Evidencias de sistema y Base de Datos

Proyecto TaskControl

Integrantes:

Michael Cid Molina

Luciano Pino Aguilar

Vania Vargas Dinen

Fecha:

17/10/2025

Contenido

[Introducción 3](#_Toc211625705)

[Evidencias del Backend 4](#_Toc211625706)

[Estructura general 5](#_Toc211625707)

[Endpoint principales 6](#_Toc211625708)

[Validaciones y respuestas 8](#_Toc211625709)

[Pruebas y conexión con la base de datos 9](#_Toc211625710)

[Evidencias del Frontend 10](#_Toc211625711)

[Conclusión 12](#_Toc211625712)

# Introducción

El presente documento reúne las evidencias funcionales del sistema TaskControl, correspondientes al desarrollo e implementación del backend, frontend y la base de datos.

Su propósito es demostrar la correcta ejecución de las etapas técnicas del proyecto, mostrando cómo los distintos componentes del sistema se integran para ofrecer una solución funcional y escalable.

El sistema TaskControl fue diseñado para optimizar la gestión de tareas, carga de evidencias y generación de reportes, integrando tecnologías modernas:

FastAPI (Python) para el backend, encargado de la lógica de negocio y comunicación con la base de datos.

React para el frontend, que permite la interacción del usuario con el sistema de forma rápida e intuitiva.

Supabase/PostgreSQL para la base de datos, responsable del almacenamiento y la integridad de la información.

Estas evidencias respaldan la funcionalidad del sistema, demostrando la correcta integración entre las capas de presentación, negocio y datos implementadas en TaskControl.

Las evidencias que se presentan a continuación demuestran el funcionamiento técnico del sistema y su integración completa entre las capas de presentación, negocio y datos.

# Evidencias del Backend

El backend del proyecto TaskControl fue desarrollado con FastAPI, el cual permitió implementar una API REST modular, eficiente y segura.

Su función principal es gestionar la lógica del sistema y procesar las solicitudes provenientes del frontend, conectándose a la base de datos en Supabase/PostgreSQL.

A continuación, se presenta la interfaz generada automáticamente por Swagger, donde se visualizan los principales endpoints del backend.

Documentación automática generada por Swagger UI.

Se observan los endpoints principales del backend: /users, /providers, /tasks y /google-auth, implementados en FastAPI.

Esta interfaz permite visualizar, probar y validar cada ruta de la API.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

# Estructura general

El backend se organizó siguiendo un esquema de capas simples:

* *main.py:* punto de inicio de la aplicación, donde se inicializa FastAPI y se configuran las rutas.
* *routes/:* contiene los controladores o endpoints de cada módulo (usuarios, proveedores, tareas).
* *models/:* define las estructuras de datos y las validaciones utilizadas por cada entidad.
* database.py: maneja la conexión con Supabase/PostgreSQL.
* *schemas/:* define los modelos Pydantic utilizados para validar y serializar datos entre el cliente y el servidor.

Esta estructura permitió mantener un código ordenado, fácil de mantener y escalable.

# Endpoint principales

Los endpoints se documentaron automáticamente mediante Swagger UI, una herramienta que facilita la interacción con la API y la verificación de las operaciones.

Entre los endpoints principales implementados se encuentran:

Endpoint Método Descripción

*/users/ GET* Retorna la lista completa de usuarios registrados en la base de datos.

*/users/{id}* GET Devuelve los datos de un usuario específico mediante su ID.

*/providers/* GET Muestra la información de los proveedores asociados al sistema.

*/providers/* POST Permite registrar un nuevo proveedor en la base de datos.

*/tasks/ GET* Obtiene todas las tareas registradas con su estado y responsable.

*/tasks/ POST* Crea una nueva tarea asignando un responsable y una dependencia.

*/tasks/{id}* PUT Actualiza los datos o estado de una tarea existente.

*/tasks/{id}* DELETE Elimina una tarea específica del sistema.

Cada endpoint fue probado y validado correctamente desde Swagger, asegurando el funcionamiento integral del backend y su correcta comunicación con la base de datos.

Ejemplo del endpoint /tasks ejecutado desde Swagger.

Permite obtener la lista completa de tareas registradas en la base de datos Supabase, validando la conexión y el funcionamiento del método GET.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

# Validaciones y respuestas

El backend utiliza modelos Pydantic para validar los datos enviados desde el frontend antes de procesarlos.

Esto garantiza que solo se registren datos válidos y completos en la base de datos.

Por ejemplo:

* En la creación de tareas, se valida que los campos obligatorios (nombre, fecha, responsable y dependencia) estén correctamente definidos.
* En la creación de usuarios o proveedores, se comprueba que los identificadores (RUT, correo, etc.) sean únicos.
* Interfaz de usuario gráfica, Texto, Aplicación

  El contenido generado por IA puede ser incorrecto.Cada solicitud HTTP genera una respuesta estandarizada en formato JSON, indicando el estado de la operación (éxito o error) y los datos procesados.

Ejemplo de respuesta del endpoint /users en formato JSON. Muestra la información de los usuarios almacenados, confirmando la correcta comunicación entre el backend y la base de datos.

# Pruebas y conexión con la base de datos

Durante las pruebas realizadas mediante Swagger UI, se validó la correcta comunicación entre FastAPI y la base de datos Supabase/PostgreSQL.

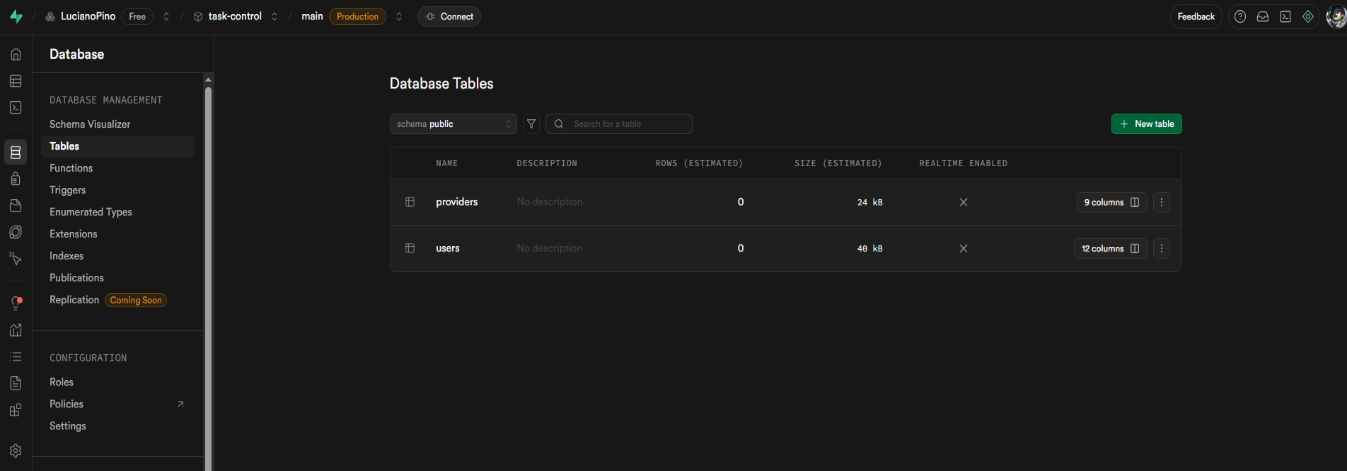
Las operaciones CRUD (crear, leer, actualizar, eliminar) fueron ejecutadas con éxito, confirmando la estabilidad del backend.

Además, se comprobó:

* La correcta creación de registros desde el endpoint /tasks/.
* La lectura y visualización de usuarios y proveedores almacenados.
* La actualización de estados de tareas en tiempo real.
* Estas pruebas confirmaron que el backend cumple su rol como intermediario entre la interfaz del usuario y la base de datos, garantizando la integridad y disponibilidad de la información.

La base de datos del sistema TaskControl fue implementada en Supabase/PostgreSQL y diseñada bajo un modelo relacional.

A través de esta estructura, el sistema puede almacenar la información de usuarios, proveedores, dependencias y tareas, manteniendo la coherencia y relación entre las distintas entidades.



# Evidencias del Frontend

El frontend de TaskControl fue desarrollado utilizando HTML, CSS, JavaScript y React, tecnologías que permitieron construir una interfaz moderna, dinámica y adaptable a diferentes dispositivos.

Su objetivo principal es facilitar la interacción entre usuarios y proveedores, entregando una experiencia visual clara y un flujo de navegación intuitivo para la gestión y seguimiento de tareas.

La estructura en HTML y CSS permitió definir una base sólida y estéticamente coherente, mientras que JavaScript y React aportaron interactividad y modularidad al sistema, posibilitando la conexión directa con la API REST implementada en FastAPI.

De esta forma, las acciones realizadas desde el frontend, como el inicio de sesión, la creación de tareas o la carga de evidencia, se comunican de manera eficiente con la base de datos alojada en Supabase/PostgreSQL.

Vista general del sistema, donde los usuarios pueden autenticarse mediante Google Auth, garantizando un acceso seguro y rápido.

Esta pantalla fue diseñada bajo principios de usabilidad asegurando una primera interacción clara.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

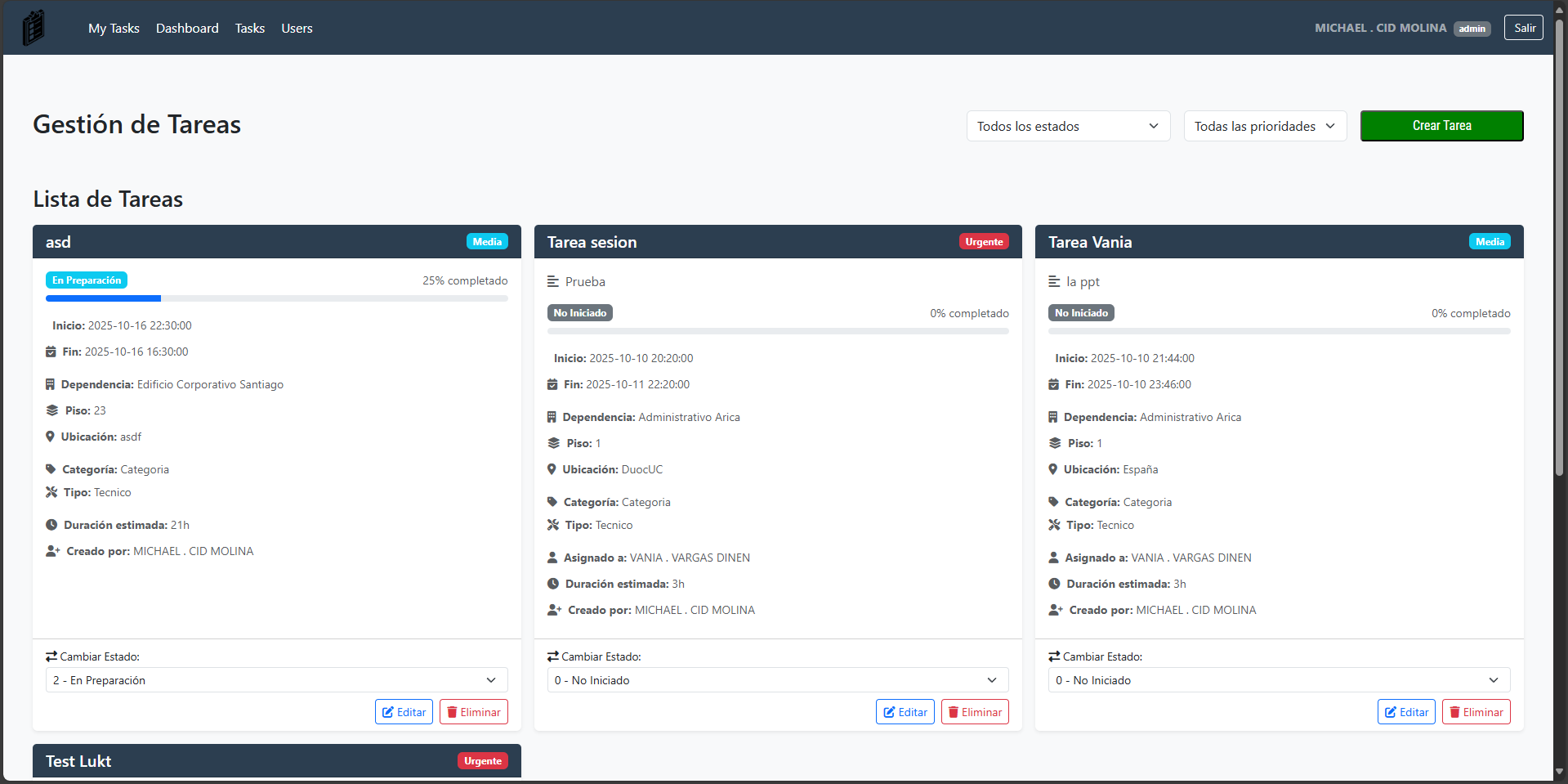
Interfaz del módulo de gestión de usuarios.

Permite agregar, editar o eliminar registros, además de visualizar los roles y dependencias asignadas a cada usuario.Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Vista del módulo de tareas.

Muestra las tareas registradas con sus estados (No iniciado, En preparación, Finalizado), lo que permite al usuario monitorear y actualizar la información en tiempo real.



# Conclusión

El proyecto TaskControl logró integrar de forma eficiente las capas de frontend, backend y base de datos, consolidando un sistema funcional, estable y alineado con los objetivos definidos.

El uso de FastAPI permitió construir un backend ágil y seguro, mientras que React, HTML, CSS y JavaScript facilitaron el desarrollo de una interfaz intuitiva, moderna y adaptable.

Por su parte, la implementación de la base de datos en Supabase/PostgreSQL garantizó la integridad y trazabilidad de la información.

Las pruebas realizadas confirmaron la correcta comunicación entre los componentes, validando la interacción fluida entre la API y la interfaz del usuario.

En conjunto, estas evidencias demuestran un proceso de desarrollo bien planificado, técnicamente sólido y resultado de un trabajo colaborativo enfocado en entregar una solución práctica y eficiente para la gestión de tareas.