Plataforma Workflow AI — Visión General

Descripción general del proyecto

Plataforma colaborativa para orquestar procesos con tareas encadenadas, telemetría y asistencia de IA. El **Frontend** ofrece un editor visual y paneles en tiempo real; el **Backend** expone contratos REST/OpenAPI que traducen formatos y coordinan la persistencia; el **Worker** materializa la ejecución y aplica estrategias de tareas, decoradores y comandos de IA.

Arquitectura global

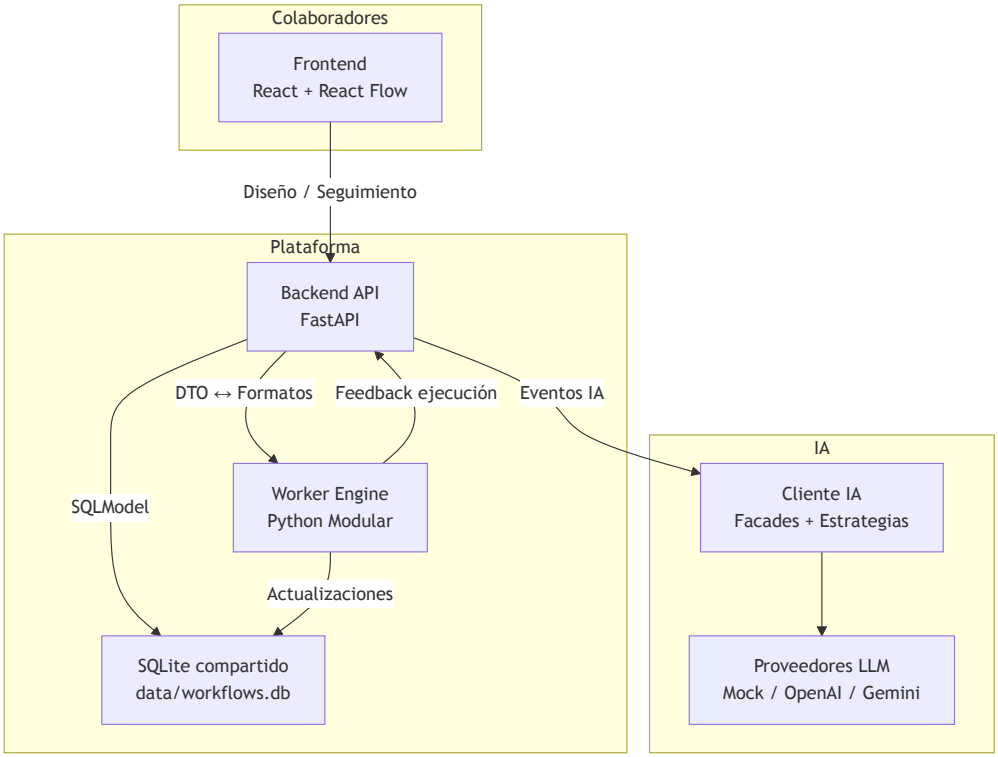

Figura 1: Arquitectura global de la plataforma

Nota: exportar el diagrama Mermaid original con `mmdc -i arquitectura.mmd -o figures/arquitectura.pdf`.

- **Persistencia compartida**: `data/workflows.db` actúa como punto de sincronía entre los servicios.
- **Contratos homogéneos**: el Backend traduce `steps/edges` (Frontend) a `nodes/depends_on` (Worker).
- **Subsistema de IA**: expuesto desde el Backend y ejecutado mediante comandos y estrategias en Worker e IA client.

Módulos y responsabilidades

Módulo	Rol principal	Tecnologías	Documentación
Frontend	Editor visual, monitoreo y experiencia colaborativa	React 19, Vite, Tailwind, React Flow	`Frontend/README.md`
Backend	API pública, orquestación, validaciones y capa IA	FastAPI, SQLModel, Alembic, Pytest	`Backend/README.md`
Worker	Ejecutor de workflows y catálogo de tareas	Python 3, Strategy/Command/Decorator, SQLite	`Worker/Documentacion/README.md`

- Cada módulo mantiene su propio README con instalación y comandos detallados.
- El diseño permite ciclos de despliegue independientes con contratos compartidos mediante OpenAPI y modelos SQLModel.

Flujos clave

Ciclo de diseño → ejecución

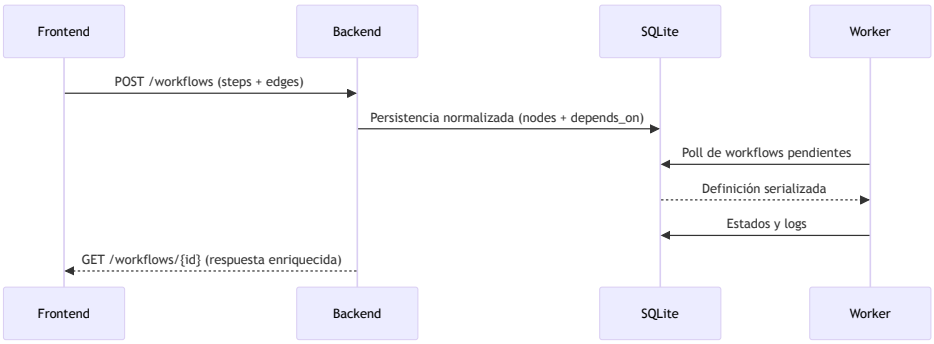

Figura 2: Ciclo de diseño, ejecución y sincronización

Nota: exportar el diagrama Mermaid original con `mmdc -i ciclo-workflow.mmd -o figures/ciclo-workflow.pdf`.

- El Frontend solicita optimizaciones y correcciones al Backend (`/ia/*`), que delega en el subsistema de IA y Worker para aplicar cambios o estimaciones.
- Los logs y estados se centralizan en la base compartida; el Frontend los expone con autoactualización y filtros.

Ambientes de trabajo

1. **Requisitos básicos**
   - Python 3.11+ y entorno virtual (para Backend y Worker).
   - Node.js 18+ (para Frontend).
   - SQLite disponible; el archivo se genera automáticamente en `data/`.
2. **Lanzar la plataforma en local**

Listing 1: Servicios locales

```
# Terminal 1 - Backend
cd Backend
uvicorn src.main:app --reload --port 8000

# Terminal 2 - Worker
cd Worker
# Sigue el README del Worker para inicializar el motor (WorkflowEngine / cli)

# Terminal 3 - Frontend
cd Frontend
npm install
npm run dev
```

- El Frontend apunta a `VITE_API_URL = http://localhost:8000`.
- `VITE_USE_MOCK = true` habilita trabajo sin Backend durante el desarrollo.

3. **Variables de entorno compartidas**

- Configuración de proveedores de IA mediante `IA_PROVIDER` y llaves gestionadas por el Backend.
- Tokens mock y credenciales definidos para pruebas locales (`POST /login`).

Testing y calidad

- **Backend**: suites de `pytest` para endpoints, repositorios y subsistema de IA (`Backend/tests`).
- **Worker**: pruebas unitarias e integración (`Worker/Tests`) con cobertura y escenarios parametrizados.
- **Frontend**: Vitest para componentes, hooks y servicios; incluye modo mock de contratos.
- **Integración**: los contratos REST se documentan en OpenAPI; úsese como referencia al sincronizar cambios entre módulos.

Recursos y referencias

- Documentación específica por módulo: `Frontend/README.md`, `Backend/README.md`, `Worker/Documentacion/R`
- Base de datos compartida: `data/workflows.db`, con esquema detallado en `Backend/BD_DISENIO.md`.
- Diagramas exportados: `figures/arquitectura.pdf` y `figures/ciclo-workflow.pdf` generados a partir de los archivos Mermaid originales.

Para información operativa o contribuciones, consulte el README correspondiente a cada módulo y sincronice los cambios de contratos en coordinación con el equipo.