INSTITUTO DE EDUCACIÓN SUPERIOR TECNOLÓGICO PÚBLICO "VÍCTOR RAÚL HAYA DE LA TORE"

INFORME DE PRÁCTICAS PREPROFESIONALES

MÓDULO II:

Desarrollo de Sistemas de Información con Gestión de Base de Datos

TÍTULO DEL PROYECTO:

Diseño y Creación de la Base de Datos para el Sistema Web Institucional de la Municipalidad Distrital de Pativilca

Presentado por: Ramos Requena Jose Leandro

Programa de estudios: Desarrollo de sistemas e información

Supervisor: Ing.Sergio Alejandro Lopez Arroyo

AÑO: 2025

I.Presentación

El presente informe detalla de manera completa el proceso de análisis, diseño e implementación de la base de datos del Sistema de Gestión de Archivos, desarrollado como parte del módulo técnico orientado a la construcción y administración de sistemas de información con soporte en bases de datos.

Este sistema tiene como finalidad gestionar de forma organizada los archivos digitales generados dentro de la institución, permitiendo controlar usuarios, roles, carpetas, archivos y procesos auxiliares como la recuperación de contraseñas. Con ello, se busca garantizar un ambiente estructurado, seguro y eficiente para la manipulación y almacenamiento de información.

A lo largo del documento se describen las decisiones técnicas adoptadas en cada etapa del desarrollo, la estructura final de la base de datos, la elaboración del modelo entidad–relación, la conversión al modelo lógico, el diccionario de datos y las validaciones implementadas para asegurar la consistencia del sistema. Asimismo, se incluye la justificación de cada diseño aplicado y la forma en que se abordaron las dependencias y restricciones surgidas durante el proceso.

El desarrollo de la base de datos se realizó aplicando principios de normalización, integridad referencial y buenas prácticas de modelado, haciendo uso adecuado de claves primarias, claves foráneas, tipos de datos, restricciones y relaciones entre entidades. Este enfoque permitió obtener una estructura sólida, escalable y preparada para integrarse con un sistema web, asegurando la correcta organización de los datos y la coherencia en su almacenamiento y mantenimiento.

II. Datos generales

1. **Nombre y Apellidos del Estudiante:** Jose Leandro Ramos Requena
2. **Programa de Estudios:** Desarrollo de sistemas e información (DSI)
3. **Nombre del Módulo Técnico Profesional:** Desarrollo de sistemas de información con gestión de base de datos
4. **Periodo Académico:** 2025
5. **Razón Social de la Empresa** o Institución: Municipalidad distrital de Pativilca
6. **Actividad de la Empresa o Institución:** Gestión administrativa y cultural del distrito, servicios al ciudadano y difusión institucional
7. **Lugar/Área de práctica:** Oficinas del área de Tecnología y Sistemas de la Municipalidad
8. **Ejecución de la EFSRT:**
   Fecha de inicio: 06/11/2025 **Fecha de término:** 06/12/2025
9. **Total, de horas acumuladas:** 128
10. **jefe o Autoridad bajo cuya orientación se realizó la EFSRT:**
    Nombre y Apellidos: Mg. Gina Tatiana Fernández Caldas
    Cargo en la Empresa o Institución: Practicante

III. Aspecto técnico de la experiencia formativa en situaciones reales de trabajo

a) Organización de las prácticas en la empresa o institución

El flujo de prácticas está orientada al diseño modelado e implementación de una base de datos para la municipalidad distrital de Pativilca utilizando XAMPP, PHPMYADMIN y DIA como herramientas principales. El proceso se estructuró de la siguiente manera

1. **Levantamiento de información y requisitos** (06/11/2025 – 08/11/2025)

Se revisó información general sobre la municipalidad, especialmente su organigrama, para identificar áreas, unidades y posibles necesidades documentales.

- No se contaba con sistemas previos internos accesibles, por lo que se tomaron como referencia proyectos similares encontrados en línea y bases de datos de gestión documental.
- Se recopilaron los requisitos mínimos funcionales: usuarios, roles, permisos, carpetas y archivos.
- Se realizaron preguntas puntuales al área usuaria y al asesor encargado para validar ciertos criterios (jerarquías, accesos y estructura general).
- Además, como parte del análisis funcional inicial, se elaboró un diagrama de casos de uso que permitió representar gráficamente las acciones principales que los distintos tipos de usuarios pueden realizar dentro del sistema. Este diagrama ayudó a visualizar los procesos clave y sirvió como base conceptual para el modelado posterior.

2. **Análisis de requisitos y normalización** (09/11/2025 – 11/11/2025)

Una vez que los requisitos fueron recopilados, se inició una fase de análisis para determinar la estructura técnica más adecuada:

- Se identificaron las entidades centrales del sistema, entre ellas: Usuarios, Roles, Permisos, Carpetas, Archivos y Password Reset. Cada entidad se evaluó para decidir si requería atributos adicionales o restricciones especiales.
- Se diseñó la estructura de "usuarios–roles–permisos" con un enfoque general y reutilizable, pensando en un modelo modular que pudiera adaptarse a otros proyectos futuros y no únicamente al contexto municipal.
- Se analizó la necesidad de implementar jerarquías de carpetas, lo cual surgió directamente del organigrama administrativo. Esta decisión llevó a considerar una relación recursiva dentro de la misma entidad Carpeta (carpeta padre → carpeta hija).
- Se revisaron los atributos obligatorios, la integridad de las claves y las reglas para evitar anomalías en la inserción o modificación de datos.
- Se aplicaron las formas normales, evitando campos redundantes o dependencias parciales. Esta etapa permitió depurar y simplificar el diseño sin perder funcionalidad.

3. **Diseño del modelo entidad–relación (ER)** (12/11/2025 – 16/11/2025)

Con los requisitos definidos y las entidades principales establecidas, se procedió al diseño del modelo ER utilizando la herramienta DIA. El proceso comenzó con la elaboración de diagramas preliminares, que fueron ajustados conforme se revisaban las relaciones y se organizaban los atributos de forma más clara.

En la etapa de depuración se establecieron las relaciones principales del sistema, basadas en el comportamiento esperado del gestor de archivos:

- **Usuario – Rol (N,N)** mediante la relación *tiene*, permitiendo la asociación conceptual de varios usuarios con varios roles.
- **Rol – Permisos (N,N)** a través de la relación *tiene*, representando que un rol puede utilizar distintos permisos y un permiso puede estar vinculado a más de un rol.
- **Usuario – Carpeta (1,N)** por medio de la relación *crea*, indicando que un usuario puede crear varias carpetas.
- **Carpeta – Archivo (1,N)** mediante la relación *contiene*, donde una carpeta puede almacenar múltiples archivos.
- **Usuario – Password Reset (1,1)** usando la relación *usa*, ya que cada token de recuperación se asocia a un único usuario.

Se representaron tanto atributos simples como compuestos, claves primarias, claves foráneas.Se verificó que el diagrama ER coincidiera con los requisitos iniciales y que su estructura fuera flexible para futuras ampliaciones del sistema.

Este diseño se convirtió en la base principal para la creación del modelo lógico y físico.

4. **Diseño físico de la base de datos** (17/11/2025 – 21/11/2025)

Tras validar el modelo entidad–relación, se procedió a convertirlo en un diseño físico listo para MySQL:

- Se tradujeron las entidades del diagrama ER en tablas normalizadas, definiendo claves primarias (INT UNSIGNED con AUTO_INCREMENT), tipos de datos y longitudes acordes al uso esperado.
- Se configuraron las claves foráneas que enlazan las tablas principales (tb_user → tb_rol, tb_carpetas → tb_user, tb_archivo → tb_carpetas, tb_password_reset → tb_user, tb_permision → tb_rol), y —en el caso de existir— la referencia para jerarquía de carpetas vía carpeta_padre_id.
- Se eligió el motor **InnoDB** y el cotejamiento **utf8mb4_unicode_ci** para asegurar integridad referencial y compatibilidad con caracteres especiales.
- Se definieron restricciones de integridad, índices en campos de consulta frecuente y se aplicó **ON DELETE CASCADE** en las relaciones que requieren eliminación ordenada de registros relacionados (principalmente en la relación carpetas→archivos y usuario→carpetas).

5. **Implementación en MySQL (XAMPP)** (22/11/2025 – 26/11/2025)

Con el diseño físico finalizado, se implementó la base de datos en MySQL utilizando XAMPP y phpMyAdmin:

- Se creó la base de datos seleccionando el cotejamiento **utf8mb4_unicode_ci**, asegurando compatibilidad con caracteres especiales.
- Se construyeron las tablas mediante sentencias SQL y se configuraron las claves primarias, atributos y restricciones definidas en el diseño físico.
- Se integraron las relaciones entre tablas, verificando que las claves foráneas fueran aceptadas correctamente y aplicando **ON DELETE CASCADE** en las relaciones que lo requerían.
- Se insertaron registros de prueba, incluyendo usuarios, roles, carpetas y archivos ficticios para comprobar el funcionamiento general.
- Se añadieron índices en campos utilizados con frecuencia en consultas para mejorar el rendimiento esperado del sistema.

La base de datos quedó operativa y lista para iniciar pruebas más avanzadas e integrarse con el sistema de gestión de archivos.

6. **Integración y pruebas de validación SQL** (27/11/2025 – 01/12/2025)

Tras la implementación, se ejecutó una fase de pruebas orientada a asegurar la integridad y funcionamiento adecuado del sistema:

- Se realizaron pruebas con operaciones básicas (SELECT, INSERT, UPDATE y DELETE) sobre todas las tablas para confirmar su correcto desempeño.
- Se probó de manera específica el funcionamiento de las **reglas de borrado en cascada**, verificando que al eliminar una carpeta se eliminaran correctamente sus subcarpetas y archivos.
- Se identificaron pequeños ajustes necesarios, como modificación de nombres, tipos de dato o restricciones, los cuales fueron aplicados para mejorar la consistencia del sistema.

Estas pruebas aseguraron que la base estuviera libre de errores estructurales antes de su validación final.

7. **Verificación y validación final con el área usuario (**02/12/2025 – 04/12/2025**)**

   Antes de generar la versión final:

   - Se presentó el diseño general de la base de datos y su funcionamiento al área solicitante.
   - Se validó que la **jerarquía de carpetas y su lógica interna** coincidieran con los niveles de organización documentaria de la municipalidad.
   - Se explicó el funcionamiento del sistema de roles y permisos, confirmando que cumplía con los requerimientos de control de acceso solicitados.

   Esta validación permitió garantizar que la base de datos respondiera a las necesidades reales del entorno institucional.

8. **Implementación final de la base de datos** (05/12/2025 – 06/12/2025)

Finalmente, se generó la versión final de la base datos:

- Se exportó el archivo SQL completo desde phpMyAdmin, incluyendo estructuras, restricciones y datos de prueba.
- Se documentó formalmente la base de datos, incorporando el modelo entidad–relación, el modelo lógico, los diagramas físicos y la descripción de cada tabla y sus relaciones.

La base de datos quedó preparada para su integración en futuros módulos de software que implementen para un futuro sistema de gestión documental de la municipalidad

b) Métodos, técnicas o instrumentos utilizados

Durante el desarrollo de la experiencia formativa se aplicaron diversos métodos, técnicas e instrumentos orientados al análisis, diseño y construcción de una base de datos funcional para la Municipalidad Distrital de Pativilca. Estas herramientas permitieron abordar el proyecto de manera ordenada y garantizar la coherencia entre los requisitos institucionales y la solución final implementada.

En primer lugar, se emplearon métodos de levantamiento y análisis de información, los cuales incluyeron la revisión del organigrama institucional, la observación de los flujos documentarios y la consulta directa con el área usuaria y el asesor de prácticas. Este proceso permitió identificar las necesidades reales de gestión documental, así como los actores que formarían parte del sistema y las acciones que debían realizarse dentro del mismo. Para complementar esta fase, se analizaron sistemas de referencia y proyectos similares encontrados en línea, debido a la ausencia de sistemas previos dentro de la institución. Estos modelos sirvieron como guía inicial para definir módulos funcionales y estructuras de datos reutilizables.

Como parte de las técnicas de análisis funcional, se elaboró un diagrama de casos de uso utilizando principios de modelado UML. Este instrumento permitió visualizar las interacciones principales entre los distintos tipos de usuarios —administradores y usuarios comunes— y las funcionalidades clave del sistema. La representación gráfica facilitó la validación de requisitos, ayudó a delimitar el alcance del proyecto y sirvió como puente conceptual hacia las fases de diseño estructural.

Para la construcción de la base de datos se utilizaron técnicas de modelado conceptual y lógico, elaborando un modelo entidad–relación (ER) en la herramienta DIA. Este modelo permitió identificar entidades, atributos, relaciones y cardinalidades, incluyendo relaciones recursivas como la jerarquía de carpetas. Una vez validado el modelo conceptual, se aplicaron técnicas de normalización (1FN, 2FN y 3FN) con el objetivo de evitar redundancias, mejorar la integridad de los datos y asegurar un diseño consistente. Posteriormente, el modelo conceptual fue traducido a un modelo físico, definiendo tipos de datos, claves primarias, claves foráneas, índices y reglas de integridad.

En la fase de implementación se emplearon como instrumentos principales las herramientas del paquete XAMPP, específicamente phpMyAdmin como entorno gráfico para la creación, gestión y prueba de la base de datos en MySQL. Estas herramientas permitieron crear las tablas, configurar sus relaciones e insertar registros de prueba. Adicionalmente, se aplicaron técnicas de validación mediante consultas SQL, evaluando operaciones básicas (SELECT, INSERT, UPDATE, DELETE) y verificando el funcionamiento correcto de restricciones como borrado en cascada y actualizaciones recursivas.

Finalmente, se utilizaron técnicas de documentación para registrar la estructura de la base de datos, justificando las decisiones de diseño y describiendo las entidades, atributos y relaciones utilizadas. Esto permitió dejar una base sólida para futuras ampliaciones o integraciones del sistema.

En conjunto, los métodos, técnicas e instrumentos empleados permitieron desarrollar un proyecto estructurado, técnicamente fundamentado y alineado con las necesidades documentales reales de la institución.

c) Secuencia de las tareas y/o actividades realizadas

Actividades realizadas	Fundamento de las actividades	Récord deseado	Récord logrado
Levantamiento de información y requisitos	**1. Reunión inicial con el gerente / responsable del área** se revisó documentación interna disponible (organigrama, estructura administrativa, funciones institucionales) y se realizaron consultas puntuales al asesor y al área usuaria cuando surgían dudas técnicas o funcionales. Esto permitió comprender el contexto, identificar qué información manejaba la municipalidad y determinar qué elementos debían ser incluidos en la base de datos. **2. Recopilación de información institucional** Se recopiló la información necesaria a partir de fuentes accesibles públicamente y referencias técnicas. Como punto de partida, se utilizó el organigrama institucional disponible para comprender la estructura de áreas y la jerarquía administrativa, ya que no se tuvo acceso directo a listados internos o documentación administrativa detallada. Además, se revisaron ejemplos de sistemas similares y modelos de bases de datos disponibles en línea, lo que permitió identificar patrones comunes en la gestión de usuarios, roles, permisos, contenidos institucionales y recuperación de contraseñas. Parte del diseño también se apoyó en estructuras y tablas de proyectos previos, que fueron adaptadas y generalizadas para que funcionaran en el contexto de la municipalidad. **3. Definición de requisitos funcional de la base de datos** Se establecieron los requisitos que la base de datos debía cumplir para que el sistema de gestión de archivos pudiera operar correctamente. Entre los requisitos funcionales, se consideraron aspectos como: • Registrar usuarios con distintos roles dentro del sistema. • Administrar permisos asociados a cada rol. • Permitir la creación de carpetas y subcarpetas bajo una estructura jerárquica. • Asociar archivos digitales a las carpetas creadas. • Gestionar procesos de recuperación de contraseña mediante tokens únicos. • Mantener integridad referencial entre todas las entidades.		

Matriz de requisitos

Tipo de requisito	Descripción	Estado	Observaciones
RF	Registro de usuarios con datos básicos (nombre, correo, contraseña, rol)	Cumplido	Validación de correo único aplicada correctamente
RF	Gestión de roles para controlar permisos del sistema	Cumplido	Vinculado mediante FK rol_id
RF	Creación de carpetas por usuario	Cumplido	Se registra usuario creador y fecha
RF	Creación de subcarpetas dentro de una carpeta principal	Cumplido	Implementado mediante campo carpeta_padre_id
RF	Registro de archivos dentro de una carpeta	Cumplido	Se guarda nombre, formato, ruta y usuario creador
RF	Eliminación automática de archivos al borrar una carpeta	Cumplido	ON DELETE CASCADE
RF	Recuperación de contraseña mediante tokens	Cumplido	Tabla password_reset enlazada a usuarios
RF	Integridad entre tablas mediante claves foráneas	Cumplido	Relaciones verificadas en MySQL
NF	Normalización de la base de datos hasta 3FN	Cumplido	Estructura libre de redundancias
NF	Uso de InnoDB para manejo seguro de FK	Cumplido	Todas las tablas usan InnoDB
NF	Uso de utf8mb4_unicode_ci para compatibilidad total	Cumplido	Evita errores con acentos y caracteres especiales
NF	Rendimiento adecuado en consultas	Parcial	Índices aplicados, faltan pruebas avanzadas
NF	Escalabilidad para más módulos o tablas	Cumplido	Estructura modular y ampliable
Limitación	Problemas iniciales con claves foráneas marcadas como UNIQUE	Resuelto	Requirió reconstrucción de tablas
Limitación	Dependencia total de XAMPP para pruebas	Declarado	No se probó en servidor real
Limitación	Sesiones sin internet afectaron la consulta de documentación	Declarado	Retrasos en momentos clave

4. Definición de requisitos no funcionales

También se establecieron características no funcionales que la base debía cumplir para garantizar un uso adecuado:

- **Seguridad:** proteger la información de usuarios, carpetas y archivos.
- **Consistencia:** evitar duplicaciones y mantener relaciones válidas entre datos.
- **Escalabilidad:** permitir añadir más entidades o campos en el futuro sin afectar la estructura existente.
- **Rendimiento:** ejecutar consultas de manera rápida mediante índices y normalización.
- **Mantenibilidad:** facilitar actualizaciones y comprensión del modelo por futuros desarrolladores.

Análisis de Requisitos

1. Revisión de la información recopilada

Una vez reunida la información inicial, se procedió a revisar y organizar todo el material obtenido sobre la estructura administrativa y el funcionamiento actual de la gestión de archivos en la municipalidad.
Esta revisión permitió identificar cuáles eran los procesos que el sistema debía representar: creación de carpetas, manejo de archivos, asignación de roles, permisos diferenciados y acceso seguro para los usuarios.

	También se repasaron todos los elementos identificados durante el levantamiento de información, como las entidades principales, las relaciones entre ellas y las necesidades específicas del área usuaria. Esta revisión fue esencial para descartar información irrelevante y centrarse únicamente en los elementos que impactaban directamente en el diseño de la base de datos. **2. Clasificación de requisitos** Con la información ya organizada, se realizó una clasificación de los requisitos identificados, separándolos en **requisitos funcionales** y **requisitos no funcionales**. • Los **funcionales** abarcaron todo lo relacionado con acciones concretas que el sistema debía permitir: registrar usuarios, crear carpetas, almacenar archivos, aplicar roles y permisos, y gestionar recuperaciones de contraseña. • Los **no funcionales** se enfocaron en las características internas que debía tener la base de datos, como seguridad, consistencia, rendimiento, integridad y facilidad de mantenimiento. Esta clasificación permitió tener una visión más clara del alcance del sistema y definir qué partes del modelo debían recibir mayor atención durante el diseño. **3. Priorización de requisitos** Luego de clasificar todos los requisitos, se procedió a priorizarlos según su importancia para el funcionamiento del sistema. Los requisitos considerados esenciales fueron aquellos directamente relacionados con: • la identificación de usuarios, • el control de roles y permisos, • la creación y organización de carpetas,		

	• el almacenamiento de archivos en cada carpeta. Estos elementos conforman la base del sistema y, por tanto, recibieron la prioridad más alta. Otros requisitos, como la auditoría, mejora del rendimiento o ampliaciones futuras, se registraron como prioridades secundarias. De esta manera se garantizó que la base de datos cumpliera primero con las funciones fundamentales y que luego pudiera ampliarse de forma sencilla **4. Dependencias y limitaciones** Durante el análisis se identificaron varias dependencias que era necesario mantener claras desde el inicio. Los usuarios requieren un rol para definir sus permisos, las carpetas dependen del usuario que las crea y los archivos deben estar asociados a una carpeta específica. Estas relaciones guiaron la forma en que se definieron las claves foráneas para asegurar que la información se mantenga ordenada. En cuanto a las limitaciones técnicas, el principal inconveniente se dio al momento de crear las claves foráneas. Por descuido, algunas relaciones terminaron marcándose como **UNIQUE** en lugar de generar un índice, lo que provocaba restricciones incorrectas y obligaba a reconstruir las tablas o ajustar la estructura. También se evaluaron distintas formas de representar la jerarquía de carpetas, pero finalmente se mantuvo la opción de ***carpeta_padre_id*** por ser más simple y compatible con la normalización.		
Diseño del diagrama entidad-relación	Con las entidades principales ya identificadas, se procedió al diseño del modelo entidad–relación utilizando la herramienta DIA. El diagrama se construyó de forma progresiva, iniciando con una disposición básica de las entidades para luego ir añadiendo atributos, relaciones y cardinalidades hasta obtener una estructura coherente. Las primeras versiones del diagrama incluían únicamente a **Usuario**, **Rol**, **Carpetas** y **Archivo**, ya que eran las entidades esenciales del sistema. A medida que el análisis avanzaba se identificó la necesidad de incorporar dos entidades adicionales: **Permisos**, para complementar el control de acceso de los roles, y		

	Password Reset, destinada al manejo de tokens de recuperación de contraseña. Una vez definidas todas las entidades, se establecieron sus relaciones conforme al diseño funcional del sistema. Las cardinalidades fueron ajustadas según lo representado gráficamente en el diagrama, quedando de la siguiente manera: • **Usuario – Rol (N,N)** a través de la relación *tiene*, permitiendo que en el modelo conceptual un usuario pueda asociarse a distintos roles y un rol pueda estar vinculado a múltiples usuarios. • **Rol – Permisos (N,N)** mediante la relación *tiene*, ya que un rol puede disponer de varios permisos y un permiso puede ser compartido por más de un rol. • **Usuario – Carpetas (1,N)** por medio de la relación *crea*, ya que un usuario puede crear varias carpetas dentro del sistema. • **Carpeta – Archivo (1,N)** a través de la relación *contiene*, indicando que cada carpeta puede almacenar múltiples archivos. • **Usuario – Password Reset (1,1)** mediante la relación *usa*, representando que el token de recuperación está asociado directamente a un usuario.		
Diseño físico de la base de datos	El diseño físico de la base de datos se elaboró a partir del modelo lógico previamente definido, utilizando este como referencia directa para determinar la estructura final de las tablas en MySQL. En esta etapa se especificaron los tipos de datos, claves primarias, claves foráneas, restricciones y reglas de integridad necesarias para garantizar un funcionamiento correcto del sistema. A partir del modelo lógico, se crearon las tablas **tb_user, tb_rol, tb_permision, tb_carpetas, tb_archivo y tb_password_reset**, cada una con sus atributos definidos según el comportamiento esperado de la aplicación. Para los identificadores principales se emplearon campos de tipo INT con la propiedad UNSIGNED y AUTO_INCREMENT, lo que asegura valores consecutivos y positivos.		

	En cuanto a los tipos de datos, se seleccionaron longitudes apropiadas para cada atributo: – VARCHAR(255) para nombres, rutas y correos, – TEXT para las contraseñas cifradas, – VARCHAR(50) para los tipos y colores, – INT para tamaños de archivos, – TIMESTAMP para los campos de auditoría (*created_at* y *updated_at*). Se definieron también las claves foráneas que enlazan las tablas, manteniendo relaciones coherentes con el modelo conceptual. Entre estas relaciones destacan: – **tb_user → tb_rol**, – **tb_carpetas → tb_user**, – **tb_archivo → tb_carpetas**, – **tb_password_reset → tb_user**, – **tb_permision → tb_rol**. Todas las claves foráneas fueron configuradas con la regla **ON DELETE CASCADE**, ya que este comportamiento facilita la eliminación ordenada de registros relacionados, especialmente en la estructura de carpetas y archivos.		
Implementación en MySQL (XAMPP)	La implementación de la base de datos se realizó en un entorno local configurado con XAMPP, utilizando MySQL como motor principal y phpMyAdmin como herramienta de administración. El proceso se desarrolló de manera ordenada siguiendo los pasos habituales para la creación y despliegue de una base de datos. En primer lugar, se inició el módulo **MySQL** desde el panel de control de XAMPP y se accedió a phpMyAdmin para gestionar de forma gráfica las operaciones iniciales. Desde la sección *Nueva Base de Datos* se creó el esquema correspondiente, seleccionando el cotejamiento **utf8mb4_unicode_ci** para asegurar compatibilidad con caracteres especiales y evitar problemas en nombres de carpetas, roles o usuarios. Una vez creado el esquema, se procedió a generar la estructura física de las tablas. Para lograr precisión y evitar		

	errores en la creación automática, todas las tablas fueron construidas mediante **sentencias SQL** en la pestaña *SQL* de phpMyAdmin. En esta etapa se declararon: • claves primarias con `AUTO_INCREMENT`; • claves foráneas para mantener la integridad entre usuarios, roles, carpetas, archivos y permisos; • índices para optimizar la búsqueda en campos como `id_usuario` o `id_carpeta`; • reglas **ON DELETE CASCADE** para garantizar la eliminación correcta de carpetas y archivos asociados. Después de crear la estructura completa se verificó que todas las relaciones fueran aceptadas por MySQL, especialmente aquellas que anteriormente generaban restricciones incorrectas por campos marcados como `UNIQUE`. Realizados los ajustes finales, se ejecutó una **carga de datos inicial**, incluyendo usuarios de prueba, roles, carpetas base y registros de permisos. Finalmente, se realizaron consultas simples de validación —inserciones, eliminaciones y relaciones entre tablas— para confirmar que la base de datos estuviera operando correctamente en el entorno XAMPP. Con estas pruebas, la implementación quedó lista para su integración con el sistema de gestión de archivos.		
Integración y pruebas de validación SQL	Con la base de datos implementada, se realizó una fase de integración y validación para comprobar que todas las tablas, relaciones y restricciones funcionaran de acuerdo al diseño establecido. Esta etapa consistió en ejecutar un conjunto de consultas SQL destinadas a evaluar el comportamiento de la base de datos ante operaciones de inserción, consulta, actualización y eliminación. En primer lugar, se insertaron registros de prueba en las tablas principales. Para ello se ejecutaron consultas de inserción en **tb_rol**, **tb_user**, **tb_carpetas** y **tb_archivo**, lo que permitió verificar que las claves primarias y foráneas estuvieran correctamente configuradas y que no se generaran errores por restricciones mal aplicadas. Estas inserciones también confirmaron que las relaciones entre usuarios, roles, carpetas y archivos estaban enlazadas correctamente. Posteriormente, se realizaron consultas con **JOIN** para validar la integridad referencial. Estas consultas permitieron comprobar		

	que los usuarios heredaban su rol correctamente y que las carpetas mostraban sus archivos asociados sin inconsistencias. Además, se ejecutaron consultas de eliminación para verificar el funcionamiento de **ON DELETE CASCADE**, confirmando que al eliminar una carpeta se eliminaban automáticamente los archivos que dependían de ella. Entre las consultas utilizadas en esta fase se incluyeron: • Inserción de roles y usuarios para validar claves foráneas • Registro de carpetas y archivos para probar la estructura jerárquica • Consultas JOIN entre usuarios y roles, y entre carpetas y archivos • Eliminaciones controladas para comprobar el comportamiento en cascada Los resultados de esta etapa permitieron confirmar que: • Las claves foráneas estaban correctamente enlazadas • No existían inconsistencias entre entidades • Las reglas de integridad funcionaban según lo esperado • La base de datos respondía adecuadamente a operaciones básicas de prueba Con estas validaciones, la estructura quedó lista para su uso		
Verificación y validación final con el área usuario	Una vez completadas las pruebas de funcionamiento, se presentó la base de datos al encargado del área de Sistemas para realizar la validación final. Durante esta revisión se explicó la estructura general del modelo, las relaciones entre las tablas y el comportamiento de las reglas de integridad, especialmente en los módulos de usuarios, roles, carpetas y archivos. El encargado evaluó el diseño en función de los requerimientos planteados al inicio del proyecto y proporcionó observaciones orientadas a mejorar la organización de la información. Entre los comentarios recibidos se incluyeron ajustes menores en la denominación de algunos campos, la necesidad de estandarizar ciertos nombres de tablas y la recomendación de reforzar la estructura de roles y permisos para facilitar su uso en el futuro. A partir de estas observaciones se realizaron las modificaciones correspondientes en el modelo físico y en la estructura SQL,		

	asegurando que los cambios no afectaran la integridad del sistema ni las relaciones ya definidas. Después de aplicar los ajustes, se presentó nuevamente la base de datos para una última revisión, obteniendo la conformidad del área usuaria. Con esta validación final, la base de datos quedó aprobada y lista para integrarse con el sistema de gestión de archivos que se desarrollará en etapas posteriores.		
Implementación final de la base de datos	Con la validación del área de Sistemas completada, se procedió a la implementación final de la base de datos. En esta etapa, la estructura ya terminada se exportó desde phpMyAdmin como archivo **.sql**, incluyendo la definición de tablas, relaciones, índices y los registros iniciales de configuración. Este archivo se preparó como versión definitiva para su uso institucional. Posteriormente, la base de datos fue importada en el servidor local de la municipalidad mediante phpMyAdmin, verificando que todas las tablas y claves foráneas se cargaran correctamente sin generar errores de integridad. Después de la importación, se realizaron pruebas básicas de funcionamiento para confirmar que las relaciones, eliminaciones en cascada y consultas de referencia operaran de la misma forma que en el entorno de desarrollo. Como parte del proceso de implementación, se entregó también la documentación técnica correspondiente, que incluye el diagrama entidad–relación, el modelo lógico, la descripción de tablas y las reglas de integridad aplicadas.		

d) Dificultades encontradas

Durante el desarrollo de la base de datos surgieron diversas dificultades que influyeron tanto en el tiempo de trabajo como en las decisiones de diseño que se tomaron a lo largo del proyecto. Una de las complicaciones más constantes estuvo relacionada con la creación y configuración de las claves foráneas. En varias ocasiones, al intentar establecer relaciones entre tablas, MySQL interpretaba algunos campos como **UNIQUE** de manera automática. Esto provocaba errores al momento de enlazar las tablas y obligaba a revisar nuevamente la estructura, corregir las relaciones y, en algunos casos, reconstruir por completo ciertas tablas para aplicar correctamente las restricciones. Este inconveniente retrasó el avance previsto, ya que era necesario validar cada modificación antes de continuar con la siguiente etapa del diseño físico.

Otra dificultad importante surgió al definir cómo organizar la estructura de carpetas dentro del sistema. Aunque inicialmente parecía sencillo, al avanzar con el modelo conceptual aparecieron diferentes opciones para representar la jerarquía, lo que generó varias versiones previas del diagrama. Se analizaron alternativas como manejar niveles de profundidad, utilizar rutas completas o aplicar estructuras más complejas. Sin embargo, estas opciones no terminaban de adaptarse a los requerimientos funcionales, por lo que fue necesario realizar pruebas repetidas y ajustar varios modelos antes de obtener una estructura coherente y fácil de implementar.

También se presentaron dificultades durante la validación de las relaciones que utilizaban **ON DELETE CASCADE**. En las primeras pruebas, algunas eliminaciones provocaban efectos encadenados que no eran los esperados, borrando más registros de los necesarios. Esto obligó a revisar cuidadosamente las reglas de integridad, identificar cuáles relaciones realmente necesitaban cascada y en cuáles era conveniente mantener un comportamiento más controlado para evitar pérdidas de información no deseadas durante las pruebas.

A nivel operativo, también hubo momentos que afectaron el desarrollo. En determinadas sesiones de trabajo no se contaba con conexión a internet, lo que dificultó consultar documentación técnica, ejemplos de implementación o referencias sobre errores específicos de MySQL. Esta limitación obligó a resolver ciertos problemas únicamente mediante prueba y error dentro del entorno local, lo cual consumió más tiempo de lo previsto. Sin embargo, este proceso permitió comprender de manera más profunda el comportamiento del motor de base de datos y mejorar la precisión del diseño.

A pesar de estas dificultades, todas fueron superadas mediante ajustes progresivos, revisión constante del modelo y ejecución de pruebas en MySQL. Los obstáculos encontrados se convirtieron en oportunidades para refinar el diseño, corregir errores que podrían haber causado problemas más adelante e incrementar la estabilidad del sistema. Gracias a estas iteraciones, se logró obtener una base de datos sólida, funcional y alineada con los requerimientos establecidos por la municipalidad.

e) Logros alcanzados

A lo largo del desarrollo del proyecto se alcanzaron una serie de logros que permitieron consolidar un sistema de base de datos sólido y ajustado a las necesidades de la municipalidad. El primero de ellos fue la correcta identificación y organización de los requisitos funcionales y no funcionales. Gracias a este proceso inicial fue posible definir con claridad las entidades principales, las relaciones necesarias y el comportamiento esperado del sistema de gestión de archivos. Este trabajo permitió sentar bases firmes para las siguientes etapas.

Otro logro importante fue la elaboración del modelo entidad–relación, el cual pasó por varias revisiones hasta obtener una estructura clara, coherente y funcional. Durante esta etapa se lograron depurar atributos, ajustar cardinalidades y representar los elementos esenciales del sistema, tales como usuarios, roles, permisos, carpetas, archivos y la gestión de recuperación de contraseñas. El diagrama resultante fue fundamental para avanzar hacia el diseño lógico.

Posteriormente, se consiguió construir un modelo lógico completo, transformando cada entidad del MER en tablas normalizadas y definiendo de manera precisa claves primarias, claves foráneas y restricciones. Este modelo fue la base directa para generar el diseño físico en MySQL, lo que representó otro logro significativo dentro del proyecto.

En el diseño físico de la base de datos, se lograron configurar adecuadamente los tipos de datos, el cotejamiento general del esquema, los índices de rendimiento y las reglas de integridad referencial, incluyendo la aplicación correcta de ON DELETE CASCADE en las relaciones que lo requerían. Además, se corrigieron diversos errores que surgieron durante la creación de claves foráneas y se superaron dificultades técnicas que aparecieron en diferentes versiones del diseño.

La implementación completa de la base de datos en MySQL, utilizando XAMPP y phpMyAdmin, representa uno de los logros más relevantes. Todas las tablas fueron creadas mediante sentencias SQL, garantizando precisión y estabilidad en la estructura. Asimismo, se realizaron pruebas de funcionamiento

mediante consultas de inserción, actualización, eliminación y consultas JOIN, las cuales confirmaron que la base de datos cumplía con los comportamientos esperados.

Otro resultado destacable fue la capacidad para resolver problemas operativos, como errores de integridad, configuraciones mal aplicadas y dificultades por la falta de conexión a internet en algunos momentos. Estos inconvenientes fueron superados mediante pruebas directas, ajustes progresivos y análisis del comportamiento de MySQL, lo que fortaleció el diseño final.

Finalmente, se logró presentar la base de datos al área de Sistemas de la municipalidad, recibir sus observaciones y aplicar los ajustes solicitados. Con ello, la base de datos quedó validada por el área usuaria y lista para su integración en el futuro sistema de gestión documental. La entrega del archivo SQL final y la documentación técnica completa consolidan el cierre exitoso del proyecto.

IV Conclusiones

El desarrollo de la base de datos permitió obtener una estructura sólida, organizada y coherente con las necesidades del sistema de gestión de archivos de la municipalidad. A lo largo del proyecto se logró diseñar correctamente el modelo entidad–relación, el modelo lógico y el modelo físico, los cuales sirvieron como base para la construcción final del esquema en MySQL. Estos elementos permitieron representar de forma clara a los usuarios, roles, permisos, carpetas y archivos, garantizando un manejo ordenado de la información.

La implementación del sistema de tablas, relaciones y reglas de integridad demostró ser funcional durante las pruebas realizadas, donde se verificó el correcto comportamiento de las claves foráneas, la eliminación en cascada y la consistencia de los datos. Asimismo, las pruebas SQL confirmaron que la estructura diseñada responde adecuadamente a operaciones básicas de inserción, consulta y eliminación, asegurando estabilidad para su integración con el futuro sistema web.

Otro aspecto importante es que, gracias al proceso de análisis y revisión con el área de Sistemas, se pudieron aplicar mejoras que fortalecieron la organización del modelo y su utilidad práctica. Como resultado, la base de datos final no solo cumple los requisitos establecidos, sino que queda preparada para escalar y adaptarse a funcionalidades adicionales.

En conjunto, el proyecto permitió reforzar conocimientos en modelado, normalización, diseño físico e implementación en MySQL, logrando un producto final funcional y alineado con las necesidades institucionales.

VRecomendaciones

- **Documentar continuamente los cambios en el modelo de datos**

  Durante el proceso surgieron ajustes en claves foráneas, cardinalidades y reglas de integridad. Para futuras versiones del sistema se recomienda mantener una documentación actualizada de cada modificación, con el fin de evitar inconsistencias entre el modelo conceptual, lógico y físico.

- **Estandarizar las reglas de integridad desde las primeras etapas del diseño**

  Se recomienda definir de manera anticipada qué relaciones utilizarán ON DELETE CASCADE y cuáles no, para evitar resultados inesperados durante las pruebas. Esto permitirá construir una estructura más controlada, especialmente en tablas relacionadas como carpetas y archivos.

- **Realizar pruebas unitarias más amplias antes de incorporar datos reales**

  Aunque las pruebas SQL realizadas fueron suficientes para validar las relaciones principales, resulta conveniente ampliar los escenarios de prueba (cargas masivas, eliminaciones encadenadas, estructuras de carpetas con varios niveles) para asegurar que la base funcione correctamente en un entorno real.

- **Optimizar el uso de índices según el crecimiento del sistema**

  A medida que aumente el volumen de usuarios, carpetas y archivos, será necesario revisar los índices actuales e incorporar nuevos cuando sea necesario para mantener un buen rendimiento en las consultas más frecuentes.

- **Integrar un sistema de respaldo y recuperación**

  Una vez que la base de datos forme parte del sistema web institucional, se recomienda implementar políticas de respaldo periódico y restauración para proteger la información ante fallos o pérdidas accidentales.

- **Revisar la nomenclatura de tablas y campos antes de la integración final**

  Aunque los nombres utilizados son claros, resulta conveniente unificar criterios de estilo (por ejemplo, uso consistente de singular/plural o prefijos como tb_) para facilitar el mantenimiento por futuros desarrolladores.

- **Planificar la ampliación de módulos futuros**

  Dado que la estructura actual es flexible, se recomienda prever posibles módulos adicionales como registros de actividad, auditoría, historial de versiones o asignación de permisos por carpeta. Esto permitirá aprovechar la base existente sin realizar cambios radicales.

- **Validar nuevamente la estructura junto con el área usuaria**

  Antes de integrar la base de datos en el sistema definitivo, es recomendable realizar una revisión conjunta con el área de Sistemas para confirmar que todas las necesidades operativas estén cubiertas y evitar reprocesos posteriores.

VI Bibliografía Consultada

Gobierno del Perú – Municipalidad de Pativilca

Gobierno del Perú. (2024). Municipalidad Distrital de Pativilca.

https://www.gob.pe/munipativilca

XAMPP – Apache Friends

Apache Friends. (2024). XAMPP: Instalador de Apache, MariaDB, PHP y Perl.
https://www.apachefriends.org/es/index.html

phpMyAdmin – Herramienta de administración MySQL

phpMyAdmin. (2024). phpMyAdmin – Open Source MySQL Tool.

https://www.phpmyadmin.net

VII Galería

Ambiente de trabajo para el diseño de la base de datos

Este fue el espacio donde desarrollé la mayor parte del trabajo relacionado con el diseño de la base de datos. Aquí organizaba la información, revisaba los requisitos del sistema y analizaba cómo debían estructurarse las entidades principales. También era el lugar donde comparaba modelos, revisaba anotaciones y realizaba ajustes cada vez que surgía una nueva idea o detectaba algún error en el planteamiento inicial.

Aunque era un ambiente sencillo, me permitió trabajar con tranquilidad y concentrarme en el proceso de diseño. Desde este escritorio fui definiendo la estructura base del sistema, la cual luego se convertiría en el modelo lógico y físico implementado en MySQL.

Desarrollo del modelo entidad–relación y documentación

En esta fotografía se aprecia parte del proceso de elaboración del modelo entidad–relación de la base de datos. Desde este espacio fui analizando la información y traduciendo los requisitos funcionales en entidades, atributos y relaciones claras. Con apoyo del software Dia, preparaba los primeros diagramas y realizaba correcciones conforme validaba la coherencia del diseño.

Aquí también revisaba documentación técnica, comparaba alternativas de diseño para las tablas y ajustaba las relaciones cuando encontraba algún conflicto. Este paso fue fundamental para dejar definido un modelo ER sólido, que luego sirvió como base para construir la estructura física en MySQL.

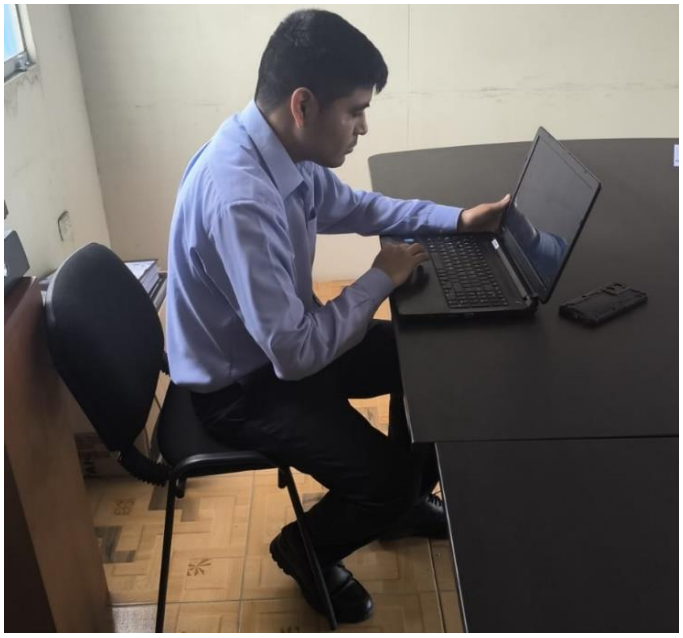


DIAGRAMA ER

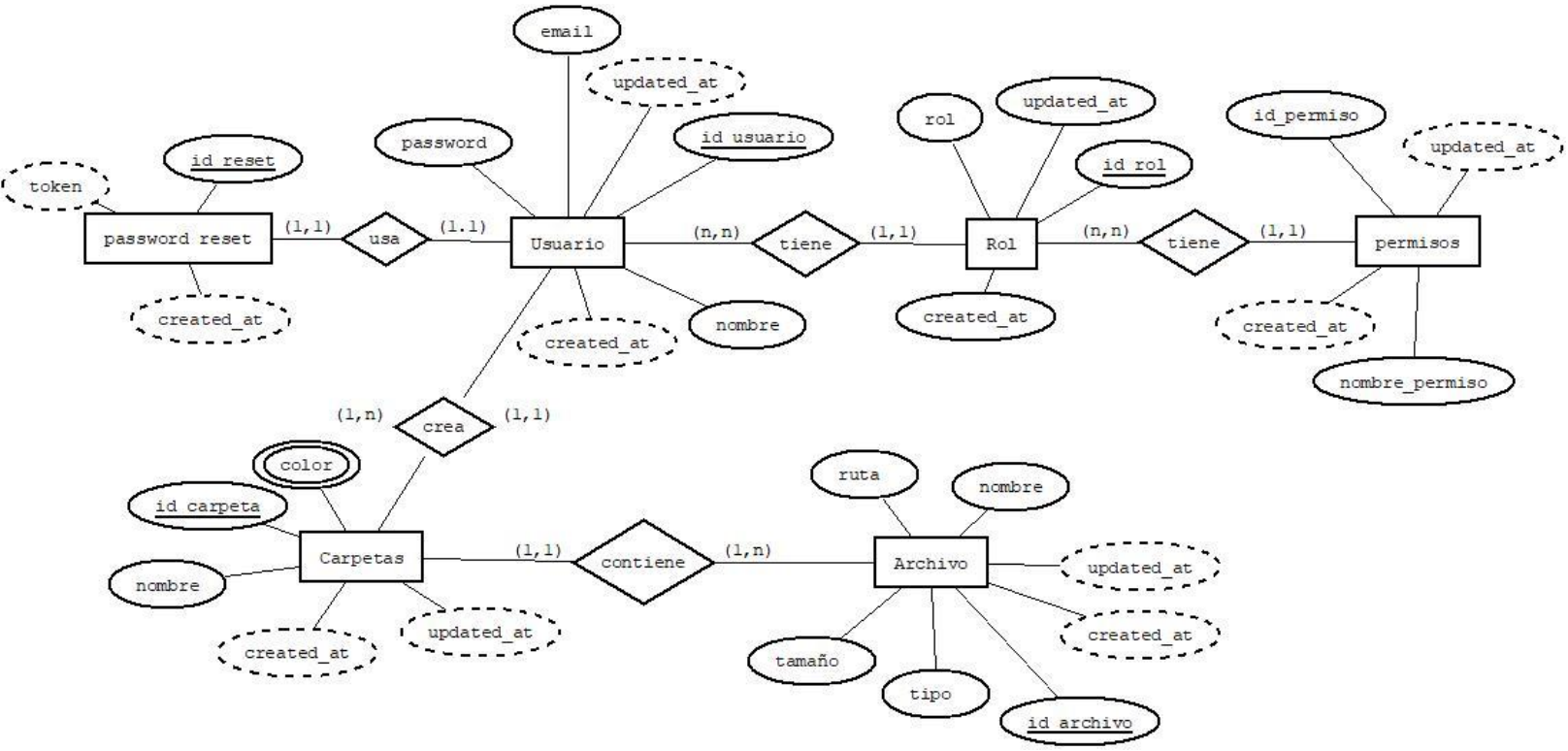

email
updated_at
password
id usuario
id reset
token
password reset
(1,1)
usa
(1.1)
Usuario
(n,n)
tiene
(1,1)
Rol
rol
updated_at
id rol
(n,n)
tiene
(1,1)
permisos
id_permiso
updated_at
created_at
nombre
created_at
created_at
created_at
nombre_permiso
(1,n)
crea
(1,1)
color
id carpeta
nombre
Carpetas
(1,1)
contiene
(1,n)
Archivo
ruta
nombre
updated_at
created_at
tamaño
tipo
id archivo
created_at
updated_at

DIAGRAMA MODELO LOGICO

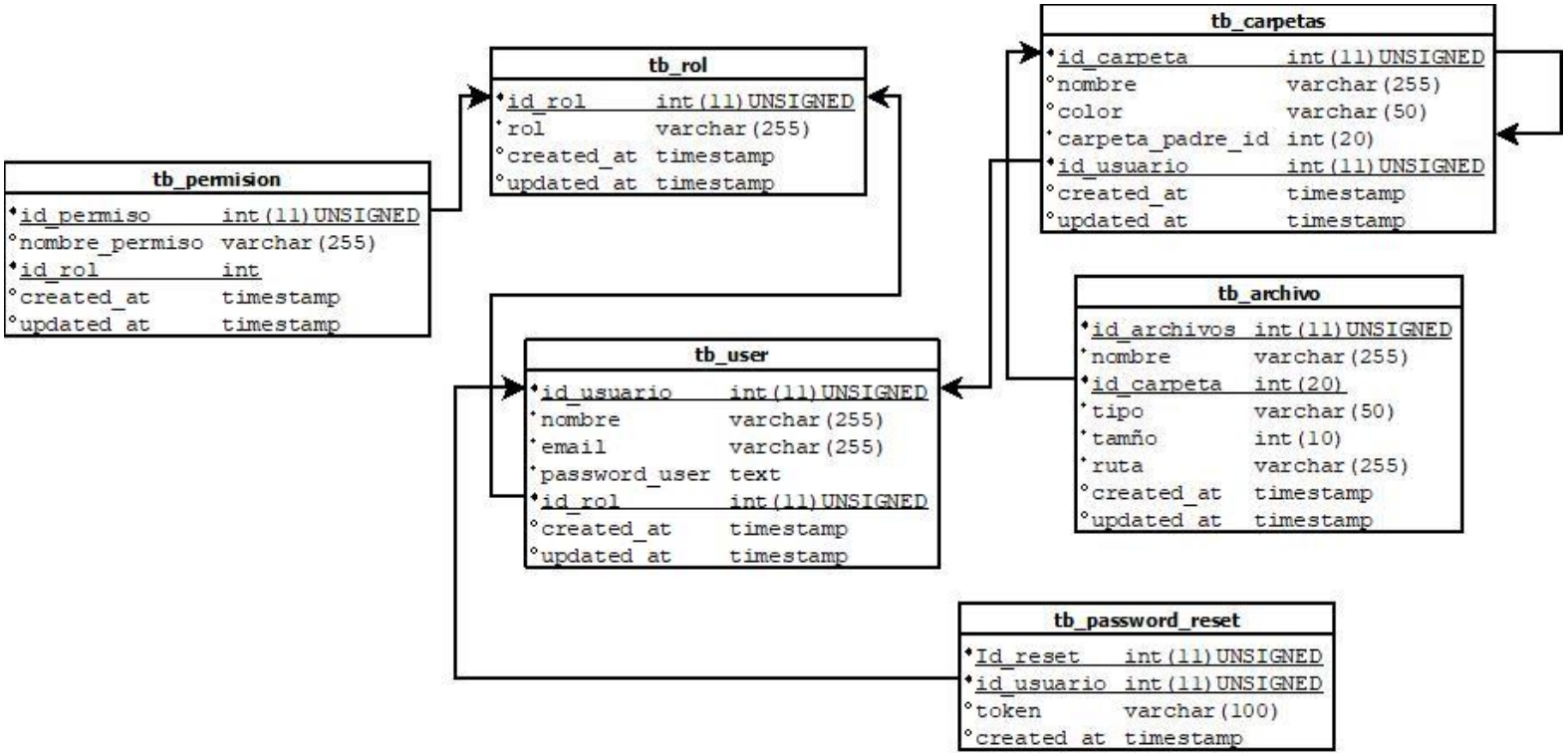
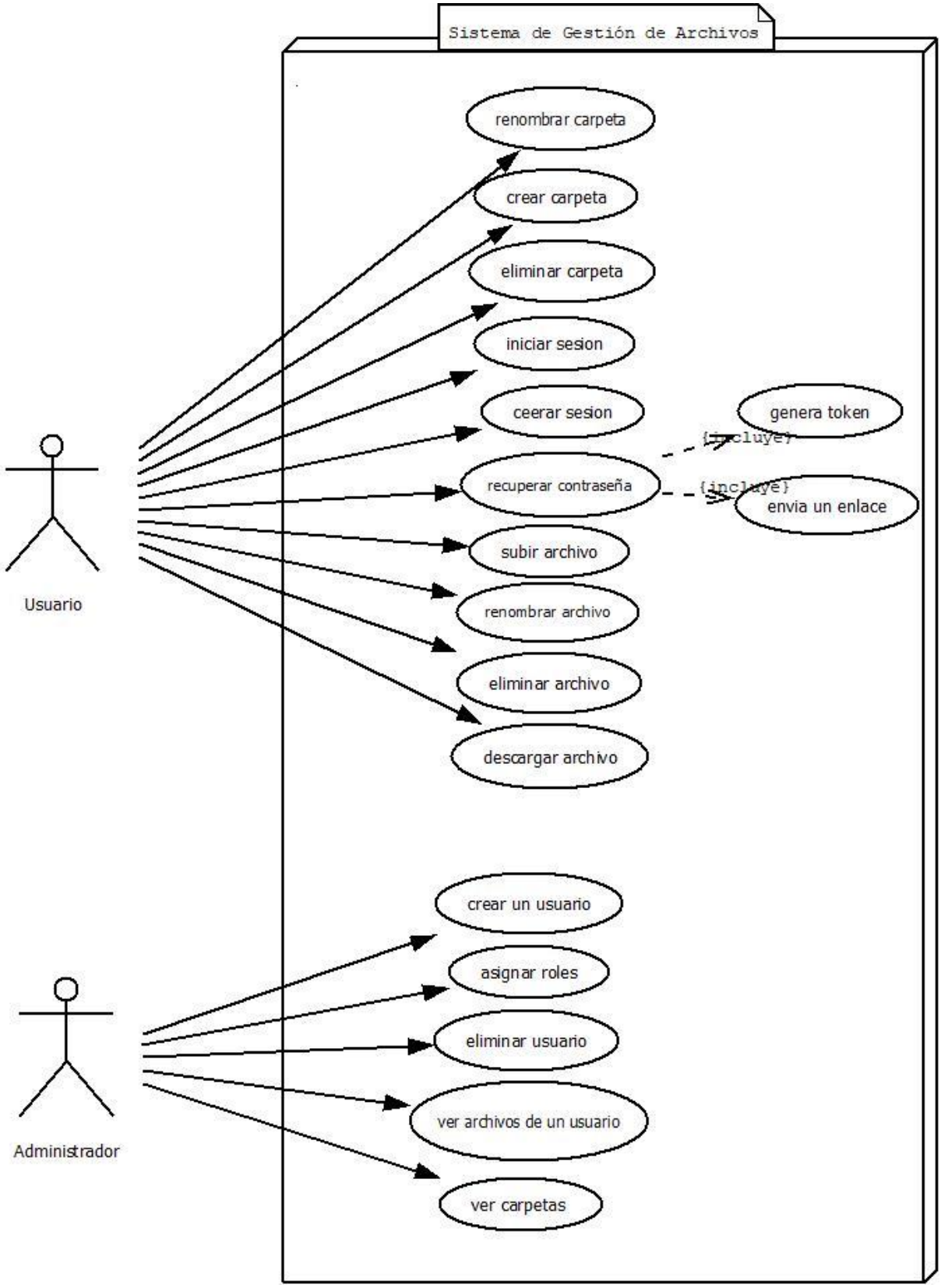


DIAGRAMA CASOS DE USO



Etapas de Creación, Configuración e Inserción de Datos en la Base de Datos para la Municipalidad

Creación de la base de datos

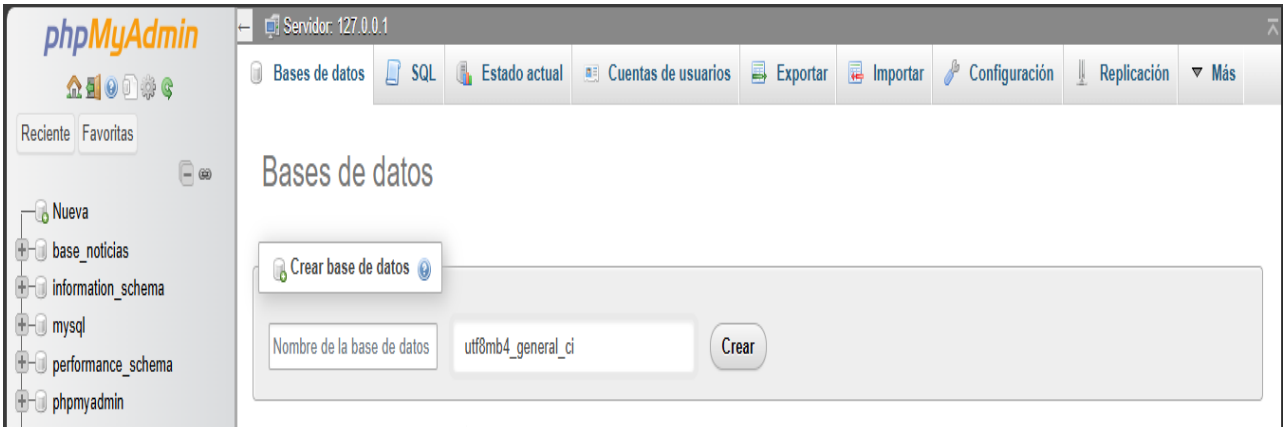
Desde XAMPP se inició el servicio MySQL y se accedió a phpMyAdmin para comenzar con la implementación.

En la opción Nueva base de datos se creó el esquema **sisgestiondearchivos**, seleccionando:

Motor InnoDB, debido a su soporte para claves foráneas y reglas de integridad.

Cotejamiento utf8mb4_unicode_ci, adecuado para nombres de usuarios, carpetas y archivos que puedan incluir tildes o caracteres especiales.

Esta configuración permitió garantizar compatibilidad y consistencia desde el inicio del proyecto.



Creación de las tablas para la base de datos

Con la base creada, se procedió a construir la estructura de tablas mediante sentencias SQL en phpMyAdmin.

El orden recomendado fue el siguiente:

Tabla tb_users:

Registra los datos principales de los usuarios, tales como:

- id_usuario (PK)
- nombre
- correo
- contraseña
- id_rol (FK)

Se definieron restricciones NOT NULL, AUTO_INCREMENT y una clave foránea que vincula cada usuario con un rol.

Estructura de tabla | Vista de relaciones

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id_usuario	int(11)		UNSIGNED	No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
2	nombre	varchar(255)	utf8mb4_unicode_ci		No	*Ninguna*			Cambiar Eliminar Más
3	email	varchar(255)	utf8mb4_unicode_ci		No	*Ninguna*			Cambiar Eliminar Más
4	password_user	text	utf8mb4_unicode_ci		No	*Ninguna*			Cambiar Eliminar Más
5	id_rol	int(11)		UNSIGNED	No	*Ninguna*			Cambiar Eliminar Más
6	created_at	timestamp			No	current_timestamp()			Cambiar Eliminar Más
7	updated_at	timestamp			Sí	current_timestamp()			Cambiar Eliminar Más

Tabla tb_roles

Almacena todos los roles disponibles en el sistema.

Campos principales:

- id_rol (PK, AUTO_INCREMENT)
- rol (VARCHAR)

Función:

Define el nivel de acceso de cada usuario en la plataforma.

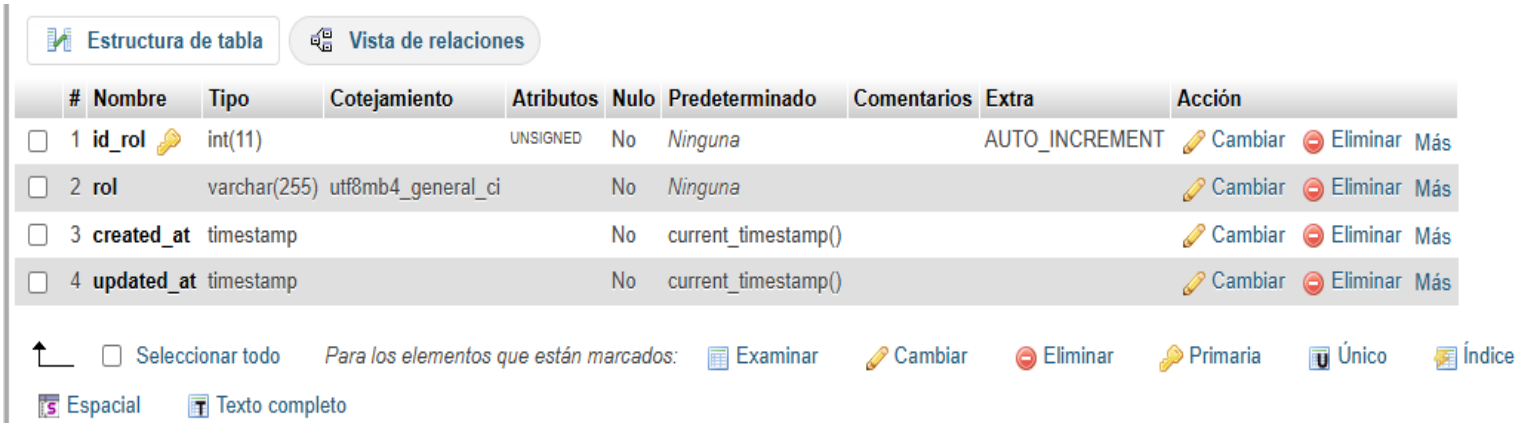


tabla tb_permision

Gestiona permisos asociados a los roles.

Campos:

- id_permiso (PK)
- nombre_permiso
- id_rol (FK)

Función:

Controla acciones que cada rol puede realizar.

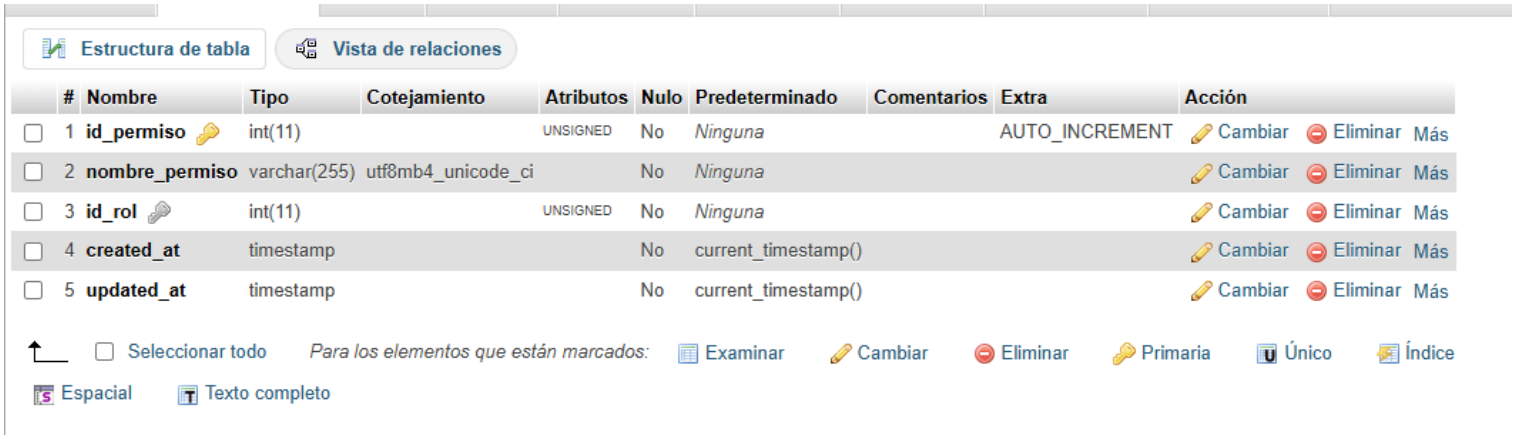


Tabla tb_carpetas

Almacena carpetas creadas por los usuarios.

Campos:

- id_carpeta (PK)
- nombre
- color
- carpeta_padre_id (opcional)
- id_usuario (FK)

Relación:

Cada carpeta pertenece a un usuario.

Estructura de tabla | Vista de relaciones

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id_carpeta	int(11)		UNSIGNED	No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
2	nombre	varchar(255)	utf8mb4_unicode_ci		No	*Ninguna*			Cambiar Eliminar Más
3	color	varchar(50)	utf8mb4_unicode_ci		Sí	*NULL*			Cambiar Eliminar Más
4	carpeta_padre_id	int(11)		UNSIGNED	Sí	*NULL*			Cambiar Eliminar Más
5	id_usuario	int(11)		UNSIGNED	Sí	*NULL*			Cambiar Eliminar Más
6	created_at	timestamp			No	current_timestamp()			Cambiar Eliminar Más
7	updated_at	timestamp			No	current_timestamp()			Cambiar Eliminar Más

Seleccionar todo *Para los elementos que están marcados:* Examinar Cambiar Eliminar Primaria Único Índice

Tabla tb_archivos

Registra los archivos que pertenecen a las carpetas.

Campos:

- id_archivo (PK)
- nombre_archivo
- tipo_archivo
- tamanio_archivo
- ruta_archivo
- id_carpeta (FK)

Relación:

Cada archivo está dentro de una carpeta específica.

Estructura de tabla | Vista de relaciones

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id_archivos	int(11)		UNSIGNED	No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
2	nombre	varchar(255)	utf8mb4_unicode_ci		No	*Ninguna*			Cambiar Eliminar Más
3	id_carpeta	int(11)		UNSIGNED	Sí	*NULL*			Cambiar Eliminar Más
4	tipo	varchar(50)	utf8mb4_unicode_ci		No	*Ninguna*			Cambiar Eliminar Más
5	tamaño	int(10)		UNSIGNED	No	*Ninguna*			Cambiar Eliminar Más
6	ruta	varchar(255)	utf8mb4_unicode_ci		No	*Ninguna*			Cambiar Eliminar Más
7	created_at	timestamp			No	current_timestamp()			Cambiar Eliminar Más
8	updated_at	timestamp			No	current_timestamp()			Cambiar Eliminar Más

Seleccionar todo *Para los elementos que están marcados:* Examinar Cambiar Eliminar Primaria Único Índice Espacial Texto completo

Tabla tb_password_reset

Maneja el proceso de recuperación de contraseña.

Campos:

- id_reset (PK)
- id_usuario (FK)
- token
- fecha_reset

Función:

Registrar los tokens de recuperación generados por los usuarios.

Estructura de tabla | Vista de relaciones

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id_reset	int(10)		UNSIGNED	No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
2	id_usuario	int(10)		UNSIGNED	No	*Ninguna*			Cambiar Eliminar Más
3	token	varchar(100)	utf8mb4_unicode_ci		No	*Ninguna*			Cambiar Eliminar Más
4	created_at	timestamp			No	current_timestamp()			Cambiar Eliminar Más

Seleccionar todo *Para los elementos que están marcados:* Examinar Cambiar Eliminar Primaria Único Índice Espacial Texto completo

Establecimiento de relaciones entre tablas

Las relaciones fueron implementadas mediante claves foráneas con reglas de integridad:

Usuario → Rol

`tb_users.id_rol` → `tb_roles.id_rol`

Rol → Permisos

`tb_permision.id_rol` → `tb_roles.id_rol`

Usuario → Carpetas

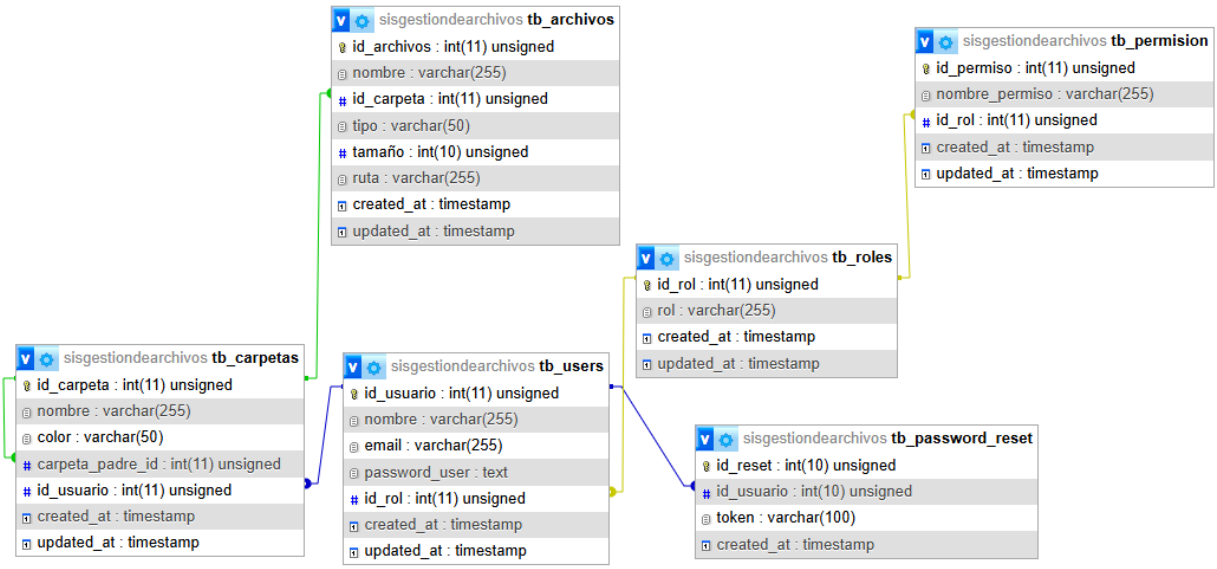
`tb_carpetas.id_usuario` → `tb_users.id_usuario`

Carpeta → Archivos

`tb_archivos.id_carpeta` → `tb_carpetas.id_carpeta`

Usuario → Password Reset

`tb_password_reset.id_usuario` → `tb_users.id_usuario`



CREACION DE CONSULTAS EN LAS BASE DE DATOS

Inserción de usuarios

Inserción de roles

Inserción de carpetas

Ejecutar la(s) consulta(s) SQL en la base de datos sisgestiondearchivos:

```
INSERT INTO tb_carpetas (nombre, color, id_usuario)
VALUES ('Documentos', 'azul', 1),
       ('Reportes', 'rojo', 1);
```

Inserción de archivos

Ejecutar la(s) consulta(s) SQL en la base de datos sisgestiondearchivos:

```
INSERT INTO tb_archivos (nombre_archivo, tipo_archivo, tamanio_archivo, ruta_archivo, id_carpeta)
VALUES ('reporte1.pdf', 'pdf', 2048, '/documentos/reporte1.pdf', 1);
```

Inserción de token de recuperación

Ejecutar la(s) consulta(s) SQL en la base de datos sisgestiondearchivos:

```
INSERT INTO tb_password_reset (id_usuario, token)
VALUES (1, 'token12345');
```

Exportación de la base de datos

Finalmente, se exportó la base de datos desde phpMyAdmin:

Exportar → SQL → Método rápido

Esto generó un archivo `.sql` con:

- estructura completa
- relaciones
- índices
- datos iniciales