
Especificación de requisitos de software

Para

Construction Company SRS

Versión Final aprobado

**Preparado por Jesus Andres Perez Fuentes, Gabriel Elias Leon
Simancas, Valery Georgina Aurela Perez**

Universidad Tecnologica de Bolivar

5/07/2025

Tabla de contenido

1. Introducción	1
2. Descripción general	2
2.1 Características del producto	2
2.2 Clases de usuarios y características	2
2.3 Limitaciones de diseño y aplicación	2
2.4 Documentación del Usuario	2
2.5 Supuestos y dependencias	3
3. Características del sistema	4
3.1 Características de Sistema	4
3.1.1 Descripción y prioridad	4
3.1.2 Secuencias de estímulo/respuesta	4
3.1.3 Requisitos funcionales	4
4. Requisitos de la interfaz	5
4.1 Interfaces de Usuario	5
4.2 Interfaces de hardware	5
4.3 Interfaces de software	5
4.4 Interfaces de Comunicación	5
5. Otros requisitos No funcionales	6
5.1 Requerimientos de desempeño	6
5.2 Requisitos de seguridad	6
5.3 Atributos de calidad del software	
6. Otros Requerimientos	6
7. Pruebas y validación	7
8. Referencias	8

Historial de revisiones

Nombre	Fecha	Motivo de los cambios	Versión
Construction Company	17/03/2025	Creación inicial del documento SRS para el Proyecto DS.	0.1
Construction Company	24/03/2025	Incorporación de arquitectura basada en microservicios, definiendo APIs y actualización del flujo de trabajo.	0.2
Construction Company	31/03/2025	Incorporación de diseño estructural 1.0, diseño del Mockup UI, y definición de restricciones.	1.0
Construction Company	8/04/2025	Actualización de requerimientos funcionales y no funcionales. Inclusión de plataformas específicas por requerimiento.	1.1
Construction Company	09/04/2025	Se actualizó el diagrama de contexto, para reflejar nuevos actores y módulos del sistema.	2
Construction Company	23/04/2025	Inclusión del microservicio de registro de equipos. Actualización de requisitos	3

Especificacion de requisitos de software para Construction Company Page 2

		funcionales, interfaces y arquitectura.	
Construction Company	06/05/2025	Actualización de avances en implementación de servicios, pruebas unitarias, integración y cobertura con SonarCloud.	4
Construction Company	29/05/2025	Optimización y Validación de Servicios	5
Construction Company	05/05/2025	Refinamiento y Optimización Avanzada	6
Construction Company	06/06/2025	Consolidación Funcional y UX/UI	Final

1. Introducción

Este documento detalla el desarrollo de un software diseñado para optimizar la gestión en empresas de construcción. La finalidad de este sistema es incrementar la eficiencia operativa a través de la automatización de procesos, la centralización de la información y la disminución de errores humanos. Entre los beneficios que ofrece se destacan una mayor visibilidad del avance de los proyectos, un mejor control sobre costos y recursos, así como una toma de decisiones más informada y oportuna.

2. Descripción general

El nuevo software ha sido diseñado especialmente para el sector de la construcción, y no se trata de una actualización de algún producto anterior, sino de una solución independiente que responde a las necesidades específicas de esta industria. Su enfoque abarca aspectos como la gestión de proyectos, el control de costos, el seguimiento de recursos y la administración del personal. Además, el sistema se ha rediseñado adoptando una arquitectura basada en microservicios, lo que mejora significativamente su escalabilidad y flexibilidad. Esta nueva estructura permite que cada componente del sistema funcione de manera independiente, facilitando su mantenimiento y la integración con tecnologías modernas.

2.1 Características del producto

- Gestión de proyectos: Planificación, cronogramas y seguimiento de progreso.
- Control de costos: Presupuestos, previsiones y comparativas en tiempo real.
- Gestión de recursos: Asignación y seguimiento de maquinaria y personal.
- Compras e inventarios: Gestión de proveedores y materiales.
- Informes y análisis: Automatización de informes para seguimiento y toma de decisiones.

2.2 Clases de usuarios y características

- Gerentes de proyecto: Usuarios avanzados con acceso a todas las funcionalidades de gestión y planificación.
- Supervisores de obra: Usuarios con acceso a la actualización del estado del proyecto y registro de avances diarios.
- Personal administrativo: Usuarios responsables de controlar los costos y gestionar las compras, con acceso limitado a la parte financiera.
- Proveedores externos: Acceso limitado a módulos de compras e inventarios para la gestión de suministros.

2.3 Limitaciones de diseño y aplicación

Las limitaciones incluyen la necesidad de integrar el software con sistemas ERP ya existentes en las

empresas de construcción. También se requiere compatibilidad con dispositivos móviles, ya que el personal en obra necesita acceso en tiempo real desde el campo. Así mismo, se seguirán las regulaciones locales de protección de datos y seguridad para asegurar la confidencialidad de la información.

El sistema deberá soportar operaciones en paralelo, ya que varios usuarios interactuarán simultáneamente. También será necesario que funcione sin problemas en infraestructuras con conexiones de red limitadas, como puede suceder en obras remotas.

2.4 Documentación del Usuario

En esta fase del proyecto, correspondiente al desarrollo del MVP (Producto Mínimo Viable), la única documentación formal disponible es la Especificación de Requisitos de Software (SRS), la cual describe detalladamente los módulos, funcionalidades, flujos y restricciones del sistema.

Se contempla para futuras etapas del desarrollo la elaboración de la siguiente documentación complementaria:

- Manual de usuario completo, organizado por módulos.
- Guías rápidas por tipo de usuario (administrador, supervisor, proveedor).
- Documentación en línea integrada al sistema.
- Videos instructivos y tutoriales interactivos.

Estas entregas están planificadas como parte de las versiones siguientes, una vez que el sistema haya alcanzado una etapa más madura y funcionalmente estable.

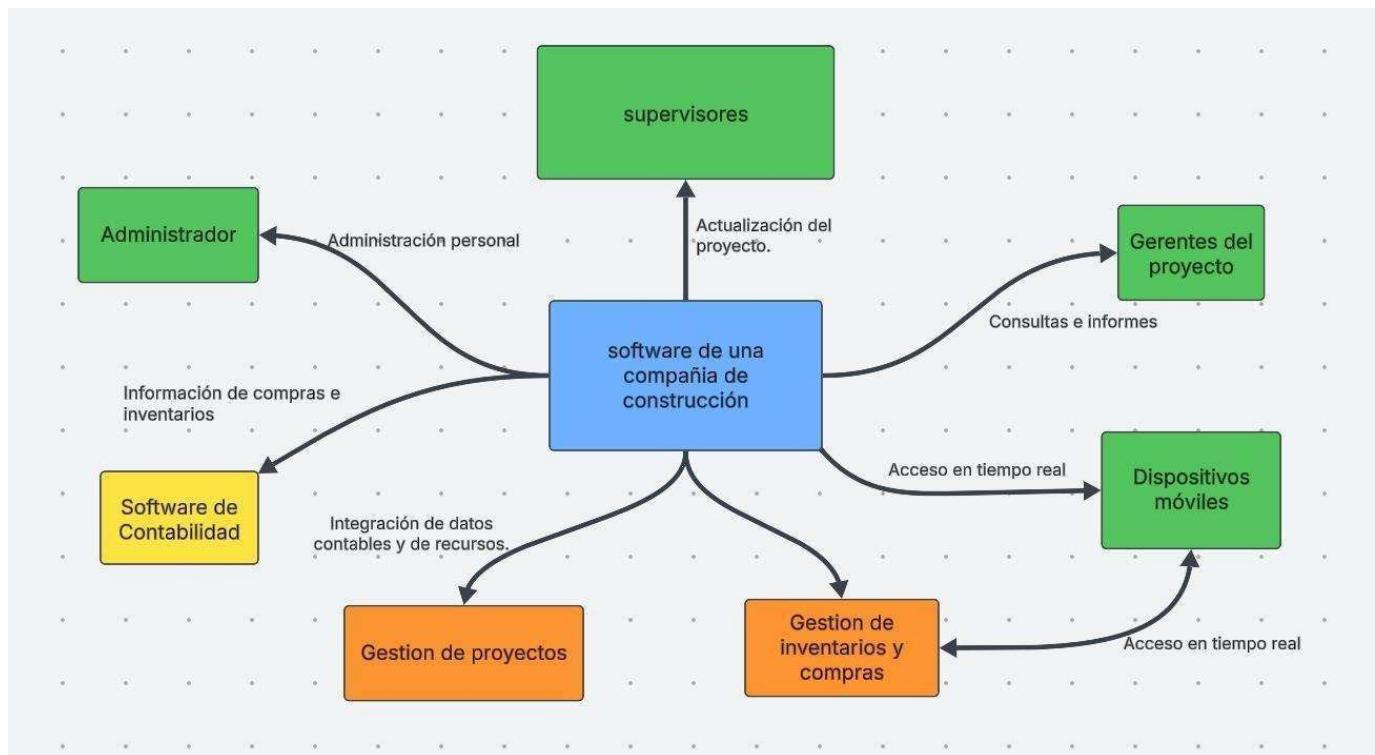
2.5 Supuestos y dependencias

El éxito del proyecto presupone que las empresas disponen de una infraestructura informática básica que facilite la integración de este software. Así mismo, se asume que los usuarios clave recibirán la capacitación necesaria para operar el sistema de manera efectiva. La efectividad del software dependerá, por tanto, de su correcta integración con los sistemas ERP ya existentes y de la disponibilidad de dispositivos móviles en las obras.

3. Características del sistema

Esta sección describe los requisitos funcionales organizados en base a las características principales del sistema. En este caso, la organización se basa en los servicios clave del producto dentro del **MVP**.

Cada módulo del sistema, como la gestión de proyectos, el control de inventarios y la gestión de proveedores, se implementa como un servicio independiente. Además, se ha optimizado la capacidad de integración con otros sistemas externos, tales como ERP e IoT, mediante el uso de APIs REST que aseguran una comunicación segura y eficiente.



3.1 MVP-Producto Mínimo Viable

El MVP de este sistema tiene como objetivo optimizar la gestión centralizada de procesos en proyectos de construcción, asegurando un control eficiente de equipos, inventarios y planificación de trabajos.

3.1.1 Funcionalidades Principales del MVP

Gestión Centralizada de Procesos

- Administración de equipos, inventarios, programación de trabajos y proyectos en una sola plataforma.
- Control de recursos y materiales para evitar retrasos y sobrecostos.

Integración de Datos mediante APIs

- Unificación de datos de diferentes sistemas en una sola plataforma.
- APIs que permitan la conexión con software de contabilidad, logística y recursos humanos.

Automatización de Flujos de Trabajo

- Notificaciones automáticas sobre disponibilidad de equipos y materiales.
- Asignación optimizada de recursos basada en programación manual o reglas predefinidas.

Plataforma en la Nube

- Acceso desde cualquier dispositivo con conexión a internet.
- Sincronización en tiempo real de cambios en inventario, horarios y disponibilidad de equipos.

Dashboards e Informes

- Visualización de datos clave mediante gráficos y reportes interactivos.
- Generación de informes sobre uso de equipos, inventario y costos.

Seguridad y Control de Acceso

- Diferentes niveles de permisos según el rol del usuario.
- Registro de auditoría para rastrear cambios en la información.

3.2 Secuencias de estímulo/respuesta

Un administrador consulta la disponibilidad de equipos

- Estimulo: El administrador accede al módulo de inventario y consulta un equipo específico.
- Respuesta: El sistema muestra el estado actual (disponible/en uso) y su ubicación.

Un supervisor asigna recursos a un proyecto

- Estimulo: El supervisor selecciona un proyecto y elige los equipos/materiales necesarios.
- Respuesta: El sistema sugiere recursos disponibles y confirma la asignación.

Integración con software externo

- Estimulo: Se solicita la sincronización de datos con el software de contabilidad.
- Respuesta: El sistema envía y recibe información de costos y pagos en tiempo real.

Seguridad y control de acceso

- Estimulo: Un usuario intenta acceder a un módulo sin permisos.
- Respuesta: El sistema deniega el acceso y registra el intento en el log de auditoría.

3.2.1 Requisitos funcionales

Los siguientes requerimientos funcionales describen las operaciones principales que el sistema debe realizar:

ID	Nombre	Descripción	Prioridad	Page
RF01	Gestión de Empleados	Permitir crear, leer, actualizar y eliminar registros de empleados. Asociar empleados a programas y equipos.	Alta	
RF02	Consulta de Empleados por Proyecto	Mostrar los empleados asignados a un programa de trabajo específico.	Media	
RF03	Gestión de Materiales	Registrar, consultar, actualizar y eliminar información de materiales. Controlar stock y movimientos de entrada/salida.	Alta	
RF04	Gestión de Equipos	Esta funcionalidad será proporcionada mediante el microservicio registro-equipos, el cual expone operaciones REST para listar, registrar, actualizar y eliminar equipos. Este servicio también maneja el historial de uso y su integración con la gestión de programas de abajo.	Alta	
RF06	Gestión de Programas de Trabajo	Crear, editar, consultar y eliminar programas de trabajo. Asociar, empleados, equipos y materiales.	Alta	
RF07	Seguimiento de Tareas por Programa	Permitir dividir el programa en tareas (TR1, TR2, etc.) y asignarlas a responsables y fechas.	Alta	
RF08	Registro de Actividades Diarias	Registrar las actividades realizadas por día, incluyendo los recursos usados.	Alta	
RF09	Visualización de Progreso	Mostrar una línea de tiempo y/o gráficas para ver el progreso diario, semanal o total, del programa.	Media	

RF10	Generación de Reportes	Generar reportes detallados por proyecto, incluyendo avance, uso de materiales, equipos y empleados.	Alta
RF11	Historial de Recursos	Consultar el historial de uso de recursos por programa o proyecto.	Media

4. Requisitos de la interfaz

4.1 Interfaces de Usuario

El sistema contara con una interfaz gráfica intuitiva y responsive, diseñada para ser utilizada en computadoras y dispositivos móviles. Se incluirán las siguientes características:

- **Panel de control** con visualización en tiempo real del estado de los equipos y materiales.
- **Módulo de gestión de proyectos**, con acceso a planificación, cronogramas y seguimiento de progreso.
- **Registro y consulta de inventarios**, permitiendo a los usuarios actualizar el estado de los equipos.
- **Acceso restringido por roles**, asegurando que solo los usuarios autorizados puedan realizar ciertas acciones.
- **Notificaciones y alertas**, para informar sobre disponibilidad de equipos y cambios en la programación.

4.2 Interfaces de hardware

El software será compatible con dispositivos como:

- **Computadoras de escritorio y laptops** con Windows, MacOS y Linux.
- **Tabletas y teléfonos móviles** con sistemas operativos Android y iOS.
- **Sensores IoT** para monitoreo de equipos y materiales en tiempo real (en futuras iteraciones).

4.3 Interfaces de software

El sistema integrara múltiples APIs para optimizar el flujo de datos, incluyendo:

- **API REST** para exponer información sobre inventarios, equipos y programación.
- **Integración con sistemas ERP** utilizados por la empresa de construcción.
- **Soporte para bases de datos SQL y NoSQL**, asegurando flexibilidad y escalabilidad.
- **Microservicio registro-equipos**, implementado en Fasta API, se comunica con el sistema principal mediante rutas definidas (/registro-equipos). Proporciona endpoints para operaciones CRUD sobre equipos.

4.4 Interfaces de Comunicación

El sistema se comunicará utilizando:

- **Protocolos HTTP y HTTPS** para transmisión de datos segura.
- **Autenticación mediante OAuth 2.0**, garantizando acceso seguro a la plataforma.
- **Soporte para comunicación vía correo electrónico y notificaciones push** para alertas y reportes.

Nota: El sistema se ha desarrollado utilizando una arquitectura de microservicios, lo que mejora la escalabilidad y facilita la integración con otros sistemas.

4.5 Microservicio: Registro de Equipos

Este microservicio, desarrollado con Python y FastAPI, forma parte de la arquitectura de microservicios implementada para el sistema. Se encarga exclusivamente de gestionar el ciclo de vida de los equipos utilizados en obra: registrar, consultar, actualizar y eliminar datos de maquinaria.

Está desacoplado del sistema principal, permitiendo su despliegue, actualización y escalado de manera independiente.

Endpoints expuestos:

- **GET /registro-equipos/equipos:** Lista todos los equipos.
- **POST /registro-equipos/equipos:** Registra un nuevo equipo.
- **PUT /registro-equipos/equipos/{id}:** Actualiza datos de un equipo.
- **DELETE /registro-equipos/equipos/{id}:** Elimina un equipo.

Este microservicio asegura separación de responsabilidades y facilita el mantenimiento del sistema a largo plazo.

5. Otros requisitos No funcionales

5.1 Requerimientos de desempeño

Durante esta fase inicial, se han establecido objetivos básicos de rendimiento para validar la viabilidad del sistema. En este MVP:

- Se busca un tiempo de respuesta promedio menor a 2 segundos para operaciones CRUD principales.
- Se ha probado el sistema con una carga básica de usuarios simultáneos, pero el soporte para 500 usuarios concurrentes queda previsto para versiones futuras.
- Se utiliza una base de datos optimizada con índices definidos para las consultas más frecuentes.
- El coverage de pruebas del backend supera el 80%, garantizando confiabilidad del código en los servicios esenciales.

5.2 Requisitos de seguridad

El sistema en su MVP implementa mecanismos básicos de seguridad:

- Autenticación con token bajo esquema OAuth 2.0.
- Cifrado de contraseñas mediante hashing seguro.
- Control de acceso por roles.

Para futuras versiones se contempla:

- Cifrado de datos sensibles (AES-256).
- Autenticación multifactor (MFA) en acciones críticas.
- Cumplimiento normativo completo con GDPR y leyes locales de protección de datos.

5.3 Atributos de calidad del software

- El sistema ya se beneficia de una arquitectura basada en microservicios, facilitando su escalabilidad futura.
- La usabilidad básica ha sido validada mediante pruebas funcionales y ajustes en la interfaz.
- Existe compatibilidad entre múltiples navegadores y dispositivos móviles gracias a Flutter.
- El código presenta una buena mantenibilidad, con integración continua activa (CI), pruebas automatizadas y manejo de errores controlado.

6. Otros requerimientos

6.1 Documentación técnica del Backend y API

El sistema backend está construido en Python utilizando FastAPI como framework principal para la creación de microservicios.

Cada entidad esencial (usuarios, equipos, materiales, tareas, programas, pagos, inventarios, entre otros) cuenta con un conjunto de endpoints RESTful.

Los microservicios exponen operaciones CRUD sobre sus entidades, autenticadas con OAuth 2.0 y documentadas automáticamente mediante OpenAPI/Swagger.

Ejemplo de endpoints:

- GET /api/users/ – Lista todos los usuarios.
- POST /api/tasks/ – Crea una nueva tarea.
- PUT /api/inventory/{id} – Actualiza el inventario.
- DELETE /api/payments/{id} – Elimina un registro de pago

Todos los endpoints fueron validados con pruebas unitarias y de integración automatizadas, alcanzando una cobertura superior al 80% según métricas de SonarCloud.

6.2 Integración continua y control de calidad

Se configuró un flujo de integración continua utilizando GitHub Actions, donde cada push o pull request ejecuta:

- Linter y chequeo de estilo de código.
- Pruebas unitarias y de integración con PyTest.
- Cálculo de cobertura de código.
- Análisis de calidad de código con SonarCloud.

Se garantiza que el backend no sea integrado a la rama main si no cumple con un coverage mínimo del 80%, y con una puntuación aceptable de mantenibilidad y duplicación de código.

6.3 Diseño y desarrollo del frontend

El frontend fue desarrollado con Flutter, partiendo de una plantilla base que fue personalizada con los estilos y funcionalidades del sistema de gestión constructora.

Mejoras implementadas:

- Visualización en tiempo real de datos del backend (proyectos, inventario, actividades).
- Corrección de errores visuales y validaciones en formularios.
- Mejora del rendimiento en la navegación.
- Experiencia de usuario fluida para perfiles como: administrador, supervisor y proveedor. Se validaron los flujos completos: login → dashboard → acciones CRUD.

6.6 Requerimientos no funcionales del MVP

6.6.1 Requerimientos de desempeño

- Tiempo de respuesta menor a 2 segundos para consultas de disponibilidad.
- Soporte para al menos 500 usuarios concurrentes (objetivo futuro).
- Uso de bases de datos optimizadas y consultas indexadas para rendimiento.
- El sistema mantiene un coverage del 80% en backend.

6.6.2 Requisitos de seguridad

- Autenticación mediante OAuth 2.0.
- Cifrado de contraseñas y control de acceso por roles.
- Futuro soporte para cifrado AES-256 y autenticación multifactor (MFA).
- Cumplimiento con normativas como la GDPR y la Ley de Protección de Datos local (etapa futura).

6.6.3 Atributos de calidad del software

- Escalabilidad gracias a la arquitectura por microservicios.
- Usabilidad validada mediante pruebas funcionales.
- Portabilidad entre dispositivos y navegadores mediante Flutter.
- Mantenibilidad, validación continua, gestión de errores robusta y experiencia de usuario mejorada.

En esta sección se explica con que plataformas se desarrollara cada requerimiento funcional:

ID	Nombre del Requerimiento	Plataformas y Herramientas de Desarrollo
RF01	Gestión de Empleados	Frontend (HTML/CSS/JS) para formularios y tablas, FastAPI (Python) para lógica CRUD, MongoDB/PostgreSQL para almacenar datos.
RF02	Consulta de Empleados por Proyecto	Dashboard en HTML/JS, endpoint en FastAPI, consulta en base de datos relacional.
RF03	Gestión de Materiales	Formularios y filtros en el frontend, FastAPI para lógica de stock, MongoDB para inventario.
RF04	Alertas de Stock de Materiales	Backend (FastAPI) con lógica condicional, notificaciones vía correo o pantalla, sincronizado con MongoDB.
RF05	Gestión de Equipos	CRUD en frontend y backend, manejo de estados con FastAPI y almacenamiento en base de datos.
RF06	Gestión de Programas de Trabajo	Pantallas interactivas para cronogramas, backend con lógica de planificación, persistencia en base de datos.
RF07	Seguimiento de Tareas por Programa	Frontend con módulos de calendario o Gantt simple, backend organiza etapas, DB registra fechas y responsables.
RF08	Registro de Actividades Diarias	Formulario diario en frontend, API en backend que guarda los datos en la base de datos con sello de tiempo.
RF09	Visualización de Progreso	Graficas con JS (Chart.js o similares), API de backend provee datos procesados, frontend renderiza.
RF10	Generación de Reportes	API en FastAPI genera reportes, exportables en PDF o visualizados en HTML, datos extraidos de la DB.
RF11	Historial de Recursos	Consulta avanzada en la DB vía API, visualización con tablas dinámicas en el frontend.

7. Pruebas y validación

7.1 Pruebas admin.py

The screenshot shows a browser window with several tabs open. The active tab is titled "Pruebas admin" and displays a terminal interface. The terminal shows the following command and its output:

```
curl -X GET "http://localhost:8000/zona-restringida"
```

Output:

```
onfig.py:321: UserWarning: Valid config keys have changed in V2:  
* 'orm_mode' has been renamed to 'from_attributes'  
    warnings.warn(message, UserWarning)  
INFO:     Started server process [27418]  
INFO:     Waiting for application startup.  
INFO:     Application startup complete.  
INFO: 127.0.0.1:44682 - "GET /zona-restringida HTTP/1.1" 404 Not F  
ound
```

The terminal interface includes sections for "PROBLEMAS", "SALIDA", "CONSOLA DE DEPURACIÓN", "TERMINAL", and "PUERTOS". On the right side, there is a "CHAT" section with examples for curl commands:

1. Sin token (debe dar error 403):
curl -X GET "http://localhost:8000/zona-restringida"
2. Con token inválido (debe dar error 403):
curl -X GET "http://localhost:8000/zona-restringida"
3. Con token válido (debe dar acceso):
curl -X GET "http://localhost:8000/zona-restringida"

The browser's address bar shows the URL: "orange-xiphophone-r4wgvgvq47r2xq69.github.dev". The left sidebar of the browser shows the project structure under "CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]".

Screenshot of a browser window showing a GitHub Codespace for a 'ConstructionCompany' repository. The URL is `orange-xylophone-r4wgwgvq47r2xq89.github.dev`. The browser tabs include `main.py - ConstructionCompany`, `_init_.py - ConstructionCompany`, `word - Yahoo Search Tus resultados`, and `Pruebas admin`.

The browser toolbar includes icons for back, forward, search, and various extensions.

The main content area shows the file structure of the repository:

```

CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]
  src
    backend
      >_pycache_
    app
      >_pycache_
    api
      >_pycache_
      > database
    endpoints
      >_pycache_
      >_init_.py
      admin.py
      auth.py
      clients.py
      dashboard.py
      inventory.py
      payments.py
      projects.py
      purchase.py

```

A tooltip for `_init_.py` reads:

src > backend > app > api > endpoints > `_init_.py`
 1 Abrir el chat (Ctrl+I), o seleccione un idioma (Ctrl+K M), o rellene con una plantilla para empezar.
 Empiece a escribir para descartar o no mostrar esto de nuevo.

The right side of the interface shows a terminal window with the following text:

```

@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X GET "http://localhost:8000/zona-restringida" -H "Authorization: Bearer token_incorrecto"
{"detail":"Not Found"}@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $

```

Below the terminal is a code editor with the `_init_.py` file open. The code editor has sections for PROBLEMAS, SALIDA, CONSOLA DE DEPURACIÓN, TERMINAL, and PUERTOS.

The bottom of the screen shows the Windows taskbar with various pinned icons and system status information.

7.2 Pruebas auth.py

The screenshot shows a browser window with several tabs open. The active tab is 'auth.py - ConstructionCompany'. The address bar shows the URL 'orange-xiphophone-r4wgvgvq47r2xq69.github.dev'. The main content area is a code editor for a Python project named 'ConstructionCompany'. The 'auth.py' file is open, containing the following code:

```

from src.backend.app.api.models.auth import Token
from typing import Optional

router = APIRouter(tags=["autenticacion"])

oauth2_scheme = OAuth2PasswordBearer(tokenUrl="/auth/token")

@router.post("/token", response_model=Token)
async def login_for_access_token(form_data: OAuth2PasswordRequestForm):
    if not form_data.username or not form_data.password:
        raise HTTPException(
            status_code=400,
            detail="Usuario y contraseña requeridos",
            headers={"WWW-Authenticate": "Bearer"},
)

```

Below the code editor is a terminal window showing the command being run:

```

@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X PO
ST "http://localhost:8000/auth/token" \
-H "Content-Type: application/x-www-form-urlencoded" \
-d "username=admin&password=1234"
{"access_token":"token_valido","token_type":"bearer"}@Gabrielleon0215
→ /workspaces/ConstructionCompany (main) $

```

To the right of the code editor, there is a sidebar with the following information:

- A note: "Aquí tienes el comando para probar el endpoint de autenticación /auth/token desde la terminal usando curl:" followed by the curl command shown in the terminal.
- A JSON response example: {"access_token": "token_valido", "token_type": "bearer"}
- A note: "Esto te devolverá un JSON con el token de acceso si las credenciales son correctas:"
- An "Edición de archivos en el área de trabajo" section with a file browser interface.

main.py - ConstructionCompany x auth.py - ConstructionCompany x word - Yahoo Search Tus resultados x Pruebas admin x + - 0 X

orange-xylophone-e4wgvvgvq472xq69.github.dev

Nueva pestaña6ffd... Aplicaciones eliminadas Ver Pacific Rim 2: In... Sistema de Atención product image Nueva pestaña Aprende Informática All Bookmarks

ConstructionCompany [Codespaces: orange-xylophone]

EXPLORADOR [Vista previa] README.md

CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]

- src
 - backend
 - app
 - api
- endpoints
 - _pycache_
 - _init_.py
 - admin.py
- auth.py
- clients.py
- dashboard.py
- inventory.py
- payments.py
- projects.py
- purchase.py
- quotes.py
- tasks.py
- teams.py
- users.py

ESQUEMA

LÍNEA DE TIEMPO

DEPENDENCIAS

[src > backend > app > api > endpoints > auth.py] ...

```

3 from src.backend.app.api.models.auth import Token
4 from typing import Optional
5
6 router = APIRouter(tags=["autenticacion"])
7
8 oauth2_scheme = OAuth2PasswordBearer(tokenUrl="/auth/token")
9
10 @router.post("/token", response_model=Token)
11 async def login_for_access_token(form_data: OAuth2PasswordRequestForm):
12     if not form_data.username or not form_data.password:
13         raise HTTPException(
14             status_code=400,
15             detail="Usuario y contraseña requeridos",
16             headers={"WWW-Authenticate": "Bearer"},
17         )

```

PROBLEMAS 90 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 7

* 'orm_mode' has been renamed to 'from_attributes'
warnings.warn(message, UserWarning)
INFO: Started server process [27418]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:44682 - "GET /zona-restringida HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:37516 - "POST /auth/token HTTP/1.1" 200 OK

+ ... ^ X Pyth... bash

{ "access_token": "token_valido", "token_type": "bearer" } Agregar contexto... auth.py Actual archivo Edición de archivos en el área de trabajo el Agent v GPT-4.1 v 108 a.m. 5/06/2025

Codespaces: orange-xylophone ConstructionCompany main* 489 ▲ 11 7 FastAPI (ConstructionCompany) UTF-8 LF Python 3.10.12 64-bit Linux (linux-x64) Diseño: Latin American

Buscar

7.3 Pruebas clients.py

The screenshot shows a browser window with several tabs open. The active tab is 'clients.py - ConstructionCompany'.

Left Sidebar: Shows the project structure under 'EXPLORADOR'. The 'clients.py' file is selected in the 'endpoints' folder.

Code Editor: Displays the content of the 'clients.py' file:

```

src > backend > app > api > endpoints > clients.py > ...
1  from fastapi import APIRouter
2  from typing import List
3  from uuid import uuid4
4  from src.backend.app.api.models.clients import Cliente # Importa el m...
5
6  router = APIRouter()
7
8  clientes_db: List[Cliente] = []
9
10 @router.post("/", status_code=201, response_model=Cliente)
11 def crear_cliente(cliente: Cliente):
12     cliente.id = str(uuid4())
13     clientes_db.append(cliente)
14     return cliente

```

Terminal: Shows a terminal session with a command to test the endpoint:

```

@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X POST "http://localhost:8000/clientes/" \
-H "Content-Type: application/json" \
-d '{"nombre": "Empresa S.A.", "email": "contacto@empresa.com"}'
{"detail":[{"type": "missing", "loc": ["body", "documento"], "msg": "Field required", "input": {"nombre": "Empresa S.A.", "email": "contacto@empresa.com"}, "url": "https://errors.pydantic.dev/2.5/v/missing"}]}@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ 

```

Right Side: Includes a 'CHAT' section with a message about creating clients, a 'curl' command example, and a JSON configuration tool. It also has sections for 'PROBLEMAS', 'SALIDA', 'CONSOLA DE DEPURACIÓN', and 'PUERTOS'.

The screenshot shows a code editor interface with the following details:

- Code Editor Area:** Displays the file `clients.py` under the `src/backend/app/api/endpoints` directory. The code implements a POST endpoint for creating a client.
- Terminal Log:** Shows application startup logs and API requests. Key logs include:
 - INFO: Started server process [27418]
 - INFO: Waiting for application startup.
 - INFO: Application startup complete.
 - INFO: 127.0.0.1:44682 - "GET /zona-restringida HTTP/1.1" 404 Not Found
 - INFO: 127.0.0.1:37516 - "POST /auth/token HTTP/1.1" 200 OK
 - INFO: 127.0.0.1:59648 - "POST /clientes/ HTTP/1.1" 422 Unprocessable Entity
- Code Completion and Hover Info:** A tooltip provides instructions for creating a client using curl, mentioning the command `curl -X POST "http://localhost:44682/clientes/" -H "Content-Type: application/json" -d '{"nombre": "Empresa S.A."}`.
- File Explorer:** Shows the project structure with files like `README.md`, `clients.py`, and various models and endpoints files.
- Bottom Status Bar:** Includes tabs for `main*`, `clients.py`, `FastAPI (ConstructionCompany)`, and `Python`. It also shows system information like `ESP LAA`, `11:11 a.m.`, and `5/06/2025`.

7.4 Pruebas dashboard.py

The screenshot shows a browser window with several tabs open. The active tab is 'ConstructionCompany [Codespaces: orange xylophone]'. The page content is a code editor for 'dashboard.py' within a 'FastAPI' application structure. The code checks for a valid token in the Authorization header. If valid, it returns a welcome message; if invalid or missing, it raises a 401 HTTPException.

```

from fastapi import APIRouter, HTTPException, Header
router = APIRouter()

def verificar_token(authorization: str = Header(None)):
    if not authorization or not authorization.startswith("Bearer "):
        raise HTTPException(status_code=401, detail="Token inválido")
    token = authorization.split(" ")[1]
    if token != "token_valido":
        raise HTTPException(status_code=401, detail="Token inválido")

@router.get("/", summary="Dashboard", description="Vista general del sistema")
def dashboard(authorization: str = Header(None)):
    verificar_token(authorization)
    return {"mensaje": "Bienvenido al dashboard"}

```

A sidebar on the right contains instructions for testing the endpoint:

- 1. Con token válido (debe dar acceso):** curl -X GET "http://localhost:8000"
- 2. Sin token o con token inválido (debe dar error 401):** curl -X GET "http://localhost:8000"

The bottom of the screen shows the Linux desktop environment with various icons and system status indicators.

orange-xylophone-r4wgvgvq47r2xq69.github.dev

ConstructionCompany [Codespaces: orange-xylophone]

EXPLORADOR

- CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]
 - src
 - backend
 - app
 - api
 - endpoints
 - _private
 - __init__.py
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
- modalc
- ESQUEMA
- LÍNEA DE TIEMPO
- DEPENDENCIAS

[Vista previa] README.md

```
src > backend > app > api > endpoints > dashboard.py > ...
1  from fastapi import APIRouter, HTTPException, Header
2
3  router = APIRouter()
4
5  def verificar_token(authorization: str = Header(None)):
6      if not authorization or not authorization.startswith("Bearer "):
7          raise HTTPException(status_code=401, detail="Token inválido")
8      token = authorization.split(" ")[1]
9      if token != "token_valido":
10         raise HTTPException(status_code=401, detail="Token inválido")
11
12 @router.get("/", summary="Dashboard", description="Vista general del sistema")
13 def dashboard(authorization: str = Header(None)):
14     verificar_token(authorization)
15     return {"mensaje": "Bienvenido al dashboard"}
```

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 7

INFO: Waiting for application startup.
 INFO: Application startup complete.
 INFO: 127.0.0.1:44682 - "GET /zona-restringida HTTP/1.1" 404 Not Found
 INFO: 127.0.0.1:37516 - "POST /auth/token HTTP/1.1" 200 OK
 INFO: 127.0.0.1:59648 - "POST /clientes/ HTTP/1.1" 422 Unprocessable Entity
 INFO: 127.0.0.1:45668 - "GET / HTTP/1.1" 200 OK

AQUÍ TIENES EL COMANDO PARA PROBAR EL ENDPOINT PROTEGIDO DEL DASHBOARD (GET /) USANDO CURL DESDE LA TERMINAL. RECUERDA QUE DEBES ENVIAR EL HEADER AUTHORIZATION CON EL TOKEN CORRECTO:

1. CON TOKEN VÁLIDO (DEBE DAR ACCESO):
`curl -X GET "http://localhost:8000"`
2. SIN TOKEN O CON TOKEN INVÁLIDO (DEBE DAR ERROR 401):
`curl -X GET "http://localhost:8000"`

Pyth... bash bash bash Agregar contexto... dashboard.py Actual archivo Edición de archivos en el área de trabajo Agent GPT-4.1 1:15 a.m. 5/06/2025

Codestar: orange-xylophone ConstructionCompany main* 489 ▲ 11 7 FastAPI (ConstructionCompany) UTF-8 LF Python 3.10.12 64-bit Linux (linux-x64) Diseño: Latin American 1:15 a.m. 5/06/2025

7.5 Pruebas inventory.py

The screenshot shows a browser-based code editor interface. The title bar indicates the current tab is 'inventory.py - ConstructionCompany'. The left sidebar shows the project structure under 'CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]' with 'src' as the root folder containing 'backend', 'app', and 'api' subfolders. The 'api' folder contains 'endpoints' which includes 'inventory.py'. The main editor area displays the following Python code:

```

from fastapi import APIRouter
from (module) src .list
from src.backend.app.api.models.inventory import Inventory # Importa
router = APIRouter()
inventarios_db: List[Inventory] = []
@router.post("/registro")
def registrar_inventario(data: Inventory):
    inventarios_db.append(data)
    return {"mensaje": "Inventario registrado", "materiales": data.mat}

```

Below the code editor is a terminal window showing the output of a curl command:

```

@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X POST "http://localhost:8000/inventario/registro" \
-H "Content-Type: application/json" \
-d '{"materiales": ["cemento", "arena", "grava"]}' \
{"detail": "Not Found"}@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ 

```

The status bar at the bottom of the screen shows the following information: 'Codespaces: orange xylophone', 'ConstructionCompany', 'main*', '489 ▲ 11', 'FastAPI (ConstructionCompany)', 'UTF-8 LF', '(Python', '3.10.12 64-bit', 'Linux (linux-x64)', 'Diseño: Latin American', 'Buscar', 'ESP LAA', '1:22 a.m.', '5/06/2025'.

The screenshot shows a browser window with several tabs open:

- main.py - ConstructionCompany
- inventory.py - ConstructionCompany
- word - Yahoo Search Tus resultados
- Pruebas admin

The main content area displays a code editor for a file named `inventory.py` under the `src/backend/app/api/endpoints` directory. The code implements a FastAPI endpoint for registering inventory items:

```

from fastapi import APIRouter
from typing import List
from src.backend.app.api.models.inventory import Inventory # Importa el modelo de inventario
router = APIRouter()
inventarios_db: List[Inventory] = []
@router.post("/registro")
def registrar_inventario(data: Inventory):
    inventarios_db.append(data)
    return {"mensaje": "Inventario registrado", "materiales": data.materiales}

```

Below the code editor is a terminal window showing log messages from a running FastAPI application:

```

INFO: 127.0.0.1:44682 - "GET /zona-restringida HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:37516 - "POST /auth/token HTTP/1.1" 200 OK
INFO: 127.0.0.1:59648 - "POST /clientes/ HTTP/1.1" 422 Unprocessable Entity
INFO: 127.0.0.1:45668 - "GET / HTTP/1.1" 200 OK
INFO: 127.0.0.1:37880 - "POST /inventario/registro HTTP/1.1" 404 Not Found

```

The bottom of the screen shows the Windows taskbar with various pinned icons and system status information.

7.6 Pruebas payments.py

The screenshot shows a GitHub Codespace interface for a project named 'ConstructionCompany'. The left sidebar lists files in the 'src' directory, including 'payments.py' which is currently selected. The main area displays the contents of 'payments.py':

```

from typing import List
from uuid import uuid4
from src.backend.app.api.models.payments import Pago
router = APIRouter(tags=["pagos"])
pagos_db: List[Pago] = []
@router.post("/", response_model=Pago, status_code=201)
def registrar_pago(pago: Pago):
    pago.id = str(uuid4())
    pagos_db.append(pago)
    return pago
@router.get("/", response_model=List[Pago])

```

Below the code editor is a terminal window showing a failed curl command:

```

@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X POST "http://localhost:8000/pagos/" \
-H "Content-Type: application/json" \
-d '{"monto": 1000, "fecha": "2025-06-05", "metodo": "transferencia"}'
curl: (7) Failed to connect to localhost port 8000 after 0 ms: Connection refused
@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $

```

The screenshot shows a GitHub Codespace interface for a project named "ConstructionCompany".

Terminal:

```
@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X GET "http://localhost:8000/pagos/"  
curl: (7) Failed to connect to localhost port 8000 after 0 ms: Connection refused
```

Code Editor:

```
from typing import List  
from uuid import uuid4  
from src.backend.app.api.models.payments import Pago  
  
router = APIRouter(tags=["pagos"])  
  
pagos_db: List[Pago] = []  
  
@router.post("/", response_model=Pago, status_code=201)  
def registrar_pago(pago: Pago):  
    pago.id = str(uuid4())  
    pagos_db.append(pago)  
    return pago  
  
@router.get("/", response_model=List[Pago])
```

Right Panel:

- CHAT:** Shows an error message about failing to connect to localhost port 8000.
- Terminal:** Shows the command `uvicorn src.backend.app.api.main:app` being run.
- File Explorer:** Shows the project structure with files like README.md, README.md, payments.py, and various API endpoints.
- Bottom Status Bar:** Shows the file path as "ConstructionCompany/main", file size as "1.1M", and a note about using FastAPI.

Explorador

CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]

- src
- backend
- app
- api
 - endpoints
 - _pycache_
 - __init__.py
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py**
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - models

Vista previa] README.md

```
src > backend > app > api > endpoints > payments.py > listar_pagos
1  from typing import List
2  from uuid import uuid4
3  from src.backend.app.api.models.payments import Pago
4
5  router = APIRouter(tags=["pagos"])
6
7  pagos_db: List[Pago] = []
8
9  @router.post("/", response_model=Pago, status_code=201)
10 def registrar_pago(pago: Pago):
11     pago.id = str(uuid4())
12     pagos_db.append(pago)
13     return pago
14
15 @router.get("/", response_model=List[Pago])
```

PROBLEMAS 801 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 15

```
@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X GET "http://localhost:8000/pagos/"
curl: (7) Failed to connect to localhost port 8000 after 0 ms: Connection refused
@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X GET "http://localhost:8000/pagos/"
curl: (7) Failed to connect to localhost port 8000 after 0 ms: Connection refused
@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $
```

CHAT

El error curl: (7) Failed to connect to localhost port 8000 after 0 ms: Connection refused significa que el servidor FastAPI no está corriendo o no está escuchando en el puerto 8000.

Para solucionarlo, debes iniciar el servidor FastAPI. Usa este comando desde la raíz de tu proyecto:

```
uvicorn src.backend.app.api.main:app --reload
```

Después de iniciar el servidor, vuelve a ejecutar tu comando curl para probar el endpoint. ¿Quieres que inicie el servidor por ti ahora?

Agregar contexto...

payments.py Actual archivo

Edición de archivos en el área de trabajo

Buscar

ESP LAA

148 a.m.

5/06/2025

The screenshot shows a browser window with several tabs open, including 'main.py - ConstructionCom', 'payments.py - Construction', 'word - Yahoo Search Tus res...', 'Pruebas admin', and 'payments.py - Construction'. The main content area is a code editor for a 'ConstructionCompany' project. The left sidebar shows a file tree for 'CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]' with a 'payments.py' file selected. The right pane displays the contents of 'payments.py':

```
16 def listar_pagos():
```

The terminal tab at the bottom shows a command prompt: '@Gabrielleon0215 → /workspaces/ConstructionCompany (main) \$'. A context menu is open over the terminal, with options like 'python3', 'bash', 'Pago', 'Agregar contexto...', 'payments.py Actual archivo', and 'Edición de archivos en el área de trabajo'.

7.7 Pruebas projects.py

The screenshot shows a browser window with several tabs open. The active tab is 'projects.py - ConstructionCompany'. The page content is a code editor showing Python code for a FastAPI endpoint to update or delete a project. To the right of the code, there is a sidebar with instructions for creating a project using curl. Below the code editor, there is a terminal window showing some command-line output. The bottom of the screen has a status bar with various system and application icons.

```

src > backend > app > api > endpoints > projects.py > ...
29 def actualizar_proyecto(proyecto_id: str, datos: Proyecto):
30     proyectos_db[1] = datos
31     return proyectos_db[1]
32     raise HTTPException(status_code=404, detail="Proyecto no encontrado")
33
34 @router.delete("/{proyecto_id}", status_code=204, summary="Eliminar proyecto")
35 def eliminar_proyecto(proyecto_id: str):
36     for i, proyecto in enumerate(proyectos_db):
37         if proyecto.id == proyecto_id:
38             proyectos_db.pop(i)
39             return
40     raise HTTPException(status_code=404, detail="Proyecto no encontrado")

```

1. Crear un proyecto (POST):

```
curl -X POST "http://localhost:8000/api/projects" -H "Content-Type: application/json" -d '{"nombre": "Edificio Central"}'
```

Ajusta los campos del JSON según los atributos requeridos por tu modelo

Proyecto .

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

Edición de archivos en el área de trabajo

1 archivo cambiado payments.py src/backend/app/api/en...

Agregar contexto...

projects.py Actual archivo

ESP LAA 2:01 a.m. 5/06/2025

The screenshot shows a browser window with three tabs open:

- main.py - ConstructionCompany
- word - Yahoo Search Tus resulta...
- Pruebas admin

The main content area displays the code for `projects.py` in a code editor:

```

src > backend > app > api > endpoints > projects.py > ...
29 def actualizar_proyecto(proyecto_id: str, datos: Proyecto):
30     proyectos_db[1] = datos
31     return proyectos_db[1]
32     raise HTTPException(status_code=404, detail="Proyecto no encontrado")
33
34 @router.delete("/{proyecto_id}", status_code=204, summary="Eliminar proyecto")
35 def eliminar_proyecto(proyecto_id: str):
36     for i, proyecto in enumerate(proyectos_db):
37         if proyecto.id == proyecto_id:
38             proyectos_db.pop(i)
39             return
40     raise HTTPException(status_code=404, detail="Proyecto no encontrado")

```

The sidebar on the left shows the project structure:

- EXPLORADOR**
 - CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]
 - src
 - backend
 - app
 - api
 - endpoints
 - /__pycache__
 - __init__.py
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - models
 - ESQUEMA
 - LÍNEA DE TIEMPO
 - DEPENDENCIAS

The bottom status bar shows:

 - Codestar: Codespaces: orange xylophone
 - File: main* (modified)
 - Line: 489 Col: 11
 - Port: 16
 - Server: FastAPI (ConstructionCompany)
 - Encoding: UTF-8
 - Line Endings: LF
 - Language: Python
 - Version: 3.10.12 64-bit
 - Platform: Linux (linux-x64)
 - Design: Latin American
 - Time: 201 a.m.
 - Date: 5/06/2025

The screenshot shows a browser-based code editor interface for a Python project named 'ConstructionCompany'.

Project Structure:

- src
- backend
- app
- api
- endpoints
- __pycache__
- __init__.py
- admin.py
- auth.py
- clients.py
- dashboard.py
- inventory.py
- payments.py
- projects.py
- purchase.py
- quotes.py
- tasks.py
- teams.py
- users.py
- models

Code Editor:

```
def actualizar_proyecto(proyecto_id: str, datos: Proyecto):
    proyectos_db[1] = datos
    return proyectos_db[1]
    raise HTTPException(status_code=404, detail="Proyecto no encontrado")

@router.delete("/{proyecto_id}", status_code=204, summary="Eliminar proyecto")
def eliminar_proyecto(proyecto_id: str):
    for i, proyecto in enumerate(proyectos_db):
        if proyecto.id == proyecto_id:
            proyectos_db.pop(i)
            return
    raise HTTPException(status_code=404, detail="Proyecto no encontrado")
```

Terminal:

```
@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X GET "http://localhost:8000/proyectos/"
```

Right Panel:

- CHAT: -d '{"nombre": "Edificio Central"}'
- Ajusta los campos del JSON según los atributos requeridos por tu modelo.
- 2. Listar todos los proyectos (GET): curl -X GET "http://localhost:8000/proyectos/"
- 3. Obtener un proyecto por ID (GET): curl -X GET "http://localhost:8000/proyectos/1"
- Edición de archivos en el área de trabajo
- Actual archivo: projects.py
- Archivos cambiados: payments.py
- Opciones: Agregar contexto..., Copilot, Agent, GPT-4.1

Bottom Bar:

- Buscar
- Buscador
- Iconos para Archivos, Terminal, Chat, etc.
- Estado: ESP, LAA, 201 a.m., 5/06/2025

The screenshot shows a browser window with multiple tabs open. The active tab is titled 'ConstructionCompany [Codespaces: orange xylophone]' and displays a code editor with Python code for a FastAPI application. The code defines endpoints for updating and deleting projects.

```
def actualizar_proyecto(proyecto_id: str, datos: Proyecto):
    proyectos_db[1] = datos
    return proyectos_db[1]
    raise HTTPException(status_code=404, detail="Proyecto no encontrado")

@router.delete("/{proyecto_id}", status_code=204, summary="Eliminar proyecto")
def eliminar_proyecto(proyecto_id: str):
    for i, proyecto in enumerate(proyectos_db):
        if proyecto.id == proyecto_id:
            proyectos_db.pop(i)
            return
    raise HTTPException(status_code=404, detail="Proyecto no encontrado")
```

The code editor interface includes a sidebar for file navigation, a terminal at the bottom, and a right-hand panel for running curl commands and viewing logs.

main.py - ConstructionCompany | word - Yahoo Search Tus result... | Pruebas admin | projects.py - ConstructionCompany | +

orange-xylophone-r4wgwvgvq47r2xqf69.github.dev

Nueva pestaña63ffd... Aplicaciones eliminadas Ver Pacific Rim 2: In... Sistema de Atención... product image Nueva pestaña Aprende Informática All Bookmarks

ConstructionCompany [Codespaces: orange xylophone]

EXPLORADOR

CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]

- src
 - backend
 - app
 - api
 - endpoints
 - _pycache_
 - _init_.py
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - models
- ESQUEMA
- LÍNEA DE TIEMPO
- DEPENDENCIAS

[Vista previa] README.md

src > backend > app > api > endpoints > projects.py > ...

```

1 from fastapi import APIRouter, HTTPException
2 from typing import List
3 from uuid import uuid4
4 from src.backend.app.api.models.projects import Proyecto
5
6 router = APIRouter()
7
8 # Base de datos en memoria para proyectos
9 proyectos_db: List[Proyecto] = []
10
11 @router.post("/", response_model=Proyecto, status_code=201, summary="C")
12 def crear_proyecto(proyecto: Proyecto):
13     proyecto.id = str(uuid4())
14     proyectos_db.append(proyecto)
15     return proyecto

```

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

-d '{"nombre": "Edificio Central Actualizado", "descripcion": "Actualización de datos"}'

{"detail":[{"type":"missing","loc":["body","estado"],"msg":"Field required","input":{"nombre":"Edificio Central Actualizado","descripcion":"Actualización de datos"}}, {"url":"https://errors.pydantic.dev/2.5/v/missing"}, {"type":"missing","loc":["body","fecha_inicio"],"msg":"Field required","input":{"nombre":"Edificio Central Actualizado","descripcion":"Actualización de datos"}}, {"url":"https://errors.pydantic.dev/2.5/v/missing"}]}])@Gabrielleon0215 → /workspaces/ConstructionCompany (main) \$

1 archivo cambiado

payments.py src/backend/app/api/en...

Agregar contexto...

projects.py Actual archivo

Edición de archivos en el área de trabajo el

Agent GPT-4.1 Agent GPT-4.1

Buscar

ESP LAA 2:04 a.m. 5/06/2025

orange-xiphophone-r4wgvgvq47r2xq69.github.dev

src > backend > app > api > endpoints > projects.py > ...

```

1  from fastapi import APIRouter, HTTPException
2  from typing import List
3  from uuid import uuid4
4  from src.backend.app.api.models.projects import Proyecto
5
6  router = APIRouter()
7
8  # Base de datos en memoria para proyectos
9  proyectos_db: List[Proyecto] = []
10
11 @router.post("/", response_model=Proyecto, status_code=201, summary="Crear un proyecto")
12 def crear_proyecto(proyecto: Proyecto):
13     proyecto.id = str(uuid4())
14     proyectos_db.append(proyecto)
15     return proyecto

```

curl -X GET "http://localhost:8000/projects"

Reemplaza <ID_DEL_PROYECTO> por el id real de un proyecto.

4. Actualizar un proyecto (PUT):

```
curl -X PUT "http://localhost:8000/projects/1" -H "Content-Type: application/json" -d '{"nombre": "Edificio Central"}'
```

5. Eliminar un proyecto (DELETE):

```
curl -X DELETE "http://localhost:8000/projects/1"
```

PROBLEMAS 300 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

INFO: Application startup complete.
INFO: 127.0.0.1:39732 - "POST /proyectos/ HTTP/1.1" 422 Unprocessable Entity
INFO: 127.0.0.1:33958 - "GET /proyectos/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:52684 - "GET /proyectos/%3CID_DEL_PROYECTO%3E HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:43272 - "PUT /proyectos/%3CID_DEL_PROYECTO%3E HTTP/1.1" 422 Unprocessable Entity

Pyth... bash bash Copilot bash bash Agent GPT-4.1

Buscar

ESP LAA 2:04 a.m. 5/06/2025

orange-xylophone-r4wgvvgvq47r2xq69.github.dev

Nueva pestaña65tfdf... Aplicaciones eliminadas Ver Pacific Rim 2: In Sistema de Atención product image Nueva pestaña Aprende Informática All Bookmarks

ConstructionCompany[Codespaces: orange.xylophone]

EXPLORADOR

- CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]
 - src
 - backend
 - app
 - api
 - endpoints
 - _pycache_
 - __init__.py
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - models
- ESQUEMA
- LÍNEA DE TIEMPO
- DEPENDENCIAS

[Vista previa] README.md

src > backend > app > api > endpoints > projects.py > ...

```

29 def actualizar_proyecto(proyecto_id: str, datos: Proyecto):
30     proyectos_db[1] = datos
31     return proyectos_db[1]
32     raise HTTPException(status_code=404, detail="Proyecto no encontrado")
33
34 @router.delete("/{proyecto_id}", status_code=204, summary="Eliminar proyecto")
35 def eliminar_proyecto(proyecto_id: str):
36     for i, proyecto in enumerate(proyectos_db):
37         if proyecto.id == proyecto_id:
38             proyectos_db.pop(i)
39             return
40     raise HTTPException(status_code=404, detail="Proyecto no encontrado")

```

CHAT

curl -X GET "http://localhost:8000/projects/1"

3. Obtener un proyecto por ID (GET):

curl -X GET "http://localhost:8000/projects/1"

Reemplaza <ID_DEL_PROYECTO> por el id real de un proyecto.

4. Actualizar un proyecto (PUT):

curl -X PUT "http://localhost:8000/projects/1"

1 archivo cambiado payments.py src/backend/app/api/endpoints/projects.py Agregar contexto... projects.py Actualizar archivo Edición de archivos en el área de trabajo Agent GPT-4.1 2:03 a.m. 5/06/2025

PROBLEMAS 300 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

INFO: Started server process [69529]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:39732 - "POST /proyectos/ HTTP/1.1" 422 Unprocessable Entity
INFO: 127.0.0.1:33958 - "GET /proyectos/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:52684 - "GET /proyectos/%3CID_DEL_PROYECTO%3E HTTP/1.1" 404 Not Found

Buscar

main.py - ConstructionCompany x word - Yahoo Search Tus resultados x Pruebas admin x projects.py - ConstructionCompany x +

UTF-8 LF Python 3.10.12 64-bit Linux (Linux-x64) Diseño: Latin American

ESP LAA 2:03 a.m. 5/06/2025

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows the project structure under "CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]" with files like README.md, src, backend, app, api, endpoints, __pycache__, _init_.py, admin.py, auth.py, clients.py, dashboard.py, inventory.py, payments.py, projects.py, purchase.py, quotes.py, tasks.py, teams.py, users.py, and models.
- Code Editor:** Displays the content of `projects.py`. The code defines an API endpoint for creating a project using FastAPI's `APIRouter` and `HTTPException`.
- Terminal:** Shows a terminal session where a `curl` command is used to delete a project from the local host. The response indicates that the project was successfully deleted.
- Right Panel:** Includes sections for "CHAT", "CURSOS", and "CURSOS DE APRENDIZAJE". It also shows a preview of the README.md file and some help text about deleting a project.
- Bottom Bar:** Shows the workspace name ("Codespaces: orange xylophone"), the current file ("ConstructionCompany/main*"), file statistics (489 lines, 11 changes, 16 tabs), the Python environment ("FastAPI (ConstructionCompany)"), encoding ("UTF-8 LF"), Python version ("3.10.12 64-bit"), operating system ("Linux (linux-x64)"), and design ("Diseño: Latin American").

The screenshot shows a browser-based development environment for a FastAPI application named "ConstructionCompany".

Project Structure:

- src > backend > app > api > endpoints > projects.py
- src > backend > app > api > endpoints > __init__.py
- src > backend > app > api > endpoints > admin.py
- src > backend > app > api > endpoints > auth.py
- src > backend > app > api > endpoints > clients.py
- src > backend > app > api > endpoints > dashboard.py
- src > backend > app > api > endpoints > inventory.py
- src > backend > app > api > endpoints > payments.py
- src > backend > app > api > endpoints > projects.py
- src > backend > app > api > endpoints > purchase.py
- src > backend > app > api > endpoints > quotes.py
- src > backend > app > api > endpoints > tasks.py
- src > backend > app > api > endpoints > teams.py
- src > backend > app > api > endpoints > users.py
- src > backend > app > models

Code Editor (projects.py):

```

from fastapi import APIRouter, HTTPException
from typing import List
from uuid import uuid4
from src.backend.app.api.models.projects import Proyecto

router = APIRouter()

# Base de datos en memoria para proyectos
proyectos_db: List[Proyecto] = []

@router.post("/", response_model=Proyecto, status_code=201, summary="Crear un proyecto")
def crear_proyecto(proyecto: Proyecto):
    proyecto.id = str(uuid4())
    proyectos_db.append(proyecto)
    return proyecto

```

Terminal (PROBLEMAS):

```

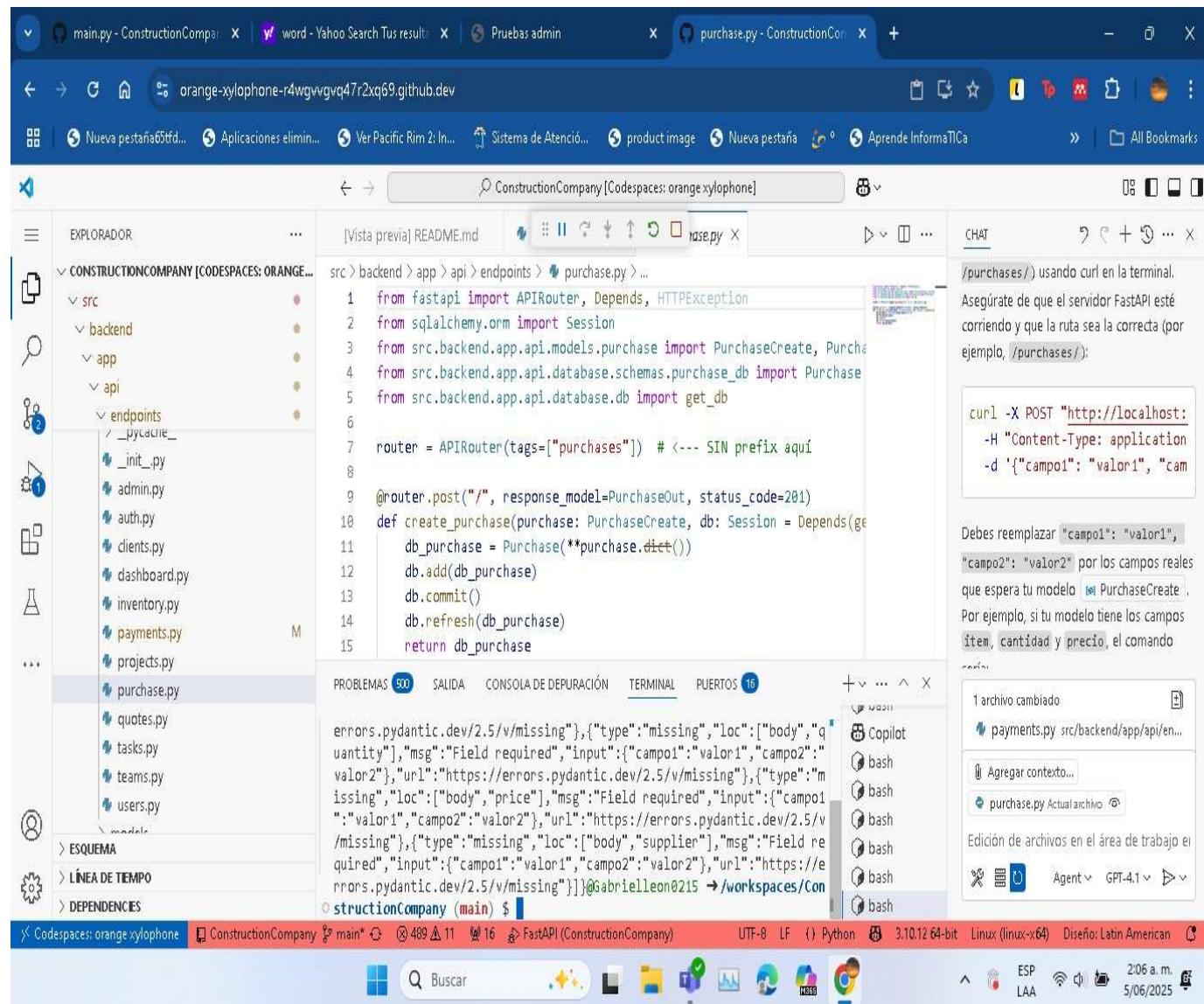
INFO: 127.0.0.1:33958 - "GET /proyectos/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:52684 - "GET /proyectos/%3CID_DEL_PROYECTO%3E HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:43272 - "PUT /proyectos/%3CID_DEL_PROYECTO%3E HTTP/1.1" 422 Unprocessable Entity
INFO: 127.0.0.1:42194 - "DELETE /proyectos/%3CID_DEL_PROYECTO%3E HTTP/1.1" 404 Not Found

```

Right Sidebar:

- CHAT
- SEARCH
- CODE SEARCH
- HELP: ¿Necesitas ayuda para obtener el ID de un proyecto o para otro endpoint?
- COMMANDS: curl -X PUT "http://localhost:8000/proyectos/" -H "Content-Type: application/json" -d '{"nombre": "Edificio Centro"}'
- COMMANDS: curl -X DELETE "http://localhost:8000/proyectos/1"
- NOTIFICATION: 5. Eliminar un proyecto (DELETE):
- FILE LIST: 1 archivo cambiado (payments.py)
- FILE CONTEXT: Agregar contexto...
- FILE HISTORY: project.py Actual archivo
- EDITION: Edición de archivos en el área de trabajo
- AGENT: Agent GPT-4.1

7.8 Pruebas purchase.py



```
curl -X POST "http://localhost:8000/purchases" -H "Content-Type: application/json" -d '{"campo1": "valor1", "campo2": "valor2"}'
```

Debes reemplazar "campo1": "valor1", "campo2": "valor2" por los campos reales que espera tu modelo PurchaseCreate. Por ejemplo, si tu modelo tiene los campos item, cantidad y precio, el comando sería:

```
curl -X POST "http://localhost:8000/purchases" -H "Content-Type: application/json" -d '{"item": "Mesa", "cantidad": 10, "precio": 500}'
```

Screenshot of a browser window showing a code editor interface for a FastAPI application named "ConstructionCompany". The URL is "orange-xylophone-r4wgvgvq47r2xq69.github.dev". The code editor displays the file "purchase.py" which contains the following Python code:

```

from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy.orm import Session
from src.backend.app.api.models.purchase import PurchaseCreate, Purchase
from src.backend.app.api.database.schemas.purchase_db import Purchase
from src.backend.app.api.database.db import get_db

router = APIRouter(tags=["purchases"]) # --- SIN prefix aquí
def create_purchase(purchase: PurchaseCreate, db: Session = Depends(get_db)):
    db_purchase = Purchase(**purchase.dict())
    db.add(db_purchase)
    db.commit()
    db.refresh(db_purchase)
    return db_purchase

```

The terminal tab shows log messages indicating various HTTP requests and their responses. A tooltip provides instructions for testing the endpoint using curl.

Terminal Log:

```

INFO: 127.0.0.1:52684 - "GET /proyectos/%CID_DEL_PROYECTO% HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:43272 - "PUT /proyectos/%CID_DEL_PROYECTO% HTTP/1.1" 422 Unprocessable Entity
INFO: 127.0.0.1:42194 - "DELETE /proyectos/%CID_DEL_PROYECTO% HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:40780 - "POST /purchases/ HTTP/1.1" 422 Unprocessable Entity

```

Code Editor Contextual Help:

curl -X POST "http://localhost:/purchases/" -H "Content-Type: application/json" -d '{"campo1": "valor1", "campo2": "valor2"}'

Debes reemplazar "campo1": "valor1", "campo2": "valor2" por los campos reales que espera tu modelo PurchaseCreate. Por ejemplo, si tu modelo tiene los campos item, cantidad y precio, el comando sería:

curl -X POST "http://localhost:/purchases/" -H "Content-Type: application/json" -d '{"item": "madera", "cantidad": 10, "precio": 50}'

Code Editor Bottom Bar:

- Archivo cambiado
- payments.py src/backend/app/api/en...
- Agregar contexto...
- purchase.py Actual archivo
- Edición de archivos en el área de trabajo
- Agent GPT-4.1

7.9 Pruebas quotes.py

The screenshot shows a browser window with several tabs open, including 'main.py - ConstructionCompany', 'word - Yahoo Search Tus resultados', 'Pruebas admin', and 'quotes.py - ConstructionCompany'. The main content area displays the code for the 'quotes.py' file within a FastAPI application structure. The terminal at the bottom shows the output of a curl command being run against the local host to test the '/quotes' endpoint.

```

from fastapi import APIRouter, HTTPException
from typing import List
from uuid import uuid4
from src.backend.app.api.models.quotes import Cotizacion

router = APIRouter()

# Base de datos en memoria para cotizaciones
cotizaciones_db: List[Cotizacion] = []

@router.post("/", response_model=Cotizacion, status_code=201, summary="Crear una cotización")
def crear_cotizacion(cotizacion: Cotizacion):
    cotizacion.id = str(uuid4())
    cotizaciones_db.append(cotizacion)
    return cotizacion

```

CHAT

FastAPI está corriendo y que la ruta sea la correcta (por ejemplo, /quotes/):

1. Crear una cotización (POST):

```
curl -X POST "http://localhost:8000/quotes/" \
-H "Content-Type: application/json" \
-d '{"cliente": "Empresa S.A.", "total": 15000}'
```

Ajusta los campos del JSON según los atributos requeridos por tu modelo [Cotizacion](#).

2. Listar todas las cotizaciones (GET):

Edición de archivos en el área de trabajo en [payments.py](#)

Archivo cambiado [payments.py](#) src/backend/app/api/en...

Agregar contexto... [quotes.py](#) Actual archivo

Agent GPT-4.1 2:08 a.m. 5/06/2025

main.py - ConstructionCompany | word - Yahoo Search Tus resultados | Pruebas admin | quotes.py - ConstructionCompany | +

orange-xylophone-r4wgwvq47r2xq69.github.dev

Nueva pestaña 65ffd... Aplicaciones eliminadas Ver Pacific Rim 2: In... Sistema de Atención product image Nueva pestaña Aprende Informática » All Bookmarks

ConstructionCompany [Codespaces: orange xylophone]

EXPLORADOR

- CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]
 - src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - models
 - utils
- ESQUEMA
- LÍNEA DE TIEMPO
- DEPENDENCIAS

[Vista previa] README.md

```
src > backend > app > api > endpoints > quotes.py > ...
1  from fastapi import APIRouter, HTTPException
2  from typing import List
3  from uuid import uuid4
4  from src.backend.app.api.models.quotes import Cotizacion
5
6  router = APIRouter()
7
8  # Base de datos en memoria para cotizaciones
9  cotizaciones_db: List[Cotizacion] = []
10
11 @router.post("/", response_model=Cotizacion, status_code=201, summary="Crear una cotización")
12 def crear_cotizacion(cotizacion: Cotizacion):
13     cotizacion.id = str(uuid4())
14     cotizaciones_db.append(cotizacion)
15     return cotizacion
```

PROBLEMAS 900 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

```
/1.1" 404 Not Found
INFO: 127.0.0.1:43272 - "PUT /proyectos/%CID_DEL_PROYECTO%3E HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:42194 - "DELETE /proyectos/%CID_DEL_PROYECTO%3E HTTP/1.1" 422 Unprocessable Entity
INFO: 127.0.0.1:40780 - "POST /purchases/" 422 Unprocessable Entity
INFO: 127.0.0.1:39856 - "POST /quotes/" 404 Not Found
```

+ ... ^ X

bash bash Pyth... bash bash Agregar contexto... quotes.py Actual archivo Copilot Agent GPT-41

1 archivo cambiado payments.py src/backend/app/api/en...

Edición de archivos en el área de trabajo

Buscar

ESP LAA 208 a.m. 5/06/2025

ConstructionCompany main* 489▲ 11 16 FastAPI (ConstructionCompany) UTF-8 LF Python 3.10.12 64-bit Linux (linux-x64) Diseño: Latin American

orange-xiphophone-r4wgvgvq47r2xq69.github.dev

src > backend > app > api > endpoints > quotes.py

```

1 from fastapi import APIRouter, HTTPException
2 from typing import List
3 from uuid import uuid4
4 from src.backend.app.api.models.quotes import Cotizacion
5
6 outer = APIRouter()
7
8 # Base de datos en memoria para cotizaciones
9 cotizaciones_db: List[Cotizacion] = []
10
11 @router.post("/", response_model=Cotizacion, status_code=201, summary=
12 def crear_cotizacion(cotizacion: Cotizacion):
13     cotizacion.id = str(uuid4())
14     cotizaciones_db.append(cotizacion)
15     return cotizacion

```

PROBLEMAS 500 SALIDA CONSOLE DE DESARROLLO TERMINAL PUERTOS 16 Enfocar carpeta en el explorador (ctrl + clic)

@Gabrielleon0215 → /workspaces/ConstructionCompany (main) \$ curl -X GET "http://localhost:8000/quotes/" {"detail": "Not Found"}@Gabrielleon0215 → /workspaces/ConstructionCompany (main) \$

Buscar Buscar

Screenshot of a browser window showing a GitHub Codespace for a "ConstructionCompany" repository. The URL is `orange-xiphophone-r4wgwgvq47r2xq59.github.dev`. The page displays the `quotes.py` file content and terminal logs.

Code Content:

```

from fastapi import APIRouter, HTTPException
from typing import List
from uuid import uuid4
from src.backend.app.api.models.quotes import Cotizacion

router = APIRouter()

# Base de datos en memoria para cotizaciones
cotizaciones_db: List[Cotizacion] = []

@router.post("/", response_model=Cotizacion, status_code=201, summary="Crear una cotización")
def crear_cotizacion(cotizacion: Cotizacion):
    cotizacion.id = str(uuid4())
    cotizaciones_db.append(cotizacion)
    return cotizacion

```

Terminal Log:

```

INFO: 127.0.0.1:43272 - "PUT /proyectos/%3CID_DEL_PROYECTO%3E HTTP/1.1" 422 Unprocessable Entity
INFO: 127.0.0.1:42194 - "DELETE /proyectos/%3CID_DEL_PROYECTO%3E HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:40780 - "POST /purchases/ HTTP/1.1" 422 Unprocessable Entity
INFO: 127.0.0.1:39856 - "POST /quotes/ HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:48142 - "GET /quotes/ HTTP/1.1" 404 Not Found

```

Right Panel:

- CHAT:** Shows a message box with JSON input fields for creating a quote.
- 2. Listar todas las cotizaciones (GET):** curl command: `curl -X GET "http://localhost:8`
- 3. Obtener una cotización por ID (GET):** curl command: `curl -X GET "http://localhost:8`
- Edición de archivos:** Shows file navigation and context options for the current file (`quotes.py`).

The screenshot shows a browser-based code editor interface for a FastAPI project named "ConstructionCompany".

File Explorer:

- src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py

Code Editor:

```
from fastapi import APIRouter, HTTPException
from typing import List
from uuid import uuid4
from src.backend.app.api.models.quotes import Cotizacion

router = APIRouter()

# Base de datos en memoria para cotizaciones
cotizaciones_db: List[Cotizacion] = []

@router.post("/", response_model=Cotizacion, status_code=201, summary="Crear una cotización")
def crear_cotizacion(cotizacion: Cotizacion):
    cotizacion.id = str(uuid4())
    cotizaciones_db.append(cotizacion)
    return cotizacion
```

Terminal:

```
@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X GET "http://localhost:8000/quotes/<ID_DE_COTIZACION>" {"detail":"Not Found"}@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $
```

Right Panel:

 2. Listar todas las cotizaciones (GET):
curl -X GET "http://localhost:8000/quotes"
 3. Obtener una cotización por ID (GET):
curl -X GET "http://localhost:8000/quotes/1"

Reemplaza <ID_DE_COTIZACION> por el id real de una cotización.

Edición de archivos en el área de trabajo el 5/06/2025

Agent GPT-4.1

orange-xylophone-r4wgvgvq47r2xq69.github.dev

src backend > app > api > endpoints > quotes.py > ...

```

1 from fastapi import APIRouter, HTTPException
2 from typing import List
3 from uuid import uuid4
4 from src.backend.app.api.models.quotes import Cotizacion
5
6 outer = APIRouter()
7
8 # Base de datos en memoria para cotizaciones
9 cotizaciones_db: List[Cotizacion] = []
10
11 @router.post("/", response_model=Cotizacion, status_code=201, summary="")
12 def crear_cotizacion(cotizacion: Cotizacion):
13     cotizacion.id = str(uuid4())
14     cotizaciones_db.append(cotizacion)
15     return cotizacion

```

PROBLEMAS 90 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

```

@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X GET "http://localhost:8000/quotes/<ID_DE_COTIZACION>" {"detail": "Not Found"}@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $

```

1 archivo cambiado payments.py src/backend/app/api/en...

Agregar contexto... quotes.py Actual archivo Edición de archivos en el área de trabajo Agent GPT-4.1 209 a.m. 5/06/2025

The screenshot shows a browser window with several tabs open, including 'main.py - ConstructionCompany', 'word - Yahoo Search Tus resultados', 'Pruebas admin', and 'quotes.py - ConstructionCompany'. Below the browser is a VS Code interface.

VS Code Explorer:

- EXPLORADOR:** Shows the project structure under 'CONSTRUCTIONCOMPANY [CODESPACES: orange xylophone]':
 - src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - models
 - utils
- ESQUEMA**
- LÍNEA DE TIEMPO**
- DEPENDENCIAS**

VS Code Editor:

File: quotes.py - ConstructionCompany [Codespaces: orange xylophone]

```

src > backend > app > api > endpoints > quotes.py > ...
1  from fastapi import APIRouter, HTTPException
2  from typing import List
3  from uuid import uuid4
4  from src.backend.app.api.models.quotes import Cotizacion
5
6  router = APIRouter()
7
8  # Base de datos en memoria para cotizaciones
9  cotizaciones_db: List[Cotizacion] = []
10
11 @router.post("/", response_model=Cotizacion, status_code=201, summary=
12 def crear_cotizacion(cotizacion: Cotizacion):
13     cotizacion.id = str(uuid4())
14
15 bash

```

VS Code Problems Panel:

- Id. del proceso (PID): 64173
- Línea de comandos: /bin/bash --init-file /vscode/bin/linux-x64/259e40fedccb8edf399a463ce3a9d32e71ff0f3/out/vs/workbench/contrib/terminal/TTP/1.1!/common/scripts/shellIntegration-bash.sh
- INFO: blle Ent: Integración de shell: Enriquecido
- INFO: Mostrar contribuciones del entorno Mostrar detalles
- INFO: 127.0.0.1:55508 - "GET /quotes/%CID_DE_COTIZACION% HTTP/1.1" 404 Not Found

VS Code Status Bar:

- Codepaces: orange xylophone
- ConstructionCompany
- main*
- 489 ▲ 11
- 16
- FastAPI (ConstructionCompany)
- UTF-8 LF
- Python
- 3.10.12 64-bit
- Linux (linux-x64)
- Diseño: Latin American
- Buscar
- ESP LAA
- 2:09 a.m.
- 5/06/2025

The screenshot shows a browser-based code editor interface for a FastAPI application named "ConstructionCompany".

File Explorer: Shows the project structure under "CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]".

- src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py

Terminal: Displays a terminal session with the user @Gabrielleon0215 interacting with the application. The user runs `curl -X GET "http://localhost:8000/quotes/<ID_DE_COTIZACION>"` and receives a response indicating "Not Found".

```

@Gabrielleon0215 → /workspaces/ConstructionCompany
curl -X GET "http://localhost:8000/quotes/<ID_DE_COTIZACION>" 
{"detail":"Not Found"}@Gabrielleon0215 → /workspaces/ConstructionCompany
ny (main) $ 
  
```

Code Editor: The main window displays the content of quotes.py.

```

src > backend > app > api > endpoints > quotes.py ...
1 from fastapi import APIRouter, HTTPException
2 from typing import List
3 from uuid import uuid4
4 from src.backend.app.api.models.quotes import Cotizacion
5
6 router = APIRouter()
7
8 # Base de datos en memoria para cotizaciones
9 cotizaciones_db: List[Cotizacion] = []
10
11 @router.post("/", response_model=Cotizacion, status_code=201, summary="")
12 def crear_cotizacion(cotizacion: Cotizacion):
13     cotizacion.id = str(uuid4())
14     cotizaciones_db.append(cotizacion)
15     return cotizacion
  
```

Right Panel: Contains a sidebar with sections for "CHAT", "3. Obtener una cotización por ID (GET)", "4. Eliminar una cotización (DELETE)", and "¿Necesitas el comando para otro endpoint o ayuda con los campos del modelo?". It also shows a list of changed files and a preview of the current file.

The screenshot shows a browser-based code editor interface for a FastAPI application named "ConstructionCompany".

Left Sidebar (Explorador):

- CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]
 - src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - models
 - utils
- ESQUEMA
- LÍNEA DE TIEMPO
- DEPENDENCIAS

Central Area:

File: README.md

```
src > backend > app > api > endpoints > quotes.py > ...
1 from fastapi import APIRouter, HTTPException
2 from typing import List
3 from uuid import uuid4
4 from src.backend.app.api.models.quotes import Cotizacion
5
6 router = APIRouter()
7
8 # Base de datos en memoria para cotizaciones
9 cotizaciones_db: List[Cotizacion] = []
10
11 @router.post("/", response_model=Cotizacion, status_code=201, summary="Crear una cotización")
12 def crear_cotizacion(cotizacion: Cotizacion):
13     cotizacion.id = str(uuid4())
14     cotizaciones_db.append(cotizacion)
15     return cotizacion
```

Terminal:

```
INFO: 127.0.0.1:40780 - "POST /purchases/ HTTP/1.1" 422 Unprocessable Entity
INFO: 127.0.0.1:39856 - "POST /quotes/ HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:48142 - "GET /quotes/ HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:55508 - "GET /quotes/%3CID_DE_COTIZACION%3E HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:54798 - "DELETE /quotes/%3CID_DE_COTIZACION%3E HTTP/1.1" 404 Not Found
```

Right Sidebar (CHAT):

3. Obtener una cotización por ID (GET):

```
curl -X GET "http://localhost:8000/quotes/1"
```

Reemplaza <ID_DE_COTIZACION> por el id real de una cotización.

4. Eliminar una cotización (DELETE):

```
curl -X DELETE "http://localhost:8000/quotes/1"
```

¿Necesitas el comando para otro endpoint o ayuda con los campos del modelo?

1 archivo cambiado

payments.py src/backend/app/api/endpoints/quotes.py

Agregar contexto...

Edición de archivos en el área de trabajo

Agent GPT-4.1

2:10 a.m. 5/06/2025

orange-xylophone-r4wgvgvq47r2xq69.github.dev

ConstructionCompany [Codespaces: orange-xylophone]

EXPLORADOR

- CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]
 - src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - models
 - utils
- ESQUEMA
- LÍNEA DE TIEMPO
- DEPENDENCIES

[Vista previa] README.md

```

src > backend > app > api > endpoints > tasks.py > ...
1  from fastapi import APIRouter, HTTPException
2  from typing import List
3  from uuid import uuid4
4  from src.backend.app.api.models.tasks import Tarea
5
6  router = APIRouter(tags=["tareas"])
7
8  tareas_db: List[Tarea] = []
9
10 @router.post("/", response_model=Tarea, status_code=201)
11 def crear_tarea(tarea: Tarea):
12     tarea.id = str(uuid4())
13     tareas_db.append(tarea)
14     return tarea
15

```

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

```

@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X GET "http://localhost:8000/tareas/"
{"detail":"Not Found"}@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $

```

1 archivo cambiado

payments.py src/backend/app/api/endpoints/tasks.py Actual archivo

Edición de archivos en el área de trabajo

Buscar Agent GPT-4.1

ESP LAA 212 a.m. 5/06/2025

The screenshot shows a GitHub Codespace interface for a project named "ConstructionCompany". The browser tab is titled "tasks.py - ConstructionCompany". The address bar shows the URL "orange-xylophone-r4wgwvvgvq47r2xq69.github.dev". The main content area is a code editor with the following code:

```

from fastapi import APIRouter, HTTPException
from typing import List
from uuid import uuid4
from src.backend.app.api.models.tasks import Tarea

router = APIRouter(tags=["tareas"])

tareas_db: List[Tarea] = []

@router.post("/", response_model=Tarea, status_code=201)
def crear_tarea(tarea: Tarea):
    tarea.id = str(uuid4())
    tareas_db.append(tarea)
    return tarea

```

The terminal at the bottom shows several INFO log messages indicating 404 Not Found errors for various endpoints. A sidebar on the right provides instructions for listing and updating tasks using curl commands.

orange-xylophone-r4wgwgvq47r2xq69.github.dev

ConstructionCompany [Codespaces: orange-xylophone]

EXPLORADOR

CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]

- src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchases.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
- models
- utils

Vista previa] README.md

```

1 from fastapi import APIRouter, HTTPException
2 from typing import List
3 from uuid import uuid4
4 from src.backend.app.api.models.tasks import Tarea
5
6 router = APIRouter(tags=["tareas"])
7
8 tareas_db: List[Tarea] = []
9
10 @router.post("/", response_model=Tarea, status_code=201)
11 def crear_tarea(tarea: Tarea):
12     tarea.id = str(uuid4())
13     tareas_db.append(tarea)
14     return tarea
15

```

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

```

@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X PUT "http://localhost:8000/tareas/<ID_DE_TAREA>" \
-H "Content-Type: application/json" \
-d '{"nombre": "Revisión de planos actualizada", "descripcion": "Plano revisado"}'
{"detail": "Not Found"}@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $

```

1 archivo cambiado

payments.py src/backend/app/api/en...

Agregar contexto...

tasks.py Actual archivo

Edición de archivos en el área de trabajo

ESP LAA 2:13 a.m. 5/06/2025

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows the project structure under "CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]" with files like main.py, word.py, and various API endpoints (admin.py, auth.py, clients.py, dashboard.py, inventory.py, payments.py, projects.py, purchase.py, quotes.py, tasks.py).
- Code Editor:** Displays tasks.py containing the following code:

```

from fastapi import APIRouter, HTTPException
from typing import List
from uuid import uuid4
from src.backend.app.api.models.tasks import Tarea

router = APIRouter(tags=["tareas"])

tareas_db: List[Tarea] = []

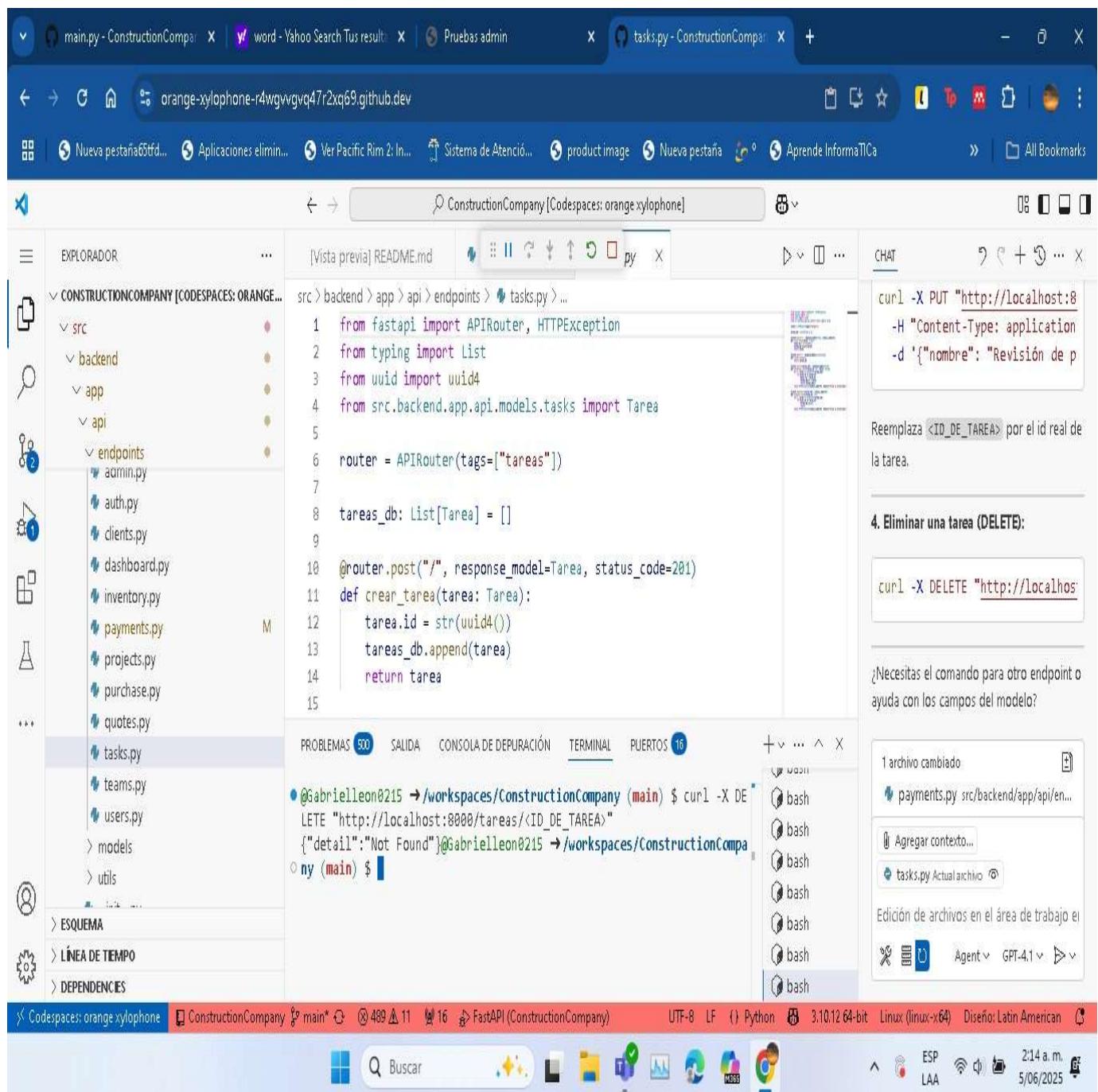
@router.post("/", response_model=Tarea, status_code=201)
def crear_tarea(tarea: Tarea):
    tarea.id = str(uuid4())
    tareas_db.append(tarea)
    return tarea

```
- Terminal:** Shows log output from a local host server:

```

INFO: 127.0.0.1:55508 - "GET /quotes/ CID_DE_COTIZACION HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:54798 - "DELETE /quotes/ CID_DE_COTIZACION HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:38022 - "POST /tareas/ HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:34576 - "GET /tareas/ HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:44352 - "PUT /tareas/ CID_DE_TAREA HTTP/1.1" 404 Not Found

```
- Sidebar:** Contains sections for "2. Listar todas las tareas (GET)" and "3. Actualizar una tarea (PUT)" with corresponding curl command examples.
- Bottom Status Bar:** Shows the file path "Codespaces: orange xylophone/main*", file statistics (489 lines, 11 changes, 16 additions), the FastAPI extension icon, encoding (UTF-8 LF), Python version (3.10.12 64-bit), operating system (Linux (linux-x64)), design (Latin American), and a timestamp (2:13 a.m., 5/06/2025).



```

from fastapi import APIRouter, HTTPException
from typing import List
from uuid import uuid4
from src.backend.app.api.models.tasks import Tarea

router = APIRouter(tags=["tareas"])

tareas_db: List[Tarea] = []

@router.post("/", response_model=Tarea, status_code=201)
def crear_tarea(tarea: Tarea):
    tarea.id = str(uuid4())
    tareas_db.append(tarea)
    return tarea

```

The screenshot shows a terminal window with the following content:

- EXPLORADOR** sidebar showing the project structure:

 - CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]
 - src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - CHAT** pane on the right showing a curl command to test the endpoint.
 - TERMINAL** pane at the bottom showing a curl command and its output.
 - PUERTOS** pane on the right showing multiple bash terminals.
 - PROBLEMAS**, **SALIDA**, **CONSOLA DE DEPURACIÓN** tabs at the bottom.
 - ESTADÍSTICAS** at the bottom: 1 archivo cambiado, payments.py src/backend/app/api/en...
 - EDICIÓN** pane on the right showing the file tasks.py.
 - INFO** pane on the right showing system information: Agent v GPT-4.1 v, 2:14 a.m., 5/06/2025.

Screenshot of a GitHub Codespace interface showing the `tasks.py` file for a FastAPI application.

The terminal shows the command:

```
@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X DELETE "http://localhost:8000/tareas/<ID_DE_TAREA>"
```

The code in `tasks.py` is as follows:

```
from fastapi import APIRouter, HTTPException
from typing import List
from uuid import uuid4
from src.backend.app.api.models.tasks import Tarea

router = APIRouter(tags=["tareas"])

tareas_db: List[Tarea] = []

@router.post("/", response_model=Tarea, status_code=201)
def crear_tarea(tarea: Tarea):
    tarea.id = str(uuid4())
    tareas_db.append(tarea)
    return tarea
```

Helpful hints in the sidebar:

- Reemplaza <ID_DE_TAREA> por el id real de la tarea.
- 4. Eliminar una tarea (DELETE):
- curl -X DELETE "http://localhost:8000/tareas/<ID_DE_TAREA>"
- ¿Necesitas el comando para otro endpoint o ayuda con los campos del modelo?

Bottom status bar:

```
1 archivo cambiado
payments.py src/backend/app/api/en...
Añadir contexto...
tasks.py Actual archivo
Edición de archivos en el área de trabajo el
Agent GPT-4.1 2:14 a.m. 5/06/2025
```

The screenshot shows a browser window with three tabs open:

- main.py - ConstructionCompany
- word - Yahoo Search Tus resultados
- Pruebas admin

The main content area shows a GitHub Codespace for the "ConstructionCompany" repository. The code editor displays the file `tasks.py` from the `src/backend/app/api/endpoints` directory. The code defines a FastAPI router for tasks:

```

from fastapi import APIRouter, HTTPException
from typing import List
from uuid import uuid4
from src.backend.app.api.models.tasks import Tarea
router = APIRouter(tags=["tareas"])
tareas_db: List[Tarea] = []
@router.post("/", response_model=Tarea, status_code=201)
def crear_tarea(tarea: Tarea):
    tarea.id = str(uuid4())
    tareas_db.append(tarea)
    return tarea

```

The terminal below shows several 404 Not Found errors:

```

INFO: 127.0.0.1:54798 - "DELETE /quotes/%CID_DE_COTIZACION% HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:38022 - "POST /tareas/ HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:34576 - "GET /tareas/ HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:44352 - "PUT /tareas/%CID_DE_TAREA% HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:36180 - "DELETE /tareas/%CID_DE_TAREA% HTTP/1.1" 404 Not Found

```

A sidebar on the right contains curl commands for API endpoints and a Chat section with a message about replacing placeholder IDs.

7.10 Pruebas teams.py

The screenshot shows a browser window with several tabs open. The active tab is 'teams.py - ConstructionCompany'. The page content displays a code editor with Python code for a FastAPI endpoint to create a team. Below the code editor is a terminal window showing a curl command being run. The status bar at the bottom of the screen shows the file size as 489 bytes.

```

src > backend > app > api > endpoints > teams.py > ...
1  from fastapi import APIRouter, HTTPException, Depends, status
2  from typing import List
3  from uuid import uuid4
4  from src.backend.app.api.endpoints.auth import get_current_user
5  from src.backend.app.api.models.teams import Equipo
6
7  router = APIRouter(tags=["equipos"])
8
9  equipos_db: List[Equipo] = []
10
11 @router.post("/", response_model=Equipo, status_code=201)
12 async def crear_equipo(equipo: Equipo, current_user: dict = Depends(get_current_user)):
13     equipo.id = str(uuid4())
14     equipos_db.append(equipo)
15     return equipo

```

```

from fastapi import APIRouter, HTTPException, Depends, status
from typing import List
from uuid import uuid4
from src.backend.app.api.endpoints.auth import get_current_user
from src.backend.app.api.models.teams import Equipo

router = APIRouter(tags=["equipos"])

equipos_db: List[Equipo] = []

@router.post("/", response_model=Equipo, status_code=201)
async def crear_equipo(equipo: Equipo, current_user: dict = Depends(get_current_user)):
    equipo.id = str(uuid4())
    equipos_db.append(equipo)
    return equipo

```

INFO: 127.0.0.1:38022 - "POST /tareas/ HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:34576 - "GET /tareas/ HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:44352 - "PUT /tareas/%3CID_DE_TAREA%3E HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:36180 - "DELETE /tareas/%3CID_DE_TAREA%3E HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:44180 - "POST /equipos/ HTTP/1.1" 422 Unprocessable Entity

header Authorization (por ejemplo, "Bearer token_valido");

1. Crear un equipo (POST):

```
curl -X POST "http://localhost:8000/equipos/"  

-H "Content-Type: application/json"  

-H "Authorization: Bearer tok...  

-d '{"nombre": "Equipo A", "d...
```

Ajusta los campos del JSON según los atributos requeridos por tu modelo
Equipo.

PROBLEMAS 800 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

payments.py src/backend/app/api/en...

Agregar contexto... teams.py Actual archivo

Edición de archivos en el área de trabajo

Agent GPT-4.1

Buscar

ESP LAA 2:15 a. m. 5/06/2025

orange-xylophone-r4wgvgvq47r2xq69.github.dev

Nueva pestaña 6/6fd... Aplicaciones eliminadas Ver Pacific Rim 2: In... Sistema de Atención product image Nueva pestaña Aprende Informática All Bookmarks

ConstructionCompany [Codespaces: orange xylophone]

EXPLORADOR

CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]

- src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - models
 - utils

ESQUEMA

LÍNEA DE TIEMPO

DEPENDENCIAS

Vista previa] README.md

teams.py

```
from fastapi import APIRouter, HTTPException, Depends, status
from typing import List
from uuid import uuid4
from src.backend.app.api.endpoints.auth import get_current_user
from src.backend.app.api.models.teams import Equipo

router = APIRouter(tags=["equipos"])

equipos_db: List[Equipo] = []

@router.post("/", response_model=Equipo, status_code=201)
async def crear_equipo(equipo: Equipo, current_user: dict = Depends(get_current_user)):
    equipo.id = str(uuid4())
    equipos_db.append(equipo)
    return equipo
```

CHAT

-d '{"nombre": "Equipo A", "d' Ajusta los campos del JSON según los atributos requeridos por tu modelo Equipo .

2. Listar todos los equipos (GET):

```
curl -X GET "http://localhost:8000/equipos/" -H "Authorization: Bearer tok
```

3. Obtener un equipo por ID (GET):

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

@Gabrielleon0215 → /workspaces/ConstructionCompany (main) \$ curl -X GET "http://localhost:8000/equipos/" -H "Authorization: Bearer token_valido"

[[@Gabrielleon0215 → /workspaces/ConstructionCompany (main) \$

1 archivo cambiado payments.py src/backend/app/api/en...

Agregar contexto... teams.py Actual archivo

Edición de archivos en el área de trabajo e Agent GPT-4.1

Codestar: main* ② 489 ▲ 11 ④ 16 FastAPI (ConstructionCompany) UTF-8 LF Python 3.10.12 64-bit Linux (linux-x64) Diseño: Latin American

Buscar

ESP LAA 2:16 a.m. 5/06/2025

orange-xylophone-r4wgvgvq47r2xq69.github.dev

ConstructionCompany [Codespaces: orange xylophone]

EXPLORADOR

- CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]
 - src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - models
 - utils
- ESQUEMA
- LÍNEA DE TIEMPO
- DEPENDENCIAS

[Vista previa] README.md

```

1 from fastapi import APIRouter, HTTPException, Depends, status
2 from typing import List
3 from uuid import uuid4
4 from src.backend.app.api.endpoints.auth import get_current_user
5 from src.backend.app.api.models.teams import Equipo
6
7 router = APIRouter(tags=["equipos"])
8
9 equipos_db: List[Equipo] = []
10
11 @router.post("/", response_model=Equipo, status_code=201)
12 async def crear_equipo(equipo: Equipo, current_user: dict = Depends(get_current_user)):
13     equipo.id = str(uuid4())
14     equipos_db.append(equipo)
15     return equipo
  
```

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

```

INFO: 127.0.0.1:34576 - "GET /tareas/ HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:44352 - "PUT /tareas/%3CID_DE_TAREA%3E HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:36180 - "DELETE /tareas/%3CID_DE_TAREA%3E HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:44180 - "POST /equipos/ HTTP/1.1" 422 Unprocessable Entity
INFO: 127.0.0.1:58214 - "GET /equipos/ HTTP/1.1" 200 OK
  
```

CHAT

-d '{"nombre": "Equipo A", "d' Ajusta los campos del JSON según los atributos requeridos por tu modelo Equipo .

2. Listar todos los equipos (GET):

```
curl -X GET "http://localhost:8000/equipos"
```

-H "Authorization: Bearer token"

3. Obtener un equipo por ID (GET):

1 archivo cambiado payments.py src/backend/app/api/endpoints/teams.py Actual archivo Copilot Agent GPT-4.1

Edición de archivos en el área de trabajo

Buscar

ESP LAA 2:16 a.m. 5/06/2025

orange-xylophone-r4wgvgvq47r2xq69.github.dev

ConstructionCompany [Codespaces: orange xylophone]

EXPLORADOR

- CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]
 - src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - models
 - utils
- ESQUEMA
- LÍNEA DE TIEMPO
- DEPENDENCIAS

[Vista previa] README.md

```
src > backend > app > api > endpoints > teams.py > ...
1 from fastapi import APIRouter, HTTPException, Depends, status
2 from typing import List
3 from uuid import uuid4
4 from src.backend.app.api.endpoints.auth import get_current_user
5 from src.backend.app.api.models.teams import Equipo
6
7 router = APIRouter(tags=["equipos"])
8
9 equipos_db: List[Equipo] = []
10
11 @router.post("/", response_model=Equipo, status_code=201)
12 async def crear_equipo(equipo: Equipo, current_user: dict = Depends(get_current_user)):
13     equipo.id = str(uuid4())
14     equipos_db.append(equipo)
15     return equipo
```

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

```
@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X GET "http://localhost:8000/equipos/<ID_DE_EQUIPO>" \
-H "Authorization: Bearer token_valido"
{"detail":"Equipo no encontrado"}@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $
```

CHAT

2. Listar todos los equipos (GET):

```
curl -X GET "http://localhost:8000/equipos" \
-H "Authorization: Bearer token_valido"
```

3. Obtener un equipo por ID (GET):

```
curl -X GET "http://localhost:8000/equipos/<ID_EQUIPO>" \
-H "Authorization: Bearer token_valido"
```

Reemplaza <ID_EQUIPO> por el id real del equipo.

1 archivo cambiado

payments.py src/backend/app/api/endpoints/teams.py

Agregar contexto...

Actualizar archivo...

Edición de archivos en el área de trabajo...

Agent GPT-4.1

Buscar

ESP LAA 2:16 a.m. 5/06/2025

orange-xylophone-r4wgvgvq472xq69.github.dev

CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]

```

src > backend > app > api > endpoints > teams.py > ...
1 from fastapi import APIRouter, HTTPException, Depends, status
2 from typing import List
3 from uuid import uuid4
4 from src.backend.app.api.endpoints.auth import get_current_user
5 from src.backend.app.api.models.teams import Equipo
6
7 router = APIRouter(tags=["equipos"])
8
9 equipos_db: List[Equipo] = []
10
11 @router.post("/", response_model=Equipo, status_code=201)
12 async def crear_equipo(equipo: Equipo, current_user: dict = Depends(get_current_user)):
13     equipo.id = str(uuid4())
14     equipos_db.append(equipo)
15     return equipo

```

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

04 Not Found
INFO: 127.0.0.1:36180 - "DELETE /tareas/%3CID_DE_TAREA%3E HTTP/1.1"
" 404 Not Found
INFO: 127.0.0.1:44180 - "POST /equipos/ HTTP/1.1" 422 Unprocessable Entity
INFO: 127.0.0.1:58214 - "GET /equipos/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:59286 - "GET /equipos/%3CID_DE_EQUIPO%3E HTTP/1.1"
404 Not Found

CHAT D R + X

2. Listar todos los equipos (GET):
curl -X GET "http://localhost:8000/equipos"
-H "Authorization: Bearer token"

3. Obtener un equipo por ID (GET):
curl -X GET "http://localhost:8000/equipos/1"
-H "Authorization: Bearer token"

Reemplaza <ID_EQUIPO> por el id real del equipo.

1 archivo cambiado payments.py src/backend/app/api/en...

Agregar contexto... teams.py Actual archivo

Edición de archivos en el área de trabajo

Bash Python Bash Bash Bash Copilot Agent GPT-4.1

Codespaces: orange-xylophone ConstructionCompany main* 489▲11 16 FastAPI (ConstructionCompany) UTF-8 LF Python 3.10.12 64-bit Linux (Linux-x64) Diseño: Latin American 217 a.m. 5/06/2025

orange-xylophone-r4wgvgvq47r2xq69.github.dev

src > backend > app > api > endpoints > teams.py > ...

```

1  from fastapi import APIRouter, HTTPException, Depends, status
2  from typing import List
3  from uuid import uuid4
4  from src.backend.app.api.endpoints.auth import get_current_user
5  from src.backend.app.api.models.teams import Equipo
6
7  router = APIRouter(tags=["equipos"])
8
9  equipos_db: List[Equipo] = []
10
11 @router.post("/", response_model=Equipo, status_code=201)
12 async def crear_equipo(equipo: Equipo, current_user: dict = Depends(get_current_user)):
13     equipo.id = str(uuid4())
14     equipos_db.append(equipo)
15     return equipo

```

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

-d '{"nombre": "Equipo A Actualizado", "descripcion": "Descripción actualizada"}'
{"detail":[{"type":"missing","loc":["body","estado"],"msg":"Field required","input":{"nombre":"Equipo A Actualizado","descripcion":"Descripción actualizada"},"url":"https://errors.pydantic.dev/2.5/v/missing"}, {"type":"missing","loc":["body","ubicacion"],"msg":"Field required","input":{"nombre":"Equipo A Actualizado","descripcion":"Descripción actualizada"},"url":"https://errors.pydantic.dev/2.5/v/missing"}]}@Gabriell

eon0215 → /workspaces/ConstructionCompany (main) \$

1 archivo cambiado payments.py src/backend/app/api/en...

Agregar contexto... teams.py Actual archivo

Edición de archivos en el área de trabajo el

Agent GPT-4.1 2:17 a.m. 5/06/2025

orange-xylophone-r4wgvvgvq47r2xq69.github.dev

src > backend > app > api > endpoints > teams.py

```

1  from fastapi import APIRouter, HTTPException, Depends, status
2  from typing import List
3  from uuid import uuid4
4  from src.backend.app.api.endpoints.auth import get_current_user
5  from src.backend.app.api.models.teams import Equipo
6
7  router = APIRouter(tags=["equipos"])
8
9  equipos_db: List[Equipo] = []
10
11 @router.post("/", response_model=Equipo, status_code=201)
12 async def crear_equipo(equipo: Equipo, current_user: dict = Depends(get_current_user)):
13     equipo.id = str(uuid4())
14     equipos_db.append(equipo)
15     return equipo

```

Reemplaza <ID_DE_EQUIPO> por el id real del equipo.

4. Actualizar un equipo (PUT):

```

curl -X PUT "http://localhost:8000/equipos/1"
-H "Content-Type: application/json"
-H "Authorization: Bearer token"
-d '{"nombre": "Equipo A Actualizado"}'

```

5. Eliminar un equipo (DELETE):

1 archivo cambiado

payments.py src/backend/app/api/endpoints/payments.py

Agregar contexto...

teams.py Actual archivo

Edición de archivos en el área de trabajo

Agent GPT-4.1

eon0215 → /workspaces/ConstructionCompany (main) \$

Buscar

ESP LAA 217 a.m. 5/06/2025

orange-xylophone-r4wgwvgvq47r2xq69.github.dev

ConstructionCompany [Codespaces: orange xylophone]

EXPLORADOR

- CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]
 - src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - > models
 - > utils
- ESQUEMA
- LÍNEA DE TIEMPO
- DEPENDENCIAS

[Vista previa] README.md

```

1 from fastapi import APIRouter, HTTPException, Depends, status
2 from typing import List
3 from uuid import uuid4
4 from src.backend.app.api.endpoints.auth import get_current_user
5 from src.backend.app.api.models.teams import Equipo
6
7 router = APIRouter(tags=["equipos"])
8
9 equipos_db: List[Equipo] = []
10
11 @router.post("/", response_model=Equipo, status_code=201)
12 async def crear_equipo(equipo: Equipo, current_user: dict = Depends(get_current_user)):
13     equipo.id = str(uuid4())
14     equipos_db.append(equipo)
15     return equipo

```

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

```

" 404 Not Found
INFO: 127.0.0.1:44180 - "POST /equipos/" 422 Unprocessable Entity
INFO: 127.0.0.1:58214 - "GET /equipos/" 200 OK
INFO: 127.0.0.1:59286 - "GET /equipos/%3CID_DE_EQUIPO%3E HTTP/1.1"
404 Not Found
INFO: 127.0.0.1:48090 - "PUT /equipos/%3CID_DE_EQUIPO%3E HTTP/1.1"
422 Unprocessable Entity

```

CHAT

Reemplaza <ID_DE_EQUIPO> por el id real del equipo.

4. Actualizar un equipo (PUT):

```

curl -X PUT "http://localhost:8000/equipos/1" \
-H "Content-Type: application/json" \
-H "Authorization: Bearer token" \
-d '{"nombre": "Equipo A Actualizado"}'

```

5. Eliminar un equipo (DELETE):

1 archivo cambiado

payments.py src/backend/app/api/endpoints/teams.py

Agregar contexto...

teams.py Actual archivo

Edición de archivos en el área de trabajo

Agent GPT-4.1

217 a.m. 5/06/2025

The screenshot displays a browser-based development environment for a FastAPI application named "ConstructionCompany".

Project Structure:

- EXPLORADOR (Sidebar): CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]
 - src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - ESQUEMA
 - LÍNEA DE TIEMPO
 - DEPENDENCIAS

Code Editor:

```

src> backend > app > api > endpoints > teams.py > ...
1  from fastapi import APIRouter, HTTPException, Depends, status
2  from typing import List
3  from uuid import uuid4
4  from src.backend.app.api.endpoints.auth import get_current_user
5  from src.backend.app.api.models.teams import Equipo
6
7  router = APIRouter(tags=["equipos"])
8
9  equipos_db: List[Equipo] = []
10
11 @router.post("/", response_model=Equipo, status_code=201)
12 async def crear_equipo(equipo: Equipo, current_user: dict = Depends(get_current_user)):
13     equipo.id = str(uuid4())
14     equipos_db.append(equipo)
15     return equipo
  
```

Terminal:

```

@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X DELETE "http://localhost:8000/equipos/<ID_EQUIPO>" \
-H "Authorization: Bearer token_valido"
{"detail":"Equipo no encontrado"}@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ 
  
```

Right Panel:

 - CHAT
 - 4. Actualizar un equipo (PUT):


```
curl -X PUT "http://localhost:8000/equipos/<ID_EQUIPO>" \
-H "Content-Type: application/json" \
-H "Authorization: Bearer token_valido" \
-d '{"nombre": "Equipo A Actualizado", "descripcion": "Nuevo equipo A"}'
```
 - 5. Eliminar un equipo (DELETE):


```
curl -X DELETE "http://localhost:8000/equipos/<ID_EQUIPO>" \
-H "Authorization: Bearer token_valido"
```
 - Necesitas ayuda para obtener el token o el nombre del equipo.
 - 1 archivo cambiado
 - payments.py src/backend/app/api/endpoints/teams.py
 - Agregar contexto...
 - teams.py Actual archivo
 - Edición de archivos en el área de trabajo es...
 - Agent GPT-4.1

Bottom Status Bar:

Codepaces: orangexylophone ConstructionCompany main* ④ 489 ▲ 11 🔍 16 ⚡ FastAPI (ConstructionCompany) UTF-8 LF Python 3.10.12 64-bit Linux (linux-x64) Diseño: Latin American 2:18 a.m. 5/06/2025

orange-xylophone-r4wgvgvq47r2xq69.github.dev

src > backend > app > api > endpoints > teams.py > ...

```

1 from fastapi import APIRouter, HTTPException, Depends, status
2 from typing import List
3 from uuid import uuid4
4 from src.backend.app.api.endpoints.auth import get_current_user
5 from src.backend.app.api.models.teams import Equipo
6
7 router = APIRouter(tags=["equipos"])
8
9 equipos_db: List[Equipo] = []
10
11 @router.post("/", response_model=Equipo, status_code=201)
12 async def crear_equipo(equipo: Equipo, current_user: dict = Depends(get_current_user)):
13     equipo.id = str(uuid4())
14     equipos_db.append(equipo)
15     return equipo

```

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

e Entity

INFO: 127.0.0.1:58214 - "GET /equipos/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:59286 - "GET /equipos/%3CID_DE_EQUIPO%3E HTTP/1.1"
404 Not Found
INFO: 127.0.0.1:48090 - "PUT /equipos/%3CID_DE_EQUIPO%3E HTTP/1.1"
422 Unprocessable Entity
INFO: 127.0.0.1:47254 - "DELETE /equipos/%3CID_DE_EQUIPO%3E HTTP/1.1"
404 Not Found

Pyth... bash Copilot Agent GPT-4.1

1 archivo cambiado payments.py src/backend/app/api/en...

Agregar contexto... teams.py Actual archivo

Necesitas ayuda para obtener el token o el de acceso

Codespaces: orange xylophone ConstructionCompany main* 489▲ 11 16 FastAPI (ConstructionCompany) UTF-8 LF Python 3.10.12 64-bit Linux (linux-x64) Diseño: Latin American 2:18 a.m. 5/06/2025

Buscar

orange-xylophone-r4wgwgvq47r2xq69.github.dev

Nueva pestaña | Aplicaciones eliminadas | Ver Pacific Rim 2: In... | Sistema de Atención | product image | Nueva pestaña | Aprende Informática | All Bookmarks

ConstructionCompany [Codespaces: orange-xylophone]

EXPLORADOR

- CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]
 - src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - models
 - utils
 - ESQUEMA
 - LÍNEA DE TIEMPO
 - DEPENDENCIAS

[Vista previa] README.md

```

src > backend > app > api > endpoints > users.py > ...
7  from src.backend.app.api.database.db import SessionLocal
8  from fastapi.responses import JSONResponse
9
10 router = APIRouter(tags=["usuarios"])
11
12 def get_db():
13     db = SessionLocal()
14     try:
15         yield db
16     finally:
17         db.close()
18
19 @router.post("/", response_model=Usuario, status_code=201)
20 def crear_usuario(usuario: Usuario, db: Session = Depends(get_db)):
21     db_usuario = UsuarioDB(

```

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

```

@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X POST "http://localhost:8000/usuarios/" \
-H "Content-Type: application/json" \
-d '{"nombre": "Juan", "email": "juan@email.com", "password": "1234", "rol": "admin"}'
{"detail": "Not Found"}@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ 

```

CHAT

ios endpoints de usuarios definidos en el archivo:

1. Crear un usuario (POST):

```
curl -X POST "http://localhost:8000/usuarios/" \
-H "Content-Type: application/json" \
-d '{"nombre": "Juan", "email": "juan@email.com", "password": "1234", "rol": "admin"}'
```

Puedes omitir "rol" si quieres que tome el valor por defecto.

2. Listar todos los usuarios (GET):

```
curl -X GET "http://localhost:8000/usuarios"
```

1 archivo cambiado

payments.py src/backend/app/api/endpoints/admin.py

Agregar contexto...

users.py Actual archivo

Edición de archivos en el área de trabajo

Agent GPT-4.1

Codestar

Buscar

ESP LAA 2:20 a.m. 5/06/2025

orange-xylophone-r4wgwvgvq47r2xq69.github.dev

ConstructionCompany [Codespaces: orange-xylophone]

```

src > backend > app > api > endpoints > users.py > ...
7   from src.backend.app.api.database.db import SessionLocal
8   from fastapi.responses import JSONResponse
9
10  router = APIRouter(tags=["usuarios"])
11
12  def get_db():
13      db = SessionLocal()
14      try:
15          yield db
16      finally:
17          db.close()
18
19  @router.post("/", response_model=Usuario, status_code=201)
20  def crear_usuario(usuario: Usuario, db: Session = Depends(get_db)):
21      db_usuario = UsuarioDB(

```

PROBLEMAS 300 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

INFO: 127.0.0.1:58214 - "GET /equipos/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:59286 - "GET /equipos/%CID_DE_EQUIPO%3E HTTP/1.1"
404 Not Found
INFO: 127.0.0.1:48090 - "PUT /equipos/%CID_DE_EQUIPO%3E HTTP/1.1"
422 Unprocessable Entity
INFO: 127.0.0.1:47254 - "DELETE /equipos/%CID_DE_EQUIPO%3E HTTP/1.1"
404 Not Found
INFO: 127.0.0.1:47066 - "POST /usuarios/ HTTP/1.1" 404 Not Found

bash Python bash bash bash Copilot bash bash bash Agent GPT-4.1

Buscar

489 ▲ 11 ⌂ 16 FastAPI (ConstructionCompany) UTF-8 LF Python 3.10.12 64-bit Linux (linux-x64) Diseño: Latin American

ESP LAA 2:20 a.m. 5/06/2025

The screenshot shows a browser window with multiple tabs open, including 'main.py - ConstructionCompany', 'users.py - ConstructionCompany', and 'Pruebas admin'. The main content area displays a code editor for a Python project named 'CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]'.

Code Editor Content:

```

    [Vista previa] README.md
    src > backend > app > api > endpoints > users.py > ...
    7   from src.backend.app.api.database.db import SessionLocal
    8   from fastapi.responses import JSONResponse
    9
    10  router = APIRouter(tags=["usuarios"])
    11
    12  def get_db():
    13      db = SessionLocal()
    14      try:
    15          yield db
    16      finally:
    17          db.close()
    18
    19  @router.post("/", response_model=Usuario, status_code=201)
    20  def crear_usuario(usuario: Usuario, db: Session = Depends(get_db)):
    21      db_usuario = UsuarioDB(
    22          nombre=usuario.nombre,
    23          email=usuario.email,
    24          rol=usuario.rol,
    25          password=usuario.password
    26      )
    27      db.add(db_usuario)
    28      db.commit()
    29      return db_usuario
    30
    31  @router.get("/{id}", response_model=Usuario)
    32  def get_usuario(id: int, db: Session = Depends(get_db)):
    33      db_usuario = db.query(UsuarioDB).filter(UsuarioDB.id == id).first()
    34      if db_usuario is None:
    35          raise HTTPException(status_code=404, detail="User not found")
    36      return db_usuario
  
```

Terminal:

```

  @Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X GET "http://localhost:8000/usuarios/"
  {"detail":"Not Found"}@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $
  
```

Right Panel:

- CHAT:** Shows a message: "-d '{"nombre": "Juan", "email": "juan@example.com"}'
- Puedes omitir "rol" si quieres que tome el valor por defecto.**
- 2. Listar todos los usuarios (GET):** curl -X GET "http://localhost:8000/usuarios"
- 3. Eliminar un usuario (DELETE):** curl -X DELETE "http://localhost:8000/usuarios/1"
- Edición de archivos en el área de trabajo:** payments.py, users.py
- Agent:** GPT-4.1

Bottom Status Bar:

Codespaces: orange xylophone | ConstructionCompany | main* | 489 ▲ 11 | 16 | FastAPI (ConstructionCompany) | UTF-8 | LF | Python | 3.10.12 64-bit | Linux (linux-x64) | Diseño: Latin American | ESP LAA | 2:20 a.m. | 5/06/2025

orange-xylophone-r4wgwgvq47r2xq69.github.dev

Nueva pestaña | Aplicaciones eliminadas | Ver Pacific Rim 2: In... | Sistema de Atención... | product image | Nueva pestaña | Aprende Informática | All Bookmarks

ConstructionCompany [Codespaces: orange xylophone]

EXPLORADOR

- CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]
 - src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - models
 - utils
 - ESQUEMA
 - LÍNEA DE TIEMPO
 - DEPENDENCIAS

Vista previa] README.md

```
src > backend > app > api > endpoints > users.py > ...
7   from src.backend.app.api.database.db import SessionLocal
8   from fastapi.responses import JSONResponse
9
10  router = APIRouter(tags=["usuarios"])
11
12  def get_db():
13      db = SessionLocal()
14      try:
15          yield db
16      finally:
17          db.close()
18
19  @router.post("/", response_model=Usuario, status_code=201)
20  def crear_usuario(usuario: Usuario, db: Session = Depends(get_db)):
21      db_usuario = UsuarioDB(
```

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

```
INFO: 127.0.0.1:59286 - "GET /equipos/%CID_DE_EQUIPO% HTTP/1.1"
404 Not Found
INFO: 127.0.0.1:48090 - "PUT /equipos/%CID_DE_EQUIPO% HTTP/1.1"
422 Unprocessable Entity
INFO: 127.0.0.1:47254 - "DELETE /equipos/%CID_DE_EQUIPO% HTTP/1.1"
404 Not Found
INFO: 127.0.0.1:47866 - "POST /usuarios/ HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:35834 - "GET /usuarios/ HTTP/1.1" 404 Not Found
```

CHAT

```
-d '{"nombre": "Juan", "email": "juan@example.com"}'
```

Puedes omitir "rol" si quieres que tome el valor por defecto.

2. Listar todos los usuarios (GET):

```
curl -X GET "http://localhost:8000/usuarios"
```

3. Eliminar un usuario (DELETE):

```
curl -X DELETE "http://localhost:8000/usuarios/1"
```

1 archivo cambiado

payments.py src/backend/app/api/endpoints.py

Agregar contexto...

users.py8 Actual archivo

Edición de archivos en el área de trabajo

Agent GPT-4.1

ESP LAA 2:21 a.m. 5/06/2025

Codespaces: orange xylophone ConstructionCompany main* 489 ▲ 11 16 FastAPI (ConstructionCompany) UTF-8 LF Python 3.10.12 64-bit Linux (linux-x64) Diseño: Latin American

The screenshot shows a browser window with several tabs open. The active tab is titled "ConstructionCompany[Codespaces: orange xylophone]" and contains a code editor with Python code for a FastAPI application. The code is as follows:

```

[Vista previa] README.md
src > backend > app > api > endpoints > users.py > ...
7   from src.backend.app.api.database.db import SessionLocal
8   from fastapi.responses import JSONResponse
9
10  router = APIRouter(tags=["usuarios"])
11
12  def get_db():
13      db = SessionLocal()
14      try:
15          yield db
16      finally:
17          db.close()
18
19  @router.post("/", response_model=Usuario, status_code=201)
20  def crear_usuario(usuario: Usuario, db: Session = Depends(get_db)):
21      db_usuario = UsuarioDB(

```

Below the code editor, there is a terminal window showing a curl command being run:

```

@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X DELETE "http://localhost:8000/usuarios/<ID_DEL_USUARIO>" {"detail":"Not Found"}@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $

```

The browser's address bar shows the URL "orange-xylophone-4wgvgvq47r2xq69.github.dev". The status bar at the bottom indicates the file is main*, line 489, and the editor is using FastAPI (ConstructionCompany).

Screenshot of a browser window showing a GitHub Codespace for a 'ConstructionCompany' repository. The code editor displays a Python file 'users.py' containing FastAPI endpoint logic for user management.

```

    from src.backend.app.api.database import SessionLocal
    from fastapi.responses import JSONResponse
    from fastapi import APIRouter
    from pydantic import BaseModel
    from typing import List
    from sqlalchemy.orm import Session
    from . import models
    from .schemas import User
    from .database import get_db
    from .dependencies import Depends

    router = APIRouter(tags=["usuarios"])

    def get_db():
        db = SessionLocal()
        try:
            yield db
        finally:
            db.close()

    @router.post("/", response_model=User, status_code=201)
    def crear_usuario(usuario: User, db: Session = Depends(get_db)):
        db_usuario = models.UsuarioDB(**usuario.dict())
        db.add(db_usuario)
        db.commit()
        db.refresh(db_usuario)
        return db_usuario

```

The terminal shows a curl command to test the endpoint:

```

@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $ curl -X DELETE "http://localhost:8000/usuarios/<ID_DEL_USUARIO>" {"detail":"Not Found"}@Gabrielleon0215 → /workspaces/ConstructionCompany (main) $

```

File navigation sidebar:

- EXPLORADOR
- CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]
- src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - models
 - utils
- ESQUEMA
- LÍNEA DE TIEMPO
- DEPENDENCIAS

Bottom status bar:

Codespaces: orange xylophone ConstructionCompany main* ⑧ 489 ▲ 11 ⌂ 16 FastAPI (ConstructionCompany) UTF-8 LF Python 3.10.12 64-bit Linux (linux-x64) Diseño: Latin American 2:21 a.m. ESP LAA 5/06/2025

orange-xylophone-r4wgwvgvq47r2xq69.github.dev

EXPLORADOR

CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]

- src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
- models
- utils

ESQUEMA

LÍNEA DE TIEMPO

DEPENDENCIAS

Vista previa] README.md

```

src > backend > app > api > endpoints > users.py > ...
7   from src.backend.app.api.database.db import SessionLocal
8   from fastapi.responses import JSONResponse
9
10  router = APIRouter(tags=["usuarios"])
11
12  def get_db():
13      db = SessionLocal()
14      try:
15          yield db
16      finally:
17          db.close()
18
19  @router.post("/", response_model=Usuario, status_code=201)
20  def crear_usuario(usuario: Usuario, db: Session = Depends(get_db)):
21      db_usuario = UsuarioDB(

```

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

```

INFO: 127.0.0.1:48090 - "PUT /equipos/%3CID_DE_EQUIPO%3E HTTP/1.1"
422 Unprocessable Entity
INFO: 127.0.0.1:47254 - "DELETE /equipos/%3CID_DE_EQUIPO%3E HTTP/1
.1" 404 Not Found
INFO: 127.0.0.1:47066 - "POST /usuarios/ HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:35834 - "GET /usuarios/ HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:59988 - "DELETE /usuarios/%3CID_DEL_USUARIO%3E HTT
P/1.1" 404 Not Found

```

CHAT

valor por defecto.

2. Listar todos los usuarios (GET):

```
curl -X GET "http://localhost:8000/usuarios"
```

3. Eliminar un usuario (DELETE):

```
curl -X DELETE "http://localhost:8000/usuarios/1"
```

Reemplaza <ID_DEL_USUARIO> por el id real del usuario que quieras eliminar.

1 archivo cambiado

payments.py src/backend/app/api/en...

Agregar contexto...

users.py Actual archivo

Edición de archivos en el área de trabajo ei

Agent GPT-4.1

Buscar

ESP LAA 221 a.m. 5/06/2025

main.py - ConstructionCompany | word - Yahoo Search Tus result... | Pruebas admin | users.py - ConstructionCompany | +

orange-xylophone-r4wgwvq47r2xq69.github.dev

Nueva pestaña 65fd... Aplicaciones eliminadas Ver Pacific Rim 2: In... Sistema de Atención... product image Nueva pestaña Aprende Informática All Bookmarks

ConstructionCompany [Codespaces: orange xylophone]

EXPLORADOR

CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]

- src
 - backend
 - app
 - api
 - endpoints
 - admin.py
 - auth.py
 - clients.py
 - dashboard.py
 - inventory.py
 - payments.py
 - projects.py
 - purchase.py
 - quotes.py
 - tasks.py
 - teams.py
 - users.py
 - models
 - utils
- ESQUEMA
- LÍNEA DE TIEMPO
- DEPENDENCIAS

[Vista previa] README.md

```

src > backend > app > api > endpoints > users.py > ...
7   from src.backend.app.api.database.db import SessionLocal
8   from fastapi.responses import JSONResponse
9
10  router = APIRouter(tags=["usuarios"])
11
12  def get_db():
13      db = SessionLocal()
14      try:
15          yield db
16      finally:
17          db.close()
18
19  @router.post("/", response_model=Usuario, status_code=201)
20  def crear_usuario(usuario: Usuario, db: Session = Depends(get_db)):
21      db_usuario = UsuarioDB(

```

Puertos - 16 puertos reenviados

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

@Gabrielleon0215 → /workspaces/ConstructionCompany (main) \$ curl -X OPTIONS "http://localhost:8000/usuarios/" {"detail": "Not Found"}@Gabrielleon0215 → /workspaces/ConstructionCompany (main) \$

1 archivo cambiado payments.py src/backend/app/api/en...

Agregar contexto... users.py Actual archivo

Edición de archivos en el área de trabajo

Agent GPT-4.1

Codespaces: orange xylophone ConstructionCompany main* 489 ▲ 11 16 FastAPI (ConstructionCompany) UTF-8 LF Python 3.10.12 64-bit Linux (linux-x64) Diseño: Latin American 2:22 a.m. 5/06/2025

Buscar

orange-xylophone-r4wgvgvq47r2xq69.github.dev

CONSTRUCTIONCOMPANY [CODESPACES: ORANGE...]

```

src > backend > app > api > endpoints > users.py > ...
7   from src.backend.app.api.database.db import SessionLocal
8   from fastapi.responses import JSONResponse
9
10  router = APIRouter(tags=["usuarios"])
11
12  def get_db():
13      db = SessionLocal()
14      try:
15          yield db
16      finally:
17          db.close()
18
19  @router.post("/", response_model=Usuario, status_code=201)
20  def crear_usuario(usuario: Usuario, db: Session = Depends(get_db)):
21      db_usuario = UsuarioDB(

```

PROBLEMAS 500 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 16

INFO: 127.0.0.1:47254 - "DELETE /equipos/%3CID_DE_EQUIPO%3E HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:47066 - "POST /usuarios/ HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:35834 - "GET /usuarios/ HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:59988 - "DELETE /usuarios/%3CID_DEL_USUARIO%3E HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:43338 - "OPTIONS /usuarios/ HTTP/1.1" 404 Not Found

1 archivo cambiado
payments.py src/backend/app/api/en...

Agregar contexto...
users.py8 Actual archivo

Edición de archivos en el área de trabajo

Bash Python Copilot

Buscar

ESP LAA 2:22 a.m. 5/06/2025

8. Referencias

Institute of Electrical and Electronics Engineers. (1998). IEEE 830-1998: Recommended practice for software requirements specifications. IEEE Standards Association. Recuperado de <https://standards.ieee.org/>

Nielsen Norman Group. (2024). Guía de estilo de interfaces de usuario para aplicaciones web. Recuperado de <https://www.nngroup.com>

International Organization for Standardization. (2011). ISO/IEC 25010:2011 - Systems and software engineering - Systems and software quality requirements and evaluation (SQuaRE). ISO. Recuperado de <https://www.iso.org>

Red Hat, IBM & Microsoft. (2023). Best practices for microservices and RESTful APIs. Recuperado de <https://microservices.io> y <https://redhat.com>

Empresa Constructora. (2025). Documento de visión y alcance del Proyecto OS. (Versión 1.0). Documento interno.

Universidad Tecnológica de Bolívar. (2024). Plantilla de Especificación de Requisitos de Software (SRS). [Documento académico].

Google. (n.d.). Get started: Install. Flutter. <https://docs.flutter.dev/get-started/install>

Arquitectura basada en Microservicios

El sistema ha sido rediseñado con una arquitectura basada en microservicios, lo que permite una mayor escalabilidad y flexibilidad. Cada módulo crítico - como la gestión de proyectos, el control de inventarios y la gestión de proveedores - se ha implementado como un servicio independiente que se comunica mediante APIs REST. Esta separación facilita el despliegue, la actualización y el mantenimiento de cada componente sin afectar el conjunto del sistema.

Definición de APIs

Para facilitar la comunicación entre los microservicios, se han definido los siguientes endpoints principales:

- GET /proyectos: Obtiene la lista de proyectos en curso.
- POST /proyectos: Crea un nuevo proyecto en el sistema.
- GET /inventarios: Recupera el estado de los inventarios.
- POST /inventarios: Afíade nuevos materiales al inventario.

Se han implementado medidas de seguridad, como la autenticación mediante OAuth 2.0 y el cifrado de datos, para garantizar la integridad y confidencialidad de la información intercambiada.

Apendice A: Glosario

- **Gateway:** Punto de acceso centralizado que gestiona y expone las API del sistema.
- **Cloud Analytics Platform:** Plataforma en la nube utilizada para almacenar, procesar y analizar datos en tiempo real.
- **Silo de Datos:** Sistema de información aislado que no se comunica con otros sistemas, limitando la integración de datos.
- **Inventario en tiempo real:** Sistema que permite conocer el estado y disponibilidad de recursos en cualquier momento.
- **Flujo de trabajo:** Secuencia de pasos y procesos en la ejecución de una tarea dentro del sistema.

Apéndice B: Modelos de análisis

Diagrama de Flujo de Datos (DFD)

- Muestra el movimiento de datos entre los módulos principales, como la gestión de proyectos, asignación de equipos y generación de reportes.

Diagrama de Clases

- Representa las principales entidades del sistema y sus relaciones, como **Proyecto**, **Usuario**, **Recurso**, **Notificación** y **Reporte**.

Diagrama Entidad-Relación (DER)

- Define cómo los datos se organizan en la base de datos, incluyendo tablas principales y

relaciones clave.

