'login_fallido', 'crear', 'actualizar', 'eliminar').
datos_anteriores
JSON
Almacena el estado de los datos del registro afectado antes de la operación (útil para actualizaciones).
datos_nuevos
JSON
Almacena el estado de los datos del registro afectado después de la operación.
fecha_hora
DATETIME
Marca de tiempo de cuándo ocurrió la operación.
id_usuario_autor
INT
FK a USUARIOS.id_usuario
Clave foránea que identifica al usuario que realizó la operación registrada.
Nota sobre restricciones adicionales: La información proporcionada para estas tablas en las fuentes no especifica explícitamente si las columnas deben ser NOT NULL, si son AUTO_INCREMENT (además de la clave primaria), o si tienen valores por defecto. Si esta granularidad es requerida, sería necesario definirla en una etapa posterior del diseño de la base de datos.

3. Estrategia de Indexación

La indexación es un componente crítico para optimizar el rendimiento de la base de datos, especialmente en operaciones de búsqueda, filtrado y ordenamiento de datos. Los índices permiten a la base de datos localizar rápidamente las filas de una tabla sin tener que escanear todas las filas, lo que reduce significativamente los tiempos de respuesta de las consultas y mejora la eficiencia operativa general del sistema.
Basado en los requisitos funcionales del "Ecosistema Digital Inteligente para Cirugía Especial", las siguientes columnas son candidatas clave para la creación de índices, dada su frecuente participación en operaciones de consulta:

- Tabla USUARIOS:
  - nombre: Fundamental para la autenticación de usuarios y búsquedas de personal, ya que el usuario ingresa su "Nombre de Usuario" para iniciar sesión.
  - id_role: Como clave foránea, esta columna será utilizada frecuentemente en uniones (JOINs) con la tabla ROLES para determinar los permisos de acceso del usuario, lo cual es vital para el control de acceso basado en roles.
  - activo: Para filtrar rápidamente los usuarios activos e inactivos del sistema.
- Tabla ROLES:
  - nombre_rol: Si se realizan búsquedas o se necesita filtrar roles por su nombre descriptivo (ej., "Admin Principal", "Asistente").
- Tabla LOGS_AUDITORIA:
  - fecha_hora: Esencial para el monitoreo de seguridad y el seguimiento cronológico de eventos, permitiendo consultas rápidas basadas en rangos de tiempo.
  - tabla_afectada y id_registro_afectado: Combinadas, estas columnas son cruciales para auditar cambios específicos en registros de tablas determinadas. Un índice compuesto o índices individuales en estas columnas facilitarían la recuperación de logs relacionados con una entidad específica.
  - tipo_operacion: Para filtrar registros de auditoría por tipo de evento (ej., 'login_exitoso', 'crear', 'actualizar', 'eliminar').
  - id_usuario_autor: Como clave foránea, se usará para identificar rápidamente todas las acciones realizadas por un usuario específico.
- Candidatos de Indexación en Tablas de Datos del Paciente (Inferido de Requisitos Funcionales): Aunque la especificación detallada de estas tablas no fue solicitada en esta sección, los requisitos funcionales del Dashboard implican la necesidad de indexación en ellas:
  - Tablas de Pacientes: Las columnas utilizadas para la "Búsqueda Avanzada" de pacientes por "nombre, ID, palabra clave en notas" son candidatas principales para índices, incluyendo nombre, apellido, y posiblemente campos relevantes en HISTORIAL_CLINICO.
  - Tablas de Citas: fecha_hora es crucial para la visualización de "Citas del Día" y la gestión de la agenda.
  - Tablas de Documentos: id_registro_afectado (o el FK al paciente/historial) y nombre_archivo para la "Gestión Documental Avanzada".

La implementación de estos índices se realizará para optimizar las consultas más frecuentes y críticas del sistema, garantizando la velocidad y la reactividad necesarias para una experiencia de usuario fluida.

4. Plan de Migración de Datos Existentes

La migración de datos existentes es una etapa crucial en la implementación del "Ecosistema Digital Inteligente para Cirugía Especial", particularmente la carga de los archivos digitales iniciales que el consultorio del Dr. Sánchez García ya posee.
El proceso general de migración de datos se realizará a un alto nivel de la siguiente manera:

1. Identificación y Recopilación de Archivos Digitales: Se identificarán y recopilarán todos los archivos digitales relevantes existentes en el consultorio (PDFs, imágenes como RX, videos cortos, etc.) que deben ser integrados en el nuevo sistema.
2. Extracción de Metadatos: Para cada archivo, se extraerá la información relevante (metadatos) que permitirá vincularlo correctamente en la base de datos relacional. Esto incluirá, pero no se limitará a, el nombre del archivo, tipo de archivo, fecha de subida (o creación original si es posible determinarla), y lo más importante, el identificador del paciente o del registro del historial clínico al que pertenece.
3. Carga de Archivos a Cloud Storage: Los archivos digitales en sí se subirán de forma segura a Google Cloud Storage (equivalente a S3/Blob). Este servicio asegura el almacenamiento centralizado y seguro, con los datos cifrados en reposo.
4. Ingesta de Metadatos en Cloud SQL: Los metadatos extraídos de los archivos, junto con cualquier otra información estructurada existente (ej., datos demográficos de pacientes

que ya tienen historial digital), se insertarán en la Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL). En la tabla DOCUMENTOS, se almacenará la url_almacenamiento que apunta a la ubicación del archivo en Cloud Storage, asegurando el vínculo entre el registro de la base de datos y el archivo físico.
5. Verificación y Validación: Una vez completada la migración, se realizarán procesos de verificación y validación para asegurar la integridad de los datos, la correcta asociación de archivos con los pacientes/historiales, y que todos los archivos sean accesibles desde el sistema.
La migración se enfocará en asegurar la integridad y seguridad de la información desde el primer día.
5. Especificaciones de Seguridad de la Base de Datos
La seguridad de la base de datos es un pilar fundamental del "Ecosistema Digital Inteligente para Cirugía Especial", dada la sensibilidad de la información de los pacientes. Las siguientes especificaciones de seguridad son primordiales para la protección de los datos:
1. Configuración de Usuarios y Permisos de Acceso (Control de Acceso Basado en Roles - RBAC):
    ◦ La base de datos, específicamente la tabla USUARIOS, integra un id_role que se vincula a la tabla ROLES, la cual define los permisos de cada tipo de usuario mediante una estructura JSON. Esto implementa un estricto "Control de Acceso Basado en Roles".
    ◦ Los roles definidos incluyen "Admin Principal" (control total, gestión de usuarios y logs), "Admin Secundario" (todo excepto gestión de usuarios y logs), "Asistente" (gestión de Pacientes, Citas, Documentos, Ver Reportes) e "Invitado (Solo Lectura)" (ver Pacientes, Citas, Reportes).
    ◦ A nivel de la base de datos (Cloud SQL), se configurarán usuarios con el "principio de privilegio mínimo", de modo que el backend (Servidor de Aplicación) solo tenga los permisos necesarios para realizar las operaciones de lectura, escritura y modificación de datos que requiera la lógica de negocio, sin otorgar acceso directo innecesario a los usuarios finales.
2. Cifrado en Reposo para Datos Sensibles:
    ◦ Todos los datos sensibles almacenados en la "Base de Datos Relacional (Cloud SQL)" y en el "Almacenamiento de Archivos (Cloud Storage)" estarán cifrados en reposo.
    ◦ Para las contraseñas, se utilizarán funciones de hash seguras (indicado por el campo hash_password en la tabla USUARIOS) en lugar de almacenar las contraseñas en texto plano. Esto protege contra la exposición de credenciales en caso de una brecha de datos.
3. Plan de Copias de Seguridad y Restauración:
    ◦ Aunque los detalles específicos no se proporcionan, un componente esencial de la seguridad y la continuidad operativa de una base de datos en la nube como Cloud SQL es la implementación de un robusto plan de copias de seguridad (backups) automáticas y la capacidad de restauración (recovery).
    ◦ Esto asegurará la disponibilidad de los datos en caso de fallos, corrupción de datos o desastres, permitiendo la recuperación rápida a un estado anterior y minimizando el tiempo de inactividad del sistema.
    ◦ Las copias de seguridad incluirán tanto la base de datos estructurada como los archivos almacenados en Cloud Storage.
Nota: La información proporcionada en las fuentes no especifica detalles técnicos granulares sobre las configuraciones exactas de indexación (ej., tipos de índices como B-tree, hash), el proceso de migración a nivel de script o herramientas, ni las configuraciones específicas de usuarios y roles dentro de Cloud SQL o la frecuencia y retención del plan de copias de seguridad. Estos detalles se definirán durante la fase de implementación y configuración de la base de datos para alinearse con las mejores prácticas de GCP y los requisitos de rendimiento específicos.

Continuando con el "Documento de Diseño de la Base de Datos (Cloud SQL)", a continuación se presentan las secciones adicionales solicitadas:
8. Estrategia de Indexación
La estrategia de indexación es fundamental para optimizar el rendimiento de la base de datos, especialmente en operaciones de búsqueda y filtrado de datos. Los índices permiten que el motor de la base de datos localice rápidamente filas sin tener que escanear una tabla entera, mejorando drásticamente la velocidad de las consultas. Dada la necesidad de "búsqueda instantánea de pacientes" y el acceso eficiente a historiales clínicos, una correcta indexación es crucial para la "eficiencia operativa" del ecosistema.
Candidatos para la Creación de Índices:

Basándose en los requisitos funcionales del "Ecosistema Digital Inteligente para Cirugía Especial", las siguientes columnas son candidatas primarias para la creación de índices, incluyendo índices únicos para claves primarias y foráneas, y índices no únicos para columnas frecuentemente utilizadas en búsquedas o filtros:

- PACIENTES Tabla:
  - id_paciente: Clave primaria, se indexará automáticamente. Es crucial para la búsqueda de pacientes por ID y la vinculación a otras tablas como HISTORIAL_CLINICO y CITAS.
  - nombre, apellido: Estas columnas serán utilizadas frecuentemente para la "búsqueda instantánea de pacientes" por nombre o apellido. Un índice combinado o índices separados en estas columnas acelerarán las búsquedas.
  - activo: Utilizada para filtrar pacientes activos en el sistema.
- HISTORIAL_CLINICO Tabla:
  - id_historial: Clave primaria, se indexará automáticamente.
  - id_paciente: Clave foránea, esencial para buscar el historial clínico de un paciente específico. La indexación de esta FK optimizará las uniones (JOINs) y las búsquedas por paciente.
  - fecha_consulta: Es probable que se realicen búsquedas y filtros por fecha de consulta para el "Registro Historial Clínico Detallado".
- CITAS Tabla:
  - id_cita: Clave primaria, se indexará automáticamente.
  - id_paciente: Clave foránea, utilizada para vincular citas a pacientes y para filtrar citas por paciente.
  - fecha_hora: Fundamental para el "Módulo de Agenda y Citas", permitiendo búsquedas rápidas por fecha y hora para visualizar las "Citas del Día".
  - estado: Utilizada para filtrar citas por su estado (ej., "Confirmada", "Cancelada", "Atendida").
- USUARIOS Tabla:
  - id_usuario: Clave primaria, se indexará automáticamente.
  - nombre: Utilizada para el login y la identificación de usuarios en los logs de auditoría.
  - id_role: Clave foránea, esencial para determinar los "permisos del usuario" y para la funcionalidad de "Seguridad, Roles y Accesos Rápidos".
- LOGS_AUDITORIA Tabla:
  - id_log: Clave primaria, se indexará automáticamente.
  - fecha_hora: Crucial para la recuperación cronológica de eventos en el "Servicio de Logs de Auditoría".
  - id_usuario_autor: Permite filtrar los logs por el usuario que realizó la acción.
  - tabla_afectada y id_registro_afectado: Permiten rastrear acciones sobre registros específicos. Un índice compuesto o índices individuales pueden ser beneficiosos aquí.

La implementación de estos índices se realizará durante la fase de modelado y diseño de la base de datos para asegurar el rendimiento óptimo del sistema desde el inicio.

9. Plan de Migración de Datos Existentes

La migración de datos existentes es una parte integral de la Fase 1 del proyecto, asegurando la "carga inicial de los documentos que usted y su equipo ya poseen en formato digital" para integrarlos al nuevo sistema. Este proceso busca "eliminar la dispersión de archivos" y centralizar la información.

El plan de migración de archivos digitales se abordará a alto nivel de la siguiente manera:

1. Identificación y Recopilación de Archivos: Se colaborará con el consultorio para identificar y recopilar todos los archivos digitales existentes (PDFs, imágenes de estudios como RX, videos cortos de procedimientos, etc.) que necesiten ser integrados al sistema. Es importante destacar que el proyecto NO incluye la digitalización (escaneo) de archivos físicos existentes; solo se migrarán los archivos que ya estén en formato digital.
2. Extracción y Estructuración de Metadatos:
   - Para cada archivo digital, se extraerán o se solicitarán los metadatos relevantes (ej., nombre del paciente asociado, fecha del documento, tipo de archivo, descripción).
   - Estos metadatos se estructurarán de acuerdo con el esquema de la "Base de Datos Relacional (Cloud SQL)". Por ejemplo, para los documentos de pacientes, se vincularán al id_historial o id_paciente en las tablas DOCUMENTOS o HISTORIAL_CLINICO.
3. Carga Segura a Cloud Storage:
   - Los archivos digitales grandes y pesados se subirán al servicio de "Almacenamiento

de Archivos (Cloud Storage - S3/Blob)" de Google Cloud Platform (GCP).
    ◦ Es crucial que, al igual que la base de datos, estos archivos estén cifrados en reposo en Cloud Storage, garantizando la seguridad de la información sensible del paciente.
4. Vinculación en la Base de Datos (Cloud SQL):
    ◦ Una vez que los archivos estén cargados en Cloud Storage, la "Base de Datos Relacional (Cloud SQL)" no almacenará los archivos directamente. En su lugar, la tabla DOCUMENTOS almacenará la url_almacenamiento (URL de acceso) que apunta a la ubicación segura del archivo en Cloud Storage, junto con otros metadatos como nombre_archivo y tipo_archivo.
    ◦ Esto permite que la aplicación de escritorio, a través del backend, acceda y visualice los documentos asociados al expediente del paciente de forma eficiente.
5. Verificación y Control de Calidad: Se realizarán verificaciones post-migración para asegurar que todos los archivos se hayan cargado correctamente, que los metadatos sean precisos y que las vinculaciones entre la base de datos y Cloud Storage funcionen según lo esperado.
La migración de datos se realizará en la Fase 1 del proyecto, aprovechando herramientas de Google Cloud Storage para este propósito.
10. Especificaciones de Seguridad de la Base de Datos
La seguridad de la base de datos es una "preocupación de seguridad de datos" fundamental que el ecosistema busca abordar. Se integra en múltiples capas de la arquitectura en la nube.
1. Configuración de Usuarios y Permisos de Acceso a la Base de Datos:
    ◦ El sistema implementa un estricto "Control de Acceso Basado en Roles (RBAC)". Al iniciar sesión, cada usuario es asociado a un rol específico ("Administrador", "Admin Principal", "Admin Secundario", "Asistente", "Invitado") que define sus permisos específicos sobre la información y funcionalidades.
    ◦ El backend verifica las credenciales contra la "Base de Datos Relacional (Cloud SQL)" y consulta el Sistema de Roles (ROLES en el modelo de datos) para determinar los permisos del usuario.
    ◦ Para la conectividad de la aplicación al backend y del backend a la base de datos, se aplicará el "principio de privilegio mínimo". Esto significa que el usuario de la base de datos que utiliza el backend para conectarse solo tendrá los permisos necesarios para realizar las operaciones requeridas (lectura, escritura, modificación en tablas específicas), sin privilegios administrativos excesivos.
    ◦ La gestión de usuarios y la asignación de roles se realiza a través del Dashboard, en la sección de "Administración", accesible solo para roles con permisos adecuados como el "Admin Principal".
2. Requisitos de Cifrado en Reposo para los Datos Sensibles:
    ◦ Una medida de seguridad crítica para proteger la información sensible del paciente es el cifrado de datos en reposo.
    ◦ Todos los datos sensibles estarán cifrados en reposo tanto en la "Base de Datos Relacional (Cloud SQL)" como en el "Almacenamiento de Archivos (Cloud Storage)". Esto asegura que, en caso de un acceso no autorizado al almacenamiento subyacente, los datos permanezcan ilegibles.
    ◦ Para las contraseñas de usuario, la base de datos incluye un campo hash_password VARCHAR en la tabla USUARIOS, lo que sugiere el uso de funciones de hash seguras en lugar de almacenar las contraseñas en texto plano. Esto es una práctica estándar esencial contra fugas de datos.
3. Plan de Copias de Seguridad y Restauración:
    ◦ Aunque no se detalla un plan específico de copias de seguridad en todos los documentos, la "Guía de Replicación Técnica" y "Arquitectura Nube" mencionan explícitamente la necesidad de "Implementar copias de seguridad automáticas y recuperación ante desastres" como una recomendación para la base de datos Cloud SQL.
    ◦ Esto es una buena práctica estándar para cualquier base de datos en producción, y la plataforma Google Cloud SQL ofrece capacidades de backup automático y point-in-time recovery, lo que garantiza la "alta disponibilidad" y minimiza el tiempo de inactividad. Un plan de restauración claro será crucial para mitigar el riesgo de pérdida de datos en caso de fallos del sistema o corrupción de datos.

6. Elección y Configuración Inicial de Cloud SQL
La base de datos relacional para el "Ecosistema Digital Inteligente para Cirugía Especial" se hospedará en Google Cloud SQL. Cloud SQL es un servicio de base de datos completamente

gestionado por Google Cloud Platform (GCP) que facilita la configuración, el mantenimiento, la gestión y el escalado de bases de datos relacionales en la nube.
Elección del Motor de Base de Datos (MySQL o PostgreSQL): Las fuentes del proyecto indican que se pueden utilizar sistemas de gestión de bases de datos como MySQL o PostgreSQL. La elección final entre MySQL y PostgreSQL se determinará en una fase posterior, basándose en consideraciones más detalladas, como:
• Requisitos Específicos de Funcionalidad: Si alguna de las bases de datos ofrece características o extensiones únicas que sean particularmente ventajosas para las funcionalidades avanzadas del ecosistema.
• Experiencia del Equipo de Desarrollo: La familiaridad y experiencia del equipo de desarrollo con un motor de base de datos en particular pueden influir en la elección para optimizar la eficiencia y reducir la curva de aprendizaje.
• Patrones de Uso y Rendimiento Esperado: Aunque ambos son robustos, pueden tener diferencias sutiles en el rendimiento para ciertos tipos de cargas de trabajo (ej., operaciones de lectura intensivas vs. escritura intensivas, complejidad de consultas).
• Consideraciones de Licenciamiento y Comunidad: MySQL es de código abierto con diversas opciones de licenciamiento, mientras que PostgreSQL es completamente de código abierto y conocido por su fuerte adherencia a los estándares SQL y su comunidad activa.
Independientemente del motor elegido, ambos son compatibles con Cloud SQL y ofrecen características de alta disponibilidad y recuperación de desastres.
Configuración Inicial de la Instancia en GCP (Alto Nivel): El proceso de configuración inicial de la instancia de Cloud SQL en GCP implica los siguientes pasos clave:
1. Creación de la Instancia:
    ◦ Acceder a la consola de Google Cloud Platform y navegar a la sección de Cloud SQL.
    ◦ Seleccionar la opción para crear una nueva instancia de base de datos.
    ◦ Elegir el motor de base de datos deseado (MySQL o PostgreSQL).
    ◦ Especificar la región donde se desplegará la instancia para asegurar baja latencia con el Servidor de Aplicación (Backend).
    ◦ Definir la configuración de la máquina (tipo de CPU, memoria) y el tipo de almacenamiento (ej., SSD) y su tamaño inicial.
2. Configuración de Red y Conectividad:
    ◦ Configurar la instancia para que opere dentro de la Virtual Private Cloud (VPC) del proyecto. Esto asegura que la base de datos esté en una red aislada y segura, fundamental para la seguridad de los datos.
    ◦ Establecer la conectividad de IP privada para el backend, lo que mejora la seguridad y el rendimiento al mantener el tráfico de la base de datos dentro de la red privada de Google.
    ◦ Configurar las redes autorizadas y reglas de firewall para permitir solo el tráfico desde el Servidor de Aplicación (Backend) y otras IPs de administración necesarias.
3. Creación de Usuarios Iniciales:
    ◦ Crear usuarios de base de datos con contraseñas seguras y el "principio de privilegio mínimo". Es decir, el usuario que utilizará el backend para conectarse a la base de datos solo tendrá los permisos necesarios para las operaciones que realice (lectura, escritura, modificación de tablas específicas), sin privilegios administrativos excesivos.
    ◦ Para la gestión de contraseñas de usuarios en la aplicación, se utilizarán funciones de hash seguras para almacenar hash_password en la tabla USUARIOS.
7. Consideraciones Adicionales
Además de las secciones anteriores, existen otras consideraciones importantes para el diseño y la implementación de la base de datos:
1. Proyección y Gestión del Almacenamiento:
    ◦ El servicio de infraestructura en la nube de Google Cloud incluye una capacidad de hasta 1 Terabyte (1000 GB) mensual como cortesía durante los primeros seis meses.
    ◦ Si bien las fuentes no mencionan explícitamente una proyección de 2TB, cualquier aumento en la capacidad de almacenamiento más allá del 1 Terabyte incluido tendrá un costo adicional.
    ◦ Es fundamental aclarar que este almacenamiento (tanto el inicial de 1TB como cualquier aumento futuro) incluye tanto el espacio utilizado por la Base de Datos Relacional (Cloud SQL) para los datos estructurados como el espacio para los archivos digitales grandes almacenados en Cloud Storage. Se deberá monitorear de cerca el consumo de almacenamiento para gestionar los costos de manera eficiente.
2. Distinción Clara entre Datos Estructurados y Archivos Grandes:
    ◦ Se reafirma que la Base de Datos Relacional (Cloud SQL) se utilizará exclusivamente

para almacenar información estructurada y crítica como perfiles de pacientes, historiales clínicos, citas, usuarios y roles.
    ◦ El Almacenamiento de Archivos (Cloud Storage - S3/Blob) se designará para el almacenamiento centralizado y seguro de archivos digitales pesados, como PDFs, imágenes (ej., RX) y videos cortos asociados a los expedientes de los pacientes. Esta separación optimiza el rendimiento y el costo de almacenamiento, y la base de datos Cloud SQL almacenará únicamente la url_almacenamiento que apunta a la ubicación del archivo en Cloud Storage.
3. Seguridad de los Datos Sensibles (Cifrado):
    ◦ Una consideración primordial es que todos los datos sensibles estarán cifrados en reposo, tanto en la Base de Datos Relacional (Cloud SQL) como en el Almacenamiento de Archivos (Cloud Storage). Esto es una medida de seguridad crítica para proteger la información del paciente contra accesos no autorizados.
    ◦ Las contraseñas de usuario se almacenarán utilizando funciones de hash seguras (campo hash_password) en lugar de texto plano.
4. Escalabilidad y Alta Disponibilidad:
    ◦ La elección de GCP y sus servicios (Cloud SQL, Cloud Storage) proporciona inherentemente escalabilidad (la capacidad de expandirse para manejar un mayor volumen de pacientes y datos sin interrupciones) y alta disponibilidad (minimizando el tiempo de inactividad del sistema). Estas características son fundamentales para la continuidad operativa del consultorio.
5. Requisito de Conectividad:
    ◦ Para un rendimiento óptimo del ecosistema, se requiere una conexión a internet estable de al menos 250 Mbps. La dependencia de la infraestructura en la nube hace que la calidad de la conexión sea crucial.
6. Vinculación del Costo de la Nube al Plan de Mantenimiento:
    ◦ La cortesía de Google Cloud por los primeros seis meses está directamente vinculada a la vigencia del Plan de Mantenimiento y Soporte mensual. Si el cliente decide no continuar con el plan de mantenimiento durante este periodo, la cortesía del servicio en la nube finalizará en ese mismo momento. Esta es una condición importante que la alta dirección debe considerar.
Con estas secciones, el "Documento de Diseño de la Base de Datos (Cloud SQL)" está completo.

Documento de Requisitos Funcionales Detallados: Dashboard de Administración

1. Propósito General del Dashboard de Administración

El Dashboard de Administración es el Núcleo Administrativo Central del "Ecosistema Digital Inteligente para Cirugía Especial". Su propósito fundamental es transformar la gestión operativa del consultorio del Dr. Joel Sánchez García [88, 110-I]. Actúa como el centro de operaciones principal para el personal médico y administrativo, permitiendo la gestión centralizada de toda la información crítica de los pacientes [31-I, 45, 185-1].

Este Dashboard está diseñado para optimizar la gestión de pacientes y mejorar significativamente la eficiencia operativa al reducir los procesos manuales y el riesgo de errores, lo que a su vez libera tiempo valioso para el personal [31-I, 46, 186-1]. Aborda directamente dolores operativos clave como la "saturación y dispersión de información" y la "dificultad en el seguimiento eficiente del paciente" [46, 94, 118-VI, 186-1]. Además, centraliza la inteligencia del sistema para asegurar que la lógica de negocio se ejecute de manera segura y escalable .

2. Estructura de Componentes de la Pantalla

La pantalla del Dashboard de Administración presenta una interfaz clara y funcional, profundamente anclada en el estilo visual "Glassmorphism Oscuro". Este diseño no es solo estético, sino funcional, concebido para ofrecer comodidad visual, una jerarquía de información clara y una experiencia de usuario fluida desde el primer punto de interacción. Se logra mediante elementos que parecen flotar, una luz sutil que emana desde el interior (efecto "Aurora"), y una textura de información nítida.

La pantalla se organiza en tres áreas principales: la Barra de Navegación Lateral (izquierda), el Header Superior (barra superior) y el Área de Contenido Principal (que será detallada posteriormente).

2.1. Barra de Navegación Lateral (Izquierda)

La barra de navegación lateral, ubicada a la izquierda de la interfaz, es el panel principal para la navegación dentro del sistema [48, 53, 154-1.2, 189-4].

- 2.1.1. Identidad de la Aplicación:
  - Logo (Icono circular): En la parte superior de la barra de navegación lateral, se mostrará un icono circular que representa la marca "Cirugía Especial" [275-2.1]. Este logo sirve como un ancla visual constante, reforzando la identidad de la aplicación en todo momento. Su diseño debe ser limpio y moderno, siguiendo la estética general del sistema.
  - Texto "CIRUGIA ESPECIAL": Acompañando al logo, se presentará el texto "CIRUGIA ESPECIAL" [275-2.1]. Este texto, junto con el logo, consolida la identidad visual y textual de la aplicación, siendo siempre visible para el usuario.
- 2.1.2. Navegación Principal (Iconos y Secciones):
  - Las secciones principales del sistema se presentan mediante iconos claros y texto, utilizando un estilo de línea (line-style) con grosor de trazo consistente y terminaciones redondeadas, en el color text-primary (#F5F5F5) para una legibilidad óptima.
  - Inicio (Icono de casa): Al hacer clic, este icono redirige al usuario a la Pantalla: Dashboard Principal (Vista General) [154-1.2, 275-2.1, 189-4]. Funciona como el punto de regreso central para obtener una visión general del estado del consultorio y acceso rápido a las funcionalidades clave.
  - Registros (Icono de documento): Este icono proporciona acceso a la sección de Gestión Integral del Expediente del Paciente [90-1, 114, 155-2.A, 209-4, 252-4]. Permite la creación de nuevos perfiles de pacientes, la búsqueda avanzada de pacientes existentes, y el registro detallado y cronológico del historial clínico, centralizando toda la información crítica del paciente [33, 90-1, 114, 155-2.A, 209-4, 252-4].
  - Colaboradores/Equipo (Icono de usuarios): Este elemento de navegación dirige al usuario a la sección de Gestión de Usuarios dentro del módulo de Administración [37-4, 154-1.2, 158-2.D, 276-2.1]. Es el punto donde los usuarios con permisos adecuados (Administrador y Admin Principal) pueden gestionar cuentas de usuario, asignar y modificar roles, definiendo sus permisos de acceso a la información y funcionalidades [158-2.D, 181-7, 193-8].
  - Reportes (Icono de gráfico de barras): Proporciona acceso a la sección de analíticas y generación de reportes [154-1.2, 276-2.1]. Aquí, el usuario puede visualizar informes de visitas y procedimientos para obtener métricas clave sobre la operación del consultorio [49, 154-1.2, 188-3].
  - Integraciones/Módulos (Icono de cuadrícula): Este icono está destinado a ofrecer acceso a otros módulos o aplicaciones integradas dentro del ecosistema [276-2.1]. Aunque no se detallan módulos específicos adicionales más allá de los principales, este punto de acceso futuro permite la expansión y la integración de nuevas funcionalidades.

• 2.1.3. Sección "Mis Archivos":
    ◦ Esta sección se enfoca en la gestión de documentos digitales asociados a los expedientes de los pacientes [276-2.1].
    ◦ Barra de Búsqueda: Un campo de texto con un icono de lupa incrustado que permite al usuario buscar archivos y carpetas por nombre o palabra clave dentro de esta sección de "Mis Archivos" [276-2.1]. La búsqueda debe ser ágil y ofrecer resultados relevantes a medida que el usuario escribe.
    ◦ Estructura de Carpetas: Se implementará un sistema de árbol de directorios para organizar la información de los pacientes de forma lógica y estructurada [277-2.1]. Esto facilita la navegación y el acceso a los documentos.
    ◦ Carpetas Visibles: Se listarán las siguientes carpetas predefinidas: "General", "Operaciones", "Consultas", "Laboratorio" [277-2.1]. Estas categorías permiten una clasificación inicial de los documentos para una mejor organización.
    ◦ Funcionalidad de Filtrado por Carpeta: Al hacer clic en cualquiera de las carpetas visibles, el contenido del Área de Contenido Principal (específicamente, la sección de la "Base de Datos" o "Registro de Pacientes") debe filtrarse automáticamente para mostrar solo los registros y documentos pertenecientes a esa categoría específica [277-2.1]. Esto proporciona una vista contextual y optimiza la búsqueda de información.
• 2.1.4. Sección "Automatizaciones":
    ◦ Esta sección contiene "placeholders para futuras funcionalidades o para que el usuario configure flujos de trabajo personalizados" [277-2.1]. Es un área designada para la expansión de las capacidades de automatización del sistema.
    ◦ Descripción: Se mostrará una descripción general de lo que se puede lograr con las automatizaciones, posiblemente con ejemplos como "Crear reporte mensual automáticamente" o "Notificar al paciente 24h antes de la consulta" [277-2.1].
    ◦ Primera Manifestación (Asistente Virtual): El Asistente Virtual (modelo de IA Gemma 3) es la primera implementación de esta capacidad de automatización [277-2.1]. Permite la redacción de correos, generación de hojas membretadas, recordatorios, y un chat interno para consultas rápidas, resúmenes de texto o redacción de informes [23, 38, 115, 159-2.E, 210-4, 254-4]. Este chat funciona únicamente con texto y no tiene la capacidad de generar archivos de ningún formato.

2.2. Header Superior (Barra Superior)

El Header Superior, ubicado en la parte superior de la interfaz, proporciona información general y accesos rápidos [53, 154-1.2, 189-4].
• 2.2.1. Accesos Rápidos a Servicios:
    ◦ Se implementará una barra central con iconos que representen accesos rápidos a servicios en la nube integrados [278-2.2, 154-1.2]. Es crucial interpretar estos iconos como accesos directos dentro de la aplicación, no como el dock del sistema operativo del usuario [278-2.2].
    ◦ Servicios Esperados: Esto incluye, pero no se limita a, accesos directos a carpetas específicas en Google Drive y a funcionalidades de Gmail, así como accesos directos para abrir aplicaciones de Office (como Docs/Sheets) [37-4, 154-1.2, 188-3, 210-4, 225-4, 254-4]. Estos widgets están siempre visibles en el dashboard, simplificando el acceso a herramientas esenciales y mejorando la eficiencia .
• 2.2.2. Controles de Usuario (Derecha):
    ◦ Ubicados en el extremo derecho del Header, estos controles proporcionan funcionalidades relacionadas con la interacción y el perfil del usuario [278-2.2].
    ◦ Búsqueda Global (Icono de lupa): Un icono de lupa que, al activarse, permite realizar una búsqueda que abarca toda la aplicación [278-2.2]. Esta búsqueda global debe permitir encontrar información relevante en todos los módulos, incluyendo pacientes, archivos, notas, citas, etc.
    ◦ Notificaciones (Icono de campana): Se incluirá un icono de campana que, al hacer clic, desplegará un centro para alertas y notificaciones del sistema [279-2.2]. Esto es crucial para mantener al personal informado sobre eventos importantes, como citas próximas, nuevos documentos subidos o cambios en el estado de los pacientes.
    ◦ Fecha y Hora: Se mostrará de forma clara la fecha y hora actuales [279-2.2]. Este elemento proporciona un contexto temporal constante para el usuario.
    ◦ Perfil de Usuario (Avatar): Se mostrará el avatar o imagen de perfil del usuario logueado [53, 154-1.2, 189-4, 279-2.2]. Al hacer clic en el avatar, se desplegará un menú contextual con las siguientes opciones:
        ▪ "Mi Perfil": Para acceder y gestionar la información personal del usuario.
        ▪ "Configuración": Para ajustar las preferencias de la aplicación específicas del usuario.

        ▪ "Cerrar Sesión": Para que el usuario pueda finalizar su sesión de forma segura [51, 154-1.2, 188-3].

2.3. Área de Contenido Principal
El Área de Contenido Principal es el espacio de trabajo dinámico del Dashboard, compuesto por diversos módulos o "widgets" que proporcionan información clave y acceso a las funcionalidades operativas del consultorio. Su diseño sigue la estética "Glassmorphism Oscuro", presentando paneles flotantes de "vidrio esmerilado" que organizan y jerarquizan la información, con un fondo dinámico de "Auras" de colores azul y violeta que realza el efecto visual y reduce la fatiga visual.
A. Módulo "Base de Datos"
Este módulo es la funcionalidad central para la gestión de registros de pacientes.
• Título: "Base de Datos".
• Subtítulo: "Registro de pacientes Cirugía Ecensial 2024".
• Barra de Acciones: Ubicada sobre la tabla de datos, proporciona herramientas para la gestión de registros en lote o individual.
    ◦ Eliminar (Icono de papelera): Este botón se activa cuando se seleccionan una o más filas en la tabla. Su funcionalidad principal es realizar un borrado suave (soft delete) de los registros. Esto significa que el registro no se elimina permanentemente de la base de datos, sino que se marca internamente como inactivo o archivado (ej., is_deleted = true), lo que permite su recuperación futura y mantiene la integridad histórica de los datos.
    ◦ Filtros (Icono de filtro): Al hacer clic en este icono, se despliega un panel o un modal que ofrece opciones avanzadas para filtrar los datos mostrados en la tabla. Los filtros pueden incluir criterios como "por fecha", "por estado", "por tipo de consulta", entre otros, permitiendo al usuario refinar rápidamente la visualización de los registros.
    ◦ Exportar: Esta función permite al usuario exportar los datos seleccionados en la tabla o todos los datos filtrados a formatos comunes de hojas de cálculo, como CSV o Excel, facilitando el análisis externo o la generación de reportes.
    ◦ + Nuevo Registro (Botón principal): Al interactuar con este botón, se abre un formulario o un modal dedicado a la creación de un nuevo registro de paciente, donde el usuario puede ingresar toda la información demográfica y de contacto inicial.
• Tabla de Datos: Presenta la información de los pacientes de manera estructurada y de fácil acceso.
    ◦ Checkbox de Selección: Ubicado en la primera columna de cada fila, permite la selección múltiple de registros para aplicar acciones en lote (como Eliminar o Exportar). El encabezado de la columna incluirá un checkbox maestro para seleccionar o deseleccionar todos los registros visibles.
    ◦ ID/Nombre del Paciente (Texto en negrita y enlace): Esta es la columna principal de identificación. El texto en negrita (font-weight: 700) le otorga jerarquía visual, y cada entrada es un enlace directo que lleva a la vista de detalle completa del paciente, donde se puede acceder a su expediente y historial clínico.
    ◦ Columnas de Datos Adicionales (Texto regular): Muestran información relevante y configurable del paciente, como "Fecha de Última Consulta", "Diagnóstico Principal", "Médico Tratante", etc., con texto regular (font-weight: 400) para diferenciarse del nombre principal.
    ◦ Estado (Píldora de color): Una representación visual mediante una "píldora" de color que indica el estado actual del paciente o registro. Los estados definidos son:
        ▪ Active (Verde): Indica que el paciente está en tratamiento o seguimiento activo.
        ▪ Inactive (Rojo/Gris): Señala que el paciente ha sido dado de alta, o que el registro ha sido archivado o está inactivo por algún otro motivo.
    ◦ Menú de Acciones (Icono de tres puntos): Al hacer clic en este icono, se despliega un menú contextual con acciones específicas para el registro de la fila correspondiente. Las opciones incluyen:
        ▪ "Ver Detalles".
        ▪ "Editar Registro".
        ▪ "Ver Historial Clínico Completo".
        ▪ "Archivar".
        ▪ "Eliminar (Suave)".
B. Módulo "Asistente virtual"
Este módulo representa la integración de la inteligencia artificial en el flujo de trabajo del consultorio, potenciada por el modelo Gemma 3 (o similar a Gemini).
• Título: "Asistente virtual".

• Funcionalidad: Permite a los usuarios interactuar con el sistema utilizando lenguaje natural para diversas tareas, optimizando la eficiencia operativa.
    ◦ Interacción Lenguaje Natural: El asistente está diseñado para comprender y procesar consultas formuladas en lenguaje coloquial, permitiendo una interacción intuitiva y conversacional.
    ◦ Búsqueda Inteligente: Los usuarios pueden realizar consultas de búsqueda complejas, como "Hola, estoy buscando un archivo me urge", "Recuerdas el nombre del paciente?", o "Si!! es Laura Vargas", y el asistente utilizará la IA para interpretar la solicitud y localizar la información o los archivos pertinentes en el sistema.
    ◦ Recuperación y Acciones Contextuales: Una vez que el asistente localiza el recurso solicitado (por ejemplo, "Lo encontré"), ofrecerá al usuario acciones contextuales relacionadas, como "te lo envío a tu correo o prefieres descargarlo?", facilitando la gestión de la información.
    ◦ Automatización de Tareas: El asistente virtual está conectado a las APIs del sistema, lo que le permite ejecutar acciones automatizadas en respuesta a las solicitudes del usuario. Esto incluye el envío de correos electrónicos, la generación de borradores de documentos (como hojas membretadas), y la creación de proyectos o recordatorios, liberando tiempo del personal.

C. Módulo "Notas"

Un widget simple diseñado para la gestión de anotaciones rápidas.
• Título: "Notas".
• Funcionalidad: Permite al usuario tomar notas personales y registrar recordatorios rápidos directamente en el dashboard. Por lo general, esta información es privada para el usuario que la creó. Se menciona la posibilidad de una función adicional para "adjuntar nota a paciente", lo que vincularía estas anotaciones directamente al expediente clínico para una referencia más contextual.

D. Módulo "Visor de Archivos Visuales"

Este módulo está dedicado a la visualización de contenido multimedia asociado a los pacientes.
• Funcionalidad: Permite a los usuarios visualizar imágenes y videos directamente relacionados con los expedientes de los pacientes. El contenido del visor se actualiza de forma contextual, mostrando los archivos visuales del paciente que está seleccionado en la tabla principal de la "Base de Datos" o que ha sido resultado de una búsqueda específica. Estos archivos se almacenan de forma segura en Cloud Storage y están cifrados en reposo.
• Componentes:
    ◦ Carrusel de Vistas Previas: Muestra miniaturas de los archivos visuales disponibles, organizados en un formato de carrusel. Las flechas a izquierda y derecha permiten al usuario navegar fácilmente por la galería de archivos.
    ◦ Filtros de Origen/Tipo: Permiten al usuario refinar la visualización del contenido en el carrusel.
        ▪ Cloud Drives: Filtra para mostrar únicamente archivos visuales que provienen de almacenamientos en la nube conectados, como Google Drive o OneDrive.
        ▪ Photos: Muestra exclusivamente imágenes, incluyendo formatos comunes como JPG, PNG, y potencialmente formatos médicos como DICOM.
        ▪ Videos: Muestra únicamente archivos de video, lo que podría incluir grabaciones de cirugías, estudios de endoscopias, o videos educativos asociados al paciente.

2.4. Flujos de Usuario Clave

Esta sección describe los flujos de usuario más importantes dentro del Área de Contenido Principal del Dashboard de Administración, el cual es el "Núcleo Administrativo Central" y el principal entregable de la Fase 1 del proyecto. El Dashboard está diseñado para optimizar la gestión de pacientes y mejorar la eficiencia operativa, abordando la dispersión de información y los procesos manuales.

A. Añadir un Nuevo Paciente

Este flujo permite al personal del consultorio registrar nuevos pacientes en el sistema.
1. Acceso a la Función: El usuario navega a la "Barra de Navegación Lateral" y selecciona la opción "Pacientes", luego hace clic en "Nuevo Paciente". Alternativamente, puede hacer clic directamente en el botón principal "+ Nuevo Registro" ubicado en la "Barra de Acciones" del módulo "Base de Datos".
2. Pantalla de Creación: Se abre un formulario o modal titulado "Creación de Perfil de Paciente".
3. Ingreso de Datos: El usuario debe ingresar los datos demográficos y de contacto del nuevo paciente, como nombre, apellido, fecha de nacimiento, teléfono, correo electrónico y

dirección.
4. Validaciones: Si el usuario intenta guardar el registro sin completar campos obligatorios o con formatos incorrectos (ej., un email inválido), el sistema mostrará mensajes de error claros, como "Campo 'Nombre' requerido" o "Formato de email inválido".
5. Guardado: Una vez que se ingresan los datos válidamente, el usuario hace clic en el botón "Guardar Paciente".
6. Resultado Esperado: La tabla del módulo "Base de Datos" se actualiza automáticamente con el nuevo registro de paciente, el cual aparecerá en estado "Active".
B. Buscar el Historial de un Paciente (Interacción con Asistente Virtual)
Este flujo demuestra la capacidad de búsqueda inteligente del Asistente Virtual integrado con Gemma 3 para acceder rápidamente a la información del paciente.
1. Acceso al Asistente: El usuario se dirige al módulo "Asistente virtual" en el Área de Contenido Principal. Se puede interactuar a través de un icono o una sección dedicada al chat.
2. Interacción de Búsqueda: El usuario utiliza lenguaje natural para realizar una consulta de búsqueda, por ejemplo, "Hola, estoy buscando un archivo me urge", "Recuerdas el nombre del paciente?", o "Si!! es Laura Vargas". El Asistente Virtual está diseñado para comprender y procesar estas consultas complejas utilizando la IA.
3. Recuperación Inteligente: El Asistente Virtual interpreta la solicitud y localiza la información o los archivos pertinentes en el sistema.
4. Respuesta y Acciones Contextuales: El asistente responde al usuario, por ejemplo, diciendo "Lo encontré", y ofrece acciones contextuales relacionadas con el recurso encontrado, como "¿te lo envío a tu correo o prefieres descargarlo?". Para la búsqueda de historial, el asistente provee un enlace directo al historial clínico o a la galería de archivos visuales del paciente. Es importante señalar que este chat funciona únicamente con texto y no tiene la capacidad de generar archivos de ningún formato.
C. Archivar Múltiples Pacientes
Este flujo permite la gestión en lote de registros de pacientes, realizando un borrado suave.
1. Selección de Registros: El usuario selecciona una o más filas en la "Tabla de Datos" del módulo "Base de Datos" utilizando los "Checkbox de Selección" ubicados en la primera columna de cada fila. Un "checkbox maestro" en el encabezado de la columna permite seleccionar o deseleccionar todos los registros visibles.
2. Activación de la Acción: Una vez seleccionadas las filas, el botón "Eliminar" (icono de papelera) en la "Barra de Acciones" se activa.
3. Confirmación: Al hacer clic en "Eliminar", aparece un modal de confirmación preguntando al usuario, por ejemplo, "¿Desea archivar los X registros seleccionados?".
4. Ejecución del Borrado Suave: Tras la confirmación del usuario, el sistema ejecuta un borrado suave (soft delete) de los registros. Esto significa que los registros no se eliminan permanentemente de la base de datos, sino que se marcan internamente como inactivos o archivados (ej., is_deleted = true).
5. Resultado Esperado: Las filas correspondientes desaparecen de la vista activa de la tabla, y su estado interno en la base de datos cambia a "Inactive".
D. Ver Detalles de Paciente
Este flujo permite acceder a una vista completa y detallada de la información de un paciente específico.
1. Acceso a los Detalles: Desde la "Tabla de Datos" del módulo "Base de Datos", el usuario puede hacer clic en el "ID/Nombre del Paciente" (texto en negrita) en la fila correspondiente al paciente deseado. Este elemento actúa como un enlace directo.
2. Pantalla de Detalles: Al hacer clic en el enlace, el sistema redirige al usuario a una vista de detalle completa del paciente, también conocida como la "ficha del paciente".
3. Contenido Esperado: En esta pantalla, el usuario espera ver el perfil completo del paciente con todos sus datos demográficos y de contacto, así como resúmenes de consultas y progreso de tratamiento, y acceso a la gestión documental avanzada, incluyendo la visualización de imágenes y videos relacionados. Esta pantalla también es el punto de partida para acciones adicionales como "Añadir a Historial".
E. Editar Registro de Paciente
Este flujo permite modificar la información existente de un paciente.
1. Acceso a la Edición: Desde la "Tabla de Datos" del módulo "Base de Datos", el usuario hace clic en el "Menú de Acciones" (icono de tres puntos) en la fila del paciente que desea editar.
2. Selección de Acción: Del menú desplegable, el usuario selecciona la opción "Editar Registro".

3. Proceso de Modificación: Se abre un formulario o modal, similar al de creación, pero pre-llenado con los datos actuales del paciente. El usuario realiza las modificaciones necesarias en los campos deseados [inferencia, similar a los formularios de creación o historial clínico que implican entrada de datos].
4. Guardado de Cambios: Una vez realizadas las modificaciones, el usuario hace clic en el botón "Guardar" o "Actualizar" [inferencia]. Los cambios se envían de forma segura al Servidor de Aplicación (Backend) a través de APIs RESTful utilizando HTTPS (TLS 1.3).
5. Registro de Auditoría: El backend procesa la solicitud, actualiza la "Base de Datos Relacional (Cloud SQL)" y registra cronológicamente esta acción de modificación de datos en el "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)", que son inmutables.
6. Confirmación: El sistema muestra una confirmación visual al usuario, indicando que los cambios han sido guardados exitosamente.
F. Ver Historial Clínico Completo
Este flujo permite una visualización detallada y cronológica del historial médico de un paciente.
1. Acceso al Historial: Desde la "Tabla de Datos" del módulo "Base de Datos", el usuario hace clic en el "Menú de Acciones" (icono de tres puntos) en la fila del paciente deseado. Alternativamente, desde la tabla "Pacientes Recientes" en el Área Central del Dashboard, puede hacer clic en la opción "Ver Historial".
2. Selección de Acción: Del menú desplegable, el usuario selecciona la opción "Ver Historial Clínico Completo".
3. Contenido Esperado: Se presenta una vista que contiene el registro cronológico y detallado de cada consulta, diagnóstico, notas de evolución, procedimientos y tratamientos del paciente. Este historial también permite la capacidad de adjuntar documentos relevantes, y se espera que se integre con el "Visor de Archivos Visuales" para mostrar imágenes y videos contextuales relacionados con el paciente.
G. Archivar un Solo Paciente
Este flujo permite archivar individualmente un registro de paciente mediante un borrado suave.
1. Acceso a la Acción: Desde la "Tabla de Datos" del módulo "Base de Datos", el usuario hace clic en el "Menú de Acciones" (icono de tres puntos) en la fila del paciente que desea archivar.
2. Selección de Acción: Del menú desplegable, el usuario selecciona la opción "Archivar". (Nota: La opción "Eliminar (Suave)" también está disponible y realiza la misma función de borrado lógico).
3. Confirmación (Implícito): Aunque no se especifica explícitamente para el archivado individual, se esperaría un modal de confirmación similar al de múltiples pacientes para prevenir acciones accidentales.
4. Ejecución del Borrado Suave: El registro del paciente se marca internamente como inactivo o archivado (is_deleted = true) en la base de datos.
5. Resultado Esperado: El estado del paciente en la tabla puede cambiar visualmente a "Inactive" (una píldora de color rojo/gris) o el registro puede desaparecer de la vista activa, dependiendo de los filtros aplicados.

2.4. Flujos de Usuario Clave (Continuación)
H. Eliminar Suave un Solo Paciente
Este flujo permite archivar individualmente un registro de paciente, realizando un borrado lógico o "suave" para mantener la integridad de los datos históricos.
1. Acceso a la Acción: Desde la "Tabla de Datos" del módulo "Base de Datos", el usuario hace clic en el "Menú de Acciones" (icono de tres puntos) ubicado en la fila del paciente que desea archivar.
2. Selección de Acción: Del menú desplegable que aparece, el usuario selecciona la opción "Archivar". Es importante notar que la opción "Eliminar (Suave)" también está disponible y realiza la misma función de borrado lógico.
3. Confirmación (Implícito): Aunque no se especifica explícitamente para el archivado individual, por buena práctica de diseño de UI/UX y para prevenir acciones accidentales, se esperaría un modal de confirmación similar al de archivado múltiple, preguntando si desea proceder con el archivado del registro.
4. Ejecución del Borrado Suave: El sistema marca el registro del paciente internamente como inactivo o archivado (ej., is_deleted = true) en la "Base de Datos Relacional (Cloud SQL)". Esto significa que los datos no se eliminan permanentemente.
5. Resultado Esperado: La fila correspondiente al paciente desaparecerá de la vista activa de la tabla, o su estado visual en la columna "Estado" cambiará a "Inactive" (una píldora

de color rojo/gris), dependiendo de los filtros aplicados.
I. Filtrar Tabla de Datos
Este flujo permite al usuario refinar la información visible en la "Tabla de Datos" para encontrar rápidamente registros específicos.
1. Acceso a la Función: El usuario localiza el "Icono de filtro" en la "Barra de Acciones" del módulo "Base de Datos" y hace clic sobre él.
2. Opciones de Filtro: Al hacer clic, se despliega un panel o modal que presenta "opciones avanzadas" para filtrar la tabla.
3. Aplicación de Filtros: El usuario puede aplicar uno o más filtros basándose en diversas categorías, como "por fecha", "por estado" (ej., Activo, Inactivo), o "por tipo de consulta".
4. Resultado Esperado: La "Tabla de Datos" se actualiza dinámicamente, mostrando solo los registros que cumplen con los criterios de filtro seleccionados, lo que permite una visualización más enfocada y eficiente.
J. Exportar Datos de la Tabla
Este flujo proporciona la capacidad de extraer los datos de la "Tabla de Datos" para su uso externo o para fines de análisis.
1. Acceso a la Función: El usuario localiza el botón "Exportar" en la "Barra de Acciones" del módulo "Base de Datos" y hace clic sobre él.
2. Selección de Datos: El sistema permite al usuario exportar los "datos seleccionados" (si se han marcado filas específicas con los "Checkbox de Selección") o "todos los datos filtrados" (si se ha aplicado algún filtro).
3. Selección de Formato: El usuario puede elegir el formato de exportación deseado, como "CSV" o "Excel".
4. Resultado Esperado: Se inicia la descarga de un archivo que contiene los datos de la tabla en el formato seleccionado, facilitando su manipulación y análisis fuera del sistema.
K. Interactuar con Asistente Virtual para Tareas Específicas
Este flujo resalta la capacidad de la IA (Gemma 3) para automatizar tareas y proporcionar asistencia contextualizada, mejorando la eficiencia operativa.
1. Acceso a la Interacción: El usuario se dirige al módulo "Asistente virtual" en el Área de Contenido Principal del Dashboard. Esto se logra a través de un "icono o sección dedicada al chat".
2. Ejemplos de Interacciones y Resultados Esperados:
    ◦ Búsqueda Inteligente: El usuario puede preguntar por archivos o información específica usando lenguaje natural (ej., "Hola, estoy buscando un archivo me urge", "Recuerdas el nombre del paciente?", "Si!! es Laura Vargas"). El asistente responde localizando el recurso ("Lo encontré") y ofreciendo acciones contextuales (ej., "¿te lo envío a tu correo o prefieres descargarlo?").
    ◦ Redacción de Contenido: El usuario puede solicitar la redacción automatizada de "correos electrónicos", la "generación de hojas membretadas" o la creación de "recordatorios". La IA puede también generar "resúmenes de texto" o facilitar la "redacción de informes" a través del chat interno.
    ◦ Asistencia para Desarrollo/Capacitación: La IA puede generar "texto de ejemplo" (lorem ipsum más inteligente) para prototipos de UI, sugerir "copys" para botones y etiquetas si se entrena con el tono deseado, o generar "scripts SQL básicos" a partir de descripciones en lenguaje natural. Adicionalmente, puede colaborar en la generación de "preguntas frecuentes (FAQs)" a partir de la documentación técnica para el manual de usuario, e incluso crear un "chatbot de ayuda inicial" para la capacitación del personal.
3. Limitación: Es crucial señalar que este chat funciona "únicamente con texto" y "no tiene la capacidad de generar archivos de ningún formato".
L. Tomar y Guardar Notas en el Widget "Notas"
Este flujo permite al personal registrar información personal y recordatorios rápidos directamente en el Dashboard.
1. Acceso al Widget: El usuario localiza el widget titulado "Notas" en el Área de Contenido Principal del Dashboard.
2. Ingreso de Notas: El usuario interactúa con el área de texto del widget para escribir sus notas o recordatorios.
3. Guardado: Aunque no se especifica un botón explícito de "Guardar" para este widget, se esperaría que las notas se guarden automáticamente o al perder el foco del campo de texto, manteniendo la simplicidad de un "widget simple".
4. Resultado Esperado: Las notas ingresadas se almacenan y son visibles para el usuario en futuras sesiones. Esta información es, por lo general, "privada para el usuario y no está

directamente ligada a un paciente", a menos que se implemente una función específica para adjuntar una nota a un paciente.
M. Navegar y Filtrar Archivos en el "Visor de Archivos Visuales"
Este flujo proporciona una forma eficiente de visualizar y organizar los documentos multimedia asociados a los pacientes.
1. Acceso al Visor: El usuario se dirige al módulo "Visor de archivos visuales" en el Área de Contenido Principal del Dashboard.
2. Navegación por Carrusel: Se presenta un "Carrusel de Vistas Previas" que muestra miniaturas de los archivos. El usuario puede navegar a través de la galería utilizando las "flechas a izquierda y derecha".
3. Filtros de Contenido: El usuario puede aplicar filtros para organizar los archivos según su origen o tipo:
    ◦ Cloud Drives: Muestra solo archivos provenientes de almacenamientos en la nube conectados (ej., Google Drive, OneDrive).
    ◦ Photos: Muestra únicamente imágenes (ej., JPG, PNG, DICOM).
    ◦ Videos: Muestra únicamente archivos de video (ej., grabaciones de cirugías, endoscopias).
4. Actualización Contextual: El contenido del visor se actualiza contextualmente según el paciente seleccionado en la tabla principal "Base de Datos" o la búsqueda realizada, asegurando que se muestren los archivos relevantes para el caso en cuestión.
5. Resultado Esperado: El usuario puede visualizar de forma organizada y eficiente todas las imágenes y videos relacionados con los expedientes de los pacientes.
N. Log In de Usuario
Este flujo es el punto de entrada fundamental al "Ecosistema Digital Inteligente para Cirugía Especial", específicamente al "Núcleo Administrativo Central" o Dashboard de Escritorio. Su propósito principal es autenticar al usuario y aplicar el control de acceso basado en roles.
1. Acceso a la Aplicación: El usuario inicia la aplicación de escritorio del "Núcleo Administrativo Central".
2. Pantalla de Login: Se presenta la "Pantalla: Login de Usuario", mostrando campos para "Nombre de Usuario" y "Contraseña".
3. Ingreso de Credenciales: El usuario ingresa su "Nombre de Usuario" y "Contraseña" en los campos designados.
4. Intento de Inicio de Sesión: El usuario hace clic en el botón "Iniciar Sesión".
5. Comunicación con el Backend: Las credenciales se envían de forma segura al Servidor de Aplicación (Backend) alojado en Google Cloud Platform (GCP). Esta comunicación utiliza HTTPS (TLS 1.3) para cifrar los datos en tránsito, protegiendo las credenciales de usuario de ser interceptadas.
6. Autenticación y Autorización en Backend:
    ◦ El backend (construido con Python/Flask/Django o Node.js/Express) recibe las credenciales.
    ◦ Verifica las credenciales contra la "Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL)". Los datos sensibles como contraseñas se almacenan cifrados en reposo en la base de datos (con un campo hash_password que sugiere el uso de funciones de hash).
    ◦ Una vez autenticado, el backend consulta el "Sistema de Roles" (ROLES en el modelo de datos) para determinar los permisos del usuario (ej., "Admin Principal", "Admin Secundario", "Asistente", "Invitado (Solo Lectura)").
7. Registro de Auditoría: El "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)" registra cronológicamente el intento de login (exitoso o fallido), la actividad del usuario y los cambios realizados en el sistema. Estos logs son inmutables (Write Once, Read Many - WORM), asegurando su integridad.
8. Redirección al Dashboard o Manejo de Errores:
    ◦ Login Exitoso: El usuario es redireccionado sin problemas a la "Pantalla: Dashboard Principal (Vista General)". La cabecera del dashboard mostrará el usuario logueado y su rol, y la barra de navegación lateral presentará las secciones a las que tiene acceso según sus permisos.
    ◦ Credenciales Incorrectas: La aplicación permanece en la pantalla de Login y mostrará un mensaje de error claro y conciso, por ejemplo: "Usuario o contraseña incorrectos".
    ◦ Campos Vacíos/Formato Inválido: Se esperaría una validación similar a otros formularios con mensajes de "Campo 'Nombre de Usuario' requerido" o "Campo 'Contraseña' requerido".
    ◦ Problemas de Conectividad: Un mensaje de error general sobre problemas de conexión o indisponibilidad del servicio. Se recomienda una conexión a internet estable de al menos

250 Mbps para un rendimiento óptimo.
O. Log Out de Usuario
Este flujo permite al usuario finalizar su sesión de forma segura en el Dashboard de Escritorio.
1. Acceso a la Acción: El usuario se dirige a la "Cabecera" (Header Superior) del Dashboard Principal.
2. Selección de "Cerrar Sesión": En la cabecera, el usuario localiza y hace clic en la opción "Cerrar Sesión".
3. Proceso de Cierre de Sesión: El sistema procede a finalizar la sesión del usuario. (Implícitamente, esto implicaría la invalidación de cualquier token de sesión en el backend y la limpieza de datos de sesión en el frontend).
4. Resultado Esperado: El usuario es automáticamente redirigido a la "Pantalla: Login de Usuario".
P. Acceder a Mi Perfil
Este flujo permite al usuario acceder a su información de perfil dentro del sistema.
1. Acceso a la Función: Desde la "Cabecera" del Dashboard Principal, el usuario hace clic en el "Perfil de Usuario (Avatar)".
2. Selección de Opción: Al hacer clic en el avatar, se despliega un menú contextual con varias opciones. El usuario selecciona la opción "Mi Perfil".
3. Resultado Esperado: El sistema redirige al usuario a una pantalla o modal donde puede visualizar su información de perfil.
Q. Acceder a Configuración
Este flujo permite al usuario acceder a las configuraciones del sistema o personales disponibles.
1. Acceso a la Función: Desde la "Cabecera" del Dashboard Principal, el usuario hace clic en el "Perfil de Usuario (Avatar)".
2. Selección de Opción: Del menú desplegable que aparece, el usuario selecciona la opción "Configuración".
3. Resultado Esperado: El sistema redirige al usuario a una pantalla o modal donde puede acceder y modificar las configuraciones pertinentes.

3.0. Requisitos No Funcionales
Los requisitos no funcionales definen las cualidades y características del sistema que no están directamente relacionadas con la funcionalidad específica, sino con el rendimiento, la usabilidad, la seguridad, la fiabilidad y la escalabilidad. Son cruciales para garantizar que el sistema cumpla con las expectativas de calidad y eficiencia operativa del consultorio del Dr. Sánchez García.
3.1. Rendimiento
El sistema debe operar de manera fluida y eficiente, incluso bajo cargas de datos significativas.
• Requisito: El Dashboard y los servicios de backend deben mantener tiempos de respuesta óptimos (inferiores a 2 segundos para la mayoría de las operaciones críticas y menos de 5 segundos para operaciones complejas como búsquedas o exportaciones de grandes volúmenes de datos) con un volumen de datos de hasta 2 Terabytes (TB). Gran parte de este volumen estará compuesta por archivos visuales y documentos (imágenes, videos, PDF) almacenados en Cloud Storage, mientras que los metadatos y la información estructurada se gestionarán en la Base de Datos Relacional (Cloud SQL). Se recomienda una conexión a internet estable de al menos 250 Mbps para un rendimiento óptimo del ecosistema, dada su dependencia de la infraestructura en la nube.
3.2. Usabilidad
El sistema debe ser intuitivo y fácil de usar, minimizando la curva de aprendizaje para el personal.
• Requisito: El Dashboard debe ofrecer una experiencia de usuario "excepcionalmente intuitiva", reflejando la calidad y el prestigio de la práctica médica. Esto implica una coherencia visual integral a través del estilo "Glassmorphism Oscuro", diseñado para ofrecer "comodidad visual y una jerarquía de información clara". La interfaz debe proporcionar retroalimentación clara a las interacciones del usuario (ej., aumento de brillo o ligero hundimiento al pulsar un botón), y el diseño debe asegurar una "legibilidad óptima" con texto e iconos nítidos y brillantes. El diseño busca mitigar la "resistencia o dificultad con la tecnología" por parte del personal.
3.3. Seguridad
La protección de la información sensible del paciente y el acceso controlado son

requisitos primordiales.

• Requisito: El sistema debe garantizar la confidencialidad, integridad y disponibilidad de todos los datos sensibles del paciente, abordando directamente las "preocupaciones de seguridad de datos". Esto se logra mediante:

    ◦ Control de Acceso Basado en Roles (RBAC): Cada usuario, al iniciar sesión, es asociado a un rol específico ("Admin Principal", "Admin Secundario", "Asistente", "Invitado") que define sus permisos sobre la información y las funcionalidades del Dashboard.

    ◦ Cifrado de Datos: Toda la comunicación entre la aplicación de escritorio y la infraestructura en la nube debe cifrarse mediante HTTPS (TLS 1.3) para proteger los datos en tránsito. Además, todos los datos sensibles del paciente y las contraseñas (como hash) deben almacenarse cifrados en reposo tanto en la "Base de Datos Relacional (Cloud SQL)" como en el "Almacenamiento de Archivos (Cloud Storage)".

    ◦ Auditoría Integral: Cada acción relevante y cada intento de inicio de sesión (exitoso o fallido) debe registrarse cronológicamente en un "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)". Estos logs deben ser inmutables (Write Once, Read Many - WORM) para asegurar su integridad y facilitar la detección de anomalías.

3.4. Fiabilidad

El sistema debe ser robusto y capaz de funcionar de manera consistente y sin interrupciones.

• Requisito: El Dashboard y su infraestructura subyacente en la nube deben operar de forma estable y continua, minimizando el tiempo de inactividad. La arquitectura de nube está diseñada para alta disponibilidad. El plan de mantenimiento y soporte mensual, que incluye "mantenimiento proactivo" y "soporte técnico 24/7", garantiza el "óptimo funcionamiento" y la "evolución continua" de la aplicación, reforzando la fiabilidad del ecosistema.

3.5. Escalabilidad

El sistema debe ser capaz de crecer y adaptarse a futuras demandas de volumen de datos y usuarios sin comprometer el rendimiento.

• Requisito: El sistema debe poder manejar un crecimiento futuro en el volumen de pacientes, el número de usuarios y la cantidad de datos almacenados (incluyendo archivos multimedia) sin degradación del rendimiento. La elección de Google Cloud Platform (GCP) como proveedor de nube principal proporciona una "escalabilidad inherente" que permite al sistema expandirse para manejar una mayor demanda. El diseño del backend, potencialmente utilizando arquitecturas de microservicios, y la capacidad de adquirir "almacenamiento adicional" más allá del terabyte inicial, aseguran que la plataforma pueda adaptarse a las necesidades futuras del consultorio.

Especificación Técnica del Frontend del Dashboard: Núcleo Administrativo Central

1. Introducción y Propósito

El presente documento tiene como objetivo principal detallar las especificaciones técnicas para la construcción del Frontend del "Dashboard de Escritorio", componente central del "Ecosistema Digital Inteligente para Cirugía Especial". Servirá como la guía definitiva para los equipos de diseño y desarrollo, asegurando una implementación coherente, de alta calidad y sin ambigüedades.

El Dashboard de Escritorio es el "Núcleo Administrativo Central" del ecosistema, diseñado para ser el corazón operativo del consultorio del Dr. Joel Sánchez García. Su propósito fundamental es centralizar toda la información crítica del consultorio en un solo lugar seguro y accesible, ofreciendo una visión general del estado actual y un acceso rápido a las funcionalidades clave. Su objetivo es optimizar la gestión de pacientes, mejorar la eficiencia operativa al reducir los procesos manuales y el riesgo de errores, y liberar tiempo valioso para el personal. Además, está concebido para ser un entorno digital potente, seguro y excepcionalmente intuitivo, reflejando la calidad y el prestigio de la práctica médica del Dr. Sánchez García.

2. Elección del Framework Frontend

Para el desarrollo del Frontend del Dashboard, se recomienda la utilización de un framework moderno y robusto, siendo React o Vue.js las opciones preferidas y viables. Estos frameworks son ampliamente adoptados en la industria debido a su popularidad, amplios ecosistemas de herramientas y librerías, y su eficiencia en la creación de interfaces de usuario complejas y reactivas. Permiten un desarrollo modular, facilitan la gestión del estado y ofrecen un rendimiento óptimo para aplicaciones dinámicas.

El Dashboard se empaquetará como una aplicación de escritorio utilizando Electron.js. Electron.js permite construir aplicaciones de escritorio nativas (compatibles con Windows y macOS) utilizando tecnologías web estándares como HTML, CSS y JavaScript. Esta elección es estratégica, ya que aprovecha la experiencia en desarrollo web para crear una aplicación de escritorio, lo que agiliza el proceso y garantiza una experiencia de usuario familiar y de alta calidad. La integración con el framework frontend elegido (React/Vue.js) es fluida, permitiendo que la lógica de negocio y la interfaz se desarrollen de manera cohesiva para el entorno de escritorio.

3. Arquitectura y Estructura de la Aplicación

3.1. Diagrama de Arquitectura de Alto Nivel del Frontend y su Interacción

El Frontend del Dashboard de Escritorio actúa como la capa de presentación que permite la interacción del usuario con la lógica de negocio y los datos alojados en la nube. Se comunica de forma segura con el Backend a través de APIs RESTful.

```
graph TD
    A[Aplicación de Escritorio (Frontend)] -- HTTPS (TLS 1.3) --> B(API Gateway / Load Balancer - GCP)
    B --> C[Servidor de Aplicación (Backend)]
    C -- Lógica de Negocio --> D[Base de Datos Relacional (Cloud SQL - GCP)]
    C -- Almacenamiento --> E[Almacenamiento de Archivos (Cloud Storage - GCP)]
    C -- Logs --> F[Servicio de Logs de Auditoría (Cloud Logging - GCP)]
    C -- Integración IA --> G[Modelo de IA (Gemma 3)]

    D -- Cifrado en Reposo --> D
    E -- Cifrado en Reposo --> E
    G -- Consultas/Generación --> C
```

Explicación del Flujo:

1. El Usuario inicia la Aplicación de Escritorio (Frontend).
2. La aplicación se conecta de forma segura a la infraestructura en la nube de Google Cloud Platform (GCP).
3. Todas las solicitudes del Frontend hacia el Backend pasan por el API Gateway / Load Balancer, que actúa como el punto de entrada seguro y distribuye el tráfico eficientemente.
4. La comunicación entre el Frontend y la nube se realiza a través de HTTPS (TLS 1.3) para cifrar los datos en tránsito, protegiendo la información sensible como las credenciales del usuario.
5. El Servidor de Aplicación (Backend) procesa las solicitudes, interactúa con la Base de Datos Relacional (Cloud SQL) para datos estructurados (pacientes, citas, usuarios) y con el Almacenamiento de Archivos (Cloud Storage) para documentos digitales (PDFs, imágenes, videos). Los datos sensibles en ambos servicios de almacenamiento están cifrados en

reposo.
6. Cada acción del usuario y cada intento de login (exitoso o fallido) se registra cronológicamente en el Servicio de Logs de Auditoría (Cloud Logging / Audit Logs), los cuales son inmutables.
7. El Backend también integra el modelo de Inteligencia Artificial Gemma 3 mediante APIs para potenciar funcionalidades de automatización y el chat interno.

3.2. Estructura Detallada de Directorios y Carpetas del Proyecto Frontend

Se propone la siguiente estructura de directorios, basada en principios de modularidad y separación de responsabilidades para facilitar la mantenibilidad, escalabilidad y colaboración del equipo:

```
src/
├── assets/                  # Contiene recursos estáticos como imágenes, iconos, fuentes
│   ├── images/
│   ├── icons/               # Iconos de estilo de línea (line-style) con grosor de trazo consistente [49-52]
│   └── fonts/               # Fuente Inter (sans-serif geométrica) [45, 50-54]
├── components/              # Componentes UI reutilizables y atómicos
│   ├── buttons/
│   ├── forms/
│   ├── inputs/
│   ├── layout/              # Componentes de layout (Header, Sidebar, MainContent) [52, 55-57]
│   └── common/              # Otros componentes generales
├── views/ (o pages/)        # Contiene las vistas principales o "pantallas" de la aplicación
│   ├── Auth/                # Pantalla de Login [3, 7, 28, 58-60]
│   ├── Dashboard/           # Pantalla Principal del Dashboard [4, 7, 8, 31, 36, 58]
│   ├── Patients/            # Gestión Integral del Expediente del Paciente [2, 61-66]
│   ├── Appointments/        # Módulo de Agenda y Citas [67-72]
│   ├── Documents/           # Gestión Documental Avanzada [2, 63, 72-75]
│   ├── Administration/      # Seguridad, Roles y Accesos Rápidos (Gestionar Usuarios, Ver Logs de Auditoría) [68, 71, 76-79]
│   ├── Reports/             # Módulo de Reportes [7, 58, 80-82]
│   └── AIAutomation/        # Inteligencia y Automatización Integrada (Chat Interno) [68, 71, 79, 83-85]
├── services/ (o api/)       # Lógica para interactuar con el Backend (APIs)
│   ├── auth.js              # Servicios de autenticación (Login, Logout) [86, 87]
│   ├── patients.js          # Servicios para la gestión de pacientes [88, 89]
│   ├── appointments.js      # Servicios para la gestión de citas [88, 89]
│   ├── documents.js         # Servicios para la gestión de documentos [88, 89]
│   ├── users.js             # Servicios para la gestión de usuarios y roles [88, 89]
│   └── ai.js                # Servicios para la interacción con la IA (Gemma 3) [88, 89]
├── styles/                  # Estilos globales y específicos
│   ├── globals.css          # Estilos CSS globales, incluyendo las Propiedades Personalizadas de CSS [49-51, 90]
│   ├── themes.css           # Definiciones de temas (Glassmorphism Oscuro) [91, 92]
│   ├── components.css       # Estilos para componentes reutilizables
│   └── views.css            # Estilos específicos para vistas/pantallas
├── utils/                   # Funciones de utilidad (ej. formateo de fechas, validación de formularios)
├── hooks/                   # Custom Hooks (si se usa React) para lógica reutilizable con estado
├── store/ (o context/)      # Gestión de estado global de la aplicación (ej. información del usuario logueado, roles)
└── App.js (o main.js)       # Punto de entrada principal de la aplicación
```

3.3. Descripción de la Lógica de Organización

- Modularidad y Reusabilidad: La aplicación se estructurará en módulos y componentes, lo que permite reutilizar el código en diferentes partes de la aplicación y facilita el desarrollo paralelo. Los componentes se diseñarán de forma atómica, encapsulando su propia lógica y estilos.
- Separación de Responsabilidades:
    - Lógica de UI en Componentes/Vistas: Los componentes y las vistas (carpetas components y views) serán responsables de la presentación visual y la interacción directa

con el usuario.
    ◦ Lógica de Negocio y Llamadas a API en Servicios: Toda la lógica de comunicación con el backend, incluyendo las llamadas a las APIs RESTful y el manejo de la autenticación/autorización, residirá en la carpeta services (o api). Esto asegura que la lógica de presentación esté desacoplada de la lógica de datos.
    ◦ Estilos Separados: Los estilos se gestionarán de forma centralizada en la carpeta styles, utilizando CSS Variables para mantener la coherencia visual del "Glassmorphism Oscuro" y facilitar futuras modificaciones. Esto evita la duplicación de código y asegura que el diseño sea adaptable y fácil de mantener.
• Manejo de Estado: Se implementará una estrategia de manejo de estado global (usando Context API/Redux para React, o Vuex para Vue.js) para gestionar datos críticos como la información del usuario logueado, su rol y los permisos de acceso, asegurando que esta información esté disponible en toda la aplicación de manera eficiente.
• Control de Acceso Basado en Roles (RBAC): El Frontend implementará la lógica para adaptar la interfaz de usuario (visibilidad de secciones, botones, etc.) basándose en el rol del usuario autenticado, cuyos permisos son validados por el Backend. Esto garantiza que solo se presenten las funcionalidades a las que el usuario tiene acceso según su perfil (Administrador Principal, Admin Secundario, Asistente, Invitado).

4.1. Layout y Contenedores
Estos componentes definen la estructura general de la aplicación y sus principales áreas de contenido.
a. Contenedor Principal (App Layout)
• Propósito y función: Sirve como el envoltorio de toda la aplicación de escritorio, estableciendo el fondo principal y la base para el efecto visual "Aurora".
• Variantes: N/A. Es el layout base.
• Estados visuales: N/A. Es un contenedor estático.
• Propiedades (props): children (para contener todos los demás componentes del Dashboard).
• Alineación visual con Glassmorphism: Presenta un fondo principal de gris carbón suave (#121212) para reducir la fatiga visual. Detrás de los paneles de vidrio, incorpora auras difusas de azul oceánico profundo (#4A69FF) y violeta vibrante (#8C52FF), esenciales para el efecto "Aurora" que proporciona el fondo dinámico para el Glassmorphism.
b. Barra de Navegación Lateral (Sidebar)
• Propósito y función: Permite al usuario navegar a las secciones principales del sistema, como "Pacientes", "Agenda", "Reportes", "Documentos" y "Administración". Su contenido se adapta al rol del usuario logueado.
• Variantes: N/A. Es una estructura fija.
• Estados visuales: N/A. Su contenido cambia dinámicamente según el rol.
• Propiedades (props): sections (un array de objetos que define las secciones y sus sub-opciones accesibles para el usuario actual), currentUserRole (para adaptar las opciones de navegación).
• Alineación visual con Glassmorphism: Actúa como un panel lateral que sigue la estética "Glassmorphism Oscuro". Aunque no se especifica explícitamente como "vidrio esmerilado" en sí, se infiere que mantendrá la paleta de colores y la tipografía para una coherencia integral.
c. Cabecera (Header)
• Propósito y función: Ubicada en la parte superior, muestra la información del usuario logueado (nombre y rol) y proporciona la opción para "Cerrar Sesión".
• Variantes: N/A.
• Estados visuales: N/A.
• Propiedades (props): userName, userRole, onLogoutClick (función para manejar el cierre de sesión).
• Alineación visual con Glassmorphism: El texto y los iconos son nítidos y brillantes, utilizando el color principal de texto blanco roto (#F5F5F5) para alta legibilidad sobre el fondo oscuro o semi-transparente.
d. Área de Contenido Principal (Main Content Area)
• Propósito y función: Es el espacio más grande y dinámico del Dashboard, donde se muestran los paneles de información clave ("Citas del Día", "Métricas Rápidas", "Pacientes Recientes") y donde se cargan las vistas de las secciones seleccionadas desde la barra lateral.
• Variantes: N/A. Es el contenedor de contenido principal.
• Estados visuales: N/A. Su contenido cambia dinámicamente.
• Propiedades (props): children (para cargar el contenido de las diferentes secciones del

Dashboard).
• Alineación visual con Glassmorphism: Este área es donde los paneles flotantes de "vidrio esmerilado" (Glassmorphism) con esquinas redondeadas y un sutil borde luminoso cobran vida, mostrando la información clave de forma jerárquica y visualmente cómoda.
e. Paneles/Tarjetas (Glassmorphism Panels/Cards)
• Propósito y función: Son los contenedores visuales principales para la información estructurada dentro del Área de Contenido Principal (ej., "Citas del Día", "Métricas Rápidas", "Pacientes Recientes").
• Variantes: Pueden variar en tamaño (ej., pequeños para métricas rápidas, grandes para listados de pacientes).
• Estados visuales: N/A. Son contenedores de información.
• Propiedades (props): title, content (el JSX/componente interno que muestra la información específica), onClick (si el panel es interactivo, como un acceso directo).
• Alineación visual con Glassmorphism: Representan la esencia del diseño "Glassmorphism Oscuro". Son semi-transparentes y desenfocados (usando backdrop-filter: blur(24px)). Tienen esquinas suavemente redondeadas (border-radius: 16px) y un sutil borde luminoso de 1px de color blanco semitransparente que "atrapa" la luz del fondo, creando una sensación de profundidad y jerarquía.

4.2. Componentes de Navegación
Estos componentes permiten la interacción del usuario con la barra lateral.
a. Elemento de Navegación (Sidebar Item)
• Propósito y función: Representa cada una de las opciones en la barra de navegación lateral (ej., "Pacientes", "Agenda", "Reportes"). Permite la selección de la sección principal y, si aplica, la expansión para mostrar sub-opciones.
• Variantes:
    ◦ Con Sub-opciones: Se puede expandir y contraer para mostrar opciones anidadas (ej., "Pacientes" > "Nuevo Paciente", "Buscar Paciente").
    ◦ Sin Sub-opciones: Un elemento de menú directo (ej., "Documentos").
• Estados visuales:
    ◦ Normal: Estado por defecto, icono y texto con el color secundario (#A8A8A8).
    ◦ Hover: Un sutil cambio de color del texto y/o icono a blanco roto (#F5F5F5) o un ligero efecto de brillo/hundimiento al pasar el cursor.
    ◦ Activo/Seleccionado: El elemento actualmente visible en el área de contenido principal. Texto y/o icono con el color primario (#F5F5F5) y posiblemente una barra lateral sutil (1px solid rgba(255, 255, 255, 0.18)) o un fondo sutilmente resaltado.
• Propiedades (props): icon (para el icono de estilo de línea), label (texto de la opción), path (ruta de navegación), hasSubOptions (booleano), subOptions (array de sub-opciones), isActive (booleano para estado activo), onClick (función para manejar la navegación).
• Alineación visual con Glassmorphism: Los iconos son de estilo de línea con un grosor de trazo consistente y terminaciones/esquinas suavemente redondeadas, utilizando el color principal de texto (#F5F5F5) para los estados activos. La tipografía "Inter" se usa para las etiquetas, asegurando legibilidad. La retroalimentación visual al interactuar (hover/activo) es sutil pero clara, manteniendo la estética pulcra y de alta calidad del Glassmorphism.
4.3. Componentes Interactivos Básicos
Estos son los bloques de construcción fundamentales para la interacción del usuario.
a. Botones (Button)
• Propósito y función: Permiten al usuario ejecutar acciones específicas o navegar a otras secciones. Son el elemento principal para la interacción directa.
• Variantes:
    ◦ Primario: Para acciones principales y destacadas (ej., "Iniciar Sesión", "Guardar Paciente", "+ Nuevo Registro"). Utiliza un relleno sólido de color de acento.
    ◦ Secundario: Para acciones menos prominentes, acciones alternativas o botones de cancelación (ej., "Cancelar", "Reprogramar"). Podrían tener un fondo transparente o un borde sutil para menor impacto visual.
    ◦ Terciario/Icono: Para acciones muy discretas o solo con un icono (ej., menú de acciones en tabla, iconos de acceso rápido).
• Estados visuales:
    ◦ Normal: Apariencia por defecto, con colores y sombreado definidos.
    ◦ Hover: Un sutil aumento de brillo o un ligero hundimiento al pasar el cursor,

ofreciendo retroalimentación clara.
    ◦ Activo/Focus: Un borde o sombra más prominente, indicando que el botón está seleccionado o en uso.
    ◦ Disabled: Reduce la opacidad y desactiva la interacción (no responde a clics), indicando que la acción no está disponible.
    ◦ Loading: Muestra un spinner o indicador de carga dentro del botón o junto a él, indicando que la acción está en progreso.
• Propiedades (props): onClick (función a ejecutar), label (texto del botón), variant (primario, secundario, etc.), disabled (booleano), loading (booleano), icon (opcional).
• Alineación visual con Glassmorphism:
    ◦ Relleno Primario: Utiliza accent-blue (#4A69FF) o button-primary-fill (#4A69FF).
    ◦ Texto: text-primary (#F5F5F5) para alta legibilidad.
    ◦ Sombra: Un box-shadow suave para un efecto de brillo que contribuye a la sensación de flotación del Glassmorphism.
    ◦ Esquinas: Suavemente redondeadas (ej., border-radius: 8px).

b. Campos de Entrada de Texto (Input / Textarea)
• Propósito y función: Permiten al usuario introducir texto, números u otros datos en la interfaz (ej., "Nombre de Usuario", "Contraseña", campos de perfil de paciente).
• Variantes:
    ◦ Input (tipo text, password, number, email, date, etc.) para campos cortos y de una sola línea.
    ◦ Textarea para campos multi-línea, como "Notas de Evolución".
• Estados visuales:
    ◦ Normal: Apariencia por defecto, con fondo sutilmente contrastante o translúcido.
    ◦ Focus: Un borde o resplandor sutil que indica que el campo está activo para la entrada de texto.
    ◦ Disabled: Desactiva la entrada y reduce la opacidad.
    ◦ Error: Un borde o texto en color de error (ej., rojo) y un mensaje de validación debajo del campo (ej., "Campo 'Nombre' requerido", "Formato de email inválido").
    ◦ Filled: Cuando el campo contiene datos.
• Propiedades (props): value, onChange (función de manejo de entrada), label (etiqueta del campo), placeholder, type (para inputs), rows (para textarea), disabled (booleano), error (booleano), errorMessage (string).
• Alineación visual con Glassmorphism:
    ◦ Fondo: Apariencia limpia y minimalista, posiblemente un fondo sutilmente contrastante o translúcido que mantenga la estética de vidrio.
    ◦ Texto: Las etiquetas y el texto introducido utilizan el color principal de texto (--color-text-primary: #F5F5F5) para una legibilidad óptima sobre el fondo oscuro.
    ◦ Bordes: Posiblemente bordes suaves o inexistentes en estado normal, y un borde sutilmente luminoso en el estado focus para indicar la interacción.

c. Checkboxes (Checkbox)
• Propósito y función: Permiten seleccionar o deseleccionar opciones (ej., selección múltiple de filas en una tabla para acciones en lote).
• Variantes: Individual, Maestro (para seleccionar/deseleccionar todos en una tabla).
• Estados visuales:
    ◦ Sin marcar: Estado por defecto.
    ◦ Marcado: Cuando la opción ha sido seleccionada.
    ◦ Disabled: No permite la interacción.
• Propiedades (props): checked (booleano), onChange (función de manejo de cambio), disabled (booleano).
• Alineación visual con Glassmorphism:
    ◦ Marcado: El color de acento interactivo (--color-accent-interactive: #34D1F3) para el relleno, sin borde.
    ◦ Sin marcar: Fondo transparente y un borde de 1px de color secundario (--color-text-secondary).
    ◦ El contraste visual asegura que los estados sean fácilmente distinguibles en el entorno oscuro.

d. Iconos (Icon)
• Propósito y función: Representan visualmente acciones, categorías o información, mejorando la usabilidad y la comprensión rápida de la interfaz (ej., iconos de navegación, iconos de acciones).
• Variantes: Diversos iconos según su propósito (casa para inicio, lupa para búsqueda, papelera para eliminar, gráfico de barras para reportes, etc.).

• Estados visuales:
    ◦ Normal: Utiliza el color secundario (#A8A8A8) o un tono más tenue.
    ◦ Hover/Activo: Cambia al color principal de texto (#F5F5F5) o a un color de acento, indicando interactividad o selección.
• Propiedades (props): name (nombre del icono), size (tamaño), color (color, que puede ser sobreescrito por el estado), onClick (si el icono es interactivo).
• Alineación visual con Glassmorphism:
    ◦ Estilo: Iconos de estilo de línea (line-style) con un grosor de trazo consistente (aproximadamente 1.5px) y terminaciones/esquinas suavemente redondeadas. Esto contribuye a la estética limpia y moderna.
    ◦ Coloración: Principalmente de un solo color (monocromático), utilizando --color-text-primary o --color-text-secondary para mantener la coherencia y legibilidad sobre los fondos oscuros o semi-transparentes.
    ◦ Biblioteca: Se recomienda el uso de bibliotecas de iconos consistentes como Iconify o Feather Icons para uniformidad y escalabilidad.

e. Elementos de Tipografía (Text / Heading / Label)

• Propósito y función: Estructuran y presentan todo el contenido textual de la interfaz, desde títulos y encabezados hasta párrafos y etiquetas de campos, asegurando legibilidad y jerarquía.
• Variantes:
    ◦ Heading (H1, H2, etc.) para títulos y secciones importantes.
    ◦ Body para párrafos de texto y contenido principal.
    ◦ Label para etiquetas de campos de entrada o elementos más pequeños.
• Estados visuales: N/A para el componente en sí, pero su color y peso pueden cambiar contextualmente (ej., texto de error, texto deshabilitado).
• Propiedades (props): children (el contenido de texto), variant (H1, body, label), color (ej., primary, secondary, error), weight (bold, regular, etc.).
• Alineación visual con Glassmorphism:
    ◦ Fuente: Inter, una fuente sans-serif geométrica optimizada para UI, seleccionada por su legibilidad en entornos digitales.
    ◦ Colores:
        ▪ --color-text-primary: #F5F5F5 (blanco roto) para títulos, texto principal y elementos clave. Proporciona alto contraste sobre el vidrio oscuro y el fondo principal, superior a 4.5:1 (WCAG).
        ▪ --color-text-secondary: #A8A8A8 (gris claro) para texto secundario, descripciones o ayuda, manteniendo el confort visual.
    ◦ Escala Tipográfica: Definida con rem (ej., H1: 1.5rem/24px Bold; Body: 0.875rem/14px Regular/Bold; Label: 0.75rem/12px Medium).

4.4. Componentes de Visualización de Datos y Estructura Compleja

Estos componentes muestran información estructurada y permiten la interacción avanzada.

a. Tabla de Datos (DataTable)

• Propósito y función: Presenta información en un formato tabular organizado, como "Pacientes Recientes", permitiendo visualizar y gestionar grandes volúmenes de datos de manera eficiente.
• Variantes: Tabla estándar con columnas configurables.
• Estados visuales:
    ◦ Normal: Filas y celdas con estilos predeterminados.
    ◦ Fila Hover: Un ligero cambio de fondo o un sutil resplandor en la fila al pasar el cursor para indicar interactividad.
    ◦ Fila Seleccionada: Resaltado del fondo de la fila para indicar que ha sido seleccionada (ej., mediante un checkbox).
    ◦ Estado Vacío: Un mensaje claro (ej., "No se encontraron pacientes") y posiblemente un icono ilustrativo, en lugar de una tabla vacía.
    ◦ Cargando: Un spinner o una superposición de carga sobre la tabla.
• Propiedades (props): data (array de objetos con los datos a mostrar), columns (definición de columnas: key, label, renderCell, sortable), onRowClick (para ver detalles), onSelectionChange (para checkboxes), isLoading (booleano), emptyMessage (string), pagination (objeto de paginación).
• Alineación visual con Glassmorphism:
    ◦ Contenida dentro de un Panel/Tarjeta Glassmorphism.
    ◦ Encabezados de Columna: Utilizan el color text-primary (#F5F5F5) y la fuente Inter en el peso adecuado.
    ◦ Contenido de Celdas: El texto de datos principal utiliza text-primary. Información

secundaria puede usar text-secondary (#A8A8A8).
    ◦ Columna de Identificación: "ID/Nombre del Paciente" en negrita y funciona como enlace para ver el detalle completo.
    ◦ Estados: Las píldoras de estado (State Pills) como "Active" (Verde) y "Inactive" (Rojo/Gris) utilizan colores acentuados para una identificación rápida.
    ◦ Checkbox de Selección: En cada fila y un maestro en el encabezado, siguiendo el estilo Glassmorphism del checkbox.
    ◦ Menú de Acciones por Fila: Un icono de tres puntos vertical (Icon) que al hacer clic despliega un menú contextual flotante (posiblemente un Modal o Panel Lateral con opciones como "Ver Detalles", "Editar Registro", "Archivar", "Eliminar (Suave)").

b. Paginación (Pagination)

• Propósito y función: Permite al usuario navegar a través de grandes conjuntos de datos presentados en tablas o listas, cargando el contenido en bloques (páginas).
• Variantes: Controles de navegación (anterior, siguiente, números de página, selector de tamaño de página).
• Estados visuales:
    ◦ Número de Página Activa: Resaltado con un color de acento o un efecto de brillo sutil para indicar la página actual.
    ◦ Controles Deshabilitados: (ej., "Anterior" en la primera página, "Siguiente" en la última página) con menor opacidad.
• Propiedades (props): currentPage, totalPages, onPageChange (función), pageSize (opcional), onPageSizeChange (opcional).
• Alineación visual con Glassmorphism: Botones y texto siguen el estilo de los demás componentes interactivos, con una retroalimentación visual clara y sutil.

c. Componente de Filtros (Filter Panel/Modal)

• Propósito y función: Proporciona opciones para que el usuario refine la información mostrada en tablas o vistas (ej., filtrar pacientes por fecha de última consulta, tipo de cita).
• Variantes:
    ◦ Panel desplegable: Que se abre in-situ desde un icono de filtro en la barra de acciones de la tabla.
    ◦ Modal: Una ventana emergente con opciones de filtro avanzadas.
    ◦ Panel Lateral: Un panel que se desliza desde un lado de la pantalla (similar a la barra lateral de navegación).
• Estados visuales:
    ◦ Abierto/Cerrado: Transiciones suaves para abrir y cerrar el panel/modal.
    ◦ Filtros Aplicados: Indicadores visuales (ej., un punto de color en el icono de filtro) que muestran que hay filtros activos.
• Propiedades (props): isOpen (booleano), onClose (función), filterOptions (definición de campos de filtro: tipo, label), onApplyFilters (función), onClearFilters (función).
• Alineación visual con Glassmorphism:
    ◦ El Panel o Modal contenedor seguiría la estética de "vidrio esmerilado" del Glassmorphism (semi-transparente, desenfocado, esquinas redondeadas, borde luminoso).
    ◦ Los campos de entrada (Input), checkboxes y botones dentro del componente de filtro seguirían sus respectivos estilos Glassmorphism.

d. Modales (Modal)

• Propósito y función: Presentan contenido importante o solicitan interacción del usuario en una ventana flotante sobre el contenido principal, sin que el usuario abandone la vista actual (ej., formularios de creación/edición de paciente, confirmaciones de eliminación).
• Variantes:
    ◦ Formulario: Para la creación o edición de registros.
    ◦ Confirmación: Para acciones críticas (ej., "¿Desea archivar los X registros seleccionados?").
    ◦ Alerta/Información: Para mensajes importantes o notificaciones.
• Estados visuales:
    ◦ Abierto: Visible sobre un fondo semi-oscurecido (backdrop).
    ◦ Cerrado: Oculto.
• Propiedades (props): isOpen (booleano), onClose (función), title (título del modal), children (contenido del modal), onConfirm (función), onCancel (función), confirmButtonLabel, cancelButtonLabel.
• Alineación visual con Glassmorphism:
    ◦ El modal en sí es un Panel/Tarjeta Glassmorphism: semi-transparente y desenfocado (backdrop-filter: blur(24px)), con esquinas suavemente redondeadas (border-radius: 16px) y

un sutil borde luminoso de 1px.
    ◦ El fondo del modal (el backdrop) se oscurecería sutilmente para enfocar la atención en la ventana emergente, manteniendo la sensación de profundidad.
e. Paneles Laterales (Side Panel / Drawer)
• Propósito y función: Ofrecen un espacio adicional para contenido o controles que no requieren una vista de página completa, como detalles adicionales de un elemento seleccionado, formularios de edición o filtros avanzados.
• Variantes: Deslizante desde la izquierda o la derecha.
• Estados visuales:
    ◦ Abierto: Visible, con el contenido principal posiblemente encogido o superpuesto.
    ◦ Cerrado: Oculto.
• Propiedades (props): isOpen (booleano), onClose (función), title (título del panel), children (contenido).
• Alineación visual con Glassmorphism:
    ◦ Similar a un Panel/Tarjeta Glassmorphism, pero extendido verticalmente. Mantendrá la semi-transparencia, el desenfoque y los bordes luminosos.
    ◦ La transición de apertura/cierre podría incluir una sutil animación de deslizamiento.
4.5. Componentes Específicos de Módulos
Estos componentes están diseñados para funcionalidades especializadas.
a. Interfaz del Asistente Virtual (AI Assistant Interface)
• Propósito y función: Permite al personal interactuar con el modelo de IA (Gemma 3) para realizar consultas rápidas, obtener resúmenes de texto o redactar informes. Es un chat interno que solo funciona con texto.
• Variantes: N/A.
• Estados visuales:
    ◦ Normal: Área de visualización de mensajes y campo de entrada activos.
    ◦ Escribiendo: Indicador de que el usuario está introduciendo texto.
    ◦ Recibiendo Respuesta/Cargando: Un indicador visual (ej., puntos suspensivos o spinner) cuando la IA está procesando o generando una respuesta.
    ◦ Errores: Mensajes que indican un problema con la consulta o la IA.
• Propiedades (props): messages (array de objetos {sender, text}), onSendMessage (función), isLoading (booleano), suggestedActions (array de strings para botones de acción sugerida).
• Alineación visual con Glassmorphism:
    ◦ El área de visualización de mensajes y el campo de entrada podrían estar contenidos dentro de un Panel/Tarjeta Glassmorphism.
    ◦ El campo de entrada seguiría el estilo de Input Glassmorphism.
    ◦ Los botones de acción sugerida (ej., "te lo envío a tu correo o prefieres descargarlo?") seguirían el estilo de Button (quizás Secundario o con el color accent-interactive).
b. Componentes del Visor de Archivos Visuales (Visual File Viewer)
• Propósito y función: Permite visualizar y navegar por archivos visuales (imágenes y videos cortos) asociados a los expedientes de los pacientes, como RX o grabaciones de cirugías.
• Variantes:
    ◦ Carrusel Contenedor: Para mostrar miniaturas y navegar entre ellas.
    ◦ Elementos de Miniatura Individual: Representación visual de cada archivo.
    ◦ Vista Ampliada: El área principal donde se muestra la imagen o video seleccionado a tamaño completo.
    ◦ Flechas de Navegación: Para avanzar o retroceder en el carrusel/galería.
    ◦ Controles de Filtro de Origen/Tipo: Botones o checkboxes para filtrar por "Cloud Drives", "Photos", "Videos".
• Estados visuales:
    ◦ Normal: Archivos visibles y navegables.
    ◦ Cargando: Indicador de carga al abrir un archivo grande.
    ◦ Vacío: Mensaje si no hay archivos visuales asociados al paciente.
• Propiedades (props): files (array de objetos de archivo), currentFileIndex (índice del archivo actual), onNavigate (función), onFilterChange (función), selectedFilter (string).
• Alineación visual con Glassmorphism:
    ◦ El Carrusel o Visor principal podría ser un Panel/Tarjeta Glassmorphism, aprovechando la transparencia y el desenfoque para integrarse con el fondo Aurora.
    ◦ Las Miniaturas y la Vista Ampliada estarían contenidas dentro de este panel.

    ◦ Las Flechas de Navegación utilizarían Iconos de estilo de línea.
    ◦ Los Controles de Filtro (botones/checkboxes) seguirían el diseño Glassmorphism de sus respectivos componentes.

c. Componentes para Widgets de Servicios Externos (External Service Widgets)

• Propósito y función: Proporcionan acceso rápido a herramientas de uso común como Gmail, Google Drive, o aplicaciones de Office, integradas visualmente en la interfaz.
• Variantes: Botones con iconos, o pequeños paneles con un icono y una etiqueta.
• Estados visuales:
    ◦ Normal: Icono y etiqueta visibles.
    ◦ Hover: Un sutil brillo o cambio de color al pasar el cursor.
• Propiedades (props): serviceName (nombre del servicio), icon (icono del servicio), onClick (función para abrir el servicio/aplicación), url (opcional, si es una integración web).
• Alineación visual con Glassmorphism:
    ◦ Podrían ser pequeños Paneles/Tarjetas Glassmorphism flotantes, cada uno representando un acceso directo, o simplemente Iconos estilizados con etiquetas de texto.
    ◦ El uso de iconos de estilo de línea y el color text-primary asegurarían la consistencia.

4.6. Componentes de Feedback y Contenedores Generales

a. Componentes de Feedback Visual (Loading Spinner / Toast Notification)

• Propósito y función: Informan al usuario sobre el estado del sistema, como operaciones en curso (carga) o resultados de acciones (notificaciones de éxito/error).
• Variantes:
    ◦ Spinner de Carga: Un indicador animado circular que aparece durante operaciones asíncronas.
    ◦ Mensajes de Notificación/Toast: Pequeños banners que aparecen brevemente para confirmar una acción o mostrar un error (ej., "Guardado con éxito", "Error al subir el archivo").
• Estados visuales:
    ◦ Visible: El componente de feedback se muestra.
    ◦ Oculto: El componente de feedback no es visible.
• Propiedades (props):
    ◦ Spinner: isVisible (booleano), size, color.
    ◦ Toast: isVisible (booleano), message (string), type (success, error, info, warning), duration (tiempo en ms).
• Alineación visual con Glassmorphism:
    ◦ Spinners: Podrían ser círculos sutilmente luminosos o con el color accent-blue o accent-interactive.
    ◦ Toast: Aparecerían como pequeños Paneles/Tarjetas Glassmorphism transitorios, con el desenfoque y los bordes luminosos. El texto dentro utilizaría text-primary y el color del fondo del toast podría variar ligeramente según el tipo (ej., un toque de verde para éxito, rojo para error), manteniendo la opacidad y el desenfoque Glassmorphism.

b. Formularios (Form Container)

• Propósito y función: Agrupan un conjunto de campos de entrada y botones de acción relacionados para la recolección estructurada de datos (ej., "Creación de Perfil de Paciente", "Agendar Nueva Cita").
• Variantes: Formulario de creación, formulario de edición, formulario de búsqueda.
• Estados visuales:
    ◦ Normal: Todos los campos y botones son visibles y están listos para la interacción.
    ◦ Validando/Enviando: Puede deshabilitar los campos o mostrar un spinner en el botón de submit.
    ◦ Error de Validación: Muestra mensajes de error junto a los campos correspondientes.
• Propiedades (props): onSubmit (función al enviar), children (los campos y botones internos), title (título del formulario).
• Alineación visual con Glassmorphism:
    ◦ El formulario en sí estaría contenido dentro de un Panel/Tarjeta Glassmorphism o un Modal (que es un panel Glassmorphism), dependiendo de su ubicación y flujo.
    ◦ Los Campos de Entrada, Checkboxes y Botones dentro del formulario seguirían sus respectivos estilos Glassmorphism, asegurando una experiencia visual coherente y agradable.

5. Implementación del Sistema de Diseño (Glassmorphism Oscuro)

El estilo visual "Glassmorphism Oscuro" es fundamental para la experiencia de usuario del ecosistema, especialmente en el Dashboard de Escritorio, y no es solo estético sino

funcional, diseñado para ofrecer comodidad visual y una jerarquía de información clara.
• Estrategia de implementación de estilos: La estrategia de implementación de estilos se centra en el uso de Propiedades Personalizadas de CSS (CSS Variables), declaradas bajo el selector :root. Esta aproximación centraliza los valores de diseño, facilitando la mantenibilidad, los cambios globales y futuras tematizaciones del ecosistema. Aunque no se especifica explícitamente el uso de CSS Modules, SCSS o Styled Components, la adopción de variables CSS sugiere un enfoque modular y escalable para la gestión de estilos.
• Definición y uso de Variables CSS (Tokens de Diseño): Los "tokens de diseño" se codificarán como variables CSS para mantener la coherencia y facilitar su aplicación a través de todos los componentes.
    ◦ Paleta de Colores:
        ▪ --color-background-main: #121212 (Fondo principal, gris carbón suave).
        ▪ --color-surface-glass-base: 29, 35, 50 (Valores RGB para la base del vidrio, azul/gris oscuro desaturado, utilizado con opacidad).
        ▪ --color-border-glass-base: 255, 255, 255 (Valores RGB para el blanco, utilizado para bordes sutiles).
        ▪ --color-text-primary: #F5F5F5 (Texto principal, blanco roto para alta legibilidad).
        ▪ --color-text-secondary: #A8A8A8 (Texto secundario/ayuda, gris claro).
        ▪ --color-accent-blue: #4A69FF (Color de acento primario, azul oceánico profundo).
        ▪ --color-accent-purple: #8C52FF (Color de acento secundario, violeta vibrante).
        ▪ --color-accent-interactive: #34D1F3 (Color para elementos interactivos, azul brillante).
        ▪ --color-button-primary-fill: #4A69FF (Color de relleno para botones primarios).
    ◦ Escala Tipográfica (Fuente: Inter, sans-serif geométrica optimizada para UI):
        ▪ --font-size-h1: 1.5rem (24px), 700 (Bold).
        ▪ --font-size-h2: 1.125rem (18px), 600 (Semibold).
        ▪ --font-size-body: 0.875rem (14px), 700 (Bold) o 400 (Regular).
        ▪ --font-size-label: 0.75rem (12px), 500 (Medium).
        ▪ Los colores de texto principales y secundarios se definirán con --color-text-primary y --color-text-secondary respectivamente.
    ◦ Espaciado y border-radius:
        ▪ --border-radius-card: 16px (Para paneles flotantes, esquinas suaves).
        ▪ --border-radius-button: 8px (Para botones).
        ▪ Aunque no se definen variables específicas para un sistema de espaciado general como --space-unit, el relleno (padding: 24px) se utiliza de manera consistente en los paneles de vidrio.
• Implementación técnica detallada del efecto Glassmorphism en los componentes de panel/tarjeta: Los paneles y tarjetas del Dashboard, como el contenedor de login o las áreas de información clave, implementarán el efecto Glassmorphism utilizando las siguientes propiedades CSS:
    ◦ background-color: rgba(var(--color-surface-glass-base), 0.6): Establece un fondo semi-transparente con un 60% de opacidad sobre el color base de la superficie de vidrio.
    ◦ backdrop-filter: blur(24px): Aplica un desenfoque significativo al contenido que se encuentra detrás del panel, creando el efecto "esmerilado".
    ◦ -webkit-backdrop-filter: blur(24px): Incluye el prefijo de navegador para asegurar la compatibilidad con Safari.
    ◦ border-radius: var(--border-radius-card): Redondea suavemente las esquinas de los paneles.
    ◦ border: 1px solid rgba(var(--color-border-glass-base), 0.18): Delinea los paneles con un borde delicado de 1 píxel, de color blanco semi-transparente (18% de opacidad), que "atrapa" la luz del fondo y define la forma del panel.
    ◦ padding: 24px: Asegura un espaciado interno consistente dentro de los paneles.
• Implementación técnica del fondo Aurora UI: La Interfaz Aurora no es un adorno, sino un componente funcional indispensable que proporciona la "materia prima" visual para que el filtro backdrop-filter del Glassmorphism actúe. Los dos estilos son codependientes: la Aurora UI proporciona la "luz" y el Glassmorphism la "lente".
    ◦ Se creará un fondo dinámico utilizando auras difusas de azul oceánico profundo (#4A69FF) y violeta vibrante (#8C52FF).
    ◦ Técnicamente, esto se logra con elementos div posicionados absolutamente con un fuerte filter: blur() o con múltiples radial-gradients. Este fondo debe tener suficiente variación tonal y de color para que el efecto de vidrio sea visible y efectivo.
• Aplicación de estilos a los componentes reutilizables: El sistema de diseño garantiza

una apariencia Glassmorphism coherente en todos los componentes interactivos:
    ◦ Botones (Iniciar Sesión):
        ▪ background-color: var(--color-button-primary-fill): Para un relleno sólido de alto contraste.
        ▪ border-radius: var(--border-radius-button): Esquinas suaves.
        ▪ padding: 12px 24px: Espaciado interno.
        ▪ color: var(--color-text-primary): Para la etiqueta de texto.
        ▪ box-shadow: 0 4px 12px rgba(var(--color-accent-blue), 0.3): Un suave efecto de brillo.
        ▪ La interacción al pulsarlo provocará un sutil aumento de brillo o un ligero hundimiento, ofreciendo una retroalimentación clara.
    ◦ Campos de Entrada (Nombre de Usuario, Contraseña):
        ▪ Diseñados con una apariencia limpia y minimalista, posiblemente con un fondo sutilmente contrastante o translúcido que mantenga la estética de vidrio.
        ▪ El texto de las etiquetas y el que el usuario introduce utilizará --color-text-primary (#F5F5F5) para una legibilidad óptima sobre el fondo oscuro.
    ◦ Casillas de Verificación (Checkboxes):
        ▪ Marcado: background-color: var(--color-accent-interactive), border: none.
        ▪ Sin marcar: background-color: transparent, border: 1px solid var(--color-text-secondary).
    ◦ Tablas (ej., PACIENTES RECIENTES): Aunque no se especifican propiedades Glassmorphism para las tablas en sí, los elementos de texto y los iconos dentro de ellas seguirán la paleta de colores y la tipografía definidas por las variables CSS, utilizando --color-text-primary y --color-text-secondary para mantener la coherencia y legibilidad.
• Uso de Iconografía:
    ◦ Estilo: Los iconos serán de estilo de línea (line-style), con un grosor de trazo consistente (aproximadamente 1.5px) y terminaciones/esquinas suavemente redondeadas.
    ◦ Color: Utilizarán un solo color, var(--color-text-primary), para una excelente integración con el modo oscuro.
    ◦ Recomendación: Se recomienda el uso de bibliotecas como Iconify o Feather Icons para asegurar consistencia y escalabilidad.
• Consideraciones específicas para el modo oscuro avanzado: El diseño del "Glassmorphism Oscuro" incorpora principios avanzados para asegurar la legibilidad y el confort visual en un tema oscuro:
    ◦ Evitar el negro puro: El fondo principal es un gris carbón muy oscuro (#121212), no negro puro. Esto reduce la fatiga visual y permite la percepción de profundidad y sombras sutiles.
    ◦ Colores de acento desaturados: Los tonos de azul (#4A69FF) y violeta (#8C52FF) utilizados para las "Auras" y acentos están desaturados. Esto previene la "vibración" óptica y mantiene un confort visual general, especialmente en exposiciones prolongadas.
    ◦ Texto de alto contraste pero suavizado: El texto principal utiliza un blanco roto o gris muy claro (#F5F5F5), lo que suaviza el contraste general sin comprometer la legibilidad. Esto ayuda a reducir el deslumbramiento y cumple con una relación de contraste superior a 4.5:1 (WCAG), un estándar clave de accesibilidad.
    ◦ Comunicación de profundidad sin sombras: En entornos oscuros, las sombras son ineficaces. El diseño utiliza un sistema multifacético para comunicar la elevación:
        ▪ La luz como indicador de elevación: Las superficies que "se acercan" al usuario se vuelven más claras.
        ▪ La superposición como señal de profundidad: El propio efecto de desenfoque del Glassmorfismo crea una separación perceptual entre el primer plano nítido y el fondo desenfocado.
        ▪ La definición del borde como sustituto de la sombra: El sutil borde luminoso define claramente el contorno del panel, separándolo del fondo donde una sombra no sería visible.
    ◦ Accesibilidad: Se aborda la accesibilidad mediante un enfoque dual de contraste:
        ▪ Contraste de Contenido (Alto): Asegura alta legibilidad entre el texto/iconos y su fondo inmediato (el panel de vidrio oscuro y semi-transparente).
        ▪ Contraste Estructural (Bajo): Mantiene el contraste entre el panel de vidrio y el fondo principal deliberadamente bajo para preservar el efecto flotante y esmerilado.
6. Gestión del Estado
La "Pantalla: Dashboard Principal (Vista General)" actúa como un centro de control dinámico que muestra datos relevantes en tiempo real y proporciona puntos de entrada intuitivos a todas las funcionalidades principales del sistema. Para que la interfaz de

usuario (UI) se mantenga sincronizada y reactiva a las acciones del usuario, es fundamental una buena gestión del estado. Dado que el frontend del Dashboard se desarrollará con frameworks como React, Vue.js o similares, la elección de una estrategia de gestión del estado es vital.
Elección de una librería o patrón para la gestión del estado global de la aplicación: Aunque los documentos no especifican una librería concreta, la necesidad de una UI reactiva y sincronizada sugiere la implementación de patrones de gestión de estado robustos.
• Para un frontend basado en React, se podrían considerar:
    ◦ Context API + useReducer: Una solución nativa de React adecuada para la gestión de estados globales de complejidad media. Es ideal para datos de usuario autenticado o estados de modales/paneles que necesitan ser accesibles por muchos componentes sin una prop drilling excesiva.
    ◦ Redux (con Redux Toolkit) o Zustand: Para escenarios de mayor complejidad, donde la gestión de datos asíncronos (llamadas a la API) y la necesidad de un flujo de datos predecible sean prioritarias. Proporciona una "fuente única de verdad" para el estado global y herramientas potentes para depuración y escalabilidad.
• Para un frontend basado en Vue.js, se podrían considerar:
    ◦ Vuex o Pinia: Los gestores de estado oficiales de Vue.js. Proporcionan un almacén centralizado de estado que facilita la reactividad y la depuración, siendo esenciales para aplicaciones Vue a gran escala.
La elección específica dependerá de la complejidad exacta de la lógica de negocio y el volumen de datos que se esperen manejar en tiempo real. Para un "Núcleo Administrativo Central", que es la "columna vertebral y la base de datos central de todo el ecosistema", una solución más estructurada como Redux/Vuex/Pinia podría ofrecer mayor mantenibilidad y escalabilidad a largo plazo.
Descripción de las partes principales del estado que necesitarán ser gestionadas globalmente: Para el Dashboard, varias piezas de información son críticas y requerirán un estado global para garantizar la coherencia y la reactividad:
• Datos del Usuario Autenticado y su Rol: La cabecera del dashboard mostrará el usuario logueado y su rol, y la barra de navegación lateral presentará las secciones a las que tiene acceso según sus permisos. Esta información es fundamental para la adaptación de la interfaz y el control de acceso basado en roles.
• Estado de Carga y Error Global de APIs: Las comunicaciones con el backend pueden presentar "problemas de conectividad" o "errores de servidor". Es crucial tener un estado global para indicar cuándo se están realizando llamadas a la API (estado loading, para mostrar spinners o deshabilitar botones) y cuándo ha ocurrido un error (estado error, para mostrar mensajes de error genéricos).
• Datos de la Tabla de Pacientes y Otros Datos Dinámicos: La sección "PACIENTES RECIENTES" y las "CITAS DEL DÍA" muestran datos dinámicos que se actualizan en tiempo real o casi real. Estos datos, junto con las "MÉTRICAS RÁPIDAS", deben ser parte del estado global para que cualquier componente que los necesite pueda acceder a ellos de forma consistente.
• Estado de Visibilidad de Modales/Paneles: Si bien no se menciona explícitamente en los documentos, una interfaz de usuario compleja como el Dashboard a menudo utiliza modales o paneles laterales (como los de filtrado o creación de nuevos registros) cuya visibilidad puede ser gestionada a nivel global para una mejor coordinación entre componentes.
• Contenido de Texto Generado por IA: Las funcionalidades de Inteligencia y Automatización Integrada permiten generar texto (ej., borradores de correos, resúmenes de texto). Este contenido, antes de ser utilizado o guardado, podría residir temporalmente en el estado global.
Cómo se actualizará y accederá al estado desde diferentes componentes: La forma en que se actualiza y accede al estado dependerá del patrón o librería de gestión de estado elegida:
• Actualización:
    ◦ Acciones/Despachos: En patrones como Redux o Vuex, los componentes "despachan" (dispatch) acciones. Estas acciones son objetos que describen lo que sucedió (ej., USER_LOGIN_SUCCESS, FETCH_PATIENTS_START, FETCH_PATIENTS_SUCCESS). Los "reducers" (en Redux) o "mutations" (en Vuex) son funciones puras que toman el estado actual y una acción, y devuelven un nuevo estado, asegurando un flujo de datos unidireccional y predecible.
    ◦ Actualizadores de estado: Para Context API + useReducer, se utilizan funciones dispatch para enviar acciones al reducer. Para estados locales o de menor alcance, se usarán funciones setState de React o ref/reactive de Vue.
• Acceso:

    ◦ Selectores/Hooks: Los componentes accederán a porciones específicas del estado global mediante "selectores" (en Redux) o "hooks personalizados" (en React con Context/Zustand), o mediante funciones mapState o useStore (en Vuex/Pinia). Esto les permite "suscribirse" solo a la parte del estado que les interesa y re-renderizarse solo cuando esa parte cambia, optimizando el rendimiento.

7. Consumo de APIs del Backend

El Dashboard de Escritorio se comunica con el "Servidor de Aplicación (Backend)" a través de APIs RESTful. Este backend está alojado en Google Cloud Platform (GCP) y maneja la lógica de negocio, la interacción con la base de datos y otros servicios.

Estrategia para realizar las llamadas a los endpoints del backend desde el frontend: Se recomienda una estrategia que garantice la robustez, mantenibilidad y seguridad de las comunicaciones:

• Módulo de Servicios Dedicado: En lugar de realizar llamadas fetch o axios directamente desde los componentes, se creará un módulo de servicios o una capa de abstracción para las APIs. Este módulo contendría funciones específicas para cada endpoint (ej., api.getPatients(), api.createPatient(data), api.login(credentials)).
    ◦ Librería HTTP: Se puede utilizar fetch (API nativa del navegador) o una librería popular como axios. axios es preferible por su simplicidad, manejo automático de JSON, intercepción de solicitudes/respuestas y mejor manejo de errores.
• Endpoints: Se deben desarrollar endpoints de API robustos para las diferentes funcionalidades del Dashboard, como:
    ◦ GET /patients con parámetros de paginación, búsqueda y filtrado.
    ◦ POST /patients para crear nuevos registros.
    ◦ PUT /patients/{id} para actualizar.
    ◦ DELETE /patients/{id} para el borrado suave (soft delete).
    ◦ Endpoints específicos para la galería de archivos y las interacciones con el asistente virtual de IA.
    ◦ POST /api/v1/auth/login para la autenticación.

Cómo se manejarán los estados de carga (loading), error (error), y éxito (data) de las llamadas a la API en la interfaz de usuario: La gestión visual de estos estados es crucial para una buena experiencia de usuario:

• Estado de Carga (loading):
    ◦ Indicadores Visuales: Durante las solicitudes a la API, la UI mostrará indicadores de carga (ej., spinners, esqueletos de contenido, o deshabilitación de botones) para comunicar al usuario que la operación está en curso y que el sistema no está "congelado". Esto es especialmente relevante cuando hay "problemas de conectividad" o el "servidor no puede comunicarse".
• Estado de Error (error):
    ◦ Mensajes Clares y Concisos: En caso de fallos en la API (ej., credenciales incorrectas, campos vacíos, o errores de servidor), se mostrarán "mensajes de error claros y concisos" al usuario. Estos mensajes pueden ser notificaciones flotantes, textos debajo de los campos de formulario, o alertas modales.
    ◦ Registro en Logs: Los errores del frontend también podrían registrarse en el "Servicio de Logs de Auditoría" si son críticos, para monitorear problemas de rendimiento o seguridad.
• Estado de Éxito (data):
    ◦ Actualización Reactiva de la UI: Una vez que una llamada a la API es exitosa, los datos recibidos se utilizarán para actualizar el estado global o local pertinente. Esto provocará que los componentes relevantes se re-rendericen automáticamente, mostrando la información actualizada (ej., una nueva entrada en el historial clínico, un nuevo paciente en la tabla, o un cambio de estado en una cita).

Cómo se gestionarán los tokens de autenticación: La seguridad de las credenciales es una "preocupación de seguridad de datos" clave. El proceso de login autentica al usuario y aplica el control de acceso basado en roles. La comunicación se realiza mediante HTTPS (TLS 1.3) para cifrar los datos en tránsito.

• Generación del Token: Tras un login exitoso, el backend emitirá un token de autenticación (probablemente un JSON Web Token o JWT, que es un estándar de la industria). Este token representará la sesión del usuario y sus permisos de rol.
• Almacenamiento Seguro en el Frontend: La forma de almacenar el token es crítica para la seguridad:
    ◦ sessionStorage: Más seguro que localStorage porque los datos se borran cuando la sesión de la pestaña se cierra. Adecuado para sesiones más cortas.
    ◦ localStorage: Permite persistir la sesión entre cierres del navegador, lo que mejora

la conveniencia del usuario. Sin embargo, es más vulnerable a ataques XSS (Cross-Site Scripting). Si se usa, es esencial tener estrictas medidas de seguridad XSS y considerar tiempos de expiración cortos para los tokens.
    ◦ Cookies HTTP-Only: Considerado el método más seguro para gestionar sesiones, especialmente si se usa con Secure y SameSite flags. El token se envía automáticamente en cada solicitud al backend sin que el JavaScript del frontend pueda acceder a él directamente, mitigando riesgos de XSS. El backend sería responsable de establecer estas cookies.
• Inclusión en los Headers de las Solicitudes a la API: Para cada solicitud subsiguiente al login que requiera autenticación, el token se incluirá en el encabezado Authorization como un token Bearer (ej., Authorization: Bearer <your_token>). Esto permitirá al backend verificar la identidad y los permisos del usuario antes de procesar la solicitud, garantizando el "control de acceso basado en roles".

8. Implementación de Funcionalidades de UI Complejas:
La interfaz del Dashboard de Escritorio, diseñada bajo la estética "Glassmorphism Oscuro", no es solo visualmente atractiva, sino que también integra lógicas complejas para ofrecer una experiencia de usuario robusta y eficiente. La implementación de estas funcionalidades requiere una atención meticulosa a la interacción entre el frontend (desarrollado con frameworks como React o Vue.js y empaquetado con Electron.js) y el backend (Python/Flask/Django o Node.js/Express en GCP).
• Tabla de Datos Interactiva (ej. Pacientes Recientes, Citas del Día):
    ◦ Estructura y Elementos: La tabla de datos principal muestra información crítica como "ID/Nombre del Paciente", "Fecha de Última Consulta", "Diagnóstico Principal", "Médico Tratante" y un "Checkbox de Selección" por fila. También incluye un "Menú de Acciones" contextual por cada fila con opciones como "Ver Detalles", "Editar Registro", "Ver Historial Clínico Completo" o "Archivar/Eliminar (Suave)". La cabecera incluye un "Checkbox maestro" para selección/deselección de todo.
    ◦ Paginación y Ordenamiento: Aunque no se especifica explícitamente en los detalles de la tabla, para manejar eficientemente la información en un "Núcleo Administrativo Central", es común que se implemente paginación. El ordenamiento podría realizarse en el frontend para conjuntos de datos pequeños, pero para grandes volúmenes de pacientes o citas, sería más eficiente delegar la lógica de ordenamiento (y filtrado complejo) al backend a través de parámetros en las peticiones API (ej., GET /pacientes?page=1&sort=nombre&order=asc).
    ◦ Filtrado: La tabla cuenta con un "Icono de filtro" que despliega un panel con "opciones avanzadas para filtrar la tabla (por fecha, por estado, por tipo de consulta, etc.)". La lógica de filtrado se implementará en el frontend para construir dinámicamente los parámetros de consulta que se enviarán al backend a través de APIs RESTful. El backend procesará estos filtros contra la "Base de Datos Relacional (Cloud SQL)" para devolver solo los datos relevantes.
    ◦ Selección Múltiple y Acciones en Lote: Los checkboxes permiten la selección múltiple. La lógica del frontend detectará los registros seleccionados y habilitará botones de acción en lote (ej., "Eliminar" para "borrado suave"). Al activar estas acciones, se enviará una petición API al backend con los IDs de los registros afectados, y el backend ejecutará la lógica de borrado suave (marcando is_deleted = true en la base de datos en lugar de eliminar permanentemente).
    ◦ Menú de Acciones por Fila: Este menú contextual se mostrará al hacer clic en un "Icono de tres puntos". La lógica del frontend gestionará la visibilidad y las acciones asociadas a cada ítem del menú (ej., redirigir a una vista de detalle, abrir un formulario de edición).
• Interacción con el Asistente Virtual (Gemma 3):
    ◦ Mecanismo de Interacción: El usuario interactúa con el chat interno a través de un "campo de entrada de texto". Al enviar el texto, la aplicación de escritorio (frontend) enviará esta consulta de usuario al Servidor de Aplicación (backend) a través de una API segura.
    ◦ Procesamiento y Respuesta: El backend, donde se integra el modelo de IA "Gemma 3" (o similar a Gemini) mediante APIs, procesará la consulta del usuario. La IA generará una respuesta (ej., resúmenes de texto, redacción de informes, búsqueda inteligente de archivos). El backend devolverá esta respuesta al frontend, que la mostrará en tiempo real o casi real en el "área de visualización de conversación" del chat. Es importante recordar que este chat "solo funciona con texto y no genera archivos de ningún formato".
    ◦ Automatización de Tareas: El Asistente Virtual también está "conectado a las APIs

del sistema para ejecutar acciones como enviar correos, generar borradores de documentos, crear proyectos, etc.". Esto implica que, además de la interacción textual, la IA puede invocar otras funcionalidades del backend basadas en la intención del usuario.
• Implementación del Visor de Archivos Visuales:
    ◦ Funcionalidad: Este widget permite visualizar imágenes y videos relacionados con los pacientes. El contenido se actualiza contextualmente según el paciente seleccionado o la búsqueda realizada.
    ◦ Carga de Miniaturas y Vista Ampliada: Las miniaturas se cargarán de forma eficiente para el "Carrusel de Vistas Previas". Al seleccionar una miniatura, se cargará la vista ampliada del archivo. Los archivos digitales (PDF, imágenes, videos cortos) se almacenan en "Cloud Storage (S3/Blob) en GCP", y "estarán cifrados en reposo". La recuperación de estos archivos desde Cloud Storage se hará a través de APIs del backend.
    ◦ Navegación y Filtros: Las flechas del carrusel (< y >) permitirán la navegación entre archivos. Se implementarán filtros (ej., "Cloud Drives", "Photos", "Videos") para refinar la visualización. La lógica del frontend aplicará estos filtros a la lista de archivos obtenida del backend, o bien el backend los aplicará a nivel de base de datos/almacenamiento para optimizar la carga.
• Integración de Widgets de Servicios Externos (Gmail, Google Drive, Office):
    ◦ Nivel de Integración: El Dashboard incluye "widgets para acceso rápido a Gmail y Google Drive, así como accesos directos para abrir aplicaciones de Office". Dada la descripción, la integración inicial se centrará en la conveniencia:
        ▪ Gmail y Google Drive: Los widgets probablemente abrirán las respectivas aplicaciones web en el navegador predeterminado del usuario o en una ventana web integrada si Electron.js lo permite de forma segura. Si se requiere una integración más profunda (ej. listar correos, buscar archivos directamente en el Dashboard), se explorará el uso de las APIs de Google Workspace (como Gmail API, Google Drive API). Esto requeriría una gestión de OAuth 2.0 y tokens de acceso para la autenticación del usuario.
        ▪ Aplicaciones de Office: Los "accesos directos" se interpretan como la capacidad de lanzar las aplicaciones de escritorio de Office (Word, Excel, PowerPoint) instaladas localmente en el sistema operativo del usuario, o abrir documentos de Office en la web si el usuario lo tiene configurado. No se implica una integración profunda a nivel de contenido o edición dentro del Dashboard.
    ◦ Seguridad: Cualquier integración más allá de un simple enlace deberá asegurar la comunicación a través de HTTPS/TLS 1.3 y manejar las credenciales de forma segura, respetando las políticas de privacidad y los permisos de acceso del usuario.
• Manejo de Formularios (ej. Creación de Perfil de Paciente):
    ◦ Validación del Lado del Cliente: Antes de enviar los datos al backend, el frontend implementará validaciones para asegurar que "campos obligatorios" no estén vacíos (ej., "Campo 'Nombre' requerido") y que el "formato" de los datos sea correcto (ej., "Formato de email inválido"). Esto se realizará mediante lógica JavaScript/TypeScript en el framework de frontend (React/Vue.js).
    ◦ Validación del Lado del Servidor: Aunque se realice una validación inicial en el frontend, el backend siempre replicará y reforzará estas validaciones para garantizar la integridad y seguridad de los datos antes de persistirlos en la base de datos.
    ◦ Retroalimentación al Usuario: Los "mensajes de validación" se mostrarán claramente en la interfaz de usuario, guiando al personal sobre cómo corregir los errores en los datos ingresados.
9. Consideraciones de Rendimiento y Optimización del Frontend:
Dado que el Dashboard de Escritorio es el "Núcleo Administrativo Central", su rendimiento y capacidad de respuesta son fundamentales para la eficiencia operativa del consultorio. Se implementarán diversas estrategias de optimización:
• Optimización de la Carga Inicial:
    ◦ Code Splitting (División de Código): Se dividirá el "bundle" (paquete) de JavaScript, CSS y otros activos en módulos más pequeños. Esto permite que el navegador solo cargue el código necesario para la vista actual, reduciendo el tiempo de carga inicial. Por ejemplo, los módulos de "Administración" o "Reportes", que pueden no ser utilizados por todos los roles o con tanta frecuencia, pueden cargarse de forma diferida.
    ◦ Lazy Loading (Carga Perezosa) de Componentes no Críticos: Los componentes de la interfaz de usuario que no son visibles de inmediato al cargar el Dashboard (ej., modales, secciones ocultas, widgets menos utilizados) se cargarán solo cuando sean necesarios. Esto mejora el tiempo de "Time to Interactive" (TTI), haciendo que la aplicación sea utilizable más rápidamente.
    ◦ Minificación y Compresión: Todos los archivos de código y activos se minificarán

(eliminando espacios en blanco y caracteres innecesarios) y se comprimirán (ej., con Gzip o Brotli) antes de su despliegue para reducir el tamaño de la transferencia de datos.
    ◦ Uso Eficiente de CDN (Content Delivery Network): Aunque no se menciona explícitamente, los activos estáticos del frontend (imágenes, CSS, JS) podrían distribuirse a través de una CDN para reducir la latencia de carga para los usuarios, aprovechando los servidores más cercanos geográficamente.
• Optimización del Renderizado de Listas Grandes:
    ◦ Virtualización de Listas/Tablas: Aunque las secciones como "Pacientes Recientes" o "Citas del Día" pueden no tener miles de filas inicialmente, para asegurar la escalabilidad a medida que el consultorio crezca, se considerará la implementación de la virtualización de listas. Esto implica renderizar solo las filas de la tabla que son visibles en la ventana de visualización del usuario, en lugar de renderizar todas las filas a la vez. Esto reduce significativamente el uso de memoria y mejora el rendimiento del scroll para conjuntos de datos extensos.
    ◦ Memoización de Componentes: En frameworks como React o Vue.js, se utilizará la memoización (ej., React.memo o Vue.js keep-alive) para evitar el re-renderizado innecesario de componentes que no han cambiado sus propiedades (props) o estado.
• Manejo Eficiente de Imágenes y Videos:
    ◦ Optimización de Formatos y Compresión: Los archivos de imágenes y videos se almacenarán en "Cloud Storage" en formatos web optimizados (ej., WebP para imágenes, MP4 con codecs eficientes para videos) y se comprimirán adecuadamente para reducir su tamaño sin comprometer excesivamente la calidad.
    ◦ Carga Adaptativa (Responsive Images/Videos): Las imágenes y videos se servirán en diferentes resoluciones o calidades, eligiendo la más apropiada según el tamaño de la pantalla del usuario y la velocidad de su conexión a internet.
    ◦ Streaming y Carga Diferida de Videos: Los videos se cargarán mediante streaming, permitiendo que la reproducción comience antes de que se descargue el archivo completo. Para la funcionalidad del "Visor de Archivos Visuales", los videos solo se cargarán y reproducirán cuando el usuario interactúe con ellos.
    ◦ Caché del Navegador: Se configurarán encabezados de caché HTTP adecuados para los activos multimedia, permitiendo que el navegador los almacene localmente y los recupere más rápidamente en visitas posteriores.
• Requisito de Conectividad: Es fundamental enfatizar que "se recomienda una conexión a internet estable de al menos 250 Mbps para un rendimiento óptimo del ecosistema", ya que el sistema "depende de la comunicación constante con la infraestructura en la nube". Esta es una consideración clave para la expectativa del usuario sobre el rendimiento.
Estas implementaciones técnicas y estrategias de optimización garantizarán que el Dashboard de Escritorio sea no solo seguro y funcional, sino también rápido, fluido y escalable, proporcionando una experiencia de usuario de alta calidad.

10. Consideraciones Específicas de Electron
El "Núcleo Administrativo Central" o Dashboard de Escritorio del "Ecosistema Digital Inteligente para Cirugía Especial" es una aplicación de escritorio. Para su desarrollo, el frontend se construirá utilizando frameworks web como React o Vue.js, y se empaquetará con Electron.js. Electron.js es una tecnología que permite construir aplicaciones de escritorio multiplataforma (compatibles con Windows y macOS) utilizando tecnologías web como HTML, CSS y JavaScript.
Esto implica varias consideraciones específicas:
• Manejo de funcionalidades específicas de escritorio: Aunque la mayoría de la información crítica y los archivos se almacenarán en Cloud Storage con cifrado en reposo, minimizando la necesidad de acceso a archivos locales, Electron.js proporciona la capacidad de interactuar con el sistema operativo nativo. Esto podría ser útil para funcionalidades futuras, como la gestión de impresoras o la integración con software local si fuera necesario. Sin embargo, el enfoque principal de almacenamiento es la nube para seguridad y centralización.
• Comunicación entre procesos de Electron: Una aplicación Electron se compone de un proceso principal (que maneja la ventana y las interacciones del sistema operativo) y procesos de renderizado (donde se ejecuta la interfaz de usuario web) [No se especifica explícitamente en las fuentes, pero es inherente a Electron]. Una implementación efectiva debe asegurar una comunicación fluida y eficiente entre estos procesos, especialmente cuando la interfaz de usuario necesita invocar funcionalidades del sistema o recibir información de ellas.
• Proceso de empaquetado y distribución: El uso de Electron requiere un proceso para

empaquetar la aplicación final en formatos ejecutables para Windows (.exe) y macOS (.dmg) [No se especifica explícitamente en las fuentes, pero es un paso necesario en el desarrollo de Electron]. Esto incluye la configuración de instaladores y el manejo de dependencias.
• Estrategia de actualizaciones de la aplicación de escritorio: Para mantener el sistema seguro y actualizado, es fundamental definir una estrategia clara para la distribución de nuevas versiones y parches [No se especifica explícitamente en las fuentes, pero es una buena práctica para aplicaciones de escritorio]. Electron admite mecanismos de auto-actualización que pueden simplificar este proceso para el usuario final.
La elección de Electron subraya el compromiso con una aplicación de escritorio robusta que aprovecha la agilidad del desarrollo web, facilitando la integración con el backend en la nube.
Siguiente paso: Podríamos explorar las ventajas y desventajas de Electron frente a un desarrollo nativo puro para aplicaciones de escritorio, en términos de costos de desarrollo y rendimiento, para una visión más completa.
11. Diseño Responsivo y Adaptabilidad
El diseño del Dashboard de Escritorio está profundamente anclado en el estilo visual "Glassmorphism Oscuro", que no es solo una cuestión estética, sino funcional, diseñada para ofrecer comodidad visual y una jerarquía de información clara.
Aunque el Dashboard es una aplicación de escritorio y no requiere un "diseño responsivo" en el sentido de adaptarse a pantallas de teléfonos móviles o tabletas (ya que no es su propósito principal), sí debe ser adaptable a diferentes resoluciones de monitor dentro del entorno de escritorio.
Esto se logra mediante la aplicación de los principios del Glassmorphism Oscuro y las buenas prácticas de diseño de interfaz de usuario (UI/UX):
• Coherencia Visual Integral: El diseño del Dashboard mantiene una coherencia visual con la pantalla de Login y el resto del ecosistema, asegurando una experiencia de usuario fluida y de alta calidad desde el primer punto de interacción.
• Comodidad Visual y Legibilidad Óptima: El fondo principal de gris carbón suave (#121212) está diseñado para reducir la fatiga visual, evitando el negro puro. El texto y los iconos son nítidos y brillantes, utilizando el color principal de texto blanco roto (#F5F5F5) para una alta legibilidad sobre el vidrio oscuro, cumpliendo con una relación de contraste superior a 4.5:1 (WCAG). Esta legibilidad es crucial para mantener la usabilidad en diversas configuraciones de pantalla de escritorio.
• Jerarquía de Información Clara: Los paneles flotantes de "vidrio esmerilado" y el enfoque multicapa (eje Z) crean una sensación de profundidad que ayuda a organizar y jerarquizar la información de manera intuitiva. Esta clara separación de elementos ayuda a que la interfaz se escale visualmente sin perder su estructura o la importancia de la información.
• Uso de CSS Variables: La implementación utilizará Propiedades Personalizadas de CSS para el sistema de color y tipografía. Esto facilita la mantenibilidad y asegura una coherencia visual integral, lo que indirectamente apoya la adaptabilidad, ya que los cambios en el tamaño o la disposición de los elementos pueden ajustarse de manera centralizada.
• Estrategias Implícitas para Layout: Aunque no se especifican directamente técnicas como flexbox o grid para el layout en las fuentes, el concepto de paneles flotantes y la necesidad de una "jerarquía espacial clara" sugieren que se emplearán principios de diseño de interfaz de usuario que permitan que los elementos se distribuyan y redimensionen de manera armoniosa en el espacio disponible, manteniendo el confort visual y la funcionalidad en diferentes tamaños de ventana de escritorio.
En resumen, la "adaptabilidad" del Dashboard se centra en mantener una experiencia de usuario consistente, cómoda y legible a través de las variaciones de tamaño de pantalla típicas de un entorno de escritorio, utilizando las propiedades inherentes del diseño Glassmorphism Oscuro y las buenas prácticas de desarrollo de UI.

Agency: dogma.black
Autor: Salvador Aguilar
Fecha: Junio 2025
Proyecto: EcosSecial.
Fase 1/3 Dashboard desktop.

Cliente:

El Dr. Joel Sánchez García, es médico Cirujano, especialista en Traumatología y Ortopedia por el Hospital Central del Sur de Concentración Nacional de Alta Especialidad de Petróleos Mexicanos, con Subespecialidad en Patología Raquídea y Cirugía de la Columna Vertebral por el Hospital Trípode, Universidad Bordeaux II, Bordeaux, Francia.

Con más de 25 años de experiencia en el tratamiento de padecimientos degenerativos, congénitos, metabólicos, traumáticos y tumorales, ha realizado más de 7 mil procedimientos quirúrgicos de columna vertebral.

Es miembro certificado de la Asociación Mexicana de Cirujanos de Columna; miembro titular del Colegio Mexicano de Ortopedia y Traumatología; miembro del Consejo Mexicano de Ortopedia y Traumatología; miembro de la North American Spine Society; miembro de AO Spine y AO Alumni de la Asociación para el Estudio de la Fijación Interna; miembro fundador de la Sociedad Iberoamericana para el Estudio de la Patología Vertebro-medular; entre otras asociaciones.

Diseño Completo de Endpoints de la API (RESTful)
La API del backend será el puente entre la aplicación de escritorio y la infraestructura en la nube, permitiendo la gestión de toda la información del consultorio. Se diseñará siguiendo los principios de RESTfulness para garantizar la coherencia, la escalabilidad y la facilidad de uso.
Para Gestión de Pacientes
El módulo de Gestión Integral del Expediente del Paciente es crucial para centralizar toda la información crítica y permitir un acceso inmediato y un seguimiento eficiente. Los siguientes endpoints facilitarán estas operaciones:

- 1. GET /patients
    - Método HTTP: GET
    - Ruta exacta: /api/v1/patients
    - Descripción: Permite obtener una lista de todos los pacientes registrados en el sistema. Soporta parámetros para búsqueda, filtrado, paginación y ordenamiento, facilitando la "Búsqueda Avanzada". Los filtros pueden incluir fecha, estado o tipo de consulta.
- 2. GET /patients/{id}
    - Método HTTP: GET
    - Ruta exacta: /api/v1/patients/{id} (donde {id} es el identificador único del paciente)
    - Descripción: Recupera los detalles completos de un paciente específico, incluyendo su perfil y la información general de su historial. Esto es útil para la función "Ver Detalles" o para abrir la ficha completa del paciente.
- 3. POST /patients
    - Método HTTP: POST
    - Ruta exacta: /api/v1/patients
    - Descripción: Crea un nuevo registro de paciente en el sistema con todos sus datos demográficos y de contacto.
- 4. PUT /patients/{id}
    - Método HTTP: PUT
    - Ruta exacta: /api/v1/patients/{id}
    - Descripción: Actualiza los datos de un paciente existente identificado por su ID. Permite modificar cualquier campo del perfil del paciente.
- 5. DELETE /patients/{id}

    - Método HTTP: DELETE
    - Ruta exacta: /api/v1/patients/{id}
    - Descripción: Realiza un "borrado suave" (soft delete) de un paciente. Esto significa que el registro no se elimina permanentemente de la base de datos, sino que se marca como inactivo o archivado (ej., is_deleted = true). El sistema está diseñado para que el registro se marque como inactivo, no se borre permanentemente.
- 6. POST /patients/{id}/archive
    - Método HTTP: POST
    - Ruta exacta: /api/v1/patients/{id}/archive
    - Descripción: Archiva un paciente específico por su ID. Esta acción complementa el borrado suave, marcando el paciente como archivado o inactivo. Esta funcionalidad puede aplicarse a múltiples pacientes seleccionados para acciones en lote.

Documento: Arq. Backend y APIs
Diseño Completo de Endpoints de la API (RESTful) (Continuación)
La API del backend continúa su diseño siguiendo los principios de RESTfulness, garantizando la coherencia, la escalabilidad y la facilidad de uso para todas las operaciones del "Ecosistema Digital Inteligente para Cirugía Especial". Los datos estructurados se almacenan en la Base de Datos Relacional (Cloud SQL), mientras que los archivos digitales residen en Cloud Storage, ambos con cifrado en reposo para asegurar la confidencialidad de la información del paciente. Todas las interacciones con la API se realizarán de forma segura a través de HTTPS (TLS 1.3).
Para Gestión de Historial Clínico
El registro cronológico y detallado del historial clínico es una funcionalidad central para el seguimiento eficiente del paciente. Los siguientes endpoints permiten la interacción con esta información:
- 1. GET /patients/{id}/history
    - Método HTTP: GET
    - Ruta exacta: /api/v1/patients/{id}/history (donde {id} es el identificador único del paciente)
    - Descripción: Permite obtener el historial clínico completo de un paciente específico, recuperando todas las entradas cronológicas de consultas, diagnósticos, notas de evolución, procedimientos y tratamientos asociados.
- 2. POST /patients/{id}/history
    - Método HTTP: POST
    - Ruta exacta: /api/v1/patients/{id}/history
    - Descripción: Permite agregar una nueva entrada al historial clínico de un paciente específico. Esta acción es el equivalente digital a registrar una nueva consulta o procedimiento, incluyendo campos como fecha de consulta, diagnóstico y notas.
- 3. PUT /history/{id}
    - Método HTTP: PUT
    - Ruta exacta: /api/v1/history/{id} (donde {id} es el identificador único de una entrada específica del historial)
    - Descripción: Actualiza los datos de una entrada existente en el historial clínico, permitiendo la corrección o modificación de detalles como diagnósticos o notas.
- 4. DELETE /history/{id}
    - Método HTTP: DELETE
    - Ruta exacta: /api/v1/history/{id}
    - Descripción: Realiza un "borrado suave" de una entrada específica del historial clínico por su ID. Esto significa que el registro se marca como inactivo o archivado en la base de datos, en lugar de ser eliminado permanentemente, preservando la integridad histórica de los datos.
Para Gestión de Citas
El módulo de agenda y citas es fundamental para optimizar la programación y el control del flujo de pacientes, reduciendo los procesos manuales. Los endpoints asociados son:
- 1. GET /appointments
    - Método HTTP: GET
    - Ruta exacta: /api/v1/appointments
    - Descripción: Obtiene la lista de todas las citas programadas en el consultorio. Soporta parámetros para filtrar por fecha (ej., "Citas del Día"), por paciente o por estado (ej., "Confirmada", "Cancelada", "Atendida"), así como paginación y ordenamiento.
- 2. GET /appointments/{id}
    - Método HTTP: GET

    ◦ Ruta exacta: /api/v1/appointments/{id} (donde {id} es el identificador único de la cita)
    ◦ Descripción: Recupera los detalles completos de una cita específica por su ID, incluyendo el paciente asociado, fecha, hora, tipo y estado.
• 3. POST /appointments
    ◦ Método HTTP: POST
    ◦ Ruta exacta: /api/v1/appointments
    ◦ Descripción: Crea una nueva cita en el calendario del consultorio. Requiere la información necesaria como el paciente, la fecha, la hora y el tipo de cita.
• 4. PUT /appointments/{id}
    ◦ Método HTTP: PUT
    ◦ Ruta exacta: /api/v1/appointments/{id}
    ◦ Descripción: Actualiza los datos de una cita existente identificada por su ID. Permite modificar su fecha, hora, tipo, o cambiar su estado (ej., de "Programada" a "Confirmada" o "Cancelada").
• 5. DELETE /appointments/{id}
    ◦ Método HTTP: DELETE
    ◦ Ruta exacta: /api/v1/appointments/{id}
    ◦ Descripción: Realiza un "borrado suave" de una cita por su ID. El registro de la cita se marca como inactivo en la base de datos, lo que permite mantener un historial completo de las citas incluso si son canceladas o no se realizan.

Para Gestión Documental

La gestión documental avanzada centraliza y asegura el almacenamiento de todos los archivos digitales del paciente, eliminando la dispersión y mejorando la seguridad. Los archivos se almacenan en Cloud Storage con cifrado en reposo.

• 1. POST /patients/{id}/documents
    ◦ Método HTTP: POST
    ◦ Ruta exacta: /api/v1/patients/{id}/documents
    ◦ Descripción: Permite subir un nuevo archivo digital (PDF, imagen, video corto, etc.) y asociarlo directamente al expediente de un paciente específico. El archivo se cargará a Cloud Storage y sus metadatos se registrarán en la base de datos.
• 2. GET /patients/{id}/documents
    ◦ Método HTTP: GET
    ◦ Ruta exacta: /api/v1/patients/{id}/documents
    ◦ Descripción: Obtiene una lista de todos los documentos digitales (metadatos como nombre, tipo, URL) asociados a un paciente específico.
• 3. GET /documents/{id}
    ◦ Método HTTP: GET
    ◦ Ruta exacta: /api/v1/documents/{id} (donde {id} es el identificador único del documento)
    ◦ Descripción: Recupera un documento específico por su ID. Esta acción puede devolver una URL pre-firmada o un stream de datos que permita la descarga o visualización directa del archivo desde Cloud Storage.
• 4. DELETE /documents/{id}
    ◦ Método HTTP: DELETE
    ◦ Ruta exacta: /api/v1/documents/{id}
    ◦ Descripción: Realiza un "borrado suave" de un documento digital por su ID. Solo se marca el registro del documento como inactivo en la base de datos, sin eliminar el archivo físico de Cloud Storage de forma inmediata, manteniendo así un registro auditable.

Documento: Arq. Backend y APIs

Diseño Completo de Endpoints de la API (RESTful) (Continuación)

La definición de los endpoints RESTful es crucial para la interacción segura y eficiente entre el Dashboard de Escritorio (Núcleo Administrativo Central) y el backend en la nube (Google Cloud Platform). Estos endpoints facilitan la autenticación de usuarios, la gestión de permisos, el registro de actividades para auditoría y la integración de capacidades de Inteligencia Artificial para automatización y asistencia. Todas las comunicaciones se realizarán a través de HTTPS (TLS 1.3), garantizando el cifrado de datos en tránsito.

Para Autenticación y Autorización

La autenticación y la autorización son la base de la seguridad del "Ecosistema Digital Inteligente para Cirugía Especial", asegurando que solo el personal autorizado acceda a la información y funcionalidades.

- 1. POST /login
  - Método HTTP: POST
  - Ruta exacta: /api/v1/auth/login
  - Descripción: Este endpoint es el punto de entrada para que un usuario inicie sesión en el sistema. Recibe las credenciales (nombre de usuario y contraseña), las valida contra la base de datos, y si son correctas, devuelve un token de autenticación que será utilizado para futuras solicitudes. Es fundamental para aplicar el control de acceso basado en roles. La comunicación se realiza cifrada vía HTTPS (TLS 1.3).
- 2. Endpoints para Gestión de Usuarios
  - La gestión de usuarios es una funcionalidad clave del "Núcleo Administrativo Central", accesible para roles como "Administrador" y "Admin Principal".
  - GET /users
    - Método HTTP: GET
    - Ruta exacta: /api/v1/users
    - Descripción: Obtiene una lista de todos los usuarios registrados en el sistema. Este endpoint permite a los administradores visualizar el personal del consultorio y sus roles asociados.
  - GET /users/{id}
    - Método HTTP: GET
    - Ruta exacta: /api/v1/users/{id} (donde {id} es el identificador único del usuario)
    - Descripción: Recupera los detalles completos del perfil de un usuario específico, incluyendo su información de contacto y el rol asignado.
  - POST /users
    - Método HTTP: POST
    - Ruta exacta: /api/v1/users
    - Descripción: Permite crear una nueva cuenta de usuario en el sistema. Esta operación incluye la asignación de un rol inicial (ej., "Asistente", "Admin Secundario").
  - PUT /users/{id}
    - Método HTTP: PUT
    - Ruta exacta: /api/v1/users/{id}
    - Descripción: Actualiza los datos de un usuario existente, como su nombre, información de contacto o, fundamentalmente, su rol en el sistema.
  - DELETE /users/{id}
    - Método HTTP: DELETE
    - Ruta exacta: /api/v1/users/{id}
    - Descripción: Realiza un "borrado suave" de una cuenta de usuario por su ID. El registro del usuario se marca como inactivo en la base de datos, manteniendo la integridad histórica y facilitando posibles recuperaciones.
- 3. Endpoints para Gestión de Roles
  - El sistema cuenta con un robusto "Sistema de Roles", con roles como "Admin Principal", "Admin Secundario", "Asistente", "Invitado (Solo Lectura)". La información de roles se almacena en la tabla ROLES.
  - GET /roles
    - Método HTTP: GET
    - Ruta exacta: /api/v1/roles
    - Descripción: Obtiene una lista de todos los roles de usuario definidos en el sistema, junto con sus nombres y las capacidades generales asociadas.
  - GET /roles/{id}
    - Método HTTP: GET
    - Ruta exacta: /api/v1/roles/{id} (donde {id} es el identificador único del rol)
    - Descripción: Recupera los detalles específicos y el conjunto de permisos detallados asociados a un rol particular por su ID.
  - POST /roles
    - Método HTTP: POST
    - Ruta exacta: /api/v1/roles
    - Descripción: Permite la creación de una nueva definición de rol en el sistema, especificando su nombre y los permisos que este tendrá sobre las diferentes funcionalidades y datos.
  - PUT /roles/{id}
    - Método HTTP: PUT
    - Ruta exacta: /api/v1/roles/{id}
    - Descripción: Actualiza el nombre o los permisos de un rol existente. Esta es una

operación crítica que debe ser manejada con extrema precaución.
    - DELETE /roles/{id}
        - Método HTTP: DELETE
        - Ruta exacta: /api/v1/roles/{id}
        - Descripción: Realiza un "borrado suave" de un rol, marcándolo como inactivo. Los roles inactivos no pueden ser asignados a nuevos usuarios, pero su registro se mantiene para auditoría y referencias históricas.
- 4. Verificación de Permisos por Rol
    - Descripción: La verificación de permisos basados en el rol del usuario es una lógica interna del backend que se ejecuta en el Servidor de Aplicación. Después de que un usuario es autenticado y su rol se consulta de la base de datos ROLES, el backend determina si el usuario tiene permiso para acceder a una funcionalidad o dato específico antes de procesar la solicitud. No se expone como un endpoint RESTful dedicado.

Para Logs de Auditoría

El "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)" es fundamental para la seguridad, registrando cronológicamente toda actividad y cambios en el sistema. Estos logs son inmutables (Write Once, Read Many - WORM).

- 1. POST /logs
    - Método HTTP: POST
    - Ruta exacta: /api/v1/logs
    - Descripción: Este endpoint es utilizado internamente por el backend para registrar de forma automática y asíncrona todas las acciones clave realizadas en el sistema. Esto incluye intentos de login (exitosos o fallidos), modificaciones de datos de pacientes, creación/actualización de citas, subida de documentos, y cualquier otra operación crítica que requiera un rastro auditable.
- 2. GET /logs
    - Método HTTP: GET
    - Ruta exacta: /api/v1/logs
    - Descripción: Obtiene un listado cronológico de los registros de auditoría del sistema. Este endpoint es accesible solo para roles con permisos elevados (como el "Admin Principal"), permitiéndoles revisar el historial de actividades, monitorear la seguridad y rastrear operaciones específicas.

Para Integración con IA

La "Inteligencia y Automatización Integrada" utiliza el modelo Gemma 3 (o similar a Gemini) para potenciar el flujo de trabajo del consultorio. La IA se integra directamente en el backend mediante APIs.

- 1. POST /ai/generate-email
    - Método HTTP: POST
    - Ruta exacta: /api/v1/ai/generate-email
    - Descripción: Envía una solicitud al modelo de IA para generar un borrador de correo electrónico. Se proporcionarían parámetros como el propósito del correo, el destinatario y puntos clave a incluir, y la IA devolvería el texto del borrador.
- 2. POST /ai/summarize
    - Método HTTP: POST
    - Ruta exacta: /api/v1/ai/summarize
    - Descripción: Recibe un bloque de texto (ej., notas de consulta, informes) y utiliza el modelo de IA para generar un resumen conciso de dicho contenido.
- 3. POST /ai/chat
    - Método HTTP: POST
    - Ruta exacta: /api/v1/ai/chat
    - Descripción: Permite enviar mensajes de texto al chat interno del sistema, donde el modelo de IA procesa la entrada y devuelve una respuesta textual. Es utilizado para consultas rápidas o para asistir en la redacción de informes. Cabe destacar que este chat funciona únicamente con texto y no genera archivos.
- 4. POST /ai/generate-template
    - Método HTTP: POST
    - Ruta exacta: /api/v1/ai/generate-template
    - Descripción: Envía una solicitud a la IA para generar diversas plantillas, como hojas membretadas, basándose en la información o el formato requerido.
- 5. POST /ai/create-reminder
    - Método HTTP: POST
    - Ruta exacta: /api/v1/ai/create-reminder
    - Descripción: Utiliza la IA para interpretar una solicitud de usuario y generar un

recordatorio dentro del sistema, posiblemente con detalles como fecha, hora y descripción.

Documento: Arq. Backend y APIs
Especificación de Solicitudes y Respuestas
Para asegurar la interoperabilidad y una comunicación fluida y predecible entre el Dashboard de Escritorio (frontend) y el Servidor de Aplicación (backend), es fundamental definir contratos claros para las APIs. Estos contratos especifican la estructura JSON esperada tanto en las solicitudes (Request Body) enviadas al backend como en las respuestas (Response Body) recibidas. Todas las comunicaciones de datos sensibles se realizarán de forma segura a través de HTTPS (TLS 1.3).
1. Estructura JSON para Solicitudes (Request Body)
Las solicitudes al backend generalmente contendrán objetos JSON con los datos necesarios para realizar una operación específica (creación, actualización, etc.). Los ejemplos a continuación ilustran estructuras genéricas:
• Para Autenticación (Ej: POST /api/v1/auth/login)
    ◦ Método: POST
    ◦ Descripción: Envía las credenciales del usuario para iniciar sesión.
    ◦ Estructura JSON:
    ◦ Consideraciones: Las contraseñas se envían como texto plano en el cuerpo de la solicitud, pero la seguridad está garantizada por el cifrado TLS 1.3 en tránsito.
• Para Creación de Recursos (Ej: POST /api/v1/patients)
    ◦ Método: POST
    ◦ Descripción: Crea un nuevo perfil de paciente en el sistema.
    ◦ Estructura JSON:
    ◦ Nota: datos_contacto se almacenaría como JSON en la base de datos.
• Para Actualización de Recursos (Ej: PUT /api/v1/patients/{id})
    ◦ Método: PUT
    ◦ Descripción: Actualiza los datos de un paciente existente.
    ◦ Estructura JSON:
    ◦ Consideraciones: Solo los campos que necesitan ser modificados deben ser incluidos en la solicitud.
• Para Agendamiento de Citas (Ej: POST /api/v1/appointments)
    ◦ Método: POST
    ◦ Descripción: Permite programar una nueva cita.
    ◦ Estructura JSON:
• Para Interacción con IA (Ej: POST /api/v1/ai/generate-email)
    ◦ Método: POST
    ◦ Descripción: Envía datos para que el modelo de IA (Gemma 3 o similar a Gemini) genere un borrador de correo electrónico.
    ◦ Estructura JSON:
    ◦ Nota: Otros endpoints de IA como /ai/summarize o /ai/chat esperarían campos como text_to_summarize o message respectivamente.
2. Estructura JSON para Respuestas (Response Body)
Las respuestas del backend indicarán el resultado de la operación solicitada. Las respuestas de éxito suelen contener los datos del recurso solicitado o modificado.
• Para Recuperación de un Recurso Único (Ej: GET /api/v1/patients/{id})
    ◦ Método: GET
    ◦ Descripción: Devuelve los detalles completos de un paciente específico.
    ◦ Estructura JSON (HTTP 200 OK):
• Para Listado de Recursos con Paginación (Ej: GET /api/v1/patients)
    ◦ Método: GET
    ◦ Descripción: Devuelve una lista de pacientes, con soporte para paginación.
    ◦ Estructura JSON (HTTP 200 OK):
• Para Creación Exitosa (Ej: POST /api/v1/patients)
    ◦ Método: POST
    ◦ Descripción: Confirma la creación exitosa del recurso, incluyendo su identificador único.
    ◦ Estructura JSON (HTTP 201 Created):
• Para Login Exitoso (Ej: POST /api/v1/auth/login)
    ◦ Método: POST
    ◦ Descripción: Devuelve un token de autenticación para futuras solicitudes, junto con información básica del usuario y su rol.
    ◦ Estructura JSON (HTTP 200 OK):

• Para Respuesta de IA (Ej: POST /api/v1/ai/generate-email)
    ◦ Método: POST
    ◦ Descripción: Devuelve el texto generado por el modelo de IA.
    ◦ Estructura JSON (HTTP 200 OK):
3. Manejo de Archivos Grandes (Cloud Storage y URLs)
Es importante destacar que los archivos grandes, como PDFs de historiales, imágenes (ej. RX) y videos cortos, no se enviarán directamente en el cuerpo de las solicitudes o respuestas JSON. En su lugar, se utilizará Google Cloud Storage (equivalente a S3/Blob) para el almacenamiento seguro de estos archivos. Los datos sensibles almacenados en Cloud Storage estarán cifrados en reposo.
• Para Subida de Archivos: La aplicación de escritorio primero enviará el archivo directamente a Cloud Storage. Una vez que el archivo esté en la nube, el backend recibirá una URL (o un identificador) del archivo, que luego se almacenará en la base de datos relacional (Cloud SQL) junto con los metadatos relevantes del paciente o del historial clínico.
• Para Acceso a Archivos: Cuando se necesite visualizar un archivo, el backend proporcionará una URL firmada o un enlace seguro de Cloud Storage en la respuesta JSON. El frontend utilizará esta URL para acceder y mostrar el archivo directamente desde Cloud Storage, evitando cargar el backend con transferencias de grandes volúmenes de datos.

Manejo de Errores
Un manejo de errores estandarizado es vital para la robustez y la facilidad de depuración de la API, proporcionando retroalimentación clara tanto al frontend como a los desarrolladores.
1. Códigos de Estado HTTP Estándar
El backend utilizará códigos de estado HTTP estándar para indicar el resultado de cada solicitud:
• 200 OK: La solicitud fue exitosa. Se utiliza para operaciones GET que devuelven datos, y para PUT/DELETE que resultaron en una modificación/eliminación exitosa.
• 201 Created: La solicitud POST fue exitosa y resultó en la creación de un nuevo recurso. La respuesta incluirá el recurso recién creado o su identificador.
• 400 Bad Request: La solicitud del cliente es incorrecta, malformada o contiene datos inválidos. Esto incluye errores de validación de entrada.
• 401 Unauthorized: La solicitud no ha sido aplicada porque le falta credenciales de autenticación válidas para el recurso solicitado. Esto ocurre si el token de autenticación está ausente, es inválido o ha expirado. También se usa cuando las credenciales de login son incorrectas.
• 403 Forbidden: La solicitud es válida, pero el usuario autenticado no tiene los permisos necesarios para acceder al recurso o realizar la acción solicitada. El control de acceso basado en roles es fundamental aquí.
• 404 Not Found: El recurso solicitado no existe en el servidor. Esto se aplica si se busca un ID de paciente que no existe.
• 500 Internal Server Error: Un error genérico del servidor que impide completar la solicitud. Esto indica un problema inesperado en el backend, no relacionado directamente con la solicitud del cliente.
2. Estructura JSON Estándar para Mensajes de Error
Todos los errores, especialmente los de tipo 4xx y 5xx, se devolverán con una estructura JSON estandarizada para facilitar su procesamiento por parte del frontend:
```
{
  "error": "Mensaje conciso del error",
  "code": 400, // Código de estado HTTP
  "details": "Mensaje detallado para el desarrollador o información de validación"
}
```
• error: Una descripción breve y legible por el usuario del problema.
• code: El código de estado HTTP relevante.
• details: Información adicional que puede ser útil para la depuración o para mostrar validaciones específicas al usuario (ej. lista de campos inválidos).
3. Ejemplos de Uso de Errores
• Escenario: Solicitud con Campos Obligatorios Vacíos (400 Bad Request)
    ◦ Descripción: El usuario intenta crear un paciente sin proporcionar el nombre.
    ◦ Respuesta JSON:
    ◦ Citas:. Mensajes como "Campo 'Nombre' requerido" o "Formato de email inválido".
• Escenario: Credenciales de Login Incorrectas (401 Unauthorized)

    ◦ Descripción: Un usuario intenta iniciar sesión con un nombre de usuario o contraseña incorrectos.
    ◦ Respuesta JSON:
    ◦ Citas:.
• Escenario: Acceso Denegado por Permisos (403 Forbidden)
    ◦ Descripción: Un usuario con rol de "Asistente" intenta acceder a la funcionalidad de "Gestionar Usuarios".
    ◦ Respuesta JSON:
    ◦ Citas: El sistema impone un control estricto de acceso basado en roles.
• Escenario: Recurso No Encontrado (404 Not Found)
    ◦ Descripción: El frontend solicita los detalles de un paciente con un id que no existe en la base de datos.
    ◦ Respuesta JSON:
    ◦ Citas:. Mensajes como "No se encontraron pacientes".
• Escenario: Error Inesperado del Servidor (500 Internal Server Error)
    ◦ Descripción: Un fallo inesperado ocurre en la lógica del backend (ej. un problema con la base de datos, un error en el código no manejado).
    ◦ Respuesta JSON:
    ◦ Citas:. Mensaje sobre "problemas de conexión o indisponibilidad del servicio".

4. Registro de Errores (Logs de Auditoría)

Es crucial que todos los errores, especialmente los errores 4xx y 5xx, sean registrados en el "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)". Estos logs son inmutables (Write Once, Read Many - WORM), lo que garantiza su integridad y son vitales para la detección de anomalías, la respuesta a incidentes y el cumplimiento normativo. Esto permite un monitoreo proactivo de la seguridad y el rendimiento del sistema.

Implementación de Autenticación y Autorización

Para asegurar la seguridad y el control de acceso en la plataforma de Remex 8, se definirá una estrategia robusta de autenticación y autorización. Dado el modelo de negocio multinivel de Remex 8, que implica la gestión de socios y distribuidores, es fundamental proteger la información y las funcionalidades de la API.

• Estrategia de Autenticación de Usuarios:
    ◦ La autenticación se implementará utilizando tokens web JSON (JWT) [información no proveniente de las fuentes]. Después de que un usuario (cliente o distribuidor) inicie sesión exitosamente proporcionando sus credenciales (nombre de usuario/correo electrónico y contraseña), el backend generará un token JWT. Este token encapsulará información relevante sobre el usuario, como su ID y roles, y será firmado digitalmente para garantizar su integridad y autenticidad [información no proveniente de las fuentes].
    ◦ Este token será enviado al cliente (aplicación web o móvil) y deberá ser incluido en las cabeceras de cada solicitud subsiguiente a la API. Esto permitirá al servidor verificar la identidad del usuario sin necesidad de consultar una base de datos en cada petición, optimizando el rendimiento [información no proveniente de las fuentes].
    ◦ La validez de los tokens se gestionará estableciendo un tiempo de expiración limitado, tras el cual el usuario deberá refrescar su sesión o volver a autenticarse [información no proveniente de las fuentes]. Se podría implementar un mecanismo de refresh token para mejorar la experiencia de usuario, permitiendo renovar el token de acceso sin requerir la reintroducción de credenciales, siempre que el refresh token sea válido [información no proveniente de las fuentes].
• Implementación de Autorización (Control de Acceso Basado en Roles - RBAC):
    ◦ La autorización se basará en un modelo de control de acceso basado en roles (RBAC) [información no proveniente de las fuentes]. Cada usuario en el sistema de Remex 8 (por ejemplo, clientes, distribuidores, administradores) tendrá uno o más roles asignados, que definirán sus permisos dentro de la plataforma [información no proveniente de las fuentes].
    ◦ Antes de procesar cualquier solicitud a un endpoint de la API, el backend verificará el token JWT para extraer los roles del usuario. Luego, se comprobará si el rol del usuario tiene los permisos necesarios para acceder a la funcionalidad o a los datos específicos solicitados [información no proveniente de las fuentes]. Por ejemplo, un distribuidor tendrá acceso a funcionalidades relacionadas con su red y ventas, mientras que un cliente solo podrá acceder a información de productos y realizar compras.
    ◦ Esto garantiza que solo los usuarios con los permisos adecuados puedan realizar ciertas acciones (ej., gestionar la red de afiliados, ver reportes de comisiones, acceder

a información de precios de distribuidor) [información no proveniente de las fuentes].
Tecnología del Backend
La elección de la tecnología para el backend es crucial para la escalabilidad, mantenibilidad y eficiencia de la plataforma digital de Remex 8. Considerando que dogma.black se especializa en desarrollo de software y programación, y que se ha mencionado el uso potencial de Modelos de Lenguaje (LLMs) y APIs para automatización, se sugieren las siguientes opciones, las cuales no están explícitamente detalladas en las fuentes de Remex 8:

- Elección del Lenguaje de Programación:
  - Se recomienda Python o Node.js [información no proveniente de las fuentes].
    - Python: Es una excelente opción por su legibilidad, su vasta comunidad y su robusto ecosistema de librerías, especialmente útil si en el futuro se planean integraciones más profundas con análisis de datos o inteligencia artificial (dado el contexto de LLMs mencionados por dogma.black) [4, información no proveniente de las fuentes]. Su versatilidad lo hace apto para el desarrollo rápido y para manejar diversas tareas del backend.
    - Node.js: Ideal para aplicaciones en tiempo real y microservicios, Node.js permite usar JavaScript tanto en el frontend como en el backend, lo que puede agilizar el desarrollo si el equipo ya está familiarizado con JavaScript. Su modelo de E/S no bloqueante es eficiente para aplicaciones con muchas conexiones concurrentes, como una plataforma de redes sociales con alta interacción [1, 2, 4, información no proveniente de las fuentes].
- Elección del Framework de Backend y Justificación:
  - Si se elige Python:
    - Django: Es un framework completo y robusto que incluye muchas funcionalidades "listas para usar", como un ORM (Object-Relational Mapper), un panel de administración y un sistema de autenticación, lo que acelera el desarrollo de aplicaciones complejas y seguras [información no proveniente de las fuentes]. Sería ideal para una plataforma que gestiona tanto productos como una red de distribuidores con diferentes roles y datos.
    - Flask: Si se prefiere una solución más ligera y modular, Flask es una excelente alternativa. Ofrece mayor flexibilidad y control sobre los componentes a utilizar, siendo ideal para construir APIs RESTful y microservicios, permitiendo escalar funcionalidades específicas según la demanda de Remex 8 [información no proveniente de las fuentes].
  - Si se elige Node.js:
    - Express.js: Es el framework más popular y flexible para Node.js, ideal para construir APIs RESTful de manera rápida y eficiente [información no proveniente de las fuentes]. Su naturaleza unopinionated permite diseñar la arquitectura del backend con gran libertad, integrándose fácilmente con bases de datos y otros servicios necesarios para una plataforma de marketing y ventas multinivel [1, 2, 3, información no proveniente de las fuentes].

Estas elecciones se alinean con la necesidad de construir una infraestructura digital sólida que soporte el crecimiento de Remex 8 y las capacidades de automatización exploradas por dogma.black.

Configuración de Servicios de GCP para el Backend
La infraestructura del backend de Remex 8 se apoyará en Google Cloud Platform (GCP) para proveer la escalabilidad, resiliencia y seguridad necesarias. A continuación, se detalla la configuración de los servicios clave:

- Servidor de Aplicación: Cloud Run
  - Para el despliegue del servidor de aplicación, se recomienda el uso de Cloud Run [información no proveniente de las fuentes]. Esta elección se alinea con la flexibilidad y la capacidad de construir APIs RESTful y microservicios mencionadas en la sección de "Tecnología del Backend" [información no proveniente de las fuentes].
  - Propósito en la Arquitectura: Cloud Run permitirá desplegar los servicios del backend como contenedores sin servidor, lo que significa que la infraestructura subyacente será gestionada automáticamente por GCP [información no proveniente de las fuentes]. Esto facilitará la escalabilidad automática basada en la demanda de tráfico, desde cero hasta miles de solicitudes concurrentes, y optimizará los costos al pagar solo por los recursos consumidos [información no proveniente de las fuentes]. Es ideal para alojar la lógica de negocio de la API, procesar las solicitudes de autenticación/autorización y servir los datos a las aplicaciones cliente.
- API Gateway / Load Balancer: Cloud Load Balancing con integración de Cloud Endpoints o API Gateway

    ◦ Para servir como el punto de entrada seguro y único para todas las solicitudes externas a la API, se configurará un Cloud Load Balancer [información no proveniente de las fuentes]. Este balanceador de carga externo distribuirá de manera eficiente el tráfico hacia las instancias del servidor de aplicación en Cloud Run [información no proveniente de las fuentes].
    ◦ Rol en la Arquitectura: Además de la distribución de tráfico, se podría integrar con Cloud Endpoints o API Gateway [información no proveniente de las fuentes]. Esto proporcionaría capacidades adicionales como la gestión de API, monitoreo, control de acceso basado en claves de API, y la imposición de límites de tasa (rate limiting) para proteger el backend de usos indebidos o ataques de denegación de servicio [información no proveniente de las fuentes]. Garantizará que solo las solicitudes válidas y autorizadas lleguen a los servicios internos.
• Almacenamiento de Archivos: Cloud Storage Buckets
    ◦ Los archivos estáticos y multimedia de la plataforma, como las imágenes y videos de alta calidad utilizados en la estrategia de contenido para redes sociales, y otros documentos necesarios, se almacenarán en Cloud Storage Buckets [información no proveniente de las fuentes].
    ◦ Configuración y Consideraciones: Se configurarán buckets dedicados para diferentes tipos de contenido (ej., uno para imágenes de productos, otro para videos de marketing, otro para documentos internos de distribuidores) [información no proveniente de las fuentes].
    ◦ Seguridad y Acceso: Los buckets se configurarán con permisos de acceso específicos, utilizando políticas de IAM (Identity and Access Management) [información no proveniente de las fuentes]. Los archivos públicos (como imágenes de productos para el sitio web o redes sociales) se harán accesibles públicamente con URLs generadas, mientras que los archivos sensibles (ej., documentos de negocio para distribuidores) tendrán acceso restringido y se servirán a través de URLs firmadas temporalmente o mediante el backend una vez que el usuario haya sido autorizado [información no proveniente de las fuentes]. Se implementará versionamiento para la recuperación de archivos y la protección contra eliminaciones accidentales [información no proveniente de las fuentes].
• Monitoreo y Registro: Cloud Logging y Cloud Monitoring
    ◦ Para la recopilación centralizada y el análisis de registros, se utilizará Cloud Logging [información no proveniente de las fuentes].
    ◦ Configuración: Cloud Logging se configurará automáticamente para recopilar los logs del servidor de aplicación (Cloud Run), así como los logs de otros servicios de GCP como el Load Balancer y Cloud SQL [información no proveniente de las fuentes]. Esto permitirá una visibilidad completa del comportamiento del sistema, facilitando la depuración, la identificación de problemas de seguridad o rendimiento, y la auditoría [información no proveniente de las fuentes].
    ◦ Complementariamente, Cloud Monitoring se utilizará para recopilar métricas de rendimiento (uso de CPU, memoria, latencia de solicitudes, errores) y configurar alertas proactivas sobre el estado y la salud de la aplicación y la infraestructura [información no proveniente de las fuentes].
• Conexión a Base de Datos: Cloud SQL
    ◦ El backend se conectará a la base de datos de la plataforma, que estará alojada en Cloud SQL [información no proveniente de las fuentes].
    ◦ Nota: La configuración detallada de Cloud SQL, incluyendo el tipo de base de datos (PostgreSQL, MySQL), la capacidad y las estrategias de respaldo, se abordará en el Documento 2, "Arq. Base de Datos y Gestión de Datos" [información proveniente de la estructura del documento].
• Consideraciones de Red y Seguridad: VPC y Firewall
    ◦ Para garantizar la comunicación segura y controlada entre los diferentes servicios de GCP, se configurará una Virtual Private Cloud (VPC) [información no proveniente de las fuentes].
    ◦ Firewall Rules: Se implementarán reglas de firewall dentro de la VPC para restringir el tráfico entre los servicios y solo permitir las comunicaciones necesarias [información no proveniente de las fuentes]. Por ejemplo, se permitirá que el servidor de aplicación se conecte a la instancia de Cloud SQL, pero se bloqueará el acceso directo a la base de datos desde internet [información no proveniente de las fuentes]. Esto crea una capa adicional de seguridad y aislamiento para la infraestructura del backend.

Consideraciones Adicionales del Backend

Además de la configuración de servicios fundamental, existen consideraciones críticas para el backend que garantizan la robustez, seguridad y confiabilidad de la plataforma de Remex

8.
• Validación de Datos en el Backend:
    ◦ Es imperativo implementar una rigurosa validación de datos en el backend para todas las entradas recibidas a través de las APIs [información no proveniente de las fuentes, sino una práctica recomendada en arquitectura de software]. Esta validación asegura la integridad de la información, previene el ingreso de datos corruptos o maliciosos, y mantiene la consistencia de la base de datos [información no proveniente de las fuentes]. Esto incluye la verificación de tipos de datos, rangos, formatos, obligatoriedad de campos y la aplicación de reglas de negocio específicas antes de procesar o almacenar cualquier dato. Una validación robusta es la primera línea de defensa contra vulnerabilidades de seguridad y errores de aplicación [información no proveniente de las fuentes].
• Registro de Actividades Clave (Logs de Auditoría):
    ◦ Reafirmando la importancia de lo mencionado previamente en la sección de "Configuración de Servicios de GCP para el Backend" sobre Cloud Logging [información no proveniente de las fuentes], es esencial que el backend genere y recopile logs de auditoría para las actividades clave [información no proveniente de las fuentes]. Estos logs deben registrar acciones significativas como inicios de sesión (exitosos y fallidos), modificaciones de datos de usuarios o productos, transacciones (ventas de productos, registros de nuevos socios), y cualquier intento de acceso no autorizado [información no proveniente de las fuentes]. La configuración de Cloud Logging para recopilar estos logs del servidor de aplicación (Cloud Run) y otros servicios de GCP es crucial [información no proveniente de las fuentes]. Los logs de auditoría son fundamentales para:
        ▪ Seguridad: Permitir la detección temprana de patrones de comportamiento sospechosos o ataques, facilitando la investigación forense en caso de una brecha de seguridad [información no proveniente de las fuentes].
        ▪ Seguimiento y Cumplimiento: Proporcionar un historial inmutable de las operaciones del sistema, lo que es vital para la resolución de problemas, la rendición de cuentas y el cumplimiento de posibles regulaciones futuras [información no proveniente de las fuentes].

Documento de Diseño de la Base de Datos (Cloud SQL)

Este documento detalla el diseño de la base de datos relacional para el "Ecosistema Digital Inteligente para Cirugía Especial", la cual se alojará en Google Cloud Platform (GCP) utilizando Cloud SQL (MySQL/PostgreSQL). El objetivo es centralizar toda la información crítica del consultorio de manera segura, garantizando la integridad de los datos sensibles mediante cifrado en reposo.

1. Modelo Entidad-Relación (ERD)

El modelo Entidad-Relación (ERD) describe las principales entidades y sus interconexiones dentro del ecosistema digital, sirviendo como la columna vertebral para la gestión de pacientes y la operación diaria del consultorio. Este modelo asegura que la información estructurada, como perfiles de pacientes, historiales clínicos, citas y usuarios, se almacene de forma coherente y segura.

Las entidades principales y sus relaciones son las siguientes:

• PACIENTES y HISTORIAL_CLINICO: La entidad PACIENTES "tiene" una relación de uno a muchos con HISTORIAL_CLINICO, lo que significa que un paciente puede tener múltiples entradas en su historial clínico a lo largo del tiempo.
• HISTORIAL_CLINICO y DOCUMENTOS: La entidad HISTORIAL_CLINICO "contiene" DOCUMENTOS. Esto permite adjuntar archivos digitales como PDFs, imágenes o videos cortos a entradas específicas del historial clínico de un paciente.
• PACIENTES y CITAS: La entidad PACIENTES "agenda" CITAS. Un paciente puede tener múltiples citas programadas con el consultorio.
• CITAS y PROCEDIMIENTOS_CITAS: La entidad CITAS "se relaciona con" PROCEDIMIENTOS_CITAS. Esta tabla intermedia permite asociar una o más citas con procedimientos específicos.
• PROCEDIMIENTOS_CITAS y PROCEDIMIENTOS: La entidad PROCEDIMIENTOS_CITAS "detalla" PROCEDIMIENTOS. Esto vincula los procedimientos realizados con las citas correspondientes.
• USUARIOS y ROLES: La entidad ROLES "asigna" USUARIOS. Cada usuario del sistema tiene un rol definido que determina sus permisos de acceso a la información y funcionalidades.
• USUARIOS y LOGS_AUDITORIA: La entidad USUARIOS "realiza" operaciones que son registradas en LOGS_AUDITORIA. Este componente registra cronológicamente todas las actividades y cambios realizados en el sistema, siendo vital para la seguridad y el seguimiento de operaciones.

2. Especificación Completa de Tablas (Parte 1)

A continuación, se detalla la estructura y las características de las tablas PACIENTES e HISTORIAL_CLINICO, incluyendo sus columnas, tipos de datos, restricciones y relaciones.

Tabla: PACIENTES
Esta tabla centraliza la información demográfica y de contacto de todos los pacientes del consultorio.
Columna
Tipo de Dato
Restricciones
Descripción
id_paciente
INT
PK, NOT NULL, AUTO_INCREMENT
Identificador único y autoincremental para cada paciente.
nombre
VARCHAR(255)
NOT NULL
Nombre(s) del paciente.
apellido
VARCHAR(255)
NOT NULL
Apellido(s) del paciente.
fecha_nac
DATE
NULLABLE
Fecha de nacimiento del paciente.
datos_contacto
JSON
NULLABLE
Almacena información de contacto adicional (ej., teléfono, email, dirección) en formato JSON.
activo
BOOLEAN
NOT NULL, DEFAULT TRUE
Indica si el registro del paciente está activo (TRUE) o ha sido eliminado suavemente/archivado (FALSE).
Tabla: HISTORIAL_CLINICO
Esta tabla almacena el registro cronológico y detallado de cada interacción clínica con un paciente.
Columna
Tipo de Dato
Restricciones
Descripción
id_historial
INT
PK, NOT NULL, AUTO_INCREMENT
Identificador único y autoincremental para cada entrada del historial clínico.
id_paciente
INT
FK a PACIENTES.id_paciente, NOT NULL
Clave foránea que vincula la entrada del historial a un paciente específico.
fecha_consulta
DATE
NOT NULL
Fecha en la que se realizó la consulta o se registró la entrada.
diagnostico
TEXT
NULLABLE
Diagnóstico asociado a la consulta.
notas_medico
TEXT
NULLABLE
Notas detalladas del médico sobre la evolución, procedimientos o tratamientos.
activo
BOOLEAN
NOT NULL, DEFAULT TRUE

Indica si la entrada del historial está activa (TRUE) o ha sido marcada como inactiva/archivada (FALSE).

2. Especificación Completa de Tablas (Parte 2)
Se presenta la estructura detallada de las tablas USUARIOS, ROLES y LOGS_AUDITORIA, cruciales para la seguridad del sistema y el control de acceso basado en roles.
Tabla: USUARIOS
Esta tabla almacena la información de los usuarios que acceden al sistema, incluyendo sus credenciales seguras y el rol asignado, lo cual es fundamental para el control de acceso.
Columna
Tipo de Dato
Restricciones
Descripción
id_usuario
INT
PK
Identificador único para cada usuario del sistema.
nombre
VARCHAR
Nombre de usuario, utilizado para el inicio de sesión.
hash_password
VARCHAR
Contraseña del usuario almacenada de forma segura mediante una función de hash.
id_role
INT
FK a ROLES.id_role
Clave foránea que vincula al usuario con un rol específico, definiendo sus permisos.
activo
BOOLEAN
Indica si la cuenta del usuario está activa (TRUE) o bloqueada/inactiva (FALSE).
Tabla: ROLES
Esta tabla define los diferentes tipos de roles dentro del sistema y los permisos asociados a cada uno, implementando el control de acceso basado en roles.
Columna
Tipo de Dato
Restricciones
Descripción
id_role
INT
PK
Identificador único para cada rol.
nombre_rol
VARCHAR
Nombre descriptivo del rol (ej., "Admin Principal", "Asistente", "Invitado (Solo Lectura)").
permisos
JSON
Estructura JSON que detalla las capacidades y permisos asociados a este rol.
Tabla: LOGS_AUDITORIA
Esta tabla es esencial para la seguridad y el monitoreo, registrando cronológicamente todas las actividades y cambios realizados en el sistema. Los logs son inmutables (Write Once, Read Many - WORM).
Columna
Tipo de Dato
Restricciones
Descripción
id_log
INT
PK
Identificador único y autoincremental para cada entrada del log de auditoría.
tabla_afectada
VARCHAR
Nombre de la tabla de la base de datos que fue afectada por la operación.

id_registro_afectado
INT
ID del registro específico dentro de la tabla_afectada que fue modificado o accedido.
tipo_operacion
VARCHAR
Tipo de operación realizada (ej., 'login_exitoso', 'login_fallido', 'crear', 'actualizar', 'eliminar').
datos_anteriores
JSON
Almacena el estado de los datos del registro afectado antes de la operación (útil para actualizaciones).
datos_nuevos
JSON
Almacena el estado de los datos del registro afectado después de la operación.
fecha_hora
DATETIME
Marca de tiempo de cuándo ocurrió la operación.
id_usuario_autor
INT
FK a USUARIOS.id_usuario
Clave foránea que identifica al usuario que realizó la operación registrada.
Nota sobre restricciones adicionales: La información proporcionada para estas tablas en las fuentes no especifica explícitamente si las columnas deben ser NOT NULL, si son AUTO_INCREMENT (además de la clave primaria), o si tienen valores por defecto. Si esta granularidad es requerida, sería necesario definirla en una etapa posterior del diseño de la base de datos.

3. Estrategia de Indexación
La indexación es un componente crítico para optimizar el rendimiento de la base de datos, especialmente en operaciones de búsqueda, filtrado y ordenamiento de datos. Los índices permiten a la base de datos localizar rápidamente las filas de una tabla sin tener que escanear todas las filas, lo que reduce significativamente los tiempos de respuesta de las consultas y mejora la eficiencia operativa general del sistema.
Basado en los requisitos funcionales del "Ecosistema Digital Inteligente para Cirugía Especial", las siguientes columnas son candidatas clave para la creación de índices, dada su frecuente participación en operaciones de consulta:
• Tabla USUARIOS:
    ◦ nombre: Fundamental para la autenticación de usuarios y búsquedas de personal, ya que el usuario ingresa su "Nombre de Usuario" para iniciar sesión.
    ◦ id_role: Como clave foránea, esta columna será utilizada frecuentemente en uniones (JOINs) con la tabla ROLES para determinar los permisos de acceso del usuario, lo cual es vital para el control de acceso basado en roles.
    ◦ activo: Para filtrar rápidamente los usuarios activos e inactivos del sistema.
• Tabla ROLES:
    ◦ nombre_rol: Si se realizan búsquedas o se necesita filtrar roles por su nombre descriptivo (ej., "Admin Principal", "Asistente").
• Tabla LOGS_AUDITORIA:
    ◦ fecha_hora: Esencial para el monitoreo de seguridad y el seguimiento cronológico de eventos, permitiendo consultas rápidas basadas en rangos de tiempo.
    ◦ tabla_afectada y id_registro_afectado: Combinadas, estas columnas son cruciales para auditar cambios específicos en registros de tablas determinadas. Un índice compuesto o índices individuales en estas columnas facilitarían la recuperación de logs relacionados con una entidad específica.
    ◦ tipo_operacion: Para filtrar registros de auditoría por tipo de evento (ej., 'login_exitoso', 'crear', 'actualizar', 'eliminar').
    ◦ id_usuario_autor: Como clave foránea, se usará para identificar rápidamente todas las acciones realizadas por un usuario específico.
• Candidatos de Indexación en Tablas de Datos del Paciente (Inferido de Requisitos Funcionales): Aunque la especificación detallada de estas tablas no fue solicitada en esta sección, los requisitos funcionales del Dashboard implican la necesidad de indexación en ellas:
    ◦ Tablas de Pacientes: Las columnas utilizadas para la "Búsqueda Avanzada" de pacientes por "nombre, ID, palabra clave en notas" son candidatas principales para

índices, incluyendo nombre, apellido, y posiblemente campos relevantes en HISTORIAL_CLINICO.
    ◦ Tablas de Citas: fecha_hora es crucial para la visualización de "Citas del Día" y la gestión de la agenda.
    ◦ Tablas de Documentos: id_registro_afectado (o el FK al paciente/historial) y nombre_archivo para la "Gestión Documental Avanzada".
La implementación de estos índices se realizará para optimizar las consultas más frecuentes y críticas del sistema, garantizando la velocidad y la reactividad necesarias para una experiencia de usuario fluida.
4. Plan de Migración de Datos Existentes
La migración de datos existentes es una etapa crucial en la implementación del "Ecosistema Digital Inteligente para Cirugía Especial", particularmente la carga de los archivos digitales iniciales que el consultorio del Dr. Sánchez García ya posee.
El proceso general de migración de datos se realizará a un alto nivel de la siguiente manera:
1. Identificación y Recopilación de Archivos Digitales: Se identificarán y recopilarán todos los archivos digitales relevantes existentes en el consultorio (PDFs, imágenes como RX, videos cortos, etc.) que deben ser integrados en el nuevo sistema.
2. Extracción de Metadatos: Para cada archivo, se extraerá la información relevante (metadatos) que permitirá vincularlo correctamente en la base de datos relacional. Esto incluirá, pero no se limitará a, el nombre del archivo, tipo de archivo, fecha de subida (o creación original si es posible determinarla), y lo más importante, el identificador del paciente o del registro del historial clínico al que pertenece.
3. Carga de Archivos a Cloud Storage: Los archivos digitales en sí se subirán de forma segura a Google Cloud Storage (equivalente a S3/Blob). Este servicio asegura el almacenamiento centralizado y seguro, con los datos cifrados en reposo.
4. Ingesta de Metadatos en Cloud SQL: Los metadatos extraídos de los archivos, junto con cualquier otra información estructurada existente (ej., datos demográficos de pacientes que ya tienen historial digital), se insertarán en la Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL). En la tabla DOCUMENTOS, se almacenará la url_almacenamiento que apunta a la ubicación del archivo en Cloud Storage, asegurando el vínculo entre el registro de la base de datos y el archivo físico.
5. Verificación y Validación: Una vez completada la migración, se realizarán procesos de verificación y validación para asegurar la integridad de los datos, la correcta asociación de archivos con los pacientes/historiales, y que todos los archivos sean accesibles desde el sistema.
La migración se enfocará en asegurar la integridad y seguridad de la información desde el primer día.
5. Especificaciones de Seguridad de la Base de Datos
La seguridad de la base de datos es un pilar fundamental del "Ecosistema Digital Inteligente para Cirugía Especial", dada la sensibilidad de la información de los pacientes. Las siguientes especificaciones de seguridad son primordiales para la protección de los datos:
1. Configuración de Usuarios y Permisos de Acceso (Control de Acceso Basado en Roles - RBAC):
    ◦ La base de datos, específicamente la tabla USUARIOS, integra un id_role que se vincula a la tabla ROLES, la cual define los permisos de cada tipo de usuario mediante una estructura JSON. Esto implementa un estricto "Control de Acceso Basado en Roles".
    ◦ Los roles definidos incluyen "Admin Principal" (control total, gestión de usuarios y logs), "Admin Secundario" (todo excepto gestión de usuarios y logs), "Asistente" (gestión de Pacientes, Citas, Documentos, Ver Reportes) e "Invitado (Solo Lectura)" (ver Pacientes, Citas, Reportes).
    ◦ A nivel de la base de datos (Cloud SQL), se configurarán usuarios con el "principio de privilegio mínimo", de modo que el backend (Servidor de Aplicación) solo tenga los permisos necesarios para realizar las operaciones de lectura, escritura y modificación de datos que requiera la lógica de negocio, sin otorgar acceso directo innecesario a los usuarios finales.
2. Cifrado en Reposo para Datos Sensibles:
    ◦ Todos los datos sensibles almacenados en la "Base de Datos Relacional (Cloud SQL)" y en el "Almacenamiento de Archivos (Cloud Storage)" estarán cifrados en reposo.
    ◦ Para las contraseñas, se utilizarán funciones de hash seguras (indicado por el campo hash_password en la tabla USUARIOS) en lugar de almacenar las contraseñas en texto plano. Esto protege contra la exposición de credenciales en caso de una brecha de datos.

3. Plan de Copias de Seguridad y Restauración:
    ◦ Aunque los detalles específicos no se proporcionan, un componente esencial de la seguridad y la continuidad operativa de una base de datos en la nube como Cloud SQL es la implementación de un robusto plan de copias de seguridad (backups) automáticas y la capacidad de restauración (recovery).
    ◦ Esto asegurará la disponibilidad de los datos en caso de fallos, corrupción de datos o desastres, permitiendo la recuperación rápida a un estado anterior y minimizando el tiempo de inactividad del sistema.
    ◦ Las copias de seguridad incluirán tanto la base de datos estructurada como los archivos almacenados en Cloud Storage.
Nota: La información proporcionada en las fuentes no especifica detalles técnicos granulares sobre las configuraciones exactas de indexación (ej., tipos de índices como B-tree, hash), el proceso de migración a nivel de script o herramientas, ni las configuraciones específicas de usuarios y roles dentro de Cloud SQL o la frecuencia y retención del plan de copias de seguridad. Estos detalles se definirán durante la fase de implementación y configuración de la base de datos para alinearse con las mejores prácticas de GCP y los requisitos de rendimiento específicos.

Continuando con el "Documento de Diseño de la Base de Datos (Cloud SQL)", a continuación se presentan las secciones adicionales solicitadas:
8. Estrategia de Indexación
La estrategia de indexación es fundamental para optimizar el rendimiento de la base de datos, especialmente en operaciones de búsqueda y filtrado de datos. Los índices permiten que el motor de la base de datos localice rápidamente filas sin tener que escanear una tabla entera, mejorando drásticamente la velocidad de las consultas. Dada la necesidad de "búsqueda instantánea de pacientes" y el acceso eficiente a historiales clínicos, una correcta indexación es crucial para la "eficiencia operativa" del ecosistema.
Candidatos para la Creación de Índices:
Basándose en los requisitos funcionales del "Ecosistema Digital Inteligente para Cirugía Especial", las siguientes columnas son candidatas primarias para la creación de índices, incluyendo índices únicos para claves primarias y foráneas, y índices no únicos para columnas frecuentemente utilizadas en búsquedas o filtros:
• PACIENTES Tabla:
    ◦ id_paciente: Clave primaria, se indexará automáticamente. Es crucial para la búsqueda de pacientes por ID y la vinculación a otras tablas como HISTORIAL_CLINICO y CITAS.
    ◦ nombre, apellido: Estas columnas serán utilizadas frecuentemente para la "búsqueda instantánea de pacientes" por nombre o apellido. Un índice combinado o índices separados en estas columnas acelerarán las búsquedas.
    ◦ activo: Utilizada para filtrar pacientes activos en el sistema.
• HISTORIAL_CLINICO Tabla:
    ◦ id_historial: Clave primaria, se indexará automáticamente.
    ◦ id_paciente: Clave foránea, esencial para buscar el historial clínico de un paciente específico. La indexación de esta FK optimizará las uniones (JOINs) y las búsquedas por paciente.
    ◦ fecha_consulta: Es probable que se realicen búsquedas y filtros por fecha de consulta para el "Registro Historial Clínico Detallado".
• CITAS Tabla:
    ◦ id_cita: Clave primaria, se indexará automáticamente.
    ◦ id_paciente: Clave foránea, utilizada para vincular citas a pacientes y para filtrar citas por paciente.
    ◦ fecha_hora: Fundamental para el "Módulo de Agenda y Citas", permitiendo búsquedas rápidas por fecha y hora para visualizar las "Citas del Día".
    ◦ estado: Utilizada para filtrar citas por su estado (ej., "Confirmada", "Cancelada", "Atendida").
• USUARIOS Tabla:
    ◦ id_usuario: Clave primaria, se indexará automáticamente.
    ◦ nombre: Utilizada para el login y la identificación de usuarios en los logs de auditoría.
    ◦ id_role: Clave foránea, esencial para determinar los "permisos del usuario" y para la funcionalidad de "Seguridad, Roles y Accesos Rápidos".
• LOGS_AUDITORIA Tabla:
    ◦ id_log: Clave primaria, se indexará automáticamente.

    ◦ fecha_hora: Crucial para la recuperación cronológica de eventos en el "Servicio de Logs de Auditoría".
    ◦ id_usuario_autor: Permite filtrar los logs por el usuario que realizó la acción.
    ◦ tabla_afectada y id_registro_afectado: Permiten rastrear acciones sobre registros específicos. Un índice compuesto o índices individuales pueden ser beneficiosos aquí.

La implementación de estos índices se realizará durante la fase de modelado y diseño de la base de datos para asegurar el rendimiento óptimo del sistema desde el inicio.

9. Plan de Migración de Datos Existentes

La migración de datos existentes es una parte integral de la Fase 1 del proyecto, asegurando la "carga inicial de los documentos que usted y su equipo ya poseen en formato digital" para integrarlos al nuevo sistema. Este proceso busca "eliminar la dispersión de archivos" y centralizar la información.

El plan de migración de archivos digitales se abordará a alto nivel de la siguiente manera:

1. Identificación y Recopilación de Archivos: Se colaborará con el consultorio para identificar y recopilar todos los archivos digitales existentes (PDFs, imágenes de estudios como RX, videos cortos de procedimientos, etc.) que necesiten ser integrados al sistema. Es importante destacar que el proyecto NO incluye la digitalización (escaneo) de archivos físicos existentes; solo se migrarán los archivos que ya estén en formato digital.
2. Extracción y Estructuración de Metadatos:
    ◦ Para cada archivo digital, se extraerán o se solicitarán los metadatos relevantes (ej., nombre del paciente asociado, fecha del documento, tipo de archivo, descripción).
    ◦ Estos metadatos se estructurarán de acuerdo con el esquema de la "Base de Datos Relacional (Cloud SQL)". Por ejemplo, para los documentos de pacientes, se vincularán al id_historial o id_paciente en las tablas DOCUMENTOS o HISTORIAL_CLINICO.
3. Carga Segura a Cloud Storage:
    ◦ Los archivos digitales grandes y pesados se subirán al servicio de "Almacenamiento de Archivos (Cloud Storage - S3/Blob)" de Google Cloud Platform (GCP).
    ◦ Es crucial que, al igual que la base de datos, estos archivos estén cifrados en reposo en Cloud Storage, garantizando la seguridad de la información sensible del paciente.
4. Vinculación en la Base de Datos (Cloud SQL):
    ◦ Una vez que los archivos estén cargados en Cloud Storage, la "Base de Datos Relacional (Cloud SQL)" no almacenará los archivos directamente. En su lugar, la tabla DOCUMENTOS almacenará la url_almacenamiento (URL de acceso) que apunta a la ubicación segura del archivo en Cloud Storage, junto con otros metadatos como nombre_archivo y tipo_archivo.
    ◦ Esto permite que la aplicación de escritorio, a través del backend, acceda y visualice los documentos asociados al expediente del paciente de forma eficiente.
5. Verificación y Control de Calidad: Se realizarán verificaciones post-migración para asegurar que todos los archivos se hayan cargado correctamente, que los metadatos sean precisos y que las vinculaciones entre la base de datos y Cloud Storage funcionen según lo esperado.

La migración de datos se realizará en la Fase 1 del proyecto, aprovechando herramientas de Google Cloud Storage para este propósito.

10. Especificaciones de Seguridad de la Base de Datos

La seguridad de la base de datos es una "preocupación de seguridad de datos" fundamental que el ecosistema busca abordar. Se integra en múltiples capas de la arquitectura en la nube.

1. Configuración de Usuarios y Permisos de Acceso a la Base de Datos:
    ◦ El sistema implementa un estricto "Control de Acceso Basado en Roles (RBAC)". Al iniciar sesión, cada usuario es asociado a un rol específico ("Administrador", "Admin Principal", "Admin Secundario", "Asistente", "Invitado") que define sus permisos específicos sobre la información y funcionalidades.
    ◦ El backend verifica las credenciales contra la "Base de Datos Relacional (Cloud SQL)" y consulta el Sistema de Roles (ROLES en el modelo de datos) para determinar los permisos del usuario.
    ◦ Para la conectividad de la aplicación al backend y del backend a la base de datos, se aplicará el "principio de privilegio mínimo". Esto significa que el usuario de la base de datos que utiliza el backend para conectarse solo tendrá los permisos necesarios para realizar las operaciones requeridas (lectura, escritura, modificación en tablas específicas), sin privilegios administrativos excesivos.

    ◦ La gestión de usuarios y la asignación de roles se realiza a través del Dashboard, en la sección de "Administración", accesible solo para roles con permisos adecuados como el "Admin Principal".
2. Requisitos de Cifrado en Reposo para los Datos Sensibles:
    ◦ Una medida de seguridad crítica para proteger la información sensible del paciente es el cifrado de datos en reposo.
    ◦ Todos los datos sensibles estarán cifrados en reposo tanto en la "Base de Datos Relacional (Cloud SQL)" como en el "Almacenamiento de Archivos (Cloud Storage)". Esto asegura que, en caso de un acceso no autorizado al almacenamiento subyacente, los datos permanezcan ilegibles.
    ◦ Para las contraseñas de usuario, la base de datos incluye un campo hash_password VARCHAR en la tabla USUARIOS, lo que sugiere el uso de funciones de hash seguras en lugar de almacenar las contraseñas en texto plano. Esto es una práctica estándar esencial contra fugas de datos.
3. Plan de Copias de Seguridad y Restauración:
    ◦ Aunque no se detalla un plan específico de copias de seguridad en todos los documentos, la "Guía de Replicación Técnica" y "Arquitectura Nube" mencionan explícitamente la necesidad de "Implementar copias de seguridad automáticas y recuperación ante desastres" como una recomendación para la base de datos Cloud SQL.
    ◦ Esto es una buena práctica estándar para cualquier base de datos en producción, y la plataforma Google Cloud SQL ofrece capacidades de backup automático y point-in-time recovery, lo que garantiza la "alta disponibilidad" y minimiza el tiempo de inactividad. Un plan de restauración claro será crucial para mitigar el riesgo de pérdida de datos en caso de fallos del sistema o corrupción de datos.

6. Elección y Configuración Inicial de Cloud SQL
La base de datos relacional para el "Ecosistema Digital Inteligente para Cirugía Especial" se hospedará en Google Cloud SQL. Cloud SQL es un servicio de base de datos completamente gestionado por Google Cloud Platform (GCP) que facilita la configuración, el mantenimiento, la gestión y el escalado de bases de datos relacionales en la nube.
Elección del Motor de Base de Datos (MySQL o PostgreSQL): Las fuentes del proyecto indican que se pueden utilizar sistemas de gestión de bases de datos como MySQL o PostgreSQL. La elección final entre MySQL y PostgreSQL se determinará en una fase posterior, basándose en consideraciones más detalladas, como:
• Requisitos Específicos de Funcionalidad: Si alguna de las bases de datos ofrece características o extensiones únicas que sean particularmente ventajosas para las funcionalidades avanzadas del ecosistema.
• Experiencia del Equipo de Desarrollo: La familiaridad y experiencia del equipo de desarrollo con un motor de base de datos en particular pueden influir en la elección para optimizar la eficiencia y reducir la curva de aprendizaje.
• Patrones de Uso y Rendimiento Esperado: Aunque ambos son robustos, pueden tener diferencias sutiles en el rendimiento para ciertos tipos de cargas de trabajo (ej., operaciones de lectura intensivas vs. escritura intensivas, complejidad de consultas).
• Consideraciones de Licenciamiento y Comunidad: MySQL es de código abierto con diversas opciones de licenciamiento, mientras que PostgreSQL es completamente de código abierto y conocido por su fuerte adherencia a los estándares SQL y su comunidad activa.
Independientemente del motor elegido, ambos son compatibles con Cloud SQL y ofrecen características de alta disponibilidad y recuperación de desastres.
Configuración Inicial de la Instancia en GCP (Alto Nivel): El proceso de configuración inicial de la instancia de Cloud SQL en GCP implica los siguientes pasos clave:
1. Creación de la Instancia:
    ◦ Acceder a la consola de Google Cloud Platform y navegar a la sección de Cloud SQL.
    ◦ Seleccionar la opción para crear una nueva instancia de base de datos.
    ◦ Elegir el motor de base de datos deseado (MySQL o PostgreSQL).
    ◦ Especificar la región donde se desplegará la instancia para asegurar baja latencia con el Servidor de Aplicación (Backend).
    ◦ Definir la configuración de la máquina (tipo de CPU, memoria) y el tipo de almacenamiento (ej., SSD) y su tamaño inicial.
2. Configuración de Red y Conectividad:
    ◦ Configurar la instancia para que opere dentro de la Virtual Private Cloud (VPC) del proyecto. Esto asegura que la base de datos esté en una red aislada y segura, fundamental para la seguridad de los datos.
    ◦ Establecer la conectividad de IP privada para el backend, lo que mejora la seguridad

y el rendimiento al mantener el tráfico de la base de datos dentro de la red privada de Google.
    ◦ Configurar las redes autorizadas y reglas de firewall para permitir solo el tráfico desde el Servidor de Aplicación (Backend) y otras IPs de administración necesarias.
3. Creación de Usuarios Iniciales:
    ◦ Crear usuarios de base de datos con contraseñas seguras y el "principio de privilegio mínimo". Es decir, el usuario que utilizará el backend para conectarse a la base de datos solo tendrá los permisos necesarios para las operaciones que realice (lectura, escritura, modificación de tablas específicas), sin privilegios administrativos excesivos.
    ◦ Para la gestión de contraseñas de usuarios en la aplicación, se utilizarán funciones de hash seguras para almacenar hash_password en la tabla USUARIOS.
7. Consideraciones Adicionales
Además de las secciones anteriores, existen otras consideraciones importantes para el diseño y la implementación de la base de datos:
1. Proyección y Gestión del Almacenamiento:
    ◦ El servicio de infraestructura en la nube de Google Cloud incluye una capacidad de hasta 1 Terabyte (1000 GB) mensual como cortesía durante los primeros seis meses.
    ◦ Si bien las fuentes no mencionan explícitamente una proyección de 2TB, cualquier aumento en la capacidad de almacenamiento más allá del 1 Terabyte incluido tendrá un costo adicional.
    ◦ Es fundamental aclarar que este almacenamiento (tanto el inicial de 1TB como cualquier aumento futuro) incluye tanto el espacio utilizado por la Base de Datos Relacional (Cloud SQL) para los datos estructurados como el espacio para los archivos digitales grandes almacenados en Cloud Storage. Se deberá monitorear de cerca el consumo de almacenamiento para gestionar los costos de manera eficiente.
2. Distinción Clara entre Datos Estructurados y Archivos Grandes:
    ◦ Se reafirma que la Base de Datos Relacional (Cloud SQL) se utilizará exclusivamente para almacenar información estructurada y crítica como perfiles de pacientes, historiales clínicos, citas, usuarios y roles.
    ◦ El Almacenamiento de Archivos (Cloud Storage - S3/Blob) se designará para el almacenamiento centralizado y seguro de archivos digitales pesados, como PDFs, imágenes (ej., RX) y videos cortos asociados a los expedientes de los pacientes. Esta separación optimiza el rendimiento y el costo de almacenamiento, y la base de datos Cloud SQL almacenará únicamente la url_almacenamiento que apunta a la ubicación del archivo en Cloud Storage.
3. Seguridad de los Datos Sensibles (Cifrado):
    ◦ Una consideración primordial es que todos los datos sensibles estarán cifrados en reposo, tanto en la Base de Datos Relacional (Cloud SQL) como en el Almacenamiento de Archivos (Cloud Storage). Esto es una medida de seguridad crítica para proteger la información del paciente contra accesos no autorizados.
    ◦ Las contraseñas de usuario se almacenarán utilizando funciones de hash seguras (campo hash_password) en lugar de texto plano.
4. Escalabilidad y Alta Disponibilidad:
    ◦ La elección de GCP y sus servicios (Cloud SQL, Cloud Storage) proporciona inherentemente escalabilidad (la capacidad de expandirse para manejar un mayor volumen de pacientes y datos sin interrupciones) y alta disponibilidad (minimizando el tiempo de inactividad del sistema). Estas características son fundamentales para la continuidad operativa del consultorio.
5. Requisito de Conectividad:
    ◦ Para un rendimiento óptimo del ecosistema, se requiere una conexión a internet estable de al menos 250 Mbps. La dependencia de la infraestructura en la nube hace que la calidad de la conexión sea crucial.
6. Vinculación del Costo de la Nube al Plan de Mantenimiento:
    ◦ La cortesía de Google Cloud por los primeros seis meses está directamente vinculada a la vigencia del Plan de Mantenimiento y Soporte mensual. Si el cliente decide no continuar con el plan de mantenimiento durante este periodo, la cortesía del servicio en la nube finalizará en ese mismo momento. Esta es una condición importante que la alta dirección debe considerar.
Con estas secciones, el "Documento de Diseño de la Base de Datos (Cloud SQL)" está completo.
Documento de Requisitos Funcionales Detallados: Dashboard de Administración
1. Propósito General del Dashboard de Administración

El Dashboard de Administración es el Núcleo Administrativo Central del "Ecosistema Digital Inteligente para Cirugía Especial". Su propósito fundamental es transformar la gestión operativa del consultorio del Dr. Joel Sánchez García [88, 110-I]. Actúa como el centro de operaciones principal para el personal médico y administrativo, permitiendo la gestión centralizada de toda la información crítica de los pacientes [31-I, 45, 185-1].
Este Dashboard está diseñado para optimizar la gestión de pacientes y mejorar significativamente la eficiencia operativa al reducir los procesos manuales y el riesgo de errores, lo que a su vez libera tiempo valioso para el personal [31-I, 46, 186-1]. Aborda directamente dolores operativos clave como la "saturación y dispersión de información" y la "dificultad en el seguimiento eficiente del paciente" [46, 94, 118-VI, 186-1]. Además, centraliza la inteligencia del sistema para asegurar que la lógica de negocio se ejecute de manera segura y escalable .

2. Estructura de Componentes de la Pantalla

La pantalla del Dashboard de Administración presenta una interfaz clara y funcional, profundamente anclada en el estilo visual "Glassmorphism Oscuro". Este diseño no es solo estético, sino funcional, concebido para ofrecer comodidad visual, una jerarquía de información clara y una experiencia de usuario fluida desde el primer punto de interacción. Se logra mediante elementos que parecen flotar, una luz sutil que emana desde el interior (efecto "Aurora"), y una textura de información nítida.
La pantalla se organiza en tres áreas principales: la Barra de Navegación Lateral (izquierda), el Header Superior (barra superior) y el Área de Contenido Principal (que será detallada posteriormente).

2.1. Barra de Navegación Lateral (Izquierda)

La barra de navegación lateral, ubicada a la izquierda de la interfaz, es el panel principal para la navegación dentro del sistema [48, 53, 154-1.2, 189-4].

- 2.1.1. Identidad de la Aplicación:
    - Logo (Icono circular): En la parte superior de la barra de navegación lateral, se mostrará un icono circular que representa la marca "Cirugía Especial" [275-2.1]. Este logo sirve como un ancla visual constante, reforzando la identidad de la aplicación en todo momento. Su diseño debe ser limpio y moderno, siguiendo la estética general del sistema.
    - Texto "CIRUGIA ESPECIAL": Acompañando al logo, se presentará el texto "CIRUGIA ESPECIAL" [275-2.1]. Este texto, junto con el logo, consolida la identidad visual y textual de la aplicación, siendo siempre visible para el usuario.
- 2.1.2. Navegación Principal (Iconos y Secciones):
    - Las secciones principales del sistema se presentan mediante iconos claros y texto, utilizando un estilo de línea (line-style) con grosor de trazo consistente y terminaciones redondeadas, en el color text-primary (#F5F5F5) para una legibilidad óptima.
    - Inicio (Icono de casa): Al hacer clic, este icono redirige al usuario a la Pantalla: Dashboard Principal (Vista General) [154-1.2, 275-2.1, 189-4]. Funciona como el punto de regreso central para obtener una visión general del estado del consultorio y acceso rápido a las funcionalidades clave.
    - Registros (Icono de documento): Este icono proporciona acceso a la sección de Gestión Integral del Expediente del Paciente [90-1, 114, 155-2.A, 209-4, 252-4]. Permite la creación de nuevos perfiles de pacientes, la búsqueda avanzada de pacientes existentes, y el registro detallado y cronológico del historial clínico, centralizando toda la información crítica del paciente [33, 90-1, 114, 155-2.A, 209-4, 252-4].
    - Colaboradores/Equipo (Icono de usuarios): Este elemento de navegación dirige al usuario a la sección de Gestión de Usuarios dentro del módulo de Administración [37-4, 154-1.2, 158-2.D, 276-2.1]. Es el punto donde los usuarios con permisos adecuados (Administrador y Admin Principal) pueden gestionar cuentas de usuario, asignar y modificar roles, definiendo sus permisos de acceso a la información y funcionalidades [158-2.D, 181-7, 193-8].
    - Reportes (Icono de gráfico de barras): Proporciona acceso a la sección de analíticas y generación de reportes [154-1.2, 276-2.1]. Aquí, el usuario puede visualizar informes de visitas y procedimientos para obtener métricas clave sobre la operación del consultorio [49, 154-1.2, 188-3].
    - Integraciones/Módulos (Icono de cuadrícula): Este icono está destinado a ofrecer acceso a otros módulos o aplicaciones integradas dentro del ecosistema [276-2.1]. Aunque no se detallan módulos específicos adicionales más allá de los principales, este punto de acceso futuro permite la expansión y la integración de nuevas funcionalidades.
- 2.1.3. Sección "Mis Archivos":
    - Esta sección se enfoca en la gestión de documentos digitales asociados a los expedientes de los pacientes [276-2.1].

    ◦ Barra de Búsqueda: Un campo de texto con un icono de lupa incrustado que permite al usuario buscar archivos y carpetas por nombre o palabra clave dentro de esta sección de "Mis Archivos" [276-2.1]. La búsqueda debe ser ágil y ofrecer resultados relevantes a medida que el usuario escribe.
    ◦ Estructura de Carpetas: Se implementará un sistema de árbol de directorios para organizar la información de los pacientes de forma lógica y estructurada [277-2.1]. Esto facilita la navegación y el acceso a los documentos.
    ◦ Carpetas Visibles: Se listarán las siguientes carpetas predefinidas: "General", "Operaciones", "Consultas", "Laboratorio" [277-2.1]. Estas categorías permiten una clasificación inicial de los documentos para una mejor organización.
    ◦ Funcionalidad de Filtrado por Carpeta: Al hacer clic en cualquiera de las carpetas visibles, el contenido del Área de Contenido Principal (específicamente, la sección de la "Base de Datos" o "Registro de Pacientes") debe filtrarse automáticamente para mostrar solo los registros y documentos pertenecientes a esa categoría específica [277-2.1]. Esto proporciona una vista contextual y optimiza la búsqueda de información.
• 2.1.4. Sección "Automatizaciones":
    ◦ Esta sección contiene "placeholders para futuras funcionalidades o para que el usuario configure flujos de trabajo personalizados" [277-2.1]. Es un área designada para la expansión de las capacidades de automatización del sistema.
    ◦ Descripción: Se mostrará una descripción general de lo que se puede lograr con las automatizaciones, posiblemente con ejemplos como "Crear reporte mensual automáticamente" o "Notificar al paciente 24h antes de la consulta" [277-2.1].
    ◦ Primera Manifestación (Asistente Virtual): El Asistente Virtual (modelo de IA Gemma 3) es la primera implementación de esta capacidad de automatización [277-2.1]. Permite la redacción de correos, generación de hojas membretadas, recordatorios, y un chat interno para consultas rápidas, resúmenes de texto o redacción de informes [23, 38, 115, 159-2.E, 210-4, 254-4]. Este chat funciona únicamente con texto y no tiene la capacidad de generar archivos de ningún formato.
2.2. Header Superior (Barra Superior)
El Header Superior, ubicado en la parte superior de la interfaz, proporciona información general y accesos rápidos [53, 154-1.2, 189-4].
• 2.2.1. Accesos Rápidos a Servicios:
    ◦ Se implementará una barra central con iconos que representen accesos rápidos a servicios en la nube integrados [278-2.2, 154-1.2]. Es crucial interpretar estos iconos como accesos directos dentro de la aplicación, no como el dock del sistema operativo del usuario [278-2.2].
    ◦ Servicios Esperados: Esto incluye, pero no se limita a, accesos directos a carpetas específicas en Google Drive y a funcionalidades de Gmail, así como accesos directos para abrir aplicaciones de Office (como Docs/Sheets) [37-4, 154-1.2, 188-3, 210-4, 225-4, 254-4]. Estos widgets están siempre visibles en el dashboard, simplificando el acceso a herramientas esenciales y mejorando la eficiencia .
• 2.2.2. Controles de Usuario (Derecha):
    ◦ Ubicados en el extremo derecho del Header, estos controles proporcionan funcionalidades relacionadas con la interacción y el perfil del usuario [278-2.2].
    ◦ Búsqueda Global (Icono de lupa): Un icono de lupa que, al activarse, permite realizar una búsqueda que abarca toda la aplicación [278-2.2]. Esta búsqueda global debe permitir encontrar información relevante en todos los módulos, incluyendo pacientes, archivos, notas, citas, etc.
    ◦ Notificaciones (Icono de campana): Se incluirá un icono de campana que, al hacer clic, desplegará un centro para alertas y notificaciones del sistema [279-2.2]. Esto es crucial para mantener al personal informado sobre eventos importantes, como citas próximas, nuevos documentos subidos o cambios en el estado de los pacientes.
    ◦ Fecha y Hora: Se mostrará de forma clara la fecha y hora actuales [279-2.2]. Este elemento proporciona un contexto temporal constante para el usuario.
    ◦ Perfil de Usuario (Avatar): Se mostrará el avatar o imagen de perfil del usuario logueado [53, 154-1.2, 189-4, 279-2.2]. Al hacer clic en el avatar, se desplegará un menú contextual con las siguientes opciones:
        ▪ "Mi Perfil": Para acceder y gestionar la información personal del usuario.
        ▪ "Configuración": Para ajustar las preferencias de la aplicación específicas del usuario.
        ▪ "Cerrar Sesión": Para que el usuario pueda finalizar su sesión de forma segura [51, 154-1.2, 188-3].

2.3. Área de Contenido Principal

El Área de Contenido Principal es el espacio de trabajo dinámico del Dashboard, compuesto por diversos módulos o "widgets" que proporcionan información clave y acceso a las funcionalidades operativas del consultorio. Su diseño sigue la estética "Glassmorphism Oscuro", presentando paneles flotantes de "vidrio esmerilado" que organizan y jerarquizan la información, con un fondo dinámico de "Auras" de colores azul y violeta que realza el efecto visual y reduce la fatiga visual.

A. Módulo "Base de Datos"

Este módulo es la funcionalidad central para la gestión de registros de pacientes.

- Título: "Base de Datos".
- Subtítulo: "Registro de pacientes Cirugía Ecensial 2024".
- Barra de Acciones: Ubicada sobre la tabla de datos, proporciona herramientas para la gestión de registros en lote o individual.
    - Eliminar (Icono de papelera): Este botón se activa cuando se seleccionan una o más filas en la tabla. Su funcionalidad principal es realizar un borrado suave (soft delete) de los registros. Esto significa que el registro no se elimina permanentemente de la base de datos, sino que se marca internamente como inactivo o archivado (ej., is_deleted = true), lo que permite su recuperación futura y mantiene la integridad histórica de los datos.
    - Filtros (Icono de filtro): Al hacer clic en este icono, se despliega un panel o un modal que ofrece opciones avanzadas para filtrar los datos mostrados en la tabla. Los filtros pueden incluir criterios como "por fecha", "por estado", "por tipo de consulta", entre otros, permitiendo al usuario refinar rápidamente la visualización de los registros.
    - Exportar: Esta función permite al usuario exportar los datos seleccionados en la tabla o todos los datos filtrados a formatos comunes de hojas de cálculo, como CSV o Excel, facilitando el análisis externo o la generación de reportes.
    - + Nuevo Registro (Botón principal): Al interactuar con este botón, se abre un formulario o un modal dedicado a la creación de un nuevo registro de paciente, donde el usuario puede ingresar toda la información demográfica y de contacto inicial.
- Tabla de Datos: Presenta la información de los pacientes de manera estructurada y de fácil acceso.
    - Checkbox de Selección: Ubicado en la primera columna de cada fila, permite la selección múltiple de registros para aplicar acciones en lote (como Eliminar o Exportar). El encabezado de la columna incluirá un checkbox maestro para seleccionar o deseleccionar todos los registros visibles.
    - ID/Nombre del Paciente (Texto en negrita y enlace): Esta es la columna principal de identificación. El texto en negrita (font-weight: 700) le otorga jerarquía visual, y cada entrada es un enlace directo que lleva a la vista de detalle completa del paciente, donde se puede acceder a su expediente y historial clínico.
    - Columnas de Datos Adicionales (Texto regular): Muestran información relevante y configurable del paciente, como "Fecha de Última Consulta", "Diagnóstico Principal", "Médico Tratante", etc., con texto regular (font-weight: 400) para diferenciarse del nombre principal.
    - Estado (Píldora de color): Una representación visual mediante una "píldora" de color que indica el estado actual del paciente o registro. Los estados definidos son:
        - Active (Verde): Indica que el paciente está en tratamiento o seguimiento activo.
        - Inactive (Rojo/Gris): Señala que el paciente ha sido dado de alta, o que el registro ha sido archivado o está inactivo por algún otro motivo.
    - Menú de Acciones (Icono de tres puntos): Al hacer clic en este icono, se despliega un menú contextual con acciones específicas para el registro de la fila correspondiente. Las opciones incluyen:
        - "Ver Detalles".
        - "Editar Registro".
        - "Ver Historial Clínico Completo".
        - "Archivar".
        - "Eliminar (Suave)".

B. Módulo "Asistente virtual"

Este módulo representa la integración de la inteligencia artificial en el flujo de trabajo del consultorio, potenciada por el modelo Gemma 3 (o similar a Gemini).

- Título: "Asistente virtual".
- Funcionalidad: Permite a los usuarios interactuar con el sistema utilizando lenguaje natural para diversas tareas, optimizando la eficiencia operativa.
    - Interacción Lenguaje Natural: El asistente está diseñado para comprender y procesar

consultas formuladas en lenguaje coloquial, permitiendo una interacción intuitiva y conversacional.
    ◦ Búsqueda Inteligente: Los usuarios pueden realizar consultas de búsqueda complejas, como "Hola, estoy buscando un archivo me urge", "Recuerdas el nombre del paciente?", o "Si!! es Laura Vargas", y el asistente utilizará la IA para interpretar la solicitud y localizar la información o los archivos pertinentes en el sistema.
    ◦ Recuperación y Acciones Contextuales: Una vez que el asistente localiza el recurso solicitado (por ejemplo, "Lo encontré"), ofrecerá al usuario acciones contextuales relacionadas, como "te lo envío a tu correo o prefieres descargarlo?", facilitando la gestión de la información.
    ◦ Automatización de Tareas: El asistente virtual está conectado a las APIs del sistema, lo que le permite ejecutar acciones automatizadas en respuesta a las solicitudes del usuario. Esto incluye el envío de correos electrónicos, la generación de borradores de documentos (como hojas membretadas), y la creación de proyectos o recordatorios, liberando tiempo del personal.

C. Módulo "Notas"

Un widget simple diseñado para la gestión de anotaciones rápidas.

• Título: "Notas".

• Funcionalidad: Permite al usuario tomar notas personales y registrar recordatorios rápidos directamente en el dashboard. Por lo general, esta información es privada para el usuario que la creó. Se menciona la posibilidad de una función adicional para "adjuntar nota a paciente", lo que vincularía estas anotaciones directamente al expediente clínico para una referencia más contextual.

D. Módulo "Visor de Archivos Visuales"

Este módulo está dedicado a la visualización de contenido multimedia asociado a los pacientes.

• Funcionalidad: Permite a los usuarios visualizar imágenes y videos directamente relacionados con los expedientes de los pacientes. El contenido del visor se actualiza de forma contextual, mostrando los archivos visuales del paciente que está seleccionado en la tabla principal de la "Base de Datos" o que ha sido resultado de una búsqueda específica. Estos archivos se almacenan de forma segura en Cloud Storage y están cifrados en reposo.

• Componentes:
    ◦ Carrusel de Vistas Previas: Muestra miniaturas de los archivos visuales disponibles, organizados en un formato de carrusel. Las flechas a izquierda y derecha permiten al usuario navegar fácilmente por la galería de archivos.
    ◦ Filtros de Origen/Tipo: Permiten al usuario refinar la visualización del contenido en el carrusel.
        ▪ Cloud Drives: Filtra para mostrar únicamente archivos visuales que provienen de almacenamientos en la nube conectados, como Google Drive o OneDrive.
        ▪ Photos: Muestra exclusivamente imágenes, incluyendo formatos comunes como JPG, PNG, y potencialmente formatos médicos como DICOM.
        ▪ Videos: Muestra únicamente archivos de video, lo que podría incluir grabaciones de cirugías, estudios de endoscopias, o videos educativos asociados al paciente.

2.4. Flujos de Usuario Clave

Esta sección describe los flujos de usuario más importantes dentro del Área de Contenido Principal del Dashboard de Administración, el cual es el "Núcleo Administrativo Central" y el principal entregable de la Fase 1 del proyecto. El Dashboard está diseñado para optimizar la gestión de pacientes y mejorar la eficiencia operativa, abordando la dispersión de información y los procesos manuales.

A. Añadir un Nuevo Paciente

Este flujo permite al personal del consultorio registrar nuevos pacientes en el sistema.

1. Acceso a la Función: El usuario navega a la "Barra de Navegación Lateral" y selecciona la opción "Pacientes", luego hace clic en "Nuevo Paciente". Alternativamente, puede hacer clic directamente en el botón principal "+ Nuevo Registro" ubicado en la "Barra de Acciones" del módulo "Base de Datos".
2. Pantalla de Creación: Se abre un formulario o modal titulado "Creación de Perfil de Paciente".
3. Ingreso de Datos: El usuario debe ingresar los datos demográficos y de contacto del nuevo paciente, como nombre, apellido, fecha de nacimiento, teléfono, correo electrónico y dirección.
4. Validaciones: Si el usuario intenta guardar el registro sin completar campos obligatorios o con formatos incorrectos (ej., un email inválido), el sistema mostrará

mensajes de error claros, como "Campo 'Nombre' requerido" o "Formato de email inválido".
5. Guardado: Una vez que se ingresan los datos válidamente, el usuario hace clic en el botón "Guardar Paciente".
6. Resultado Esperado: La tabla del módulo "Base de Datos" se actualiza automáticamente con el nuevo registro de paciente, el cual aparecerá en estado "Active".
B. Buscar el Historial de un Paciente (Interacción con Asistente Virtual)
Este flujo demuestra la capacidad de búsqueda inteligente del Asistente Virtual integrado con Gemma 3 para acceder rápidamente a la información del paciente.
1. Acceso al Asistente: El usuario se dirige al módulo "Asistente virtual" en el Área de Contenido Principal. Se puede interactuar a través de un icono o una sección dedicada al chat.
2. Interacción de Búsqueda: El usuario utiliza lenguaje natural para realizar una consulta de búsqueda, por ejemplo, "Hola, estoy buscando un archivo me urge", "Recuerdas el nombre del paciente?", o "Si!! es Laura Vargas". El Asistente Virtual está diseñado para comprender y procesar estas consultas complejas utilizando la IA.
3. Recuperación Inteligente: El Asistente Virtual interpreta la solicitud y localiza la información o los archivos pertinentes en el sistema.
4. Respuesta y Acciones Contextuales: El asistente responde al usuario, por ejemplo, diciendo "Lo encontré", y ofrece acciones contextuales relacionadas con el recurso encontrado, como "¿te lo envío a tu correo o prefieres descargarlo?". Para la búsqueda de historial, el asistente provee un enlace directo al historial clínico o a la galería de archivos visuales del paciente. Es importante señalar que este chat funciona únicamente con texto y no tiene la capacidad de generar archivos de ningún formato.
C. Archivar Múltiples Pacientes
Este flujo permite la gestión en lote de registros de pacientes, realizando un borrado suave.
1. Selección de Registros: El usuario selecciona una o más filas en la "Tabla de Datos" del módulo "Base de Datos" utilizando los "Checkbox de Selección" ubicados en la primera columna de cada fila. Un "checkbox maestro" en el encabezado de la columna permite seleccionar o deseleccionar todos los registros visibles.
2. Activación de la Acción: Una vez seleccionadas las filas, el botón "Eliminar" (icono de papelera) en la "Barra de Acciones" se activa.
3. Confirmación: Al hacer clic en "Eliminar", aparece un modal de confirmación preguntando al usuario, por ejemplo, "¿Desea archivar los X registros seleccionados?".
4. Ejecución del Borrado Suave: Tras la confirmación del usuario, el sistema ejecuta un borrado suave (soft delete) de los registros. Esto significa que los registros no se eliminan permanentemente de la base de datos, sino que se marcan internamente como inactivos o archivados (ej., is_deleted = true).
5. Resultado Esperado: Las filas correspondientes desaparecen de la vista activa de la tabla, y su estado interno en la base de datos cambia a "Inactive".
D. Ver Detalles de Paciente
Este flujo permite acceder a una vista completa y detallada de la información de un paciente específico.
1. Acceso a los Detalles: Desde la "Tabla de Datos" del módulo "Base de Datos", el usuario puede hacer clic en el "ID/Nombre del Paciente" (texto en negrita) en la fila correspondiente al paciente deseado. Este elemento actúa como un enlace directo.
2. Pantalla de Detalles: Al hacer clic en el enlace, el sistema redirige al usuario a una vista de detalle completa del paciente, también conocida como la "ficha del paciente".
3. Contenido Esperado: En esta pantalla, el usuario espera ver el perfil completo del paciente con todos sus datos demográficos y de contacto, así como resúmenes de consultas y progreso de tratamiento, y acceso a la gestión documental avanzada, incluyendo la visualización de imágenes y videos relacionados. Esta pantalla también es el punto de partida para acciones adicionales como "Añadir a Historial".
E. Editar Registro de Paciente
Este flujo permite modificar la información existente de un paciente.
1. Acceso a la Edición: Desde la "Tabla de Datos" del módulo "Base de Datos", el usuario hace clic en el "Menú de Acciones" (icono de tres puntos) en la fila del paciente que desea editar.
2. Selección de Acción: Del menú desplegable, el usuario selecciona la opción "Editar Registro".
3. Proceso de Modificación: Se abre un formulario o modal, similar al de creación, pero pre-llenado con los datos actuales del paciente. El usuario realiza las modificaciones necesarias en los campos deseados [inferencia, similar a los formularios de creación o

historial clínico que implican entrada de datos].
4. Guardado de Cambios: Una vez realizadas las modificaciones, el usuario hace clic en el botón "Guardar" o "Actualizar" [inferencia]. Los cambios se envían de forma segura al Servidor de Aplicación (Backend) a través de APIs RESTful utilizando HTTPS (TLS 1.3).
5. Registro de Auditoría: El backend procesa la solicitud, actualiza la "Base de Datos Relacional (Cloud SQL)" y registra cronológicamente esta acción de modificación de datos en el "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)", que son inmutables.
6. Confirmación: El sistema muestra una confirmación visual al usuario, indicando que los cambios han sido guardados exitosamente.
F. Ver Historial Clínico Completo
Este flujo permite una visualización detallada y cronológica del historial médico de un paciente.
1. Acceso al Historial: Desde la "Tabla de Datos" del módulo "Base de Datos", el usuario hace clic en el "Menú de Acciones" (icono de tres puntos) en la fila del paciente deseado. Alternativamente, desde la tabla "Pacientes Recientes" en el Área Central del Dashboard, puede hacer clic en la opción "Ver Historial".
2. Selección de Acción: Del menú desplegable, el usuario selecciona la opción "Ver Historial Clínico Completo".
3. Contenido Esperado: Se presenta una vista que contiene el registro cronológico y detallado de cada consulta, diagnóstico, notas de evolución, procedimientos y tratamientos del paciente. Este historial también permite la capacidad de adjuntar documentos relevantes, y se espera que se integre con el "Visor de Archivos Visuales" para mostrar imágenes y videos contextuales relacionados con el paciente.
G. Archivar un Solo Paciente
Este flujo permite archivar individualmente un registro de paciente mediante un borrado suave.
1. Acceso a la Acción: Desde la "Tabla de Datos" del módulo "Base de Datos", el usuario hace clic en el "Menú de Acciones" (icono de tres puntos) en la fila del paciente que desea archivar.
2. Selección de Acción: Del menú desplegable, el usuario selecciona la opción "Archivar". (Nota: La opción "Eliminar (Suave)" también está disponible y realiza la misma función de borrado lógico).
3. Confirmación (Implícito): Aunque no se especifica explícitamente para el archivado individual, se esperaría un modal de confirmación similar al de múltiples pacientes para prevenir acciones accidentales.
4. Ejecución del Borrado Suave: El registro del paciente se marca internamente como inactivo o archivado (is_deleted = true) en la base de datos.
5. Resultado Esperado: El estado del paciente en la tabla puede cambiar visualmente a "Inactive" (una píldora de color rojo/gris) o el registro puede desaparecer de la vista activa, dependiendo de los filtros aplicados.

2.4. Flujos de Usuario Clave (Continuación)
H. Eliminar Suave un Solo Paciente
Este flujo permite archivar individualmente un registro de paciente, realizando un borrado lógico o "suave" para mantener la integridad de los datos históricos.
1. Acceso a la Acción: Desde la "Tabla de Datos" del módulo "Base de Datos", el usuario hace clic en el "Menú de Acciones" (icono de tres puntos) ubicado en la fila del paciente que desea archivar.
2. Selección de Acción: Del menú desplegable que aparece, el usuario selecciona la opción "Archivar". Es importante notar que la opción "Eliminar (Suave)" también está disponible y realiza la misma función de borrado lógico.
3. Confirmación (Implícito): Aunque no se especifica explícitamente para el archivado individual, por buena práctica de diseño de UI/UX y para prevenir acciones accidentales, se esperaría un modal de confirmación similar al de archivado múltiple, preguntando si desea proceder con el archivado del registro.
4. Ejecución del Borrado Suave: El sistema marca el registro del paciente internamente como inactivo o archivado (ej., is_deleted = true) en la "Base de Datos Relacional (Cloud SQL)". Esto significa que los datos no se eliminan permanentemente.
5. Resultado Esperado: La fila correspondiente al paciente desaparecerá de la vista activa de la tabla, o su estado visual en la columna "Estado" cambiará a "Inactive" (una píldora de color rojo/gris), dependiendo de los filtros aplicados.
I. Filtrar Tabla de Datos
Este flujo permite al usuario refinar la información visible en la "Tabla de Datos" para

encontrar rápidamente registros específicos.
1. Acceso a la Función: El usuario localiza el "Icono de filtro" en la "Barra de Acciones" del módulo "Base de Datos" y hace clic sobre él.
2. Opciones de Filtro: Al hacer clic, se despliega un panel o modal que presenta "opciones avanzadas" para filtrar la tabla.
3. Aplicación de Filtros: El usuario puede aplicar uno o más filtros basándose en diversas categorías, como "por fecha", "por estado" (ej., Activo, Inactivo), o "por tipo de consulta".
4. Resultado Esperado: La "Tabla de Datos" se actualiza dinámicamente, mostrando solo los registros que cumplen con los criterios de filtro seleccionados, lo que permite una visualización más enfocada y eficiente.
J. Exportar Datos de la Tabla
Este flujo proporciona la capacidad de extraer los datos de la "Tabla de Datos" para su uso externo o para fines de análisis.
1. Acceso a la Función: El usuario localiza el botón "Exportar" en la "Barra de Acciones" del módulo "Base de Datos" y hace clic sobre él.
2. Selección de Datos: El sistema permite al usuario exportar los "datos seleccionados" (si se han marcado filas específicas con los "Checkbox de Selección") o "todos los datos filtrados" (si se ha aplicado algún filtro).
3. Selección de Formato: El usuario puede elegir el formato de exportación deseado, como "CSV" o "Excel".
4. Resultado Esperado: Se inicia la descarga de un archivo que contiene los datos de la tabla en el formato seleccionado, facilitando su manipulación y análisis fuera del sistema.
K. Interactuar con Asistente Virtual para Tareas Específicas
Este flujo resalta la capacidad de la IA (Gemma 3) para automatizar tareas y proporcionar asistencia contextualizada, mejorando la eficiencia operativa.
1. Acceso a la Interacción: El usuario se dirige al módulo "Asistente virtual" en el Área de Contenido Principal del Dashboard. Esto se logra a través de un "icono o sección dedicada al chat".
2. Ejemplos de Interacciones y Resultados Esperados:
    ◦ Búsqueda Inteligente: El usuario puede preguntar por archivos o información específica usando lenguaje natural (ej., "Hola, estoy buscando un archivo me urge", "Recuerdas el nombre del paciente?", "Si!! es Laura Vargas"). El asistente responde localizando el recurso ("Lo encontré") y ofreciendo acciones contextuales (ej., "¿te lo envío a tu correo o prefieres descargarlo?").
    ◦ Redacción de Contenido: El usuario puede solicitar la redacción automatizada de "correos electrónicos", la "generación de hojas membretadas" o la creación de "recordatorios". La IA puede también generar "resúmenes de texto" o facilitar la "redacción de informes" a través del chat interno.
    ◦ Asistencia para Desarrollo/Capacitación: La IA puede generar "texto de ejemplo" (lorem ipsum más inteligente) para prototipos de UI, sugerir "copys" para botones y etiquetas si se entrena con el tono deseado, o generar "scripts SQL básicos" a partir de descripciones en lenguaje natural. Adicionalmente, puede colaborar en la generación de "preguntas frecuentes (FAQs)" a partir de la documentación técnica para el manual de usuario, e incluso crear un "chatbot de ayuda inicial" para la capacitación del personal.
3. Limitación: Es crucial señalar que este chat funciona "únicamente con texto" y "no tiene la capacidad de generar archivos de ningún formato".
L. Tomar y Guardar Notas en el Widget "Notas"
Este flujo permite al personal registrar información personal y recordatorios rápidos directamente en el Dashboard.
1. Acceso al Widget: El usuario localiza el widget titulado "Notas" en el Área de Contenido Principal del Dashboard.
2. Ingreso de Notas: El usuario interactúa con el área de texto del widget para escribir sus notas o recordatorios.
3. Guardado: Aunque no se especifica un botón explícito de "Guardar" para este widget, se esperaría que las notas se guarden automáticamente o al perder el foco del campo de texto, manteniendo la simplicidad de un "widget simple".
4. Resultado Esperado: Las notas ingresadas se almacenan y son visibles para el usuario en futuras sesiones. Esta información es, por lo general, "privada para el usuario y no está directamente ligada a un paciente", a menos que se implemente una función específica para adjuntar una nota a un paciente.
M. Navegar y Filtrar Archivos en el "Visor de Archivos Visuales"

Este flujo proporciona una forma eficiente de visualizar y organizar los documentos multimedia asociados a los pacientes.
1. Acceso al Visor: El usuario se dirige al módulo "Visor de archivos visuales" en el Área de Contenido Principal del Dashboard.
2. Navegación por Carrusel: Se presenta un "Carrusel de Vistas Previas" que muestra miniaturas de los archivos. El usuario puede navegar a través de la galería utilizando las "flechas a izquierda y derecha".
3. Filtros de Contenido: El usuario puede aplicar filtros para organizar los archivos según su origen o tipo:
    ◦ Cloud Drives: Muestra solo archivos provenientes de almacenamientos en la nube conectados (ej., Google Drive, OneDrive).
    ◦ Photos: Muestra únicamente imágenes (ej., JPG, PNG, DICOM).
    ◦ Videos: Muestra únicamente archivos de video (ej., grabaciones de cirugías, endoscopias).
4. Actualización Contextual: El contenido del visor se actualiza contextualmente según el paciente seleccionado en la tabla principal "Base de Datos" o la búsqueda realizada, asegurando que se muestren los archivos relevantes para el caso en cuestión.
5. Resultado Esperado: El usuario puede visualizar de forma organizada y eficiente todas las imágenes y videos relacionados con los expedientes de los pacientes.
N. Log In de Usuario
Este flujo es el punto de entrada fundamental al "Ecosistema Digital Inteligente para Cirugía Especial", específicamente al "Núcleo Administrativo Central" o Dashboard de Escritorio. Su propósito principal es autenticar al usuario y aplicar el control de acceso basado en roles.
1. Acceso a la Aplicación: El usuario inicia la aplicación de escritorio del "Núcleo Administrativo Central".
2. Pantalla de Login: Se presenta la "Pantalla: Login de Usuario", mostrando campos para "Nombre de Usuario" y "Contraseña".
3. Ingreso de Credenciales: El usuario ingresa su "Nombre de Usuario" y "Contraseña" en los campos designados.
4. Intento de Inicio de Sesión: El usuario hace clic en el botón "Iniciar Sesión".
5. Comunicación con el Backend: Las credenciales se envían de forma segura al Servidor de Aplicación (Backend) alojado en Google Cloud Platform (GCP). Esta comunicación utiliza HTTPS (TLS 1.3) para cifrar los datos en tránsito, protegiendo las credenciales de usuario de ser interceptadas.
6. Autenticación y Autorización en Backend:
    ◦ El backend (construido con Python/Flask/Django o Node.js/Express) recibe las credenciales.
    ◦ Verifica las credenciales contra la "Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL)". Los datos sensibles como contraseñas se almacenan cifrados en reposo en la base de datos (con un campo hash_password que sugiere el uso de funciones de hash).
    ◦ Una vez autenticado, el backend consulta el "Sistema de Roles" (ROLES en el modelo de datos) para determinar los permisos del usuario (ej., "Admin Principal", "Admin Secundario", "Asistente", "Invitado (Solo Lectura)").
7. Registro de Auditoría: El "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)" registra cronológicamente el intento de login (exitoso o fallido), la actividad del usuario y los cambios realizados en el sistema. Estos logs son inmutables (Write Once, Read Many - WORM), asegurando su integridad.
8. Redirección al Dashboard o Manejo de Errores:
    ◦ Login Exitoso: El usuario es redireccionado sin problemas a la "Pantalla: Dashboard Principal (Vista General)". La cabecera del dashboard mostrará el usuario logueado y su rol, y la barra de navegación lateral presentará las secciones a las que tiene acceso según sus permisos.
    ◦ Credenciales Incorrectas: La aplicación permanece en la pantalla de Login y mostrará un mensaje de error claro y conciso, por ejemplo: "Usuario o contraseña incorrectos".
    ◦ Campos Vacíos/Formato Inválido: Se esperaría una validación similar a otros formularios con mensajes de "Campo 'Nombre de Usuario' requerido" o "Campo 'Contraseña' requerido".
    ◦ Problemas de Conectividad: Un mensaje de error general sobre problemas de conexión o indisponibilidad del servicio. Se recomienda una conexión a internet estable de al menos 250 Mbps para un rendimiento óptimo.
O. Log Out de Usuario
Este flujo permite al usuario finalizar su sesión de forma segura en el Dashboard de

Escritorio.
1. Acceso a la Acción: El usuario se dirige a la "Cabecera" (Header Superior) del Dashboard Principal.
2. Selección de "Cerrar Sesión": En la cabecera, el usuario localiza y hace clic en la opción "Cerrar Sesión".
3. Proceso de Cierre de Sesión: El sistema procede a finalizar la sesión del usuario. (Implícitamente, esto implicaría la invalidación de cualquier token de sesión en el backend y la limpieza de datos de sesión en el frontend).
4. Resultado Esperado: El usuario es automáticamente redirigido a la "Pantalla: Login de Usuario".
P. Acceder a Mi Perfil
Este flujo permite al usuario acceder a su información de perfil dentro del sistema.
1. Acceso a la Función: Desde la "Cabecera" del Dashboard Principal, el usuario hace clic en el "Perfil de Usuario (Avatar)".
2. Selección de Opción: Al hacer clic en el avatar, se despliega un menú contextual con varias opciones. El usuario selecciona la opción "Mi Perfil".
3. Resultado Esperado: El sistema redirige al usuario a una pantalla o modal donde puede visualizar su información de perfil.
Q. Acceder a Configuración
Este flujo permite al usuario acceder a las configuraciones del sistema o personales disponibles.
1. Acceso a la Función: Desde la "Cabecera" del Dashboard Principal, el usuario hace clic en el "Perfil de Usuario (Avatar)".
2. Selección de Opción: Del menú desplegable que aparece, el usuario selecciona la opción "Configuración".
3. Resultado Esperado: El sistema redirige al usuario a una pantalla o modal donde puede acceder y modificar las configuraciones pertinentes.

3.0. Requisitos No Funcionales
Los requisitos no funcionales definen las cualidades y características del sistema que no están directamente relacionadas con la funcionalidad específica, sino con el rendimiento, la usabilidad, la seguridad, la fiabilidad y la escalabilidad. Son cruciales para garantizar que el sistema cumpla con las expectativas de calidad y eficiencia operativa del consultorio del Dr. Sánchez García.
3.1. Rendimiento
El sistema debe operar de manera fluida y eficiente, incluso bajo cargas de datos significativas.
• Requisito: El Dashboard y los servicios de backend deben mantener tiempos de respuesta óptimos (inferiores a 2 segundos para la mayoría de las operaciones críticas y menos de 5 segundos para operaciones complejas como búsquedas o exportaciones de grandes volúmenes de datos) con un volumen de datos de hasta 2 Terabytes (TB). Gran parte de este volumen estará compuesta por archivos visuales y documentos (imágenes, videos, PDF) almacenados en Cloud Storage, mientras que los metadatos y la información estructurada se gestionarán en la Base de Datos Relacional (Cloud SQL). Se recomienda una conexión a internet estable de al menos 250 Mbps para un rendimiento óptimo del ecosistema, dada su dependencia de la infraestructura en la nube.
3.2. Usabilidad
El sistema debe ser intuitivo y fácil de usar, minimizando la curva de aprendizaje para el personal.
• Requisito: El Dashboard debe ofrecer una experiencia de usuario "excepcionalmente intuitiva", reflejando la calidad y el prestigio de la práctica médica. Esto implica una coherencia visual integral a través del estilo "Glassmorphism Oscuro", diseñado para ofrecer "comodidad visual y una jerarquía de información clara". La interfaz debe proporcionar retroalimentación clara a las interacciones del usuario (ej., aumento de brillo o ligero hundimiento al pulsar un botón), y el diseño debe asegurar una "legibilidad óptima" con texto e iconos nítidos y brillantes. El diseño busca mitigar la "resistencia o dificultad con la tecnología" por parte del personal.
3.3. Seguridad
La protección de la información sensible del paciente y el acceso controlado son requisitos primordiales.
• Requisito: El sistema debe garantizar la confidencialidad, integridad y disponibilidad de todos los datos sensibles del paciente, abordando directamente las "preocupaciones de

seguridad de datos". Esto se logra mediante:
    ◦ Control de Acceso Basado en Roles (RBAC): Cada usuario, al iniciar sesión, es asociado a un rol específico ("Admin Principal", "Admin Secundario", "Asistente", "Invitado") que define sus permisos sobre la información y las funcionalidades del Dashboard.
    ◦ Cifrado de Datos: Toda la comunicación entre la aplicación de escritorio y la infraestructura en la nube debe cifrarse mediante HTTPS (TLS 1.3) para proteger los datos en tránsito. Además, todos los datos sensibles del paciente y las contraseñas (como hash) deben almacenarse cifrados en reposo tanto en la "Base de Datos Relacional (Cloud SQL)" como en el "Almacenamiento de Archivos (Cloud Storage)".
    ◦ Auditoría Integral: Cada acción relevante y cada intento de inicio de sesión (exitoso o fallido) debe registrarse cronológicamente en un "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)". Estos logs deben ser inmutables (Write Once, Read Many - WORM) para asegurar su integridad y facilitar la detección de anomalías.

3.4. Fiabilidad

El sistema debe ser robusto y capaz de funcionar de manera consistente y sin interrupciones.

• Requisito: El Dashboard y su infraestructura subyacente en la nube deben operar de forma estable y continua, minimizando el tiempo de inactividad. La arquitectura de nube está diseñada para alta disponibilidad. El plan de mantenimiento y soporte mensual, que incluye "mantenimiento proactivo" y "soporte técnico 24/7", garantiza el "óptimo funcionamiento" y la "evolución continua" de la aplicación, reforzando la fiabilidad del ecosistema.

3.5. Escalabilidad

El sistema debe ser capaz de crecer y adaptarse a futuras demandas de volumen de datos y usuarios sin comprometer el rendimiento.

• Requisito: El sistema debe poder manejar un crecimiento futuro en el volumen de pacientes, el número de usuarios y la cantidad de datos almacenados (incluyendo archivos multimedia) sin degradación del rendimiento. La elección de Google Cloud Platform (GCP) como proveedor de nube principal proporciona una "escalabilidad inherente" que permite al sistema expandirse para manejar una mayor demanda. El diseño del backend, potencialmente utilizando arquitecturas de microservicios, y la capacidad de adquirir "almacenamiento adicional" más allá del terabyte inicial, aseguran que la plataforma pueda adaptarse a las necesidades futuras del consultorio.

Especificación Técnica del Frontend del Dashboard: Núcleo Administrativo Central

1. Introducción y Propósito

El presente documento tiene como objetivo principal detallar las especificaciones técnicas para la construcción del Frontend del "Dashboard de Escritorio", componente central del "Ecosistema Digital Inteligente para Cirugía Especial". Servirá como la guía definitiva para los equipos de diseño y desarrollo, asegurando una implementación coherente, de alta calidad y sin ambigüedades.

El Dashboard de Escritorio es el "Núcleo Administrativo Central" del ecosistema, diseñado para ser el corazón operativo del consultorio del Dr. Joel Sánchez García. Su propósito fundamental es centralizar toda la información crítica del consultorio en un solo lugar seguro y accesible, ofreciendo una visión general del estado actual y un acceso rápido a las funcionalidades clave. Su objetivo es optimizar la gestión de pacientes, mejorar la eficiencia operativa al reducir los procesos manuales y el riesgo de errores, y liberar tiempo valioso para el personal. Además, está concebido para ser un entorno digital potente, seguro y excepcionalmente intuitivo, reflejando la calidad y el prestigio de la práctica médica del Dr. Sánchez García.

2. Elección del Framework Frontend

Para el desarrollo del Frontend del Dashboard, se recomienda la utilización de un framework moderno y robusto, siendo React o Vue.js las opciones preferidas y viables. Estos frameworks son ampliamente adoptados en la industria debido a su popularidad, amplios ecosistemas de herramientas y librerías, y su eficiencia en la creación de interfaces de usuario complejas y reactivas. Permiten un desarrollo modular, facilitan la gestión del estado y ofrecen un rendimiento óptimo para aplicaciones dinámicas.

El Dashboard se empaquetará como una aplicación de escritorio utilizando Electron.js. Electron.js permite construir aplicaciones de escritorio nativas (compatibles con Windows y macOS) utilizando tecnologías web estándares como HTML, CSS y JavaScript. Esta elección es estratégica, ya que aprovecha la experiencia en desarrollo web para crear una aplicación de escritorio, lo que agiliza el proceso y garantiza una experiencia de usuario familiar y de alta calidad. La integración con el framework frontend elegido

(React/Vue.js) es fluida, permitiendo que la lógica de negocio y la interfaz se desarrollen de manera cohesiva para el entorno de escritorio.

3. Arquitectura y Estructura de la Aplicación

3.1. Diagrama de Arquitectura de Alto Nivel del Frontend y su Interacción

El Frontend del Dashboard de Escritorio actúa como la capa de presentación que permite la interacción del usuario con la lógica de negocio y los datos alojados en la nube. Se comunica de forma segura con el Backend a través de APIs RESTful.

```
graph TD
    A[Aplicación de Escritorio (Frontend)] -- HTTPS (TLS 1.3) --> B(API Gateway / Load Balancer - GCP)
    B --> C[Servidor de Aplicación (Backend)]
    C -- Lógica de Negocio --> D[Base de Datos Relacional (Cloud SQL - GCP)]
    C -- Almacenamiento --> E[Almacenamiento de Archivos (Cloud Storage - GCP)]
    C -- Logs --> F[Servicio de Logs de Auditoría (Cloud Logging - GCP)]
    C -- Integración IA --> G[Modelo de IA (Gemma 3)]

    D -- Cifrado en Reposo --> D
    E -- Cifrado en Reposo --> E
    G -- Consultas/Generación --> C
```

Explicación del Flujo:

1. El Usuario inicia la Aplicación de Escritorio (Frontend).
2. La aplicación se conecta de forma segura a la infraestructura en la nube de Google Cloud Platform (GCP).
3. Todas las solicitudes del Frontend hacia el Backend pasan por el API Gateway / Load Balancer, que actúa como el punto de entrada seguro y distribuye el tráfico eficientemente.
4. La comunicación entre el Frontend y la nube se realiza a través de HTTPS (TLS 1.3) para cifrar los datos en tránsito, protegiendo la información sensible como las credenciales del usuario.
5. El Servidor de Aplicación (Backend) procesa las solicitudes, interactúa con la Base de Datos Relacional (Cloud SQL) para datos estructurados (pacientes, citas, usuarios) y con el Almacenamiento de Archivos (Cloud Storage) para documentos digitales (PDFs, imágenes, videos). Los datos sensibles en ambos servicios de almacenamiento están cifrados en reposo.
6. Cada acción del usuario y cada intento de login (exitoso o fallido) se registra cronológicamente en el Servicio de Logs de Auditoría (Cloud Logging / Audit Logs), los cuales son inmutables.
7. El Backend también integra el modelo de Inteligencia Artificial Gemma 3 mediante APIs para potenciar funcionalidades de automatización y el chat interno.

3.2. Estructura Detallada de Directorios y Carpetas del Proyecto Frontend

Se propone la siguiente estructura de directorios, basada en principios de modularidad y separación de responsabilidades para facilitar la mantenibilidad, escalabilidad y colaboración del equipo:

```
src/
├── assets/                 # Contiene recursos estáticos como imágenes, iconos, fuentes
│   ├── images/
│   ├── icons/              # Iconos de estilo de línea (line-style) con grosor de trazo consistente [49-52]
│   └── fonts/              # Fuente Inter (sans-serif geométrica) [45, 50-54]
├── components/             # Componentes UI reutilizables y atómicos
│   ├── buttons/
│   ├── forms/
│   ├── inputs/
│   ├── layout/             # Componentes de layout (Header, Sidebar, MainContent) [52, 55-57]
│   └── common/             # Otros componentes generales
├── views/ (o pages/)       # Contiene las vistas principales o "pantallas" de la aplicación
│   ├── Auth/               # Pantalla de Login [3, 7, 28, 58-60]
│   ├── Dashboard/          # Pantalla Principal del Dashboard [4, 7, 8, 31, 36, 58]
│   ├── Patients/           # Gestión Integral del Expediente del Paciente [2, 61-66]
│   ├── Appointments/       # Módulo de Agenda y Citas [67-72]
│   ├── Documents/          # Gestión Documental Avanzada [2, 63, 72-75]
```

```
│   ├── Administration/      # Seguridad, Roles y Accesos Rápidos (Gestionar Usuarios, Ver Logs de Auditoría) [68, 71, 76-79]
│   ├── Reports/             # Módulo de Reportes [7, 58, 80-82]
│   └── AIAutomation/        # Inteligencia y Automatización Integrada (Chat Interno) [68, 71, 79, 83-85]
├── services/ (o api/)       # Lógica para interactuar con el Backend (APIs)
│   ├── auth.js              # Servicios de autenticación (Login, Logout) [86, 87]
│   ├── patients.js          # Servicios para la gestión de pacientes [88, 89]
│   ├── appointments.js      # Servicios para la gestión de citas [88, 89]
│   ├── documents.js         # Servicios para la gestión de documentos [88, 89]
│   ├── users.js             # Servicios para la gestión de usuarios y roles [88, 89]
│   └── ai.js                # Servicios para la interacción con la IA (Gemma 3) [88, 89]
├── styles/                  # Estilos globales y específicos
│   ├── globals.css          # Estilos CSS globales, incluyendo las Propiedades Personalizadas de CSS [49-51, 90]
│   ├── themes.css           # Definiciones de temas (Glassmorphism Oscuro) [91, 92]
│   ├── components.css       # Estilos para componentes reutilizables
│   └── views.css            # Estilos específicos para vistas/pantallas
├── utils/                   # Funciones de utilidad (ej. formateo de fechas, validación de formularios)
├── hooks/                   # Custom Hooks (si se usa React) para lógica reutilizable con estado
├── store/ (o context/)      # Gestión de estado global de la aplicación (ej. información del usuario logueado, roles)
└── App.js (o main.js)       # Punto de entrada principal de la aplicación
```

3.3. Descripción de la Lógica de Organización

• Modularidad y Reusabilidad: La aplicación se estructurará en módulos y componentes, lo que permite reutilizar el código en diferentes partes de la aplicación y facilita el desarrollo paralelo. Los componentes se diseñarán de forma atómica, encapsulando su propia lógica y estilos.
• Separación de Responsabilidades:
    ◦ Lógica de UI en Componentes/Vistas: Los componentes y las vistas (carpetas components y views) serán responsables de la presentación visual y la interacción directa con el usuario.
    ◦ Lógica de Negocio y Llamadas a API en Servicios: Toda la lógica de comunicación con el backend, incluyendo las llamadas a las APIs RESTful y el manejo de la autenticación/autorización, residirá en la carpeta services (o api). Esto asegura que la lógica de presentación esté desacoplada de la lógica de datos.
    ◦ Estilos Separados: Los estilos se gestionarán de forma centralizada en la carpeta styles, utilizando CSS Variables para mantener la coherencia visual del "Glassmorphism Oscuro" y facilitar futuras modificaciones. Esto evita la duplicación de código y asegura que el diseño sea adaptable y fácil de mantener.
• Manejo de Estado: Se implementará una estrategia de manejo de estado global (usando Context API/Redux para React, o Vuex para Vue.js) para gestionar datos críticos como la información del usuario logueado, su rol y los permisos de acceso, asegurando que esta información esté disponible en toda la aplicación de manera eficiente.
• Control de Acceso Basado en Roles (RBAC): El Frontend implementará la lógica para adaptar la interfaz de usuario (visibilidad de secciones, botones, etc.) basándose en el rol del usuario autenticado, cuyos permisos son validados por el Backend. Esto garantiza que solo se presenten las funcionalidades a las que el usuario tiene acceso según su perfil (Administrador Principal, Admin Secundario, Asistente, Invitado).

4.1. Layout y Contenedores

Estos componentes definen la estructura general de la aplicación y sus principales áreas de contenido.

a. Contenedor Principal (App Layout)

• Propósito y función: Sirve como el envoltorio de toda la aplicación de escritorio, estableciendo el fondo principal y la base para el efecto visual "Aurora".
• Variantes: N/A. Es el layout base.
• Estados visuales: N/A. Es un contenedor estático.
• Propiedades (props): children (para contener todos los demás componentes del Dashboard).
• Alineación visual con Glassmorphism: Presenta un fondo principal de gris carbón suave (#121212) para reducir la fatiga visual. Detrás de los paneles de vidrio, incorpora auras

difusas de azul oceánico profundo (#4A69FF) y violeta vibrante (#8C52FF), esenciales para el efecto "Aurora" que proporciona el fondo dinámico para el Glassmorphism.
b. Barra de Navegación Lateral (Sidebar)
• Propósito y función: Permite al usuario navegar a las secciones principales del sistema, como "Pacientes", "Agenda", "Reportes", "Documentos" y "Administración". Su contenido se adapta al rol del usuario logueado.
• Variantes: N/A. Es una estructura fija.
• Estados visuales: N/A. Su contenido cambia dinámicamente según el rol.
• Propiedades (props): sections (un array de objetos que define las secciones y sus sub-opciones accesibles para el usuario actual), currentUserRole (para adaptar las opciones de navegación).
• Alineación visual con Glassmorphism: Actúa como un panel lateral que sigue la estética "Glassmorphism Oscuro". Aunque no se especifica explícitamente como "vidrio esmerilado" en sí, se infiere que mantendrá la paleta de colores y la tipografía para una coherencia integral.
c. Cabecera (Header)
• Propósito y función: Ubicada en la parte superior, muestra la información del usuario logueado (nombre y rol) y proporciona la opción para "Cerrar Sesión".
• Variantes: N/A.
• Estados visuales: N/A.
• Propiedades (props): userName, userRole, onLogoutClick (función para manejar el cierre de sesión).
• Alineación visual con Glassmorphism: El texto y los iconos son nítidos y brillantes, utilizando el color principal de texto blanco roto (#F5F5F5) para alta legibilidad sobre el fondo oscuro o semi-transparente.
d. Área de Contenido Principal (Main Content Area)
• Propósito y función: Es el espacio más grande y dinámico del Dashboard, donde se muestran los paneles de información clave ("Citas del Día", "Métricas Rápidas", "Pacientes Recientes") y donde se cargan las vistas de las secciones seleccionadas desde la barra lateral.
• Variantes: N/A. Es el contenedor de contenido principal.
• Estados visuales: N/A. Su contenido cambia dinámicamente.
• Propiedades (props): children (para cargar el contenido de las diferentes secciones del Dashboard).
• Alineación visual con Glassmorphism: Este área es donde los paneles flotantes de "vidrio esmerilado" (Glassmorphism) con esquinas redondeadas y un sutil borde luminoso cobran vida, mostrando la información clave de forma jerárquica y visualmente cómoda.
e. Paneles/Tarjetas (Glassmorphism Panels/Cards)
• Propósito y función: Son los contenedores visuales principales para la información estructurada dentro del Área de Contenido Principal (ej., "Citas del Día", "Métricas Rápidas", "Pacientes Recientes").
• Variantes: Pueden variar en tamaño (ej., pequeños para métricas rápidas, grandes para listados de pacientes).
• Estados visuales: N/A. Son contenedores de información.
• Propiedades (props): title, content (el JSX/componente interno que muestra la información específica), onClick (si el panel es interactivo, como un acceso directo).
• Alineación visual con Glassmorphism: Representan la esencia del diseño "Glassmorphism Oscuro". Son semi-transparentes y desenfocados (usando backdrop-filter: blur(24px)). Tienen esquinas suavemente redondeadas (border-radius: 16px) y un sutil borde luminoso de 1px de color blanco semitransparente que "atrapa" la luz del fondo, creando una sensación de profundidad y jerarquía.

4.2. Componentes de Navegación
Estos componentes permiten la interacción del usuario con la barra lateral.
a. Elemento de Navegación (Sidebar Item)
• Propósito y función: Representa cada una de las opciones en la barra de navegación lateral (ej., "Pacientes", "Agenda", "Reportes"). Permite la selección de la sección principal y, si aplica, la expansión para mostrar sub-opciones.
• Variantes:
    ◦ Con Sub-opciones: Se puede expandir y contraer para mostrar opciones anidadas (ej., "Pacientes" > "Nuevo Paciente", "Buscar Paciente").
    ◦ Sin Sub-opciones: Un elemento de menú directo (ej., "Documentos").

• Estados visuales:
    ◦ Normal: Estado por defecto, icono y texto con el color secundario (#A8A8A8).
    ◦ Hover: Un sutil cambio de color del texto y/o icono a blanco roto (#F5F5F5) o un ligero efecto de brillo/hundimiento al pasar el cursor.
    ◦ Activo/Seleccionado: El elemento actualmente visible en el área de contenido principal. Texto y/o icono con el color primario (#F5F5F5) y posiblemente una barra lateral sutil (1px solid rgba(255, 255, 255, 0.18)) o un fondo sutilmente resaltado.
• Propiedades (props): icon (para el icono de estilo de línea), label (texto de la opción), path (ruta de navegación), hasSubOptions (booleano), subOptions (array de sub-opciones), isActive (booleano para estado activo), onClick (función para manejar la navegación).
• Alineación visual con Glassmorphism: Los iconos son de estilo de línea con un grosor de trazo consistente y terminaciones/esquinas suavemente redondeadas, utilizando el color principal de texto (#F5F5F5) para los estados activos. La tipografía "Inter" se usa para las etiquetas, asegurando legibilidad. La retroalimentación visual al interactuar (hover/activo) es sutil pero clara, manteniendo la estética pulcra y de alta calidad del Glassmorphism.

4.3. Componentes Interactivos Básicos

Estos son los bloques de construcción fundamentales para la interacción del usuario.

a. Botones (Button)

• Propósito y función: Permiten al usuario ejecutar acciones específicas o navegar a otras secciones. Son el elemento principal para la interacción directa.
• Variantes:
    ◦ Primario: Para acciones principales y destacadas (ej., "Iniciar Sesión", "Guardar Paciente", "+ Nuevo Registro"). Utiliza un relleno sólido de color de acento.
    ◦ Secundario: Para acciones menos prominentes, acciones alternativas o botones de cancelación (ej., "Cancelar", "Reprogramar"). Podrían tener un fondo transparente o un borde sutil para menor impacto visual.
    ◦ Terciario/Icono: Para acciones muy discretas o solo con un icono (ej., menú de acciones en tabla, iconos de acceso rápido).
• Estados visuales:
    ◦ Normal: Apariencia por defecto, con colores y sombreado definidos.
    ◦ Hover: Un sutil aumento de brillo o un ligero hundimiento al pasar el cursor, ofreciendo retroalimentación clara.
    ◦ Activo/Focus: Un borde o sombra más prominente, indicando que el botón está seleccionado o en uso.
    ◦ Disabled: Reduce la opacidad y desactiva la interacción (no responde a clics), indicando que la acción no está disponible.
    ◦ Loading: Muestra un spinner o indicador de carga dentro del botón o junto a él, indicando que la acción está en progreso.
• Propiedades (props): onClick (función a ejecutar), label (texto del botón), variant (primario, secundario, etc.), disabled (booleano), loading (booleano), icon (opcional).
• Alineación visual con Glassmorphism:
    ◦ Relleno Primario: Utiliza accent-blue (#4A69FF) o button-primary-fill (#4A69FF).
    ◦ Texto: text-primary (#F5F5F5) para alta legibilidad.
    ◦ Sombra: Un box-shadow suave para un efecto de brillo que contribuye a la sensación de flotación del Glassmorphism.
    ◦ Esquinas: Suavemente redondeadas (ej., border-radius: 8px).

b. Campos de Entrada de Texto (Input / Textarea)

• Propósito y función: Permiten al usuario introducir texto, números u otros datos en la interfaz (ej., "Nombre de Usuario", "Contraseña", campos de perfil de paciente).
• Variantes:
    ◦ Input (tipo text, password, number, email, date, etc.) para campos cortos y de una sola línea.
    ◦ Textarea para campos multi-línea, como "Notas de Evolución".
• Estados visuales:
    ◦ Normal: Apariencia por defecto, con fondo sutilmente contrastante o translúcido.
    ◦ Focus: Un borde o resplandor sutil que indica que el campo está activo para la entrada de texto.
    ◦ Disabled: Desactiva la entrada y reduce la opacidad.
    ◦ Error: Un borde o texto en color de error (ej., rojo) y un mensaje de validación debajo del campo (ej., "Campo 'Nombre' requerido", "Formato de email inválido").
    ◦ Filled: Cuando el campo contiene datos.

• Propiedades (props): value, onChange (función de manejo de entrada), label (etiqueta del campo), placeholder, type (para inputs), rows (para textarea), disabled (booleano), error (booleano), errorMessage (string).
• Alineación visual con Glassmorphism:
    ◦ Fondo: Apariencia limpia y minimalista, posiblemente un fondo sutilmente contrastante o translúcido que mantenga la estética de vidrio.
    ◦ Texto: Las etiquetas y el texto introducido utilizan el color principal de texto (--color-text-primary: #F5F5F5) para una legibilidad óptima sobre el fondo oscuro.
    ◦ Bordes: Posiblemente bordes suaves o inexistentes en estado normal, y un borde sutilmente luminoso en el estado focus para indicar la interacción.
c. Checkboxes (Checkbox)
• Propósito y función: Permiten seleccionar o deseleccionar opciones (ej., selección múltiple de filas en una tabla para acciones en lote).
• Variantes: Individual, Maestro (para seleccionar/deseleccionar todos en una tabla).
• Estados visuales:
    ◦ Sin marcar: Estado por defecto.
    ◦ Marcado: Cuando la opción ha sido seleccionada.
    ◦ Disabled: No permite la interacción.
• Propiedades (props): checked (booleano), onChange (función de manejo de cambio), disabled (booleano).
• Alineación visual con Glassmorphism:
    ◦ Marcado: El color de acento interactivo (--color-accent-interactive: #34D1F3) para el relleno, sin borde.
    ◦ Sin marcar: Fondo transparente y un borde de 1px de color secundario (--color-text-secondary).
    ◦ El contraste visual asegura que los estados sean fácilmente distinguibles en el entorno oscuro.
d. Iconos (Icon)
• Propósito y función: Representan visualmente acciones, categorías o información, mejorando la usabilidad y la comprensión rápida de la interfaz (ej., iconos de navegación, iconos de acciones).
• Variantes: Diversos iconos según su propósito (casa para inicio, lupa para búsqueda, papelera para eliminar, gráfico de barras para reportes, etc.).
• Estados visuales:
    ◦ Normal: Utiliza el color secundario (#A8A8A8) o un tono más tenue.
    ◦ Hover/Activo: Cambia al color principal de texto (#F5F5F5) o a un color de acento, indicando interactividad o selección.
• Propiedades (props): name (nombre del icono), size (tamaño), color (color, que puede ser sobreescrito por el estado), onClick (si el icono es interactivo).
• Alineación visual con Glassmorphism:
    ◦ Estilo: Iconos de estilo de línea (line-style) con un grosor de trazo consistente (aproximadamente 1.5px) y terminaciones/esquinas suavemente redondeadas. Esto contribuye a la estética limpia y moderna.
    ◦ Coloración: Principalmente de un solo color (monocromático), utilizando --color-text-primary o --color-text-secondary para mantener la coherencia y legibilidad sobre los fondos oscuros o semi-transparentes.
    ◦ Biblioteca: Se recomienda el uso de bibliotecas de iconos consistentes como Iconify o Feather Icons para uniformidad y escalabilidad.
e. Elementos de Tipografía (Text / Heading / Label)
• Propósito y función: Estructuran y presentan todo el contenido textual de la interfaz, desde títulos y encabezados hasta párrafos y etiquetas de campos, asegurando legibilidad y jerarquía.
• Variantes:
    ◦ Heading (H1, H2, etc.) para títulos y secciones importantes.
    ◦ Body para párrafos de texto y contenido principal.
    ◦ Label para etiquetas de campos de entrada o elementos más pequeños.
• Estados visuales: N/A para el componente en sí, pero su color y peso pueden cambiar contextualmente (ej., texto de error, texto deshabilitado).
• Propiedades (props): children (el contenido de texto), variant (H1, body, label), color (ej., primary, secondary, error), weight (bold, regular, etc.).
• Alineación visual con Glassmorphism:
    ◦ Fuente: Inter, una fuente sans-serif geométrica optimizada para UI, seleccionada por su legibilidad en entornos digitales.

    - Colores:
        - --color-text-primary: #F5F5F5 (blanco roto) para títulos, texto principal y elementos clave. Proporciona alto contraste sobre el vidrio oscuro y el fondo principal, superior a 4.5:1 (WCAG).
        - --color-text-secondary: #A8A8A8 (gris claro) para texto secundario, descripciones o ayuda, manteniendo el confort visual.
    - Escala Tipográfica: Definida con rem (ej., H1: 1.5rem/24px Bold; Body: 0.875rem/14px Regular/Bold; Label: 0.75rem/12px Medium).

4.4. Componentes de Visualización de Datos y Estructura Compleja

Estos componentes muestran información estructurada y permiten la interacción avanzada.

a. Tabla de Datos (DataTable)

- Propósito y función: Presenta información en un formato tabular organizado, como "Pacientes Recientes", permitiendo visualizar y gestionar grandes volúmenes de datos de manera eficiente.
- Variantes: Tabla estándar con columnas configurables.
- Estados visuales:
    - Normal: Filas y celdas con estilos predeterminados.
    - Fila Hover: Un ligero cambio de fondo o un sutil resplandor en la fila al pasar el cursor para indicar interactividad.
    - Fila Seleccionada: Resaltado del fondo de la fila para indicar que ha sido seleccionada (ej., mediante un checkbox).
    - Estado Vacío: Un mensaje claro (ej., "No se encontraron pacientes") y posiblemente un icono ilustrativo, en lugar de una tabla vacía.
    - Cargando: Un spinner o una superposición de carga sobre la tabla.
- Propiedades (props): data (array de objetos con los datos a mostrar), columns (definición de columnas: key, label, renderCell, sortable), onRowClick (para ver detalles), onSelectionChange (para checkboxes), isLoading (booleano), emptyMessage (string), pagination (objeto de paginación).
- Alineación visual con Glassmorphism:
    - Contenida dentro de un Panel/Tarjeta Glassmorphism.
    - Encabezados de Columna: Utilizan el color text-primary (#F5F5F5) y la fuente Inter en el peso adecuado.
    - Contenido de Celdas: El texto de datos principal utiliza text-primary. Información secundaria puede usar text-secondary (#A8A8A8).
    - Columna de Identificación: "ID/Nombre del Paciente" en negrita y funciona como enlace para ver el detalle completo.
    - Estados: Las píldoras de estado (State Pills) como "Active" (Verde) y "Inactive" (Rojo/Gris) utilizan colores acentuados para una identificación rápida.
    - Checkbox de Selección: En cada fila y un maestro en el encabezado, siguiendo el estilo Glassmorphism del checkbox.
    - Menú de Acciones por Fila: Un icono de tres puntos vertical (Icon) que al hacer clic despliega un menú contextual flotante (posiblemente un Modal o Panel Lateral con opciones como "Ver Detalles", "Editar Registro", "Archivar", "Eliminar (Suave)").

b. Paginación (Pagination)

- Propósito y función: Permite al usuario navegar a través de grandes conjuntos de datos presentados en tablas o listas, cargando el contenido en bloques (páginas).
- Variantes: Controles de navegación (anterior, siguiente, números de página, selector de tamaño de página).
- Estados visuales:
    - Número de Página Activa: Resaltado con un color de acento o un efecto de brillo sutil para indicar la página actual.
    - Controles Deshabilitados: (ej., "Anterior" en la primera página, "Siguiente" en la última página) con menor opacidad.
- Propiedades (props): currentPage, totalPages, onPageChange (función), pageSize (opcional), onPageSizeChange (opcional).
- Alineación visual con Glassmorphism: Botones y texto siguen el estilo de los demás componentes interactivos, con una retroalimentación visual clara y sutil.

c. Componente de Filtros (Filter Panel/Modal)

- Propósito y función: Proporciona opciones para que el usuario refine la información mostrada en tablas o vistas (ej., filtrar pacientes por fecha de última consulta, tipo de cita).
- Variantes:
    - Panel desplegable: Que se abre in-situ desde un icono de filtro en la barra de

acciones de la tabla.
    ◦ Modal: Una ventana emergente con opciones de filtro avanzadas.
    ◦ Panel Lateral: Un panel que se desliza desde un lado de la pantalla (similar a la barra lateral de navegación).
• Estados visuales:
    ◦ Abierto/Cerrado: Transiciones suaves para abrir y cerrar el panel/modal.
    ◦ Filtros Aplicados: Indicadores visuales (ej., un punto de color en el icono de filtro) que muestran que hay filtros activos.
• Propiedades (props): isOpen (booleano), onClose (función), filterOptions (definición de campos de filtro: tipo, label), onApplyFilters (función), onClearFilters (función).
• Alineación visual con Glassmorphism:
    ◦ El Panel o Modal contenedor seguiría la estética de "vidrio esmerilado" del Glassmorphism (semi-transparente, desenfocado, esquinas redondeadas, borde luminoso).
    ◦ Los campos de entrada (Input), checkboxes y botones dentro del componente de filtro seguirían sus respectivos estilos Glassmorphism.

d. Modales (Modal)
• Propósito y función: Presentan contenido importante o solicitan interacción del usuario en una ventana flotante sobre el contenido principal, sin que el usuario abandone la vista actual (ej., formularios de creación/edición de paciente, confirmaciones de eliminación).
• Variantes:
    ◦ Formulario: Para la creación o edición de registros.
    ◦ Confirmación: Para acciones críticas (ej., "¿Desea archivar los X registros seleccionados?").
    ◦ Alerta/Información: Para mensajes importantes o notificaciones.
• Estados visuales:
    ◦ Abierto: Visible sobre un fondo semi-oscurecido (backdrop).
    ◦ Cerrado: Oculto.
• Propiedades (props): isOpen (booleano), onClose (función), title (título del modal), children (contenido del modal), onConfirm (función), onCancel (función), confirmButtonLabel, cancelButtonLabel.
• Alineación visual con Glassmorphism:
    ◦ El modal en sí es un Panel/Tarjeta Glassmorphism: semi-transparente y desenfocado (backdrop-filter: blur(24px)), con esquinas suavemente redondeadas (border-radius: 16px) y un sutil borde luminoso de 1px.
    ◦ El fondo del modal (el backdrop) se oscurecería sutilmente para enfocar la atención en la ventana emergente, manteniendo la sensación de profundidad.

e. Paneles Laterales (Side Panel / Drawer)
• Propósito y función: Ofrecen un espacio adicional para contenido o controles que no requieren una vista de página completa, como detalles adicionales de un elemento seleccionado, formularios de edición o filtros avanzados.
• Variantes: Deslizante desde la izquierda o la derecha.
• Estados visuales:
    ◦ Abierto: Visible, con el contenido principal posiblemente encogido o superpuesto.
    ◦ Cerrado: Oculto.
• Propiedades (props): isOpen (booleano), onClose (función), title (título del panel), children (contenido).
• Alineación visual con Glassmorphism:
    ◦ Similar a un Panel/Tarjeta Glassmorphism, pero extendido verticalmente. Mantendrá la semi-transparencia, el desenfoque y los bordes luminosos.
    ◦ La transición de apertura/cierre podría incluir una sutil animación de deslizamiento.

4.5. Componentes Específicos de Módulos
Estos componentes están diseñados para funcionalidades especializadas.

a. Interfaz del Asistente Virtual (AI Assistant Interface)
• Propósito y función: Permite al personal interactuar con el modelo de IA (Gemma 3) para realizar consultas rápidas, obtener resúmenes de texto o redactar informes. Es un chat interno que solo funciona con texto.
• Variantes: N/A.
• Estados visuales:
    ◦ Normal: Área de visualización de mensajes y campo de entrada activos.
    ◦ Escribiendo: Indicador de que el usuario está introduciendo texto.
    ◦ Recibiendo Respuesta/Cargando: Un indicador visual (ej., puntos suspensivos o spinner) cuando la IA está procesando o generando una respuesta.

    ◦ Errores: Mensajes que indican un problema con la consulta o la IA.
• Propiedades (props): messages (array de objetos {sender, text}), onSendMessage (función), isLoading (booleano), suggestedActions (array de strings para botones de acción sugerida).
• Alineación visual con Glassmorphism:
    ◦ El área de visualización de mensajes y el campo de entrada podrían estar contenidos dentro de un Panel/Tarjeta Glassmorphism.
    ◦ El campo de entrada seguiría el estilo de Input Glassmorphism.
    ◦ Los botones de acción sugerida (ej., "te lo envío a tu correo o prefieres descargarlo?") seguirían el estilo de Button (quizás Secundario o con el color accent-interactive).
b. Componentes del Visor de Archivos Visuales (Visual File Viewer)
• Propósito y función: Permite visualizar y navegar por archivos visuales (imágenes y videos cortos) asociados a los expedientes de los pacientes, como RX o grabaciones de cirugías.
• Variantes:
    ◦ Carrusel Contenedor: Para mostrar miniaturas y navegar entre ellas.
    ◦ Elementos de Miniatura Individual: Representación visual de cada archivo.
    ◦ Vista Ampliada: El área principal donde se muestra la imagen o video seleccionado a tamaño completo.
    ◦ Flechas de Navegación: Para avanzar o retroceder en el carrusel/galería.
    ◦ Controles de Filtro de Origen/Tipo: Botones o checkboxes para filtrar por "Cloud Drives", "Photos", "Videos".
• Estados visuales:
    ◦ Normal: Archivos visibles y navegables.
    ◦ Cargando: Indicador de carga al abrir un archivo grande.
    ◦ Vacío: Mensaje si no hay archivos visuales asociados al paciente.
• Propiedades (props): files (array de objetos de archivo), currentFileIndex (índice del archivo actual), onNavigate (función), onFilterChange (función), selectedFilter (string).
• Alineación visual con Glassmorphism:
    ◦ El Carrusel o Visor principal podría ser un Panel/Tarjeta Glassmorphism, aprovechando la transparencia y el desenfoque para integrarse con el fondo Aurora.
    ◦ Las Miniaturas y la Vista Ampliada estarían contenidas dentro de este panel.
    ◦ Las Flechas de Navegación utilizarían Iconos de estilo de línea.
    ◦ Los Controles de Filtro (botones/checkboxes) seguirían el diseño Glassmorphism de sus respectivos componentes.
c. Componentes para Widgets de Servicios Externos (External Service Widgets)
• Propósito y función: Proporcionan acceso rápido a herramientas de uso común como Gmail, Google Drive, o aplicaciones de Office, integradas visualmente en la interfaz.
• Variantes: Botones con iconos, o pequeños paneles con un icono y una etiqueta.
• Estados visuales:
    ◦ Normal: Icono y etiqueta visibles.
    ◦ Hover: Un sutil brillo o cambio de color al pasar el cursor.
• Propiedades (props): serviceName (nombre del servicio), icon (icono del servicio), onClick (función para abrir el servicio/aplicación), url (opcional, si es una integración web).
• Alineación visual con Glassmorphism:
    ◦ Podrían ser pequeños Paneles/Tarjetas Glassmorphism flotantes, cada uno representando un acceso directo, o simplemente Iconos estilizados con etiquetas de texto.
    ◦ El uso de iconos de estilo de línea y el color text-primary asegurarían la consistencia.
4.6. Componentes de Feedback y Contenedores Generales
a. Componentes de Feedback Visual (Loading Spinner / Toast Notification)
• Propósito y función: Informan al usuario sobre el estado del sistema, como operaciones en curso (carga) o resultados de acciones (notificaciones de éxito/error).
• Variantes:
    ◦ Spinner de Carga: Un indicador animado circular que aparece durante operaciones asíncronas.
    ◦ Mensajes de Notificación/Toast: Pequeños banners que aparecen brevemente para confirmar una acción o mostrar un error (ej., "Guardado con éxito", "Error al subir el archivo").
• Estados visuales:
    ◦ Visible: El componente de feedback se muestra.

    ◦ Oculto: El componente de feedback no es visible.
• Propiedades (props):
    ◦ Spinner: isVisible (booleano), size, color.
    ◦ Toast: isVisible (booleano), message (string), type (success, error, info, warning), duration (tiempo en ms).
• Alineación visual con Glassmorphism:
    ◦ Spinners: Podrían ser círculos sutilmente luminosos o con el color accent-blue o accent-interactive.
    ◦ Toast: Aparecerían como pequeños Paneles/Tarjetas Glassmorphism transitorios, con el desenfoque y los bordes luminosos. El texto dentro utilizaría text-primary y el color del fondo del toast podría variar ligeramente según el tipo (ej., un toque de verde para éxito, rojo para error), manteniendo la opacidad y el desenfoque Glassmorphism.
b. Formularios (Form Container)
• Propósito y función: Agrupan un conjunto de campos de entrada y botones de acción relacionados para la recolección estructurada de datos (ej., "Creación de Perfil de Paciente", "Agendar Nueva Cita").
• Variantes: Formulario de creación, formulario de edición, formulario de búsqueda.
• Estados visuales:
    ◦ Normal: Todos los campos y botones son visibles y están listos para la interacción.
    ◦ Validando/Enviando: Puede deshabilitar los campos o mostrar un spinner en el botón de submit.
    ◦ Error de Validación: Muestra mensajes de error junto a los campos correspondientes.
• Propiedades (props): onSubmit (función al enviar), children (los campos y botones internos), title (título del formulario).
• Alineación visual con Glassmorphism:
    ◦ El formulario en sí estaría contenido dentro de un Panel/Tarjeta Glassmorphism o un Modal (que es un panel Glassmorphism), dependiendo de su ubicación y flujo.
    ◦ Los Campos de Entrada, Checkboxes y Botones dentro del formulario seguirían sus respectivos estilos Glassmorphism, asegurando una experiencia visual coherente y agradable.
5. Implementación del Sistema de Diseño (Glassmorphism Oscuro)
El estilo visual "Glassmorphism Oscuro" es fundamental para la experiencia de usuario del ecosistema, especialmente en el Dashboard de Escritorio, y no es solo estético sino funcional, diseñado para ofrecer comodidad visual y una jerarquía de información clara.
• Estrategia de implementación de estilos: La estrategia de implementación de estilos se centra en el uso de Propiedades Personalizadas de CSS (CSS Variables), declaradas bajo el selector :root. Esta aproximación centraliza los valores de diseño, facilitando la mantenibilidad, los cambios globales y futuras tematizaciones del ecosistema. Aunque no se especifica explícitamente el uso de CSS Modules, SCSS o Styled Components, la adopción de variables CSS sugiere un enfoque modular y escalable para la gestión de estilos.
• Definición y uso de Variables CSS (Tokens de Diseño): Los "tokens de diseño" se codificarán como variables CSS para mantener la coherencia y facilitar su aplicación a través de todos los componentes.
    ◦ Paleta de Colores:
        ▪ --color-background-main: #121212 (Fondo principal, gris carbón suave).
        ▪ --color-surface-glass-base: 29, 35, 50 (Valores RGB para la base del vidrio, azul/gris oscuro desaturado, utilizado con opacidad).
        ▪ --color-border-glass-base: 255, 255, 255 (Valores RGB para el blanco, utilizado para bordes sutiles).
        ▪ --color-text-primary: #F5F5F5 (Texto principal, blanco roto para alta legibilidad).
        ▪ --color-text-secondary: #A8A8A8 (Texto secundario/ayuda, gris claro).
        ▪ --color-accent-blue: #4A69FF (Color de acento primario, azul oceánico profundo).
        ▪ --color-accent-purple: #8C52FF (Color de acento secundario, violeta vibrante).
        ▪ --color-accent-interactive: #34D1F3 (Color para elementos interactivos, azul brillante).
        ▪ --color-button-primary-fill: #4A69FF (Color de relleno para botones primarios).
    ◦ Escala Tipográfica (Fuente: Inter, sans-serif geométrica optimizada para UI):
        ▪ --font-size-h1: 1.5rem (24px), 700 (Bold).
        ▪ --font-size-h2: 1.125rem (18px), 600 (Semibold).
        ▪ --font-size-body: 0.875rem (14px), 700 (Bold) o 400 (Regular).
        ▪ --font-size-label: 0.75rem (12px), 500 (Medium).
        ▪ Los colores de texto principales y secundarios se definirán con --color-text-

primary y --color-text-secondary respectivamente.
    - Espaciado y border-radius:
        - --border-radius-card: 16px (Para paneles flotantes, esquinas suaves).
        - --border-radius-button: 8px (Para botones).
        - Aunque no se definen variables específicas para un sistema de espaciado general como --space-unit, el relleno (padding: 24px) se utiliza de manera consistente en los paneles de vidrio.
• Implementación técnica detallada del efecto Glassmorphism en los componentes de panel/tarjeta: Los paneles y tarjetas del Dashboard, como el contenedor de login o las áreas de información clave, implementarán el efecto Glassmorphism utilizando las siguientes propiedades CSS:
    - background-color: rgba(var(--color-surface-glass-base), 0.6): Establece un fondo semi-transparente con un 60% de opacidad sobre el color base de la superficie de vidrio.
    - backdrop-filter: blur(24px): Aplica un desenfoque significativo al contenido que se encuentra detrás del panel, creando el efecto "esmerilado".
    - -webkit-backdrop-filter: blur(24px): Incluye el prefijo de navegador para asegurar la compatibilidad con Safari.
    - border-radius: var(--border-radius-card): Redondea suavemente las esquinas de los paneles.
    - border: 1px solid rgba(var(--color-border-glass-base), 0.18): Delinea los paneles con un borde delicado de 1 píxel, de color blanco semi-transparente (18% de opacidad), que "atrapa" la luz del fondo y define la forma del panel.
    - padding: 24px: Asegura un espaciado interno consistente dentro de los paneles.
• Implementación técnica del fondo Aurora UI: La Interfaz Aurora no es un adorno, sino un componente funcional indispensable que proporciona la "materia prima" visual para que el filtro backdrop-filter del Glassmorphism actúe. Los dos estilos son codependientes: la Aurora UI proporciona la "luz" y el Glassmorphism la "lente".
    - Se creará un fondo dinámico utilizando auras difusas de azul oceánico profundo (#4A69FF) y violeta vibrante (#8C52FF).
    - Técnicamente, esto se logra con elementos div posicionados absolutamente con un fuerte filter: blur() o con múltiples radial-gradients. Este fondo debe tener suficiente variación tonal y de color para que el efecto de vidrio sea visible y efectivo.
• Aplicación de estilos a los componentes reutilizables: El sistema de diseño garantiza una apariencia Glassmorphism coherente en todos los componentes interactivos:
    - Botones (Iniciar Sesión):
        - background-color: var(--color-button-primary-fill): Para un relleno sólido de alto contraste.
        - border-radius: var(--border-radius-button): Esquinas suaves.
        - padding: 12px 24px: Espaciado interno.
        - color: var(--color-text-primary): Para la etiqueta de texto.
        - box-shadow: 0 4px 12px rgba(var(--color-accent-blue), 0.3): Un suave efecto de brillo.
        - La interacción al pulsarlo provocará un sutil aumento de brillo o un ligero hundimiento, ofreciendo una retroalimentación clara.
    - Campos de Entrada (Nombre de Usuario, Contraseña):
        - Diseñados con una apariencia limpia y minimalista, posiblemente con un fondo sutilmente contrastante o translúcido que mantenga la estética de vidrio.
        - El texto de las etiquetas y el que el usuario introduce utilizará --color-text-primary (#F5F5F5) para una legibilidad óptima sobre el fondo oscuro.
    - Casillas de Verificación (Checkboxes):
        - Marcado: background-color: var(--color-accent-interactive), border: none.
        - Sin marcar: background-color: transparent, border: 1px solid var(--color-text-secondary).
    - Tablas (ej., PACIENTES RECIENTES): Aunque no se especifican propiedades Glassmorphism para las tablas en sí, los elementos de texto y los iconos dentro de ellas seguirán la paleta de colores y la tipografía definidas por las variables CSS, utilizando --color-text-primary y --color-text-secondary para mantener la coherencia y legibilidad.
• Uso de Iconografía:
    - Estilo: Los iconos serán de estilo de línea (line-style), con un grosor de trazo consistente (aproximadamente 1.5px) y terminaciones/esquinas suavemente redondeadas.
    - Color: Utilizarán un solo color, var(--color-text-primary), para una excelente integración con el modo oscuro.
    - Recomendación: Se recomienda el uso de bibliotecas como Iconify o Feather Icons para

asegurar consistencia y escalabilidad.
• Consideraciones específicas para el modo oscuro avanzado: El diseño del "Glassmorphism Oscuro" incorpora principios avanzados para asegurar la legibilidad y el confort visual en un tema oscuro:
    ◦ Evitar el negro puro: El fondo principal es un gris carbón muy oscuro (#121212), no negro puro. Esto reduce la fatiga visual y permite la percepción de profundidad y sombras sutiles.
    ◦ Colores de acento desaturados: Los tonos de azul (#4A69FF) y violeta (#8C52FF) utilizados para las "Auras" y acentos están desaturados. Esto previene la "vibración" óptica y mantiene un confort visual general, especialmente en exposiciones prolongadas.
    ◦ Texto de alto contraste pero suavizado: El texto principal utiliza un blanco roto o gris muy claro (#F5F5F5), lo que suaviza el contraste general sin comprometer la legibilidad. Esto ayuda a reducir el deslumbramiento y cumple con una relación de contraste superior a 4.5:1 (WCAG), un estándar clave de accesibilidad.
    ◦ Comunicación de profundidad sin sombras: En entornos oscuros, las sombras son ineficaces. El diseño utiliza un sistema multifacético para comunicar la elevación:
        ▪ La luz como indicador de elevación: Las superficies que "se acercan" al usuario se vuelven más claras.
        ▪ La superposición como señal de profundidad: El propio efecto de desenfoque del Glassmorfismo crea una separación perceptual entre el primer plano nítido y el fondo desenfocado.
        ▪ La definición del borde como sustituto de la sombra: El sutil borde luminoso define claramente el contorno del panel, separándolo del fondo donde una sombra no sería visible.
    ◦ Accesibilidad: Se aborda la accesibilidad mediante un enfoque dual de contraste:
        ▪ Contraste de Contenido (Alto): Asegura alta legibilidad entre el texto/iconos y su fondo inmediato (el panel de vidrio oscuro y semi-transparente).
        ▪ Contraste Estructural (Bajo): Mantiene el contraste entre el panel de vidrio y el fondo principal deliberadamente bajo para preservar el efecto flotante y esmerilado.

6. Gestión del Estado

La "Pantalla: Dashboard Principal (Vista General)" actúa como un centro de control dinámico que muestra datos relevantes en tiempo real y proporciona puntos de entrada intuitivos a todas las funcionalidades principales del sistema. Para que la interfaz de usuario (UI) se mantenga sincronizada y reactiva a las acciones del usuario, es fundamental una buena gestión del estado. Dado que el frontend del Dashboard se desarrollará con frameworks como React, Vue.js o similares, la elección de una estrategia de gestión del estado es vital.

Elección de una librería o patrón para la gestión del estado global de la aplicación: Aunque los documentos no especifican una librería concreta, la necesidad de una UI reactiva y sincronizada sugiere la implementación de patrones de gestión de estado robustos.
• Para un frontend basado en React, se podrían considerar:
    ◦ Context API + useReducer: Una solución nativa de React adecuada para la gestión de estados globales de complejidad media. Es ideal para datos de usuario autenticado o estados de modales/paneles que necesitan ser accesibles por muchos componentes sin una prop drilling excesiva.
    ◦ Redux (con Redux Toolkit) o Zustand: Para escenarios de mayor complejidad, donde la gestión de datos asíncronos (llamadas a la API) y la necesidad de un flujo de datos predecible sean prioritarias. Proporciona una "fuente única de verdad" para el estado global y herramientas potentes para depuración y escalabilidad.
• Para un frontend basado en Vue.js, se podrían considerar:
    ◦ Vuex o Pinia: Los gestores de estado oficiales de Vue.js. Proporcionan un almacén centralizado de estado que facilita la reactividad y la depuración, siendo esenciales para aplicaciones Vue a gran escala.

La elección específica dependerá de la complejidad exacta de la lógica de negocio y el volumen de datos que se esperen manejar en tiempo real. Para un "Núcleo Administrativo Central", que es la "columna vertebral y la base de datos central de todo el ecosistema", una solución más estructurada como Redux/Vuex/Pinia podría ofrecer mayor mantenibilidad y escalabilidad a largo plazo.

Descripción de las partes principales del estado que necesitarán ser gestionadas globalmente: Para el Dashboard, varias piezas de información son críticas y requerirán un estado global para garantizar la coherencia y la reactividad:
• Datos del Usuario Autenticado y su Rol: La cabecera del dashboard mostrará el usuario

logueado y su rol, y la barra de navegación lateral presentará las secciones a las que tiene acceso según sus permisos. Esta información es fundamental para la adaptación de la interfaz y el control de acceso basado en roles.
• Estado de Carga y Error Global de APIs: Las comunicaciones con el backend pueden presentar "problemas de conectividad" o "errores de servidor". Es crucial tener un estado global para indicar cuándo se están realizando llamadas a la API (estado loading, para mostrar spinners o deshabilitar botones) y cuándo ha ocurrido un error (estado error, para mostrar mensajes de error genéricos).
• Datos de la Tabla de Pacientes y Otros Datos Dinámicos: La sección "PACIENTES RECIENTES" y las "CITAS DEL DÍA" muestran datos dinámicos que se actualizan en tiempo real o casi real. Estos datos, junto con las "MÉTRICAS RÁPIDAS", deben ser parte del estado global para que cualquier componente que los necesite pueda acceder a ellos de forma consistente.
• Estado de Visibilidad de Modales/Paneles: Si bien no se menciona explícitamente en los documentos, una interfaz de usuario compleja como el Dashboard a menudo utiliza modales o paneles laterales (como los de filtrado o creación de nuevos registros) cuya visibilidad puede ser gestionada a nivel global para una mejor coordinación entre componentes.
• Contenido de Texto Generado por IA: Las funcionalidades de Inteligencia y Automatización Integrada permiten generar texto (ej., borradores de correos, resúmenes de texto). Este contenido, antes de ser utilizado o guardado, podría residir temporalmente en el estado global.
Cómo se actualizará y accederá al estado desde diferentes componentes: La forma en que se actualiza y accede al estado dependerá del patrón o librería de gestión de estado elegida:
• Actualización:
    ◦ Acciones/Despachos: En patrones como Redux o Vuex, los componentes "despachan" (dispatch) acciones. Estas acciones son objetos que describen lo que sucedió (ej., USER_LOGIN_SUCCESS, FETCH_PATIENTS_START, FETCH_PATIENTS_SUCCESS). Los "reducers" (en Redux) o "mutations" (en Vuex) son funciones puras que toman el estado actual y una acción, y devuelven un nuevo estado, asegurando un flujo de datos unidireccional y predecible.
    ◦ Actualizadores de estado: Para Context API + useReducer, se utilizan funciones dispatch para enviar acciones al reducer. Para estados locales o de menor alcance, se usarán funciones setState de React o ref/reactive de Vue.
• Acceso:
    ◦ Selectores/Hooks: Los componentes accederán a porciones específicas del estado global mediante "selectores" (en Redux) o "hooks personalizados" (en React con Context/Zustand), o mediante funciones mapState o useStore (en Vuex/Pinia). Esto les permite "suscribirse" solo a la parte del estado que les interesa y re-renderizarse solo cuando esa parte cambia, optimizando el rendimiento.
7. Consumo de APIs del Backend
El Dashboard de Escritorio se comunica con el "Servidor de Aplicación (Backend)" a través de APIs RESTful. Este backend está alojado en Google Cloud Platform (GCP) y maneja la lógica de negocio, la interacción con la base de datos y otros servicios.
Estrategia para realizar las llamadas a los endpoints del backend desde el frontend: Se recomienda una estrategia que garantice la robustez, mantenibilidad y seguridad de las comunicaciones:
• Módulo de Servicios Dedicado: En lugar de realizar llamadas fetch o axios directamente desde los componentes, se creará un módulo de servicios o una capa de abstracción para las APIs. Este módulo contendría funciones específicas para cada endpoint (ej., api.getPatients(), api.createPatient(data), api.login(credentials)).
    ◦ Librería HTTP: Se puede utilizar fetch (API nativa del navegador) o una librería popular como axios. axios es preferible por su simplicidad, manejo automático de JSON, intercepción de solicitudes/respuestas y mejor manejo de errores.
• Endpoints: Se deben desarrollar endpoints de API robustos para las diferentes funcionalidades del Dashboard, como:
    ◦ GET /patients con parámetros de paginación, búsqueda y filtrado.
    ◦ POST /patients para crear nuevos registros.
    ◦ PUT /patients/{id} para actualizar.
    ◦ DELETE /patients/{id} para el borrado suave (soft delete).
    ◦ Endpoints específicos para la galería de archivos y las interacciones con el asistente virtual de IA.
    ◦ POST /api/v1/auth/login para la autenticación.
Cómo se manejarán los estados de carga (loading), error (error), y éxito (data) de las llamadas a la API en la interfaz de usuario: La gestión visual de estos estados es crucial

para una buena experiencia de usuario:
• Estado de Carga (loading):
    ◦ Indicadores Visuales: Durante las solicitudes a la API, la UI mostrará indicadores de carga (ej., spinners, esqueletos de contenido, o deshabilitación de botones) para comunicar al usuario que la operación está en curso y que el sistema no está "congelado". Esto es especialmente relevante cuando hay "problemas de conectividad" o el "servidor no puede comunicarse".
• Estado de Error (error):
    ◦ Mensajes Clares y Concisos: En caso de fallos en la API (ej., credenciales incorrectas, campos vacíos, o errores de servidor), se mostrarán "mensajes de error claros y concisos" al usuario. Estos mensajes pueden ser notificaciones flotantes, textos debajo de los campos de formulario, o alertas modales.
    ◦ Registro en Logs: Los errores del frontend también podrían registrarse en el "Servicio de Logs de Auditoría" si son críticos, para monitorear problemas de rendimiento o seguridad.
• Estado de Éxito (data):
    ◦ Actualización Reactiva de la UI: Una vez que una llamada a la API es exitosa, los datos recibidos se utilizarán para actualizar el estado global o local pertinente. Esto provocará que los componentes relevantes se re-rendericen automáticamente, mostrando la información actualizada (ej., una nueva entrada en el historial clínico, un nuevo paciente en la tabla, o un cambio de estado en una cita).
Cómo se gestionarán los tokens de autenticación: La seguridad de las credenciales es una "preocupación de seguridad de datos" clave. El proceso de login autentica al usuario y aplica el control de acceso basado en roles. La comunicación se realiza mediante HTTPS (TLS 1.3) para cifrar los datos en tránsito.
• Generación del Token: Tras un login exitoso, el backend emitirá un token de autenticación (probablemente un JSON Web Token o JWT, que es un estándar de la industria). Este token representará la sesión del usuario y sus permisos de rol.
• Almacenamiento Seguro en el Frontend: La forma de almacenar el token es crítica para la seguridad:
    ◦ sessionStorage: Más seguro que localStorage porque los datos se borran cuando la sesión de la pestaña se cierra. Adecuado para sesiones más cortas.
    ◦ localStorage: Permite persistir la sesión entre cierres del navegador, lo que mejora la conveniencia del usuario. Sin embargo, es más vulnerable a ataques XSS (Cross-Site Scripting). Si se usa, es esencial tener estrictas medidas de seguridad XSS y considerar tiempos de expiración cortos para los tokens.
    ◦ Cookies HTTP-Only: Considerado el método más seguro para gestionar sesiones, especialmente si se usa con Secure y SameSite flags. El token se envía automáticamente en cada solicitud al backend sin que el JavaScript del frontend pueda acceder a él directamente, mitigando riesgos de XSS. El backend sería responsable de establecer estas cookies.
• Inclusión en los Headers de las Solicitudes a la API: Para cada solicitud subsiguiente al login que requiera autenticación, el token se incluirá en el encabezado Authorization como un token Bearer (ej., Authorization: Bearer <your_token>). Esto permitirá al backend verificar la identidad y los permisos del usuario antes de procesar la solicitud, garantizando el "control de acceso basado en roles".

8. Implementación de Funcionalidades de UI Complejas:
La interfaz del Dashboard de Escritorio, diseñada bajo la estética "Glassmorphism Oscuro", no es solo visualmente atractiva, sino que también integra lógicas complejas para ofrecer una experiencia de usuario robusta y eficiente. La implementación de estas funcionalidades requiere una atención meticulosa a la interacción entre el frontend (desarrollado con frameworks como React o Vue.js y empaquetado con Electron.js) y el backend (Python/Flask/Django o Node.js/Express en GCP).
• Tabla de Datos Interactiva (ej. Pacientes Recientes, Citas del Día):
    ◦ Estructura y Elementos: La tabla de datos principal muestra información crítica como "ID/Nombre del Paciente", "Fecha de Última Consulta", "Diagnóstico Principal", "Médico Tratante" y un "Checkbox de Selección" por fila. También incluye un "Menú de Acciones" contextual por cada fila con opciones como "Ver Detalles", "Editar Registro", "Ver Historial Clínico Completo" o "Archivar/Eliminar (Suave)". La cabecera incluye un "Checkbox maestro" para selección/deselección de todo.
    ◦ Paginación y Ordenamiento: Aunque no se especifica explícitamente en los detalles de la tabla, para manejar eficientemente la información en un "Núcleo Administrativo

Central", es común que se implemente paginación. El ordenamiento podría realizarse en el frontend para conjuntos de datos pequeños, pero para grandes volúmenes de pacientes o citas, sería más eficiente delegar la lógica de ordenamiento (y filtrado complejo) al backend a través de parámetros en las peticiones API (ej., GET /pacientes?page=1&sort=nombre&order=asc).

- Filtrado: La tabla cuenta con un "Icono de filtro" que despliega un panel con "opciones avanzadas para filtrar la tabla (por fecha, por estado, por tipo de consulta, etc.)". La lógica de filtrado se implementará en el frontend para construir dinámicamente los parámetros de consulta que se enviarán al backend a través de APIs RESTful. El backend procesará estos filtros contra la "Base de Datos Relacional (Cloud SQL)" para devolver solo los datos relevantes.
- Selección Múltiple y Acciones en Lote: Los checkboxes permiten la selección múltiple. La lógica del frontend detectará los registros seleccionados y habilitará botones de acción en lote (ej., "Eliminar" para "borrado suave"). Al activar estas acciones, se enviará una petición API al backend con los IDs de los registros afectados, y el backend ejecutará la lógica de borrado suave (marcando is_deleted = true en la base de datos en lugar de eliminar permanentemente).
- Menú de Acciones por Fila: Este menú contextual se mostrará al hacer clic en un "Icono de tres puntos". La lógica del frontend gestionará la visibilidad y las acciones asociadas a cada ítem del menú (ej., redirigir a una vista de detalle, abrir un formulario de edición).

• Interacción con el Asistente Virtual (Gemma 3):

- Mecanismo de Interacción: El usuario interactúa con el chat interno a través de un "campo de entrada de texto". Al enviar el texto, la aplicación de escritorio (frontend) enviará esta consulta de usuario al Servidor de Aplicación (backend) a través de una API segura.
- Procesamiento y Respuesta: El backend, donde se integra el modelo de IA "Gemma 3" (o similar a Gemini) mediante APIs, procesará la consulta del usuario. La IA generará una respuesta (ej., resúmenes de texto, redacción de informes, búsqueda inteligente de archivos). El backend devolverá esta respuesta al frontend, que la mostrará en tiempo real o casi real en el "área de visualización de conversación" del chat. Es importante recordar que este chat "solo funciona con texto y no genera archivos de ningún formato".
- Automatización de Tareas: El Asistente Virtual también está "conectado a las APIs del sistema para ejecutar acciones como enviar correos, generar borradores de documentos, crear proyectos, etc.". Esto implica que, además de la interacción textual, la IA puede invocar otras funcionalidades del backend basadas en la intención del usuario.

• Implementación del Visor de Archivos Visuales:

- Funcionalidad: Este widget permite visualizar imágenes y videos relacionados con los pacientes. El contenido se actualiza contextualmente según el paciente seleccionado o la búsqueda realizada.
- Carga de Miniaturas y Vista Ampliada: Las miniaturas se cargarán de forma eficiente para el "Carrusel de Vistas Previas". Al seleccionar una miniatura, se cargará la vista ampliada del archivo. Los archivos digitales (PDF, imágenes, videos cortos) se almacenan en "Cloud Storage (S3/Blob) en GCP", y "estarán cifrados en reposo". La recuperación de estos archivos desde Cloud Storage se hará a través de APIs del backend.
- Navegación y Filtros: Las flechas del carrusel (< y >) permitirán la navegación entre archivos. Se implementarán filtros (ej., "Cloud Drives", "Photos", "Videos") para refinar la visualización. La lógica del frontend aplicará estos filtros a la lista de archivos obtenida del backend, o bien el backend los aplicará a nivel de base de datos/almacenamiento para optimizar la carga.

• Integración de Widgets de Servicios Externos (Gmail, Google Drive, Office):

- Nivel de Integración: El Dashboard incluye "widgets para acceso rápido a Gmail y Google Drive, así como accesos directos para abrir aplicaciones de Office". Dada la descripción, la integración inicial se centrará en la conveniencia:
  - Gmail y Google Drive: Los widgets probablemente abrirán las respectivas aplicaciones web en el navegador predeterminado del usuario o en una ventana web integrada si Electron.js lo permite de forma segura. Si se requiere una integración más profunda (ej. listar correos, buscar archivos directamente en el Dashboard), se explorará el uso de las APIs de Google Workspace (como Gmail API, Google Drive API). Esto requeriría una gestión de OAuth 2.0 y tokens de acceso para la autenticación del usuario.
  - Aplicaciones de Office: Los "accesos directos" se interpretan como la capacidad de lanzar las aplicaciones de escritorio de Office (Word, Excel, PowerPoint) instaladas localmente en el sistema operativo del usuario, o abrir documentos de Office en la web si

el usuario lo tiene configurado. No se implica una integración profunda a nivel de contenido o edición dentro del Dashboard.
    ◦ Seguridad: Cualquier integración más allá de un simple enlace deberá asegurar la comunicación a través de HTTPS/TLS 1.3 y manejar las credenciales de forma segura, respetando las políticas de privacidad y los permisos de acceso del usuario.
• Manejo de Formularios (ej. Creación de Perfil de Paciente):
    ◦ Validación del Lado del Cliente: Antes de enviar los datos al backend, el frontend implementará validaciones para asegurar que "campos obligatorios" no estén vacíos (ej., "Campo 'Nombre' requerido") y que el "formato" de los datos sea correcto (ej., "Formato de email inválido"). Esto se realizará mediante lógica JavaScript/TypeScript en el framework de frontend (React/Vue.js).
    ◦ Validación del Lado del Servidor: Aunque se realice una validación inicial en el frontend, el backend siempre replicará y reforzará estas validaciones para garantizar la integridad y seguridad de los datos antes de persistirlos en la base de datos.
    ◦ Retroalimentación al Usuario: Los "mensajes de validación" se mostrarán claramente en la interfaz de usuario, guiando al personal sobre cómo corregir los errores en los datos ingresados.
9. Consideraciones de Rendimiento y Optimización del Frontend:
Dado que el Dashboard de Escritorio es el "Núcleo Administrativo Central", su rendimiento y capacidad de respuesta son fundamentales para la eficiencia operativa del consultorio. Se implementarán diversas estrategias de optimización:
• Optimización de la Carga Inicial:
    ◦ Code Splitting (División de Código): Se dividirá el "bundle" (paquete) de JavaScript, CSS y otros activos en módulos más pequeños. Esto permite que el navegador solo cargue el código necesario para la vista actual, reduciendo el tiempo de carga inicial. Por ejemplo, los módulos de "Administración" o "Reportes", que pueden no ser utilizados por todos los roles o con tanta frecuencia, pueden cargarse de forma diferida.
    ◦ Lazy Loading (Carga Perezosa) de Componentes no Críticos: Los componentes de la interfaz de usuario que no son visibles de inmediato al cargar el Dashboard (ej., modales, secciones ocultas, widgets menos utilizados) se cargarán solo cuando sean necesarios. Esto mejora el tiempo de "Time to Interactive" (TTI), haciendo que la aplicación sea utilizable más rápidamente.
    ◦ Minificación y Compresión: Todos los archivos de código y activos se minificarán (eliminando espacios en blanco y caracteres innecesarios) y se comprimirán (ej., con Gzip o Brotli) antes de su despliegue para reducir el tamaño de la transferencia de datos.
    ◦ Uso Eficiente de CDN (Content Delivery Network): Aunque no se menciona explícitamente, los activos estáticos del frontend (imágenes, CSS, JS) podrían distribuirse a través de una CDN para reducir la latencia de carga para los usuarios, aprovechando los servidores más cercanos geográficamente.
• Optimización del Renderizado de Listas Grandes:
    ◦ Virtualización de Listas/Tablas: Aunque las secciones como "Pacientes Recientes" o "Citas del Día" pueden no tener miles de filas inicialmente, para asegurar la escalabilidad a medida que el consultorio crezca, se considerará la implementación de la virtualización de listas. Esto implica renderizar solo las filas de la tabla que son visibles en la ventana de visualización del usuario, en lugar de renderizar todas las filas a la vez. Esto reduce significativamente el uso de memoria y mejora el rendimiento del scroll para conjuntos de datos extensos.
    ◦ Memoización de Componentes: En frameworks como React o Vue.js, se utilizará la memoización (ej., React.memo o Vue.js keep-alive) para evitar el re-renderizado innecesario de componentes que no han cambiado sus propiedades (props) o estado.
• Manejo Eficiente de Imágenes y Videos:
    ◦ Optimización de Formatos y Compresión: Los archivos de imágenes y videos se almacenarán en "Cloud Storage" en formatos web optimizados (ej., WebP para imágenes, MP4 con codecs eficientes para videos) y se comprimirán adecuadamente para reducir su tamaño sin comprometer excesivamente la calidad.
    ◦ Carga Adaptativa (Responsive Images/Videos): Las imágenes y videos se servirán en diferentes resoluciones o calidades, eligiendo la más apropiada según el tamaño de la pantalla del usuario y la velocidad de su conexión a internet.
    ◦ Streaming y Carga Diferida de Videos: Los videos se cargarán mediante streaming, permitiendo que la reproducción comience antes de que se descargue el archivo completo. Para la funcionalidad del "Visor de Archivos Visuales", los videos solo se cargarán y reproducirán cuando el usuario interactúe con ellos.
    ◦ Caché del Navegador: Se configurarán encabezados de caché HTTP adecuados para los

activos multimedia, permitiendo que el navegador los almacene localmente y los recupere más rápidamente en visitas posteriores.
• Requisito de Conectividad: Es fundamental enfatizar que "se recomienda una conexión a internet estable de al menos 250 Mbps para un rendimiento óptimo del ecosistema", ya que el sistema "depende de la comunicación constante con la infraestructura en la nube". Esta es una consideración clave para la expectativa del usuario sobre el rendimiento.
Estas implementaciones técnicas y estrategias de optimización garantizarán que el Dashboard de Escritorio sea no solo seguro y funcional, sino también rápido, fluido y escalable, proporcionando una experiencia de usuario de alta calidad.

10. Consideraciones Específicas de Electron
El "Núcleo Administrativo Central" o Dashboard de Escritorio del "Ecosistema Digital Inteligente para Cirugía Especial" es una aplicación de escritorio. Para su desarrollo, el frontend se construirá utilizando frameworks web como React o Vue.js, y se empaquetará con Electron.js. Electron.js es una tecnología que permite construir aplicaciones de escritorio multiplataforma (compatibles con Windows y macOS) utilizando tecnologías web como HTML, CSS y JavaScript.
Esto implica varias consideraciones específicas:
• Manejo de funcionalidades específicas de escritorio: Aunque la mayoría de la información crítica y los archivos se almacenarán en Cloud Storage con cifrado en reposo, minimizando la necesidad de acceso a archivos locales, Electron.js proporciona la capacidad de interactuar con el sistema operativo nativo. Esto podría ser útil para funcionalidades futuras, como la gestión de impresoras o la integración con software local si fuera necesario. Sin embargo, el enfoque principal de almacenamiento es la nube para seguridad y centralización.
• Comunicación entre procesos de Electron: Una aplicación Electron se compone de un proceso principal (que maneja la ventana y las interacciones del sistema operativo) y procesos de renderizado (donde se ejecuta la interfaz de usuario web) [No se especifica explícitamente en las fuentes, pero es inherente a Electron]. Una implementación efectiva debe asegurar una comunicación fluida y eficiente entre estos procesos, especialmente cuando la interfaz de usuario necesita invocar funcionalidades del sistema o recibir información de ellas.
• Proceso de empaquetado y distribución: El uso de Electron requiere un proceso para empaquetar la aplicación final en formatos ejecutables para Windows (.exe) y macOS (.dmg) [No se especifica explícitamente en las fuentes, pero es un paso necesario en el desarrollo de Electron]. Esto incluye la configuración de instaladores y el manejo de dependencias.
• Estrategia de actualizaciones de la aplicación de escritorio: Para mantener el sistema seguro y actualizado, es fundamental definir una estrategia clara para la distribución de nuevas versiones y parches [No se especifica explícitamente en las fuentes, pero es una buena práctica para aplicaciones de escritorio]. Electron admite mecanismos de auto-actualización que pueden simplificar este proceso para el usuario final.
La elección de Electron subraya el compromiso con una aplicación de escritorio robusta que aprovecha la agilidad del desarrollo web, facilitando la integración con el backend en la nube.
Siguiente paso: Podríamos explorar las ventajas y desventajas de Electron frente a un desarrollo nativo puro para aplicaciones de escritorio, en términos de costos de desarrollo y rendimiento, para una visión más completa.
11. Diseño Responsivo y Adaptabilidad
El diseño del Dashboard de Escritorio está profundamente anclado en el estilo visual "Glassmorphism Oscuro", que no es solo una cuestión estética, sino funcional, diseñada para ofrecer comodidad visual y una jerarquía de información clara.
Aunque el Dashboard es una aplicación de escritorio y no requiere un "diseño responsivo" en el sentido de adaptarse a pantallas de teléfonos móviles o tabletas (ya que no es su propósito principal), sí debe ser adaptable a diferentes resoluciones de monitor dentro del entorno de escritorio.
Esto se logra mediante la aplicación de los principios del Glassmorphism Oscuro y las buenas prácticas de diseño de interfaz de usuario (UI/UX):
• Coherencia Visual Integral: El diseño del Dashboard mantiene una coherencia visual con la pantalla de Login y el resto del ecosistema, asegurando una experiencia de usuario fluida y de alta calidad desde el primer punto de interacción.
• Comodidad Visual y Legibilidad Óptima: El fondo principal de gris carbón suave (#121212) está diseñado para reducir la fatiga visual, evitando el negro puro. El texto y los iconos

son nítidos y brillantes, utilizando el color principal de texto blanco roto (#F5F5F5) para una alta legibilidad sobre el vidrio oscuro, cumpliendo con una relación de contraste superior a 4.5:1 (WCAG). Esta legibilidad es crucial para mantener la usabilidad en diversas configuraciones de pantalla de escritorio.
• Jerarquía de Información Clara: Los paneles flotantes de "vidrio esmerilado" y el enfoque multicapa (eje Z) crean una sensación de profundidad que ayuda a organizar y jerarquizar la información de manera intuitiva. Esta clara separación de elementos ayuda a que la interfaz se escale visualmente sin perder su estructura o la importancia de la información.
• Uso de CSS Variables: La implementación utilizará Propiedades Personalizadas de CSS para el sistema de color y tipografía. Esto facilita la mantenibilidad y asegura una coherencia visual integral, lo que indirectamente apoya la adaptabilidad, ya que los cambios en el tamaño o la disposición de los elementos pueden ajustarse de manera centralizada.
• Estrategias Implícitas para Layout: Aunque no se especifican directamente técnicas como flexbox o grid para el layout en las fuentes, el concepto de paneles flotantes y la necesidad de una "jerarquía espacial clara" sugieren que se emplearán principios de diseño de interfaz de usuario que permitan que los elementos se distribuyan y redimensionen de manera armoniosa en el espacio disponible, manteniendo el confort visual y la funcionalidad en diferentes tamaños de ventana de escritorio.
En resumen, la "adaptabilidad" del Dashboard se centra en mantener una experiencia de usuario consistente, cómoda y legible a través de las variaciones de tamaño de pantalla típicas de un entorno de escritorio, utilizando las propiedades inherentes del diseño Glassmorphism Oscuro y las buenas prácticas de desarrollo de UI.
Especificación de la Integración de Inteligencia Artificial en el Dashboard de Administración
1. Introducción y Propósito
Este documento tiene como objetivo principal detallar la integración técnica y funcional del modelo de Inteligencia Artificial (IA) en el "Núcleo Administrativo Central" o Dashboard de Escritorio del "Ecosistema Digital Inteligente para Cirugía Especial". Su propósito fundamental es ofrecer una guía exhaustiva para los equipos de diseño y desarrollo, asegurando una implementación coherente y de alta calidad de las funcionalidades impulsadas por IA.
La integración de la inteligencia artificial busca potenciar el flujo de trabajo, la automatización de tareas repetitivas, y ofrecer asistencia rápida al personal. Esto se traduce en una optimización significativa de la "eficiencia operativa", reduciendo los procesos manuales y el riesgo de errores, lo que a su vez libera tiempo valioso para el equipo del consultorio. La IA centraliza la inteligencia del sistema para asegurar que la lógica de negocio se ejecute de manera segura y escalable.
Esta funcionalidad clave, denominada "Inteligencia y Automatización Integrada", es un componente fundamental desarrollado en la Fase 1: El Núcleo Administrativo (Dashboard de Escritorio) del proyecto.
2. Modelo de Inteligencia Artificial
El modelo de Inteligencia Artificial a utilizar en este ecosistema es Gemma 3 o uno similar a Gemini. Este modelo se integrará directamente en el Servidor de Aplicación (Backend) mediante el uso de APIs (Interfaces de Programación de Aplicaciones). Esto significa que el backend actuará como intermediario entre la interfaz de usuario del Dashboard y el modelo de IA.
Las capacidades relevantes del modelo Gemma 3 (o similar) para este proyecto incluyen, pero no se limitan a:
• Generación de Texto y Automatizaciones Básicas: El modelo tiene la capacidad de generar texto coherente y relevante para diversas tareas, permitiendo la redacción automatizada de correos electrónicos, la generación de hojas membretadas y la creación de recordatorios. Estas funciones se activan desde módulos específicos de la aplicación o pueden ser invocadas a través del chat interno.
• Resumen y Comprensión de Lenguaje Natural (Chat Interno Básico): El modelo potenciará un chat interno dentro del Dashboard donde el personal puede interactuar con la IA. A través de este chat, los usuarios podrán realizar consultas rápidas, obtener resúmenes de texto complejos o facilitar la redacción de informes. Es importante señalar que este chat funciona únicamente con texto y no tiene la capacidad de generar archivos de ningún formato.
• Asistencia para el Desarrollo y Capacitación: La IA también puede ser utilizada para generar texto de ejemplo (denominado "lorem ipsum más inteligente") para prototipos de interfaz de usuario (UI). Asimismo, puede sugerir "copys" para botones y etiquetas si se

le entrena con el tono deseado. Desde una perspectiva de desarrollo, la IA puede generar scripts SQL básicos a partir de descripciones en lenguaje natural. Para la capacitación del personal, puede colaborar en la generación de preguntas frecuentes (FAQs) a partir de la documentación técnica para el manual de usuario, e incluso crear un chatbot de ayuda inicial.
• Procesamiento de Solicitudes y Asistencia Rápida: De manera general, el modelo de IA está diseñado para procesar solicitudes en lenguaje natural y proporcionar asistencia rápida, optimizando el flujo de trabajo y la eficiencia general del sistema.
La elección de Gemma 3 (o similar) subraya el compromiso del proyecto con la innovación y la automatización para resolver el "dolor" de los "procesos manuales que consumen tiempo".

Especificación de la Integración de Inteligencia Artificial en el Dashboard de Administración
3. Casos de Uso de IA en el Dashboard (Funcionalidades Detalladas):
La integración del modelo de Inteligencia Artificial Gemma 3 (o similar a Gemini) en el backend del Dashboard de Administración es un pilar fundamental para potenciar la "Inteligencia y Automatización Integrada", resolviendo el "dolor" de los "procesos manuales que consumen tiempo" y optimizando la "eficiencia operativa". Las funcionalidades que hacen uso de la IA se activarán de manera intuitiva desde módulos específicos de la aplicación o mediante el chat interno.
A continuación, se describen en detalle los casos de uso específicos de la IA:
3.1. Generación de Borradores de Correo Electrónico:
• Propósito: Esta funcionalidad tiene como objetivo principal acelerar y estandarizar la comunicación escrita con los pacientes o terceros, permitiendo al personal del consultorio redactar correos electrónicos de manera rápida y eficiente. Al automatizar la generación de borradores, se reduce el tiempo dedicado a tareas repetitivas y se minimiza el riesgo de errores u omisiones en mensajes importantes.
• Entrada para la IA: La IA recibirá información contextual directamente del sistema, así como datos específicos proporcionados por el usuario para generar un borrador relevante:
    ◦ Datos Contextuales del Sistema: El usuario seleccionará un paciente o una cita desde el Dashboard. El sistema proporcionará automáticamente a la IA el ID del Paciente, el nombre completo del paciente, datos de contacto (ej., email), fechas y detalles de la última consulta o próxima cita y diagnósticos relevantes asociados al expediente.
    ◦ Entrada de Usuario para la Personalización: El personal deberá especificar a la IA:
        ▪ Motivo del Correo: (ej., recordatorio de cita, seguimiento post-operatorio, notificación de resultados de laboratorio, solicitud de información adicional, confirmación de cita).
        ▪ Puntos Clave a Incluir: Información adicional específica que el usuario desea que se incorpore en el cuerpo del correo (ej., instrucciones pre/post-operatorias, medicamentos, fechas de próxima revisión, requisitos de preparación para estudios).
        ▪ Tono Deseado: (ej., formal, empático, conciso).
• Salida Esperada de la IA: La IA generará un borrador de texto para un correo electrónico, estructurado y coherente, que contendrá:
    ◦ Un asunto sugerido para el correo.
    ◦ Un saludo personalizado dirigido al paciente (o destinatario si no es un paciente).
    ◦ El cuerpo del mensaje que integre el motivo, los puntos clave proporcionados y la información contextual del paciente.
    ◦ Un cierre profesional con la información del consultorio y del Dr. Sánchez.
    ◦ Es importante recalcar que la IA generará únicamente el texto del borrador y no producirá un archivo de correo electrónico final ni lo enviará directamente. El borrador de texto será presentado en la interfaz del Dashboard para que el usuario pueda revisarlo, editarlo y luego copiarlo o integrarlo en su cliente de correo electrónico (ej., Gmail, accesible vía widget).
• Cómo se Iniciará esta Función en la Interfaz:
    ◦ Desde un Módulo Específico: Se habilitará un botón o una opción de "Generar Borrador de Correo" en la vista de detalle de cada expediente del paciente, en el módulo de Agenda y Citas (al seleccionar una cita) o en una sección dedicada a "Automatizaciones".
    ◦ Vía Chat Interno Básico: El personal podrá iniciar la solicitud escribiendo directamente en el chat interno del Dashboard comandos o preguntas en lenguaje natural, como: "Genera un correo de recordatorio para Laura Vargas para su cita de mañana" o "Redacta un correo de seguimiento post-cirugía para Juan Pérez, incluyendo instrucciones de medicación". El asistente virtual integrado con Gemma 3 será capaz de procesar estas solicitudes y generar el borrador.

3.2. Generación de Plantillas para Hojas Membretadas:
• Propósito: Esta funcionalidad está diseñada para simplificar la creación de documentos oficiales del consultorio, como certificados médicos, cartas de referencia o plantillas de prescripción, asegurando que mantengan un formato coherente con la identidad de la marca (Dr. Sánchez García) y la información clave necesaria, reduciendo el esfuerzo manual y los tiempos de redacción.
• Entrada para la IA: Para generar el contenido de la plantilla, la IA requerirá:
    ◦ Datos Contextuales del Sistema: Similar a la generación de correos, el sistema podrá proveer información del paciente (ID, nombre, apellido, diagnósticos), así como datos del médico (nombre del Dr. Sánchez García) y datos del consultorio (dirección, teléfono, logo, información de contacto) que ya estén configurados en el sistema.
    ◦ Entrada de Usuario para la Personalización: El usuario definirá:
        ▪ Tipo de Documento: (ej., Certificado Médico, Carta de Referencia, Prescripción, Informe Médico).
        ▪ Contenido Principal: Breve descripción o puntos clave que deben incluirse en el cuerpo del documento.
        ▪ Instrucciones Específicas: Cualquier requisito adicional o formato particular (ej., "Incluir el diagnóstico exacto de [Diagnóstico]", "Especificar la fecha de la última consulta", "Añadir detalles sobre el procedimiento X").
        ▪ Tono y Estilo: Si el usuario desea un lenguaje más formal, técnico o explicativo para el documento.
• Salida Esperada de la IA: La IA producirá el texto formateado del contenido principal de la hoja membretada. Este texto incluirá:
    ◦ Los campos dinámicos (ej., nombre del paciente, fecha, diagnóstico) que el sistema podrá rellenar automáticamente.
    ◦ El cuerpo principal del documento basado en el tipo y el contenido especificado por el usuario.
    ◦ Un espacio para la firma y los datos del médico.
    ◦ Al igual que con los correos, la IA generará únicamente el contenido de texto y no un archivo de hoja membretada con el diseño gráfico (membrete, logo, etc.). El texto generado se presentará en la interfaz para que el usuario pueda copiarlo y pegarlo en una plantilla de documento predefinida (ej., en Word, accesible vía widget de Office).
• Cómo se Iniciará esta Función en la Interfaz:
    ◦ Desde un Módulo Específico: Se dispondrá de un botón o una opción de "Generar Documento / Plantilla" dentro de la vista de detalle del expediente del paciente, el Historial Clínico Detallado o en la sección general de "Documentos" / "Automatizaciones" en el Dashboard.
    ◦ Vía Chat Interno Básico: El personal podrá solicitar la generación de plantillas a través del chat interno, por ejemplo: "Genera una plantilla de certificado médico para María López con diagnóstico de amigdalitis" o "Prepara un borrador de carta de referencia para el especialista en traumatología para el paciente Roberto Salas". El asistente virtual integrado con Gemma 3 facilitará la delegación de esta tarea.

Especificación de la Integración de Inteligencia Artificial en el Dashboard de Administración
3. Casos de Uso de IA en el Dashboard (Funcionalidades Detalladas):
La integración del modelo de Inteligencia Artificial Gemma 3 (o similar a Gemini) en el backend del Dashboard de Administración es un pilar fundamental para potenciar la "Inteligencia y Automatización Integrada", resolviendo el "dolor" de los "procesos manuales que consumen tiempo" y optimizando la "eficiencia operativa". Las funcionalidades que hacen uso de la IA se activarán de manera intuitiva desde módulos específicos de la aplicación o mediante el chat interno.
A continuación, se describen en detalle los casos de uso específicos de la IA:
3.1. Generación de Borradores de Correo Electrónico:
• Propósito: Esta funcionalidad tiene como objetivo principal acelerar y estandarizar la comunicación escrita con los pacientes o terceros, permitiendo al personal del consultorio redactar correos electrónicos de manera rápida y eficiente [Previous turn]. Al automatizar la generación de borradores, se reduce el tiempo dedicado a tareas repetitivas y se minimiza el riesgo de errores u omisiones en mensajes importantes [Previous turn, 22, 93, 94].
• Entrada para la IA: La IA recibirá información contextual directamente del sistema, así como datos específicos proporcionados por el usuario para generar un borrador relevante [Previous turn]:

    ◦ Datos Contextuales del Sistema: El usuario seleccionará un paciente o una cita desde el Dashboard. El sistema proporcionará automáticamente a la IA el ID del Paciente, el nombre completo del paciente, datos de contacto (ej., email), fechas y detalles de la última consulta o próxima cita y diagnósticos relevantes asociados al expediente [Previous turn].
    ◦ Entrada de Usuario para la Personalización: El personal deberá especificar a la IA:
        ▪ Motivo del Correo: (ej., recordatorio de cita, seguimiento post-operatorio, notificación de resultados de laboratorio, solicitud de información adicional, confirmación de cita) [Previous turn].
        ▪ Puntos Clave a Incluir: Información adicional específica que el usuario desea que se incorpore en el cuerpo del correo (ej., instrucciones pre/post-operatorias, medicamentos, fechas de próxima revisión, requisitos de preparación para estudios) [Previous turn].
        ▪ Tono Deseado: (ej., formal, empático, conciso) [Previous turn].
• Salida Esperada de la IA: La IA generará un borrador de texto para un correo electrónico, estructurado y coherente, que contendrá un asunto sugerido, un saludo personalizado (al paciente o destinatario), el cuerpo del mensaje que integre el motivo, los puntos clave y la información contextual, y un cierre profesional con la información del consultorio y del Dr. Sánchez [Previous turn]. Es importante recalcar que la IA generará únicamente el texto del borrador y no producirá un archivo de correo electrónico final ni lo enviará directamente [Previous turn]. El borrador de texto será presentado en la interfaz del Dashboard para que el usuario pueda revisarlo, editarlo y luego copiarlo o integrarlo en su cliente de correo electrónico (ej., Gmail, accesible vía widget) [Previous turn, 37, 51, 154, 188, 189, 254, 266].
• Cómo se Iniciará esta Función en la Interfaz: Esta función es una de las "Automatizaciones Básicas" que ofrece la integración de IA.
    ◦ Desde Módulos Específicos: Se habilitará un botón o una opción explícita de "Generar Borrador de Correo" en las siguientes ubicaciones:
        ▪ La vista de detalle de cada expediente del paciente [Previous turn].
        ▪ El módulo de Agenda y Citas, al seleccionar una cita específica [Previous turn].
        ▪ Una sección dedicada a "Automatizaciones" o "Inteligencia y Automatización Integrada" en el Dashboard [Previous turn, 39, 90, 159, 255, 271].
    ◦ Vía Chat Interno Básico: El personal podrá iniciar la solicitud escribiendo directamente en el chat interno del Dashboard comandos o preguntas en lenguaje natural [Previous turn, 159]. Ejemplos incluyen: "Genera un correo de recordatorio para Laura Vargas para su cita de mañana" o "Redacta un correo de seguimiento post-cirugía para Juan Pérez, incluyendo instrucciones de medicación" [Previous turn]. El asistente virtual integrado con Gemma 3 será capaz de procesar estas solicitudes y generar el borrador [Previous turn, 93].

3.2. Generación de Plantillas para Hojas Membretadas:

• Propósito: Esta funcionalidad está diseñada para simplificar la creación de documentos oficiales del consultorio, como certificados médicos, cartas de referencia o plantillas de prescripción, asegurando que mantengan un formato coherente con la identidad de la marca (Dr. Sánchez García) y la información clave necesaria, reduciendo el esfuerzo manual y los tiempos de redacción [Previous turn, 22, 93, 94].
• Entrada para la IA: Para generar el contenido de la plantilla, la IA requerirá [Previous turn]:
    ◦ Datos Contextuales del Sistema: El sistema podrá proveer información del paciente (ID, nombre, apellido, diagnósticos), así como datos del médico (nombre del Dr. Sánchez García) y datos del consultorio (dirección, teléfono, logo, información de contacto) que ya estén configurados en el sistema [Previous turn].
    ◦ Entrada de Usuario para la Personalización: El usuario definirá:
        ▪ Tipo de Documento: (ej., Certificado Médico, Carta de Referencia, Prescripción, Informe Médico) [Previous turn].
        ▪ Contenido Principal: Breve descripción o puntos clave que deben incluirse en el cuerpo del documento [Previous turn].
        ▪ Instrucciones Específicas: Cualquier requisito adicional o formato particular (ej., "Incluir el diagnóstico exacto de [Diagnóstico]", "Especificar la fecha de la última consulta", "Añadir detalles sobre el procedimiento X") [Previous turn].
        ▪ Tono y Estilo: Si el usuario desea un lenguaje más formal, técnico o explicativo para el documento [Previous turn].
• Salida Esperada de la IA: La IA producirá el texto formateado del contenido principal de la hoja membretada [Previous turn]. Este texto incluirá los campos dinámicos (ej., nombre

del paciente, fecha, diagnóstico) que el sistema podrá rellenar automáticamente, el cuerpo principal del documento basado en el tipo y el contenido especificado por el usuario, y un espacio para la firma y los datos del médico [Previous turn]. Al igual que con los correos, la IA generará únicamente el contenido de texto y no un archivo de hoja membretada con el diseño gráfico (membrete, logo, etc.) [Previous turn]. El texto generado se presentará en la interfaz para que el usuario pueda copiarlo y pegarlo en una plantilla de documento predefinida (ej., en Word, accesible vía widget de Office) [Previous turn, 37, 51, 154, 188, 189, 254, 266].

- Cómo se Iniciará esta Función en la Interfaz: Esta función es parte de las "Automatizaciones Básicas" que la IA puede realizar.
  - Desde Módulos Específicos: Se dispondrá de un botón o una opción de "Generar Documento / Plantilla" dentro de las siguientes vistas:
    - La vista de detalle del expediente del paciente [Previous turn].
    - El Historial Clínico Detallado del paciente [Previous turn].
    - En la sección general de "Documentos" o "Automatizaciones" en el Dashboard [Previous turn, 39, 90, 159, 255, 271].
  - Vía Chat Interno Básico: El personal podrá solicitar la generación de plantillas a través del chat interno, por ejemplo: "Genera una plantilla de certificado médico para María López con diagnóstico de amigdalitis" o "Prepara un borrador de carta de referencia para el especialista en traumatología para el paciente Roberto Salas" [Previous turn]. El asistente virtual integrado con Gemma 3 facilitará la delegación de esta tarea [Previous turn, 93].

3.3. Generación de Recordatorios Automatizados:

- Propósito: Esta funcionalidad tiene como propósito principal liberar tiempo valioso para el personal del consultorio al automatizar la creación y gestión de recordatorios. Ayuda a garantizar que tanto el personal como los pacientes estén informados sobre citas, eventos o tareas importantes, reduciendo los "procesos manuales que consumen tiempo" y mejorando la "eficiencia operativa". Esto incluye la capacidad de enviar "Notificaciones y Recordatorios Automatizados" a los pacientes, especialmente a través de la integración con WhatsApp.
- Entrada para la IA: La IA y el sistema consumirán datos relevantes para la generación de recordatorios:
  - Datos de Citas: Información del Módulo de Agenda y Citas, incluyendo fecha, hora, tipo de cita y estado (Confirmada, Programada, etc.).
  - Datos del Paciente: Nombre y datos de contacto (ej., número de teléfono para WhatsApp) asociados a la cita.
  - Eventos o Tareas: Datos sobre tareas internas del personal (ej., "revisar expedientes mañana", "preparar materiales para cirugía") o eventos específicos que requieran un recordatorio. Si bien no se detalla un módulo de "tareas" para el personal en el Dashboard, la capacidad de la IA para generar texto y automatizar sugiere que podría procesar estos datos si se ingresan.
- Posible Rol de la IA: El modelo Gemma 3 se integrará en el backend mediante APIs para potenciar estas automatizaciones. El rol de la IA en la generación de recordatorios podría incluir:
  - Generación de Contenido Sugerido: Interpretar los datos de la cita o tarea para sugerir el texto más apropiado para el recordatorio (ej., "Su cita para el 15 de julio a las 10:00 AM ha sido confirmada").
  - Personalización del Tono: Ajustar el tono del recordatorio (ej., más formal para una cita inicial, más conciso para un recordatorio rápido de tarea).
  - Sugerencia de Momentos Clave: Si se le entrena con patrones, la IA podría sugerir el momento óptimo para enviar un recordatorio (ej., 24 horas antes de la cita, 1 hora antes de la cita, al finalizar el día para tareas pendientes).
- Cómo se Configurarán estos Recordatorios:
  - Los recordatorios son parte de las "Automatizaciones Básicas" ofrecidas por la IA.
  - Se configurarán a través de módulos específicos del Dashboard (como el "Módulo de Agenda y Citas") o, en algunos casos, pueden ser invocados a través del chat interno.
  - Para los recordatorios de pacientes vía WhatsApp, el sistema actuará de forma automática basándose en la información del calendario de citas. Esto implica que las plantillas de recordatorio (posiblemente generadas con asistencia de IA) y las reglas de envío se establecerán previamente en el sistema.

3.4. Chat Interno con el Modelo de IA:

- Propósito: El chat interno básico tiene como propósito principal permitir al personal del consultorio interactuar directamente con el modelo de IA (Gemma 3 o similar a Gemini)

para obtener asistencia rápida en tareas de texto y consultas. Su objetivo es mejorar la eficiencia operativa al ofrecer una herramienta de consulta inmediata y automatización de procesos de texto repetitivos.
• Funcionalidades Específicas del Chat: El chat interno, potenciado por la IA, ofrecerá las siguientes capacidades:
    ◦ Resúmenes de Texto:
        ▪ Entrada: El usuario proporcionará texto para resumir (ej., notas de evolución del historial clínico, una serie de interacciones con el paciente, o un informe largo) [Previous turn, 23, 38, 93].
        ▪ Salida: La IA generará un resumen conciso y coherente del texto proporcionado [Previous turn, 23, 38, 93].
    ◦ Redacción de Informes Cortos:
        ▪ Entrada: El usuario proporcionará puntos clave, datos específicos, o una descripción breve del propósito del informe [Previous turn].
        ▪ Salida: La IA redactará un borrador de texto para un informe corto, estructurando la información y aplicando un tono adecuado [Previous turn, 23, 38, 93].
    ◦ Respuestas a Consultas Rápidas:
        ▪ El personal podrá realizar preguntas directas o comandos en lenguaje natural (ej., "Encuéntrame los últimos estudios de Laura Vargas", "Qué paciente tiene cita a las 11 AM?", "Resúmeme la última consulta de Juan Pérez") [Previous turn, 159, 283].
        ▪ Acceso y Uso de Datos Relevantes: La IA accedería o utilizaría datos relevantes a través de su integración con el backend del sistema. El backend, construido con Python/Flask/Django o Node.js/Express, se comunicará con la Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL) donde se almacena toda la información estructurada del consultorio (perfiles de pacientes, historiales clínicos, citas, usuarios y roles). También accederá al Almacenamiento de Archivos (Cloud Storage) para documentos digitales. De esta manera, la IA podrá localizar recursos y ofrecer acciones contextuales (ej., "Lo encontré", "te lo envío a tu correo o prefieres descargarlo?").
• Descripción de la Interfaz de Usuario del Chat:
    ◦ Ubicación: El chat se accederá a través de un icono o una sección dedicada, visible dentro de la interfaz del Dashboard de Escritorio.
    ◦ Área de Entrada: Contará con un campo de entrada de texto claro y conciso donde el usuario escribirá sus consultas o comandos.
    ◦ Área de Visualización: Encima del campo de entrada, habrá un área de visualización donde se mostrará el historial de la conversación, con las preguntas del usuario y las respuestas o el contenido generado por la IA.
    ◦ Botón de Envío: Un botón (ej., "Enviar") para enviar la consulta a la IA.
    ◦ Formato de Salida: Es fundamental recalcar que este chat funciona únicamente con texto y no tiene la capacidad de generar archivos de ningún formato (ej., PDFs, imágenes).
3.5. Otros Casos de Uso de Automatización Básicos Mencionados:
Además de las funcionalidades de redacción de correos, generación de hojas membretadas, recordatorios y el chat interno básico, la integración de la IA Gemma 3 extiende sus capacidades a otras áreas de automatización y asistencia, principalmente en el ámbito de texto y soporte:
• Asistencia para el Desarrollo y Capacitación: La IA puede ser una herramienta valiosa para el equipo de desarrollo y para la capacitación del personal.
    ◦ Generación de Texto de Ejemplo (Lorem Ipsum Inteligente): Puede generar texto de relleno relevante para prototipos de interfaz de usuario (UI), agilizando el proceso de diseño.
    ◦ Sugerencia de "Copys": Si se le entrena con el tono deseado, la IA puede sugerir textos para botones y etiquetas de la interfaz.
    ◦ Generación de Scripts SQL Básicos: A partir de descripciones en lenguaje natural, la IA puede generar scripts SQL fundamentales para la base de datos.
    ◦ Generación de Preguntas Frecuentes (FAQs): Puede colaborar en la creación de FAQs a partir de la documentación técnica, que serán útiles para el manual de usuario y la capacitación inicial.
    ◦ Creación de un Chatbot de Ayuda Inicial para Capacitación: La IA puede sentar las bases para un chatbot que asista al personal durante su capacitación, respondiendo preguntas básicas sobre el uso del sistema.
• Asistente Virtual (Bot) en Página Web y WhatsApp: Aunque son parte de fases posteriores (Fase 2 y 3), la IA Gemma 3 es fundamental para potenciar los asistentes virtuales externos.
    ◦ Respuestas a Preguntas Frecuentes (FAQ): Los bots en la página web y WhatsApp,

impulsados por IA, podrán responder a las preguntas más comunes de los pacientes.
    ◦ Agendamiento de Citas: La IA facilitará el proceso de agendamiento de citas a través de los bots, sincronizándose con el calendario del Dashboard.
    ◦ Delegación de Gestión Inicial: El Asistente Virtual Inteligente (AVI) delegará la gestión inicial de consultas y agendamiento a la IA, liberando tiempo valioso del personal del consultorio.

3.6. Límites de las Capacidades de la IA en esta Fase Específica del Proyecto:

Es crucial establecer los límites claros de la inteligencia artificial en esta fase inicial del proyecto para gestionar expectativas y garantizar un uso responsable y seguro de la tecnología, especialmente con datos sensibles de pacientes. La IA se enfoca en la "Inteligencia y Automatización Integrada" para optimizar la "eficiencia operativa".

• Enfoque en Texto y Generación de Borradores: La IA está diseñada principalmente para procesar y generar texto.
    ◦ Produce borradores y sugerencias de texto (correos, plantillas, informes, resúmenes, FAQs) [Previous turn, 23, 38, 93, 115, 159].
    ◦ No generará archivos finales (ej., PDFs con diseño gráfico, documentos de Word completos, ni enviará correos directamente) [Previous turn, 23, 38, 93, 115, 159]. La interacción final y el envío recaen en el usuario.
• No Generación de Contenido Multimedia: La IA no tiene la capacidad de crear ni manipular imágenes, videos u otros tipos de archivos multimedia [No source mentions this capability for AI]. La "Gestión Documental Avanzada" se encarga de subir, asociar y visualizar estos archivos, que son provistos externamente o por el sistema.
• No Toma de Decisiones Médicas o Legales: La IA es una herramienta de asistencia y automatización. No está diseñada para interpretar datos médicos complejos con fines de diagnóstico, recomendación de tratamientos, toma de decisiones clínicas ni para ofrecer asesoramiento legal o ético [No source mentions this capability for AI, and it contradicts the "asistencia" purpose]. Su rol es de apoyo al personal, procesando información existente o generando texto basado en entradas definidas.
• No Autonomía en Acciones Críticas: Si bien puede automatizar tareas, estas automatizaciones están dentro de los flujos de negocio predefinidos y, en muchos casos, requieren una revisión y confirmación humana (como en la generación de borradores de correos) [Previous turn]. La IA se integra en el backend del sistema, lo que significa que opera dentro de la lógica de negocio y las APIs establecidas, no de forma autónoma fuera de los parámetros del sistema.
• Dependencia de la Calidad de la Entrada: La calidad de la salida de la IA dependerá directamente de la claridad y completitud de la información de entrada proporcionada por el usuario o el sistema. No "adivinará" información ausente.

En resumen, la IA en esta fase es un potente asistente de texto y automatización de procesos administrativos, diseñado para aumentar la eficiencia y liberar tiempo del personal, siempre bajo la supervisión y control del usuario.

Interacción Técnica con el Modelo de IA (Gemma 3)

La integración del modelo de Inteligencia Artificial Gemma 3 (o similar a Gemini) es un componente fundamental del "Ecosistema Digital Inteligente para Cirugía Especial". Esta inteligencia artificial potencia funcionalidades clave como la automatización de tareas repetitivas y la asistencia rápida al personal del consultorio.

Método de Integración

El modelo Gemma 3 se integra directamente en el Servidor de Aplicación (Backend) del ecosistema mediante el uso de APIs (Interfaces de Programación de Aplicaciones). Esto significa que el backend actúa como el intermediario entre la interfaz de usuario (el Dashboard de Escritorio o las aplicaciones móviles/web) y el modelo de IA.

El backend se construirá utilizando lenguajes y frameworks robustos como Python (Flask/Django) o Node.js (Express). El despliegue de este backend se realiza en Google Cloud Platform (GCP), utilizando servicios como Compute Engine o Cloud Run.

En cuanto a la forma específica de acceso al modelo Gemma 3, las fuentes indican que puede ser a través de dos métodos principales:

• Acceso mediante API pública (ej. Google AI API para Gemini): Si Gemma 3 se ofrece como un servicio gestionado a través de una API pública, el backend se comunicará con ella utilizando endpoints HTTP específicos. Esto implicaría el envío de solicitudes (probablemente POST) a URLs definidas por el proveedor de la IA, con los prompts y datos necesarios en el cuerpo de la solicitud. La comunicación se realizará siempre a través de HTTPS (TLS 1.3) para cifrar los datos en tránsito, protegiendo tanto las credenciales del sistema como cualquier información sensible que pudiera acompañar a los prompts.

• Despliegue del Modelo en Google Cloud (Vertex AI): Otra opción, sugerida en las fuentes, es que el modelo Gemma 3 se despliegue de forma autónoma en Vertex AI, el servicio de aprendizaje automático de GCP. En este caso, la interacción del backend con el modelo se realizaría utilizando los SDKs o APIs de Vertex AI, lo que podría ofrecer una integración más profunda y optimizada dentro del ecosistema de GCP, manteniendo la comunicación dentro de la Virtual Private Cloud (VPC) para mayor seguridad.

Especificación de Entradas (Prompts)

Las entradas al modelo de IA (prompts) serán principalmente de texto, ya que el chat interno básico de la IA funciona únicamente con texto y no genera archivos de ningún formato. La especificación y estructura de estos prompts variará según el caso de uso:

1. Automatizaciones Básicas (Redacción de Correos, Hojas Membretadas, Recordatorios):
    ◦ Formato del Prompt: Se espera que el prompt incluya instrucciones claras y concisas sobre el tipo de documento o mensaje a generar, el tono deseado, y los datos específicos que deben insertarse. Por ejemplo, para un recordatorio de cita, el prompt podría incluir el nombre del paciente, la fecha y hora de la cita, el tipo de consulta y cualquier instrucción especial.
    ◦ Contexto: Los datos de contexto (ej., nombre del paciente, detalles de la cita) serían extraídos de la Base de Datos Relacional (Cloud SQL) por el backend antes de ser formateados en el prompt para la IA.
2. Chat Interno Básico (Consultas Rápidas, Resúmenes de Texto, Redacción de Informes):
    ◦ Formato del Prompt: Para consultas rápidas o solicitudes de resumen, el prompt sería el texto directo de la consulta o el texto a resumir. Para la redacción de informes, el prompt incluiría las directrices para el informe y los datos relevantes del paciente o la consulta.
    ◦ Manejo del Contexto en Conversaciones: Aunque no se detalla explícitamente el mecanismo, un chat interno requiere que el backend mantenga y envíe el contexto de la conversación previa al modelo de IA. Esto podría implicar el envío de las últimas 'N' interacciones (pares pregunta-respuesta) o un resumen condensado del diálogo anterior junto con la nueva consulta del usuario, permitiendo que la IA mantenga la coherencia y relevancia en sus respuestas.
3. Asistencia para el Desarrollo y Capacitación (Generación de Texto de Ejemplo, "Copys", Scripts SQL, FAQs):
    ◦ Formato del Prompt:
        ▪ Para texto de ejemplo o "copys" de UI: El prompt incluiría el propósito del texto, la interfaz o elemento donde se usará, y el tono deseado.
        ▪ Para scripts SQL básicos: El prompt sería una descripción en lenguaje natural de la operación de base de datos deseada (ej., "crea una tabla para pacientes con nombre, apellido y fecha de nacimiento").
        ▪ Para FAQs: El prompt podría consistir en segmentos de la documentación técnica o consultas comunes de usuarios, pidiéndole a la IA que genere preguntas y respuestas concisas.

Especificación de Salidas (Respuestas)

Las respuestas del modelo de IA, como se mencionó, serán principalmente de texto. El formato exacto y la estructura esperada del modelo de IA para cada caso de uso son:

1. Automatizaciones Básicas:
    ◦ Formato: Texto plano o texto formateado (ej., con saltos de línea, listas) que representa el borrador completo del correo electrónico, la hoja membretada o el recordatorio.
    ◦ Ejemplo: Un borrador de correo con campos como Asunto, Cuerpo del mensaje, Saludo y Firma ya rellenos por la IA.
2. Chat Interno Básico:
    ◦ Formato: Texto plano que constituye la respuesta directa a la consulta del usuario, el resumen solicitado, o el borrador del informe.
    ◦ Restricción: Es crucial que la IA no genere archivos de ningún formato como salida de este chat.
3. Asistencia para el Desarrollo y Capacitación:
    ◦ Formato:
        ▪ Texto de ejemplo o "copys": Texto plano o frases cortas para usar en la interfaz.
        ▪ Scripts SQL: Texto que contiene las sentencias SQL generadas.
        ▪ FAQs: Pares de preguntas y respuestas en formato de texto estructurado (posiblemente un JSON si la IA se entrena para ello, aunque no especificado directamente, sería una buena práctica para facilitar su consumo por el backend).

Manejo de Errores y Respuestas Inesperadas de la IA
La robustez del sistema depende de cómo el backend detecta y gestiona los errores o respuestas de la IA que no se ajusten al formato esperado o sean irrelevantes. La integración de la IA debe realizarse de forma segura, garantizando la privacidad de los datos sensibles de los pacientes.
1. Detección de Errores y Anomalías:
    ◦ Errores de Conectividad/Red: Fallos al establecer comunicación con la API de la IA (ej., timeouts, errores de DNS).
    ◦ Respuestas Malformadas: La IA podría devolver una respuesta que no sigue el formato de texto esperado, o si se esperara JSON, un JSON inválido o incompleto.
    ◦ Respuestas Incoherentes/Irrelevantes (Alucinaciones): La IA podría generar texto que, aunque gramaticalmente correcto, es ilógico o no responde a la solicitud del prompt, especialmente si el contexto es ambiguo o los datos de entrada son insuficientes.
    ◦ Limitaciones de Tasa (Rate Limiting): Si se utiliza una API pública, se podrían alcanzar los límites de llamadas por minuto/segundo, resultando en errores HTTP 429.
    ◦ Errores del Modelo Interno: Problemas en el lado del proveedor de la IA o del despliegue en Vertex AI que impidan una respuesta exitosa (errores 5xx del servicio de la IA).
    ◦ Fallos de Validación de Contenido: Aunque la IA genere texto, el backend podría aplicar validaciones post-generación (ej., longitud mínima/máxima, presencia de palabras clave, ausencia de información prohibida).
2. Estrategias de Manejo de Errores:
    ◦ Mensajes de Error al Usuario: En caso de fallo de la IA o respuesta inesperada, el frontend mostrará un mensaje de error claro y conciso al usuario (ej., "Lo sentimos, no pudimos procesar su solicitud en este momento. Por favor, intente de nuevo más tarde." o "Hubo un problema al conectar con la IA.").
    ◦ Registro de Logs de Auditoría: Cada interacción con la IA, incluyendo los intentos, las respuestas y, crucialmente, los errores, debe registrarse cronológicamente en el Servicio de Logs de Auditoría (Cloud Logging / Audit Logs). Estos logs son inmutables (Write Once, Read Many - WORM), lo que permite rastrear, depurar y analizar cualquier anomalía, siendo vitales para la seguridad y el cumplimiento normativo.
    ◦ Mecanismos de Reintento (Retry Mechanisms): Para errores transitorios (ej., problemas de red, límites de tasa), el backend podría implementar reintentos con retraso exponencial antes de reportar un fallo definitivo.
    ◦ Fallback a Respuestas por Defecto/Manuales: Para funciones críticas donde la IA falla, se podría tener un mecanismo de fallback que proporcione una respuesta predefinida o dirija al usuario a una opción manual.
    ◦ Monitoreo y Alertas: Configuración de alertas en Cloud Logging o Cloud Monitoring para notificar a los administradores sobre patrones de errores de la IA o sobre respuestas que requieran revisión humana.
    ◦ Control de Seguridad de Datos: El backend es el guardián de la información. Antes de enviar datos sensibles (ej., historial clínico del paciente) a la IA, el backend debe asegurarse de que la información se anonimice o se envíe solo la parte estrictamente necesaria. Las respuestas de la IA deben ser cuidadosamente validadas para evitar que devuelva inadvertidamente información sensible o incorrecta. El control de acceso basado en roles (RBAC) es crucial para asegurar que solo usuarios autorizados puedan interactuar con funcionalidades que impliquen la IA y datos sensibles.
Esta integración asegura que la IA se utilice de manera efectiva para optimizar el flujo de trabajo, manteniendo siempre la seguridad y la integridad de la información del paciente.

5. Manejo de Datos y Privacidad en la Integración con IA
La protección de los datos sensibles del paciente es una prioridad máxima en el diseño y la implementación del Ecosistema Digital Inteligente para Cirugía Especial [1_2, 1_6, 1_14, 1_15, 2_9, 2_10, 2_11, 2_12, 2_14, 4_185, 4_186, 4_193, 7_74, 7_85, 7_97, 7_108, 1_261]. La integración con el modelo de IA Gemma 3 se rige por estrictos protocolos de seguridad y privacidad para garantizar el cumplimiento normativo y la confianza del usuario.
• Cómo se asegurará que los datos sensibles del paciente utilizados por la IA se manejen de forma segura y cumplan con las normativas de privacidad:
    ◦ Cifrado de Datos en Tránsito (HTTPS/TLS 1.3): Toda la comunicación entre la aplicación de escritorio y la infraestructura en la nube (Google Cloud Platform - GCP), donde reside el backend y la IA, se realiza exclusivamente a través de HTTPS (TLS 1.3)

[1_5, 1_14, 2_19, 3_26, 4_61, 4_69, 4_190, 4_193, 7_76, 7_99, 1_181, 1_199, 1_203]. Esto asegura que las credenciales de usuario y cualquier dato sensible del paciente transmitido al modelo de IA estén cifrados y protegidos contra la interceptación o manipulación [1_14, 1_181, 1_203].
    ◦ Cifrado de Datos en Reposo: Los datos sensibles de los pacientes, incluidos los historiales clínicos y los archivos digitales (PDF, imágenes, videos), se almacenan cifrados en reposo tanto en la Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL) como en el Almacenamiento de Archivos (Cloud Storage - S3/Blob) [1_6, 1_14, 2_18, 2_19, 4_59, 4_61, 4_69, 4_190, 4_193, 7_75, 7_76, 7_98, 7_99, 1_181, 1_203, 1_214, 1_221, 1_253, 1_274]. Las contraseñas de los usuarios se almacenan como hash_password en la base de datos, lo que indica el uso de funciones de hash seguras en lugar de texto plano [1_6, 1_14, 1_67, 1_181, 1_203, 1_218, 1_202]. Este cifrado es una medida esencial contra fugas de datos [1_14, 2_19, 4_69, 4_193, 7_76, 7_99].
    ◦ Control de Acceso Basado en Roles (RBAC): El sistema impone un control estricto de acceso. Cada usuario autenticado, al iniciar sesión, es asociado a un rol específico (ej., "Admin Principal", "Admin Secundario", "Asistente", "Invitado (Solo Lectura)") que define sus permisos sobre la información y las funcionalidades del sistema [1_2, 1_6, 1_13, 2_19, 3_28, 4_68, 4_193, 7_76, 7_99, 1_181, 1_203, 1_254, 1_267, 1_268]. Esto limita el riesgo de acceso indebido a datos sensibles, asegurando que la IA solo procese la información a la que el usuario que inicia la interacción tiene permiso [1_13, 1_181, 1_197, 1_203].
    ◦ Servicio de Logs de Auditoría Inmutables (WORM): Cada interacción con la IA y cualquier acción relevante del usuario (como la modificación de datos de paciente o la creación de citas) se registra cronológicamente en el Servicio de Logs de Auditoría (Cloud Logging / Audit Logs) [1_7, 1_15, 2_18, 4_62, 4_69, 4_190, 4_193, 7_75, 7_76, 7_98, 7_99, 1_181, 1_199, 1_203, 1_219, 1_228, 1_274]. Estos logs son inmutables (Write Once, Read Many - WORM), lo que asegura su integridad y los hace vitales para rastrear actividades, detectar anomalías, responder a incidentes y asegurar el cumplimiento normativo [1_7, 1_15, 2_18, 4_69, 4_190, 4_193, 7_75, 7_76, 7_98, 7_99, 1_181, 1_199, 1_203].
    ◦ Backend como Guardián de Datos: El backend del sistema actúa como el guardián de la información sensible [Conversación previa]. Antes de enviar cualquier dato a la IA, el backend debe asegurarse de que solo se envíe la parte estrictamente necesaria de la información [Conversación previa]. Además, las respuestas de la IA deben ser cuidadosamente validadas para evitar que devuelva inadvertidamente información sensible o incorrecta [Conversación previa].
• Especificar si los datos sensibles se anonimizan o pseudonimizan antes de enviarlos a la IA (si aplica y es necesario/posible con el modelo):
    ◦ Aunque las fuentes no detallan explícitamente el mecanismo de anonimización o pseudonimización, la directriz es clara: el backend "debe asegurarse de que la información se anonimice o se envíe solo la parte estrictamente necesaria" antes de enviarla a la IA [Conversación previa]. Esto implica que, si la naturaleza de la consulta a la IA lo permite (ej., la IA necesita solo el contexto general o un atributo específico, no la identidad del paciente), se priorizará el envío de datos anonimizados o pseudonimizados. En casos donde se requiere información específica del paciente (ej., para redactar un recordatorio con su nombre), se enviará solo la información mínima indispensable para la tarea.
• Consideraciones sobre dónde se procesan los datos (en la nube del proveedor de IA, en tu propia VPC en GCP):
    ◦ El modelo de IA Gemma 3 (o similar a Gemini) se integra directamente en el Servidor de Aplicación (Backend) del ecosistema mediante el uso de APIs [2_18, 3_25, 3_26, 4_44, 4_193, 1_220]. El backend, a su vez, está desplegado en Google Cloud Platform (GCP), utilizando servicios como Compute Engine o Cloud Run [2_18, 3_26, 1_5, 1_221].
    ◦ La infraestructura completa en la nube reside dentro de una Virtual Private Cloud (VPC) [1_15, 2_18, 7_75, 7_98, 1_199, 1_203, 1_220]. Esto crea un entorno de red privado y seguro, permitiendo al equipo de desarrollo definir su propia topología de red virtual y controlar el tráfico [2_18, 7_75, 7_98].
    ◦ Si el modelo Gemma 3 es accedido a través de una API pública de Google AI (como se menciona para Gemini), el procesamiento real de la IA ocurriría en los servidores del proveedor de IA (Google). Sin embargo, la comunicación del backend con esta API se mantendría dentro del entorno seguro de GCP y, en la medida de lo posible, dentro de la VPC para mayor seguridad, limitando la exposición de los datos [2_18, 1_15, 1_199, 1_203, Conversación previa].
    ◦ Si el modelo se despliega directamente en Vertex AI (el servicio de Machine Learning de GCP), el procesamiento de los datos por la IA ocurriría dentro de la propia

infraestructura de GCP del proyecto, lo que podría ofrecer un control más granular sobre la ubicación y seguridad de los datos, potencialmente manteniéndolos dentro de la VPC definida para el ecosistema [2_18, 7_75, 7_98, 1_222]. Esto es preferible para datos altamente sensibles.

6. Rendimiento y Escalabilidad de la Integración con IA

El rendimiento y la escalabilidad de la integración con IA son fundamentales para asegurar la "eficiencia operativa" y una experiencia de usuario fluida [2_18, 3_22, 4_186].

• Requisitos de rendimiento: Tiempo de respuesta esperado de la IA para cada caso de uso (ej. la respuesta del chat debe ser rápida):
    ◦ Para el Chat Interno Básico y las Automatizaciones Básicas (redacción de correos, generación de hojas membretadas, recordatorios), el tiempo de respuesta de la IA debe ser rápido para no interrumpir el flujo de trabajo del personal [3_22, 1_254, 1_159]. Si bien no se especifica un número exacto en las fuentes, la meta de "asistencia rápida" y "optimizar la eficiencia operativa" implica una latencia muy baja, idealmente sub-segundo para interacciones de chat simples y unos pocos segundos para tareas de generación de texto más complejas. Un tiempo de respuesta lento impactaría negativamente la adopción y el valor percibido de la IA.
    ◦ Para Asistencia para el Desarrollo y Capacitación (generación de scripts SQL básicos, FAQs), los requisitos de tiempo de respuesta pueden ser un poco más flexibles, pero aún se espera un rendimiento eficiente para mantener la productividad.
• Consideraciones sobre la latencia en las llamadas a la API de la IA:
    ◦ La latencia es un factor crítico en las comunicaciones de la aplicación de escritorio con la nube. La comunicación se realiza a través de APIs RESTful utilizando HTTPS (TLS 1.3) [4_61, 4_190, 1_5, 1_14, 2_19, 3_26].
    ◦ El rendimiento óptimo del ecosistema requiere una conexión a internet estable de al menos 250 Mbps [1_10, 2_19, 4_63, 4_193, 7_76, 7_99, 1_118, 1_245, 1_259, 1_272]. Esta especificación de conectividad de red de alta velocidad indica la expectativa de una baja latencia general del sistema.
    ◦ Si el modelo de IA se accede a través de una API pública externa, la latencia también dependerá de la ubicación geográfica del endpoint de la API y de la congestión de la red de internet. Si el modelo se despliega en Vertex AI dentro de la misma región de GCP que el backend del ecosistema, la latencia interna de la nube se minimizará significativamente.
• Cómo la infraestructura soportará el volumen esperado de interacciones con la IA a medida que crezca el uso del Dashboard:
    ◦ La arquitectura de nube en GCP está diseñada para ofrecer escalabilidad y disponibilidad inherente [2_17, 2_19, 7_75, 7_76, 7_98, 7_99]. Esto es crucial para soportar el crecimiento del volumen de interacciones con la IA.
    ◦ El Servidor de Aplicación (Backend), donde se integra la IA, se desplegará utilizando servicios de GCP como Compute Engine o Cloud Run [2_18, 3_26, 1_221]. Cloud Run, en particular, ofrece escalado automático basado en la demanda, lo que permite que el backend maneje un mayor volumen de solicitudes de IA sin intervención manual.
    ◦ El API Gateway / Load Balancer actúa como el punto de entrada seguro y distribuye el tráfico de manera eficiente entre los servidores de aplicación, optimizando el rendimiento y garantizando la disponibilidad del sistema incluso bajo alta demanda [2_18, 7_75, 7_98, 1_181, 1_203, 1_220]. Esto es fundamental para absorber picos de uso de la IA.
    ◦ La Base de Datos Relacional (Cloud SQL) y el Almacenamiento de Archivos (Cloud Storage) también son servicios escalables de GCP, capaces de manejar un creciente volumen de datos de pacientes que podrían ser utilizados como contexto para la IA [2_18, 7_75, 7_98].
• Consideraciones de costo basadas en el uso (por token, por llamada):
    ◦ Las fuentes establecen un costo general para la infraestructura en la nube de GCP: una cortesía de 6 meses (hasta 1 TB mensual) desde la entrega de la Fase 1, tras lo cual el costo será de $65.00 USD mensuales, adicionales al plan de mantenimiento [2_20, 1_117, 1_257, 1_270, 7_76, 7_99]. Esta cortesía está directamente vinculada a la vigencia del Plan de Mantenimiento y Soporte [2_20, 1_117, 1_257, 1_270, 7_76, 7_99].
    ◦ Cualquier almacenamiento adicional más allá del 1 Terabyte incluido tendrá un costo extra según las tarifas del proveedor de nube [2_20, 1_118, 1_259, 1_272, 7_76, 7_99].
    ◦ Las fuentes no desglosan los costos específicamente por el uso del modelo de IA (ej., por token, por llamada a la API de Gemma 3). Sin embargo, el costo de $65.00 USD mensuales después de la cortesía es el monto total por el servicio en la nube que alberga toda la infraestructura, incluyendo el backend que interactúa con la IA [2_20, 1_117, 1_257, 1_270]. Si el uso de la IA se incrementara significativamente y esto supusiera un

aumento sustancial en el consumo de recursos de cómputo (CPU, memoria) o de red del backend, podría implicar que los costos operativos de GCP superen el monto fijo y requieran una revisión, aunque las fuentes solo especifican costos adicionales por almacenamiento. Para la alta dirección, es clave entender que la IA es parte de un ecosistema cuyo costo operativo está detallado en los términos contractuales, y que la escalabilidad del sistema podría generar costos adicionales si se exceden los límites del plan actual de 1TB o el volumen de cómputo asociado.

---

7. Consideraciones Adicionales sobre la Integración de IA

La integración del modelo de Inteligencia Artificial Gemma 3 (o similar a Gemini) es un pilar fundamental para potenciar la eficiencia operativa y la asistencia al personal dentro del "Ecosistema Digital Inteligente para Cirugía Especial". Para garantizar que esta capacidad transversal sea efectiva, segura y escalable a largo plazo, es esencial considerar no solo su implementación inicial sino también su monitoreo y las potenciales vías de expansión futura.

7.1. Monitoreo del Uso y Rendimiento de la IA

Aunque los documentos no especifican un servicio de monitoreo exclusivo para la IA, la arquitectura del ecosistema ya incluye un "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)". Este servicio es vital para la seguridad y el seguimiento de operaciones, registrando cronológicamente todas las actividades y cambios realizados en el sistema, como modificaciones de datos de pacientes o accesos de usuarios. Estos logs son inmutables (Write Once, Read Many - WORM), asegurando su integridad y siendo cruciales para la detección de anomalías y la respuesta a incidentes.

Para la IA, la extensión de este monitoreo podría incluir:

• Registro de Interacciones de IA: Cada solicitud al modelo Gemma 3 (ya sea desde el chat interno o para automatizaciones como redacción de correos) y su respuesta deberían ser registradas en los logs de auditoría. Esto permitiría un seguimiento de la frecuencia de uso de las funcionalidades de IA, identificar patrones de consulta y auditar el contenido generado.

• Métricas de Rendimiento: Aunque no se detalla, la implementación de la IA como parte del backend permite el monitoreo de métricas técnicas como la latencia de respuesta del modelo, la tasa de errores en las peticiones y el consumo de recursos (CPU, memoria) en Google Cloud Platform (GCP). Esto es fundamental para asegurar que la IA responde de manera oportuna y eficiente.

• Calidad de la Generación de Contenido: Para funciones como la redacción de correos o informes, es importante considerar mecanismos, aunque sean manuales inicialmente, para evaluar la calidad y precisión del texto generado por la IA. Esto podría implicar que los usuarios validen el contenido o proporcionen retroalimentación directa en la interfaz, lo que, a su vez, podría alimentar futuras mejoras del modelo o ajustes en los "prompts".

• Privacidad de Datos en Interacciones con IA: Dado que la IA procesará consultas relacionadas con información sensible del consultorio (aunque el chat sea de texto solamente y no genere archivos), es "crucial que esta integración se realice de forma segura, garantizando la privacidad de los datos sensibles de los pacientes". El monitoreo debe asegurar que no haya exposición indebida de datos y que las políticas de retención y anonimización sean respetadas.

7.2. Posibles Futuras Expansiones de Funcionalidades de IA

La integración de Gemma 3 en el backend mediante APIs establece una base sólida para el crecimiento y la evolución de las capacidades de IA del ecosistema. Más allá de las funcionalidades iniciales de automatización básica (redacción de correos, hojas membretadas, recordatorios) y el chat interno (consultas rápidas, resúmenes, redacción de informes), así como la asistencia en la generación de FAQs para manuales y scripts SQL, y la potenciación del Asistente Virtual Inteligente (AVI) en WhatsApp para FAQ y agendamiento, se pueden vislumbrar las siguientes expansiones estratégicas:

• Generación Avanzada de Documentos y Reportes Clínicos: La capacidad actual de "redacción de informes" y "generación de hojas membretadas" podría evolucionar. La IA podría generar borradores más completos de informes médicos, resúmenes de altas o consentimientos informados, extrayendo información directamente del "Historial Clínico Detallado" del paciente de forma estructurada.

• Asistencia en la Clasificación y Etiquetado de Documentos: Dada la "Gestión Documental Avanzada" que permite subir y asociar archivos digitales como PDFs e imágenes, la IA podría ayudar a etiquetar automáticamente estos documentos (ej. "Radiografía", "Resultados de Laboratorio", "Formulario de Admisión") o incluso a resumir su contenido textual si son

PDFs, mejorando la organización y búsqueda.
• Análisis y Sugerencias Basadas en Datos del Paciente: Si bien los documentos enfatizan la seguridad y el control, en un futuro, con estrictas consideraciones de privacidad, la IA podría analizar el historial clínico para sugerir posibles diagnósticos diferenciales, recordar al médico sobre interacciones medicamentosas o alertas de alergias (si los datos estructurados lo permiten), o proponer planes de seguimiento personalizados basados en guías clínicas preestablecidas. Esto requeriría modelos de IA entrenados específicamente en datos médicos y validados rigurosamente.
• Interfaz de Voz para el Dashboard Interno: La aplicación móvil para pacientes ya contempla una "Interfaz de Voz" para accesibilidad. Una futura expansión lógica sería integrar esta capacidad de interacción por voz directamente en el Dashboard de Escritorio, permitiendo al personal del consultorio interactuar con el sistema y la IA mediante comandos de voz para búsquedas, agendamiento de citas o activación de automatizaciones, mejorando aún más la "eficiencia operativa".
• Personalización y Monitoreo Proactivo de Pacientes (con el AVI): El Asistente Virtual Inteligente (AVI) en WhatsApp podría expandir sus capacidades más allá de FAQs y agendamiento. Podría enviar mensajes personalizados de recordatorio para toma de medicación, o solicitar actualizaciones de estado de forma proactiva, basándose en el plan de tratamiento del paciente.
• Soporte de Desarrollo y Mantenimiento: La IA ya puede generar "scripts SQL básicos a partir de descripciones en lenguaje natural" y colaborar en "generación de preguntas frecuentes (FAQs)" para el manual de usuario y "crear un chatbot de ayuda inicial para la capacitación del personal". Estas capacidades podrían ampliarse para incluir la asistencia en la depuración de código, la generación de casos de prueba o incluso la sugerencia de mejoras en la arquitectura del sistema, acelerando los ciclos de desarrollo y mantenimiento.
La implementación de estas futuras funcionalidades dependerá de la evolución de las necesidades del consultorio, la madurez de la tecnología de IA, y, crucialmente, el cumplimiento de las normativas de privacidad y seguridad de datos, dada la naturaleza sensible de la información manejada. La flexibilidad de la arquitectura de nube de GCP y la integración modular de la IA a través de APIs facilitan estas expansiones.
Plan Detallado de Pruebas para el Dashboard del Ecosistema Digital para Cirugía Especial

1. Introducción y Propósito

Este documento detalla el plan de pruebas exhaustivas para el "Dashboard de Escritorio", el cual es el "Núcleo Administrativo Central" y el entregable principal de la Fase 1 del proyecto "Ecosistema Digital Inteligente para Cirugía Especial". El Dashboard de Escritorio es la columna vertebral y la base de datos central de todo el ecosistema, y su propósito es optimizar la gestión de pacientes, mejorar la eficiencia operativa, y centralizar la información crítica del consultorio en un lugar seguro y accesible.
El objetivo de este plan es asegurar la calidad, fiabilidad, rendimiento y seguridad de la aplicación antes de su despliegue y uso en un entorno real. Las pruebas son de vital importancia para el "Ecosistema Digital Inteligente para Cirugía Especial", ya que el sistema busca ser "potente, seguro y excepcionalmente intuitivo". A través de un proceso de pruebas riguroso, se busca reducir los "procesos manuales que consumen tiempo" y el "riesgo de errores", liberando tiempo valioso para el personal.
Un enfoque primordial de este ecosistema es la "seguridad de datos", ya que maneja información sensible del paciente. Las pruebas de seguridad son cruciales para validar las múltiples capas de protección implementadas, incluyendo:
• El Control de Acceso Basado en Roles (RBAC), que garantiza que solo usuarios autorizados (Administradores, Admin Principal, Admin Secundario, Asistentes) accedan a funcionalidades e información según sus permisos.
• El cifrado de datos en tránsito a través de HTTPS (TLS 1.3) entre la aplicación de escritorio y la infraestructura en la nube.
• El almacenamiento seguro de credenciales mediante hashing y el cifrado de datos sensibles en reposo en la Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL) y el Almacenamiento de Archivos (Cloud Storage - S3/Blob).
• El Servicio de Logs de Auditoría (Cloud Logging / Audit Logs), que registra inmutablemente (Write Once, Read Many - WORM) todos los intentos de login y las actividades del usuario, siendo vital para la detección de anomalías y la respuesta a incidentes.
• La Virtual Private Cloud (VPC) y el API Gateway / Load Balancer, que actúan como la

primera línea de defensa para la infraestructura en la nube.
La necesidad de una conexión a internet estable de al menos 250 Mbps para un rendimiento óptimo del ecosistema también subraya la importancia de las pruebas de rendimiento y estrés. En resumen, este plan de pruebas es un pilar fundamental para garantizar que el Dashboard cumpla con su promesa de ser un centro de control digital potente, seguro y confiable para el consultorio del Dr. Sánchez García.

2. Alcance de las Pruebas

El presente Plan Detallado de Pruebas se enfoca exclusivamente en el "Dashboard de Escritorio", también conocido como el "Núcleo Administrativo Central", el cual es el entregable principal de la Fase 1 del proyecto "Ecosistema Digital Inteligente para Cirugía Especial". Este alcance específico garantiza que la columna vertebral del ecosistema sea robusta, segura y funcional antes de proceder con fases posteriores.

2.1. Componentes y Funcionalidades Incluidas en las Pruebas (Fase 1 del Dashboard)

Las pruebas cubrirán exhaustivamente todos los módulos y flujos de usuario detallados para el Dashboard de Escritorio en la Fase 1 del proyecto. Esto incluye, pero no se limita a:

- Flujo de Acceso: Login de Usuario y Dashboard Principal:
    - Login de Usuario: Autenticación de usuarios con "Nombre de Usuario" y "Contraseña". Pruebas de inicio de sesión exitoso y fallido, incluyendo mensajes de error por credenciales incorrectas o campos vacíos. Validación de la persistencia de la sesión tras un login exitoso y la opción de "Cerrar Sesión".
    - Dashboard Principal (Vista General): Verificación de la visualización correcta de la "Cabecera" (usuario logueado y rol) y la adaptación de la "Barra de Navegación Lateral" según el rol del usuario. Pruebas de la visualización y actualización de los paneles del "Área Central del Dashboard" como "CITAS DEL DÍA", "MÉTRICAS RÁPIDAS" y "PACIENTES RECIENTES".
    - Navegación: Pruebas de la navegación a todas las secciones principales y sub-secciones a través de la barra lateral: "Pacientes", "Agenda", "Procedimientos", "Reportes", "Documentos", "Administración".
    - Widgets y Accesos Directos: Verificación del acceso rápido a Gmail, Google Drive y aplicaciones de Office a través de los widgets visibles en la interfaz.
- Gestión Integral del Expediente del Paciente:
    - Creación de Perfiles de Paciente: Verificación del formulario de "Creación de Perfil de Paciente", incluyendo todos los campos demográficos y de contacto. Pruebas de validación de campos obligatorios y formatos inválidos.
    - Búsqueda Avanzada de Pacientes: Pruebas de la funcionalidad de búsqueda por cualquier campo (nombre, ID, palabra clave en notas) y la visualización de resultados.
    - Registro Historial Clínico Detallado: Verificación del formulario "Registro de Historial Clínico", incluyendo campos para "Fecha de Consulta", "Diagnóstico", "Notas de Evolución", "Procedimientos y Tratamientos". Pruebas de la capacidad para añadir nuevas entradas cronológicas.
- Gestión Documental Avanzada:
    - Subida y Asociación de Documentos: Pruebas de la funcionalidad para subir y asociar todo tipo de archivos digitales (PDF, imágenes, videos cortos) directamente al expediente del paciente o a una entrada específica del historial clínico. Verificación de mensajes de error para tipos de archivo no soportados o tamaños excedidos.
    - Visualización de Documentos: Pruebas del visor de archivos visuales con su carrusel de vistas previas y filtros por origen/tipo (Cloud Drives, Photos, Videos).
- Módulo de Agenda y Citas:
    - Agendar Nueva Cita: Pruebas del formulario para agendar citas, incluyendo la selección de paciente, fecha, hora, tipo de cita y estado. Verificación de la detección de conflictos de horario.
    - Gestión de Citas Existentes: Pruebas de las funcionalidades para reprogramar, cancelar y cambiar el estado de las citas existentes.
    - Visualización de Calendario: Verificación del calendario interactivo y la visualización de citas del día.
- Seguridad, Roles y Accesos Rápidos:
    - Gestión de Usuarios y Asignación de Roles: Pruebas de la capacidad para crear nuevas cuentas de usuario y asignar roles (Administrador, Admin Principal, Admin Secundario, Asistente, Invitado (Solo Lectura)). Esta funcionalidad será probada exhaustivamente para verificar que cada rol solo tiene acceso a sus permisos definidos.
    - Acceso a Logs de Auditoría: Verificación de la funcionalidad para visualizar los registros cronológicos de actividades y cambios realizados en el sistema.
- Inteligencia y Automatización Integrada (Modelo de IA Gemma 3):

    ◦ Chat Interno Básico: Pruebas de la interacción con el chat interno para consultas rápidas, resúmenes de texto o redacción de informes (solo texto).
    ◦ Automatizaciones Básicas: Verificación de la capacidad para generar borradores de correos electrónicos, hojas membretadas y recordatorios a través de la IA.
• Capacitación Inicial:
    ◦ Generación de Material: Aunque la capacitación es una actividad programada, se probará la disponibilidad y relevancia del manual de funcionamiento y las preguntas frecuentes (FAQs) que la IA puede ayudar a generar.

2.2. Aspectos Técnicos Incluidos en las Pruebas

Las pruebas técnicas validarán la correcta interconexión y funcionamiento de los distintos componentes del ecosistema que soportan el Dashboard:

• Integración Frontend-Backend:
    ◦ Verificación de la comunicación fluida entre la aplicación de escritorio (frontend desarrollada con React/Vue.js y empaquetada con Electron.js) y el Servidor de Aplicación (backend construido con Python/Flask/Django o Node.js/Express).
    ◦ Pruebas de la exactitud y consistencia de los datos entre el frontend y el backend en todas las operaciones CRUD (Crear, Leer, Actualizar, Eliminar - suave).
    ◦ Medición de los tiempos de respuesta de las APIs RESTful.
• Integración Backend-Base de Datos:
    ◦ Verificación de la interacción del backend con la Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL).
    ◦ Pruebas de la integridad de los datos almacenados.
    ◦ Evaluación del rendimiento de las consultas a la base de datos para asegurar el acceso rápido a la información.
    ◦ Confirmación de que los datos sensibles están cifrados en reposo en la base de datos.
• Integración Backend-Cloud Storage:
    ◦ Verificación de la interacción del backend con el Almacenamiento de Archivos (Cloud Storage - S3/Blob) para la gestión de documentos digitales.
    ◦ Pruebas de las funcionalidades de subida y descarga de archivos, y su correcta asociación con los expedientes de pacientes.
    ◦ Confirmación de que los archivos están cifrados en reposo en Cloud Storage.
• Integración Backend-IA (Gemma 3):
    ◦ Verificación de la integración del modelo de Inteligencia Artificial Gemma 3 (o similar a Gemini) en el backend mediante APIs.
    ◦ Pruebas de la precisión y relevancia de las respuestas y contenidos generados por la IA (redacción de correos, hojas membretadas, recordatorios, respuestas del chat, resúmenes).
    ◦ Evaluación del rendimiento y la latencia en las interacciones con la IA.
• Seguridad de APIs y Control de Acceso (RBAC):
    ◦ Validación de que toda la comunicación entre la aplicación de escritorio y la infraestructura en la nube se realiza exclusivamente a través de HTTPS (TLS 1.3), garantizando el cifrado de datos en tránsito.
    ◦ Pruebas exhaustivas del Control de Acceso Basado en Roles (RBAC) para asegurar que cada rol (Administrador, Admin Principal, Admin Secundario, Asistente, Invitado - Solo Lectura) tiene acceso solo a las funcionalidades y datos que le corresponden. Esto incluye intentos de acceso no autorizado y verificación de mensajes de error adecuados.
    ◦ Confirmación de que las credenciales de usuario se almacenan de forma segura mediante hashing y cifrado en reposo.
    ◦ Verificación de que el API Gateway / Load Balancer actúan como la primera línea de defensa para el backend.
• Registro y Auditoría (Logs de Auditoría):
    ◦ Verificación de que todos los intentos de login (exitosos y fallidos) y cada acción relevante del usuario (ej., modificación de datos de paciente, creación de citas, gestión de usuarios) se registran cronológicamente en el "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)".
    ◦ Confirmación de la inmutabilidad de los logs (Write Once, Read Many - WORM) para asegurar su integridad y trazabilidad.
• Manejo de Errores y Estados Vacíos:
    ◦ Pruebas de todos los escenarios de error especificados (credenciales incorrectas, campos vacíos/formato inválido, problemas de conectividad) y la visualización de mensajes de error claros y concisos.
    ◦ Verificación del comportamiento del sistema cuando no hay datos para mostrar en

ciertas secciones (ej., "No se encontraron pacientes", "No hay citas programadas para hoy").
• Rendimiento y Conectividad:
    ◦ Pruebas del rendimiento general de la aplicación bajo una conexión a internet estable de al menos 250 Mbps, según la recomendación para un óptimo funcionamiento del ecosistema. Esto incluirá la carga de datos, la navegación y la interacción con funcionalidades clave.
• Diseño Visual y Usabilidad (Glassmorphism Oscuro):
    ◦ Verificación de la implementación del estilo visual "Glassmorphism Oscuro" en todas las pantallas y componentes del Dashboard, asegurando la coherencia visual integral con los principios de diseño definidos (fondo gris carbón, paneles de vidrio esmerilado, efecto "Aurora", tipografía Inter, bordes sutiles y luminosos).
    ◦ Pruebas de la legibilidad óptima y el doble enfoque de contraste para la accesibilidad.
    ◦ Verificación de la retroalimentación interactiva al pulsar botones (aumento de brillo o ligero hundimiento).

2.3. Exclusiones Explícitas de las Pruebas en esta Fase del Dashboard

Es fundamental entender que este plan de pruebas se limita estrictamente a la Fase 1: El Núcleo Administrativo (Dashboard de Escritorio). Por lo tanto, las siguientes funcionalidades y componentes del ecosistema no se incluirán en las pruebas de esta fase, ya que corresponden a etapas futuras del proyecto o no forman parte del desarrollo principal:
• Componentes de la Fase 2: Expansión a la Movilidad y Presencia Web:
    ◦ La Aplicación Móvil para el equipo.
    ◦ La Página web profesional del consultorio (incluyendo su diseño "Luxury, Tech, Accesible", secciones informativas, asistente virtual para FAQ y agendamiento, y pasarela de pagos).
• Componentes de la Fase 3: Innovación y Experiencia del Paciente:
    ◦ La Aplicación Móvil exclusiva para Clientes/Pacientes (incluyendo acceso a registro médico, agendamiento, material educativo, interfaz de voz, mensajería segura, pasarela de pagos).
    ◦ La Integración del Asistente Virtual Inteligente (AVI) en WhatsApp (incluyendo su sincronización con la web, FAQ, agendamiento y notificaciones automatizadas).
• Otros Aspectos Excluidos del Alcance del Proyecto o del Desarrollo Directo:
    ◦ La digitalización (escaneo) de archivos físicos existentes.
    ◦ Los servicios de marketing digital, gestión de redes sociales o campañas de promoción.
    ◦ La compra de dominio y contratación del servicio de hosting para la página web, ya que son responsabilidad del cliente.
    ◦ Cualquier funcionalidad o solicitud de cambio que no esté explícitamente detallada en el alcance definido para la Fase 1 del Dashboard.

Aunque el backend y la infraestructura en la nube están diseñados para soportar futuras fases, el enfoque de estas pruebas estará en la interacción del Dashboard de Escritorio con estos componentes, sin extenderse a la validación de las funcionalidades o interfaces de las aplicaciones móviles o la página web en sí.

Tipos de Pruebas

Pruebas Unitarias

• Propósito: Las pruebas unitarias tienen como objetivo verificar que unidades individuales de código, como funciones, métodos o componentes pequeños, funcionan correctamente de forma aislada. Esto permite identificar y corregir defectos en las etapas tempranas del ciclo de desarrollo, reduciendo costos y complejidad.
• Aplicación:
    ◦ En el Backend: Se aplicarán en módulos clave de la lógica de negocio, como las funciones de validación de datos (ej., validación de entrada para nombres de usuario, contraseñas, o campos de paciente), la lógica de autenticación y autorización basada en roles (verificando que un usuario con un rol específico tenga o no acceso a ciertas funcionalidades), y la interacción básica con APIs externas (como las llamadas a la API de Gemma 3 para automatizaciones).
    ◦ En el Frontend: Se realizarán en componentes reutilizables como botones de acción ("Iniciar Sesión", "Guardar Paciente", "Agendar Cita"), campos de entrada, elementos de navegación, y la lógica de manipulación local de datos o estados de la UI.
• Enfoque: Se centran en la corrección interna del código y su funcionamiento aislado, sin depender de otros módulos o servicios.

Pruebas de Integración

• Propósito: Las pruebas de integración buscan verificar la correcta interacción y comunicación entre diferentes módulos o servicios que componen el ecosistema. Esto es crucial para un sistema distribuido como este, que depende de la nube y de múltiples componentes interconectados.

• Aplicación:

    ◦ Frontend-Backend API Communication: Se probará la comunicación entre la aplicación de escritorio y el Servidor de Aplicación (Backend) a través de APIs RESTful, asegurando que los datos se envíen y reciban correctamente utilizando HTTPS (TLS 1.3) para cifrar los datos en tránsito.

    ◦ Backend-Base de Datos (Cloud SQL): Verificación de que el backend interactúa correctamente con la Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL) para almacenar, recuperar y actualizar datos de pacientes, historiales clínicos, citas, usuarios y roles. Esto incluye el almacenamiento seguro y cifrado en reposo.

    ◦ Backend-Cloud Storage: Comprobar que el backend puede subir, descargar y asociar archivos digitales (PDFs, imágenes, videos cortos) a Cloud Storage de forma segura y cifrada.

    ◦ Backend-IA (Gemma 3): Asegurar que las llamadas del backend a la API del modelo de Inteligencia Artificial Gemma 3 (o similar a Gemini) para funciones de automatización y el chat interno se realicen de manera eficiente y devuelvan los resultados esperados.

• Enfoque: Garantizar que los datos se transfieran correctamente entre los componentes y que las interfaces (APIs) funcionen según lo esperado, sin errores de comunicación o formato.

Pruebas Funcionales (End-to-End)

• Propósito: Estas pruebas verifican que cada funcionalidad completa del Dashboard, desde la interacción del usuario en el frontend hasta la acción en el backend y la base de datos, funciona de acuerdo con los requisitos de negocio. Simulan el uso real del sistema por parte del personal del consultorio.

• Aplicación: Se enfocarán en probar cada flujo de usuario clave detallado en los documentos proporcionados, especialmente en el "Mapa Detallado: Recorrido del Usuario en el Dashboard de Escritorio".

    ◦ Flujo de Login: Desde el ingreso de credenciales hasta la redirección exitosa al Dashboard principal y la correcta visualización de las secciones según el rol del usuario.

    ◦ Gestión de Pacientes: Creación de perfiles, búsqueda avanzada, y el registro cronológico de historial clínico detallado con la capacidad de adjuntar documentos.

    ◦ Gestión Documental: Subida, asociación y visualización de archivos digitales (PDF, imágenes, videos cortos) directamente en el expediente del paciente.

    ◦ Módulo de Agenda y Citas: Programación, reprogramación y gestión de citas con estados personalizables, incluyendo la visualización de citas del día.

    ◦ Gestión de Usuarios y Roles: Creación de nuevas cuentas de usuario y asignación de roles con permisos específicos, así como el acceso (para Admin Principal) a los logs de auditoría.

    ◦ Interacciones de IA y Automatización: Uso del chat interno para consultas rápidas, resúmenes de texto o redacción de informes, y activación de automatizaciones básicas como redacción de correos o generación de hojas membretadas.

• Enfoque: Validar que el sistema completo, desde el punto de vista del usuario final, cumple con todas las especificaciones funcionales y de negocio. Se confirmará que los datos se transfieren y persisten correctamente en cada paso del flujo, y que las respuestas del sistema son las esperadas en diferentes escenarios (éxito, error, estados vacíos).

Pruebas de UI/Usabilidad

• Propósito: Estas pruebas se centran en verificar que la interfaz de usuario es intuitiva, fácil de navegar, visualmente consistente y que los elementos interactivos se comportan como se espera. También buscan asegurar la comodidad visual y la legibilidad.

• Aplicación: Se aplicarán a toda la interfaz de usuario del Dashboard, desde la pantalla de Login hasta todas las secciones y paneles del Dashboard Principal, siguiendo rigurosamente el estilo visual "Glassmorphism Oscuro".

• Enfoque:

    ◦ Experiencia del Usuario: Evaluar la facilidad de interacción con componentes como botones, formularios, tablas y menús.

    ◦ Navegación: Asegurar que la navegación entre vistas sea fluida e intuitiva.

    ◦ Coherencia Visual: Confirmar que el diseño "Glassmorphism Oscuro", con su fondo gris carbón (#121212), paneles flotantes de "vidrio esmerilado" (backdrop-filter: blur(24px)),

auras de color (azul #4A69FF y violeta #8C52FF), texto nítido y brillante (#F5F5F5), y bordes sutiles y luminosos, se aplica de manera consistente en toda la interfaz.
    ◦ Accesibilidad: Verificar que la legibilidad del texto sea óptima (contraste > 4.5:1 WCAG) y que el diseño sea accesible, especialmente considerando a pacientes mayores o con dificultades tecnológicas, como se busca en el ecosistema general.
    ◦ Retroalimentación Interactiva: Probar que la interacción al pulsar botones (sutil aumento de brillo o ligero hundimiento) proporcione una retroalimentación clara e inmediata al usuario.

Pruebas de Seguridad

• Propósito: Las pruebas de seguridad tienen como propósito identificar vulnerabilidades y asegurar que la aplicación protege los datos sensibles y controla el acceso adecuadamente. Es crucial para abordar las "preocupaciones de seguridad de datos".
• Enfoque:
    ◦ Autenticación (Login con credenciales válidas/inválidas): Se probará rigurosamente la pantalla de Login, que es el punto de entrada fundamental al "Ecosistema Digital Inteligente para Cirugía Especial".
        ▪ Se verificará que solo los usuarios autorizados (Administradores, Admin Principal, Admin Secundario, Asistentes) puedan acceder al sistema.
        ▪ Las credenciales se enviarán de forma segura al Servidor de Aplicación (Backend) alojado en Google Cloud Platform (GCP), utilizando HTTPS (TLS 1.3) para cifrar los datos en tránsito, protegiéndolos de interceptaciones.
        ▪ El backend verificará las credenciales contra la Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL), donde los datos sensibles como contraseñas se almacenan cifrados en reposo y mediante funciones de hash (indicado por hash_password VARCHAR).
        ▪ En caso de credenciales incorrectas, se espera que la aplicación permanezca en la pantalla de Login y muestre un mensaje de error claro y conciso, como "Usuario o contraseña incorrectos". El Servicio de Logs de Auditoría registrará este intento fallido.
        ▪ Se probarán escenarios de campos vacíos o formatos inválidos, esperando mensajes de validación como "Campo 'Nombre de Usuario' requerido" o "Campo 'Contraseña' requerido".
    ◦ Autorización (Verificar que cada rol solo tiene acceso a las funcionalidades y datos permitidos): El sistema impone un Control de Acceso Basado en Roles (RBAC) estricto.
        ▪ Se probará cada rol ("Admin Principal", "Admin Secundario", "Asistente", "Invitado (Solo Lectura)") para asegurar que sus permisos sobre la información y funcionalidades del Dashboard se aplican correctamente.
        ▪ Por ejemplo, se verificará que solo el "Admin Principal" tenga acceso a la "Gestión de Usuarios" y a la visualización de los "Logs de Auditoría".
        ▪ Se asegurará que un "Asistente" pueda gestionar pacientes, citas y documentos, pero no tenga acceso a la gestión de usuarios o logs.
    ◦ Validación de entrada para prevenir ataques (ej. inyección SQL, XSS): Aunque las fuentes no especifican ataques concretos, se menciona la validación de entrada para campos obligatorios y formatos válidos en formularios (ej., creación de perfil de paciente, login). Esto implica que se implementarán controles a nivel de backend y frontend para sanear y validar las entradas de usuario, lo cual es fundamental para mitigar ataques de inyección (SQL, XSS) al interactuar con la Base de Datos Relacional (Cloud SQL) y al mostrar contenido en la interfaz. El uso de frameworks como Python/Flask/Django o Node.js/Express facilita la implementación de estas defensas.
    ◦ Seguridad en el manejo de archivos subidos: Se probará la funcionalidad de "Gestión Documental Avanzada" que permite subir, asociar y visualizar archivos digitales (PDFs, imágenes, videos cortos) a Cloud Storage. Se verificará que estos archivos se almacenen de forma segura y cifrados en reposo en Cloud Storage. Las pruebas incluirán la validación de tipos de archivo soportados y límites de tamaño para prevenir la subida de contenido malicioso.
    ◦ Seguridad de las APIs: Todas las comunicaciones entre la aplicación de escritorio y el Servidor de Aplicación (Backend) se realizan a través de APIs RESTful utilizando HTTPS (TLS 1.3). El API Gateway / Load Balancer actuará como el punto de entrada seguro y primera línea de defensa para todas las solicitudes, distribuyendo el tráfico y gestionando las conexiones de forma segura. Se probará la robustez de estos endpoints frente a intentos de acceso no autorizado y manipulación de datos.

Pruebas de Rendimiento

• Propósito: Medir la velocidad, capacidad de respuesta y estabilidad de la aplicación bajo diferentes cargas. El objetivo del ecosistema es optimizar la gestión de pacientes y mejorar la eficiencia operativa.
• Enfoque:

    ◦ Probar el tiempo que tarda en cargar la tabla de datos con un volumen grande de registros: El Dashboard Principal muestra tablas de datos dinámicos como "Pacientes Recientes". Se simulará un alto volumen de pacientes para asegurar que la carga de estos listados sea eficiente y no afecte la experiencia del usuario.
    ◦ El tiempo de respuesta de las APIs críticas: Se evaluará la velocidad de respuesta de las APIs más utilizadas y fundamentales, tales como:
        ▪ Búsqueda de pacientes: La "Búsqueda Avanzada" permite la búsqueda instantánea por cualquier campo. La API GET /patients se utilizará para estas pruebas.
        ▪ Obtención de historial clínico: La visualización y el registro del "Historial Clínico Detallado" son funcionalidades centrales.
        ▪ El backend procesa las solicitudes, interactúa con la Base de Datos Relacional (Cloud SQL) y el Almacenamiento de Archivos (Cloud Storage), y envía la información actualizada de vuelta al frontend.
    ◦ El rendimiento de la interfaz al navegar o interactuar con componentes complejos: Se probará la fluidez de la navegación entre las diferentes secciones del Dashboard (Pacientes, Agenda, etc.), así como la interacción con elementos que requieren mayor procesamiento, como los formularios de creación o edición, y el calendario interactivo del "Módulo de Agenda y Citas". El diseño "Glassmorphism Oscuro" busca una experiencia de usuario fluida y de alta calidad, lo que implica un rendimiento de UI óptimo.
    ◦ El impacto del uso del Visor de Archivos Visuales: Se probará cómo el sistema maneja la visualización de archivos digitales (imágenes, videos cortos) asociados a los expedientes de los pacientes, especialmente al navegar por galerías con múltiples elementos. Se evaluará el tiempo de carga y la capacidad de respuesta del visor bajo diferentes condiciones.
    ◦ Considerar el impacto del volumen proyectado de 2TB de datos (principalmente en Cloud Storage) en el rendimiento general: El sistema se apoya en una arquitectura de nube robusta y escalable en GCP. El Almacenamiento de Archivos (Cloud Storage) está diseñado para almacenar grandes volúmenes de archivos digitales y se menciona una capacidad incluida de hasta 1 Terabyte (1000 GB) mensual como cortesía inicial, con costos adicionales para volúmenes mayores. El rendimiento dependerá en gran medida de una conexión a internet estable de al menos 250 Mbps, que es un requisito de conectividad para el ecosistema.

Pruebas de Regresión

• Propósito: Asegurar que los cambios o nuevas funcionalidades no han introducido errores en partes existentes del sistema.
• Cuándo se ejecutarán: Después de cada nueva implementación significativa o corrección de errores.
    ◦ El "Núcleo Administrativo Central" o Dashboard de Escritorio es la columna vertebral y la base de datos central de todo el ecosistema, desarrollado en la Fase 1. Cualquier cambio en esta fase fundacional, como la implementación de la "Inteligencia y Automatización Integrada" con Gemma 3 o la "Gestión de Usuarios y Roles", requerirá pruebas de regresión.
    ◦ El proyecto se ejecuta en tres fases secuenciales, donde cada fase construye sobre la anterior. Esto implica que las funcionalidades desarrolladas en fases posteriores (como la Página Web Profesional en Fase 2 o la Aplicación Móvil para Pacientes en Fase 3) que interactúan con el backend centralizado y la base de datos de la Fase 1, necesitarán pruebas de regresión para garantizar la compatibilidad y estabilidad del sistema integral.
    ◦ El "Plan de Mantenimiento y Soporte" mensual de $6,000.00 MXN incluye "actualizaciones de seguridad y la evolución continua de la aplicación". Esta evolución continua, que implica nuevas funcionalidades o mejoras, requerirá un ciclo constante de pruebas de regresión para mantener la integridad del sistema a lo largo del tiempo.

Casos de Prueba Detallados:

1. Pruebas de Login y Acceso

ID: CP-LOGIN-001

• Descripción Breve del Escenario que se Prueba: Inicio de sesión exitoso de un usuario con rol "Admin Principal".
• Precondiciones:
    ◦ La aplicación de escritorio del "Núcleo Administrativo Central" está instalada y ejecutándose.
    ◦ Existe un usuario de prueba en la base de datos con el rol "Admin Principal" y credenciales válidas.
    ◦ Conexión a internet estable de al menos 250 Mbps.
• Pasos Específicos a Seguir para Ejecutar la Prueba:

1. Abrir la aplicación de escritorio.
2. En la "Pantalla: Login de Usuario", ingresar el "Nombre de Usuario" y "Contraseña" válidos del Admin Principal en los campos designados.
3. Hacer clic en el botón "Iniciar Sesión".

• Datos de Entrada de Prueba:
  ◦ Nombre de Usuario: admin.principal
  ◦ Contraseña: PasswordSeguro123
• Resultado Esperado:
  ◦ Las credenciales se envían al backend de forma segura mediante HTTPS (TLS 1.3).
  ◦ El backend verifica las credenciales con la Base de Datos Relacional (Cloud SQL) donde las contraseñas están hasheadas y cifradas en reposo.
  ◦ El "Servicio de Logs de Auditoría" registra un intento de login exitoso.
  ◦ El sistema redirecciona al usuario al "Dashboard Principal (Vista General)".
  ◦ La cabecera del dashboard muestra "Bienvenido, Dr. Admin Principal (Admin)".
  ◦ La barra de navegación lateral presenta todas las secciones, incluyendo "Administración (Gestionar Usuarios, Ver Logs de Auditoría)".
• Criterios de Éxito/Fallo: La prueba pasa si el usuario es redirigido al Dashboard principal con el rol correcto y acceso a todas las funcionalidades esperadas para un "Admin Principal"; de lo contrario, falla.

ID: CP-LOGIN-002
• Descripción Breve del Escenario que se Prueba: Intento de inicio de sesión con credenciales incorrectas.
• Precondiciones:
  ◦ La aplicación de escritorio está instalada y ejecutándose.
  ◦ Conexión a internet estable de al menos 250 Mbps.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
1. Abrir la aplicación de escritorio.
2. En la "Pantalla: Login de Usuario", ingresar un "Nombre de Usuario" existente y una "Contraseña" incorrecta.
3. Hacer clic en el botón "Iniciar Sesión".

• Datos de Entrada de Prueba:
  ◦ Nombre de Usuario: admin.principal
  ◦ Contraseña: ContrasenaIncorrecta
• Resultado Esperado:
  ◦ El backend rechaza las credenciales después de verificarlas con la base de datos.
  ◦ El "Servicio de Logs de Auditoría" registra un intento de login fallido.
  ◦ La aplicación permanece en la "Pantalla: Login de Usuario".
  ◦ Se muestra un mensaje de error claro y conciso: "Usuario o contraseña incorrectos".
• Criterios de Éxito/Fallo: La prueba pasa si la aplicación permanece en la pantalla de Login y muestra el mensaje de error "Usuario o contraseña incorrectos"; de lo contrario, falla.

ID: CP-LOGIN-003
• Descripción Breve del Escenario que se Prueba: Intento de inicio de sesión dejando campos vacíos.
• Precondiciones:
  ◦ La aplicación de escritorio está instalada y ejecutándose.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
1. Abrir la aplicación de escritorio.
2. En la "Pantalla: Login de Usuario", dejar el campo "Nombre de Usuario" vacío y/o el campo "Contraseña" vacío.
3. Hacer clic en el botón "Iniciar Sesión".

• Datos de Entrada de Prueba:
  ◦ Nombre de Usuario: (vacío)
  ◦ Contraseña: (vacío)
• Resultado Esperado:
  ◦ La aplicación permanece en la "Pantalla: Login de Usuario".
  ◦ Se muestran mensajes de validación para los campos requeridos: "Campo 'Nombre de Usuario' requerido" y/o "Campo 'Contraseña' requerido".
  ◦ No se realiza una llamada al backend si la validación es a nivel de frontend.
• Criterios de Éxito/Fallo: La prueba pasa si la aplicación permanece en la pantalla de Login y muestra los mensajes de validación para campos vacíos; de lo contrario, falla.

ID: CP-LOGIN-004
• Descripción Breve del Escenario que se Prueba: Intento de inicio de sesión sin conexión

a internet.
• Precondiciones:
    ◦ La aplicación de escritorio está instalada y ejecutándose.
    ◦ La computadora no tiene conexión a internet.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Desconectar la computadora de internet.
    2. Abrir la aplicación de escritorio.
    3. En la "Pantalla: Login de Usuario", ingresar el "Nombre de Usuario" y "Contraseña" (válidos o inválidos) en los campos designados.
    4. Hacer clic en el botón "Iniciar Sesión".
• Datos de Entrada de Prueba:
    ◦ Nombre de Usuario: cualquier_usuario
    ◦ Contraseña: cualquier_password
• Resultado Esperado:
    ◦ La aplicación intenta comunicarse con el backend en la nube pero falla debido a la falta de conectividad.
    ◦ La aplicación permanece en la "Pantalla: Login de Usuario".
    ◦ Se muestra un mensaje de error general sobre problemas de conexión o indisponibilidad del servicio (ej., "No se pudo conectar al servidor. Por favor, intente de nuevo más tarde.").
• Criterios de Éxito/Fallo: La prueba pasa si la aplicación muestra un mensaje de error de conectividad y permanece en la pantalla de Login; de lo contrario, falla.

2. Pruebas de Gestión Integral del Expediente del Paciente

ID: CP-PACIENTES-001
• Descripción Breve del Escenario que se Prueba: Creación de un nuevo perfil de paciente con datos completos y válidos.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Pacientes" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Nuevo Paciente".
    3. En el "Formulario de Creación de Perfil de Paciente", llenar todos los campos obligatorios y opcionales con datos válidos.
    4. Hacer clic en el botón "Guardar Paciente".
• Datos de Entrada de Prueba:
    ◦ Nombre: Juan
    ◦ Apellido: Pérez García
    ◦ Fecha de Nacimiento: 1985-07-15
    ◦ Teléfono: 5512345678
    ◦ Email: juan.perez@example.com
    ◦ Dirección: Calle Falsa 123
• Resultado Esperado:
    ◦ Se envía una solicitud POST al backend para crear el nuevo paciente.
    ◦ El nuevo perfil de paciente se crea y se guarda en la "Base de Datos Relacional (Cloud SQL)".
    ◦ La tabla "Pacientes Recientes" en el Dashboard Principal se actualiza para incluir al nuevo paciente.
    ◦ El "Servicio de Logs de Auditoría" registra la creación del nuevo paciente.
• Criterios de Éxito/Fallo: La prueba pasa si el paciente se crea correctamente, se visualiza en la tabla de pacientes recientes y se registra la acción en los logs de auditoría; de lo contrario, falla.

ID: CP-PACIENTES-002
• Descripción Breve del Escenario que se Prueba: Intento de creación de un nuevo perfil de paciente con campos obligatorios vacíos.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Pacientes" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Nuevo Paciente".
    3. En el "Formulario de Creación de Perfil de Paciente", dejar el campo "Nombre" vacío y/o el campo "Apellido" vacío.
    4. Hacer clic en el botón "Guardar Paciente".
• Datos de Entrada de Prueba:

    ◦ Nombre: (vacío)
    ◦ Apellido: Pérez García
    ◦ Fecha de Nacimiento: 1985-07-15
    ◦ Teléfono: 5512345678
    ◦ Email: juan.perez@example.com
• Resultado Esperado:
    ◦ La aplicación permanece en el formulario de "Creación de Perfil de Paciente".
    ◦ Se muestra un mensaje de validación para el campo faltante: "Campo 'Nombre' requerido".
    ◦ No se realiza una solicitud de creación al backend.
• Criterios de Éxito/Fallo: La prueba pasa si la aplicación muestra el mensaje de validación y no permite guardar el paciente; de lo contrario, falla.

ID: CP-PACIENTES-003
• Descripción Breve del Escenario que se Prueba: Búsqueda avanzada de un paciente existente por nombre.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal", "Admin Secundario", "Asistente" o "Invitado (Solo Lectura)".
    ◦ Existe al menos un paciente de prueba en la base de datos.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Pacientes" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Buscar Paciente".
    3. En el campo de búsqueda, ingresar el nombre completo o parcial de un paciente existente.
    4. Hacer clic en el botón "Buscar".
• Datos de Entrada de Prueba:
    ◦ Criterio de Búsqueda: Laura Vargas
• Resultado Esperado:
    ◦ Se envía una solicitud GET al backend (ej. GET /patients?query=Laura Vargas).
    ◦ La "Búsqueda de Pacientes" muestra una "Tabla de resultados" con los pacientes que coinciden con el criterio de búsqueda.
    ◦ Las opciones de "Acción" ("Ver Historial", "Nueva Cita") están disponibles para los resultados.
• Criterios de Éxito/Fallo: La prueba pasa si la tabla de resultados muestra correctamente los pacientes coincidentes; de lo contrario, falla.

ID: CP-PACIENTES-004
• Descripción Breve del Escenario que se Prueba: Registro de una nueva entrada en el historial clínico detallado de un paciente.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".
    ◦ Existe un paciente al que se le registrará el historial.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Buscar un paciente existente (ej. CP-PACIENTES-003) y hacer clic en "Ver Historial" desde la tabla de resultados o "Pacientes Recientes".
    2. Desde la "Ficha del Paciente", hacer clic en "Añadir a Historial".
    3. En el "Formulario de Registro de Historial Clínico", completar los campos "Fecha de Consulta", "Diagnóstico" y "Notas de Evolución".
    4. Hacer clic en el botón "Guardar Historial".
• Datos de Entrada de Prueba:
    ◦ Fecha de Consulta: 2024-07-20
    ◦ Diagnóstico: Fractura de peroné
    ◦ Notas de Evolución: Paciente presenta dolor leve, se indica reposo y antiinflamatorios.
• Resultado Esperado:
    ◦ Se envía una solicitud al backend para guardar la nueva entrada del historial.
    ◦ La nueva entrada se guarda cronológicamente en el "Historial Clínico Detallado" del paciente en la "Base de Datos Relacional (Cloud SQL)".
    ◦ El "Servicio de Logs de Auditoría" registra la acción de modificación de datos del paciente.
• Criterios de Éxito/Fallo: La prueba pasa si la nueva entrada del historial clínico aparece correctamente en la ficha del paciente y se registra la acción en los logs de auditoría; de lo contrario, falla.

3. Pruebas de Gestión Documental Avanzada

ID: CP-DOCUMENTOS-001
• Descripción Breve del Escenario que se Prueba: Subida y asociación de un archivo PDF al expediente de un paciente.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".
    ◦ Existe un paciente al que se le adjuntará el documento.
    ◦ Se dispone de un archivo PDF de prueba válido.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la ficha de un paciente existente (ej. tras búsqueda o desde "Pacientes Recientes").
    2. Hacer clic en la opción para "Adjuntar Documentos" (ya sea en el historial clínico o en una sección dedicada de documentos).
    3. En el diálogo de selección de archivos, seleccionar un archivo PDF de prueba.
    4. Confirmar la subida o etiquetar el documento si la interfaz lo requiere.
• Datos de Entrada de Prueba:
    ◦ Archivo: Informe_Radiografia_Juan_Perez.pdf
    ◦ Tamaño: 1.5 MB
• Resultado Esperado:
    ◦ El archivo se sube a "Cloud Storage" y se almacena cifrado en reposo.
    ◦ El documento digital se asocia directamente al expediente del paciente (o a una entrada específica del historial clínico).
    ◦ El "Servicio de Logs de Auditoría" registra la subida del documento.
• Criterios de Éxito/Fallo: La prueba pasa si el PDF se sube, se asocia correctamente al paciente y es visible en su expediente, y la acción se registra en los logs de auditoría; de lo contrario, falla.
ID: CP-DOCUMENTOS-002
• Descripción Breve del Escenario que se Prueba: Intento de subida de un archivo con tipo no soportado.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".
    ◦ Existe un paciente para adjuntar el documento.
    ◦ Se dispone de un archivo de prueba con extensión no soportada (ej. .exe, .zip).
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la ficha de un paciente existente.
    2. Hacer clic en la opción para "Adjuntar Documentos".
    3. En el diálogo de selección de archivos, seleccionar un archivo de tipo no soportado (ej. malware.exe).
    4. Confirmar la subida.
• Datos de Entrada de Prueba:
    ◦ Archivo: documento_malicioso.exe
• Resultado Esperado:
    ◦ La aplicación bloquea la subida en el frontend o el backend la rechaza.
    ◦ Se muestra un mensaje de error: "Tipo de archivo no válido".
    ◦ No se registra ninguna acción de subida exitosa en el "Servicio de Logs de Auditoría" para este intento.
• Criterios de Éxito/Fallo: La prueba pasa si el sistema impide la subida del archivo no soportado y muestra un mensaje de error relevante; de lo contrario, falla.
ID: CP-DOCUMENTOS-003
• Descripción Breve del Escenario que se Prueba: Visualización de un documento digital previamente subido en el expediente del paciente.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal", "Admin Secundario" o "Asistente".
    ◦ Existe un paciente con al menos un documento digital (ej. PDF, imagen, video corto) asociado a su expediente en "Cloud Storage".
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la ficha de un paciente existente.
    2. Acceder a la sección donde se listan los documentos asociados al paciente o al "Visor de Archivos Visuales".
    3. Hacer clic en la miniatura o el nombre de un documento para visualizarlo.
• Datos de Entrada de Prueba: (N/A, se utiliza un documento existente)
• Resultado Esperado:
    ◦ El sistema recupera el archivo desde "Cloud Storage".
    ◦ El documento se abre y visualiza correctamente dentro del visor integrado de la

aplicación o en una ventana emergente.
    ◦ La visualización de imágenes y videos se realiza de manera fluida, sin demoras significativas, considerando la conectividad de al menos 250 Mbps.
• Criterios de Éxito/Fallo: La prueba pasa si el documento se visualiza correctamente y de forma fluida; de lo contrario, falla.

4. Pruebas del Módulo de Agenda y Citas

ID: CP-AGENDA-001
• Descripción Breve del Escenario que se Prueba: Agendamiento de una nueva cita para un paciente.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".
    ◦ Existe un paciente al que se le agendará la cita.
    ◦ El horario seleccionado está disponible.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Agenda" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Agendar Cita".
    3. En el "Formulario de Agendar Nueva Cita", seleccionar un paciente existente, una "Fecha", una "Hora" y un "Tipo de Cita".
    4. Seleccionar el "Estado" de la cita como "Programada".
    5. Hacer clic en el botón "Guardar Cita".
• Datos de Entrada de Prueba:
    ◦ Paciente: Juan Pérez García
    ◦ Fecha: 2024-07-25
    ◦ Hora: 10:00 AM
    ◦ Tipo de Cita: Primera Consulta
    ◦ Estado: Programada
• Resultado Esperado:
    ◦ La nueva cita se guarda en la "Base de Datos Relacional (Cloud SQL)".
    ◦ La cita aparece visible en el "Calendario Interactivo" y en el panel "CITAS DEL DÍA" en el Dashboard Principal.
    ◦ El "Servicio de Logs de Auditoría" registra la creación de la cita.
• Criterios de Éxito/Fallo: La prueba pasa si la cita se crea correctamente, es visible en el calendario y se registra en los logs de auditoría; de lo contrario, falla.

ID: CP-AGENDA-002
• Descripción Breve del Escenario que se Prueba: Intento de agendamiento de una cita en un horario ya ocupado.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".
    ◦ Existe una cita ya agendada para una fecha y hora específica.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Agenda" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Agendar Cita".
    3. En el "Formulario de Agendar Nueva Cita", intentar agendar una cita para la misma "Fecha" y "Hora" de una cita ya existente.
    4. Hacer clic en el botón "Guardar Cita".
• Datos de Entrada de Prueba:
    ◦ Paciente: Nuevo Paciente
    ◦ Fecha: 2024-07-25 (misma que cita existente)
    ◦ Hora: 10:00 AM (misma que cita existente)
    ◦ Tipo de Cita: Seguimiento
• Resultado Esperado:
    ◦ La aplicación impide el agendamiento y permanece en el formulario.
    ◦ Se muestra un mensaje de conflicto: "Horario no disponible".
• Criterios de Éxito/Fallo: La prueba pasa si el sistema detecta el conflicto de horario y muestra el mensaje de error correspondiente; de lo contrario, falla.

ID: CP-AGENDA-003
• Descripción Breve del Escenario que se Prueba: Reprogramación de una cita existente a una nueva fecha y hora.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".
    ◦ Existe una cita de prueba ya agendada y visible en el agenda.
    ◦ La nueva fecha y hora seleccionada están disponibles.
• Pasos Específicos a Seguir para Ejecutar la Prueba:

    1. Navegar a la sección "Agenda" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Ver Calendario".
    3. Seleccionar una cita existente en el calendario y hacer clic en ella para ver los "Detalles de Cita".
    4. Hacer clic en el botón "Reprogramar" o editar los campos de fecha/hora.
    5. Seleccionar una nueva "Fecha" y "Hora".
    6. Guardar los cambios.
• Datos de Entrada de Prueba:
    ◦ Cita Original: 2024-07-25 10:00 AM - Juan Pérez
    ◦ Nueva Fecha: 2024-08-01
    ◦ Nueva Hora: 11:30 AM
• Resultado Esperado:
    ◦ La cita se actualiza en la "Base de Datos Relacional (Cloud SQL)" con la nueva fecha y hora.
    ◦ La cita se muestra en el calendario en su nueva posición y desaparece de la anterior.
    ◦ El "Servicio de Logs de Auditoría" registra la modificación de la cita.
• Criterios de Éxito/Fallo: La prueba pasa si la cita se reprograma correctamente y la acción se registra en los logs de auditoría; de lo contrario, falla.
5. Pruebas de Seguridad, Roles y Accesos Rápidos
ID: CP-SEGURIDAD-001
• Descripción Breve del Escenario que se Prueba: Creación de una nueva cuenta de usuario con rol "Asistente" por un "Admin Principal".
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal".
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Administración" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Gestionar Usuarios".
    3. Hacer clic en el botón "Crear Nuevo Usuario".
    4. Completar los campos de usuario (Nombre de Usuario, Contraseña) y seleccionar "Asistente" del desplegable de roles.
    5. Hacer clic en "Guardar Usuario".
• Datos de Entrada de Prueba:
    ◦ Nombre de Usuario: asistente.prueba
    ◦ Contraseña: PasswordAsistente1!
    ◦ Rol: Asistente
• Resultado Esperado:
    ◦ La nueva cuenta de usuario se crea y se guarda en la "Base de Datos Relacional (Cloud SQL)", con la contraseña hasheada y el rol asignado.
    ◦ El usuario aparece en el listado de usuarios en la pantalla "Gestión de Usuarios".
    ◦ El "Servicio de Logs de Auditoría" registra la creación del nuevo usuario.
• Criterios de Éxito/Fallo: La prueba pasa si el nuevo usuario con rol "Asistente" se crea correctamente y la acción se registra en los logs de auditoría; de lo contrario, falla.
ID: CP-SEGURIDAD-002
• Descripción Breve del Escenario que se Prueba: Intento de acceso a "Gestionar Usuarios" por un usuario con rol "Asistente" (permiso denegado).
• Precondiciones:
    ◦ Usuario logueado con rol "Asistente".
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Administración" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Gestionar Usuarios".
• Datos de Entrada de Prueba: (N/A)
• Resultado Esperado:
    ◦ El sistema deniega el acceso a la pantalla "Gestión de Usuarios".
    ◦ Se muestra un mensaje de "Acceso Denegado" o simplemente la opción no es visible/clicable.
    ◦ El "Servicio de Logs de Auditoría" puede registrar un intento de acceso no autorizado, si la implementación lo contempla a este nivel.
• Criterios de Éxito/Fallo: La prueba pasa si el usuario "Asistente" no puede acceder a la gestión de usuarios y se le notifica la denegación de acceso; de lo contrario, falla.
ID: CP-SEGURIDAD-003
• Descripción Breve del Escenario que se Prueba: Acceso y visualización de los logs de auditoría por un "Admin Principal".

• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal".
    ◦ Existen registros en el "Servicio de Logs de Auditoría".
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Administración" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Ver Logs de Auditoría".
• Datos de Entrada de Prueba: (N/A)
• Resultado Esperado:
    ◦ La aplicación muestra la "Pantalla: Logs de Auditoría" con una "Tabla cronológica" de todas las actividades y cambios realizados en el sistema (ej., creación de pacientes, modificaciones de citas, intentos de login).
    ◦ Los logs son inmutables (Write Once, Read Many - WORM), asegurando su integridad.
• Criterios de Éxito/Fallo: La prueba pasa si el "Admin Principal" puede acceder y visualizar la tabla completa de logs de auditoría sin errores; de lo contrario, falla.

6. Pruebas de Inteligencia y Automatización Integrada (Modelo de IA Gemma 3)

ID: CP-IA-001
• Descripción Breve del Escenario que se Prueba: Interacción con el "Chat Interno Básico" para una consulta rápida.
• Precondiciones:
    ◦ Usuario logueado con cualquier rol.
    ◦ El modelo de IA "Gemma 3" está integrado y operativo en el backend a través de APIs.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Hacer clic en el icono o navegar a la sección dedicada al chat interno en el Dashboard.
    2. En el "Campo de entrada de texto", ingresar una consulta simple.
    3. Hacer clic en "Enviar".
• Datos de Entrada de Prueba:
    ◦ Consulta: Resúmeme el historial de Juan Pérez García.
• Resultado Esperado:
    ◦ El backend envía la consulta al modelo "Gemma 3".
    ◦ El chat interno muestra una respuesta relevante generada por la IA en formato de texto.
    ◦ La interacción es solo de texto y no genera archivos.
• Criterios de Éxito/Fallo: La prueba pasa si la IA responde correctamente a la consulta en formato de texto y no intenta generar un archivo; de lo contrario, falla.

ID: CP-IA-002
• Descripción Breve del Escenario que se Prueba: Uso de la IA para generar un borrador de correo electrónico.
• Precondiciones:
    ◦ Usuario logueado con cualquier rol.
    ◦ El modelo de IA "Gemma 3" está integrado y operativo en el backend a través de APIs.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a un módulo que permita la automatización de la redacción de correos o usar el chat interno.
    2. Ingresar una instrucción para generar un borrador de correo.
    3. Hacer clic en "Enviar" o activar la función de generación.
• Datos de Entrada de Prueba:
    ◦ Instrucción: Redacta un correo de recordatorio de cita para Juan Pérez para el 25 de julio a las 10 AM.
• Resultado Esperado:
    ◦ La IA genera un borrador de correo electrónico basado en la instrucción y lo muestra en la interfaz (ej., en el chat o en un campo de texto de un formulario de correo).
    ◦ El contenido es coherente con la solicitud y en formato de texto.
• Criterios de Éxito/Fallo: La prueba pasa si la IA genera un borrador de correo electrónico relevante y en el formato de texto esperado; de lo contrario, falla.

7. Pruebas de Cerrar Sesión

ID: CP-LOGOUT-001
• Descripción Breve del Escenario que se Prueba: Cierre de sesión exitoso desde el Dashboard Principal.
• Precondiciones:
    ◦ Usuario logueado con cualquier rol en el Dashboard Principal.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. En la "Cabecera (Header)" del Dashboard Principal, hacer clic en la opción "Cerrar

Sesión".
• Datos de Entrada de Prueba: (N/A)
• Resultado Esperado:
    ◦ La sesión del usuario se invalida en el backend.
    ◦ El usuario es redirigido a la "Pantalla: Login de Usuario".
    ◦ El "Servicio de Logs de Auditoría" registra la acción de cierre de sesión.
• Criterios de Éxito/Fallo: La prueba pasa si el usuario es redirigido a la pantalla de Login y su sesión es correctamente terminada; de lo contrario, falla.

5. Pruebas Específicas Importantes (Ejemplos de Enfoque):
• Enfoque en probar a fondo las funcionalidades críticas como:
    ◦ Ciclo completo de gestión de Pacientes (Crear, Ver, Editar, Archivar, Eliminar Suave):
        ▪ Creación de Perfiles: Se debe verificar la correcta creación de perfiles de paciente con todos sus datos demográficos y de contacto. Esto incluye probar la validación de campos obligatorios y formatos incorrectos, como el mensaje "Campo 'Nombre' requerido" o "Formato de email inválido".
        ▪ Visualización: Es crucial probar la capacidad de búsqueda instantánea de pacientes por cualquier campo (nombre, ID, palabra clave en notas) y que el sistema devuelva resultados precisos. También, verificar que el "Dashboard Principal" muestre a los "Pacientes Recientes" en una tabla con la información adecuada y opciones de "Acción" como "Ver Historial" o "Nueva Cita".
        ▪ Edición: Se debe asegurar que los datos de un paciente puedan ser modificados y guardados correctamente, y que estos cambios se reflejen en la base de datos.
        ▪ Registro Historial Clínico Detallado: Probar la adición de nuevas entradas cronológicas en el historial clínico de un paciente, verificando que se incluyan campos como "Fecha de Consulta", "Diagnóstico", "Notas de Evolución" y "Procedimientos y Tratamientos". Cada entrada debe vincularse al paciente específico.
        ▪ Archivar / Eliminar Suave: Un aspecto crítico es la implementación del "borrado suave" (soft delete). Las pruebas deben confirmar que, al seleccionar registros y hacer clic en "Eliminar", los datos no se borran permanentemente de la base de datos, sino que se marcan como inactivos o archivados (ej., is_deleted = true). Esto garantiza que la información no se pierda accidentalmente y se mantenga para auditoría o recuperación futura. Esta funcionalidad aborda la "saturación y dispersión de información" y la "dificultad en el seguimiento eficiente del paciente".
    ◦ Funcionalidades de Búsqueda y Filtrado (probar diferentes criterios, casos borde, búsqueda vacía):
        ▪ Búsqueda Avanzada: Se debe verificar la eficacia de la búsqueda por múltiples criterios (nombre, ID, palabra clave en notas) y a nivel global en la aplicación.
        ▪ Casos Borde y Búsqueda Vacía: Probar búsquedas con caracteres especiales, nombres muy largos/cortos, y especialmente qué ocurre cuando una búsqueda no encuentra resultados, asegurando que se muestre un mensaje claro como "No se encontraron pacientes".
        ▪ Filtros: El sistema ofrece opciones de filtrado avanzado (por fecha, estado, tipo de consulta) y una estructura de carpetas (General, Operaciones, Consultas, Laboratorio) que funciona como filtro. Se debe validar que al aplicar estos filtros, el contenido del área principal se actualice correctamente para mostrar solo los registros pertenecientes a la categoría seleccionada.
    ◦ Gestión Documental (Subir diferentes tipos de archivos, ver archivos, eliminar suave):
        ▪ Subida y Asociación: Probar la capacidad de subir, asociar y visualizar todo tipo de archivos digitales (PDF, imágenes, videos cortos) directamente en el expediente del paciente o en una entrada específica del historial clínico. Es crucial verificar que los archivos se almacenen en Cloud Storage y que estén "cifrados en reposo".
        ▪ Visualización: Comprobar el visor de archivos visuales, asegurando que el carrusel de vistas previas funcione correctamente y que los filtros de origen/tipo (Cloud Drives, Photos, Videos) funcionen como se espera.
        ▪ Manejo de Errores de Subida: Probar escenarios donde se intentan subir tipos de archivo no soportados, archivos que exceden el tamaño máximo, o fallos de conexión, y verificar que los mensajes de error sean claros (ej., "Tipo de archivo no válido", "Error al subir el archivo").
        ▪ Eliminación Suave: Al igual que con los pacientes, se debe verificar que los documentos se marquen como inactivos en lugar de eliminarse físicamente, manteniendo la integridad y seguridad de los datos.

- Gestión de Citas (Crear, Editar, Eliminar, probar con fechas pasadas/futuras):
  - Creación y Agendamiento: Probar la creación de nuevas citas, verificando la correcta selección de paciente, fecha, hora, tipo de cita y estado (Programada, Confirmada, etc.). Es vital probar los escenarios de conflicto de horario para asegurar que se muestre el mensaje "Horario no disponible" si la hora ya está ocupada.
  - Gestión de Citas Existentes: Probar las acciones de reprogramación, cancelación y cambio de estado de las citas (Confirmada, Atendida, Cancelada), asegurando que los cambios se reflejen correctamente en el calendario interactivo.
  - Fechas Pasadas/Futuras: Asegurar que el sistema maneje correctamente la visualización y gestión de citas en el pasado y futuro, incluyendo la visualización de "Citas del Día" en el dashboard. Estas funcionalidades son clave para abordar los "procesos manuales que consumen tiempo" y la "falta de control ágil de visitas diarias".
- Todas las interacciones definidas con el Asistente Virtual (probar diferentes comandos/preguntas, verificar respuestas):
  - Chat Interno: Probar las capacidades de interacción con el chat interno básico, incluyendo la realización de consultas rápidas, obtención de resúmenes de texto y redacción de informes, confirmando que solo funciona con texto y no genera archivos.
  - Automatizaciones Básicas: Validar la redacción automatizada de correos electrónicos, la generación de hojas membretadas y la creación de recordatorios a través de la IA.
  - Búsqueda Inteligente: Probar comandos en lenguaje natural para la búsqueda de archivos o información específica (ej., "Hola, estoy buscando un archivo me urge", "Recuerdas el nombre del paciente?", "Si!! es Laura Vargas").
  - Recuperación y Acciones Contextuales: Verificar que el asistente localice los recursos y ofrezca acciones contextuales como enviar por correo o descargar.
  - Automatización de Tareas: Confirmar que el asistente puede ejecutar acciones conectadas a las APIs del sistema, como enviar correos o generar borradores de documentos.
  - Generación de Contenido para Capacitación: La IA (Gemma 3) puede ayudar a generar preguntas frecuentes (FAQs) para el manual de usuario, lo cual también debe ser probado.
- Pruebas de Seguridad: Intentar acceder a funcionalidades de Administrador con un usuario Asistente, intentar eliminar un registro con borrado fuerte si solo se permite borrado suave, probar inyecciones simples en campos de entrada.
  - Control de Acceso Basado en Roles (RBAC): Realizar pruebas exhaustivas para asegurar que cada rol (Administrador, Admin Principal, Admin Secundario, Asistente, Invitado (Solo Lectura)) solo pueda acceder a las funcionalidades y la información correspondiente a sus permisos. Por ejemplo, un "Asistente" no debería poder gestionar usuarios o ver logs de auditoría. Este es un pilar fundamental para las "preocupaciones de seguridad de datos".
  - Borrado Suave vs. Fuerte: Verificar que, incluso si se intenta una eliminación directa o "fuerte", el sistema aplique consistentemente el "borrado suave".
  - Cifrado de Datos: Aunque las pruebas directas de cifrado pueden requerir acceso a la infraestructura, se debe confirmar que las contraseñas se almacenen "cifradas en reposo" (hashing) y que los "datos sensibles" en la base de datos (Cloud SQL) y el almacenamiento de archivos (Cloud Storage) estén "cifrados en reposo".
  - Cifrado en Tránsito: Asegurar que toda la comunicación entre la aplicación de escritorio y la infraestructura en la nube utilice "HTTPS (TLS 1.3)" para cifrar los datos en tránsito.
  - Logs de Auditoría: Verificar que cada intento de inicio de sesión (exitoso o fallido) y cada acción del usuario (ej., modificación de datos de paciente, creación de citas) se registre "cronológicamente" en el "Servicio de Logs de Auditoría" y que estos logs sean "inmutables (WORM)".
  - Pruebas de Inyección: Aunque no se detalla cómo se realizará, es una buena práctica probar con "inyecciones simples en campos de entrada" para detectar vulnerabilidades comunes.
- Pruebas de Roles y Permisos: Crear matriz de permisos y probar que cada rol solo puede ejecutar las acciones permitidas.
  - Matriz de Permisos: Se debe desarrollar una matriz detallada que especifique qué acciones y secciones son accesibles para cada rol: "Admin Principal" (control total, gestión de usuarios, ver logs), "Admin Secundario" (todo excepto gestión de usuarios y logs), "Asistente" (gestión de pacientes, citas, documentos, reportes), y "Invitado (Solo Lectura)" (ver pacientes, citas, reportes).
  - Validación de Acciones: Ejecutar pruebas con usuarios de cada rol para confirmar

que solo pueden realizar las acciones definidas por sus permisos y que se les niegue el acceso a funcionalidades restringidas, mostrando mensajes de error apropiados si intentan acceder a ellas.
    - Pruebas de Borrado Suave: Verificar en la base de datos que los registros se marcan como inactivos y no se eliminan físicamente.
        - Inspección de Base de Datos: Después de realizar operaciones de "eliminación" desde la interfaz de usuario, se debe acceder directamente a la base de datos (Cloud SQL) para confirmar que los registros correspondientes no han sido eliminados físicamente, sino que su estado ha cambiado a "inactivo" o que se ha establecido un indicador como is_deleted = true. Esto es crucial para la integridad de los datos y el cumplimiento de las políticas de seguridad.

Criterios de Entrada y Salida de las Pruebas para la Fase 1: El Núcleo Administrativo (Dashboard de Escritorio)

La Fase 1 se enfoca en la construcción del "Núcleo Administrativo (Dashboard de Escritorio)", que es la columna vertebral y base de datos central del ecosistema. Su objetivo es asegurar la integridad y seguridad de la información desde el primer día. Las pruebas de esta fase son cruciales para validar las funcionalidades clave antes de la entrega.

Criterios de Entrada (Qué debe estar listo antes de comenzar las pruebas)

Para iniciar las pruebas de la Fase 1 del Dashboard de Escritorio, las siguientes condiciones deben cumplirse:

1. Código Implementado para Todas las Funcionalidades de la Fase 1:
    - La interfaz de usuario (frontend) de la aplicación de escritorio debe estar completamente desarrollada utilizando React, Vue.js o un framework similar, y empaquetada con Electron.js para compatibilidad con Windows/macOS.
    - Toda la lógica de negocio y las APIs (backend) para las funcionalidades de la Fase 1 deben estar codificadas, utilizando Python (Flask/Django) o Node.js (Express).
    - Esto incluye el desarrollo completo de la Gestión Integral del Expediente del Paciente (creación, búsqueda, historial clínico detallado), la Gestión Documental Avanzada (subida, asociación y visualización de archivos), el Módulo de Agenda y Citas (programación y gestión de citas), la Seguridad, Roles y Accesos Rápidos (login, roles), y la Inteligencia y Automatización Integrada (chat interno, automatizaciones básicas de IA).
2. Entorno de Pruebas Configurado y Desplegado:
    - La infraestructura en la nube en Google Cloud Platform (GCP) debe estar completamente configurada y desplegada. Esto incluye:
        - Virtual Private Cloud (VPC) lógicamente aislada y segura.
        - API Gateway / Load Balancer como puntos de entrada seguros y distribuidores de tráfico para el backend.
        - Servidor de Aplicación (Backend) desplegado en servicios como Compute Engine o Cloud Run.
        - Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL) para almacenar información estructurada.
        - Almacenamiento de Archivos (Cloud Storage) para documentos digitales.
        - Servicio de Logs de Auditoría (Cloud Logging / Audit Logs) configurado para registrar cronológicamente actividades.
    - La conectividad entre la aplicación de escritorio y la infraestructura en la nube debe estar establecida a través de HTTPS (TLS 1.3).
    - Se recomienda una conexión a internet estable de al menos 250 Mbps para un rendimiento óptimo del ecosistema.
3. Datos de Prueba Disponibles y Seguros:
    - El esquema de la base de datos para USUARIOS, PACIENTES, HISTORIAL_CLINICO, CITAS, DOCUMENTOS, ROLES, y LOGS_AUDITORIA debe estar completamente definido e implementado.
    - Los datos sensibles, como contraseñas, deben estar cifrados en reposo en la base de datos (mediante hashing) y en Cloud Storage.
    - Se deben haber cargado datos de prueba representativos para cada módulo funcional de la Fase 1 (ej., perfiles de pacientes, historiales clínicos, documentos, citas programadas, usuarios con diferentes roles).
4. Integración con IA Funcional:
    - El modelo de Inteligencia Artificial Gemma 3 (o similar a Gemini) debe estar integrado directamente en el backend mediante APIs, listo para potenciar las funcionalidades de automatización y el chat interno.
5. Especificaciones de Diseño e Interfaz Listas:
    - Todos los elementos visuales del diseño "Glassmorphism Oscuro" (fondo gris carbón

suave, auras de color azul oceánico y violeta, paneles flotantes de vidrio esmerilado, texto nítido y brillante con la fuente Inter) deben estar implementados según las especificaciones técnicas para la interfaz de usuario del Dashboard y el Login. Esto asegura una coherencia visual integral y comodidad para el usuario.

Criterios de Salida (Qué condiciones deben cumplirse para considerar que las pruebas de la Fase 1 han finalizado con éxito)

Para considerar que las pruebas de la Fase 1 han finalizado con éxito, la aplicación de escritorio debe cumplir con las siguientes condiciones:

1. Funcionalidad Completa y Correcta:
    - Todas las funcionalidades de la Fase 1 deben operar como se espera y sin errores significativos, abarcando:
        - Gestión Integral del Expediente del Paciente: Creación, búsqueda avanzada, y registro detallado del historial clínico (consultas, diagnósticos, notas, procedimientos, tratamientos).
        - Gestión Documental Avanzada: Capacidad de subir, asociar y visualizar archivos digitales (PDF, imágenes, videos cortos) al expediente del paciente.
        - Módulo de Agenda y Citas: Programación, reprogramación y gestión de citas con estados personalizables, y visualización efectiva en el calendario.
        - Seguridad, Roles y Accesos Rápidos: El sistema de login debe autenticar correctamente a los usuarios, aplicar el control de acceso basado en roles (Admin Principal, Admin Secundario, Asistente, Invitado), y gestionar usuarios y roles de forma segura.
        - Inteligencia y Automatización Integrada: El chat interno con Gemma 3 debe responder a consultas rápidas, generar resúmenes de texto y permitir la redacción de correos, hojas membretadas y recordatorios.
    - Los widgets y accesos directos a servicios externos como Gmail y Google Drive deben funcionar correctamente.
2. Rendimiento y Estabilidad:
    - El sistema debe demostrar un rendimiento óptimo en las operaciones críticas, como la búsqueda de pacientes o la carga de historiales clínicos, con tiempos de respuesta aceptables.
    - La aplicación debe ser estable y no presentar cierres inesperados, bloqueos o degradación de rendimiento durante el uso continuo.
3. Seguridad y Privacidad de Datos:
    - La autenticación y autorización basada en roles debe funcionar sin fallos, impidiendo el acceso indebido a información o funcionalidades.
    - El cifrado de datos en tránsito (HTTPS/TLS 1.3) y en reposo (Cloud SQL, Cloud Storage) debe ser verificado como completamente funcional, protegiendo la información sensible del paciente.
    - El Servicio de Logs de Auditoría debe registrar de forma inmutable (WORM) todos los intentos de login (exitosos y fallidos), así como las acciones relevantes de los usuarios (ej., modificación de datos de paciente, creación de citas), y estos logs deben ser accesibles para revisión.
4. Manejo Robusto de Errores y Mensajes:
    - Todos los escenarios de error previstos (ej., credenciales incorrectas, campos vacíos/formato inválido, problemas de conectividad) deben mostrar mensajes claros y concisos al usuario, permitiéndole entender el problema y tomar la acción adecuada.
    - Los "estados vacíos" (ej., "No se encontraron pacientes", "No hay citas programadas para hoy") deben ser manejados con mensajes informativos en lugar de espacios en blanco.
5. Coherencia de la Interfaz de Usuario (UI/UX) y Accesibilidad:
    - El diseño "Glassmorphism Oscuro" debe aplicarse de manera consistente en toda la aplicación de escritorio, manteniendo la coherencia visual, la comodidad visual, y una jerarquía de información clara.
    - La legibilidad del texto y los iconos debe ser óptima (cumpliendo con una relación de contraste superior a 4.5:1 WCAG).
    - La retroalimentación interactiva (ej., aumento de brillo o hundimiento al pulsar botones) debe ser clara y consistente.
6. Preparación para Capacitación y Migración de Datos:
    - La aplicación debe ser suficientemente estable para la sesión de capacitación inicial del Dr. Sánchez y su equipo.
    - Los manuales de funcionamiento deben estar preparados, posiblemente con la ayuda de la IA para generar FAQs.
    - La migración inicial de archivos digitales existentes del consultorio al nuevo

sistema debe haberse completado con éxito.
7. Resolución de Defectos:
    ◦ Todos los defectos de prioridad "Crítica" y "Mayor" identificados durante las pruebas deben haber sido resueltos y verificados.
Recursos para las Pruebas: Ecosistema Digital Inteligente para Cirugía Especial (Fase 1: El Núcleo Administrativo)
La Fase 1 se enfoca en la construcción del "Núcleo Administrativo (Dashboard de Escritorio)", que es la columna vertebral y base de datos central del ecosistema. Las pruebas de esta fase son cruciales para validar las funcionalidades clave antes de la entrega.
1. Personal de Pruebas
El "equipo de desarrollo" o "Su Nombre/Nombre de su Empresa" es el encargado de diseñar, construir e implementar el ecosistema. Este mismo equipo es responsable de "Realizar pruebas exhaustivas de todas las funcionalidades, asegurando la seguridad y el rendimiento". No se especifica la existencia de un equipo de pruebas independiente; por lo tanto, las actividades de prueba recaen en el equipo de desarrollo. El Dr. Joel Sánchez García y su equipo son los usuarios clave y beneficiarios finales del ecosistema.
2. Entorno de Pruebas (Configuración)
El entorno de pruebas será una réplica de la infraestructura de producción, basada en Google Cloud Platform (GCP). Los componentes clave de este entorno en la nube incluyen:
• Virtual Private Cloud (VPC): Una sección lógicamente aislada dentro de la nube pública de GCP donde se despliegan todos los recursos del sistema, creando un entorno de red privado y seguro.
• API Gateway / Load Balancer: Actúan como los puntos de entrada seguros para todas las solicitudes provenientes de las aplicaciones cliente, distribuyendo el tráfico de manera eficiente y garantizando la disponibilidad y seguridad del sistema.
• Servidor de Aplicación (Backend): Donde reside la lógica de negocio central y las APIs, construido con Python (Flask/Django) o Node.js (Express), y desplegado utilizando servicios de GCP como Compute Engine o Cloud Run. La integración del modelo de IA Gemma 3 también reside aquí.
• Base de Datos Relacional (Cloud SQL): Almacena toda la información estructurada y crítica, como perfiles de pacientes, historiales clínicos, citas, usuarios y roles, utilizando MySQL o PostgreSQL. Los datos sensibles estarán cifrados en reposo.
• Almacenamiento de Archivos (Cloud Storage): Utilizado para el almacenamiento centralizado y seguro de archivos digitales asociados a los expedientes de los pacientes (PDFs, imágenes, videos cortos). Estos archivos también estarán cifrados en reposo.
• Servicio de Logs de Auditoría (Cloud Logging / Audit Logs): Vital para el monitoreo de la seguridad y el seguimiento de operaciones, registrando cronológicamente todas las actividades y cambios realizados; estos logs son inmutables (Write Once, Read Many - WORM).
• Conectividad: La comunicación entre la aplicación de escritorio y la infraestructura en la nube se realizará a través de HTTPS (TLS 1.3), cifrando los datos en tránsito. Se recomienda una conexión a internet estable de al menos 250 Mbps para un rendimiento óptimo del ecosistema.
• Aplicación de Escritorio: El frontend de la aplicación de escritorio se desarrollará con React, Vue.js o similar, y se empaquetará con Electron.js para compatibilidad con Windows/macOS.
3. Herramientas de Prueba
La documentación no especifica herramientas de prueba automatizadas (ej., para pruebas unitarias, de integración o funcionales). Sin embargo, las herramientas de desarrollo y configuración mencionadas, que facilitan un desarrollo "controlado y de alta calidad", pueden contribuir indirectamente a la capacidad de prueba:
• Herramientas de Diseño (UI/UX): Figma y Adobe Suite (XD, Photoshop, Illustrator) se utilizan para definir flujos de usuario, wireframes y prototipos. Aunque no son herramientas de prueba directa, una UI/UX bien definida y prototipada facilita las pruebas de usabilidad y funcionalidad desde etapas tempranas.
• Herramientas de Modelado de Datos: MySQL o PostgreSQL como sistemas de gestión de bases de datos, y herramientas de modelado ERD como Lucidchart, dbdiagram.io o Google Drawings. Un modelo de datos bien estructurado y documentado es esencial para el diseño de pruebas de integración y base de datos.
• Herramientas de Desarrollo Backend: Python (Flask/Django) o Node.js (Express) para el backend.
• Herramientas de Infraestructura como Código (IaC): Terraform o Cloud Deployment Manager

se mencionan para automatizar la creación y configuración de recursos en GCP. Esto permite la provisión de entornos de prueba consistentes y replicables.
• Herramientas de Desarrollo Frontend: React, Vue.js o similar, y Electron.js para la aplicación de escritorio.
• Herramientas de Colaboración: Google Workspace (Docs/Sheets) para la creación de manuales de usuario colaborativos. La IA (Gemma 3) puede ayudar a generar preguntas frecuentes (FAQs) para el manual de usuario, lo que podría asistir en la creación de casos de prueba o guías de capacitación.
4. Datos de Prueba
La disponibilidad de datos de prueba es un criterio de entrada clave para iniciar las pruebas de la Fase 1.
• Datos de Prueba Representativos: Se deben haber cargado datos de prueba representativos para cada módulo funcional de la Fase 1. Esto incluye, por ejemplo:
    ◦ Perfiles de pacientes.
    ◦ Historiales clínicos.
    ◦ Documentos.
    ◦ Citas programadas.
    ◦ Usuarios con diferentes roles.
• Esquema de la Base de Datos: El esquema de la base de datos para USUARIOS, PACIENTES, HISTORIAL_CLINICO, CITAS, DOCUMENTOS, ROLES, y LOGS_AUDITORIA debe estar completamente definido e implementado.
• Datos Sensibles: Los datos sensibles, como contraseñas, deben estar cifrados en reposo en la base de datos (mediante hashing) y en Cloud Storage. El campo hash_password en la tabla USUARIOS indica el uso de funciones de hash para almacenar contraseñas de forma segura.
• Migración de Datos: La "migración inicial de archivos digitales existentes del consultorio al nuevo sistema" se realiza en la Fase 1. Esto sugiere que datos reales o muy realistas serán utilizados, lo cual es crucial para pruebas de volumen y rendimiento.
Proceso de Gestión de Defectos (Bugs)
La detección y resolución de bugs son cruciales para asegurar la "integridad y seguridad de la información" y la "eficiencia operativa". El ecosistema está diseñado para ser "potente, seguro y excepcionalmente intuitivo", y el manejo de defectos contribuye a mantener estos atributos.
1. Detección y Reporte de Bugs
• Durante el Desarrollo y Pruebas Internas (Fase 1 y subsiguientes): El "equipo de desarrollo" o "Su Nombre/Nombre de su Empresa" es directamente responsable de "Realizar pruebas exhaustivas de todas las funcionalidades, asegurando la seguridad y el rendimiento" [Recursos para las Pruebas: 1, Plan de Acción Detallado: Paso 5]. Durante estas pruebas, los desarrolladores y el equipo de QA (Control de Calidad, implícito en "pruebas exhaustivas") identificarán los bugs. La información de los fallos y comportamientos inesperados se documentará internamente para su posterior análisis.
• Post-Implementación (Soporte al Cliente): Una vez que las fases del proyecto son entregadas, el Dr. Joel Sánchez García y su equipo ("usuarios clave" [Recursos para las Pruebas: 1]) serán los principales reportadores de bugs o incidentes. El "Plan de Mantenimiento y Soporte del Ecosistema" incluye "soporte técnico 24/7 (remoto y presencial según diagnóstico)" [Plan de Trabajo: 6, Preguntas Frecuentes: 7, Línea de Tiempo: Después de la Finalización del Proyecto y la Fase 1]. Esto implica que cualquier error o comportamiento anómalo que detecten en la operación diaria será reportado directamente al equipo de "Su Nombre/Nombre de su Empresa" a través de los canales de soporte que se establezcan (aunque no se especifican los canales exactos, se infiere una comunicación directa y constante).
2. Priorización de Bugs
Las fuentes no especifican una matriz de priorización formal (ej., crítico, mayor, medio, menor), sin embargo, los objetivos y soluciones del ecosistema guían la prioridad:
• Alta Prioridad: Los defectos que comprometan la "seguridad de datos" [Acceso Seguro: 1, 6, 14-16, 18, 23-29; Arquitectura Nube: 9-14; Dashboard: I, II, 4, 6, 9; Ecosistema Digital: Arquitectura y Diseño Glassmorphism Oscuro: Introducción, Pilar 1, Consideraciones Clave; Guía de Estudio: I, II, III, IV, VI, VII, VIII; Plan de Acción Detallado: FASE 1, IMPACTO; Pantalla #1 Login: 1, 3, 7; Pantalla #2 Home: 1, 5, 8; Pantalla #Login: 1, 3, 7], impidan el acceso al sistema (como un fallo en el login [Acceso Seguro: 9, Pantalla #1 Login: 3, Pantalla #2 Home: 6, Pantalla #Login: 3]), o detengan funcionalidades críticas para la "eficiencia operativa" [Automatización e Inteligencia: 1; Dashboard: I, II; Ecosistema Digital: Arquitectura y Diseño Glassmorphism Oscuro:

Introducción, Pilar 1; Arquitectura Nube: 2, 3; Guía de Estudio: I, IV, VI; Plan de Acción Detallado: IMPACTO; Pantalla #2 Home: 1, 2, 8] (ej. gestión de expedientes, agendamiento de citas) serían tratados con la máxima urgencia. La "actualización de seguridad" es parte del mantenimiento continuo [Plan de Trabajo: 6, Preguntas Frecuentes: 7, Línea de Tiempo: Después de la Finalización del Proyecto y la Fase 1].
• Prioridad Media a Baja: Bugs que afecten la usabilidad sin impedir la funcionalidad, o que generen errores menores, serían priorizados según su impacto en el flujo de trabajo y la cantidad de usuarios afectados. Los mensajes de error definidos para diversos escenarios (ej., "Usuario o contraseña incorrectos", "Campo 'Nombre' requerido", "Problemas de Conectividad") [Acceso Seguro: 9, Mapa Detallado: 1. Flujo de Acceso, Pantalla #1 Login: 3, Pantalla #2 Home: 6, Pantalla #Login: 3] sugieren que la claridad de la retroalimentación al usuario es importante, por lo que los defectos en el manejo de errores también serían abordados.
3. Asignación de Bugs
• La responsabilidad de la investigación y resolución de los bugs recae directamente en "Su Nombre/Nombre de su Empresa" [Línea de Tiempo: Elenco de Personajes, Su Nombre/Nombre de su Empresa], el equipo de desarrollo encargado de la "diseño, construcción e implementación" del ecosistema [Recursos para las Pruebas: 1, Plan de Acción Detallado: Situación] y del "mantenimiento y soporte continuo" [Línea de Tiempo: Después de la Finalización del Proyecto y la Fase 1].
4. Seguimiento y Resolución de Bugs
• Registro y Seguimiento (Implícito): Aunque no se especifica una herramienta de seguimiento de bugs dedicada (como Jira o Trello), el uso de "Google Workspace (Docs/Sheets)" para "crear manuales de usuario colaborativos" [Recursos para las Pruebas: 3, Plan de Acción Detallado: Paso 5] sugiere que se podrían emplear herramientas colaborativas simples para registrar y seguir los bugs internamente. Cada bug reportado sería documentado con detalles como la descripción, pasos para reproducir, fecha de reporte, estado (abierto, en progreso, resuelto, cerrado), y la persona asignada.
• Análisis con Logs de Auditoría: Una herramienta técnica fundamental para el seguimiento y diagnóstico de bugs es el "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)" [Acceso Seguro: 7, Arquitectura Nube: 6, Dashboard: 44, Pantalla #1 Login: 5, Pantalla #2 Home: 5, Pantalla #Login: 5, Ecosistema Digital: Arquitectura y Diseño Glassmorphism Oscuro: 6, Plan de Acción Detallado: Paso 3]. Estos logs registran "cronológicamente todas las actividades y cambios realizados" [Acceso Seguro: 7, Arquitectura Nube: 6, Plan de Acción Detallado: Paso 3] y son "inmutables (Write Once, Read Many - WORM)" [Acceso Seguro: 7, Arquitectura Nube: 6, Pantalla #1 Login: 5, Pantalla #2 Home: 5, Pantalla #Login: 5]. Esta capacidad permite al equipo de desarrollo rastrear la secuencia de eventos que llevó al bug, entender su contexto y verificar la correcta aplicación de las soluciones.
• Proceso de Resolución: Una vez diagnosticado, el equipo de desarrollo trabajará en la solución, la implementará, y realizará pruebas internas para asegurar que el bug ha sido corregido y que no se introducen nuevas regresiones. Las "actualizaciones de seguridad y evolución continua" [Plan de Trabajo: 6, Preguntas Frecuentes: 7, Línea de Tiempo: Después de la Finalización del Proyecto y la Fase 1] garantizan que estas correcciones sean parte del ciclo de vida del software.
• Comunicación Continua: El "soporte técnico 24/7" [Plan de Trabajo: 6, Preguntas Frecuentes: 7, Línea de Tiempo: Después de la Finalización del Proyecto y la Fase 1] implica que se mantendrá una comunicación constante con el Dr. Sánchez y su equipo sobre el progreso y la resolución de los problemas reportados, cerrando el ciclo de retroalimentación.
Plan de Ejecución de Pruebas
1. Pruebas Unitarias y de Integración (Durante el Desarrollo)
• Cuándo: Estas pruebas se ejecutarán de forma continua por el equipo de desarrollo de "Su Nombre/Nombre de su Empresa" durante la construcción de cada componente del sistema, tanto en el backend como en el frontend. Esto es fundamental en la Fase 1: El Núcleo Administrativo (Dashboard de Escritorio), que es la columna vertebral del ecosistema.
• Cómo:
    ◦ Pruebas Unitarias: Los desarrolladores verificarán individualmente el correcto funcionamiento de cada unidad de código (funciones, métodos, clases). Esto aplica a la lógica de negocio del Servidor de Aplicación (Backend), construido con Python/Flask/Django o Node.js/Express, y a los componentes de la interfaz de usuario del Dashboard de Escritorio, desarrollado con React, Vue.js o similar.
    ◦ Pruebas de Integración: Se probará la interacción entre diferentes módulos del

sistema y con los servicios de la nube. Esto incluye la comunicación segura mediante HTTPS (TLS 1.3) entre la aplicación de escritorio y el backend en Google Cloud Platform (GCP), la correcta interacción del backend con la Base de Datos Relacional (Cloud SQL) y el Almacenamiento de Archivos (Cloud Storage), y la integración del modelo de IA Gemma 3.
    ◦ El objetivo es asegurar un "desarrollo controlado y de alta calidad" desde las primeras etapas.

2. Pruebas Funcionales y de UI (Al Finalizar Funcionalidades)

• Cuándo: Estas pruebas se realizarán una vez que las funcionalidades clave del sistema estén desarrolladas y listas para una verificación más amplia. Se llevarán a cabo principalmente al concluir cada fase del proyecto, con un enfoque intensivo en el Paso 5: Pruebas y Capacitación de la Fase 1, antes de la entrega.
• Cómo:
    ◦ Pruebas Funcionales: Se validará que cada funcionalidad del Dashboard de Escritorio cumple con los requisitos definidos. Por ejemplo, se probará la creación de perfiles de paciente, el registro detallado del historial clínico, la subida y asociación de documentos digitales, la programación y gestión de citas, el correcto funcionamiento del sistema de roles y accesos, y las automatizaciones y el chat interno con IA. Se verificarán los flujos de usuario detallados, como añadir un nuevo paciente o buscar el historial.
    ◦ Pruebas de UI (Interfaz de Usuario): Se asegurará que el diseño "Glassmorphism Oscuro" se implementa correctamente, garantizando la "comodidad visual y una jerarquía de información clara". Esto incluye la legibilidad de textos y iconos, la aplicación de colores y tipografías, y la retroalimentación interactiva al pulsar botones (sutil aumento de brillo o ligero hundimiento).
    ◦ Se comprobará el manejo de errores para escenarios como credenciales incorrectas, campos vacíos o problemas de conectividad, asegurando que los mensajes de error sean claros y concisos.
    ◦ Las pruebas se realizarán tanto por el equipo de desarrollo como, de forma colaborativa, durante la Capacitación Inicial con el Dr. Sánchez y su equipo, quienes utilizarán el sistema en un entorno controlado. Se utilizarán herramientas como Google Workspace (Docs/Sheets) para documentar los hallazgos.

3. Pruebas de Seguridad (Antes del Despliegue)

• Cuándo: Las pruebas de seguridad son de máxima prioridad y se ejecutarán rigurosamente antes del despliegue de cada fase, especialmente de la Fase 1, dado que el Núcleo Administrativo Central maneja "información sensible".
• Cómo:
    ◦ Control de Acceso Basado en Roles (RBAC): Se validará que el sistema de roles ("Admin Principal", "Admin Secundario", "Asistente", "Invitado (Solo Lectura)") funcione correctamente, impidiendo el acceso no autorizado a funcionalidades o datos sensibles.
    ◦ Cifrado de Datos: Se verificará que el cifrado de datos en tránsito (mediante HTTPS/TLS 1.3) y el cifrado en reposo de los datos sensibles en la Base de Datos (Cloud SQL) y el Almacenamiento de Archivos (Cloud Storage) se apliquen de forma efectiva.
    ◦ Logs de Auditoría: Se confirmará que el Servicio de Logs de Auditoría (Cloud Logging / Audit Logs) registra cronológicamente todos los intentos de login (exitosos y fallidos) y cada acción del usuario (ej., modificación de datos de paciente), y que estos logs son inmutables (Write Once, Read Many - WORM), crucial para la detección de anomalías y la respuesta a incidentes.
    ◦ Infraestructura de Red: Se asegurará que la Virtual Private Cloud (VPC) y el API Gateway / Load Balancer funcionen como la primera línea de defensa, aislando y protegiendo la infraestructura en la nube.

4. Pruebas de Rendimiento (Antes del Despliegue)

• Cuándo: Se realizarán pruebas de rendimiento antes del despliegue de cada fase para garantizar la "eficiencia operativa" y la escalabilidad del sistema.
• Cómo:
    ◦ Escalabilidad y Carga: Se someterá el sistema a pruebas de carga para asegurar que la infraestructura en la nube de GCP puede escalar automáticamente y manejar un mayor volumen de usuarios y datos sin degradación del rendimiento.
    ◦ Tiempos de Respuesta: Se medirán los tiempos de respuesta para las operaciones críticas (ej., búsqueda de pacientes, carga de historiales, agendamiento de citas) para asegurar una experiencia de usuario fluida.
    ◦ Conectividad: Se validará que el ecosistema funciona de manera óptima con una conexión a internet estable de al menos 250 Mbps, que es la velocidad recomendada para un rendimiento óptimo.

Especificación de la Integración de Inteligencia Artificial en el Dashboard de Administración

1. Introducción y Propósito

Este documento tiene como objetivo principal detallar la integración técnica y funcional del modelo de Inteligencia Artificial (IA) en el "Núcleo Administrativo Central" o Dashboard de Escritorio del "Ecosistema Digital Inteligente para Cirugía Especial". Su propósito fundamental es ofrecer una guía exhaustiva para los equipos de diseño y desarrollo, asegurando una implementación coherente y de alta calidad de las funcionalidades impulsadas por IA.

La integración de la inteligencia artificial busca potenciar el flujo de trabajo, la automatización de tareas repetitivas, y ofrecer asistencia rápida al personal. Esto se traduce en una optimización significativa de la "eficiencia operativa", reduciendo los procesos manuales y el riesgo de errores, lo que a su vez libera tiempo valioso para el equipo del consultorio. La IA centraliza la inteligencia del sistema para asegurar que la lógica de negocio se ejecute de manera segura y escalable.

Esta funcionalidad clave, denominada "Inteligencia y Automatización Integrada", es un componente fundamental desarrollado en la Fase 1: El Núcleo Administrativo (Dashboard de Escritorio) del proyecto.

2. Modelo de Inteligencia Artificial

El modelo de Inteligencia Artificial a utilizar en este ecosistema es Gemma 3 o uno similar a Gemini. Este modelo se integrará directamente en el Servidor de Aplicación (Backend) mediante el uso de APIs (Interfaces de Programación de Aplicaciones). Esto significa que el backend actuará como intermediario entre la interfaz de usuario del Dashboard y el modelo de IA.

Las capacidades relevantes del modelo Gemma 3 (o similar) para este proyecto incluyen, pero no se limitan a:

• Generación de Texto y Automatizaciones Básicas: El modelo tiene la capacidad de generar texto coherente y relevante para diversas tareas, permitiendo la redacción automatizada de correos electrónicos, la generación de hojas membretadas y la creación de recordatorios. Estas funciones se activan desde módulos específicos de la aplicación o pueden ser invocadas a través del chat interno.

• Resumen y Comprensión de Lenguaje Natural (Chat Interno Básico): El modelo potenciará un chat interno dentro del Dashboard donde el personal puede interactuar con la IA. A través de este chat, los usuarios podrán realizar consultas rápidas, obtener resúmenes de texto complejos o facilitar la redacción de informes. Es importante señalar que este chat funciona únicamente con texto y no tiene la capacidad de generar archivos de ningún formato.

• Asistencia para el Desarrollo y Capacitación: La IA también puede ser utilizada para generar texto de ejemplo (denominado "lorem ipsum más inteligente") para prototipos de interfaz de usuario (UI). Asimismo, puede sugerir "copys" para botones y etiquetas si se le entrena con el tono deseado. Desde una perspectiva de desarrollo, la IA puede generar scripts SQL básicos a partir de descripciones en lenguaje natural. Para la capacitación del personal, puede colaborar en la generación de preguntas frecuentes (FAQs) a partir de la documentación técnica para el manual de usuario, e incluso crear un chatbot de ayuda inicial.

• Procesamiento de Solicitudes y Asistencia Rápida: De manera general, el modelo de IA está diseñado para procesar solicitudes en lenguaje natural y proporcionar asistencia rápida, optimizando el flujo de trabajo y la eficiencia general del sistema.

La elección de Gemma 3 (o similar) subraya el compromiso del proyecto con la innovación y la automatización para resolver el "dolor" de los "procesos manuales que consumen tiempo".

Especificación de la Integración de Inteligencia Artificial en el Dashboard de Administración

3. Casos de Uso de IA en el Dashboard (Funcionalidades Detalladas):

La integración del modelo de Inteligencia Artificial Gemma 3 (o similar a Gemini) en el backend del Dashboard de Administración es un pilar fundamental para potenciar la "Inteligencia y Automatización Integrada", resolviendo el "dolor" de los "procesos manuales que consumen tiempo" y optimizando la "eficiencia operativa". Las funcionalidades que hacen uso de la IA se activarán de manera intuitiva desde módulos específicos de la aplicación o mediante el chat interno.

A continuación, se describen en detalle los casos de uso específicos de la IA:

3.1. Generación de Borradores de Correo Electrónico:

• Propósito: Esta funcionalidad tiene como objetivo principal acelerar y estandarizar la comunicación escrita con los pacientes o terceros, permitiendo al personal del consultorio redactar correos electrónicos de manera rápida y eficiente. Al automatizar la generación de borradores, se reduce el tiempo dedicado a tareas repetitivas y se minimiza el riesgo de errores u omisiones en mensajes importantes.
• Entrada para la IA: La IA recibirá información contextual directamente del sistema, así como datos específicos proporcionados por el usuario para generar un borrador relevante:
    ◦ Datos Contextuales del Sistema: El usuario seleccionará un paciente o una cita desde el Dashboard. El sistema proporcionará automáticamente a la IA el ID del Paciente, el nombre completo del paciente, datos de contacto (ej., email), fechas y detalles de la última consulta o próxima cita y diagnósticos relevantes asociados al expediente.
    ◦ Entrada de Usuario para la Personalización: El personal deberá especificar a la IA:
        ▪ Motivo del Correo: (ej., recordatorio de cita, seguimiento post-operatorio, notificación de resultados de laboratorio, solicitud de información adicional, confirmación de cita).
        ▪ Puntos Clave a Incluir: Información adicional específica que el usuario desea que se incorpore en el cuerpo del correo (ej., instrucciones pre/post-operatorias, medicamentos, fechas de próxima revisión, requisitos de preparación para estudios).
        ▪ Tono Deseado: (ej., formal, empático, conciso).
• Salida Esperada de la IA: La IA generará un borrador de texto para un correo electrónico, estructurado y coherente, que contendrá:
    ◦ Un asunto sugerido para el correo.
    ◦ Un saludo personalizado dirigido al paciente (o destinatario si no es un paciente).
    ◦ El cuerpo del mensaje que integre el motivo, los puntos clave proporcionados y la información contextual del paciente.
    ◦ Un cierre profesional con la información del consultorio y del Dr. Sánchez.
    ◦ Es importante recalcar que la IA generará únicamente el texto del borrador y no producirá un archivo de correo electrónico final ni lo enviará directamente. El borrador de texto será presentado en la interfaz del Dashboard para que el usuario pueda revisarlo, editarlo y luego copiarlo o integrarlo en su cliente de correo electrónico (ej., Gmail, accesible vía widget).
• Cómo se Iniciará esta Función en la Interfaz:
    ◦ Desde un Módulo Específico: Se habilitará un botón o una opción de "Generar Borrador de Correo" en la vista de detalle de cada expediente del paciente, en el módulo de Agenda y Citas (al seleccionar una cita) o en una sección dedicada a "Automatizaciones".
    ◦ Vía Chat Interno Básico: El personal podrá iniciar la solicitud escribiendo directamente en el chat interno del Dashboard comandos o preguntas en lenguaje natural, como: "Genera un correo de recordatorio para Laura Vargas para su cita de mañana" o "Redacta un correo de seguimiento post-cirugía para Juan Pérez, incluyendo instrucciones de medicación". El asistente virtual integrado con Gemma 3 será capaz de procesar estas solicitudes y generar el borrador.

3.2. Generación de Plantillas para Hojas Membretadas:

• Propósito: Esta funcionalidad está diseñada para simplificar la creación de documentos oficiales del consultorio, como certificados médicos, cartas de referencia o plantillas de prescripción, asegurando que mantengan un formato coherente con la identidad de la marca (Dr. Sánchez García) y la información clave necesaria, reduciendo el esfuerzo manual y los tiempos de redacción.
• Entrada para la IA: Para generar el contenido de la plantilla, la IA requerirá:
    ◦ Datos Contextuales del Sistema: Similar a la generación de correos, el sistema podrá proveer información del paciente (ID, nombre, apellido, diagnósticos), así como datos del médico (nombre del Dr. Sánchez García) y datos del consultorio (dirección, teléfono, logo, información de contacto) que ya estén configurados en el sistema.
    ◦ Entrada de Usuario para la Personalización: El usuario definirá:
        ▪ Tipo de Documento: (ej., Certificado Médico, Carta de Referencia, Prescripción, Informe Médico).
        ▪ Contenido Principal: Breve descripción o puntos clave que deben incluirse en el cuerpo del documento.
        ▪ Instrucciones Específicas: Cualquier requisito adicional o formato particular (ej., "Incluir el diagnóstico exacto de [Diagnóstico]", "Especificar la fecha de la última consulta", "Añadir detalles sobre el procedimiento X").
        ▪ Tono y Estilo: Si el usuario desea un lenguaje más formal, técnico o explicativo para el documento.
• Salida Esperada de la IA: La IA producirá el texto formateado del contenido principal de

la hoja membretada. Este texto incluirá:
    ◦ Los campos dinámicos (ej., nombre del paciente, fecha, diagnóstico) que el sistema podrá rellenar automáticamente.
    ◦ El cuerpo principal del documento basado en el tipo y el contenido especificado por el usuario.
    ◦ Un espacio para la firma y los datos del médico.
    ◦ Al igual que con los correos, la IA generará únicamente el contenido de texto y no un archivo de hoja membretada con el diseño gráfico (membrete, logo, etc.). El texto generado se presentará en la interfaz para que el usuario pueda copiarlo y pegarlo en una plantilla de documento predefinida (ej., en Word, accesible vía widget de Office).
• Cómo se Iniciará esta Función en la Interfaz:
    ◦ Desde un Módulo Específico: Se dispondrá de un botón o una opción de "Generar Documento / Plantilla" dentro de la vista de detalle del expediente del paciente, el Historial Clínico Detallado o en la sección general de "Documentos" / "Automatizaciones" en el Dashboard.
    ◦ Vía Chat Interno Básico: El personal podrá solicitar la generación de plantillas a través del chat interno, por ejemplo: "Genera una plantilla de certificado médico para María López con diagnóstico de amigdalitis" o "Prepara un borrador de carta de referencia para el especialista en traumatología para el paciente Roberto Salas". El asistente virtual integrado con Gemma 3 facilitará la delegación de esta tarea.

Especificación de la Integración de Inteligencia Artificial en el Dashboard de Administración
3. Casos de Uso de IA en el Dashboard (Funcionalidades Detalladas):
La integración del modelo de Inteligencia Artificial Gemma 3 (o similar a Gemini) en el backend del Dashboard de Administración es un pilar fundamental para potenciar la "Inteligencia y Automatización Integrada", resolviendo el "dolor" de los "procesos manuales que consumen tiempo" y optimizando la "eficiencia operativa". Las funcionalidades que hacen uso de la IA se activarán de manera intuitiva desde módulos específicos de la aplicación o mediante el chat interno.
A continuación, se describen en detalle los casos de uso específicos de la IA:
3.1. Generación de Borradores de Correo Electrónico:
• Propósito: Esta funcionalidad tiene como objetivo principal acelerar y estandarizar la comunicación escrita con los pacientes o terceros, permitiendo al personal del consultorio redactar correos electrónicos de manera rápida y eficiente [Previous turn]. Al automatizar la generación de borradores, se reduce el tiempo dedicado a tareas repetitivas y se minimiza el riesgo de errores u omisiones en mensajes importantes [Previous turn, 22, 93, 94].
• Entrada para la IA: La IA recibirá información contextual directamente del sistema, así como datos específicos proporcionados por el usuario para generar un borrador relevante [Previous turn]:
    ◦ Datos Contextuales del Sistema: El usuario seleccionará un paciente o una cita desde el Dashboard. El sistema proporcionará automáticamente a la IA el ID del Paciente, el nombre completo del paciente, datos de contacto (ej., email), fechas y detalles de la última consulta o próxima cita y diagnósticos relevantes asociados al expediente [Previous turn].
    ◦ Entrada de Usuario para la Personalización: El personal deberá especificar a la IA:
        ▪ Motivo del Correo: (ej., recordatorio de cita, seguimiento post-operatorio, notificación de resultados de laboratorio, solicitud de información adicional, confirmación de cita) [Previous turn].
        ▪ Puntos Clave a Incluir: Información adicional específica que el usuario desea que se incorpore en el cuerpo del correo (ej., instrucciones pre/post-operatorias, medicamentos, fechas de próxima revisión, requisitos de preparación para estudios) [Previous turn].
        ▪ Tono Deseado: (ej., formal, empático, conciso) [Previous turn].
• Salida Esperada de la IA: La IA generará un borrador de texto para un correo electrónico, estructurado y coherente, que contendrá un asunto sugerido, un saludo personalizado (al paciente o destinatario), el cuerpo del mensaje que integre el motivo, los puntos clave y la información contextual, y un cierre profesional con la información del consultorio y del Dr. Sánchez [Previous turn]. Es importante recalcar que la IA generará únicamente el texto del borrador y no producirá un archivo de correo electrónico final ni lo enviará directamente [Previous turn]. El borrador de texto será presentado en la interfaz del Dashboard para que el usuario pueda revisarlo, editarlo y luego copiarlo o

integrarlo en su cliente de correo electrónico (ej., Gmail, accesible vía widget) [Previous turn, 37, 51, 154, 188, 189, 254, 266].
• Cómo se Iniciará esta Función en la Interfaz: Esta función es una de las "Automatizaciones Básicas" que ofrece la integración de IA.
    ◦ Desde Módulos Específicos: Se habilitará un botón o una opción explícita de "Generar Borrador de Correo" en las siguientes ubicaciones:
        ▪ La vista de detalle de cada expediente del paciente [Previous turn].
        ▪ El módulo de Agenda y Citas, al seleccionar una cita específica [Previous turn].
        ▪ Una sección dedicada a "Automatizaciones" o "Inteligencia y Automatización Integrada" en el Dashboard [Previous turn, 39, 90, 159, 255, 271].
    ◦ Vía Chat Interno Básico: El personal podrá iniciar la solicitud escribiendo directamente en el chat interno del Dashboard comandos o preguntas en lenguaje natural [Previous turn, 159]. Ejemplos incluyen: "Genera un correo de recordatorio para Laura Vargas para su cita de mañana" o "Redacta un correo de seguimiento post-cirugía para Juan Pérez, incluyendo instrucciones de medicación" [Previous turn]. El asistente virtual integrado con Gemma 3 será capaz de procesar estas solicitudes y generar el borrador [Previous turn, 93].

3.2. Generación de Plantillas para Hojas Membretadas:
• Propósito: Esta funcionalidad está diseñada para simplificar la creación de documentos oficiales del consultorio, como certificados médicos, cartas de referencia o plantillas de prescripción, asegurando que mantengan un formato coherente con la identidad de la marca (Dr. Sánchez García) y la información clave necesaria, reduciendo el esfuerzo manual y los tiempos de redacción [Previous turn, 22, 93, 94].
• Entrada para la IA: Para generar el contenido de la plantilla, la IA requerirá [Previous turn]:
    ◦ Datos Contextuales del Sistema: El sistema podrá proveer información del paciente (ID, nombre, apellido, diagnósticos), así como datos del médico (nombre del Dr. Sánchez García) y datos del consultorio (dirección, teléfono, logo, información de contacto) que ya estén configurados en el sistema [Previous turn].
    ◦ Entrada de Usuario para la Personalización: El usuario definirá:
        ▪ Tipo de Documento: (ej., Certificado Médico, Carta de Referencia, Prescripción, Informe Médico) [Previous turn].
        ▪ Contenido Principal: Breve descripción o puntos clave que deben incluirse en el cuerpo del documento [Previous turn].
        ▪ Instrucciones Específicas: Cualquier requisito adicional o formato particular (ej., "Incluir el diagnóstico exacto de [Diagnóstico]", "Especificar la fecha de la última consulta", "Añadir detalles sobre el procedimiento X") [Previous turn].
        ▪ Tono y Estilo: Si el usuario desea un lenguaje más formal, técnico o explicativo para el documento [Previous turn].
• Salida Esperada de la IA: La IA producirá el texto formateado del contenido principal de la hoja membretada [Previous turn]. Este texto incluirá los campos dinámicos (ej., nombre del paciente, fecha, diagnóstico) que el sistema podrá rellenar automáticamente, el cuerpo principal del documento basado en el tipo y el contenido especificado por el usuario, y un espacio para la firma y los datos del médico [Previous turn]. Al igual que con los correos, la IA generará únicamente el contenido de texto y no un archivo de hoja membretada con el diseño gráfico (membrete, logo, etc.) [Previous turn]. El texto generado se presentará en la interfaz para que el usuario pueda copiarlo y pegarlo en una plantilla de documento predefinida (ej., en Word, accesible vía widget de Office) [Previous turn, 37, 51, 154, 188, 189, 254, 266].
• Cómo se Iniciará esta Función en la Interfaz: Esta función es parte de las "Automatizaciones Básicas" que la IA puede realizar.
    ◦ Desde Módulos Específicos: Se dispondrá de un botón o una opción de "Generar Documento / Plantilla" dentro de las siguientes vistas:
        ▪ La vista de detalle del expediente del paciente [Previous turn].
        ▪ El Historial Clínico Detallado del paciente [Previous turn].
        ▪ En la sección general de "Documentos" o "Automatizaciones" en el Dashboard [Previous turn, 39, 90, 159, 255, 271].
    ◦ Vía Chat Interno Básico: El personal podrá solicitar la generación de plantillas a través del chat interno, por ejemplo: "Genera una plantilla de certificado médico para María López con diagnóstico de amigdalitis" o "Prepara un borrador de carta de referencia para el especialista en traumatología para el paciente Roberto Salas" [Previous turn]. El asistente virtual integrado con Gemma 3 facilitará la delegación de esta tarea [Previous turn, 93].

3.3. Generación de Recordatorios Automatizados:
• Propósito: Esta funcionalidad tiene como propósito principal liberar tiempo valioso para el personal del consultorio al automatizar la creación y gestión de recordatorios. Ayuda a garantizar que tanto el personal como los pacientes estén informados sobre citas, eventos o tareas importantes, reduciendo los "procesos manuales que consumen tiempo" y mejorando la "eficiencia operativa". Esto incluye la capacidad de enviar "Notificaciones y Recordatorios Automatizados" a los pacientes, especialmente a través de la integración con WhatsApp.
• Entrada para la IA: La IA y el sistema consumirán datos relevantes para la generación de recordatorios:
    ◦ Datos de Citas: Información del Módulo de Agenda y Citas, incluyendo fecha, hora, tipo de cita y estado (Confirmada, Programada, etc.).
    ◦ Datos del Paciente: Nombre y datos de contacto (ej., número de teléfono para WhatsApp) asociados a la cita.
    ◦ Eventos o Tareas: Datos sobre tareas internas del personal (ej., "revisar expedientes mañana", "preparar materiales para cirugía") o eventos específicos que requieran un recordatorio. Si bien no se detalla un módulo de "tareas" para el personal en el Dashboard, la capacidad de la IA para generar texto y automatizar sugiere que podría procesar estos datos si se ingresan.
• Posible Rol de la IA: El modelo Gemma 3 se integrará en el backend mediante APIs para potenciar estas automatizaciones. El rol de la IA en la generación de recordatorios podría incluir:
    ◦ Generación de Contenido Sugerido: Interpretar los datos de la cita o tarea para sugerir el texto más apropiado para el recordatorio (ej., "Su cita para el 15 de julio a las 10:00 AM ha sido confirmada").
    ◦ Personalización del Tono: Ajustar el tono del recordatorio (ej., más formal para una cita inicial, más conciso para un recordatorio rápido de tarea).
    ◦ Sugerencia de Momentos Clave: Si se le entrena con patrones, la IA podría sugerir el momento óptimo para enviar un recordatorio (ej., 24 horas antes de la cita, 1 hora antes de la cita, al finalizar el día para tareas pendientes).
• Cómo se Configurarán estos Recordatorios:
    ◦ Los recordatorios son parte de las "Automatizaciones Básicas" ofrecidas por la IA.
    ◦ Se configurarán a través de módulos específicos del Dashboard (como el "Módulo de Agenda y Citas") o, en algunos casos, pueden ser invocados a través del chat interno.
    ◦ Para los recordatorios de pacientes vía WhatsApp, el sistema actuará de forma automática basándose en la información del calendario de citas. Esto implica que las plantillas de recordatorio (posiblemente generadas con asistencia de IA) y las reglas de envío se establecerán previamente en el sistema.
3.4. Chat Interno con el Modelo de IA:
• Propósito: El chat interno básico tiene como propósito principal permitir al personal del consultorio interactuar directamente con el modelo de IA (Gemma 3 o similar a Gemini) para obtener asistencia rápida en tareas de texto y consultas. Su objetivo es mejorar la eficiencia operativa al ofrecer una herramienta de consulta inmediata y automatización de procesos de texto repetitivos.
• Funcionalidades Específicas del Chat: El chat interno, potenciado por la IA, ofrecerá las siguientes capacidades:
    ◦ Resúmenes de Texto:
        ▪ Entrada: El usuario proporcionará texto para resumir (ej., notas de evolución del historial clínico, una serie de interacciones con el paciente, o un informe largo) [Previous turn, 23, 38, 93].
        ▪ Salida: La IA generará un resumen conciso y coherente del texto proporcionado [Previous turn, 23, 38, 93].
    ◦ Redacción de Informes Cortos:
        ▪ Entrada: El usuario proporcionará puntos clave, datos específicos, o una descripción breve del propósito del informe [Previous turn].
        ▪ Salida: La IA redactará un borrador de texto para un informe corto, estructurando la información y aplicando un tono adecuado [Previous turn, 23, 38, 93].
    ◦ Respuestas a Consultas Rápidas:
        ▪ El personal podrá realizar preguntas directas o comandos en lenguaje natural (ej., "Encuéntrame los últimos estudios de Laura Vargas", "Qué paciente tiene cita a las 11 AM?", "Resúmeme la última consulta de Juan Pérez") [Previous turn, 159, 283].
        ▪ Acceso y Uso de Datos Relevantes: La IA accedería o utilizaría datos relevantes a través de su integración con el backend del sistema. El backend, construido con

Python/Flask/Django o Node.js/Express, se comunicará con la Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL) donde se almacena toda la información estructurada del consultorio (perfiles de pacientes, historiales clínicos, citas, usuarios y roles). También accederá al Almacenamiento de Archivos (Cloud Storage) para documentos digitales. De esta manera, la IA podrá localizar recursos y ofrecer acciones contextuales (ej., "Lo encontré", "te lo envío a tu correo o prefieres descargarlo?").

- Descripción de la Interfaz de Usuario del Chat:
  - Ubicación: El chat se accederá a través de un icono o una sección dedicada, visible dentro de la interfaz del Dashboard de Escritorio.
  - Área de Entrada: Contará con un campo de entrada de texto claro y conciso donde el usuario escribirá sus consultas o comandos.
  - Área de Visualización: Encima del campo de entrada, habrá un área de visualización donde se mostrará el historial de la conversación, con las preguntas del usuario y las respuestas o el contenido generado por la IA.
  - Botón de Envío: Un botón (ej., "Enviar") para enviar la consulta a la IA.
  - Formato de Salida: Es fundamental recalcar que este chat funciona únicamente con texto y no tiene la capacidad de generar archivos de ningún formato (ej., PDFs, imágenes).

3.5. Otros Casos de Uso de Automatización Básicos Mencionados:

Además de las funcionalidades de redacción de correos, generación de hojas membretadas, recordatorios y el chat interno básico, la integración de la IA Gemma 3 extiende sus capacidades a otras áreas de automatización y asistencia, principalmente en el ámbito de texto y soporte:

- Asistencia para el Desarrollo y Capacitación: La IA puede ser una herramienta valiosa para el equipo de desarrollo y para la capacitación del personal.
  - Generación de Texto de Ejemplo (Lorem Ipsum Inteligente): Puede generar texto de relleno relevante para prototipos de interfaz de usuario (UI), agilizando el proceso de diseño.
  - Sugerencia de "Copys": Si se le entrena con el tono deseado, la IA puede sugerir textos para botones y etiquetas de la interfaz.
  - Generación de Scripts SQL Básicos: A partir de descripciones en lenguaje natural, la IA puede generar scripts SQL fundamentales para la base de datos.
  - Generación de Preguntas Frecuentes (FAQs): Puede colaborar en la creación de FAQs a partir de la documentación técnica, que serán útiles para el manual de usuario y la capacitación inicial.
  - Creación de un Chatbot de Ayuda Inicial para Capacitación: La IA puede sentar las bases para un chatbot que asista al personal durante su capacitación, respondiendo preguntas básicas sobre el uso del sistema.
- Asistente Virtual (Bot) en Página Web y WhatsApp: Aunque son parte de fases posteriores (Fase 2 y 3), la IA Gemma 3 es fundamental para potenciar los asistentes virtuales externos.
  - Respuestas a Preguntas Frecuentes (FAQ): Los bots en la página web y WhatsApp, impulsados por IA, podrán responder a las preguntas más comunes de los pacientes.
  - Agendamiento de Citas: La IA facilitará el proceso de agendamiento de citas a través de los bots, sincronizándose con el calendario del Dashboard.
  - Delegación de Gestión Inicial: El Asistente Virtual Inteligente (AVI) delegará la gestión inicial de consultas y agendamiento a la IA, liberando tiempo valioso del personal del consultorio.

3.6. Límites de las Capacidades de la IA en esta Fase Específica del Proyecto:

Es crucial establecer los límites claros de la inteligencia artificial en esta fase inicial del proyecto para gestionar expectativas y garantizar un uso responsable y seguro de la tecnología, especialmente con datos sensibles de pacientes. La IA se enfoca en la "Inteligencia y Automatización Integrada" para optimizar la "eficiencia operativa".

- Enfoque en Texto y Generación de Borradores: La IA está diseñada principalmente para procesar y generar texto.
  - Produce borradores y sugerencias de texto (correos, plantillas, informes, resúmenes, FAQs) [Previous turn, 23, 38, 93, 115, 159].
  - No generará archivos finales (ej., PDFs con diseño gráfico, documentos de Word completos, ni enviará correos directamente) [Previous turn, 23, 38, 93, 115, 159]. La interacción final y el envío recaen en el usuario.
- No Generación de Contenido Multimedia: La IA no tiene la capacidad de crear ni manipular imágenes, videos u otros tipos de archivos multimedia [No source mentions this capability for AI]. La "Gestión Documental Avanzada" se encarga de subir, asociar y visualizar estos archivos, que son provistos externamente o por el sistema.

• No Toma de Decisiones Médicas o Legales: La IA es una herramienta de asistencia y automatización. No está diseñada para interpretar datos médicos complejos con fines de diagnóstico, recomendación de tratamientos, toma de decisiones clínicas ni para ofrecer asesoramiento legal o ético [No source mentions this capability for AI, and it contradicts the "asistencia" purpose]. Su rol es de apoyo al personal, procesando información existente o generando texto basado en entradas definidas.
• No Autonomía en Acciones Críticas: Si bien puede automatizar tareas, estas automatizaciones están dentro de los flujos de negocio predefinidos y, en muchos casos, requieren una revisión y confirmación humana (como en la generación de borradores de correos) [Previous turn]. La IA se integra en el backend del sistema, lo que significa que opera dentro de la lógica de negocio y las APIs establecidas, no de forma autónoma fuera de los parámetros del sistema.
• Dependencia de la Calidad de la Entrada: La calidad de la salida de la IA dependerá directamente de la claridad y completitud de la información de entrada proporcionada por el usuario o el sistema. No "adivinará" información ausente.
En resumen, la IA en esta fase es un potente asistente de texto y automatización de procesos administrativos, diseñado para aumentar la eficiencia y liberar tiempo del personal, siempre bajo la supervisión y control del usuario.

Interacción Técnica con el Modelo de IA (Gemma 3)
La integración del modelo de Inteligencia Artificial Gemma 3 (o similar a Gemini) es un componente fundamental del "Ecosistema Digital Inteligente para Cirugía Especial". Esta inteligencia artificial potencia funcionalidades clave como la automatización de tareas repetitivas y la asistencia rápida al personal del consultorio.
Método de Integración
El modelo Gemma 3 se integra directamente en el Servidor de Aplicación (Backend) del ecosistema mediante el uso de APIs (Interfaces de Programación de Aplicaciones). Esto significa que el backend actúa como el intermediario entre la interfaz de usuario (el Dashboard de Escritorio o las aplicaciones móviles/web) y el modelo de IA.
El backend se construirá utilizando lenguajes y frameworks robustos como Python (Flask/Django) o Node.js (Express). El despliegue de este backend se realiza en Google Cloud Platform (GCP), utilizando servicios como Compute Engine o Cloud Run.
En cuanto a la forma específica de acceso al modelo Gemma 3, las fuentes indican que puede ser a través de dos métodos principales:
• Acceso mediante API pública (ej. Google AI API para Gemini): Si Gemma 3 se ofrece como un servicio gestionado a través de una API pública, el backend se comunicará con ella utilizando endpoints HTTP específicos. Esto implicaría el envío de solicitudes (probablemente POST) a URLs definidas por el proveedor de la IA, con los prompts y datos necesarios en el cuerpo de la solicitud. La comunicación se realizará siempre a través de HTTPS (TLS 1.3) para cifrar los datos en tránsito, protegiendo tanto las credenciales del sistema como cualquier información sensible que pudiera acompañar a los prompts.
• Despliegue del Modelo en Google Cloud (Vertex AI): Otra opción, sugerida en las fuentes, es que el modelo Gemma 3 se despliegue de forma autónoma en Vertex AI, el servicio de aprendizaje automático de GCP. En este caso, la interacción del backend con el modelo se realizaría utilizando los SDKs o APIs de Vertex AI, lo que podría ofrecer una integración más profunda y optimizada dentro del ecosistema de GCP, manteniendo la comunicación dentro de la Virtual Private Cloud (VPC) para mayor seguridad.
Especificación de Entradas (Prompts)
Las entradas al modelo de IA (prompts) serán principalmente de texto, ya que el chat interno básico de la IA funciona únicamente con texto y no genera archivos de ningún formato. La especificación y estructura de estos prompts variará según el caso de uso:
1. Automatizaciones Básicas (Redacción de Correos, Hojas Membretadas, Recordatorios):
    ◦ Formato del Prompt: Se espera que el prompt incluya instrucciones claras y concisas sobre el tipo de documento o mensaje a generar, el tono deseado, y los datos específicos que deben insertarse. Por ejemplo, para un recordatorio de cita, el prompt podría incluir el nombre del paciente, la fecha y hora de la cita, el tipo de consulta y cualquier instrucción especial.
    ◦ Contexto: Los datos de contexto (ej., nombre del paciente, detalles de la cita) serían extraídos de la Base de Datos Relacional (Cloud SQL) por el backend antes de ser formateados en el prompt para la IA.
2. Chat Interno Básico (Consultas Rápidas, Resúmenes de Texto, Redacción de Informes):
    ◦ Formato del Prompt: Para consultas rápidas o solicitudes de resumen, el prompt sería el texto directo de la consulta o el texto a resumir. Para la redacción de informes, el

prompt incluiría las directrices para el informe y los datos relevantes del paciente o la consulta.
    - Manejo del Contexto en Conversaciones: Aunque no se detalla explícitamente el mecanismo, un chat interno requiere que el backend mantenga y envíe el contexto de la conversación previa al modelo de IA. Esto podría implicar el envío de las últimas 'N' interacciones (pares pregunta-respuesta) o un resumen condensado del diálogo anterior junto con la nueva consulta del usuario, permitiendo que la IA mantenga la coherencia y relevancia en sus respuestas.

3. Asistencia para el Desarrollo y Capacitación (Generación de Texto de Ejemplo, "Copys", Scripts SQL, FAQs):
    - Formato del Prompt:
        - Para texto de ejemplo o "copys" de UI: El prompt incluiría el propósito del texto, la interfaz o elemento donde se usará, y el tono deseado.
        - Para scripts SQL básicos: El prompt sería una descripción en lenguaje natural de la operación de base de datos deseada (ej., "crea una tabla para pacientes con nombre, apellido y fecha de nacimiento").
        - Para FAQs: El prompt podría consistir en segmentos de la documentación técnica o consultas comunes de usuarios, pidiéndole a la IA que genere preguntas y respuestas concisas.

Especificación de Salidas (Respuestas)

Las respuestas del modelo de IA, como se mencionó, serán principalmente de texto. El formato exacto y la estructura esperada del modelo de IA para cada caso de uso son:

1. Automatizaciones Básicas:
    - Formato: Texto plano o texto formateado (ej., con saltos de línea, listas) que representa el borrador completo del correo electrónico, la hoja membretada o el recordatorio.
    - Ejemplo: Un borrador de correo con campos como Asunto, Cuerpo del mensaje, Saludo y Firma ya rellenos por la IA.
2. Chat Interno Básico:
    - Formato: Texto plano que constituye la respuesta directa a la consulta del usuario, el resumen solicitado, o el borrador del informe.
    - Restricción: Es crucial que la IA no genere archivos de ningún formato como salida de este chat.
3. Asistencia para el Desarrollo y Capacitación:
    - Formato:
        - Texto de ejemplo o "copys": Texto plano o frases cortas para usar en la interfaz.
        - Scripts SQL: Texto que contiene las sentencias SQL generadas.
        - FAQs: Pares de preguntas y respuestas en formato de texto estructurado (posiblemente un JSON si la IA se entrena para ello, aunque no especificado directamente, sería una buena práctica para facilitar su consumo por el backend).

Manejo de Errores y Respuestas Inesperadas de la IA

La robustez del sistema depende de cómo el backend detecta y gestiona los errores o respuestas de la IA que no se ajusten al formato esperado o sean irrelevantes. La integración de la IA debe realizarse de forma segura, garantizando la privacidad de los datos sensibles de los pacientes.

1. Detección de Errores y Anomalías:
    - Errores de Conectividad/Red: Fallos al establecer comunicación con la API de la IA (ej., timeouts, errores de DNS).
    - Respuestas Malformadas: La IA podría devolver una respuesta que no sigue el formato de texto esperado, o si se esperara JSON, un JSON inválido o incompleto.
    - Respuestas Incoherentes/Irrelevantes (Alucinaciones): La IA podría generar texto que, aunque gramaticalmente correcto, es ilógico o no responde a la solicitud del prompt, especialmente si el contexto es ambiguo o los datos de entrada son insuficientes.
    - Limitaciones de Tasa (Rate Limiting): Si se utiliza una API pública, se podrían alcanzar los límites de llamadas por minuto/segundo, resultando en errores HTTP 429.
    - Errores del Modelo Interno: Problemas en el lado del proveedor de la IA o del despliegue en Vertex AI que impidan una respuesta exitosa (errores 5xx del servicio de la IA).
    - Fallos de Validación de Contenido: Aunque la IA genere texto, el backend podría aplicar validaciones post-generación (ej., longitud mínima/máxima, presencia de palabras clave, ausencia de información prohibida).
2. Estrategias de Manejo de Errores:

◦ Mensajes de Error al Usuario: En caso de fallo de la IA o respuesta inesperada, el frontend mostrará un mensaje de error claro y conciso al usuario (ej., "Lo sentimos, no pudimos procesar su solicitud en este momento. Por favor, intente de nuevo más tarde." o "Hubo un problema al conectar con la IA.").
◦ Registro de Logs de Auditoría: Cada interacción con la IA, incluyendo los intentos, las respuestas y, crucialmente, los errores, debe registrarse cronológicamente en el Servicio de Logs de Auditoría (Cloud Logging / Audit Logs). Estos logs son inmutables (Write Once, Read Many - WORM), lo que permite rastrear, depurar y analizar cualquier anomalía, siendo vitales para la seguridad y el cumplimiento normativo.
◦ Mecanismos de Reintento (Retry Mechanisms): Para errores transitorios (ej., problemas de red, límites de tasa), el backend podría implementar reintentos con retraso exponencial antes de reportar un fallo definitivo.
◦ Fallback a Respuestas por Defecto/Manuales: Para funciones críticas donde la IA falla, se podría tener un mecanismo de fallback que proporcione una respuesta predefinida o dirija al usuario a una opción manual.
◦ Monitoreo y Alertas: Configuración de alertas en Cloud Logging o Cloud Monitoring para notificar a los administradores sobre patrones de errores de la IA o sobre respuestas que requieran revisión humana.
◦ Control de Seguridad de Datos: El backend es el guardián de la información. Antes de enviar datos sensibles (ej., historial clínico del paciente) a la IA, el backend debe asegurarse de que la información se anonimice o se envíe solo la parte estrictamente necesaria. Las respuestas de la IA deben ser cuidadosamente validadas para evitar que devuelva inadvertidamente información sensible o incorrecta. El control de acceso basado en roles (RBAC) es crucial para asegurar que solo usuarios autorizados puedan interactuar con funcionalidades que impliquen la IA y datos sensibles.
Esta integración asegura que la IA se utilice de manera efectiva para optimizar el flujo de trabajo, manteniendo siempre la seguridad y la integridad de la información del paciente.

5. Manejo de Datos y Privacidad en la Integración con IA
La protección de los datos sensibles del paciente es una prioridad máxima en el diseño y la implementación del Ecosistema Digital Inteligente para Cirugía Especial [1_2, 1_6, 1_14, 1_15, 2_9, 2_10, 2_11, 2_12, 2_14, 4_185, 4_186, 4_193, 7_74, 7_85, 7_97, 7_108, 1_261]. La integración con el modelo de IA Gemma 3 se rige por estrictos protocolos de seguridad y privacidad para garantizar el cumplimiento normativo y la confianza del usuario.
• Cómo se asegurará que los datos sensibles del paciente utilizados por la IA se manejen de forma segura y cumplan con las normativas de privacidad:
◦ Cifrado de Datos en Tránsito (HTTPS/TLS 1.3): Toda la comunicación entre la aplicación de escritorio y la infraestructura en la nube (Google Cloud Platform - GCP), donde reside el backend y la IA, se realiza exclusivamente a través de HTTPS (TLS 1.3) [1_5, 1_14, 2_19, 3_26, 4_61, 4_69, 4_190, 4_193, 7_76, 7_99, 1_181, 1_199, 1_203]. Esto asegura que las credenciales de usuario y cualquier dato sensible del paciente transmitido al modelo de IA estén cifrados y protegidos contra la interceptación o manipulación [1_14, 1_181, 1_203].
◦ Cifrado de Datos en Reposo: Los datos sensibles de los pacientes, incluidos los historiales clínicos y los archivos digitales (PDF, imágenes, videos), se almacenan cifrados en reposo tanto en la Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL) como en el Almacenamiento de Archivos (Cloud Storage - S3/Blob) [1_6, 1_14, 2_18, 2_19, 4_59, 4_61, 4_69, 4_190, 4_193, 7_75, 7_76, 7_98, 7_99, 1_181, 1_203, 1_214, 1_221, 1_253, 1_274]. Las contraseñas de los usuarios se almacenan como hash_password en la base de datos, lo que indica el uso de funciones de hash seguras en lugar de texto plano [1_6, 1_14, 1_67, 1_181, 1_203, 1_218, 1_202]. Este cifrado es una medida esencial contra fugas de datos [1_14, 2_19, 4_69, 4_193, 7_76, 7_99].
◦ Control de Acceso Basado en Roles (RBAC): El sistema impone un control estricto de acceso. Cada usuario autenticado, al iniciar sesión, es asociado a un rol específico (ej., "Admin Principal", "Admin Secundario", "Asistente", "Invitado (Solo Lectura)") que define sus permisos sobre la información y las funcionalidades del sistema [1_2, 1_6, 1_13, 2_19, 3_28, 4_68, 4_193, 7_76, 7_99, 1_181, 1_203, 1_254, 1_267, 1_268]. Esto limita el riesgo de acceso indebido a datos sensibles, asegurando que la IA solo procese la información a la que el usuario que inicia la interacción tiene permiso [1_13, 1_181, 1_197, 1_203].
◦ Servicio de Logs de Auditoría Inmutables (WORM): Cada interacción con la IA y cualquier acción relevante del usuario (como la modificación de datos de paciente o la

creación de citas) se registra cronológicamente en el Servicio de Logs de Auditoría (Cloud Logging / Audit Logs) [1_7, 1_15, 2_18, 4_62, 4_69, 4_190, 4_193, 7_75, 7_76, 7_98, 7_99, 1_181, 1_199, 1_203, 1_219, 1_228, 1_274]. Estos logs son inmutables (Write Once, Read Many - WORM), lo que asegura su integridad y los hace vitales para rastrear actividades, detectar anomalías, responder a incidentes y asegurar el cumplimiento normativo [1_7, 1_15, 2_18, 4_69, 4_190, 4_193, 7_75, 7_76, 7_98, 7_99, 1_181, 1_199, 1_203].
    ◦ Backend como Guardián de Datos: El backend del sistema actúa como el guardián de la información sensible [Conversación previa]. Antes de enviar cualquier dato a la IA, el backend debe asegurarse de que solo se envíe la parte estrictamente necesaria de la información [Conversación previa]. Además, las respuestas de la IA deben ser cuidadosamente validadas para evitar que devuelva inadvertidamente información sensible o incorrecta [Conversación previa].
• Especificar si los datos sensibles se anonimizan o pseudonimizan antes de enviarlos a la IA (si aplica y es necesario/posible con el modelo):
    ◦ Aunque las fuentes no detallan explícitamente el mecanismo de anonimización o pseudonimización, la directriz es clara: el backend "debe asegurarse de que la información se anonimice o se envíe solo la parte estrictamente necesaria" antes de enviarla a la IA [Conversación previa]. Esto implica que, si la naturaleza de la consulta a la IA lo permite (ej., la IA necesita solo el contexto general o un atributo específico, no la identidad del paciente), se priorizará el envío de datos anonimizados o pseudonimizados. En casos donde se requiere información específica del paciente (ej., para redactar un recordatorio con su nombre), se enviará solo la información mínima indispensable para la tarea.
• Consideraciones sobre dónde se procesan los datos (en la nube del proveedor de IA, en tu propia VPC en GCP):
    ◦ El modelo de IA Gemma 3 (o similar a Gemini) se integra directamente en el Servidor de Aplicación (Backend) del ecosistema mediante el uso de APIs [2_18, 3_25, 3_26, 4_44, 4_193, 1_220]. El backend, a su vez, está desplegado en Google Cloud Platform (GCP), utilizando servicios como Compute Engine o Cloud Run [2_18, 3_26, 1_5, 1_221].
    ◦ La infraestructura completa en la nube reside dentro de una Virtual Private Cloud (VPC) [1_15, 2_18, 7_75, 7_98, 1_199, 1_203, 1_220]. Esto crea un entorno de red privado y seguro, permitiendo al equipo de desarrollo definir su propia topología de red virtual y controlar el tráfico [2_18, 7_75, 7_98].
    ◦ Si el modelo Gemma 3 es accedido a través de una API pública de Google AI (como se menciona para Gemini), el procesamiento real de la IA ocurriría en los servidores del proveedor de IA (Google). Sin embargo, la comunicación del backend con esta API se mantendría dentro del entorno seguro de GCP y, en la medida de lo posible, dentro de la VPC para mayor seguridad, limitando la exposición de los datos [2_18, 1_15, 1_199, 1_203, Conversación previa].
    ◦ Si el modelo se despliega directamente en Vertex AI (el servicio de Machine Learning de GCP), el procesamiento de los datos por la IA ocurriría dentro de la propia infraestructura de GCP del proyecto, lo que podría ofrecer un control más granular sobre la ubicación y seguridad de los datos, potencialmente manteniéndolos dentro de la VPC definida para el ecosistema [2_18, 7_75, 7_98, 1_222]. Esto es preferible para datos altamente sensibles.

6. Rendimiento y Escalabilidad de la Integración con IA

El rendimiento y la escalabilidad de la integración con IA son fundamentales para asegurar la "eficiencia operativa" y una experiencia de usuario fluida [2_18, 3_22, 4_186].
• Requisitos de rendimiento: Tiempo de respuesta esperado de la IA para cada caso de uso (ej. la respuesta del chat debe ser rápida):
    ◦ Para el Chat Interno Básico y las Automatizaciones Básicas (redacción de correos, generación de hojas membretadas, recordatorios), el tiempo de respuesta de la IA debe ser rápido para no interrumpir el flujo de trabajo del personal [3_22, 1_254, 1_159]. Si bien no se especifica un número exacto en las fuentes, la meta de "asistencia rápida" y "optimizar la eficiencia operativa" implica una latencia muy baja, idealmente sub-segundo para interacciones de chat simples y unos pocos segundos para tareas de generación de texto más complejas. Un tiempo de respuesta lento impactaría negativamente la adopción y el valor percibido de la IA.
    ◦ Para Asistencia para el Desarrollo y Capacitación (generación de scripts SQL básicos, FAQs), los requisitos de tiempo de respuesta pueden ser un poco más flexibles, pero aún se espera un rendimiento eficiente para mantener la productividad.
• Consideraciones sobre la latencia en las llamadas a la API de la IA:
    ◦ La latencia es un factor crítico en las comunicaciones de la aplicación de

escritorio con la nube. La comunicación se realiza a través de APIs RESTful utilizando HTTPS (TLS 1.3) [4_61, 4_190, 1_5, 1_14, 2_19, 3_26].
    ◦ El rendimiento óptimo del ecosistema requiere una conexión a internet estable de al menos 250 Mbps [1_10, 2_19, 4_63, 4_193, 7_76, 7_99, 1_118, 1_245, 1_259, 1_272]. Esta especificación de conectividad de red de alta velocidad indica la expectativa de una baja latencia general del sistema.
    ◦ Si el modelo de IA se accede a través de una API pública externa, la latencia también dependerá de la ubicación geográfica del endpoint de la API y de la congestión de la red de internet. Si el modelo se despliega en Vertex AI dentro de la misma región de GCP que el backend del ecosistema, la latencia interna de la nube se minimizará significativamente.
• Cómo la infraestructura soportará el volumen esperado de interacciones con la IA a medida que crezca el uso del Dashboard:
    ◦ La arquitectura de nube en GCP está diseñada para ofrecer escalabilidad y disponibilidad inherente [2_17, 2_19, 7_75, 7_76, 7_98, 7_99]. Esto es crucial para soportar el crecimiento del volumen de interacciones con la IA.
    ◦ El Servidor de Aplicación (Backend), donde se integra la IA, se desplegará utilizando servicios de GCP como Compute Engine o Cloud Run [2_18, 3_26, 1_221]. Cloud Run, en particular, ofrece escalado automático basado en la demanda, lo que permite que el backend maneje un mayor volumen de solicitudes de IA sin intervención manual.
    ◦ El API Gateway / Load Balancer actúa como el punto de entrada seguro y distribuye el tráfico de manera eficiente entre los servidores de aplicación, optimizando el rendimiento y garantizando la disponibilidad del sistema incluso bajo alta demanda [2_18, 7_75, 7_98, 1_181, 1_203, 1_220]. Esto es fundamental para absorber picos de uso de la IA.
    ◦ La Base de Datos Relacional (Cloud SQL) y el Almacenamiento de Archivos (Cloud Storage) también son servicios escalables de GCP, capaces de manejar un creciente volumen de datos de pacientes que podrían ser utilizados como contexto para la IA [2_18, 7_75, 7_98].
• Consideraciones de costo basadas en el uso (por token, por llamada):
    ◦ Las fuentes establecen un costo general para la infraestructura en la nube de GCP: una cortesía de 6 meses (hasta 1 TB mensual) desde la entrega de la Fase 1, tras lo cual el costo será de $65.00 USD mensuales, adicionales al plan de mantenimiento [2_20, 1_117, 1_257, 1_270, 7_76, 7_99]. Esta cortesía está directamente vinculada a la vigencia del Plan de Mantenimiento y Soporte [2_20, 1_117, 1_257, 1_270, 7_76, 7_99].
    ◦ Cualquier almacenamiento adicional más allá del 1 Terabyte incluido tendrá un costo extra según las tarifas del proveedor de nube [2_20, 1_118, 1_259, 1_272, 7_76, 7_99].
    ◦ Las fuentes no desglosan los costos específicamente por el uso del modelo de IA (ej., por token, por llamada a la API de Gemma 3). Sin embargo, el costo de $65.00 USD mensuales después de la cortesía es el monto total por el servicio en la nube que alberga toda la infraestructura, incluyendo el backend que interactúa con la IA [2_20, 1_117, 1_257, 1_270]. Si el uso de la IA se incrementara significativamente y esto supusiera un aumento sustancial en el consumo de recursos de cómputo (CPU, memoria) o de red del backend, podría implicar que los costos operativos de GCP superen el monto fijo y requieran una revisión, aunque las fuentes solo especifican costos adicionales por almacenamiento. Para la alta dirección, es clave entender que la IA es parte de un ecosistema cuyo costo operativo está detallado en los términos contractuales, y que la escalabilidad del sistema podría generar costos adicionales si se exceden los límites del plan actual de 1TB o el volumen de cómputo asociado.

---

7. Consideraciones Adicionales sobre la Integración de IA
La integración del modelo de Inteligencia Artificial Gemma 3 (o similar a Gemini) es un pilar fundamental para potenciar la eficiencia operativa y la asistencia al personal dentro del "Ecosistema Digital Inteligente para Cirugía Especial". Para garantizar que esta capacidad transversal sea efectiva, segura y escalable a largo plazo, es esencial considerar no solo su implementación inicial sino también su monitoreo y las potenciales vías de expansión futura.
7.1. Monitoreo del Uso y Rendimiento de la IA
Aunque los documentos no especifican un servicio de monitoreo exclusivo para la IA, la arquitectura del ecosistema ya incluye un "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)". Este servicio es vital para la seguridad y el seguimiento de operaciones, registrando cronológicamente todas las actividades y cambios realizados en el sistema, como modificaciones de datos de pacientes o accesos de usuarios. Estos logs son inmutables

(Write Once, Read Many - WORM), asegurando su integridad y siendo cruciales para la detección de anomalías y la respuesta a incidentes.
Para la IA, la extensión de este monitoreo podría incluir:
• Registro de Interacciones de IA: Cada solicitud al modelo Gemma 3 (ya sea desde el chat interno o para automatizaciones como redacción de correos) y su respuesta deberían ser registradas en los logs de auditoría. Esto permitiría un seguimiento de la frecuencia de uso de las funcionalidades de IA, identificar patrones de consulta y auditar el contenido generado.
• Métricas de Rendimiento: Aunque no se detalla, la implementación de la IA como parte del backend permite el monitoreo de métricas técnicas como la latencia de respuesta del modelo, la tasa de errores en las peticiones y el consumo de recursos (CPU, memoria) en Google Cloud Platform (GCP). Esto es fundamental para asegurar que la IA responde de manera oportuna y eficiente.
• Calidad de la Generación de Contenido: Para funciones como la redacción de correos o informes, es importante considerar mecanismos, aunque sean manuales inicialmente, para evaluar la calidad y precisión del texto generado por la IA. Esto podría implicar que los usuarios validen el contenido o proporcionen retroalimentación directa en la interfaz, lo que, a su vez, podría alimentar futuras mejoras del modelo o ajustes en los "prompts".
• Privacidad de Datos en Interacciones con IA: Dado que la IA procesará consultas relacionadas con información sensible del consultorio (aunque el chat sea de texto solamente y no genere archivos), es "crucial que esta integración se realice de forma segura, garantizando la privacidad de los datos sensibles de los pacientes". El monitoreo debe asegurar que no haya exposición indebida de datos y que las políticas de retención y anonimización sean respetadas.
7.2. Posibles Futuras Expansiones de Funcionalidades de IA
La integración de Gemma 3 en el backend mediante APIs establece una base sólida para el crecimiento y la evolución de las capacidades de IA del ecosistema. Más allá de las funcionalidades iniciales de automatización básica (redacción de correos, hojas membretadas, recordatorios) y el chat interno (consultas rápidas, resúmenes, redacción de informes), así como la asistencia en la generación de FAQs para manuales y scripts SQL, y la potenciación del Asistente Virtual Inteligente (AVI) en WhatsApp para FAQ y agendamiento, se pueden vislumbrar las siguientes expansiones estratégicas:
• Generación Avanzada de Documentos y Reportes Clínicos: La capacidad actual de "redacción de informes" y "generación de hojas membretadas" podría evolucionar. La IA podría generar borradores más completos de informes médicos, resúmenes de altas o consentimientos informados, extrayendo información directamente del "Historial Clínico Detallado" del paciente de forma estructurada.
• Asistencia en la Clasificación y Etiquetado de Documentos: Dada la "Gestión Documental Avanzada" que permite subir y asociar archivos digitales como PDFs e imágenes, la IA podría ayudar a etiquetar automáticamente estos documentos (ej. "Radiografía", "Resultados de Laboratorio", "Formulario de Admisión") o incluso a resumir su contenido textual si son PDFs, mejorando la organización y búsqueda.
• Análisis y Sugerencias Basadas en Datos del Paciente: Si bien los documentos enfatizan la seguridad y el control, en un futuro, con estrictas consideraciones de privacidad, la IA podría analizar el historial clínico para sugerir posibles diagnósticos diferenciales, recordar al médico sobre interacciones medicamentosas o alertas de alergias (si los datos estructurados lo permiten), o proponer planes de seguimiento personalizados basados en guías clínicas preestablecidas. Esto requeriría modelos de IA entrenados específicamente en datos médicos y validados rigurosamente.
• Interfaz de Voz para el Dashboard Interno: La aplicación móvil para pacientes ya contempla una "Interfaz de Voz" para accesibilidad. Una futura expansión lógica sería integrar esta capacidad de interacción por voz directamente en el Dashboard de Escritorio, permitiendo al personal del consultorio interactuar con el sistema y la IA mediante comandos de voz para búsquedas, agendamiento de citas o activación de automatizaciones, mejorando aún más la "eficiencia operativa".
• Personalización y Monitoreo Proactivo de Pacientes (con el AVI): El Asistente Virtual Inteligente (AVI) en WhatsApp podría expandir sus capacidades más allá de FAQs y agendamiento. Podría enviar mensajes personalizados de recordatorio para toma de medicación, o solicitar actualizaciones de estado de forma proactiva, basándose en el plan de tratamiento del paciente.
• Soporte de Desarrollo y Mantenimiento: La IA ya puede generar "scripts SQL básicos a partir de descripciones en lenguaje natural" y colaborar en "generación de preguntas frecuentes (FAQs)" para el manual de usuario y "crear un chatbot de ayuda inicial para la

capacitación del personal". Estas capacidades podrían ampliarse para incluir la asistencia en la depuración de código, la generación de casos de prueba o incluso la sugerencia de mejoras en la arquitectura del sistema, acelerando los ciclos de desarrollo y mantenimiento.

La implementación de estas futuras funcionalidades dependerá de la evolución de las necesidades del consultorio, la madurez de la tecnología de IA, y, crucialmente, el cumplimiento de las normativas de privacidad y seguridad de datos, dada la naturaleza sensible de la información manejada. La flexibilidad de la arquitectura de nube de GCP y la integración modular de la IA a través de APIs facilitan estas expansiones.

Lista de Documentación Esencial y Estructurada para el Desarrollo del Dashboard de Administración (Fase 1) - DETALLADA:

Documento de Requisitos Funcionales Detallados del Dashboard:

Propósito Específico: El Dashboard es el centro de operaciones principal para el personal médico y administrativo, permitiendo la gestión centralizada de la información de pacientes.
Componentes de la Pantalla:
Barra de Navegación Lateral (Izquierda):
Identidad de la Aplicación: Logo (Icono circular) y Texto "CIRUGIA ESPECIAL".
Navegación Principal (Iconos): Inicio (Icono de casa), Registros (Icono de documento), Colaboradores/Equipo (Icono de usuarios), Reportes (Icono de gráfico de barras), Integraciones/Módulos (Icono de cuadrícula).
Sección "Mis Archivos": Barra de Búsqueda (Campo de texto con icono de lupa), Estructura de Carpetas (Sistema de árbol), Carpetas Visibles (General, Operaciones, Consultas, Laboratorio), Funcionalidad de Filtrado por Carpeta.
Sección "Automatizaciones": Descripción y placeholders.
Header Superior (Barra Superior):
Accesos Rápidos a Servicios (Iconos de aplicaciones, interpretado como accesos a Google Drive, SharePoint, herramientas de colaboración).
Controles de Usuario (Derecha): Búsqueda Global (Icono de lupa), Notificaciones (Icono de campana), Fecha y Hora (Muestra actual), Perfil de Usuario (Avatar con menú desplegable: Mi Perfil, Configuración, Cerrar Sesión).
Área de Contenido Principal:
Módulo "Base de Datos":
Título ("Base de Datos").
Subtítulo ("Registro de pacientes Cirugía Ecensial 2024").
Barra de Acciones: Eliminar (Icono papelera - borrado suave), Filtros (Icono filtro - panel/modal con opciones avanzadas), Exportar (CSV/Excel), + Nuevo Registro (Botón principal).
Tabla de Datos: Columnas (Checkbox de Selección, ID/Nombre del Paciente - negrita y enlace a detalles, Columnas de Datos Adicionales - texto regular, Estado - píldora de color Active/Inactive, Menú de Acciones - icono tres puntos con opciones: Ver Detalles, Editar Registro, Ver Historial Clínico Completo, Archivar, Eliminar Suave).
Módulo "Asistente virtual":
Título ("Asistente virtual").
Funcionalidad: Interacción por lenguaje natural, Búsqueda Inteligente (ej. buscar archivos, recordar nombres), Recuperación y Acciones Contextuales (ej. enviar por correo, descargar), Automatización de Tareas (ej. enviar correos, generar borradores, crear proyectos).
Módulo "Notas":
Título ("Notas").
Funcionalidad: Tomar notas personales/recordatorios, Posible función "adjuntar nota a paciente".
Módulo "Visor de Archivos Visuales":
Funcionalidad: Visualizar imágenes/videos relacionados con pacientes, Actualización contextual según paciente/búsqueda.
Componentes: Carrusel de Vistas Previas (miniaturas, navegación con flechas), Filtros de Origen/Tipo (Cloud Drives, Photos, Videos).
Flujos de Usuario Clave Detallados:
Añadir un nuevo paciente (pasos completos).
Buscar el historial de un paciente (interacción con Asistente Virtual).
Archivar múltiples pacientes (selección, acción Eliminar, modal de confirmación, cambio de estado).
Ver Detalles de Paciente (desde la tabla).
Editar Registro de Paciente (desde el menú de acciones).
Ver Historial Clínico Completo (desde el menú de acciones).
Archivar un solo paciente (desde el menú de acciones).
Eliminar Suave un solo paciente (desde el menú de acciones).
Filtrar Tabla de Datos (uso del icono de filtro y opciones avanzadas).
Exportar Datos de la Tabla.

Interactuar con el Asistente Virtual para tareas específicas.
Tomar y guardar notas en el widget "Notas".
Navegar y filtrar archivos en el "Visor de Archivos Visuales".
Log In y Log Out de usuario.
Acceder a "Mi Perfil" y "Configuración".
Requisitos No Funcionales: Rendimiento adecuado con 2TB de datos (considerando que gran parte estará en Cloud Storage), Usabilidad intuitiva, Seguridad (protección de datos sensibles, control de acceso), Fiabilidad, Escalabilidad.
Documento de Diseño de la Base de Datos (Cloud SQL) - Detallado:

Modelo Entidad-Relación (ERD) Completo: Incluyendo todas las entidades identificadas en los requisitos funcionales (Pacientes, Historial_Clinico, Citas, Documentos, Usuarios, Roles, Logs_Auditoria, Procedimientos, Procedimientos_Citas) y sus relaciones con cardinalidad definida.
Especificación Completa de Tablas:
Por cada tabla: Nombre exacto, Lista de columnas (Nombre, Tipo de Dato - ej. VARCHAR(255), INT, BOOLEAN, DATE, DATETIME, JSON, TEXT), Restricciones (NOT NULL, UNIQUE), Clave Primaria, Claves Foráneas (referenciando a qué tabla y columna).
Diagrama Relacional Detallado.
Estrategia de Indexación: Definición de índices en columnas clave para búsquedas y filtros frecuentes (ej. en nombre de paciente, fechas de citas).
Plan de Migración de Datos Existentes: Proceso para cargar los archivos digitales iniciales (metadatos a la base de datos, archivos a Cloud Storage).
Especificaciones de Seguridad de la Base de Datos: Configuración de usuarios y permisos de acceso, Requisitos de cifrado en reposo (manejo de claves), Plan de copias de seguridad y restauración.
Elección y Configuración Inicial de Cloud SQL: Decisión entre MySQL/PostgreSQL y los pasos para configurar la instancia inicial en GCP.
Documento de Especificación de la Arquitectura del Backend y Diseño de APIs - Detallado:

Diagrama de Arquitectura del Sistema para la Fase 1: Mostrando la interacción entre la Aplicación de Escritorio (Frontend), API Gateway/Load Balancer, Servidor de Aplicación (Backend), Cloud SQL, Cloud Storage, Servicio de Logs, y la Integración con IA.
Diseño Completo de Endpoints de la API (RESTful):
Para Gestión de Pacientes: GET /patients (con parámetros para búsqueda, filtrado, paginación, ordenamiento), GET /patients/{id}, POST /patients, PUT /patients/{id}, DELETE /patients/{id} (para borrado suave), POST /patients/{id}/archive.
Para Gestión de Historial Clínico: GET /patients/{id}/history, POST /patients/{id}/history, PUT /history/{id}, DELETE /history/{id} (borrado suave si aplica).
Para Gestión de Citas: GET /appointments (con filtros por fecha, paciente, estado), GET /appointments/{id}, POST /appointments, PUT /appointments/{id}, DELETE /appointments/{id} (borrado suave).
Para Gestión Documental: POST /patients/{id}/documents (subida de archivo), GET /patients/{id}/documents, GET /documents/{id} (para descargar/visualizar), DELETE /documents/{id} (borrado suave).
Para Autenticación y Autorización: POST /login, Endpoints para gestionar usuarios (si se incluye en Fase 1), Endpoints para verificar permisos por rol.
Para Integración con IA: Endpoints específicos para interactuar con el modelo (ej. POST /ai/generate-email, POST /ai/summarize, POST /ai/chat).
Para Logs de Auditoría: Endpoints para registrar acciones clave.
Especificación de Solicitudes y Respuestas: Estructura JSON esperada para cada endpoint (ejemplos de request body y response body).
Manejo de Errores: Definición de códigos de estado HTTP estándar (200 OK, 201 Created, 400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found, 500 Internal Server Error) y estructura de los mensajes de error.
Implementación de Autenticación (ej. tokens JWT) y Autorización (verificar roles y permisos en cada endpoint).
Tecnología del Backend: Elección del lenguaje (Python/Node.js) y framework (Flask/Django/Express) y justificación.
Configuración de Servicios de GCP: Detalle de la configuración de Compute Engine/Cloud Run, API Gateway/Load Balancer, Cloud Storage Buckets, Cloud Logging.
Documento de Especificación Técnica del Frontend del Dashboard:

Elección del Framework Frontend (React, Vue, Angular) y justificación.
Estructura de la Aplicación: Organización de directorios, módulos y componentes.
Diseño de Componentes Reutilizables: Especificación de los componentes de UI basados en la guía de estilo "Glassmorphism Oscuro" (botones, inputs, tablas, modales, tarjetas/paneles, etc.) con sus propiedades (props) y estados.
Implementación del Sistema de Diseño: Cómo se aplicarán los estilos (CSS-in-JS, SCSS, etc.), uso de variables CSS (tokens de diseño). Traducción detallada de las propiedades de la guía de estilo [2] a código.
Gestión del Estado: Elección e implementación de una estrategia de gestión del estado (ej. Redux, Vuex, Context API, Zustand).
Consumo de APIs del Backend: Cómo se realizarán las llamadas a los endpoints, manejo de estados de carga y error en la UI.
Implementación de la Tabla de Datos Interactiva: Paginación, ordenamiento, filtrado del lado del cliente/servidor, selección múltiple, menú de acciones por fila.
Implementación del Asistente Virtual en la UI: Campo de entrada, visualización de respuestas, manejo de interacciones.
Integración de Widgets Externos: Cómo se incrustarán o accederán a servicios como Gmail y Google Drive.
Consideraciones Específicas de Electron (si aplica): Empaquetado, comunicación entre proceso principal y renderizador.
Diseño Responsivo (si aplica para alguna parte del Dashboard o si se piensa en el futuro).
Documento de Especificación de la Integración de Inteligencia Artificial - Detallado:

Casos de Uso de IA Específicos en el Dashboard:
Generación de borradores de correo electrónico (especificar qué información se usaría).
Generación de plantillas para hojas membretadas (especificar entrada y salida).
Generación de recordatorios automatizados (basado en fechas de citas, por ejemplo).
Chat interno: Resúmenes de texto (ej. historial clínico), Redacción de informes cortos, Respuestas a consultas rápidas basadas en datos accesibles por la IA.
Interacción con el Modelo de IA:
Especificación de las APIs del modelo Gemma 3 o Gemini (si se accede vía API externa).
Integración a través de Vertex AI (si se despliega allí).
Formato exacto de los prompts de entrada para el modelo.
Formato exacto de las respuestas esperadas del modelo.
Manejo de Respuestas de la IA en el Backend/Frontend: Cómo se procesará y presentará la información generada por la IA al usuario.
Consideraciones de Costo y Rendimiento de la IA.
Plan de Pruebas Detallado para el Dashboard:

Tipos de Pruebas: Pruebas Unitarias (por función/componente), Pruebas de Integración (frontend-backend, backend-BD, backend-IA), Pruebas Funcionales (cada funcionalidad listada en el punto 1), Pruebas de UI/Usabilidad, Pruebas de Seguridad (autenticación, autorización, inyecciones), Pruebas de Rendimiento (carga de datos en tablas grandes, respuesta de APIs).
Casos de Prueba Específicos: Por cada funcionalidad y flujo de usuario detallado en el punto 1, definir los pasos para probar, los datos de entrada, y los resultados esperados.
Pruebas de Roles y Permisos: Casos de prueba específicos para verificar que cada rol solo puede acceder y realizar las acciones permitidas.
Pruebas de Borrado Suave: Verificar que los registros se marcan como inactivos en lugar de eliminarse permanentemente.
Pruebas de Filtrado, Búsqueda y Paginación en la tabla.
Pruebas de la integración con IA: Verificar que las respuestas son relevantes y se muestran correctamente.
Criterios de Aceptación Claros: Qué significa que una prueba sea exitosa.

Plan Detallado de Pruebas para el Dashboard del Ecosistema Digital para Cirugía Especial

1. Introducción y Propósito

Este documento detalla el plan de pruebas exhaustivas para el "Dashboard de Escritorio", el cual es el "Núcleo Administrativo Central" y el entregable principal de la Fase 1 del proyecto "Ecosistema Digital Inteligente para Cirugía Especial". El Dashboard de Escritorio es la columna vertebral y la base de datos central de todo el ecosistema, y su propósito es optimizar la gestión de pacientes, mejorar la eficiencia operativa, y centralizar la información crítica del consultorio en un lugar seguro y accesible.

El objetivo de este plan es asegurar la calidad, fiabilidad, rendimiento y seguridad de la aplicación antes de su despliegue y uso en un entorno real. Las pruebas son de vital importancia para el "Ecosistema Digital Inteligente para Cirugía Especial", ya que el sistema busca ser "potente, seguro y excepcionalmente intuitivo". A través de un proceso de pruebas riguroso, se busca reducir los "procesos manuales que consumen tiempo" y el "riesgo de errores", liberando tiempo valioso para el personal.

Un enfoque primordial de este ecosistema es la "seguridad de datos", ya que maneja información sensible del paciente. Las pruebas de seguridad son cruciales para validar las múltiples capas de protección implementadas, incluyendo:

- El Control de Acceso Basado en Roles (RBAC), que garantiza que solo usuarios autorizados (Administradores, Admin Principal, Admin Secundario, Asistentes) accedan a funcionalidades e información según sus permisos.
- El cifrado de datos en tránsito a través de HTTPS (TLS 1.3) entre la aplicación de escritorio y la infraestructura en la nube.
- El almacenamiento seguro de credenciales mediante hashing y el cifrado de datos sensibles en reposo en la Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL) y el Almacenamiento de Archivos (Cloud Storage - S3/Blob).
- El Servicio de Logs de Auditoría (Cloud Logging / Audit Logs), que registra inmutablemente (Write Once, Read Many - WORM) todos los intentos de login y las actividades del usuario, siendo vital para la detección de anomalías y la respuesta a incidentes.
- La Virtual Private Cloud (VPC) y el API Gateway / Load Balancer, que actúan como la primera línea de defensa para la infraestructura en la nube.

La necesidad de una conexión a internet estable de al menos 250 Mbps para un rendimiento óptimo del ecosistema también subraya la importancia de las pruebas de rendimiento y estrés. En resumen, este plan de pruebas es un pilar fundamental para garantizar que el Dashboard cumpla con su promesa de ser un centro de control digital potente, seguro y confiable para el consultorio del Dr. Sánchez García.

2. Alcance de las Pruebas

El presente Plan Detallado de Pruebas se enfoca exclusivamente en el "Dashboard de Escritorio", también conocido como el "Núcleo Administrativo Central", el cual es el entregable principal de la Fase 1 del proyecto "Ecosistema Digital Inteligente para Cirugía Especial". Este alcance específico garantiza que la columna vertebral del ecosistema sea robusta, segura y funcional antes de proceder con fases posteriores.

2.1. Componentes y Funcionalidades Incluidas en las Pruebas (Fase 1 del Dashboard)

Las pruebas cubrirán exhaustivamente todos los módulos y flujos de usuario detallados para el Dashboard de Escritorio en la Fase 1 del proyecto. Esto incluye, pero no se limita a:

- Flujo de Acceso: Login de Usuario y Dashboard Principal:
    - Login de Usuario: Autenticación de usuarios con "Nombre de Usuario" y "Contraseña". Pruebas de inicio de sesión exitoso y fallido, incluyendo mensajes de error por credenciales incorrectas o campos vacíos. Validación de la persistencia de la sesión tras un login exitoso y la opción de "Cerrar Sesión".
    - Dashboard Principal (Vista General): Verificación de la visualización correcta de la "Cabecera" (usuario logueado y rol) y la adaptación de la "Barra de Navegación Lateral" según el rol del usuario. Pruebas de la visualización y actualización de los paneles del "Área Central del Dashboard" como "CITAS DEL DÍA", "MÉTRICAS RÁPIDAS" y "PACIENTES RECIENTES".
    - Navegación: Pruebas de la navegación a todas las secciones principales y sub-secciones a través de la barra lateral: "Pacientes", "Agenda", "Procedimientos", "Reportes", "Documentos", "Administración".
    - Widgets y Accesos Directos: Verificación del acceso rápido a Gmail, Google Drive y aplicaciones de Office a través de los widgets visibles en la interfaz.

• Gestión Integral del Expediente del Paciente:
    ◦ Creación de Perfiles de Paciente: Verificación del formulario de "Creación de Perfil de Paciente", incluyendo todos los campos demográficos y de contacto. Pruebas de validación de campos obligatorios y formatos inválidos.
    ◦ Búsqueda Avanzada de Pacientes: Pruebas de la funcionalidad de búsqueda por cualquier campo (nombre, ID, palabra clave en notas) y la visualización de resultados.
    ◦ Registro Historial Clínico Detallado: Verificación del formulario "Registro de Historial Clínico", incluyendo campos para "Fecha de Consulta", "Diagnóstico", "Notas de Evolución", "Procedimientos y Tratamientos". Pruebas de la capacidad para añadir nuevas entradas cronológicas.
• Gestión Documental Avanzada:
    ◦ Subida y Asociación de Documentos: Pruebas de la funcionalidad para subir y asociar todo tipo de archivos digitales (PDF, imágenes, videos cortos) directamente al expediente del paciente o a una entrada específica del historial clínico. Verificación de mensajes de error para tipos de archivo no soportados o tamaños excedidos.
    ◦ Visualización de Documentos: Pruebas del visor de archivos visuales con su carrusel de vistas previas y filtros por origen/tipo (Cloud Drives, Photos, Videos).
• Módulo de Agenda y Citas:
    ◦ Agendar Nueva Cita: Pruebas del formulario para agendar citas, incluyendo la selección de paciente, fecha, hora, tipo de cita y estado. Verificación de la detección de conflictos de horario.
    ◦ Gestión de Citas Existentes: Pruebas de las funcionalidades para reprogramar, cancelar y cambiar el estado de las citas existentes.
    ◦ Visualización de Calendario: Verificación del calendario interactivo y la visualización de citas del día.
• Seguridad, Roles y Accesos Rápidos:
    ◦ Gestión de Usuarios y Asignación de Roles: Pruebas de la capacidad para crear nuevas cuentas de usuario y asignar roles (Administrador, Admin Principal, Admin Secundario, Asistente, Invitado (Solo Lectura)). Esta funcionalidad será probada exhaustivamente para verificar que cada rol solo tiene acceso a sus permisos definidos.
    ◦ Acceso a Logs de Auditoría: Verificación de la funcionalidad para visualizar los registros cronológicos de actividades y cambios realizados en el sistema.
• Inteligencia y Automatización Integrada (Modelo de IA Gemma 3):
    ◦ Chat Interno Básico: Pruebas de la interacción con el chat interno para consultas rápidas, resúmenes de texto o redacción de informes (solo texto).
    ◦ Automatizaciones Básicas: Verificación de la capacidad para generar borradores de correos electrónicos, hojas membretadas y recordatorios a través de la IA.
• Capacitación Inicial:
    ◦ Generación de Material: Aunque la capacitación es una actividad programada, se probará la disponibilidad y relevancia del manual de funcionamiento y las preguntas frecuentes (FAQs) que la IA puede ayudar a generar.

2.2. Aspectos Técnicos Incluidos en las Pruebas

Las pruebas técnicas validarán la correcta interconexión y funcionamiento de los distintos componentes del ecosistema que soportan el Dashboard:

• Integración Frontend-Backend:
    ◦ Verificación de la comunicación fluida entre la aplicación de escritorio (frontend desarrollada con React/Vue.js y empaquetada con Electron.js) y el Servidor de Aplicación (backend construido con Python/Flask/Django o Node.js/Express).
    ◦ Pruebas de la exactitud y consistencia de los datos entre el frontend y el backend en todas las operaciones CRUD (Crear, Leer, Actualizar, Eliminar - suave).
    ◦ Medición de los tiempos de respuesta de las APIs RESTful.
• Integración Backend-Base de Datos:
    ◦ Verificación de la interacción del backend con la Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL).
    ◦ Pruebas de la integridad de los datos almacenados.
    ◦ Evaluación del rendimiento de las consultas a la base de datos para asegurar el acceso rápido a la información.
    ◦ Confirmación de que los datos sensibles están cifrados en reposo en la base de datos.
• Integración Backend-Cloud Storage:
    ◦ Verificación de la interacción del backend con el Almacenamiento de Archivos (Cloud Storage - S3/Blob) para la gestión de documentos digitales.
    ◦ Pruebas de las funcionalidades de subida y descarga de archivos, y su correcta

asociación con los expedientes de pacientes.
    ◦ Confirmación de que los archivos están cifrados en reposo en Cloud Storage.
• Integración Backend-IA (Gemma 3):
    ◦ Verificación de la integración del modelo de Inteligencia Artificial Gemma 3 (o similar a Gemini) en el backend mediante APIs.
    ◦ Pruebas de la precisión y relevancia de las respuestas y contenidos generados por la IA (redacción de correos, hojas membretadas, recordatorios, respuestas del chat, resúmenes).
    ◦ Evaluación del rendimiento y la latencia en las interacciones con la IA.
• Seguridad de APIs y Control de Acceso (RBAC):
    ◦ Validación de que toda la comunicación entre la aplicación de escritorio y la infraestructura en la nube se realiza exclusivamente a través de HTTPS (TLS 1.3), garantizando el cifrado de datos en tránsito.
    ◦ Pruebas exhaustivas del Control de Acceso Basado en Roles (RBAC) para asegurar que cada rol (Administrador, Admin Principal, Admin Secundario, Asistente, Invitado - Solo Lectura) tiene acceso solo a las funcionalidades y datos que le corresponden. Esto incluye intentos de acceso no autorizado y verificación de mensajes de error adecuados.
    ◦ Confirmación de que las credenciales de usuario se almacenan de forma segura mediante hashing y cifrado en reposo.
    ◦ Verificación de que el API Gateway / Load Balancer actúan como la primera línea de defensa para el backend.
• Registro y Auditoría (Logs de Auditoría):
    ◦ Verificación de que todos los intentos de login (exitosos y fallidos) y cada acción relevante del usuario (ej., modificación de datos de paciente, creación de citas, gestión de usuarios) se registran cronológicamente en el "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)".
    ◦ Confirmación de la inmutabilidad de los logs (Write Once, Read Many - WORM) para asegurar su integridad y trazabilidad.
• Manejo de Errores y Estados Vacíos:
    ◦ Pruebas de todos los escenarios de error especificados (credenciales incorrectas, campos vacíos/formato inválido, problemas de conectividad) y la visualización de mensajes de error claros y concisos.
    ◦ Verificación del comportamiento del sistema cuando no hay datos para mostrar en ciertas secciones (ej., "No se encontraron pacientes", "No hay citas programadas para hoy").
• Rendimiento y Conectividad:
    ◦ Pruebas del rendimiento general de la aplicación bajo una conexión a internet estable de al menos 250 Mbps, según la recomendación para un óptimo funcionamiento del ecosistema. Esto incluirá la carga de datos, la navegación y la interacción con funcionalidades clave.
• Diseño Visual y Usabilidad (Glassmorphism Oscuro):
    ◦ Verificación de la implementación del estilo visual "Glassmorphism Oscuro" en todas las pantallas y componentes del Dashboard, asegurando la coherencia visual integral con los principios de diseño definidos (fondo gris carbón, paneles de vidrio esmerilado, efecto "Aurora", tipografía Inter, bordes sutiles y luminosos).
    ◦ Pruebas de la legibilidad óptima y el doble enfoque de contraste para la accesibilidad.
    ◦ Verificación de la retroalimentación interactiva al pulsar botones (aumento de brillo o ligero hundimiento).

2.3. Exclusiones Explícitas de las Pruebas en esta Fase del Dashboard

Es fundamental entender que este plan de pruebas se limita estrictamente a la Fase 1: El Núcleo Administrativo (Dashboard de Escritorio). Por lo tanto, las siguientes funcionalidades y componentes del ecosistema no se incluirán en las pruebas de esta fase, ya que corresponden a etapas futuras del proyecto o no forman parte del desarrollo principal:

• Componentes de la Fase 2: Expansión a la Movilidad y Presencia Web:
    ◦ La Aplicación Móvil para el equipo.
    ◦ La Página web profesional del consultorio (incluyendo su diseño "Luxury, Tech, Accesible", secciones informativas, asistente virtual para FAQ y agendamiento, y pasarela de pagos).
• Componentes de la Fase 3: Innovación y Experiencia del Paciente:
    ◦ La Aplicación Móvil exclusiva para Clientes/Pacientes (incluyendo acceso a registro médico, agendamiento, material educativo, interfaz de voz, mensajería segura, pasarela de

pagos).
    ◦ La Integración del Asistente Virtual Inteligente (AVI) en WhatsApp (incluyendo su sincronización con la web, FAQ, agendamiento y notificaciones automatizadas).
• Otros Aspectos Excluidos del Alcance del Proyecto o del Desarrollo Directo:
    ◦ La digitalización (escaneo) de archivos físicos existentes.
    ◦ Los servicios de marketing digital, gestión de redes sociales o campañas de promoción.
    ◦ La compra de dominio y contratación del servicio de hosting para la página web, ya que son responsabilidad del cliente.
    ◦ Cualquier funcionalidad o solicitud de cambio que no esté explícitamente detallada en el alcance definido para la Fase 1 del Dashboard.
Aunque el backend y la infraestructura en la nube están diseñados para soportar futuras fases, el enfoque de estas pruebas estará en la interacción del Dashboard de Escritorio con estos componentes, sin extenderse a la validación de las funcionalidades o interfaces de las aplicaciones móviles o la página web en sí.
Tipos de Pruebas
Pruebas Unitarias
• Propósito: Las pruebas unitarias tienen como objetivo verificar que unidades individuales de código, como funciones, métodos o componentes pequeños, funcionan correctamente de forma aislada. Esto permite identificar y corregir defectos en las etapas tempranas del ciclo de desarrollo, reduciendo costos y complejidad.
• Aplicación:
    ◦ En el Backend: Se aplicarán en módulos clave de la lógica de negocio, como las funciones de validación de datos (ej., validación de entrada para nombres de usuario, contraseñas, o campos de paciente), la lógica de autenticación y autorización basada en roles (verificando que un usuario con un rol específico tenga o no acceso a ciertas funcionalidades), y la interacción básica con APIs externas (como las llamadas a la API de Gemma 3 para automatizaciones).
    ◦ En el Frontend: Se realizarán en componentes reutilizables como botones de acción ("Iniciar Sesión", "Guardar Paciente", "Agendar Cita"), campos de entrada, elementos de navegación, y la lógica de manipulación local de datos o estados de la UI.
• Enfoque: Se centran en la corrección interna del código y su funcionamiento aislado, sin depender de otros módulos o servicios.
Pruebas de Integración
• Propósito: Las pruebas de integración buscan verificar la correcta interacción y comunicación entre diferentes módulos o servicios que componen el ecosistema. Esto es crucial para un sistema distribuido como este, que depende de la nube y de múltiples componentes interconectados.
• Aplicación:
    ◦ Frontend-Backend API Communication: Se probará la comunicación entre la aplicación de escritorio y el Servidor de Aplicación (Backend) a través de APIs RESTful, asegurando que los datos se envíen y reciban correctamente utilizando HTTPS (TLS 1.3) para cifrar los datos en tránsito.
    ◦ Backend-Base de Datos (Cloud SQL): Verificación de que el backend interactúa correctamente con la Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL) para almacenar, recuperar y actualizar datos de pacientes, historiales clínicos, citas, usuarios y roles. Esto incluye el almacenamiento seguro y cifrado en reposo.
    ◦ Backend-Cloud Storage: Comprobar que el backend puede subir, descargar y asociar archivos digitales (PDFs, imágenes, videos cortos) a Cloud Storage de forma segura y cifrada.
    ◦ Backend-IA (Gemma 3): Asegurar que las llamadas del backend a la API del modelo de Inteligencia Artificial Gemma 3 (o similar a Gemini) para funciones de automatización y el chat interno se realicen de manera eficiente y devuelvan los resultados esperados.
• Enfoque: Garantizar que los datos se transfieran correctamente entre los componentes y que las interfaces (APIs) funcionen según lo esperado, sin errores de comunicación o formato.
Pruebas Funcionales (End-to-End)
• Propósito: Estas pruebas verifican que cada funcionalidad completa del Dashboard, desde la interacción del usuario en el frontend hasta la acción en el backend y la base de datos, funciona de acuerdo con los requisitos de negocio. Simulan el uso real del sistema por parte del personal del consultorio.
• Aplicación: Se enfocarán en probar cada flujo de usuario clave detallado en los documentos proporcionados, especialmente en el "Mapa Detallado: Recorrido del Usuario en

el Dashboard de Escritorio".
    ◦ Flujo de Login: Desde el ingreso de credenciales hasta la redirección exitosa al Dashboard principal y la correcta visualización de las secciones según el rol del usuario.
    ◦ Gestión de Pacientes: Creación de perfiles, búsqueda avanzada, y el registro cronológico de historial clínico detallado con la capacidad de adjuntar documentos.
    ◦ Gestión Documental: Subida, asociación y visualización de archivos digitales (PDF, imágenes, videos cortos) directamente en el expediente del paciente.
    ◦ Módulo de Agenda y Citas: Programación, reprogramación y gestión de citas con estados personalizables, incluyendo la visualización de citas del día.
    ◦ Gestión de Usuarios y Roles: Creación de nuevas cuentas de usuario y asignación de roles con permisos específicos, así como el acceso (para Admin Principal) a los logs de auditoría.
    ◦ Interacciones de IA y Automatización: Uso del chat interno para consultas rápidas, resúmenes de texto o redacción de informes, y activación de automatizaciones básicas como redacción de correos o generación de hojas membretadas.
• Enfoque: Validar que el sistema completo, desde el punto de vista del usuario final, cumple con todas las especificaciones funcionales y de negocio. Se confirmará que los datos se transfieren y persisten correctamente en cada paso del flujo, y que las respuestas del sistema son las esperadas en diferentes escenarios (éxito, error, estados vacíos).

Pruebas de UI/Usabilidad

• Propósito: Estas pruebas se centran en verificar que la interfaz de usuario es intuitiva, fácil de navegar, visualmente consistente y que los elementos interactivos se comportan como se espera. También buscan asegurar la comodidad visual y la legibilidad.
• Aplicación: Se aplicarán a toda la interfaz de usuario del Dashboard, desde la pantalla de Login hasta todas las secciones y paneles del Dashboard Principal, siguiendo rigurosamente el estilo visual "Glassmorphism Oscuro".
• Enfoque:
    ◦ Experiencia del Usuario: Evaluar la facilidad de interacción con componentes como botones, formularios, tablas y menús.
    ◦ Navegación: Asegurar que la navegación entre vistas sea fluida e intuitiva.
    ◦ Coherencia Visual: Confirmar que el diseño "Glassmorphism Oscuro", con su fondo gris carbón (#121212), paneles flotantes de "vidrio esmerilado" (backdrop-filter: blur(24px)), auras de color (azul #4A69FF y violeta #8C52FF), texto nítido y brillante (#F5F5F5), y bordes sutiles y luminosos, se aplica de manera consistente en toda la interfaz.
    ◦ Accesibilidad: Verificar que la legibilidad del texto sea óptima (contraste > 4.5:1 WCAG) y que el diseño sea accesible, especialmente considerando a pacientes mayores o con dificultades tecnológicas, como se busca en el ecosistema general.
    ◦ Retroalimentación Interactiva: Probar que la interacción al pulsar botones (sutil aumento de brillo o ligero hundimiento) proporcione una retroalimentación clara e inmediata al usuario.

Pruebas de Seguridad

• Propósito: Las pruebas de seguridad tienen como propósito identificar vulnerabilidades y asegurar que la aplicación protege los datos sensibles y controla el acceso adecuadamente. Es crucial para abordar las "preocupaciones de seguridad de datos".
• Enfoque:
    ◦ Autenticación (Login con credenciales válidas/inválidas): Se probará rigurosamente la pantalla de Login, que es el punto de entrada fundamental al "Ecosistema Digital Inteligente para Cirugía Especial".
        ▪ Se verificará que solo los usuarios autorizados (Administradores, Admin Principal, Admin Secundario, Asistentes) puedan acceder al sistema.
        ▪ Las credenciales se enviarán de forma segura al Servidor de Aplicación (Backend) alojado en Google Cloud Platform (GCP), utilizando HTTPS (TLS 1.3) para cifrar los datos en tránsito, protegiéndolos de interceptaciones.
        ▪ El backend verificará las credenciales contra la Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL), donde los datos sensibles como contraseñas se almacenan cifrados en reposo y mediante funciones de hash (indicado por hash_password VARCHAR).
        ▪ En caso de credenciales incorrectas, se espera que la aplicación permanezca en la pantalla de Login y muestre un mensaje de error claro y conciso, como "Usuario o contraseña incorrectos". El Servicio de Logs de Auditoría registrará este intento fallido.
        ▪ Se probarán escenarios de campos vacíos o formatos inválidos, esperando mensajes de validación como "Campo 'Nombre de Usuario' requerido" o "Campo 'Contraseña' requerido".
    ◦ Autorización (Verificar que cada rol solo tiene acceso a las funcionalidades y datos

permitidos): El sistema impone un Control de Acceso Basado en Roles (RBAC) estricto.
        ▪ Se probará cada rol ("Admin Principal", "Admin Secundario", "Asistente", "Invitado (Solo Lectura)") para asegurar que sus permisos sobre la información y funcionalidades del Dashboard se aplican correctamente.
        ▪ Por ejemplo, se verificará que solo el "Admin Principal" tenga acceso a la "Gestión de Usuarios" y a la visualización de los "Logs de Auditoría".
        ▪ Se asegurará que un "Asistente" pueda gestionar pacientes, citas y documentos, pero no tenga acceso a la gestión de usuarios o logs.
    ◦ Validación de entrada para prevenir ataques (ej. inyección SQL, XSS): Aunque las fuentes no especifican ataques concretos, se menciona la validación de entrada para campos obligatorios y formatos válidos en formularios (ej., creación de perfil de paciente, login). Esto implica que se implementarán controles a nivel de backend y frontend para sanear y validar las entradas de usuario, lo cual es fundamental para mitigar ataques de inyección (SQL, XSS) al interactuar con la Base de Datos Relacional (Cloud SQL) y al mostrar contenido en la interfaz. El uso de frameworks como Python/Flask/Django o Node.js/Express facilita la implementación de estas defensas.
    ◦ Seguridad en el manejo de archivos subidos: Se probará la funcionalidad de "Gestión Documental Avanzada" que permite subir, asociar y visualizar archivos digitales (PDFs, imágenes, videos cortos) a Cloud Storage. Se verificará que estos archivos se almacenen de forma segura y cifrados en reposo en Cloud Storage. Las pruebas incluirán la validación de tipos de archivo soportados y límites de tamaño para prevenir la subida de contenido malicioso.
    ◦ Seguridad de las APIs: Todas las comunicaciones entre la aplicación de escritorio y el Servidor de Aplicación (Backend) se realizan a través de APIs RESTful utilizando HTTPS (TLS 1.3). El API Gateway / Load Balancer actuará como el punto de entrada seguro y primera línea de defensa para todas las solicitudes, distribuyendo el tráfico y gestionando las conexiones de forma segura. Se probará la robustez de estos endpoints frente a intentos de acceso no autorizado y manipulación de datos.

Pruebas de Rendimiento

• Propósito: Medir la velocidad, capacidad de respuesta y estabilidad de la aplicación bajo diferentes cargas. El objetivo del ecosistema es optimizar la gestión de pacientes y mejorar la eficiencia operativa.
• Enfoque:
    ◦ Probar el tiempo que tarda en cargar la tabla de datos con un volumen grande de registros: El Dashboard Principal muestra tablas de datos dinámicos como "Pacientes Recientes". Se simulará un alto volumen de pacientes para asegurar que la carga de estos listados sea eficiente y no afecte la experiencia del usuario.
    ◦ El tiempo de respuesta de las APIs críticas: Se evaluará la velocidad de respuesta de las APIs más utilizadas y fundamentales, tales como:
        ▪ Búsqueda de pacientes: La "Búsqueda Avanzada" permite la búsqueda instantánea por cualquier campo. La API GET /patients se utilizará para estas pruebas.
        ▪ Obtención de historial clínico: La visualización y el registro del "Historial Clínico Detallado" son funcionalidades centrales.
        ▪ El backend procesa las solicitudes, interactúa con la Base de Datos Relacional (Cloud SQL) y el Almacenamiento de Archivos (Cloud Storage), y envía la información actualizada de vuelta al frontend.
    ◦ El rendimiento de la interfaz al navegar o interactuar con componentes complejos: Se probará la fluidez de la navegación entre las diferentes secciones del Dashboard (Pacientes, Agenda, etc.), así como la interacción con elementos que requieren mayor procesamiento, como los formularios de creación o edición, y el calendario interactivo del "Módulo de Agenda y Citas". El diseño "Glassmorphism Oscuro" busca una experiencia de usuario fluida y de alta calidad, lo que implica un rendimiento de UI óptimo.
    ◦ El impacto del uso del Visor de Archivos Visuales: Se probará cómo el sistema maneja la visualización de archivos digitales (imágenes, videos cortos) asociados a los expedientes de los pacientes, especialmente al navegar por galerías con múltiples elementos. Se evaluará el tiempo de carga y la capacidad de respuesta del visor bajo diferentes condiciones.
    ◦ Considerar el impacto del volumen proyectado de 2TB de datos (principalmente en Cloud Storage) en el rendimiento general: El sistema se apoya en una arquitectura de nube robusta y escalable en GCP. El Almacenamiento de Archivos (Cloud Storage) está diseñado para almacenar grandes volúmenes de archivos digitales y se menciona una capacidad incluida de hasta 1 Terabyte (1000 GB) mensual como cortesía inicial, con costos adicionales para volúmenes mayores. El rendimiento dependerá en gran medida de una

conexión a internet estable de al menos 250 Mbps, que es un requisito de conectividad para el ecosistema.
Pruebas de Regresión
• Propósito: Asegurar que los cambios o nuevas funcionalidades no han introducido errores en partes existentes del sistema.
• Cuándo se ejecutarán: Después de cada nueva implementación significativa o corrección de errores.
    ◦ El "Núcleo Administrativo Central" o Dashboard de Escritorio es la columna vertebral y la base de datos central de todo el ecosistema, desarrollado en la Fase 1. Cualquier cambio en esta fase fundacional, como la implementación de la "Inteligencia y Automatización Integrada" con Gemma 3 o la "Gestión de Usuarios y Roles", requerirá pruebas de regresión.
    ◦ El proyecto se ejecuta en tres fases secuenciales, donde cada fase construye sobre la anterior. Esto implica que las funcionalidades desarrolladas en fases posteriores (como la Página Web Profesional en Fase 2 o la Aplicación Móvil para Pacientes en Fase 3) que interactúan con el backend centralizado y la base de datos de la Fase 1, necesitarán pruebas de regresión para garantizar la compatibilidad y estabilidad del sistema integral.
    ◦ El "Plan de Mantenimiento y Soporte" mensual de $6,000.00 MXN incluye "actualizaciones de seguridad y la evolución continua de la aplicación". Esta evolución continua, que implica nuevas funcionalidades o mejoras, requerirá un ciclo constante de pruebas de regresión para mantener la integridad del sistema a lo largo del tiempo.
Casos de Prueba Detallados:
1. Pruebas de Login y Acceso
ID: CP-LOGIN-001
• Descripción Breve del Escenario que se Prueba: Inicio de sesión exitoso de un usuario con rol "Admin Principal".
• Precondiciones:
    ◦ La aplicación de escritorio del "Núcleo Administrativo Central" está instalada y ejecutándose.
    ◦ Existe un usuario de prueba en la base de datos con el rol "Admin Principal" y credenciales válidas.
    ◦ Conexión a internet estable de al menos 250 Mbps.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Abrir la aplicación de escritorio.
    2. En la "Pantalla: Login de Usuario", ingresar el "Nombre de Usuario" y "Contraseña" válidos del Admin Principal en los campos designados.
    3. Hacer clic en el botón "Iniciar Sesión".
• Datos de Entrada de Prueba:
    ◦ Nombre de Usuario: admin.principal
    ◦ Contraseña: PasswordSeguro123
• Resultado Esperado:
    ◦ Las credenciales se envían al backend de forma segura mediante HTTPS (TLS 1.3).
    ◦ El backend verifica las credenciales con la Base de Datos Relacional (Cloud SQL) donde las contraseñas están hasheadas y cifradas en reposo.
    ◦ El "Servicio de Logs de Auditoría" registra un intento de login exitoso.
    ◦ El sistema redirecciona al usuario al "Dashboard Principal (Vista General)".
    ◦ La cabecera del dashboard muestra "Bienvenido, Dr. Admin Principal (Admin)".
    ◦ La barra de navegación lateral presenta todas las secciones, incluyendo "Administración (Gestionar Usuarios, Ver Logs de Auditoría)".
• Criterios de Éxito/Fallo: La prueba pasa si el usuario es redirigido al Dashboard principal con el rol correcto y acceso a todas las funcionalidades esperadas para un "Admin Principal"; de lo contrario, falla.
ID: CP-LOGIN-002
• Descripción Breve del Escenario que se Prueba: Intento de inicio de sesión con credenciales incorrectas.
• Precondiciones:
    ◦ La aplicación de escritorio está instalada y ejecutándose.
    ◦ Conexión a internet estable de al menos 250 Mbps.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Abrir la aplicación de escritorio.
    2. En la "Pantalla: Login de Usuario", ingresar un "Nombre de Usuario" existente y una "Contraseña" incorrecta.
    3. Hacer clic en el botón "Iniciar Sesión".

• Datos de Entrada de Prueba:
    ◦ Nombre de Usuario: admin.principal
    ◦ Contraseña: ContrasenaIncorrecta
• Resultado Esperado:
    ◦ El backend rechaza las credenciales después de verificarlas con la base de datos.
    ◦ El "Servicio de Logs de Auditoría" registra un intento de login fallido.
    ◦ La aplicación permanece en la "Pantalla: Login de Usuario".
    ◦ Se muestra un mensaje de error claro y conciso: "Usuario o contraseña incorrectos".
• Criterios de Éxito/Fallo: La prueba pasa si la aplicación permanece en la pantalla de Login y muestra el mensaje de error "Usuario o contraseña incorrectos"; de lo contrario, falla.

ID: CP-LOGIN-003
• Descripción Breve del Escenario que se Prueba: Intento de inicio de sesión dejando campos vacíos.
• Precondiciones:
    ◦ La aplicación de escritorio está instalada y ejecutándose.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Abrir la aplicación de escritorio.
    2. En la "Pantalla: Login de Usuario", dejar el campo "Nombre de Usuario" vacío y/o el campo "Contraseña" vacío.
    3. Hacer clic en el botón "Iniciar Sesión".
• Datos de Entrada de Prueba:
    ◦ Nombre de Usuario: (vacío)
    ◦ Contraseña: (vacío)
• Resultado Esperado:
    ◦ La aplicación permanece en la "Pantalla: Login de Usuario".
    ◦ Se muestran mensajes de validación para los campos requeridos: "Campo 'Nombre de Usuario' requerido" y/o "Campo 'Contraseña' requerido".
    ◦ No se realiza una llamada al backend si la validación es a nivel de frontend.
• Criterios de Éxito/Fallo: La prueba pasa si la aplicación permanece en la pantalla de Login y muestra los mensajes de validación para campos vacíos; de lo contrario, falla.

ID: CP-LOGIN-004
• Descripción Breve del Escenario que se Prueba: Intento de inicio de sesión sin conexión a internet.
• Precondiciones:
    ◦ La aplicación de escritorio está instalada y ejecutándose.
    ◦ La computadora no tiene conexión a internet.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Desconectar la computadora de internet.
    2. Abrir la aplicación de escritorio.
    3. En la "Pantalla: Login de Usuario", ingresar el "Nombre de Usuario" y "Contraseña" (válidos o inválidos) en los campos designados.
    4. Hacer clic en el botón "Iniciar Sesión".
• Datos de Entrada de Prueba:
    ◦ Nombre de Usuario: cualquier_usuario
    ◦ Contraseña: cualquier_password
• Resultado Esperado:
    ◦ La aplicación intenta comunicarse con el backend en la nube pero falla debido a la falta de conectividad.
    ◦ La aplicación permanece en la "Pantalla: Login de Usuario".
    ◦ Se muestra un mensaje de error general sobre problemas de conexión o indisponibilidad del servicio (ej., "No se pudo conectar al servidor. Por favor, intente de nuevo más tarde.").
• Criterios de Éxito/Fallo: La prueba pasa si la aplicación muestra un mensaje de error de conectividad y permanece en la pantalla de Login; de lo contrario, falla.

2. Pruebas de Gestión Integral del Expediente del Paciente

ID: CP-PACIENTES-001
• Descripción Breve del Escenario que se Prueba: Creación de un nuevo perfil de paciente con datos completos y válidos.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Pacientes" en la "Barra de Navegación Lateral".

    2. Hacer clic en la sub-opción "Nuevo Paciente".
    3. En el "Formulario de Creación de Perfil de Paciente", llenar todos los campos obligatorios y opcionales con datos válidos.
    4. Hacer clic en el botón "Guardar Paciente".
• Datos de Entrada de Prueba:
    ◦ Nombre: Juan
    ◦ Apellido: Pérez García
    ◦ Fecha de Nacimiento: 1985-07-15
    ◦ Teléfono: 5512345678
    ◦ Email: juan.perez@example.com
    ◦ Dirección: Calle Falsa 123
• Resultado Esperado:
    ◦ Se envía una solicitud POST al backend para crear el nuevo paciente.
    ◦ El nuevo perfil de paciente se crea y se guarda en la "Base de Datos Relacional (Cloud SQL)".
    ◦ La tabla "Pacientes Recientes" en el Dashboard Principal se actualiza para incluir al nuevo paciente.
    ◦ El "Servicio de Logs de Auditoría" registra la creación del nuevo paciente.
• Criterios de Éxito/Fallo: La prueba pasa si el paciente se crea correctamente, se visualiza en la tabla de pacientes recientes y se registra la acción en los logs de auditoría; de lo contrario, falla.
ID: CP-PACIENTES-002
• Descripción Breve del Escenario que se Prueba: Intento de creación de un nuevo perfil de paciente con campos obligatorios vacíos.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Pacientes" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Nuevo Paciente".
    3. En el "Formulario de Creación de Perfil de Paciente", dejar el campo "Nombre" vacío y/o el campo "Apellido" vacío.
    4. Hacer clic en el botón "Guardar Paciente".
• Datos de Entrada de Prueba:
    ◦ Nombre: (vacío)
    ◦ Apellido: Pérez García
    ◦ Fecha de Nacimiento: 1985-07-15
    ◦ Teléfono: 5512345678
    ◦ Email: juan.perez@example.com
• Resultado Esperado:
    ◦ La aplicación permanece en el formulario de "Creación de Perfil de Paciente".
    ◦ Se muestra un mensaje de validación para el campo faltante: "Campo 'Nombre' requerido".
    ◦ No se realiza una solicitud de creación al backend.
• Criterios de Éxito/Fallo: La prueba pasa si la aplicación muestra el mensaje de validación y no permite guardar el paciente; de lo contrario, falla.
ID: CP-PACIENTES-003
• Descripción Breve del Escenario que se Prueba: Búsqueda avanzada de un paciente existente por nombre.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal", "Admin Secundario", "Asistente" o "Invitado (Solo Lectura)".
    ◦ Existe al menos un paciente de prueba en la base de datos.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Pacientes" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Buscar Paciente".
    3. En el campo de búsqueda, ingresar el nombre completo o parcial de un paciente existente.
    4. Hacer clic en el botón "Buscar".
• Datos de Entrada de Prueba:
    ◦ Criterio de Búsqueda: Laura Vargas
• Resultado Esperado:
    ◦ Se envía una solicitud GET al backend (ej. GET /patients?query=Laura Vargas).
    ◦ La "Búsqueda de Pacientes" muestra una "Tabla de resultados" con los pacientes que

coinciden con el criterio de búsqueda.
    ◦ Las opciones de "Acción" ("Ver Historial", "Nueva Cita") están disponibles para los resultados.
• Criterios de Éxito/Fallo: La prueba pasa si la tabla de resultados muestra correctamente los pacientes coincidentes; de lo contrario, falla.
ID: CP-PACIENTES-004
• Descripción Breve del Escenario que se Prueba: Registro de una nueva entrada en el historial clínico detallado de un paciente.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".
    ◦ Existe un paciente al que se le registrará el historial.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Buscar un paciente existente (ej. CP-PACIENTES-003) y hacer clic en "Ver Historial" desde la tabla de resultados o "Pacientes Recientes".
    2. Desde la "Ficha del Paciente", hacer clic en "Añadir a Historial".
    3. En el "Formulario de Registro de Historial Clínico", completar los campos "Fecha de Consulta", "Diagnóstico" y "Notas de Evolución".
    4. Hacer clic en el botón "Guardar Historial".
• Datos de Entrada de Prueba:
    ◦ Fecha de Consulta: 2024-07-20
    ◦ Diagnóstico: Fractura de peroné
    ◦ Notas de Evolución: Paciente presenta dolor leve, se indica reposo y antiinflamatorios.
• Resultado Esperado:
    ◦ Se envía una solicitud al backend para guardar la nueva entrada del historial.
    ◦ La nueva entrada se guarda cronológicamente en el "Historial Clínico Detallado" del paciente en la "Base de Datos Relacional (Cloud SQL)".
    ◦ El "Servicio de Logs de Auditoría" registra la acción de modificación de datos del paciente.
• Criterios de Éxito/Fallo: La prueba pasa si la nueva entrada del historial clínico aparece correctamente en la ficha del paciente y se registra la acción en los logs de auditoría; de lo contrario, falla.
3. Pruebas de Gestión Documental Avanzada
ID: CP-DOCUMENTOS-001
• Descripción Breve del Escenario que se Prueba: Subida y asociación de un archivo PDF al expediente de un paciente.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".
    ◦ Existe un paciente al que se le adjuntará el documento.
    ◦ Se dispone de un archivo PDF de prueba válido.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la ficha de un paciente existente (ej. tras búsqueda o desde "Pacientes Recientes").
    2. Hacer clic en la opción para "Adjuntar Documentos" (ya sea en el historial clínico o en una sección dedicada de documentos).
    3. En el diálogo de selección de archivos, seleccionar un archivo PDF de prueba.
    4. Confirmar la subida o etiquetar el documento si la interfaz lo requiere.
• Datos de Entrada de Prueba:
    ◦ Archivo: Informe_Radiografia_Juan_Perez.pdf
    ◦ Tamaño: 1.5 MB
• Resultado Esperado:
    ◦ El archivo se sube a "Cloud Storage" y se almacena cifrado en reposo.
    ◦ El documento digital se asocia directamente al expediente del paciente (o a una entrada específica del historial clínico).
    ◦ El "Servicio de Logs de Auditoría" registra la subida del documento.
• Criterios de Éxito/Fallo: La prueba pasa si el PDF se sube, se asocia correctamente al paciente y es visible en su expediente, y la acción se registra en los logs de auditoría; de lo contrario, falla.
ID: CP-DOCUMENTOS-002
• Descripción Breve del Escenario que se Prueba: Intento de subida de un archivo con tipo no soportado.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".

    ◦ Existe un paciente para adjuntar el documento.
    ◦ Se dispone de un archivo de prueba con extensión no soportada (ej. .exe, .zip).
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la ficha de un paciente existente.
    2. Hacer clic en la opción para "Adjuntar Documentos".
    3. En el diálogo de selección de archivos, seleccionar un archivo de tipo no soportado (ej. malware.exe).
    4. Confirmar la subida.
• Datos de Entrada de Prueba:
    ◦ Archivo: documento_malicioso.exe
• Resultado Esperado:
    ◦ La aplicación bloquea la subida en el frontend o el backend la rechaza.
    ◦ Se muestra un mensaje de error: "Tipo de archivo no válido".
    ◦ No se registra ninguna acción de subida exitosa en el "Servicio de Logs de Auditoría" para este intento.
• Criterios de Éxito/Fallo: La prueba pasa si el sistema impide la subida del archivo no soportado y muestra un mensaje de error relevante; de lo contrario, falla.
ID: CP-DOCUMENTOS-003
• Descripción Breve del Escenario que se Prueba: Visualización de un documento digital previamente subido en el expediente del paciente.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal", "Admin Secundario" o "Asistente".
    ◦ Existe un paciente con al menos un documento digital (ej. PDF, imagen, video corto) asociado a su expediente en "Cloud Storage".
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la ficha de un paciente existente.
    2. Acceder a la sección donde se listan los documentos asociados al paciente o al "Visor de Archivos Visuales".
    3. Hacer clic en la miniatura o el nombre de un documento para visualizarlo.
• Datos de Entrada de Prueba: (N/A, se utiliza un documento existente)
• Resultado Esperado:
    ◦ El sistema recupera el archivo desde "Cloud Storage".
    ◦ El documento se abre y visualiza correctamente dentro del visor integrado de la aplicación o en una ventana emergente.
    ◦ La visualización de imágenes y videos se realiza de manera fluida, sin demoras significativas, considerando la conectividad de al menos 250 Mbps.
• Criterios de Éxito/Fallo: La prueba pasa si el documento se visualiza correctamente y de forma fluida; de lo contrario, falla.
4. Pruebas del Módulo de Agenda y Citas
ID: CP-AGENDA-001
• Descripción Breve del Escenario que se Prueba: Agendamiento de una nueva cita para un paciente.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".
    ◦ Existe un paciente al que se le agendará la cita.
    ◦ El horario seleccionado está disponible.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Agenda" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Agendar Cita".
    3. En el "Formulario de Agendar Nueva Cita", seleccionar un paciente existente, una "Fecha", una "Hora" y un "Tipo de Cita".
    4. Seleccionar el "Estado" de la cita como "Programada".
    5. Hacer clic en el botón "Guardar Cita".
• Datos de Entrada de Prueba:
    ◦ Paciente: Juan Pérez García
    ◦ Fecha: 2024-07-25
    ◦ Hora: 10:00 AM
    ◦ Tipo de Cita: Primera Consulta
    ◦ Estado: Programada
• Resultado Esperado:
    ◦ La nueva cita se guarda en la "Base de Datos Relacional (Cloud SQL)".
    ◦ La cita aparece visible en el "Calendario Interactivo" y en el panel "CITAS DEL DÍA" en el Dashboard Principal.

    ◦ El "Servicio de Logs de Auditoría" registra la creación de la cita.
• Criterios de Éxito/Fallo: La prueba pasa si la cita se crea correctamente, es visible en el calendario y se registra en los logs de auditoría; de lo contrario, falla.
ID: CP-AGENDA-002
• Descripción Breve del Escenario que se Prueba: Intento de agendamiento de una cita en un horario ya ocupado.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".
    ◦ Existe una cita ya agendada para una fecha y hora específica.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Agenda" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Agendar Cita".
    3. En el "Formulario de Agendar Nueva Cita", intentar agendar una cita para la misma "Fecha" y "Hora" de una cita ya existente.
    4. Hacer clic en el botón "Guardar Cita".
• Datos de Entrada de Prueba:
    ◦ Paciente: Nuevo Paciente
    ◦ Fecha: 2024-07-25 (misma que cita existente)
    ◦ Hora: 10:00 AM (misma que cita existente)
    ◦ Tipo de Cita: Seguimiento
• Resultado Esperado:
    ◦ La aplicación impide el agendamiento y permanece en el formulario.
    ◦ Se muestra un mensaje de conflicto: "Horario no disponible".
• Criterios de Éxito/Fallo: La prueba pasa si el sistema detecta el conflicto de horario y muestra el mensaje de error correspondiente; de lo contrario, falla.
ID: CP-AGENDA-003
• Descripción Breve del Escenario que se Prueba: Reprogramación de una cita existente a una nueva fecha y hora.
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal" o "Asistente".
    ◦ Existe una cita de prueba ya agendada y visible en el agenda.
    ◦ La nueva fecha y hora seleccionada están disponibles.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Agenda" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Ver Calendario".
    3. Seleccionar una cita existente en el calendario y hacer clic en ella para ver los "Detalles de Cita".
    4. Hacer clic en el botón "Reprogramar" o editar los campos de fecha/hora.
    5. Seleccionar una nueva "Fecha" y "Hora".
    6. Guardar los cambios.
• Datos de Entrada de Prueba:
    ◦ Cita Original: 2024-07-25 10:00 AM - Juan Pérez
    ◦ Nueva Fecha: 2024-08-01
    ◦ Nueva Hora: 11:30 AM
• Resultado Esperado:
    ◦ La cita se actualiza en la "Base de Datos Relacional (Cloud SQL)" con la nueva fecha y hora.
    ◦ La cita se muestra en el calendario en su nueva posición y desaparece de la anterior.
    ◦ El "Servicio de Logs de Auditoría" registra la modificación de la cita.
• Criterios de Éxito/Fallo: La prueba pasa si la cita se reprograma correctamente y la acción se registra en los logs de auditoría; de lo contrario, falla.
5. Pruebas de Seguridad, Roles y Accesos Rápidos
ID: CP-SEGURIDAD-001
• Descripción Breve del Escenario que se Prueba: Creación de una nueva cuenta de usuario con rol "Asistente" por un "Admin Principal".
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal".
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Administración" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Gestionar Usuarios".
    3. Hacer clic en el botón "Crear Nuevo Usuario".
    4. Completar los campos de usuario (Nombre de Usuario, Contraseña) y seleccionar

"Asistente" del desplegable de roles.
    5. Hacer clic en "Guardar Usuario".
• Datos de Entrada de Prueba:
    ◦ Nombre de Usuario: asistente.prueba
    ◦ Contraseña: PasswordAsistente1!
    ◦ Rol: Asistente
• Resultado Esperado:
    ◦ La nueva cuenta de usuario se crea y se guarda en la "Base de Datos Relacional (Cloud SQL)", con la contraseña hasheada y el rol asignado.
    ◦ El usuario aparece en el listado de usuarios en la pantalla "Gestión de Usuarios".
    ◦ El "Servicio de Logs de Auditoría" registra la creación del nuevo usuario.
• Criterios de Éxito/Fallo: La prueba pasa si el nuevo usuario con rol "Asistente" se crea correctamente y la acción se registra en los logs de auditoría; de lo contrario, falla.
ID: CP-SEGURIDAD-002
• Descripción Breve del Escenario que se Prueba: Intento de acceso a "Gestionar Usuarios" por un usuario con rol "Asistente" (permiso denegado).
• Precondiciones:
    ◦ Usuario logueado con rol "Asistente".
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Administración" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Gestionar Usuarios".
• Datos de Entrada de Prueba: (N/A)
• Resultado Esperado:
    ◦ El sistema deniega el acceso a la pantalla "Gestión de Usuarios".
    ◦ Se muestra un mensaje de "Acceso Denegado" o simplemente la opción no es visible/clicable.
    ◦ El "Servicio de Logs de Auditoría" puede registrar un intento de acceso no autorizado, si la implementación lo contempla a este nivel.
• Criterios de Éxito/Fallo: La prueba pasa si el usuario "Asistente" no puede acceder a la gestión de usuarios y se le notifica la denegación de acceso; de lo contrario, falla.
ID: CP-SEGURIDAD-003
• Descripción Breve del Escenario que se Prueba: Acceso y visualización de los logs de auditoría por un "Admin Principal".
• Precondiciones:
    ◦ Usuario logueado con rol "Admin Principal".
    ◦ Existen registros en el "Servicio de Logs de Auditoría".
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a la sección "Administración" en la "Barra de Navegación Lateral".
    2. Hacer clic en la sub-opción "Ver Logs de Auditoría".
• Datos de Entrada de Prueba: (N/A)
• Resultado Esperado:
    ◦ La aplicación muestra la "Pantalla: Logs de Auditoría" con una "Tabla cronológica" de todas las actividades y cambios realizados en el sistema (ej., creación de pacientes, modificaciones de citas, intentos de login).
    ◦ Los logs son inmutables (Write Once, Read Many - WORM), asegurando su integridad.
• Criterios de Éxito/Fallo: La prueba pasa si el "Admin Principal" puede acceder y visualizar la tabla completa de logs de auditoría sin errores; de lo contrario, falla.
6. Pruebas de Inteligencia y Automatización Integrada (Modelo de IA Gemma 3)
ID: CP-IA-001
• Descripción Breve del Escenario que se Prueba: Interacción con el "Chat Interno Básico" para una consulta rápida.
• Precondiciones:
    ◦ Usuario logueado con cualquier rol.
    ◦ El modelo de IA "Gemma 3" está integrado y operativo en el backend a través de APIs.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Hacer clic en el icono o navegar a la sección dedicada al chat interno en el Dashboard.
    2. En el "Campo de entrada de texto", ingresar una consulta simple.
    3. Hacer clic en "Enviar".
• Datos de Entrada de Prueba:
    ◦ Consulta: Resúmeme el historial de Juan Pérez García.
• Resultado Esperado:
    ◦ El backend envía la consulta al modelo "Gemma 3".

    ◦ El chat interno muestra una respuesta relevante generada por la IA en formato de texto.
    ◦ La interacción es solo de texto y no genera archivos.
• Criterios de Éxito/Fallo: La prueba pasa si la IA responde correctamente a la consulta en formato de texto y no intenta generar un archivo; de lo contrario, falla.
ID: CP-IA-002
• Descripción Breve del Escenario que se Prueba: Uso de la IA para generar un borrador de correo electrónico.
• Precondiciones:
    ◦ Usuario logueado con cualquier rol.
    ◦ El modelo de IA "Gemma 3" está integrado y operativo en el backend a través de APIs.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. Navegar a un módulo que permita la automatización de la redacción de correos o usar el chat interno.
    2. Ingresar una instrucción para generar un borrador de correo.
    3. Hacer clic en "Enviar" o activar la función de generación.
• Datos de Entrada de Prueba:
    ◦ Instrucción: Redacta un correo de recordatorio de cita para Juan Pérez para el 25 de julio a las 10 AM.
• Resultado Esperado:
    ◦ La IA genera un borrador de correo electrónico basado en la instrucción y lo muestra en la interfaz (ej., en el chat o en un campo de texto de un formulario de correo).
    ◦ El contenido es coherente con la solicitud y en formato de texto.
• Criterios de Éxito/Fallo: La prueba pasa si la IA genera un borrador de correo electrónico relevante y en el formato de texto esperado; de lo contrario, falla.
7. Pruebas de Cerrar Sesión
ID: CP-LOGOUT-001
• Descripción Breve del Escenario que se Prueba: Cierre de sesión exitoso desde el Dashboard Principal.
• Precondiciones:
    ◦ Usuario logueado con cualquier rol en el Dashboard Principal.
• Pasos Específicos a Seguir para Ejecutar la Prueba:
    1. En la "Cabecera (Header)" del Dashboard Principal, hacer clic en la opción "Cerrar Sesión".
• Datos de Entrada de Prueba: (N/A)
• Resultado Esperado:
    ◦ La sesión del usuario se invalida en el backend.
    ◦ El usuario es redirigido a la "Pantalla: Login de Usuario".
    ◦ El "Servicio de Logs de Auditoría" registra la acción de cierre de sesión.
• Criterios de Éxito/Fallo: La prueba pasa si el usuario es redirigido a la pantalla de Login y su sesión es correctamente terminada; de lo contrario, falla.

5. Pruebas Específicas Importantes (Ejemplos de Enfoque):
• Enfoque en probar a fondo las funcionalidades críticas como:
    ◦ Ciclo completo de gestión de Pacientes (Crear, Ver, Editar, Archivar, Eliminar Suave):
        ▪ Creación de Perfiles: Se debe verificar la correcta creación de perfiles de paciente con todos sus datos demográficos y de contacto. Esto incluye probar la validación de campos obligatorios y formatos incorrectos, como el mensaje "Campo 'Nombre' requerido" o "Formato de email inválido".
        ▪ Visualización: Es crucial probar la capacidad de búsqueda instantánea de pacientes por cualquier campo (nombre, ID, palabra clave en notas) y que el sistema devuelva resultados precisos. También, verificar que el "Dashboard Principal" muestre a los "Pacientes Recientes" en una tabla con la información adecuada y opciones de "Acción" como "Ver Historial" o "Nueva Cita".
        ▪ Edición: Se debe asegurar que los datos de un paciente puedan ser modificados y guardados correctamente, y que estos cambios se reflejen en la base de datos.
        ▪ Registro Historial Clínico Detallado: Probar la adición de nuevas entradas cronológicas en el historial clínico de un paciente, verificando que se incluyan campos como "Fecha de Consulta", "Diagnóstico", "Notas de Evolución" y "Procedimientos y Tratamientos". Cada entrada debe vincularse al paciente específico.
        ▪ Archivar / Eliminar Suave: Un aspecto crítico es la implementación del "borrado suave" (soft delete). Las pruebas deben confirmar que, al seleccionar registros y hacer

clic en "Eliminar", los datos no se borran permanentemente de la base de datos, sino que se marcan como inactivos o archivados (ej., is_deleted = true). Esto garantiza que la información no se pierda accidentalmente y se mantenga para auditoría o recuperación futura. Esta funcionalidad aborda la "saturación y dispersión de información" y la "dificultad en el seguimiento eficiente del paciente".

- Funcionalidades de Búsqueda y Filtrado (probar diferentes criterios, casos borde, búsqueda vacía):
  - Búsqueda Avanzada: Se debe verificar la eficacia de la búsqueda por múltiples criterios (nombre, ID, palabra clave en notas) y a nivel global en la aplicación.
  - Casos Borde y Búsqueda Vacía: Probar búsquedas con caracteres especiales, nombres muy largos/cortos, y especialmente qué ocurre cuando una búsqueda no encuentra resultados, asegurando que se muestre un mensaje claro como "No se encontraron pacientes".
  - Filtros: El sistema ofrece opciones de filtrado avanzado (por fecha, estado, tipo de consulta) y una estructura de carpetas (General, Operaciones, Consultas, Laboratorio) que funciona como filtro. Se debe validar que al aplicar estos filtros, el contenido del área principal se actualice correctamente para mostrar solo los registros pertenecientes a la categoría seleccionada.
- Gestión Documental (Subir diferentes tipos de archivos, ver archivos, eliminar suave):
  - Subida y Asociación: Probar la capacidad de subir, asociar y visualizar todo tipo de archivos digitales (PDF, imágenes, videos cortos) directamente en el expediente del paciente o en una entrada específica del historial clínico. Es crucial verificar que los archivos se almacenen en Cloud Storage y que estén "cifrados en reposo".
  - Visualización: Comprobar el visor de archivos visuales, asegurando que el carrusel de vistas previas funcione correctamente y que los filtros de origen/tipo (Cloud Drives, Photos, Videos) funcionen como se espera.
  - Manejo de Errores de Subida: Probar escenarios donde se intentan subir tipos de archivo no soportados, archivos que exceden el tamaño máximo, o fallos de conexión, y verificar que los mensajes de error sean claros (ej., "Tipo de archivo no válido", "Error al subir el archivo").
  - Eliminación Suave: Al igual que con los pacientes, se debe verificar que los documentos se marquen como inactivos en lugar de eliminarse físicamente, manteniendo la integridad y seguridad de los datos.
- Gestión de Citas (Crear, Editar, Eliminar, probar con fechas pasadas/futuras):
  - Creación y Agendamiento: Probar la creación de nuevas citas, verificando la correcta selección de paciente, fecha, hora, tipo de cita y estado (Programada, Confirmada, etc.). Es vital probar los escenarios de conflicto de horario para asegurar que se muestre el mensaje "Horario no disponible" si la hora ya está ocupada.
  - Gestión de Citas Existentes: Probar las acciones de reprogramación, cancelación y cambio de estado de las citas (Confirmada, Atendida, Cancelada), asegurando que los cambios se reflejen correctamente en el calendario interactivo.
  - Fechas Pasadas/Futuras: Asegurar que el sistema maneje correctamente la visualización y gestión de citas en el pasado y futuro, incluyendo la visualización de "Citas del Día" en el dashboard. Estas funcionalidades son clave para abordar los "procesos manuales que consumen tiempo" y la "falta de control ágil de visitas diarias".
- Todas las interacciones definidas con el Asistente Virtual (probar diferentes comandos/preguntas, verificar respuestas):
  - Chat Interno: Probar las capacidades de interacción con el chat interno básico, incluyendo la realización de consultas rápidas, obtención de resúmenes de texto y redacción de informes, confirmando que solo funciona con texto y no genera archivos.
  - Automatizaciones Básicas: Validar la redacción automatizada de correos electrónicos, la generación de hojas membretadas y la creación de recordatorios a través de la IA.
  - Búsqueda Inteligente: Probar comandos en lenguaje natural para la búsqueda de archivos o información específica (ej., "Hola, estoy buscando un archivo me urge", "Recuerdas el nombre del paciente?", "Si!! es Laura Vargas").
  - Recuperación y Acciones Contextuales: Verificar que el asistente localice los recursos y ofrezca acciones contextuales como enviar por correo o descargar.
  - Automatización de Tareas: Confirmar que el asistente puede ejecutar acciones conectadas a las APIs del sistema, como enviar correos o generar borradores de documentos.
  - Generación de Contenido para Capacitación: La IA (Gemma 3) puede ayudar a generar preguntas frecuentes (FAQs) para el manual de usuario, lo cual también debe ser probado.

    ◦ Pruebas de Seguridad: Intentar acceder a funcionalidades de Administrador con un usuario Asistente, intentar eliminar un registro con borrado fuerte si solo se permite borrado suave, probar inyecciones simples en campos de entrada.
        ▪ Control de Acceso Basado en Roles (RBAC): Realizar pruebas exhaustivas para asegurar que cada rol (Administrador, Admin Principal, Admin Secundario, Asistente, Invitado (Solo Lectura)) solo pueda acceder a las funcionalidades y la información correspondiente a sus permisos. Por ejemplo, un "Asistente" no debería poder gestionar usuarios o ver logs de auditoría. Este es un pilar fundamental para las "preocupaciones de seguridad de datos".
        ▪ Borrado Suave vs. Fuerte: Verificar que, incluso si se intenta una eliminación directa o "fuerte", el sistema aplique consistentemente el "borrado suave".
        ▪ Cifrado de Datos: Aunque las pruebas directas de cifrado pueden requerir acceso a la infraestructura, se debe confirmar que las contraseñas se almacenen "cifradas en reposo" (hashing) y que los "datos sensibles" en la base de datos (Cloud SQL) y el almacenamiento de archivos (Cloud Storage) estén "cifrados en reposo".
        ▪ Cifrado en Tránsito: Asegurar que toda la comunicación entre la aplicación de escritorio y la infraestructura en la nube utilice "HTTPS (TLS 1.3)" para cifrar los datos en tránsito.
        ▪ Logs de Auditoría: Verificar que cada intento de inicio de sesión (exitoso o fallido) y cada acción del usuario (ej., modificación de datos de paciente, creación de citas) se registre "cronológicamente" en el "Servicio de Logs de Auditoría" y que estos logs sean "inmutables (WORM)".
        ▪ Pruebas de Inyección: Aunque no se detalla cómo se realizará, es una buena práctica probar con "inyecciones simples en campos de entrada" para detectar vulnerabilidades comunes.
    ◦ Pruebas de Roles y Permisos: Crear matriz de permisos y probar que cada rol solo puede ejecutar las acciones permitidas.
        ▪ Matriz de Permisos: Se debe desarrollar una matriz detallada que especifique qué acciones y secciones son accesibles para cada rol: "Admin Principal" (control total, gestión de usuarios, ver logs), "Admin Secundario" (todo excepto gestión de usuarios y logs), "Asistente" (gestión de pacientes, citas, documentos, reportes), y "Invitado (Solo Lectura)" (ver pacientes, citas, reportes).
        ▪ Validación de Acciones: Ejecutar pruebas con usuarios de cada rol para confirmar que solo pueden realizar las acciones definidas por sus permisos y que se les niegue el acceso a funcionalidades restringidas, mostrando mensajes de error apropiados si intentan acceder a ellas.
    ◦ Pruebas de Borrado Suave: Verificar en la base de datos que los registros se marcan como inactivos y no se eliminan físicamente.
        ▪ Inspección de Base de Datos: Después de realizar operaciones de "eliminación" desde la interfaz de usuario, se debe acceder directamente a la base de datos (Cloud SQL) para confirmar que los registros correspondientes no han sido eliminados físicamente, sino que su estado ha cambiado a "inactivo" o que se ha establecido un indicador como is_deleted = true. Esto es crucial para la integridad de los datos y el cumplimiento de las políticas de seguridad.

Criterios de Entrada y Salida de las Pruebas para la Fase 1: El Núcleo Administrativo (Dashboard de Escritorio)

La Fase 1 se enfoca en la construcción del "Núcleo Administrativo (Dashboard de Escritorio)", que es la columna vertebral y base de datos central del ecosistema. Su objetivo es asegurar la integridad y seguridad de la información desde el primer día. Las pruebas de esta fase son cruciales para validar las funcionalidades clave antes de la entrega.

Criterios de Entrada (Qué debe estar listo antes de comenzar las pruebas)

Para iniciar las pruebas de la Fase 1 del Dashboard de Escritorio, las siguientes condiciones deben cumplirse:

1. Código Implementado para Todas las Funcionalidades de la Fase 1:
    ◦ La interfaz de usuario (frontend) de la aplicación de escritorio debe estar completamente desarrollada utilizando React, Vue.js o un framework similar, y empaquetada con Electron.js para compatibilidad con Windows/macOS.
    ◦ Toda la lógica de negocio y las APIs (backend) para las funcionalidades de la Fase 1 deben estar codificadas, utilizando Python (Flask/Django) o Node.js (Express).
    ◦ Esto incluye el desarrollo completo de la Gestión Integral del Expediente del Paciente (creación, búsqueda, historial clínico detallado), la Gestión Documental Avanzada (subida, asociación y visualización de archivos), el Módulo de Agenda y Citas

(programación y gestión de citas), la Seguridad, Roles y Accesos Rápidos (login, roles), y la Inteligencia y Automatización Integrada (chat interno, automatizaciones básicas de IA).
2. Entorno de Pruebas Configurado y Desplegado:
    ◦ La infraestructura en la nube en Google Cloud Platform (GCP) debe estar completamente configurada y desplegada. Esto incluye:
        ▪ Virtual Private Cloud (VPC) lógicamente aislada y segura.
        ▪ API Gateway / Load Balancer como puntos de entrada seguros y distribuidores de tráfico para el backend.
        ▪ Servidor de Aplicación (Backend) desplegado en servicios como Compute Engine o Cloud Run.
        ▪ Base de Datos Relacional (Cloud SQL - MySQL/PostgreSQL) para almacenar información estructurada.
        ▪ Almacenamiento de Archivos (Cloud Storage) para documentos digitales.
        ▪ Servicio de Logs de Auditoría (Cloud Logging / Audit Logs) configurado para registrar cronológicamente actividades.
    ◦ La conectividad entre la aplicación de escritorio y la infraestructura en la nube debe estar establecida a través de HTTPS (TLS 1.3).
    ◦ Se recomienda una conexión a internet estable de al menos 250 Mbps para un rendimiento óptimo del ecosistema.
3. Datos de Prueba Disponibles y Seguros:
    ◦ El esquema de la base de datos para USUARIOS, PACIENTES, HISTORIAL_CLINICO, CITAS, DOCUMENTOS, ROLES, y LOGS_AUDITORIA debe estar completamente definido e implementado.
    ◦ Los datos sensibles, como contraseñas, deben estar cifrados en reposo en la base de datos (mediante hashing) y en Cloud Storage.
    ◦ Se deben haber cargado datos de prueba representativos para cada módulo funcional de la Fase 1 (ej., perfiles de pacientes, historiales clínicos, documentos, citas programadas, usuarios con diferentes roles).
4. Integración con IA Funcional:
    ◦ El modelo de Inteligencia Artificial Gemma 3 (o similar a Gemini) debe estar integrado directamente en el backend mediante APIs, listo para potenciar las funcionalidades de automatización y el chat interno.
5. Especificaciones de Diseño e Interfaz Listas:
    ◦ Todos los elementos visuales del diseño "Glassmorphism Oscuro" (fondo gris carbón suave, auras de color azul oceánico y violeta, paneles flotantes de vidrio esmerilado, texto nítido y brillante con la fuente Inter) deben estar implementados según las especificaciones técnicas para la interfaz de usuario del Dashboard y el Login. Esto asegura una coherencia visual integral y comodidad para el usuario.
Criterios de Salida (Qué condiciones deben cumplirse para considerar que las pruebas de la Fase 1 han finalizado con éxito)
Para considerar que las pruebas de la Fase 1 han finalizado con éxito, la aplicación de escritorio debe cumplir con las siguientes condiciones:
1. Funcionalidad Completa y Correcta:
    ◦ Todas las funcionalidades de la Fase 1 deben operar como se espera y sin errores significativos, abarcando:
        ▪ Gestión Integral del Expediente del Paciente: Creación, búsqueda avanzada, y registro detallado del historial clínico (consultas, diagnósticos, notas, procedimientos, tratamientos).
        ▪ Gestión Documental Avanzada: Capacidad de subir, asociar y visualizar archivos digitales (PDF, imágenes, videos cortos) al expediente del paciente.
        ▪ Módulo de Agenda y Citas: Programación, reprogramación y gestión de citas con estados personalizables, y visualización efectiva en el calendario.
        ▪ Seguridad, Roles y Accesos Rápidos: El sistema de login debe autenticar correctamente a los usuarios, aplicar el control de acceso basado en roles (Admin Principal, Admin Secundario, Asistente, Invitado), y gestionar usuarios y roles de forma segura.
        ▪ Inteligencia y Automatización Integrada: El chat interno con Gemma 3 debe responder a consultas rápidas, generar resúmenes de texto y permitir la redacción de correos, hojas membretadas y recordatorios.
    ◦ Los widgets y accesos directos a servicios externos como Gmail y Google Drive deben funcionar correctamente.
2. Rendimiento y Estabilidad:
    ◦ El sistema debe demostrar un rendimiento óptimo en las operaciones críticas, como la búsqueda de pacientes o la carga de historiales clínicos, con tiempos de respuesta

aceptables.
    ◦ La aplicación debe ser estable y no presentar cierres inesperados, bloqueos o degradación de rendimiento durante el uso continuo.
3. Seguridad y Privacidad de Datos:
    ◦ La autenticación y autorización basada en roles debe funcionar sin fallos, impidiendo el acceso indebido a información o funcionalidades.
    ◦ El cifrado de datos en tránsito (HTTPS/TLS 1.3) y en reposo (Cloud SQL, Cloud Storage) debe ser verificado como completamente funcional, protegiendo la información sensible del paciente.
    ◦ El Servicio de Logs de Auditoría debe registrar de forma inmutable (WORM) todos los intentos de login (exitosos y fallidos), así como las acciones relevantes de los usuarios (ej., modificación de datos de paciente, creación de citas), y estos logs deben ser accesibles para revisión.
4. Manejo Robusto de Errores y Mensajes:
    ◦ Todos los escenarios de error previstos (ej., credenciales incorrectas, campos vacíos/formato inválido, problemas de conectividad) deben mostrar mensajes claros y concisos al usuario, permitiéndole entender el problema y tomar la acción adecuada.
    ◦ Los "estados vacíos" (ej., "No se encontraron pacientes", "No hay citas programadas para hoy") deben ser manejados con mensajes informativos en lugar de espacios en blanco.
5. Coherencia de la Interfaz de Usuario (UI/UX) y Accesibilidad:
    ◦ El diseño "Glassmorphism Oscuro" debe aplicarse de manera consistente en toda la aplicación de escritorio, manteniendo la coherencia visual, la comodidad visual, y una jerarquía de información clara.
    ◦ La legibilidad del texto y los iconos debe ser óptima (cumpliendo con una relación de contraste superior a 4.5:1 WCAG).
    ◦ La retroalimentación interactiva (ej., aumento de brillo o hundimiento al pulsar botones) debe ser clara y consistente.
6. Preparación para Capacitación y Migración de Datos:
    ◦ La aplicación debe ser suficientemente estable para la sesión de capacitación inicial del Dr. Sánchez y su equipo.
    ◦ Los manuales de funcionamiento deben estar preparados, posiblemente con la ayuda de la IA para generar FAQs.
    ◦ La migración inicial de archivos digitales existentes del consultorio al nuevo sistema debe haberse completado con éxito.
7. Resolución de Defectos:
    ◦ Todos los defectos de prioridad "Crítica" y "Mayor" identificados durante las pruebas deben haber sido resueltos y verificados.

Recursos para las Pruebas: Ecosistema Digital Inteligente para Cirugía Especial (Fase 1: El Núcleo Administrativo)

La Fase 1 se enfoca en la construcción del "Núcleo Administrativo (Dashboard de Escritorio)", que es la columna vertebral y base de datos central del ecosistema. Las pruebas de esta fase son cruciales para validar las funcionalidades clave antes de la entrega.

1. Personal de Pruebas

El "equipo de desarrollo" o "Su Nombre/Nombre de su Empresa" es el encargado de diseñar, construir e implementar el ecosistema. Este mismo equipo es responsable de "Realizar pruebas exhaustivas de todas las funcionalidades, asegurando la seguridad y el rendimiento". No se especifica la existencia de un equipo de pruebas independiente; por lo tanto, las actividades de prueba recaen en el equipo de desarrollo. El Dr. Joel Sánchez García y su equipo son los usuarios clave y beneficiarios finales del ecosistema.

2. Entorno de Pruebas (Configuración)

El entorno de pruebas será una réplica de la infraestructura de producción, basada en Google Cloud Platform (GCP). Los componentes clave de este entorno en la nube incluyen:

• Virtual Private Cloud (VPC): Una sección lógicamente aislada dentro de la nube pública de GCP donde se despliegan todos los recursos del sistema, creando un entorno de red privado y seguro.

• API Gateway / Load Balancer: Actúan como los puntos de entrada seguros para todas las solicitudes provenientes de las aplicaciones cliente, distribuyendo el tráfico de manera eficiente y garantizando la disponibilidad y seguridad del sistema.

• Servidor de Aplicación (Backend): Donde reside la lógica de negocio central y las APIs, construido con Python (Flask/Django) o Node.js (Express), y desplegado utilizando servicios de GCP como Compute Engine o Cloud Run. La integración del modelo de IA Gemma 3 también reside aquí.

• Base de Datos Relacional (Cloud SQL): Almacena toda la información estructurada y crítica, como perfiles de pacientes, historiales clínicos, citas, usuarios y roles, utilizando MySQL o PostgreSQL. Los datos sensibles estarán cifrados en reposo.
• Almacenamiento de Archivos (Cloud Storage): Utilizado para el almacenamiento centralizado y seguro de archivos digitales asociados a los expedientes de los pacientes (PDFs, imágenes, videos cortos). Estos archivos también estarán cifrados en reposo.
• Servicio de Logs de Auditoría (Cloud Logging / Audit Logs): Vital para el monitoreo de la seguridad y el seguimiento de operaciones, registrando cronológicamente todas las actividades y cambios realizados; estos logs son inmutables (Write Once, Read Many - WORM).
• Conectividad: La comunicación entre la aplicación de escritorio y la infraestructura en la nube se realizará a través de HTTPS (TLS 1.3), cifrando los datos en tránsito. Se recomienda una conexión a internet estable de al menos 250 Mbps para un rendimiento óptimo del ecosistema.
• Aplicación de Escritorio: El frontend de la aplicación de escritorio se desarrollará con React, Vue.js o similar, y se empaquetará con Electron.js para compatibilidad con Windows/macOS.

3. Herramientas de Prueba

La documentación no especifica herramientas de prueba automatizadas (ej., para pruebas unitarias, de integración o funcionales). Sin embargo, las herramientas de desarrollo y configuración mencionadas, que facilitan un desarrollo "controlado y de alta calidad", pueden contribuir indirectamente a la capacidad de prueba:
• Herramientas de Diseño (UI/UX): Figma y Adobe Suite (XD, Photoshop, Illustrator) se utilizan para definir flujos de usuario, wireframes y prototipos. Aunque no son herramientas de prueba directa, una UI/UX bien definida y prototipada facilita las pruebas de usabilidad y funcionalidad desde etapas tempranas.
• Herramientas de Modelado de Datos: MySQL o PostgreSQL como sistemas de gestión de bases de datos, y herramientas de modelado ERD como Lucidchart, dbdiagram.io o Google Drawings. Un modelo de datos bien estructurado y documentado es esencial para el diseño de pruebas de integración y base de datos.
• Herramientas de Desarrollo Backend: Python (Flask/Django) o Node.js (Express) para el backend.
• Herramientas de Infraestructura como Código (IaC): Terraform o Cloud Deployment Manager se mencionan para automatizar la creación y configuración de recursos en GCP. Esto permite la provisión de entornos de prueba consistentes y replicables.
• Herramientas de Desarrollo Frontend: React, Vue.js o similar, y Electron.js para la aplicación de escritorio.
• Herramientas de Colaboración: Google Workspace (Docs/Sheets) para la creación de manuales de usuario colaborativos. La IA (Gemma 3) puede ayudar a generar preguntas frecuentes (FAQs) para el manual de usuario, lo que podría asistir en la creación de casos de prueba o guías de capacitación.

4. Datos de Prueba

La disponibilidad de datos de prueba es un criterio de entrada clave para iniciar las pruebas de la Fase 1.
• Datos de Prueba Representativos: Se deben haber cargado datos de prueba representativos para cada módulo funcional de la Fase 1. Esto incluye, por ejemplo:
    ◦ Perfiles de pacientes.
    ◦ Historiales clínicos.
    ◦ Documentos.
    ◦ Citas programadas.
    ◦ Usuarios con diferentes roles.
• Esquema de la Base de Datos: El esquema de la base de datos para USUARIOS, PACIENTES, HISTORIAL_CLINICO, CITAS, DOCUMENTOS, ROLES, y LOGS_AUDITORIA debe estar completamente definido e implementado.
• Datos Sensibles: Los datos sensibles, como contraseñas, deben estar cifrados en reposo en la base de datos (mediante hashing) y en Cloud Storage. El campo hash_password en la tabla USUARIOS indica el uso de funciones de hash para almacenar contraseñas de forma segura.
• Migración de Datos: La "migración inicial de archivos digitales existentes del consultorio al nuevo sistema" se realiza en la Fase 1. Esto sugiere que datos reales o muy realistas serán utilizados, lo cual es crucial para pruebas de volumen y rendimiento.

Proceso de Gestión de Defectos (Bugs)

La detección y resolución de bugs son cruciales para asegurar la "integridad y seguridad

de la información" y la "eficiencia operativa". El ecosistema está diseñado para ser "potente, seguro y excepcionalmente intuitivo", y el manejo de defectos contribuye a mantener estos atributos.

1. Detección y Reporte de Bugs

• Durante el Desarrollo y Pruebas Internas (Fase 1 y subsiguientes): El "equipo de desarrollo" o "Su Nombre/Nombre de su Empresa" es directamente responsable de "Realizar pruebas exhaustivas de todas las funcionalidades, asegurando la seguridad y el rendimiento" [Recursos para las Pruebas: 1, Plan de Acción Detallado: Paso 5]. Durante estas pruebas, los desarrolladores y el equipo de QA (Control de Calidad, implícito en "pruebas exhaustivas") identificarán los bugs. La información de los fallos y comportamientos inesperados se documentará internamente para su posterior análisis.

• Post-Implementación (Soporte al Cliente): Una vez que las fases del proyecto son entregadas, el Dr. Joel Sánchez García y su equipo ("usuarios clave" [Recursos para las Pruebas: 1]) serán los principales reportadores de bugs o incidentes. El "Plan de Mantenimiento y Soporte del Ecosistema" incluye "soporte técnico 24/7 (remoto y presencial según diagnóstico)" [Plan de Trabajo: 6, Preguntas Frecuentes: 7, Línea de Tiempo: Después de la Finalización del Proyecto y la Fase 1]. Esto implica que cualquier error o comportamiento anómalo que detecten en la operación diaria será reportado directamente al equipo de "Su Nombre/Nombre de su Empresa" a través de los canales de soporte que se establezcan (aunque no se especifican los canales exactos, se infiere una comunicación directa y constante).

2. Priorización de Bugs

Las fuentes no especifican una matriz de priorización formal (ej., crítico, mayor, medio, menor), sin embargo, los objetivos y soluciones del ecosistema guían la prioridad:

• Alta Prioridad: Los defectos que comprometan la "seguridad de datos" [Acceso Seguro: 1, 6, 14-16, 18, 23-29; Arquitectura Nube: 9-14; Dashboard: I, II, 4, 6, 9; Ecosistema Digital: Arquitectura y Diseño Glassmorphism Oscuro: Introducción, Pilar 1, Consideraciones Clave; Guía de Estudio: I, II, III, IV, VI, VII, VIII; Plan de Acción Detallado: FASE 1, IMPACTO; Pantalla #1 Login: 1, 3, 7; Pantalla #2 Home: 1, 5, 8; Pantalla #Login: 1, 3, 7], impidan el acceso al sistema (como un fallo en el login [Acceso Seguro: 9, Pantalla #1 Login: 3, Pantalla #2 Home: 6, Pantalla #Login: 3]), o detengan funcionalidades críticas para la "eficiencia operativa" [Automatización e Inteligencia: 1; Dashboard: I, II; Ecosistema Digital: Arquitectura y Diseño Glassmorphism Oscuro: Introducción, Pilar 1; Arquitectura Nube: 2, 3; Guía de Estudio: I, IV, VI; Plan de Acción Detallado: IMPACTO; Pantalla #2 Home: 1, 2, 8] (ej. gestión de expedientes, agendamiento de citas) serían tratados con la máxima urgencia. La "actualización de seguridad" es parte del mantenimiento continuo [Plan de Trabajo: 6, Preguntas Frecuentes: 7, Línea de Tiempo: Después de la Finalización del Proyecto y la Fase 1].

• Prioridad Media a Baja: Bugs que afecten la usabilidad sin impedir la funcionalidad, o que generen errores menores, serían priorizados según su impacto en el flujo de trabajo y la cantidad de usuarios afectados. Los mensajes de error definidos para diversos escenarios (ej., "Usuario o contraseña incorrectos", "Campo 'Nombre' requerido", "Problemas de Conectividad") [Acceso Seguro: 9, Mapa Detallado: 1. Flujo de Acceso, Pantalla #1 Login: 3, Pantalla #2 Home: 6, Pantalla #Login: 3] sugieren que la claridad de la retroalimentación al usuario es importante, por lo que los defectos en el manejo de errores también serían abordados.

3. Asignación de Bugs

• La responsabilidad de la investigación y resolución de los bugs recae directamente en "Su Nombre/Nombre de su Empresa" [Línea de Tiempo: Elenco de Personajes, Su Nombre/Nombre de su Empresa], el equipo de desarrollo encargado de la "diseño, construcción e implementación" del ecosistema [Recursos para las Pruebas: 1, Plan de Acción Detallado: Situación] y del "mantenimiento y soporte continuo" [Línea de Tiempo: Después de la Finalización del Proyecto y la Fase 1].

4. Seguimiento y Resolución de Bugs

• Registro y Seguimiento (Implícito): Aunque no se especifica una herramienta de seguimiento de bugs dedicada (como Jira o Trello), el uso de "Google Workspace (Docs/Sheets)" para "crear manuales de usuario colaborativos" [Recursos para las Pruebas: 3, Plan de Acción Detallado: Paso 5] sugiere que se podrían emplear herramientas colaborativas simples para registrar y seguir los bugs internamente. Cada bug reportado sería documentado con detalles como la descripción, pasos para reproducir, fecha de reporte, estado (abierto, en progreso, resuelto, cerrado), y la persona asignada.

• Análisis con Logs de Auditoría: Una herramienta técnica fundamental para el seguimiento y diagnóstico de bugs es el "Servicio de Logs de Auditoría (Cloud Logging / Audit Logs)"

[Acceso Seguro: 7, Arquitectura Nube: 6, Dashboard: 44, Pantalla #1 Login: 5, Pantalla #2 Home: 5, Pantalla #Login: 5, Ecosistema Digital: Arquitectura y Diseño Glassmorphism Oscuro: 6, Plan de Acción Detallado: Paso 3]. Estos logs registran "cronológicamente todas las actividades y cambios realizados" [Acceso Seguro: 7, Arquitectura Nube: 6, Plan de Acción Detallado: Paso 3] y son "inmutables (Write Once, Read Many - WORM)" [Acceso Seguro: 7, Arquitectura Nube: 6, Pantalla #1 Login: 5, Pantalla #2 Home: 5, Pantalla #Login: 5]. Esta capacidad permite al equipo de desarrollo rastrear la secuencia de eventos que llevó al bug, entender su contexto y verificar la correcta aplicación de las soluciones.

• Proceso de Resolución: Una vez diagnosticado, el equipo de desarrollo trabajará en la solución, la implementará, y realizará pruebas internas para asegurar que el bug ha sido corregido y que no se introducen nuevas regresiones. Las "actualizaciones de seguridad y evolución continua" [Plan de Trabajo: 6, Preguntas Frecuentes: 7, Línea de Tiempo: Después de la Finalización del Proyecto y la Fase 1] garantizan que estas correcciones sean parte del ciclo de vida del software.

• Comunicación Continua: El "soporte técnico 24/7" [Plan de Trabajo: 6, Preguntas Frecuentes: 7, Línea de Tiempo: Después de la Finalización del Proyecto y la Fase 1] implica que se mantendrá una comunicación constante con el Dr. Sánchez y su equipo sobre el progreso y la resolución de los problemas reportados, cerrando el ciclo de retroalimentación.

Plan de Ejecución de Pruebas

1. Pruebas Unitarias y de Integración (Durante el Desarrollo)

• Cuándo: Estas pruebas se ejecutarán de forma continua por el equipo de desarrollo de "Su Nombre/Nombre de su Empresa" durante la construcción de cada componente del sistema, tanto en el backend como en el frontend. Esto es fundamental en la Fase 1: El Núcleo Administrativo (Dashboard de Escritorio), que es la columna vertebral del ecosistema.

• Cómo:

    ◦ Pruebas Unitarias: Los desarrolladores verificarán individualmente el correcto funcionamiento de cada unidad de código (funciones, métodos, clases). Esto aplica a la lógica de negocio del Servidor de Aplicación (Backend), construido con Python/Flask/Django o Node.js/Express, y a los componentes de la interfaz de usuario del Dashboard de Escritorio, desarrollado con React, Vue.js o similar.

    ◦ Pruebas de Integración: Se probará la interacción entre diferentes módulos del sistema y con los servicios de la nube. Esto incluye la comunicación segura mediante HTTPS (TLS 1.3) entre la aplicación de escritorio y el backend en Google Cloud Platform (GCP), la correcta interacción del backend con la Base de Datos Relacional (Cloud SQL) y el Almacenamiento de Archivos (Cloud Storage), y la integración del modelo de IA Gemma 3.

    ◦ El objetivo es asegurar un "desarrollo controlado y de alta calidad" desde las primeras etapas.

2. Pruebas Funcionales y de UI (Al Finalizar Funcionalidades)

• Cuándo: Estas pruebas se realizarán una vez que las funcionalidades clave del sistema estén desarrolladas y listas para una verificación más amplia. Se llevarán a cabo principalmente al concluir cada fase del proyecto, con un enfoque intensivo en el Paso 5: Pruebas y Capacitación de la Fase 1, antes de la entrega.

• Cómo:

    ◦ Pruebas Funcionales: Se validará que cada funcionalidad del Dashboard de Escritorio cumple con los requisitos definidos. Por ejemplo, se probará la creación de perfiles de paciente, el registro detallado del historial clínico, la subida y asociación de documentos digitales, la programación y gestión de citas, el correcto funcionamiento del sistema de roles y accesos, y las automatizaciones y el chat interno con IA. Se verificarán los flujos de usuario detallados, como añadir un nuevo paciente o buscar el historial.

    ◦ Pruebas de UI (Interfaz de Usuario): Se asegurará que el diseño "Glassmorphism Oscuro" se implementa correctamente, garantizando la "comodidad visual y una jerarquía de información clara". Esto incluye la legibilidad de textos y iconos, la aplicación de colores y tipografías, y la retroalimentación interactiva al pulsar botones (sutil aumento de brillo o ligero hundimiento).

    ◦ Se comprobará el manejo de errores para escenarios como credenciales incorrectas, campos vacíos o problemas de conectividad, asegurando que los mensajes de error sean claros y concisos.

    ◦ Las pruebas se realizarán tanto por el equipo de desarrollo como, de forma colaborativa, durante la Capacitación Inicial con el Dr. Sánchez y su equipo, quienes utilizarán el sistema en un entorno controlado. Se utilizarán herramientas como Google

Workspace (Docs/Sheets) para documentar los hallazgos.
3. Pruebas de Seguridad (Antes del Despliegue)
• Cuándo: Las pruebas de seguridad son de máxima prioridad y se ejecutarán rigurosamente antes del despliegue de cada fase, especialmente de la Fase 1, dado que el Núcleo Administrativo Central maneja "información sensible".
• Cómo:
    ◦ Control de Acceso Basado en Roles (RBAC): Se validará que el sistema de roles ("Admin Principal", "Admin Secundario", "Asistente", "Invitado (Solo Lectura)") funcione correctamente, impidiendo el acceso no autorizado a funcionalidades o datos sensibles.
    ◦ Cifrado de Datos: Se verificará que el cifrado de datos en tránsito (mediante HTTPS/TLS 1.3) y el cifrado en reposo de los datos sensibles en la Base de Datos (Cloud SQL) y el Almacenamiento de Archivos (Cloud Storage) se apliquen de forma efectiva.
    ◦ Logs de Auditoría: Se confirmará que el Servicio de Logs de Auditoría (Cloud Logging / Audit Logs) registra cronológicamente todos los intentos de login (exitosos y fallidos) y cada acción del usuario (ej., modificación de datos de paciente), y que estos logs son inmutables (Write Once, Read Many - WORM), crucial para la detección de anomalías y la respuesta a incidentes.
    ◦ Infraestructura de Red: Se asegurará que la Virtual Private Cloud (VPC) y el API Gateway / Load Balancer funcionen como la primera línea de defensa, aislando y protegiendo la infraestructura en la nube.
4. Pruebas de Rendimiento (Antes del Despliegue)
• Cuándo: Se realizarán pruebas de rendimiento antes del despliegue de cada fase para garantizar la "eficiencia operativa" y la escalabilidad del sistema.
• Cómo:
    ◦ Escalabilidad y Carga: Se someterá el sistema a pruebas de carga para asegurar que la infraestructura en la nube de GCP puede escalar automáticamente y manejar un mayor volumen de usuarios y datos sin degradación del rendimiento.
    ◦ Tiempos de Respuesta: Se medirán los tiempos de respuesta para las operaciones críticas (ej., búsqueda de pacientes, carga de historiales, agendamiento de citas) para asegurar una experiencia de usuario fluida.
    ◦ Conectividad: Se validará que el ecosistema funciona de manera óptima con una conexión a internet estable de al menos 250 Mbps, que es la velocidad recomendada para un rendimiento óptimo.