# ÍNDICE

# Manual Técnico - Sistema de Gestión de Proyectos

## Introducción

El Sistema de Gestión de Proyectos es una aplicación web desarrollada con Flask que permite a los usuarios crear, compartir y colaborar en proyectos de desarrollo de software. El sistema proporciona funcionalidades similares a un sistema de control de versiones simplificado, permitiendo gestionar archivos, mantener un historial de cambios, y colaborar con otros usuarios.

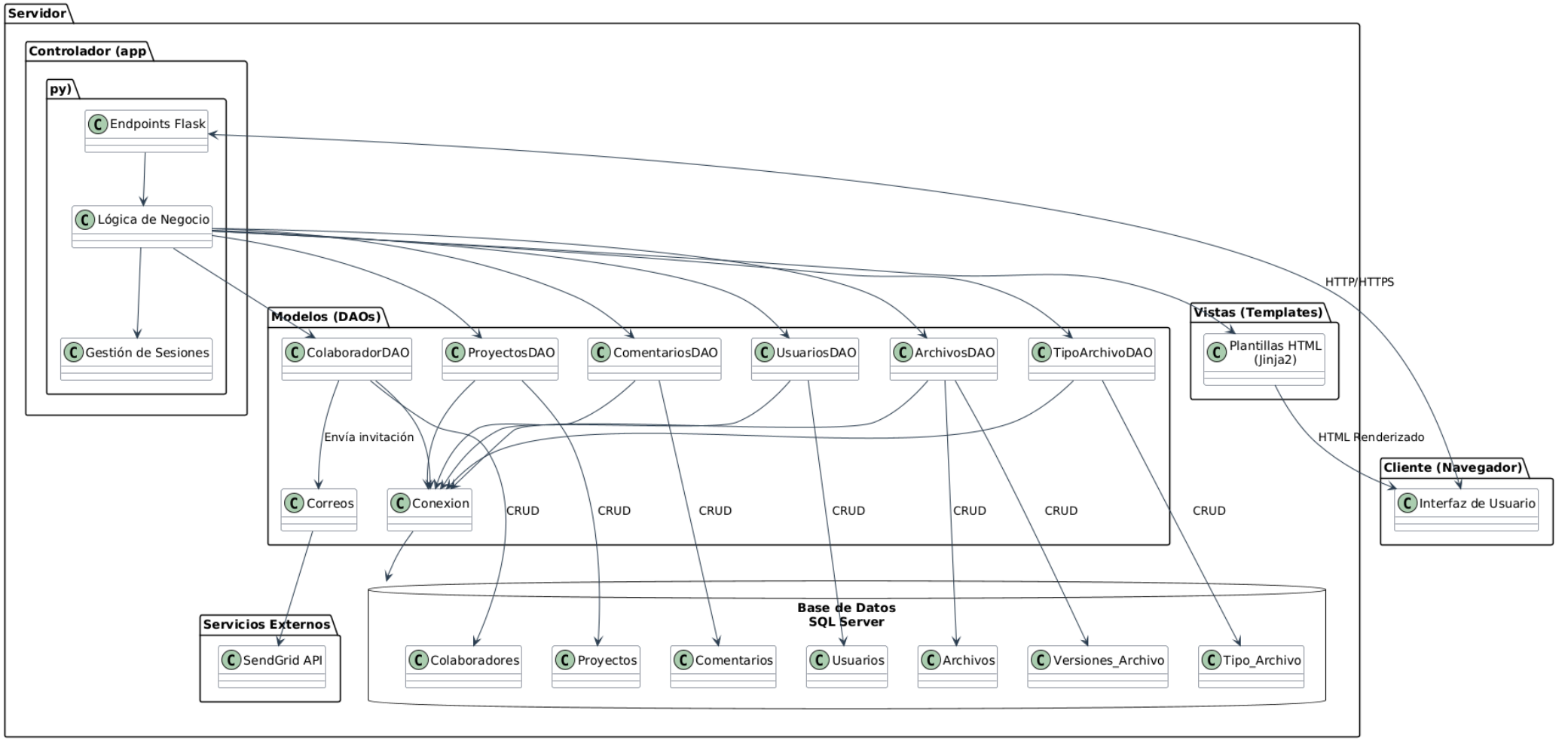
### Características Principales

- Gestión de usuarios (registro, inicio de sesión, perfil)
- Gestión de proyectos (creación, visualización, configuración)
- Gestión de archivos (creación, edición, versionado)
- Sistema de colaboración (invitaciones a proyectos privados)
- Sistema de comentarios (a nivel de proyecto y líneas de código)

## Arquitectura del Sistema

El sistema sigue una arquitectura Cliente-Servidor utilizando el patrón MVC (Modelo-Vista-Controlador):

1. **Modelo**: Componentes DAO (Data Access Object) que manejan la lógica de acceso a la base de datos.
2. **Vista**: Plantillas HTML con Jinja2 para renderizar la interfaz de usuario.
3. **Controlador**: La aplicación Flask (app.py) que maneja las rutas y la lógica de negocio.


Servidor
Controlador (app
py)
Endpoints Flask
Lógica de Negocio
Gestión de Sesiones
HTTP/HTTPS
Modelos (DAOs)
ColaboradorDAO
ProyectosDAO
ComentariosDAO
UsuariosDAO
ArchivosDAO
TipoArchivoDAO
Vistas (Templates)
Plantillas HTML (Jinja2)
HTML Renderizado
Cliente (Navegador)
Interfaz de Usuario
Envía invitación
Correos
Conexion
CRUD
CRUD
CRUD
CRUD
CRUD
CRUD
CRUD
Servicios Externos
SendGrid API
Base de Datos SQL Server
Colaboradores
Proyectos
Comentarios
Usuarios
Archivos
Versiones_Archivo
Tipo_Archivo

La comunicación entre componentes es la siguiente:

- El usuario interactúa con las vistas a través del navegador
- Las vistas envían solicitudes al controlador
- El controlador procesa las solicitudes y utiliza los modelos para interactuar con la base de datos
- El controlador envía la respuesta a las vistas para mostrar los resultados al usuario

# Requisitos del Sistema

### Software

- Python 3.12.4
- Microsoft SQL Server
- Navegador web compatible (Chrome, Firefox, Edge, etc.)

### Librerías Python

- Flask: Framework web
- pyodbc: Conector para SQL Server
- python-dotenv: Gestión de variables de entorno
- werkzeug.security: Herramientas de seguridad para contraseñas
- sendgrid: API para envío de correos electrónicos

### Variables de Entorno (.env)

DB_HOST=servidor_sql

DB_NAME=GestionProyectos

DB_USER=usuario_db

DB_PASSWORD=contraseña_db

DB_DRIVER={ODBC Driver 17 for SQL Server}

SENDGRID_API_KEY=clave_api_sendgrid

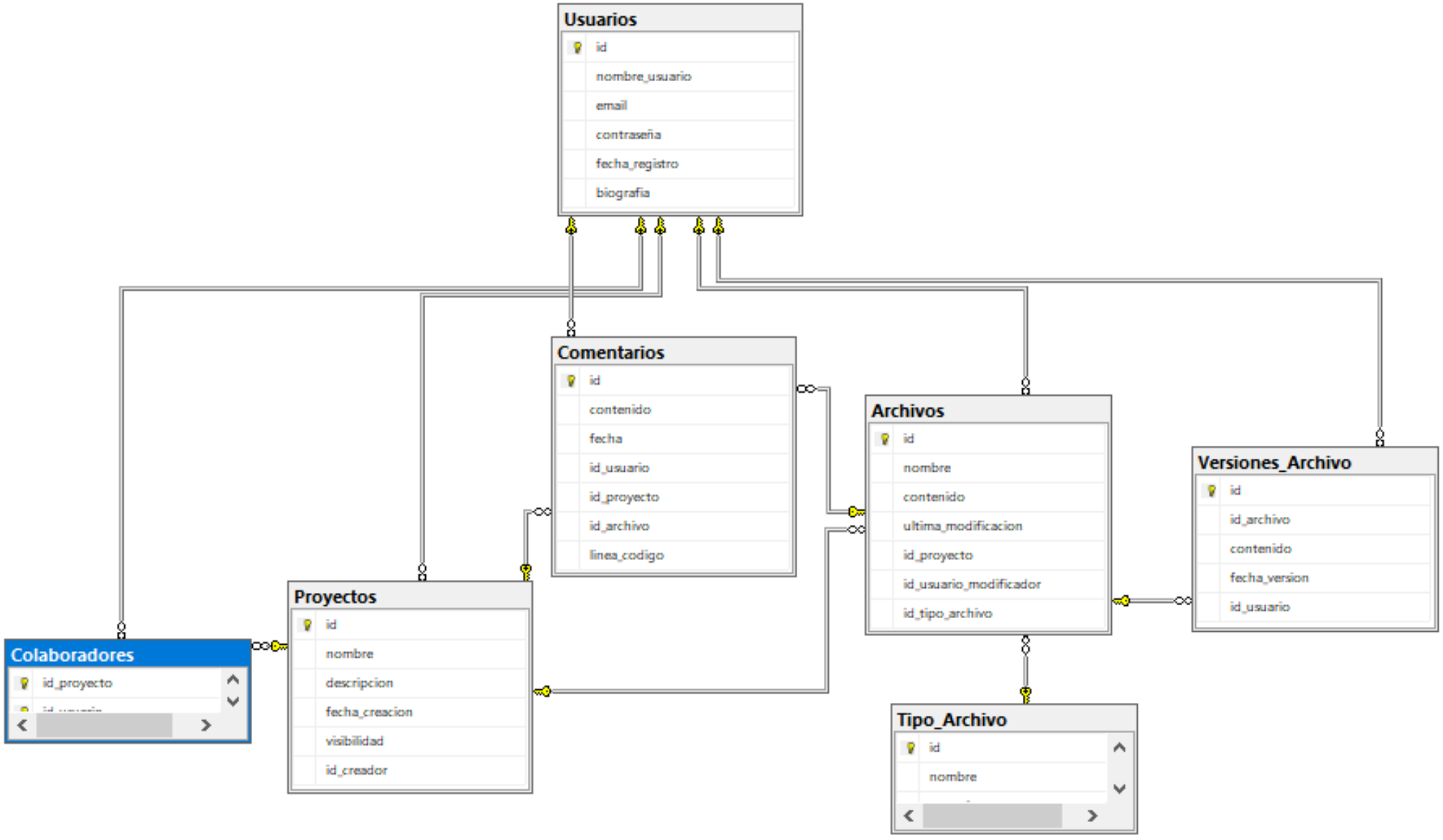
# Estructura del Proyecto

```
C:.
    .env                        # Variables de entorno
    .env.example                # Ejemplo de variables de entorno
    .gitignore                  # Archivos ignorados por git
    app.py                      # Aplicación principal (controlador)
    README.md                   # Documentación general
    Script GestionDeProyectos.sql # Script para crear la base de datos

    ---docs                     # Documentación adicional
        api.md
        Documentacion_Endpoints.pdf

    ---models                   # Modelos DAO para acceso a datos
    |   ArchivosDAO.py          # Gestión de archivos
    |   ColaboradorDAO.py       # Gestión de colaboradores
    |   ComentariosDAO.py       # Gestión de comentarios
    |   Conexion.py             # Gestión de conexión a base de datos
    |   Correos.py              # Envío de correos electrónicos
    |   ProyectosDAO.py         # Gestión de proyectos
    |   TipoArchivoDAO.py       # Gestión de tipos de archivo
    |   UsuariosDAO.py          # Gestión de usuarios

    ---templates                # Plantillas HTML (vistas)
        add_project.html        # Formulario para crear proyectos
        base.html               # Plantilla base con elementos comunes
        edit_project.html       # Formulario para editar proyectos
        file_version.html       # Vista de versiones de archivo
        home.html               # Página principal
        index.html              # Página de redirección
        login.html              # Página de inicio de sesión
        my_projects.html        # Lista de proyectos del usuario
        profile.html            # Perfil de usuario
        project.html            # Vista de un proyecto
        project_public.html     # Vista pública de un proyecto
        register.html           # Página de registro
```

# Base de Datos

El sistema utiliza Microsoft SQL Server para almacenar los datos. La estructura de la base de datos está compuesta por las siguientes tablas:

### Diagrama de Tablas


Usuarios
id
nombre_usuario
email
contraseña
fecha_registro
biografia
Comentarios
id
contenido
fecha
id_usuario
id_proyecto
id_archivo
linea_codigo
Archivos
id
nombre
contenido
ultima_modificacion
id_proyecto
id_usuario_modificador
id_tipo_archivo
Versiones_Archivo
id
id_archivo
contenido
fecha_version
id_usuario
Colaboradores
id_proyecto
Proyectos
id
nombre
descripcion
fecha_creacion
visibilidad
id_creador
Tipo_Archivo
id
nombre

## Tablas

1. **Usuarios**
    - id: Identificador único (PK, autoincremental)
    - nombre_usuario: Nombre de usuario (único)
    - email: Correo electrónico (único)
    - contraseña: Contraseña encriptada
    - fecha_registro: Fecha de creación de la cuenta
    - biografia: Descripción del usuario
2. **Proyectos**
    - id: Identificador único (PK, autoincremental)
    - nombre: Nombre del proyecto
    - descripcion: Descripción del proyecto
    - fecha_creacion: Fecha de creación
    - visibilidad: 'publico' o 'privado'
    - id_creador: ID del usuario creador (FK)
3. **Tipo_Archivo**
    - id: Identificador único (PK, autoincremental)
    - nombre: Nombre del tipo de archivo
    - extension: Extensión del archivo
4. **Archivos**
    - id: Identificador único (PK, autoincremental)
    - nombre: Nombre del archivo
    - contenido: Contenido del archivo
    - ultima_modificacion: Fecha de última modificación
    - id_proyecto: ID del proyecto (FK)
    - id_usuario_modificador: ID del último usuario que lo modificó (FK)
    - id_tipo_archivo: ID del tipo de archivo (FK)
5. **Versiones_Archivo**

- id: Identificador único (PK, autoincremental)
- id_archivo: ID del archivo (FK)
- contenido: Contenido de la versión
- fecha_version: Fecha de creación de la versión
- id_usuario: ID del usuario que creó la versión (FK)

6. **Colaboradores**

   - id_proyecto: ID del proyecto (PK, FK)
   - id_usuario: ID del usuario colaborador (PK, FK)

7. **Comentarios**

   - id: Identificador único (PK, autoincremental)
   - contenido: Contenido del comentario
   - fecha: Fecha de creación
   - id_usuario: ID del usuario que comentó (FK)
   - id_proyecto: ID del proyecto (FK)
   - id_archivo: ID del archivo (FK, opcional)
   - linea_codigo: Número de línea (opcional)

### Script SQL

El script `Script GestionDeProyectos.sql` proporciona la creación completa de la base de datos, incluyendo:

- Creación de tablas con restricciones de integridad
- Relaciones entre tablas (claves foráneas)
- Datos iniciales de prueba

# Modelos de Datos (DAOs)

El sistema utiliza el patrón DAO (Data Access Object) para separar la lógica de acceso a datos de la lógica de negocio. Cada clase DAO se encarga de gestionar las operaciones CRUD para una entidad específica:

### UsuariosDAO.py

Gestiona la autenticación y operaciones de usuario:

- create_user: Crea un nuevo usuario
- authenticate_user: Autentica un usuario por email y contraseña
- get_user_by_id: Obtiene un usuario por ID
- get_user_by_email: Obtiene un usuario por email
- update_user: Actualiza información del usuario
- update_password: Actualiza la contraseña del usuario

### ProyectosDAO.py

Gestiona operaciones relacionadas con proyectos:

- create_project: Crea un nuevo proyecto
- get_project_by_id: Obtiene un proyecto por ID
- get_projects_by_user: Obtiene proyectos creados por un usuario
- get_projects_collaborative: Obtiene proyectos donde el usuario es colaborador
- get_projects_publics_not_myself: Obtiene proyectos públicos de otros usuarios
- get_project_owner: Obtiene el propietario de un proyecto
- update_project: Actualiza información del proyecto
- delete_project: Elimina un proyecto

### ArchivosDAO.py

Gestiona operaciones de archivos y versiones:

- create_file: Crea un nuevo archivo
- get_file_by_id: Obtiene un archivo por ID
- get_files_by_project_id: Obtiene archivos de un proyecto
- update_content: Actualiza el contenido de un archivo
- delete_file: Elimina un archivo
- get_file_versions: Obtiene versiones de un archivo
- get_version_by_id: Obtiene una versión específica

### ComentariosDAO.py

Gestiona operaciones relacionadas con comentarios:

- create_comment: Crea un nuevo comentario
- get_comment_by_project_id: Obtiene comentarios de un proyecto

### ColaboradorDAO.py

Gestiona la colaboración en proyectos:

- add_colaborator_gmail: Añade un colaborador por email
- get_all_colaboradores_by_project: Obtiene colaboradores de un proyecto
- get_content_project_by_colaborador: Verifica si un usuario es colaborador
- remove_colaborador: Elimina un colaborador

### TipoArchivoDAO.py

Gestiona los tipos de archivo permitidos:

- get_all_file_types: Obtiene todos los tipos de archivo

### Conexion.py

Gestiona la conexión a la base de datos:

- get_conexion: Obtiene una conexión a la base de datos

### Correos.py

Gestiona el envío de correos electrónicos utilizando SendGrid:

- send_invitation_email: Envía invitación a colaborar en un proyecto

# Endpoints y Funcionalidades

### Autenticación y Usuarios

#### `/login` (GET, POST)

- **GET**: Muestra formulario de inicio de sesión
- **POST**: Autentica al usuario y crea sesión

**`/register` (GET, POST)**

- **GET**: Muestra formulario de registro
- **POST**: Crea un nuevo usuario

**`/logout` (GET)**

- Cierra la sesión del usuario

**`/home/profile` (GET, POST)**

- **GET**: Muestra perfil del usuario
- **POST**: Actualiza información del perfil

**`/home/profile/change_password` (POST)**

- Actualiza la contraseña del usuario

## Gestión de Proyectos

**`/home` (GET)**

- Muestra página principal con proyectos del usuario, colaborativos y públicos

**`/home/my_projects` (GET)**

- Muestra proyectos creados por el usuario

**`/home/my_projects/add_project` (GET, POST)**

- **GET**: Muestra formulario para crear proyecto
- **POST**: Crea un nuevo proyecto

**`/home/my_projects/edit_project/<project_id>` (GET, POST)**

- **GET**: Muestra formulario para editar proyecto
- **POST**: Actualiza información del proyecto

**`/home/delete_projects/<project_id>` (GET)**

- Elimina un proyecto

## Visualización de Proyectos

**`/home/view_project/<project_id>` (GET)**

- Muestra detalles completos de un proyecto propio o colaborativo

**`/home/view_project_public/<project_id>` (GET)**

- Muestra detalles de un proyecto público

## Gestión de Archivos

**`/home/view_project/new_file/<project_id>` (POST)**

- Crea un nuevo archivo en el proyecto

**`/home/view_project/edit_file/<file_id>` (POST)**

- Actualiza el contenido de un archivo

**`/home/view_project/delete_file/<file_id>` (POST)**

- Elimina un archivo

**`/home/view_project/version_file/<file_id>` (GET)**

- Muestra historial de versiones de un archivo

## Descargas

**`/home/view_project/download_file/<file_id>` (GET)**

- Descarga un archivo

**`/home/view_project/download_version/<version_id>` (GET)**

- Descarga una versión específica de un archivo

**`/home/view_project/download_project/<project_id>` (GET)**

- Descarga todos los archivos del proyecto como ZIP

**`/home/view_project_public/download_file/<file_id>` (GET)**

- Descarga un archivo de un proyecto público

### Colaboradores

**`/home/view_project/add_colaborator/<project_id>` (POST)**

- Añade un colaborador al proyecto

**`/home/view_project/delete_colaborator/<colaborador_id>` (POST)**

- Elimina un colaborador del proyecto

### Comentarios

**`/home/view_project/add_comment` (POST)**

- Añade un comentario a un proyecto o archivo

## Seguridad

El sistema implementa varias medidas de seguridad:

### Protección de Contraseñas

Se utiliza `werkzeug.security` para el hashing de contraseñas mediante el algoritmo Scrypt:

- `generate_password_hash`: Genera hash seguro de contraseñas
- `check_password_hash`: Verifica contraseñas sin almacenarlas en texto plano

### Control de Acceso

- Todas las rutas verifican la existencia de sesión activa
- Redirección a login para usuarios no autenticados
- Verificación de permisos para acciones en proyectos

### Variables de Entorno

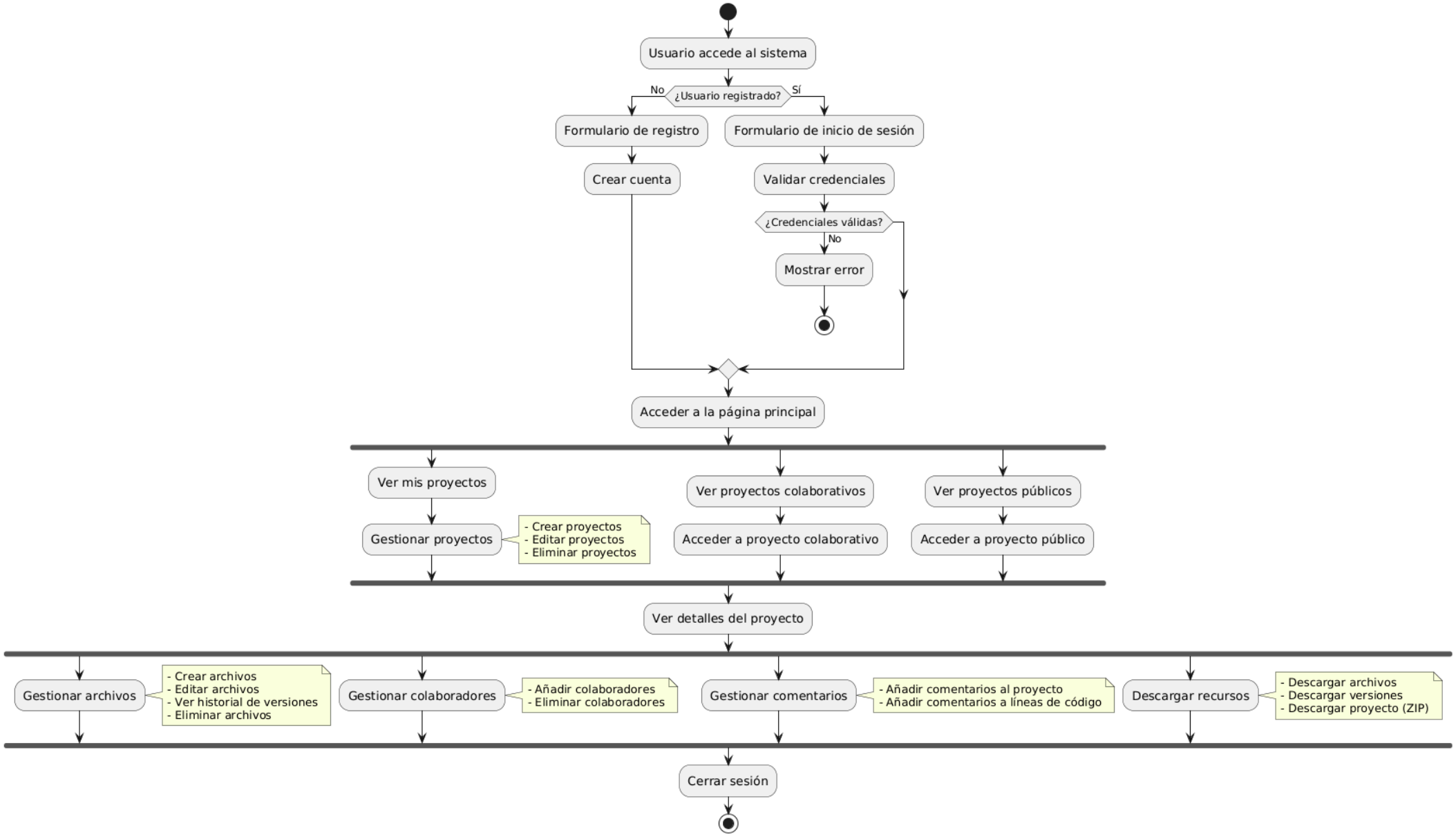
- Información sensible (credenciales de BD, API keys) almacenada en archivo `.env`
- No se incluyen archivos sensibles en el control de versiones

### Sesiones

- Uso de sesiones Flask para mantener el estado de autenticación
- Clave secreta para firma de cookies de sesión

## Flujo de Trabajo

### Diagrama de Flujo:


Usuario accede al sistema
¿Usuario registrado?
No
Sí
Formulario de registro
Formulario de inicio de sesión
Crear cuenta
Validar credenciales
¿Credenciales válidas?
No
Mostrar error
Acceder a la página principal
Ver mis proyectos
Gestionar proyectos
- Crear proyectos
- Editar proyectos
- Eliminar proyectos
Ver proyectos colaborativos
Acceder a proyecto colaborativo
Ver proyectos públicos
Acceder a proyecto público
Ver detalles del proyecto
Gestionar archivos
- Crear archivos
- Editar archivos
- Ver historial de versiones
- Eliminar archivos
Gestionar colaboradores
- Añadir colaboradores
- Eliminar colaboradores
Gestionar comentarios
- Añadir comentarios al proyecto
- Añadir comentarios a líneas de código
Descargar recursos
- Descargar archivos
- Descargar versiones
- Descargar proyecto (ZIP)
Cerrar sesión

### Registro e Inicio de Sesión

1. El usuario accede a `/register` y completa el formulario
2. El sistema verifica que no exista otro usuario con el mismo email
3. Se guarda el usuario con contraseña encriptada
4. El usuario inicia sesión en `/login`
5. Se crea una sesión y se redirige a `/home`

### Creación de Proyectos

1. El usuario accede a `/home/my_projects/add_project`
2. Completa el formulario con nombre, descripción y visibilidad
3. El sistema crea el proyecto y asocia al usuario como creador
4. Se redirige a `/home` mostrando el nuevo proyecto

### Gestión de Archivos

1. En la vista de proyecto, el usuario crea un archivo
2. Selecciona el tipo de archivo y asigna un nombre
3. Edita el contenido en el editor integrado
4. Al guardar, se crea una nueva versión y se actualiza el archivo

### Colaboración

1. El creador del proyecto invita colaboradores mediante email
2. El sistema verifica que el usuario exista
3. Se envía notificación por correo
4. El colaborador accede al proyecto desde su página principal

### Comentarios

1. Los usuarios pueden comentar a nivel de proyecto
2. También pueden comentar líneas específicas de código
3. Los comentarios incluyen autor y fecha

# Guía de Instalación

## Requisitos Previos

1. Python 3.12.4 instalado
2. Microsoft SQL Server instalado y configurado
3. Driver ODBC para SQL Server instalado

## Pasos de Instalación

1. **Clonar el repositorio**:

   git clone <url-repositorio>

   cd <directorio-proyecto>

2. **Crear entorno virtual**:
   python -m venv venv

   # Activar entorno virtual

   # Windows:

   venv\Scripts\activate

   # Linux/macOS:

   source venv/bin/activate

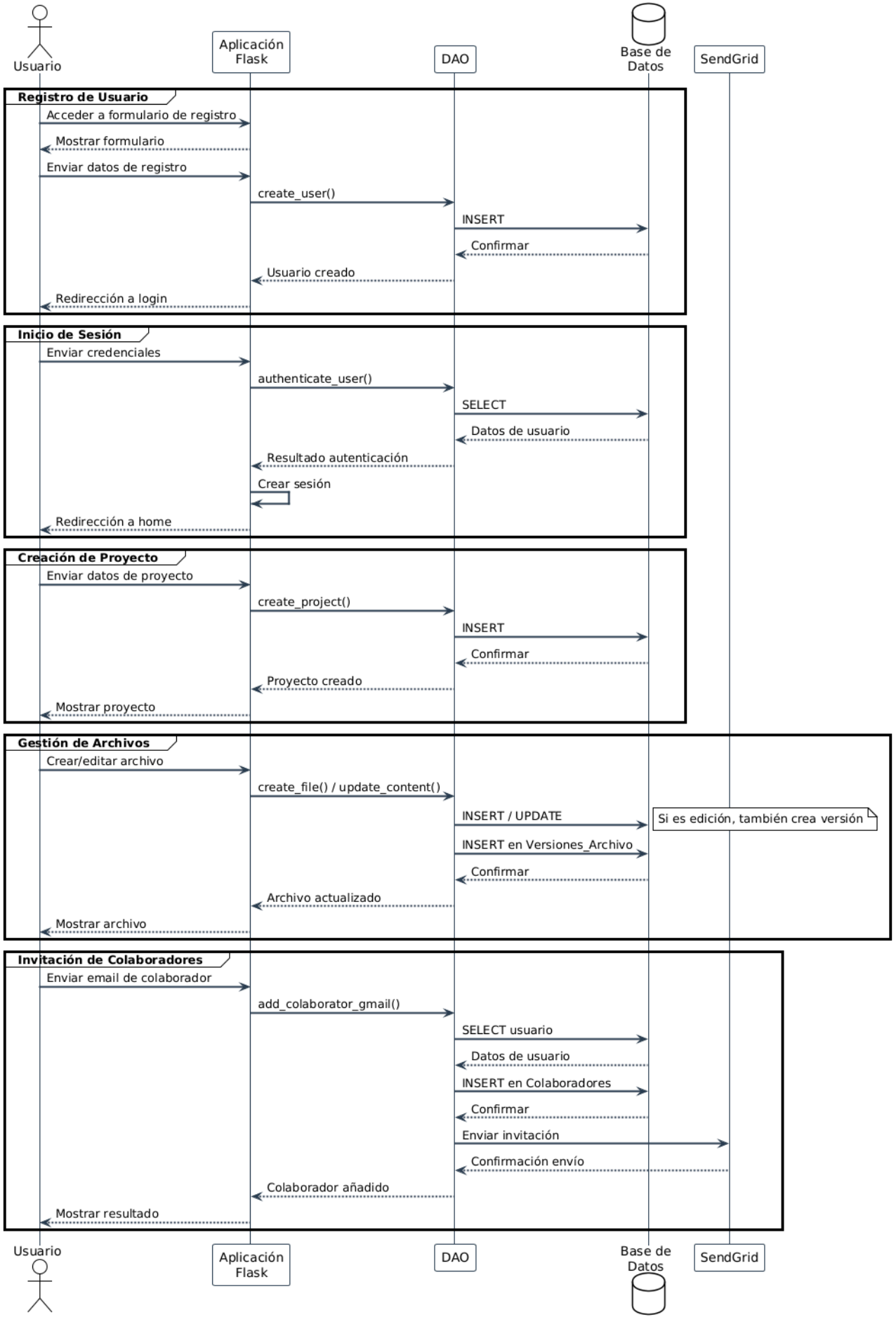
3. **Instalar dependencias**:
   pip install flask pyodbc python-dotenv sendgrid
4. **Configurar variables de entorno**:
   - Copiar `.env.example` a `.env`
   - Editar `.env` con la configuración correcta
5. **Crear base de datos**:

   - Ejecutar el script `Script GestionDeProyectos.sql` en SQL Server
6. **Ejecutar la aplicación**:
   python app.py

7. **Acceder a la aplicación**:

    - Abrir navegador en http://localhost:5000

# Diagrama de Secuencia


Usuario
Aplicación Flask
DAO
Base de Datos
SendGrid
Registro de Usuario
Acceder a formulario de registro
Mostrar formulario
Enviar datos de registro
create_user()
INSERT
Confirmar
Usuario creado
Redirección a login
Inicio de Sesión
Enviar credenciales
authenticate_user()
SELECT
Datos de usuario
Resultado autenticación
Crear sesión
Redirección a home
Creación de Proyecto
Enviar datos de proyecto
create_project()
INSERT
Confirmar
Proyecto creado
Mostrar proyecto
Gestión de Archivos
Crear/editar archivo
create_file() / update_content()
INSERT / UPDATE
Si es edición, también crea versión
INSERT en Versiones_Archivo
Confirmar
Archivo actualizado
Mostrar archivo
Invitación de Colaboradores
Enviar email de colaborador
add_colaborator_gmail()
SELECT usuario
Datos de usuario
INSERT en Colaboradores
Confirmar
Enviar invitación
Confirmación envío
Colaborador añadido
Mostrar resultado
Usuario
Aplicación Flask
DAO
Base de Datos
SendGrid