**Manual paso a paso**

|  |  |
| --- | --- |
| Profile for Universidad Nacional Politécnica UNP, CUR-Rivas |  |

Integrantes:

* Carlos Eliel Ramirez Martinez

Maestro:

* Danilo rodriguez

Índice

[Módulo de Adjunto 4](#_Toc206107265)

[**🧩 1. Estructura General del Módulo** 4](#_Toc206107266)

[**📍 2. Endpoints Disponibles** 5](#_Toc206107267)

[**🧾 3. Flujo de Gestión de Archivos** 5](#_Toc206107268)

[**➕ Subir adjunto (/adjuntos/nuevo)** 5](#_Toc206107269)

[**🗂️ Listar adjuntos (/adjuntos/)** 6](#_Toc206107270)

[**📥 Descargar adjunto (/adjuntos/descargar/<id>)** 6](#_Toc206107271)

[**🗑️ Eliminar adjunto (/adjuntos/eliminar/<id>)** 6](#_Toc206107272)

[**⚙️ 4. Funciones CRUD Internas – app/crud/adjunto\_crud.py** 7](#_Toc206107273)

[**🛡️ 5. Recomendaciones de Mejora** 7](#_Toc206107274)

[Módulo de Estado 9](#_Toc206107275)

[**🧩 1. Estructura General del Módulo** 9](#_Toc206107276)

[**📍 2. Endpoints Disponibles** 9](#_Toc206107277)

[**🧾 3. Flujo de Gestión de Estados** 9](#_Toc206107278)

[**🧍 Ver estados (/estados/)** 9](#_Toc206107279)

[**➕ Crear nuevo estado (/estados/nuevo)** 9](#_Toc206107280)

[**✏️ Editar estado (/estados/editar/<idestado>)** 10](#_Toc206107281)

[**🗑️ Eliminar estado (/estados/eliminar/<idestado>)** 10](#_Toc206107282)

[**📥 API REST (/estados/api)** 10](#_Toc206107283)

[**⚙️ 4. Funciones CRUD Internas – app/crud/estado\_crud.py** 11](#_Toc206107284)

[**🛡️ 5. Recomendaciones y Mejoras** 11](#_Toc206107285)

[Módulo de Historial 12](#_Toc206107286)

[**🧩 1. Estructura General del Módulo** 12](#_Toc206107287)

[**📍 2. Endpoints Disponibles** 12](#_Toc206107288)

[**🧾 3. Flujo de Gestión de Historial** 12](#_Toc206107289)

[**🧍 Ver historial (/historial/)** 12](#_Toc206107290)

[**➕ Crear historial (/historial/nuevo)** 13](#_Toc206107291)

[**✏️ Editar historial (/historial/editar/<idhistorial>)** 13](#_Toc206107292)

[**🗑️ Eliminar historial (/historial/eliminar/<idhistorial>)** 14](#_Toc206107293)

[**📥 API REST (/historial/api)** 14](#_Toc206107294)

[**⚙️ 4. Funciones CRUD Internas – app/crud/historial\_crud.py** 15](#_Toc206107295)

[**🛡️ 5. Recomendaciones y Mejoras** 15](#_Toc206107296)

[Módulo de Notificaciones 16](#_Toc206107297)

[**🧩 1. Estructura General del Módulo** 16](#_Toc206107298)

[**📍 2. Endpoints Disponibles** 16](#_Toc206107299)

[**🧾 3. Flujo de Gestión de Notificaciones** 16](#_Toc206107300)

[**🧍 Ver notificaciones (/notificaciones/)** 16](#_Toc206107301)

[**➕ Crear notificación (/notificaciones/nuevo)** 17](#_Toc206107302)

[**✏️ Editar notificación (/notificaciones/editar/<idnotificacion>)** 18](#_Toc206107303)

[**🗑️ Eliminar notificación (/notificaciones/eliminar/<idnotificacion>)** 18](#_Toc206107304)

[**📥 API REST (/notificaciones/api)** 19](#_Toc206107305)

[**⚙️ 4. Funciones CRUD Internas – app/crud/notificacion\_crud.py** 19](#_Toc206107306)

[**🛡️ 5. Recomendaciones y Mejoras** 20](#_Toc206107307)

[Módulo de Prioridades 21](#_Toc206107308)

[**🧩 1. Estructura General del Módulo** 21](#_Toc206107309)

[**📍 2. Endpoints Disponibles** 21](#_Toc206107310)

[**🧾 3. Flujo de Gestión de Prioridades** 21](#_Toc206107311)

[**🧍 Ver prioridades (/prioridades/)** 21](#_Toc206107312)

[**➕ Crear prioridad (/prioridades/nuevo)** 21](#_Toc206107313)

[**✏️ Editar prioridad (/prioridades/editar/<idprioridad>)** 22](#_Toc206107314)

[**🗑️ Eliminar prioridad (/prioridades/eliminar/<idprioridad>)** 23](#_Toc206107315)

[**📥 API REST (/prioridades/api)** 23](#_Toc206107316)

[**⚙️ 4. Funciones CRUD Internas** 23](#_Toc206107317)

[**🛡️ 5. Recomendaciones y Mejora** 24](#_Toc206107318)

[Módulo de Roll 25](#_Toc206107319)

[**🧩 1. Estructura General del Módulo** 25](#_Toc206107320)

[**📍 2. Endpoints Disponibles** 25](#_Toc206107321)

[**🧾 3. Flujo de Gestión de Roles** 25](#_Toc206107322)

[**🧍 Ver roles (/roles/)** 25](#_Toc206107323)

[**➕ Crear rol (/roles/nuevo)** 26](#_Toc206107324)

[**🗑️ Eliminar rol (/roles/eliminar/<idroll>)** 26](#_Toc206107325)

[**📥 API REST (/roles/api)** 27](#_Toc206107326)

[**⚙️ 4. Funciones CRUD Internas – app/crud/roll\_crud.py** 27](#_Toc206107327)

[**🛡️ 5. Recomendaciones y Mejora** 27](#_Toc206107328)

[Módulo de Tipo\_Tarea 28](#_Toc206107329)

[**🧩 1. Estructura General del Módulo** 28](#_Toc206107330)

[**📍 2. Endpoints Disponibles** 28](#_Toc206107331)

[**\🧾 3. Flujo de Gestión de Tipos de Tarea** 28](#_Toc206107332)

[**🧍 Ver tipos de tarea (/tipos-tarea/)** 28](#_Toc206107333)

[**➕ Crear tipo de tarea (/tipos-tarea/nuevo)** 29](#_Toc206107334)

[**✏️ Editar tipo de tarea (/tipos-tarea/editar/<idtipo\_tarea>)** 29](#_Toc206107335)

[**🗑️ Eliminar tipo de tarea (/tipos-tarea/eliminar/<idtipo\_tarea>)** 30](#_Toc206107336)

[**📥 API REST (/tipos-tarea/api)** 30](#_Toc206107337)

[**⚙️ 4. Funciones CRUD Internas – app/crud/tipo\_tarea\_crud.py** 31](#_Toc206107338)

[**🛡️ 5. Recomendaciones y Mejora** 31](#_Toc206107339)

[Modulo Tarea 32](#_Toc206107340)

[**🧩 1. Estructura General del Módulo** 32](#_Toc206107341)

[**📍 2. Endpoints Disponibles** 32](#_Toc206107342)

[**🧠 3. Flujo Interno de Funcionalidades** 32](#_Toc206107343)

[**✅ Crear una Tarea** 32](#_Toc206107344)

[**🗑️ Eliminar una Tarea** 33](#_Toc206107345)

[**📥 API REST: Listado de Tareas** 34](#_Toc206107346)

[**🧾 4. Vista Principal /tareas/** 34](#_Toc206107347)

[**🔄 Agrupación dinámica:** 35](#_Toc206107348)

[**⚙️ 5. CRUD Interno (tarea\_crud.py)** 35](#_Toc206107349)

[**🔗 6. Relaciones y Dependencias** 36](#_Toc206107350)

[**🛡️ 7. Recomendaciones de Mejora** 36](#_Toc206107351)

[Modulo Usuario 37](#_Toc206107352)

[**🧩 1. Estructura General del Módulo** 37](#_Toc206107353)

[**📍 2. Endpoints Disponibles** 37](#_Toc206107354)

[**🔐 3. Flujo de Autenticación** 37](#_Toc206107355)

[**✅ Inicio de sesión (/usuarios/login)** 37](#_Toc206107356)

[**🚪 Cierre de sesión (/usuarios/logout)** 38](#_Toc206107357)

[**🧾 4. Gestión de Usuarios** 38](#_Toc206107358)

[**🧍 Ver usuarios (GET /usuarios/)** 38](#_Toc206107359)

[**➕ Crear nuevo usuario (POST /usuarios/nuevo)** 39](#_Toc206107360)

[**✏️ Editar usuario (POST /usuarios/editar/<idusuario>)** 39](#_Toc206107361)

[**🗑️ Eliminar usuario (POST /usuarios/eliminar/<idusuario>)** 39](#_Toc206107362)

[**📥 5. API REST: Lista de Usuarios** 40](#_Toc206107363)

[**⚙️ 6. Funciones CRUD Internas - app/crud/usuario\_crud.py** 40](#_Toc206107364)

[**🔍 obtener\_usuario\_por\_id(idusuario)** 41](#_Toc206107365)

[**📧 obtener\_usuario\_por\_correo(correo)** 41](#_Toc206107366)

[**📋 listar\_usuarios()** 41](#_Toc206107367)

[**✏️ actualizar\_usuario(idusuario, datos)** 41](#_Toc206107368)

[**🗑️ eliminar\_usuario(idusuario)** 42](#_Toc206107369)

[**🛡️ 7. Recomendaciones de Mejora** 42](#_Toc206107370)

[**Modulo Comentario** 43](#_Toc206107371)

[**🧩 1. Estructura General del Módulo** 43](#_Toc206107372)

[**📍 2. Endpoints Disponibles** 43](#_Toc206107373)

[**🧾 3. Vista de Comentarios** 43](#_Toc206107374)

[**🔍 Filtrado y Visualización (GET /comentarios/)** 43](#_Toc206107375)

[**➕ 4. Crear Comentario (POST /comentarios/)** 44](#_Toc206107376)

[**🧠 Flujo:** 44](#_Toc206107377)

[**📝 Datos que se registran:** 45](#_Toc206107378)

[**✏️ 5. Editar Comentario (POST /comentarios/editar/<id>)** 45](#_Toc206107379)

[**🗑️ 6. Eliminar Comentario (POST /comentarios/eliminar/<id>)** 45](#_Toc206107380)

[**Pasos:** 45](#_Toc206107381)

[**📥 7. API REST: Listado de Comentarios** 46](#_Toc206107382)

[**⚙️ 8. Funciones CRUD - app/crud/comentario\_crud.py** 46](#_Toc206107383)

[**🔧 crear\_comentario(...)** 46](#_Toc206107384)

[**🔍 obtener\_comentario\_por\_id(idcomentario)** 47](#_Toc206107385)

[**📋 listar\_comentarios()** 47](#_Toc206107386)

[**✏️ actualizar\_comentario(idcomentario, datos)** 47](#_Toc206107387)

[**🗑️ eliminar\_comentario(idcomentario)** 47](#_Toc206107388)

[**🔗 9. Relaciones y Dependencias** 47](#_Toc206107389)

[**🛡️ 10. Recomendaciones de Mejora** 48](#_Toc206107390)

[**Modulo Proyecto** 49](#_Toc206107391)

[**🧩 1. Estructura General del Módulo** 49](#_Toc206107392)

[**📍 2. Endpoints Disponibles** 49](#_Toc206107393)

[**🧾 3. Gestión de Proyectos** 49](#_Toc206107394)

[**🧍 Ver proyectos (GET /proyectos/)** 49](#_Toc206107395)

[**➕ Crear proyecto (POST /proyectos/nuevo)** 49](#_Toc206107396)

[**✏️ Editar proyecto (POST /proyectos/editar/<id>)** 50](#_Toc206107397)

[**🗑️ Eliminar proyecto (POST /proyectos/eliminar/<id>)** 50](#_Toc206107398)

[**📥 4. API REST: Listado de Proyectos** 51](#_Toc206107399)

[**⚙️ 5. Funciones CRUD - app/crud/proyecto\_crud.py** 51](#_Toc206107400)

[**🔧 crear\_proyecto(nombre\_proyecto, idusuario, fecha\_creacion, logoproyecto=None)** 51](#_Toc206107401)

[**🔍 obtener\_proyecto\_por\_id(idproyecto)** 51](#_Toc206107402)

[**📋 listar\_proyectos()** 52](#_Toc206107403)

[**✏️ actualizar\_proyecto(idproyecto, datos)** 52](#_Toc206107404)

[**🗑️ eliminar\_proyecto(idproyecto)** 52](#_Toc206107405)

[**🔗 6. Relaciones y Dependencias** 52](#_Toc206107406)

[**🛡️ 7. Recomendaciones de Mejora** 53](#_Toc206107407)

# Módulo de Adjunto

**🧩 1. Estructura General del Módulo**

|  |  |
| --- | --- |
| **Archivo** | **Función** |
| app/crud/adjunto\_crud.py | Funciones CRUD para adjuntos usando SQLAlchemy |
| app/routes/adjunto\_routes.py | Define rutas HTML y API para gestionar adjuntos |
| Carpeta de subida | uploads\_adjuntos/ donde se guardan los archivos subidos |
| Librerías utilizadas | werkzeug.utils.secure\_filename para nombres seguros de archivos, uuid para generar prefijos únicos |

**📍 2. Endpoints Disponibles**

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **Ruta** | **Acción** | **Tipo** |
| GET | /adjuntos/ | Vista HTML con listado de adjuntos | Vista (HTML) |
| POST | /adjuntos/nuevo | Subir nuevo adjunto | Vista (HTML) |
| GET | /adjuntos/descargar/<id> | Descargar/ver adjunto | Vista (HTML) |
| POST | /adjuntos/eliminar/<id> | Eliminar adjunto | Vista (HTML) |
| GET | /adjuntos/api | Listar adjuntos en formato JSON | API REST |

**🧾 3. Flujo de Gestión de Archivos**

**➕ Subir adjunto (/adjuntos/nuevo)**

1. Se recibe el archivo desde un formulario (request.files['adjunto']).
2. Se valida que exista y que tenga un tipo permitido (jpg, pdf, docx, etc.).
3. Se genera un nombre único con uuid4() y secure\_filename.
4. Se guarda en la carpeta uploads\_adjuntos/.
5. Se crea el registro en la base de datos usando crear\_adjunto(...):
   * ruta\_archivo
   * tipo (MIME)
   * idusuario (usuario que sube el archivo)
   * descripcion
   * fecha\_subida (datetime.now())
6. Mensaje flash de éxito o error según corresponda.
7. Redirecciona al listado de adjuntos.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

**🗂️ Listar adjuntos (/adjuntos/)**

* Se obtiene la lista completa usando listar\_adjuntos().
* Se renderiza la plantilla adjuntos\_view.html con los adjuntos disponibles.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**📥 Descargar adjunto (/adjuntos/descargar/<id>)**

1. Se busca el adjunto por ID (obtener\_adjunto\_por\_id).
2. Si existe y el archivo está en la carpeta, se retorna con send\_file.
3. Si no existe, se muestra un mensaje flash de error y se redirige al listado.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**🗑️ Eliminar adjunto (/adjuntos/eliminar/<id>)**

1. Se busca el adjunto por ID.
2. Se elimina de la base de datos con eliminar\_adjunto(id).
3. Mensaje flash de éxito o error.
4. Redirecciona al listado de adjuntos.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**⚙️ 4. Funciones CRUD Internas – app/crud/adjunto\_crud.py**

|  |  |
| --- | --- |
| **Función** | **Descripción** |
| crear\_adjunto(ruta\_archivo, tipo, idusuario, descripcion, fecha\_subida) | Crea un nuevo registro en la base de datos y lo retorna. |
| obtener\_adjunto\_por\_id(idadjunto) | Retorna un adjunto según su ID. |
| listar\_adjuntos() | Retorna todos los registros de adjuntos. |
| actualizar\_adjunto(idadjunto, datos) | Actualiza los campos dinámicamente usando un diccionario. Retorna el objeto actualizado o None si no existe. |
| eliminar\_adjunto(idadjunto) | Elimina el adjunto por ID. Retorna True si se eliminó, False si no existe. |

**🛡️ 5. Recomendaciones de Mejora**

|  |  |
| --- | --- |
| **Área** | **Mejora sugerida** |
| Validaciones | Verificar duplicados de archivos si es necesario, validar tamaño máximo de archivo |
| Seguridad | Validar sesión del usuario, restringir acceso a archivos privados, agregar CSRF en formularios |
| Accesos | Decoradores como @login\_required o @admin\_required para proteger rutas |
| Auditoría | Registrar logs de subida, descarga y eliminación de archivos |
| API | Añadir endpoints para subir/editar archivos por JSON |

# Módulo de Estado

**🧩 1. Estructura General del Módulo**

|  |  |
| --- | --- |
| **Archivo** | **Función** |
| app/crud/estado\_crud.py | Funciones CRUD para Estados usando SQLAlchemy |
| app/routes/estado\_routes.py | Define rutas HTML y API para gestionar Estados |
| Librerías utilizadas | Flask Blueprint, SQLAlchemy, jsonify |

**📍 2. Endpoints Disponibles**

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **Ruta** | **Acción** | **Tipo** |
| GET | /estados/ | Listar todos los estados en HTML | Vista (HTML) |
| POST | /estados/nuevo | Crear un nuevo estado | Vista (HTML) |
| POST | /estados/editar/<idestado> | Editar un estado existente | Vista (HTML) |
| POST | /estados/eliminar/<idestado> | Eliminar un estado | Vista (HTML) |
| GET | /estados/api | Listar todos los estados en formato JSON | API REST |

**🧾 3. Flujo de Gestión de Estados**

**🧍 Ver estados (/estados/)**

* Se obtiene la lista completa usando listar\_estados().
* Se renderiza la plantilla estados\_view.html mostrando todos los registros.

Texto

El contenido generado por IA puede ser incorrecto.

**➕ Crear nuevo estado (/estados/nuevo)**

1. Se reciben los datos del formulario (request.form.to\_dict()):
   * nombre\_estado
   * orden (opcional)
   * color (opcional)
2. Se crea el registro en la base de datos usando crear\_estado(...).
3. Redirecciona a la vista de listado de estados.

Texto

El contenido generado por IA puede ser incorrecto.

**✏️ Editar estado (/estados/editar/<idestado>)**

1. Se obtienen los datos del formulario.
2. Se actualizan dinámicamente los campos con actualizar\_estado(idestado, datos).
3. Redirecciona a la vista principal de estados.

Texto

El contenido generado por IA puede ser incorrecto.

**🗑️ Eliminar estado (/estados/eliminar/<idestado>)**

1. Se elimina el registro con eliminar\_estado(idestado).
2. Redirecciona al listado de estados.

Texto

El contenido generado por IA puede ser incorrecto.

**📥 API REST (/estados/api)**

* Retorna JSON con todos los estados.
* Requiere que el modelo Estado implemente .as\_dict() para conversión a JSON.

Un reloj digital en la pantalla

El contenido generado por IA puede ser incorrecto.

**⚙️ 4. Funciones CRUD Internas – app/crud/estado\_crud.py**

|  |  |
| --- | --- |
| **Función** | **Descripción** |
| crear\_estado(nombre\_estado, orden=None, color=None) | Crea un nuevo estado y lo retorna. |
| obtener\_estado\_por\_id(idestado) | Retorna un estado según su ID. |
| listar\_estados() | Retorna todos los registros de estados. |
| actualizar\_estado(idestado, datos) | Actualiza campos dinámicamente. Retorna el objeto actualizado o None si no existe. |
| eliminar\_estado(idestado) | Elimina un estado por ID. Retorna True si se eliminó, False si no existe. |

**🛡️ 5. Recomendaciones y Mejoras**

|  |  |
| --- | --- |
| **Área** | **Mejora sugerida** |
| Validaciones | Validar campos duplicados (nombre\_estado), vacíos o inválidos |
| Seguridad | Agregar CSRF en formularios, protección de rutas con @login\_required |
| Accesos | Restringir edición y eliminación solo a administradores |
| Auditoría | Registrar logs de creación, edición y eliminación de estados |
| API | Añadir endpoints para creación y edición vía JSON (REST) |
| Interfaz | Colores por defecto según tipo de estado, ordenar lista por orden para mayor claridad |
| UX | Confirmaciones de eliminación y mensajes flash para acciones exitosas o errores |

# Módulo de Historial

**🧩 1. Estructura General del Módulo**

|  |  |
| --- | --- |
| **Archivo** | **Función** |
| app/crud/historial\_crud.py | Funciones CRUD para Historial usando SQLAlchemy |
| app/routes/historial\_routes.py | Define rutas HTML y API para gestionar Historial |
| Librerías utilizadas | Flask Blueprint, SQLAlchemy, jsonify |

**📍 2. Endpoints Disponibles**

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **Ruta** | **Acción** | **Tipo** |
| GET | /historial/ | Listar todos los historiales en HTML | Vista (HTML) |
| POST | /historial/nuevo | Crear un nuevo historial | Vista (HTML) |
| POST | /historial/editar/<idhistorial> | Editar historial existente | Vista (HTML) |
| POST | /historial/eliminar/<idhistorial> | Eliminar historial | Vista (HTML) |
| GET | /historial/api | Listar todos los historiales en formato JSON | API REST |

**🧾 3. Flujo de Gestión de Historial**

**🧍 Ver historial (/historial/)**

* Se obtiene la lista completa usando listar\_historiales().
* Se renderiza la plantilla historial\_view.html mostrando todos los registros con detalles como:
  + Tarea asociada (idtarea)
  + Usuario que realizó la acción (idusuario)
  + Fecha de registro (fecha)
  + Descripción
  + Estado anterior y nuevo
  + Adjunto asociado (idadjunto)

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**➕ Crear historial (/historial/nuevo)**

1. Se reciben los datos del formulario (request.form.to\_dict()):
   * idtarea
   * idusuario
   * fecha
   * descripcion
   * estado\_anterior
   * estado\_nuevo
   * idadjunto
2. Se crea el registro en la base de datos usando crear\_historial(...).
3. Redirecciona a la vista de historial.

Texto

El contenido generado por IA puede ser incorrecto.

**✏️ Editar historial (/historial/editar/<idhistorial>)**

1. Se obtienen los datos del formulario.
2. Se actualizan dinámicamente los campos con actualizar\_historial(idhistorial, datos).
3. Redirecciona a la vista principal del historial.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**🗑️ Eliminar historial (/historial/eliminar/<idhistorial>)**

1. Se elimina el registro con eliminar\_historial(idhistorial).
2. Redirecciona al listado de historiales.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**📥 API REST (/historial/api)**

* Retorna JSON con todos los historiales.
* Requiere que el modelo Historial tenga implementado .as\_dict() para conversión a JSON.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

**⚙️ 4. Funciones CRUD Internas – app/crud/historial\_crud.py**

|  |  |
| --- | --- |
| **Función** | **Descripción** |
| crear\_historial(idtarea, idusuario, fecha, descripcion, estado\_anterior, estado\_nuevo, idadjunto) | Crea un nuevo historial y lo retorna. |
| obtener\_historial\_por\_id(idhistorial) | Retorna un historial según su ID. |
| listar\_historiales() | Retorna todos los registros de historial. |
| actualizar\_historial(idhistorial, datos) | Actualiza campos dinámicamente. Retorna el objeto actualizado o None si no existe. |
| eliminar\_historial(idhistorial) | Elimina un historial por ID. Retorna True si se eliminó, False si no existe. |

**🛡️ 5. Recomendaciones y Mejoras**

|  |  |
| --- | --- |
| **Área** | **Mejora sugerida** |
| Validaciones | Verificar campos vacíos, tipos de datos correctos y existencia de idtarea, idusuario y idadjunto. |
| Seguridad | Agregar CSRF en formularios, protección de rutas con @login\_required y permisos según rol. |
| Accesos | Restringir edición y eliminación solo a administradores o usuarios responsables. |
| Auditoría | Registrar logs de creación, edición y eliminación de historiales. |
| API | Añadir endpoints para crear y actualizar historiales vía JSON (REST). |
| UX | Ordenar historiales por fecha descendente y destacar cambios de estado críticos. |

# Módulo de Notificaciones

**🧩 1. Estructura General del Módulo**

|  |  |
| --- | --- |
| **Archivo** | **Función** |
| app/crud/notificacion\_crud.py | Funciones CRUD para Notificaciones usando SQLAlchemy |
| app/routes/notificacion\_routes.py | Define rutas HTML y API para gestionar Notificaciones |
| Librerías utilizadas | Flask Blueprint, SQLAlchemy, jsonify |

**📍 2. Endpoints Disponibles**

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **Ruta** | **Acción** | **Tipo** |
| GET | /notificaciones/ | Listar todas las notificaciones en HTML | Vista (HTML) |
| POST | /notificaciones/nuevo | Crear una nueva notificación | Vista (HTML) |
| POST | /notificaciones/editar/<idnotificacion> | Editar notificación existente | Vista (HTML) |
| POST | /notificaciones/eliminar/<idnotificacion> | Eliminar notificación | Vista (HTML) |
| GET | /notificaciones/api | Listar todas las notificaciones en JSON | API REST |

**🧾 3. Flujo de Gestión de Notificaciones**

**🧍 Ver notificaciones (/notificaciones/)**

* Se obtiene la lista completa usando listar\_notificaciones().
* También se obtienen tareas pendientes usando listar\_tareas\_pendientes().
* Se renderiza la plantilla notificaciones\_view.html mostrando:
  + Notificaciones del usuario.
  + Tareas pendientes relacionadas.

Texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

**➕ Crear notificación (/notificaciones/nuevo)**

1. Se reciben los datos del formulario (request.form.to\_dict()):
   * idusuario
   * idtarea
   * mensaje
   * fecha\_envio
   * tipo
   * leida (opcional, default False)
2. Se crea el registro en la base de datos usando crear\_notificacion(...).
3. Redirecciona a la vista de notificaciones.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**✏️ Editar notificación (/notificaciones/editar/<idnotificacion>)**

1. Se obtienen los datos del formulario.
2. Se actualizan dinámicamente los campos con actualizar\_notificacion(idnotificacion, datos).
3. Redirecciona a la vista principal de notificaciones.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**🗑️ Eliminar notificación (/notificaciones/eliminar/<idnotificacion>)**

1. Se elimina el registro con eliminar\_notificacion(idnotificacion).
2. Redirecciona al listado de notificaciones.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**📥 API REST (/notificaciones/api)**

* Retorna JSON con todas las notificaciones.
* Requiere que el modelo Notificacion tenga implementado .as\_dict() para conversión a JSON.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

**⚙️ 4. Funciones CRUD Internas – app/crud/notificacion\_crud.py**

|  |  |
| --- | --- |
| **Función** | **Descripción** |
| crear\_notificacion(idusuario, idtarea, mensaje, fecha\_envio, tipo, leida=False) | Crea una nueva notificación y la retorna. |
| obtener\_notificacion\_por\_id(idnotificacion) | Retorna una notificación según su ID. |
| listar\_notificaciones() | Retorna todos los registros de notificaciones. |
| actualizar\_notificacion(idnotificacion, datos) | Actualiza campos dinámicamente. Retorna el objeto actualizado o None si no existe. |
| eliminar\_notificacion(idnotificacion) | Elimina una notificación por ID. Retorna True si se eliminó, False si no existe. |
| listar\_tareas\_pendientes() | Retorna todas las tareas que no están en estado "Completada", para relacionarlas con notificaciones. |

**🛡️ 5. Recomendaciones y Mejoras**

|  |  |
| --- | --- |
| **Área** | **Mejora sugerida** |
| Validaciones | Verificar que idtarea y idusuario existan antes de crear notificación. |
| Seguridad | Agregar CSRF en formularios y proteger rutas con @login\_required. |
| Accesos | Permitir edición y eliminación solo a usuarios responsables o administradores. |
| Auditoría | Registrar logs de envío, lectura y eliminación de notificaciones. |
| UX | Marcar notificaciones leídas/no leídas, notificaciones urgentes resaltadas por color. |
| API | Agregar endpoints para marcar notificación como leída vía JSON y filtrar por usuario. |

# Módulo de Prioridades

**🧩 1. Estructura General del Módulo**

|  |  |
| --- | --- |
| **Archivo** | **Función** |
| app/crud/prioridad\_crud.py | CRUD de prioridades con SQLAlchemy |
| app/routes/prioridad\_routes.py | Rutas HTML y API para prioridades |
| Librerías | Flask Blueprint, SQLAlchemy, jsonify |

**📍 2. Endpoints Disponibles**

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **Ruta** | **Acción** | **Tipo** |
| GET | /prioridades/ | Listar prioridades en HTML | Vista 🖥️ |
| POST | /prioridades/nuevo | Crear nueva prioridad | Vista ➕ |
| POST | /prioridades/editar/<idprioridad> | Editar prioridad | Vista ✏️ |
| POST | /prioridades/eliminar/<idprioridad> | Eliminar prioridad | Vista 🗑️ |
| GET | /prioridades/api | Listar prioridades en JSON | API 📡 |

**🧾 3. Flujo de Gestión de Prioridades**

**🧍 Ver prioridades (/prioridades/)**

* Se obtiene la lista completa con listar\_prioridades().
* Se renderiza la plantilla prioridades\_view.html mostrando:
  + Nombre de la prioridad
  + Color asociado

Texto

El contenido generado por IA puede ser incorrecto.

**➕ Crear prioridad (/prioridades/nuevo)**

1. Datos recibidos desde formulario:
   * nombre\_prioridad (obligatorio)
   * color (opcional) 🎨
2. Se crea registro con crear\_prioridad(...).
3. Redirecciona automáticamente a la vista de prioridades.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**✏️ Editar prioridad (/prioridades/editar/<idprioridad>)**

1. Datos recibidos desde formulario.
2. Actualización dinámica usando actualizar\_prioridad(idprioridad, datos).
3. Redirecciona a vista principal.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**🗑️ Eliminar prioridad (/prioridades/eliminar/<idprioridad>)**

1. Se elimina el registro con eliminar\_prioridad(idprioridad).
2. Redirecciona al listado de prioridades.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**📥 API REST (/prioridades/api)**

* Retorna JSON con todas las prioridades.
* Requiere que el modelo Prioridad implemente .as\_dict().

Un reloj digital en la pantalla

El contenido generado por IA puede ser incorrecto.

**⚙️ 4. Funciones CRUD Internas**

|  |  |
| --- | --- |
| **Función** | **Descripción** |
| crear\_prioridad(nombre\_prioridad, color=None) | Crea y retorna una nueva prioridad ➕ |
| obtener\_prioridad\_por\_id(idprioridad) | Obtiene prioridad por ID 🔎 |
| listar\_prioridades() | Lista todos los registros 📋 |
| actualizar\_prioridad(idprioridad, datos) | Actualiza campos dinámicamente ✏️ |
| eliminar\_prioridad(idprioridad) | Elimina prioridad por ID 🗑️ |

**🛡️ 5. Recomendaciones y Mejora**

|  |  |
| --- | --- |
| **Área** | **Mejora sugerida** |
| Validaciones | Evitar duplicados, validar campos vacíos y colores HEX 🎯 |
| Seguridad | Implementar CSRF y proteger rutas con @login\_required 🔒 |
| Accesos | Solo administradores pueden editar o eliminar 👮 |
| Auditoría | Registrar logs de creación, edición y eliminación 📜 |
| UX | Ordenar prioridades y mostrar colores visualmente 🎨 |
| API | Endpoints para crear o actualizar vía JSON y filtros 📡 |

# Módulo de Roll

**🧩 1. Estructura General del Módulo**

|  |  |
| --- | --- |
| **Archivo** | **Función** |
| app/crud/roll\_crud.py | CRUD de roles con SQLAlchemy |
| app/routes/roll\_routes.py | Rutas HTML y API para roles |
| Librerías | Flask Blueprint, SQLAlchemy, jsonify |

**📍 2. Endpoints Disponibles**

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **Ruta** | **Acción** | **Tipo** |
| GET | /roles/ | Listar roles en HTML | Vista 🖥️ |
| POST | /roles/nuevo | Crear nuevo rol | Vista ➕ |
| POST | /roles/eliminar/<idroll> | Eliminar rol | Vista 🗑️ |
| GET | /roles/api | Listar roles en JSON | API 📡 |

**🧾 3. Flujo de Gestión de Roles**

**🧍 Ver roles (/roles/)**

* Se obtiene la lista completa usando listar\_rolls().
* Se renderiza plantilla rolls\_view.html mostrando todos los roles registrados.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**➕ Crear rol (/roles/nuevo)**

1. Recibe datos del formulario:
   * nombre\_roll (obligatorio)
2. Se crea registro con crear\_roll(nombre\_roll).
3. Redirecciona automáticamente a la vista principal.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**🗑️ Eliminar rol (/roles/eliminar/<idroll>)**

1. Se elimina registro usando eliminar\_roll(idroll).
2. Redirecciona al listado de roles.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**📥 API REST (/roles/api)**

* Retorna JSON con todos los roles usando .as\_dict(). 🛠️

Un reloj digital en la pantalla

El contenido generado por IA puede ser incorrecto.

**⚙️ 4. Funciones CRUD Internas – app/crud/roll\_crud.py**

|  |  |
| --- | --- |
| **Función** | **Descripción** |
| crear\_roll(nombre\_roll) | Crea un nuevo rol ➕ |
| obtener\_roll\_por\_id(idroll) | Obtiene rol por ID 🔎 |
| listar\_rolls() | Devuelve todos los roles 📋 |
| actualizar\_roll(idroll, nombre\_roll) | Actualiza nombre de rol ✏️ |
| eliminar\_roll(idroll) | Elimina rol por ID 🗑️ |

**🛡️ 5. Recomendaciones y Mejora**

|  |  |
| --- | --- |
| **Área** | **Mejora sugerida** |
| Validaciones | Evitar duplicados y nombres vacíos 🎯 |
| Seguridad | Proteger rutas con @login\_required 🔒 |
| Accesos | Solo administradores pueden eliminar o editar 👮 |
| Auditoría | Registrar logs de creación y eliminación de roles 📜 |
| UX | Mostrar roles en lista ordenada y amigable 🖥️ |
| API | Endpoint de creación y edición vía JSON 📡 |

# Módulo de Tipo\_Tarea

**🧩 1. Estructura General del Módulo**

|  |  |
| --- | --- |
| **Archivo** | **Función** |
| app/crud/tipo\_tarea\_crud.py | CRUD de tipos de tarea con SQLAlchemy |
| app/routes/tipo\_tarea\_routes.py | Rutas HTML y API para tipos de tarea |
| Librerías | Flask Blueprint, SQLAlchemy, jsonify |

**📍 2. Endpoints Disponibles**

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **Ruta** | **Acción** | **Tipo** |
| GET | /tipos-tarea/ | Listar tipos de tarea en HTML | Vista 🖥️ |
| POST | /tipos-tarea/nuevo | Crear nuevo tipo de tarea | Vista ➕ |
| POST | /tipos-tarea/editar/<idtipo\_tarea> | Editar tipo de tarea | Vista ✏️ |
| POST | /tipos-tarea/eliminar/<idtipo\_tarea> | Eliminar tipo de tarea | Vista 🗑️ |
| GET | /tipos-tarea/api | Listar tipos de tarea en JSON | API 📡 |

**\🧾 3. Flujo de Gestión de Tipos de Tarea**

**🧍 Ver tipos de tarea (/tipos-tarea/)**

* Se obtiene la lista completa con listar\_tipos\_tarea().
* Se renderiza la plantilla tipos\_tarea\_view.html mostrando:
  + Nombre del tipo de tarea
  + Descripción (opcional)

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**➕ Crear tipo de tarea (/tipos-tarea/nuevo)**

1. Datos recibidos desde formulario:
   * nombre\_tipo (obligatorio)
   * descripcion (opcional) 📝
2. Se crea registro con crear\_tipo\_tarea(nombre\_tipo, descripcion).
3. Redirecciona automáticamente a la vista de tipos de tarea.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**✏️ Editar tipo de tarea (/tipos-tarea/editar/<idtipo\_tarea>)**

1. Se obtienen los datos del formulario.
2. Se actualizan dinámicamente con actualizar\_tipo\_tarea(idtipo\_tarea, datos).
3. Redirecciona a la vista principal.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**🗑️ Eliminar tipo de tarea (/tipos-tarea/eliminar/<idtipo\_tarea>)**

1. Se elimina el registro con eliminar\_tipo\_tarea(idtipo\_tarea).
2. Redirecciona al listado de tipos de tarea.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**📥 API REST (/tipos-tarea/api)**

* Retorna JSON con todos los tipos de tarea usando .as\_dict()

Un reloj digital en la pantalla

El contenido generado por IA puede ser incorrecto.

**⚙️ 4. Funciones CRUD Internas – app/crud/tipo\_tarea\_crud.py**

|  |  |
| --- | --- |
| **Función** | **Descripción** |
| crear\_tipo\_tarea(nombre\_tipo, descripcion=None) | Crea un nuevo tipo de tarea ➕ |
| obtener\_tipo\_tarea\_por\_id(idtipo\_tarea) | Obtiene un tipo de tarea por ID 🔎 |
| listar\_tipos\_tarea() | Devuelve todos los registros 📋 |
| actualizar\_tipo\_tarea(idtipo\_tarea, datos) | Actualiza campos dinámicamente ✏️ |
| eliminar\_tipo\_tarea(idtipo\_tarea) | Elimina tipo de tarea por ID 🗑️ |

**🛡️ 5. Recomendaciones y Mejora**

|  |  |
| --- | --- |
| **Área** | **Mejora sugerida** |
| Validaciones | Evitar nombres duplicados y campos vacíos 🎯 |
| Seguridad | Proteger rutas con @login\_required 🔒 |
| Accesos | Solo administradores pueden eliminar o editar 👮 |
| Auditoría | Registrar logs de creación, edición y eliminación 📜 |
| UX | Ordenar tipos de tarea y mostrar descripciones de forma clara 🖥️ |
| API | Endpoints de creación y edición vía JSON 📡 |

# Modulo Tarea

**🧩 1. Estructura General del Módulo**

|  |  |
| --- | --- |
| **Archivo** | **Función** |
| app/routes/tarea\_routes.py | Define las rutas del módulo, tanto HTML como API REST |
| app/crud/tarea\_crud.py | Contiene las funciones de acceso a datos (CRUD) con SQLAlchemy |
| Modelos relacionados | Tarea, Estado, Usuario, Proyecto, Prioridad, Adjunto, Notificación |

**📍 2. Endpoints Disponibles**

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **Ruta** | **Acción** | **Tipo** |
| GET | /tareas/ | Vista principal HTML (panel + registro de tareas) | Vista (HTML) |
| POST | /tareas/nuevo | Registrar nueva tarea | Vista (HTML) |
| POST | /tareas/eliminar/<idtarea> | Eliminar tarea existente | Vista (HTML) |
| GET | /tareas/api | Listar todas las tareas en formato JSON | API REST |

**🧠 3. Flujo Interno de Funcionalidades**

**✅ Crear una Tarea**

1. Formulario envía POST a /tareas/nuevo
2. Se extraen los campos del formulario HTML
3. Se ejecuta **crear\_tarea**(...) en tarea\_crud.py
4. Texto

   El contenido generado por IA puede ser incorrecto.Se guarda la tarea con db.session.commit()
5. Si **enviar\_notificacion** está marcado, se registra una notificación

Texto

El contenido generado por IA puede ser incorrecto.

1. Se muestra flash() de éxito y redirecciona

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

**🗑️ Eliminar una Tarea**

* Se accede vía POST a /tareas/eliminar/<idtarea>
* Texto

  El contenido generado por IA puede ser incorrecto.Se llama a eliminar\_tarea(id) en el CRUD
* Se muestra mensaje flash() de confirmación

Texto

El contenido generado por IA puede ser incorrecto.

**📥 API REST: Listado de Tareas**

Texto

El contenido generado por IA puede ser incorrecto.

* Devuelve un JSON con todas las tareas
* Utiliza método .as\_dict() en cada tarea para serialización
* Útil para frontend o consumo externo (apps móviles, JS dinámico)

**🧾 4. Vista Principal /tareas/**

* Renderiza vista HTML con:
  + Panel tipo Kanban
  + Formulario de nueva tarea
  + Filtros por estado/proyecto

Texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación, Teams

El contenido generado por IA puede ser incorrecto.

**🔄 Agrupación dinámica:**

Texto

El contenido generado por IA puede ser incorrecto.

* Las tareas se agrupan por nombre de proyecto
* Permite visualización por bloques usando JS

**⚙️ 5. CRUD Interno (tarea\_crud.py)**

| **Función** | **Descripción** |
| --- | --- |
| crear\_tarea(...) | Inserta nueva tarea en la base de datos |
| listar\_tareas() | Devuelve todas las tareas |
| listar\_tareas\_pendientes() | Solo tareas cuyo estado ≠ 'Completada' |
| obtener\_tarea\_por\_id() | Busca tarea por ID |
| actualizar tarea() | Actualiza campos usando un dict dinámico |
| eliminar\_tarea() | Elimina tarea por su ID |

**🔗 6. Relaciones y Dependencias**

* **Proyecto**: una tarea está asociada a un proyecto (idproyecto)
* **Usuario asignado**: identifica quién ejecutará la tarea
* **Estado** y **Prioridad**: definen su seguimiento y urgencia
* **Adjunto**: opción para agregar archivos
* **Notificación**: sistema interno que avisa de nuevas tareas

**🛡️ 7. Recomendaciones de Mejora**

|  |  |
| --- | --- |
| **Área** | **Mejora sugerida** |
| Validaciones | Asegurar que todos los campos estén correctamente validados |
| Seguridad | Implementar protección CSRF |
| Acceso | Añadir autenticación y control de roles |
| API REST | Ampliar con filtros: /tareas/api?estado=pendiente |
| Métodos | Crear endpoints REST POST, PUT, DELETE con JSON |

# Modulo Usuario

**🧩 1. Estructura General del Módulo**

|  |  |
| --- | --- |
| **Archivo** | **Función** |
| app/routes/usuario\_routes.py | Define rutas HTML y API para gestionar usuarios |
| app/crud/usuario\_crud.py | Funciones CRUD para usuarios usando SQLAlchemy |
| app/crud/roll\_crud.py | Carga de roles (perfiles de usuario) |
| Utiliza librería | werkzeug.security para hashear contraseñas |

**📍 2. Endpoints Disponibles**

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **Ruta** | **Acción** | **Tipo** |
| GET | /usuarios/ | Vista HTML con listado de usuarios | Vista (HTML) |
| POST | /usuarios/nuevo | Crear nuevo usuario desde formulario | Vista (HTML) |
| POST | /usuarios/editar/<id> | Editar usuario existente | Vista (HTML) |
| POST | /usuarios/eliminar/<id> | Eliminar usuario | Vista (HTML) |
| GET | /usuarios/api | Obtener todos los usuarios en JSON | API REST |
| GET/POST | /usuarios/login | Página de login y autenticación | Vista (HTML) |
| GET | /usuarios/logout | Cerrar sesión del usuario | Vista (HTML) |
| GET | /usuarios/dashboard | Página principal después del login | Vista (HTML) |

**🔐 3. Flujo de Autenticación**

**✅ Inicio de sesión (/usuarios/login)**

* Se recibe correo y password por POST
* Se valida el usuario con check\_password\_hash
* Se guardan datos clave en la sesión:

Texto

El contenido generado por IA puede ser incorrecto.

* Redirecciona al dashboard en caso exitoso
* Muestra error si las credenciales no coinciden

**🚪 Cierre de sesión (/usuarios/logout)**

* Se limpia la sesión con session.clear()
* Redirecciona al formulario de login

Texto

El contenido generado por IA puede ser incorrecto.

**🧾 4. Gestión de Usuarios**

**🧍 Ver usuarios (GET /usuarios/)**

* Renderiza usuarios\_view.html con:
  + Lista de todos los usuarios
  + Lista de roles disponibles

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

**➕ Crear nuevo usuario (POST /usuarios/nuevo)**

* Recibe datos de formulario: nombre, correo, contraseña, rol
* La contraseña se encripta con generate\_password\_hash(...)
* Se guarda usando crear\_usuario(...)

Texto

El contenido generado por IA puede ser incorrecto.

**✏️ Editar usuario (POST /usuarios/editar/<idusuario>)**

* Convierte los datos del form en un dict
* Si se incluye password, se hashea antes de actualizar
* Si la contraseña está vacía, **no se actualiza**
* Llama a actualizar\_usuario(id, datos)

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

**🗑️ Eliminar usuario (POST /usuarios/eliminar/<idusuario>)**

* Llama directamente a eliminar\_usuario(id)
* Redirecciona a la vista principal

Texto

El contenido generado por IA puede ser incorrecto.

**📥 5. API REST: Lista de Usuarios**

Texto

El contenido generado por IA puede ser incorrecto.

* Retorna JSON con todos los usuarios registrados
* Requiere que el modelo Usuario implemente .as\_dict()

**⚙️ 6. Funciones CRUD Internas - app/crud/usuario\_crud.py**

Este archivo contiene todas las funciones responsables de interactuar con la base de datos usando SQLAlchemy, específicamente para la tabla Usuario.

****Crea un nuevo registro de usuario con los datos proporcionados.

* Encripta la contraseña automáticamente con generate\_password\_hash()
* El parámetro activo por defecto está en True

Texto

El contenido generado por IA puede ser incorrecto.

✅ **Retorna** el objeto usuario ya guardado en la base de datos.

Texto

El contenido generado por IA puede ser incorrecto.

**🔍 obtener\_usuario\_por\_id(idusuario)**

Busca un usuario por su ID primario.

Texto

El contenido generado por IA puede ser incorrecto.

**📧 obtener\_usuario\_por\_correo(correo)**

Busca un usuario usando el campo correo.  
Se usa principalmente para login o validación de duplicados.

Pantalla de un celular con letras

El contenido generado por IA puede ser incorrecto.

**📋 listar\_usuarios()**

Devuelve todos los registros de usuarios.

Una captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

**✏️ actualizar\_usuario(idusuario, datos)**

Actualiza campos dinámicamente usando un diccionario.

* Si el campo es password, automáticamente lo encripta.
* Usa setattr() para actualizar atributos de forma flexible.

Texto

El contenido generado por IA puede ser incorrecto.

✅ **Retorna** el objeto actualizado si existe, o None si no se encuentra.

**🗑️ eliminar\_usuario(idusuario)**

Elimina el usuario por ID si existe.  
✅ Retorna True si se eliminó correctamente, o False si no se encontró.

Texto

El contenido generado por IA puede ser incorrecto.

**🛡️ 7. Recomendaciones de Mejora**

|  |  |
| --- | --- |
| **Área** | **Mejora sugerida** |
| Validaciones | Validar campos duplicados (correo), vacíos o incorrectos |
| Seguridad | Agregar CSRF, protección contra inyecciones y sesiones seguras |
| Accesos | Proteger rutas con decoradores (@login\_required, @admin\_required) |
| Auditoría | Registrar logs de acceso, errores y cambios |
| API | Añadir endpoints para login y edición por JSON (REST) |

## **Modulo Comentario**

**🧩 1. Estructura General del Módulo**

|  |  |
| --- | --- |
| **Archivo** | **Función** |
| app/routes/comentario\_routes.py | Define rutas HTML y API para gestionar comentarios |
| app/crud/comentario\_crud.py | Funciones CRUD con SQLAlchemy |
| Relacionado con | Tarea, Usuario, Adjunto, Estado |
| Extras | Soporte para archivos adjuntos y cambio de estado |

**📍 2. Endpoints Disponibles**

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **Ruta** | **Acción** | **Tipo** |
| GET/POST | /comentarios/ | Vista principal y creación de comentarios | Vista (HTML) |
| POST | /comentarios/editar/<id> | Editar comentario existente | Vista (HTML) |
| POST | /comentarios/eliminar/<id> | Eliminar comentario (con adjunto) | Vista (HTML) |
| GET | /comentarios/api | Obtener todos los comentarios en JSON | API REST |

**🧾 3. Vista de Comentarios**

**🔍 Filtrado y Visualización (GET /comentarios/)**

* Muestra tareas y comentarios filtrados por proyecto (idproyecto) y tarea (idtarea)
* Carga los siguientes elementos:
  + Lista de tareas (listar\_tareas)
  + Lista de usuarios (listar\_usuarios)
  + Estados disponibles (listar\_estados)
  + Adjuntos existentes (listar\_adjuntos)
* Si se pasa idtarea, muestra los comentarios relacionados

Texto

El contenido generado por IA puede ser incorrecto.

**➕ 4. Crear Comentario (POST /comentarios/)**

**🧠 Flujo:**

1. Se envía un formulario con comentario, idtarea, idestado, archivo (opcional)
2. Se valida el archivo con allowed\_file(...)
3. Si se adjunta archivo:
   * Se guarda con nombre único
   * Se crea un registro con crear\_adjunto(...)
4. Se crea el comentario con crear\_comentario(...)
5. Si el idestado es distinto al actual de la tarea:
   * Se actualiza con actualizar\_tarea(...)
6. Se redirige con flash("Comentario agregado correctamente")

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**📝 Datos que se registran:**

* idtarea, idusuario, comentario
* fecha del comentario (puede ser ingresada manualmente o actual)
* idadjunto (si se sube uno)
* idestado (nuevo estado de la tarea, si aplica)

**✏️ 5. Editar Comentario (POST /comentarios/editar/<id>)**

* Extrae los datos del formulario con request.form.to\_dict()
* Llama a actualizar\_comentario(idcomentario, datos)

Texto

El contenido generado por IA puede ser incorrecto.

**🗑️ 6. Eliminar Comentario (POST /comentarios/eliminar/<id>)**

**Pasos:**

1. Obtiene el comentario por ID
2. Si tiene adjunto asociado:
   * Elimina el archivo físico con os.remove(...) (si existe)
   * Elimina el registro del adjunto (eliminar\_adjunto(...))
3. Elimina el comentario con eliminar\_comentario(...)
4. Redirige a la vista principal

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

**📥 7. API REST: Listado de Comentarios**

Un reloj digital en la pantalla

El contenido generado por IA puede ser incorrecto.

* Devuelve la lista completa de comentarios en JSON
* Requiere que el modelo Comentario implemente .as\_dict()

**⚙️ 8. Funciones CRUD - app/crud/comentario\_crud.py**

**🔧 crear\_comentario(...)**

Crea un nuevo comentario relacionado a una tarea.  
Opcionalmente se registra adjunto y cambio de estado.

Texto

El contenido generado por IA puede ser incorrecto.

**🔍 obtener\_comentario\_por\_id(idcomentario)**

Busca y retorna un comentario por su ID.

Una captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

**📋 listar\_comentarios()**

Devuelve todos los comentarios registrados.

Un conjunto de letras blancas en un fondo negro

El contenido generado por IA puede ser incorrecto.

**✏️ actualizar\_comentario(idcomentario, datos)**

Actualiza dinámicamente los campos del comentario con los datos del formulario.

Texto

El contenido generado por IA puede ser incorrecto.

**🗑️ eliminar\_comentario(idcomentario)**

Elimina el comentario por ID.

Texto

El contenido generado por IA puede ser incorrecto.

**🔗 9. Relaciones y Dependencias**

|  |  |
| --- | --- |
| **Entidad** | **Relación** |
| Tarea | Cada comentario está asociado a una tarea |
| Usuario | El autor del comentario |
| Adjunto | Se puede subir archivo opcional |
| Estado | Posible cambio de estado de la tarea en el mismo comentario |

**🛡️ 10. Recomendaciones de Mejora**

|  |  |
| --- | --- |
| **Área** | **Mejora** |
| Validaciones | Verificar que el archivo no exceda tamaño límite |
| Seguridad | Implementar CSRF, validación de permisos para modificar/eliminar |
| API | Crear endpoints POST/PUT/DELETE para gestionar comentarios vía REST |
| Logs | Registrar eventos de comentarios, cambios de estado y archivos |

## **Modulo Proyecto**

**🧩 1. Estructura General del Módulo**

|  |  |
| --- | --- |
| **Archivo** | **Función** |
| app/routes/proyecto\_routes.py | Rutas HTML y API para gestión de proyectos |
| app/crud/proyecto\_crud.py | Funciones CRUD con SQLAlchemy |
| Modelo involucrado | Proyecto |
| Relación | Cada proyecto está asociado a un usuario (idusuario) |

**📍 2. Endpoints Disponibles**

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **Ruta** | **Acción** | **Tipo** |
| GET | /proyectos/ | Vista de proyectos | Vista (HTML) |
| POST | /proyectos/nuevo | Crear nuevo proyecto | Vista (HTML) |
| POST | /proyectos/editar/<id> | Editar proyecto | Vista (HTML) |
| POST | /proyectos/eliminar/<id> | Eliminar proyecto | Vista (HTML) |
| GET | /proyectos/api | Obtener todos los proyectos en JSON | API REST |

**🧾 3. Gestión de Proyectos**

**🧍 Ver proyectos (GET /proyectos/)**

* Renderiza proyectos\_view.html
* Carga todos los proyectos registrados
* Muestra los usuarios (para asignar responsables)

Texto

El contenido generado por IA puede ser incorrecto.

**➕ Crear proyecto (POST /proyectos/nuevo)**

* Recibe datos desde un formulario HTML:
  + nombre\_proyecto
  + idusuario
  + fecha\_creacion (en formato YYYY-MM-DD)
* Convierte la fecha con datetime.strptime(...)
* Llama a crear\_proyecto(...) para guardar

Texto

El contenido generado por IA puede ser incorrecto.

**✏️ Editar proyecto (POST /proyectos/editar/<id>)**

* Extrae y convierte los nuevos datos del formulario
* Llama a actualizar\_proyecto(...) con un diccionario

Texto

El contenido generado por IA puede ser incorrecto.

**🗑️ Eliminar proyecto (POST /proyectos/eliminar/<id>)**

* Llama a eliminar\_proyecto(id)
* Redirecciona a la vista principal de proyectos

Texto

El contenido generado por IA puede ser incorrecto.

**📥 4. API REST: Listado de Proyectos**

Una captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

* Devuelve todos los proyectos registrados en formato JSON
* Requiere que el modelo Proyecto tenga el método .as\_dict()

**⚙️ 5. Funciones CRUD - app/crud/proyecto\_crud.py**

**🔧 crear\_proyecto(nombre\_proyecto, idusuario, fecha\_creacion, logoproyecto=None)**

Crea un nuevo proyecto y lo guarda en la base de datos.  
Opción para incluir un logotipo (por defecto None).

Texto

El contenido generado por IA puede ser incorrecto.

**🔍 obtener\_proyecto\_por\_id(idproyecto)**

Busca un proyecto específico por su ID.

Texto

El contenido generado por IA puede ser incorrecto.

**📋 listar\_proyectos()**

Devuelve todos los proyectos almacenados en la base de datos.

Texto

El contenido generado por IA puede ser incorrecto.

**✏️ actualizar\_proyecto(idproyecto, datos)**

* Modifica los atributos indicados en el diccionario datos
* Usa setattr() para aplicar los cambios dinámicamente

Texto

El contenido generado por IA puede ser incorrecto.

**🗑️ eliminar\_proyecto(idproyecto)**

* Elimina el proyecto por su ID si existe
* Retorna True si la operación fue exitosa

Texto

El contenido generado por IA puede ser incorrecto.

**🔗 6. Relaciones y Dependencias**

* Cada proyecto tiene un usuario responsable (idusuario)
* Puede asociarse a tareas (dependiendo de otros módulos)
* Tiene fecha de creación y, opcionalmente, logotipo (logoproyecto)

**🛡️ 7. Recomendaciones de Mejora**

|  |  |
| --- | --- |
| **Área** | **Mejora sugerida** |
| Validaciones | Comprobar existencia del usuario antes de crear |
| Seguridad | Agregar CSRF y control de acceso a las rutas |
| Interfaz | Permitir carga de logotipo de proyecto |
| API | Añadir filtros por usuario, fecha, etc. |
| REST | Implementar métodos POST/PUT/DELETE para API JSON |