Documentación Técnica: API y Base de Datos

1. Arquitectura de la Base de Datos

La estructura de la base de datos es el pilar fundamental sobre el que se construye toda la aplicación. Más que un simple almacén de datos, el esquema define las reglas de negocio, garantiza la integridad de la información y establece las bases lógicas para cada funcionalidad expuesta por la API. Un diseño de base de datos bien planificado es crucial para la escalabilidad, el rendimiento y la seguridad del sistema.

1.1 Entidades Principales del Sistema

A continuación, se describen las tablas que representan los conceptos centrales del modelo de negocio de la aplicación.

- **Entidad `usuarios`:** Esta tabla constituye el pilar de la identidad del usuario en el sistema. Almacena toda la información necesaria para gestionar el perfil y el acceso de una persona. Incluye datos de perfil como el nombre y la ruta a una foto, credenciales de acceso como el correo electrónico (que debe ser único) y una contraseña almacenada de forma segura mediante un hash. Adicionalmente, contiene metadatos de auditoría como la fecha de creación del registro.
- **Entidad `grupos`:** Esta entidad representa los espacios de trabajo colaborativos donde los usuarios interactúan. Su función es agrupar a usuarios, calendarios y tareas bajo un mismo contexto. La tabla almacena la información que define a un grupo, como su nombre y una descripción detallada, así como la fecha de su creación.
- **Entidad `calendarios`:** Funciona como el contenedor principal para los eventos y tareas dentro de un grupo de trabajo. Cada calendario está estrictamente asociado a un único grupo, permitiendo una organización temática o por proyectos (por ejemplo, "Calendario de Marketing", "Calendario General"). Además de su nombre, almacena atributos visuales como un color en formato hexadecimal, que facilita su identificación en la interfaz de usuario.
- **Entidad `eventos`:** Esta entidad es responsable de almacenar la información relativa a citas, reuniones o cualquier actividad con una duración definida en el tiempo. Sus campos clave incluyen un título, una descripción, las fechas y horas exactas de inicio y fin, y una ubicación opcional. Destaca su capacidad para gestionar la recurrencia, permitiendo que un evento se repita de forma diaria, semanal o mensual, entre otras opciones.
- **Entidad `tareas`:** Esta tabla gestiona los elementos de una lista de "cosas por hacer" o "to-do list". Cada tarea se define por su descripción, su estado actual (que puede ser 'pendiente', 'en_proceso' o 'completada') y una fecha límite opcional para

su finalización. Las tareas, al igual que los eventos, están contenidas dentro de un calendario específico.

- **Entidad `invitaciones`:** Esta entidad implementa el mecanismo que permite a nuevos usuarios unirse a los grupos de forma autónoma, sin necesidad de ser añadidos manualmente por un administrador. Gestiona la información de cada token de invitación, incluyendo un token alfanumérico único, el grupo al que da acceso, y las reglas de validez, como una fecha de expiración o un límite máximo de usos, que por defecto a nivel de base de datos se establece en 1.

1.2 Entidades de Relación (Tablas Pivote)

Las tablas intermedias o pivote son esenciales para modelar relaciones complejas de "muchos a muchos" (N:M). Permiten conectar dos entidades principales de una manera flexible que no sería posible utilizando solo claves foráneas directas, resolviendo así necesidades críticas del negocio.

- **Relación `usuarios_grupos`:** Esta tabla es el nexo entre los usuarios y los grupos, especificando qué usuarios pertenecen a qué grupos. Su importancia es crítica debido al campo `rol_en_grupo`, que puede tener valores como `administrador`, `editor` o `miembro`. Este campo es el mecanismo central de control de permisos en toda la aplicación, determinando qué acciones puede realizar un usuario dentro de un grupo específico.
- **Relación `tareas_asignadas`:** Esta tabla vincula las tareas con los usuarios responsables de llevarlas a cabo. Al ser una tabla pivote, permite que una misma tarea sea asignada a múltiples personas y, a su vez, que un usuario tenga múltiples tareas asignadas. Es fundamental para la gestión de responsabilidades y el seguimiento del trabajo en equipo.
- **Relación `eventos_excepciones`:** Esta tabla representa una solución de negocio elegante para la gestión de eventos recurrentes. En lugar de eliminar un evento principal que se repite (por ejemplo, "Reunión semanal todos los lunes"), esta tabla permite registrar fechas específicas en las que una ocurrencia de ese evento debe ser ignorada o cancelada. De esta forma, se mantiene la integridad de la serie recurrente mientras se gestionan casos particulares.

1.3 Relaciones y Lógica de Integridad

Las entidades de la base de datos están interconectadas para reflejar la lógica del negocio y garantizar la consistencia de los datos.

- **Usuarios y Grupos:** Se establece una relación de muchos a muchos (N:M). Un usuario puede pertenecer a múltiples grupos, y un grupo puede contener a múltiples usuarios. La tabla intermedia `usuarios_grupos` define esta relación y especifica el rol de cada usuario dentro de cada grupo.
- **Grupos y Calendarios:** Existe una relación de uno a muchos (1:N). Cada grupo puede poseer uno o más calendarios (siempre se crea uno "General" por defecto), pero cada calendario pertenece exclusivamente a un único grupo.

- **Calendarios, Eventos y Tareas:** El calendario actúa como un contenedor. Mantiene una relación de uno a muchos (1:N) tanto con la tabla de eventos como con la de tareas. Un calendario puede tener muchos eventos y tareas, pero cada evento o tarea pertenece a un solo calendario.
- **Tareas y Usuarios:** Se define una relación de muchos a muchos (N:M) a través de la tabla `tareas_asignadas`. Una tarea puede ser asignada a varios usuarios, y un usuario puede tener varias tareas asignadas.
- **Grupos e Invitaciones:** Se establece una relación de uno a muchos (1:N). Un grupo puede tener múltiples tokens de invitación generados a lo largo del tiempo, pero cada token da acceso a un único y específico grupo.
- **Seguridad de Datos (Triggers):** El sistema impone una capa adicional de lógica de negocio directamente a nivel de base de datos mediante *triggers*. Estas reglas automáticas protegen la integridad de los datos ante ciertas operaciones. Por ejemplo, impiden que se elimine al último administrador de un grupo, aseguran que un grupo no pueda tener más de dos administradores simultáneamente y evitan que un administrador se degrade a sí mismo si es el único con ese rol, garantizando así que ningún grupo quede sin gestión.

Esta estructura de datos define el "qué" y el "cómo" de la información que maneja el sistema. A continuación, se detallará la API, que actúa como la capa de lógica que opera sobre esta estructura para dar vida a la aplicación.

2. Documentación de la API

La API (Interfaz de Programación de Aplicaciones) es la capa de lógica de negocio que se sitúa entre la base de datos y las aplicaciones cliente (como una interfaz web o móvil). Los siguientes módulos exponen funcionalidades específicas de manera segura y controlada, actuando como el único punto de entrada para solicitar o modificar datos, garantizando que todas las operaciones respeten las reglas de negocio y los permisos de los usuarios.

2.1 Módulo de Autenticación (`AutenticacionControlador`)

Este módulo es el responsable de gestionar el ciclo de vida de la sesión de un usuario. Su función es ser el guardián del sistema, controlando el acceso a todos los recursos protegidos y asegurando que solo los usuarios autenticados puedan realizar acciones.

- **Función: `login` (Inicio de Sesión)**
  - **Propósito:** Validar las credenciales de un usuario y establecer una sesión autenticada.
  - **Lógica Interna:** El proceso es secuencial y seguro. El sistema busca al usuario por su correo. Si no lo encuentra, o si la contraseña es incorrecta, devuelve un error genérico de "Credenciales incorrectas". Esta decisión arquitectónica prioriza la seguridad al no revelar qué correos electrónicos están registrados en el sistema, previniendo ataques de enumeración de usuarios. Si el usuario existe, se utiliza la función estándar de PHP `password_verify` para comparar de forma segura la contraseña

proporcionada con el hash almacenado. Si la verificación es exitosa, se crea una variable de sesión en el servidor que asocia esa sesión con el ID del usuario, marcándolo como "autenticado".
    - **Entradas Conceptuales:** Correo electrónico y contraseña del usuario.
    - **Salidas y Efectos:** Si las credenciales son válidas, devuelve una respuesta de éxito con los datos básicos del perfil del usuario (sin contraseña). En caso contrario, devuelve un error de credenciales incorrectas.
- **Función: `logout` (Cierre de Sesión)**
    - **Propósito:** Terminar de forma segura la sesión activa de un usuario.
    - **Lógica Interna:** La desconexión se garantiza mediante tres pasos. Primero, se limpian todas las variables de sesión en el servidor (`$_SESSION`). Segundo, se instruye al navegador del cliente para que invalide y elimine su cookie de sesión. Tercero, se destruye el archivo de sesión correspondiente en el servidor, eliminando cualquier rastro de la misma.
    - **Entradas Conceptuales:** Ninguna, opera sobre la sesión actual.
    - **Salidas y Efectos:** Una respuesta de confirmación de cierre de sesión.

Una vez que el usuario está autenticado, el sistema le permite gestionar su propia información a través del siguiente módulo.

2.2 Módulo de Gestión de Usuarios (`UsuarioControlador`)

Este módulo se encarga de todo lo relacionado con el perfil de los usuarios, abarcando desde la creación de una nueva cuenta hasta la actualización de su información personal.

- **Función: `registrar` (Registro de Usuario)**
    - **Propósito:** Crear una nueva cuenta de usuario en el sistema.
    - **Lógica Interna:** El flujo de registro sigue varios pasos de validación. Verifica que los campos requeridos estén completos y comprueba que el correo no exista previamente para evitar duplicados. Si se incluye una foto de perfil, procesa la carga del archivo, generando un nombre único para evitar colisiones. Aplica un hash seguro a la contraseña utilizando `password_hash` antes de insertar el nuevo registro en la base de datos.
    - **Entradas Conceptuales:** Nombre, correo electrónico, contraseña y, opcionalmente, un archivo de imagen para la foto de perfil.
    - **Salidas y Efectos:** Una respuesta de éxito con el ID del nuevo usuario o un mensaje de error si el correo ya existe o faltan datos.
- **Función: `actualizar` (Actualización de Perfil)**
    - **Propósito:** Permitir a un usuario autenticado modificar su propia información de perfil.
    - **Lógica Interna:** El proceso comienza verificando la autenticación del usuario. Valida los nuevos datos y, si el correo ha sido modificado, comprueba que la nueva dirección no esté en uso por otro usuario. Si se sube una nueva foto de perfil, el sistema guarda el nuevo archivo y elimina el antiguo del servidor para no acumular archivos huérfanos. Finalmente, actualiza los datos en la base de datos.

    - **Entradas Conceptuales:** Nombre, correo electrónico y, opcionalmente, un nuevo archivo de imagen. El ID del usuario se obtiene de la sesión activa.
    - **Salidas y Efectos:** Una respuesta de éxito con los datos actualizados del usuario para que la interfaz se refresque inmediatamente.

La gestión de usuarios individuales es la base, pero el verdadero potencial de la aplicación se desbloquea al organizar a estos usuarios en entidades colaborativas: los grupos.

2.3 Módulo de Gestión de Grupos (`GrupoControlador`)

Este módulo es el núcleo de la funcionalidad colaborativa. Gestiona la creación, la membresía y la administración de los grupos de trabajo, que son los espacios donde los usuarios comparten calendarios y tareas.

- **Función: `crear` (Creación de Grupo)**
    - **Propósito:** Crear un nuevo grupo de trabajo, asignando al creador como primer administrador.
    - **Lógica Interna:** Esta operación se ejecuta como una transacción de base de datos para garantizar su atomicidad. El uso de una transacción es esencial para prevenir la corrupción de datos, como la creación de un grupo sin su calendario "General" o sin un administrador asignado. El sistema inicia la transacción, inserta el nuevo grupo, añade al usuario creador a la tabla `usuarios_grupos` con el rol de 'administrador' y crea el calendario "General". Si todos los pasos son exitosos, la transacción se confirma (`commit`). Si algo falla, se revierten (`rollback`) todos los cambios.
    - **Entradas Conceptuales:** Nombre del grupo y una descripción opcional.
    - **Salidas y Efectos:** Crea un nuevo grupo y un calendario asociado. Devuelve un mensaje de éxito con el ID del nuevo grupo.
- **Función: `unirsePorCodigo` (Unirse a un Grupo)**
    - **Propósito:** Permitir a un usuario unirse a un grupo existente mediante un token de invitación.
    - **Lógica Interna:** Sigue una secuencia estricta de validaciones. Busca el token en la base de datos. Si no se encuentra, la API devuelve un error 404 (No Encontrado). Si el token existe pero ha expirado o ha alcanzado su límite de usos, la API devuelve un error 410 (Gone), indicando que el recurso ya no es válido. Finalmente, comprueba que el usuario no sea ya miembro del grupo. Si todas las validaciones son correctas, lo añade como 'miembro' e incrementa el contador de usos del token.
    - **Entradas Conceptuales:** Un token de invitación alfanumérico.
    - **Salidas y Efectos:** Añade una fila en la tabla `usuarios_grupos`. Devuelve un mensaje de éxito.
- **Función: `obtenerMiembros` (Listar Miembros de un Grupo)**
    - **Propósito:** Devolver la lista de todos los usuarios que pertenecen a un grupo específico.
    - **Lógica Interna:** La lógica de seguridad verifica que el usuario solicitante sea miembro del grupo que desea consultar. Si tiene permiso, realiza una

consulta que une las tablas de usuarios y `usuarios_grupos` para obtener el nombre, foto y rol de cada integrante.
    - **Entradas Conceptuales:** El ID del grupo a consultar.
    - **Salidas y Efectos:** Devuelve un array con la información de los miembros del grupo.
- **Función: `cambiarRol` y `expulsarMiembro` (Administración de Miembros)**
    - **Propósito:** Permitir a los administradores gestionar los permisos y la permanencia de otros miembros.
    - **Lógica Interna:** El sistema verifica que el usuario que realiza la acción sea 'administrador' del grupo. Incluye validaciones de negocio para impedir que un administrador se elimine a sí mismo o cambie su propio rol, asegurando la integridad administrativa del grupo.
    - **Entradas Conceptuales:** Para cambiar rol: ID del grupo, ID del usuario objetivo y el nuevo rol. Para expulsar: ID del grupo y ID del usuario objetivo.
    - **Salidas y Efectos:** Modifica o elimina una fila en la tabla `usuarios_grupos`.

Una de las funciones administrativas clave dentro de la gestión de grupos es la capacidad de generar tokens de acceso, gestionada por su propio módulo.

2.4 Módulo de Invitaciones (`InvitacionControlador`)

Este módulo es una herramienta de administración que permite a los líderes de grupo generar tokens de acceso para facilitar el crecimiento de sus equipos.

- **Función: `crear` (Crear Invitación)**
    - **Propósito:** Generar un nuevo token de invitación para un grupo específico.
    - **Lógica Interna:** El sistema comprueba que el usuario solicitante sea 'administrador' del grupo especificado. Luego, genera una cadena de texto aleatoria y única para el token. Finalmente, lo registra en la base de datos junto con el ID del grupo, el ID del creador y las reglas de uso. Es importante destacar que si no se especifica un número máximo de usos, la lógica de la API establece un valor por defecto de 10, sobrescribiendo el valor por defecto de 1 definido en la base de datos.
    - **Entradas Conceptuales:** El ID del grupo y, opcionalmente, el número máximo de usos.
    - **Salidas y Efectos:** Crea un nuevo registro en la tabla `invitaciones` y devuelve la información completa del token generado.

Una vez gestionadas las personas y su acceso, la API se centra en el contenido principal de la aplicación: los eventos y tareas del calendario.

2.5 Módulo de Gestión de Eventos (`EventoControlador`)

Este módulo es responsable de la lógica más compleja de la aplicación, manejando la visualización, creación y modificación de eventos, incluyendo la gestión avanzada de la recurrencia.

- **Función: `listar` (Listar Eventos)**
    - **Propósito:** Mostrar al usuario los eventos de sus calendarios en un rango de fechas determinado.
    - **Lógica Interna:** El proceso de renderizado se desarrolla en cuatro etapas:
        - **Obtención de datos base:** Consulta los eventos que se solapan con el rango de fechas solicitado, asegurando mediante `JOINs` que el usuario solo vea eventos de los grupos a los que pertenece.
        - **Manejo de excepciones:** Recupera todas las fechas de la tabla `eventos_excepciones` que correspondan a los eventos obtenidos.
        - **Expansión de recurrencia:** Itera sobre cada evento recurrente y calcula virtualmente cada una de sus ocurrencias dentro del rango de fechas visible.
        - **Filtrado y construcción final:** Para cada ocurrencia virtual generada, verifica que no esté en la lista de excepciones. Si no lo está, la añade a la lista final que se enviará al cliente.
    - **Entradas Conceptuales:** Un rango de fechas (inicio y fin) o un modo de vista (ej. 'mes').
    - **Salidas y Efectos:** Devuelve un array de objetos de evento, donde cada objeto representa una única ocurrencia visible en el calendario.
- **Función: `crear` (Creación de Evento)**
    - **Propósito:** Añadir un nuevo evento a un calendario específico.
    - **Lógica Interna:** Verifica que el usuario esté autenticado y que su rol en el grupo asociado al calendario sea 'administrador' o 'editor'. Si tiene permisos, valida los datos e inserta el nuevo evento en la base de datos.
    - **Entradas Conceptuales:** ID del calendario, título, descripción, fecha de inicio, fecha de fin y una regla de repetición opcional.
    - **Salidas y Efectos:** Crea un nuevo registro en la tabla `eventos`.
- **Función: `eliminar` (Eliminación de Evento)**
    - **Propósito:** Eliminar un evento o una única ocurrencia de un evento recurrente.
    - **Lógica Interna:** La función presenta un comportamiento dual:
        - **Modo 'serie':** Si se elimina la serie completa, se ejecuta una eliminación directa del registro del evento en la base de datos.
        - **Modo 'instancia':** Si se elimina solo una ocurrencia (ej. "cancelar la reunión de este martes"), el sistema no borra el evento principal. En su lugar, añade una entrada en la tabla `eventos_excepciones` con la fecha de esa instancia específica.
    - **Entradas Conceptuales:** El ID del evento a eliminar y un 'modo' que especifica si se borra la serie completa o solo una instancia, en cuyo caso se requiere también la fecha de la instancia.
    - **Salidas y Efectos:** O bien elimina un registro de la tabla `eventos`, o bien añade un registro a `eventos_excepciones`.

Además de los eventos, los calendarios también son contenedores de listas de tareas, cuya lógica es gestionada por el siguiente módulo.

2.6 Módulo de Gestión de Tareas (`TareaControlador`)

Este módulo gestiona la funcionalidad de "lista de tareas" o "to-do list" asociada a los calendarios. Destaca su lógica de permisos diferenciada, que depende tanto del rol del usuario como de si una tarea le ha sido asignada directamente.

- **Función: `listar` (Listar Tareas)**
    - **Propósito:** Mostrar las tareas relevantes para un usuario, ya sea en un grupo específico o en una vista global.
    - **Lógica Interna:** Presenta un comportamiento dual:
        - **Vista de Grupo:** Si se proporciona un ID de calendario, los administradores y editores ven todas las tareas. Los miembros, en cambio, solo ven las tareas que les han sido asignadas directamente.
        - **Vista Global ("Mis Tareas"):** Si no se proporciona un ID de calendario, la función devuelve una lista de todas las tareas asignadas al usuario en todos los grupos a los que pertenece.
    - **Entradas Conceptuales:** Opcionalmente, el ID de un calendario.
    - **Salidas y Efectos:** Devuelve un array de objetos de tarea, enriquecido con la información de los usuarios asignados a cada una.
- **Función: `crear` (Creación de Tarea)**
    - **Propósito:** Añadir una nueva tarea a un calendario y asignarla a uno o más usuarios.
    - **Lógica Interna:** El proceso es transaccional. El sistema verifica que el usuario tiene permisos de escritura ('administrador' o 'editor'), inicia una transacción, inserta la tarea principal en la tabla `tareas` y luego itera sobre la lista de usuarios asignados para insertar múltiples registros en la tabla `tareas_asignadas`.
    - **Entradas Conceptuales:** ID del calendario, descripción, fecha límite opcional y un array opcional con los IDs de los usuarios a asignar.
    - **Salidas y Efectos:** Crea registros en las tablas `tareas` y `tareas_asignadas`.
- **Función: `actualizar` (Actualización de Tarea)**
    - **Propósito:** Permitir a un administrador o editor modificar los detalles de una tarea, incluyendo su descripción, fecha límite y la lista de usuarios asignados.
    - **Lógica Interna:** La operación es transaccional y verifica primero los permisos de escritura del usuario. Actualiza los campos básicos (descripción, fecha límite) en la tabla `tareas`. Para la lista de asignados, aplica una estrategia de "borrar y reinsertar": elimina todas las entradas existentes para esa tarea en `tareas_asignadas` y luego inserta los nuevos registros. Este patrón garantiza que la lista de asignados refleje exactamente el estado enviado en la solicitud.
    - **Entradas Conceptuales:** ID de la tarea y, opcionalmente, la nueva descripción, fecha límite y un array con la lista completa de IDs de usuarios asignados.
    - **Salidas y Efectos:** Modifica la tarea y sus asignaciones en la base de datos.
- **Función: `cambiarEstado` (Cambiar Estado de Tarea)**

- **Propósito:** Marcar una tarea como 'pendiente', 'en_proceso' o 'completada'.
- **Lógica Interna:** Analiza una lógica de permisos especial: un usuario puede cambiar el estado si es 'administrador' o 'editor' del grupo, O si es un 'miembro' pero la tarea le ha sido asignada específicamente. Esto permite que cualquier responsable pueda marcar su trabajo como hecho sin necesidad de permisos elevados.
- **Entradas Conceptuales:** El ID de la tarea y el nuevo estado.
- **Salidas y Efectos:** Actualiza el campo `estado` en la tabla `tareas`.