Tecnológico de Monterrey

UNIVERSIDAD DE BURGOS

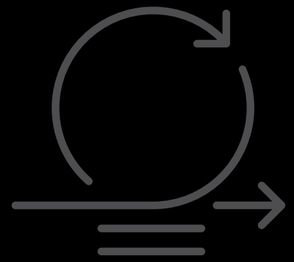
# DESARROLLO DE UN GESTOR DE TAREAS CON METODOLOGÍA SCRUM

Rodrigo Gutierrez Garcia
Liliana Odette Ortega Quezada
Victoria Rodríguez Domínguez
Carlos Julian Lopez Chavez
Vicente Jesus Ramos Chavez

# OBJETIVO DE LA PRÁCTICA

Gestionar tareas.

Aplicar reglas de negocio realistas.



Trabajar siguiendo los principios y eventos de Scrum.

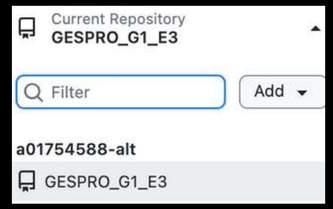
# QUÉ SE HA CONSTRUIDO (DEMO)

## INICIO

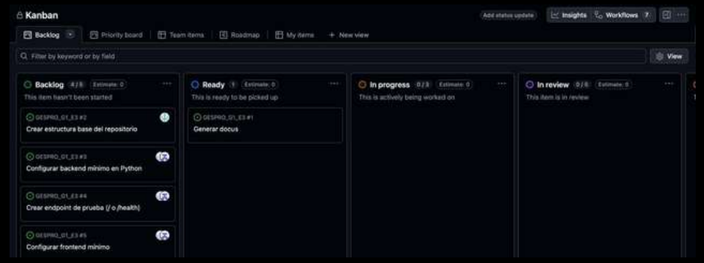
- Configuración del entorno

ANACONDA GitHub Visual Studio

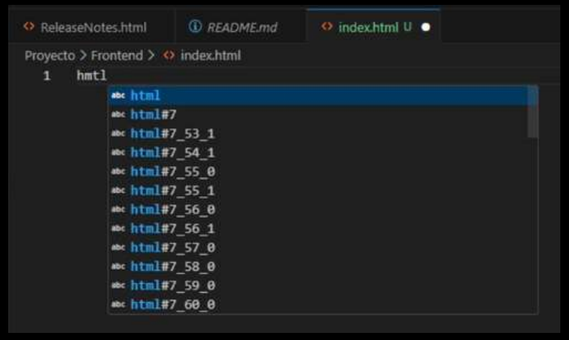
- Crear repository en GitHub y agregar colaboradores




- Crear Kanban en GitHub Projects




- Clonar el repo
- Crear entorno y abrir VS Code



# QUÉ SE HA CONSTRUIDO (DEMO)

# SPRINT 1 – CONFIGURACIÓN Y BASE DEL PROYECTO

## Crear estructura base del repositorio

Se crea la estructura inicial del proyecto para mantener el código organizado:

- Carpeta `backend/` para el servidor
- Carpeta `frontend/`para la parte visual
- Carpeta `docs/` para documentación
- Archivo `.gitignore` para excluir archivos innecesarios (entorno virtual, caché, etc.)

## Backend en Visual Studio

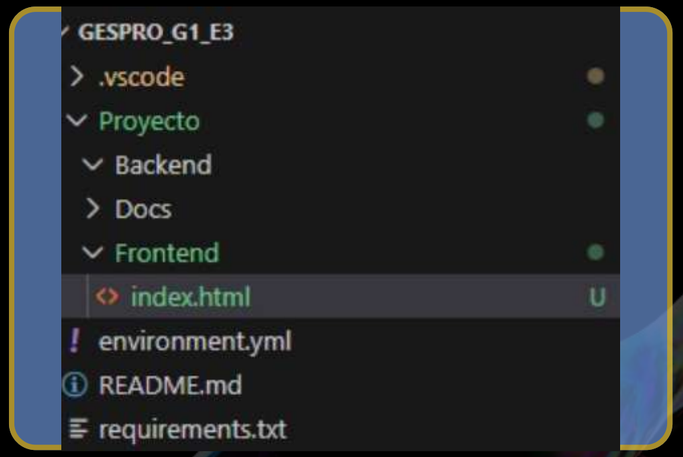
Configuración del entorno Python
Creación de aplicación básica con Flask/FastAPI
Verificación de que el servidor arranca correctamente

## Crear 1 endpoint de prueba

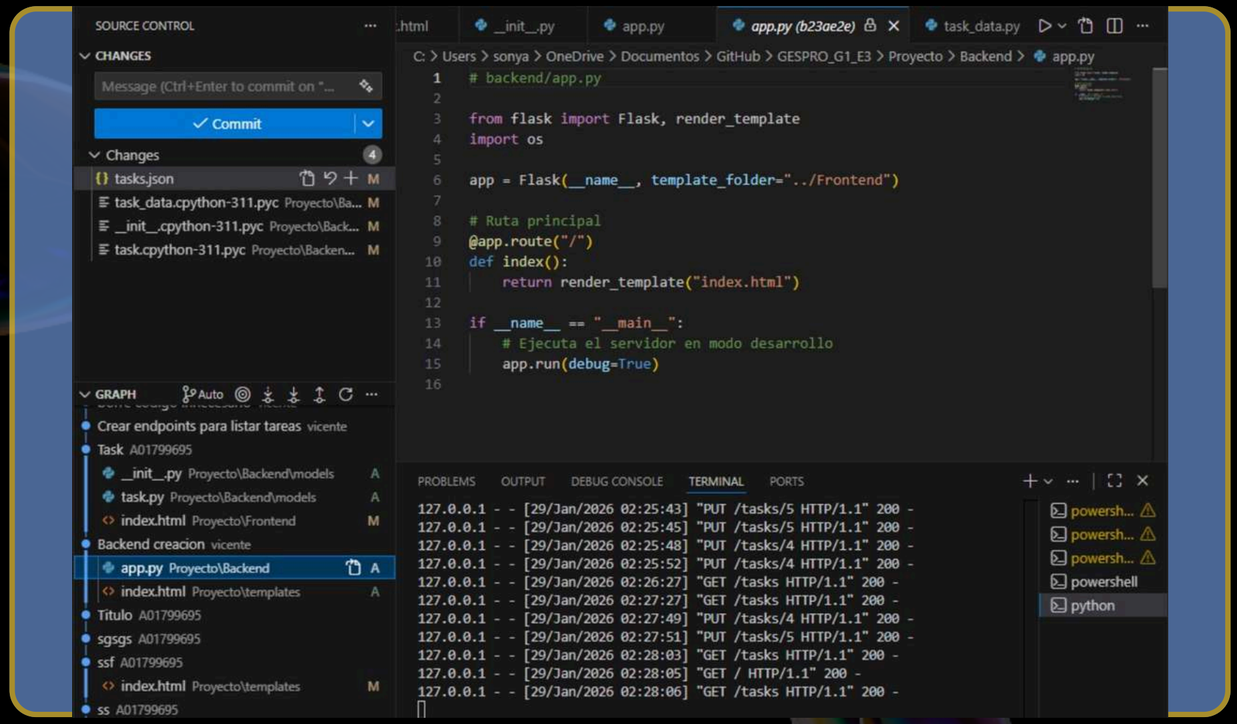
Endpoint `/health` o `/`
Devuelve mensaje o JSON de prueba
Accesible desde navegador

## Frontend

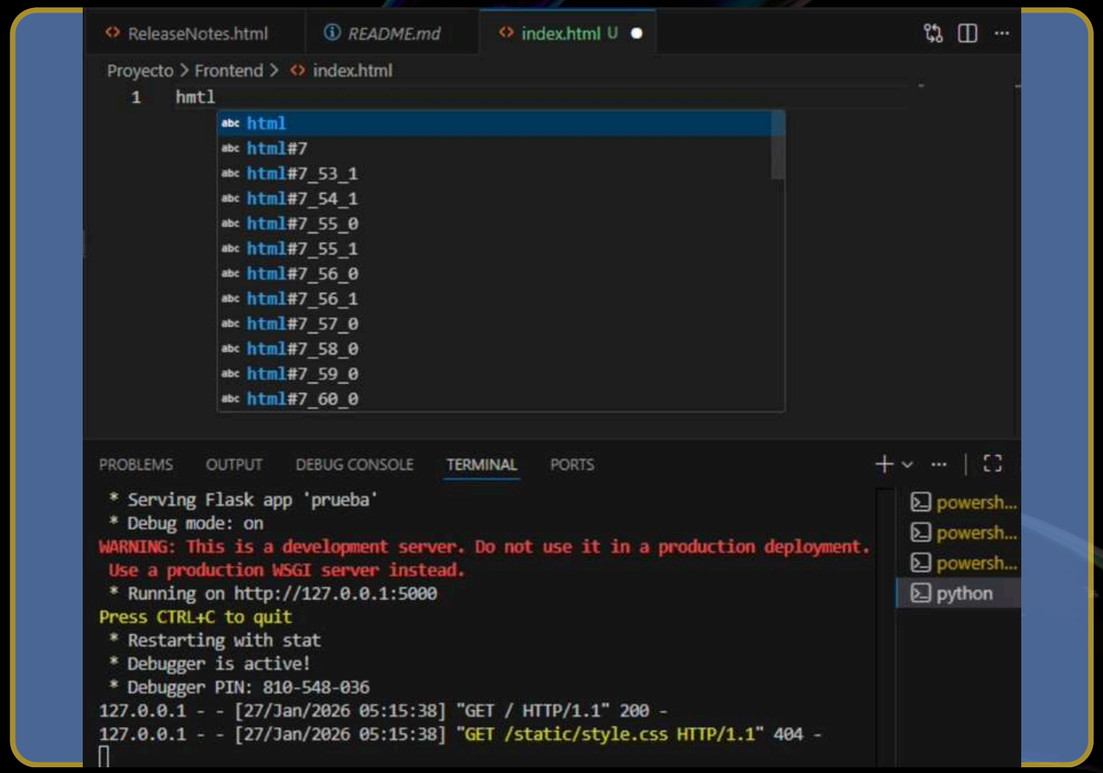
Llamada básica al backend
Mostrar en pantalla el texto devuelto por el endpoint



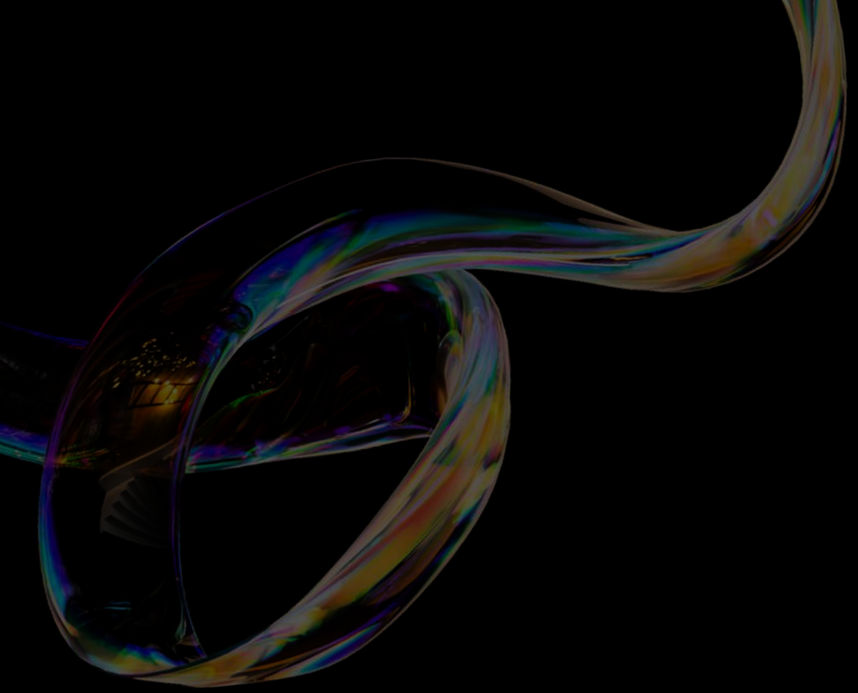

ESTRUCTURA DEL PROYECTO EN VS CODE




CREACION DE APP . PY (APLICACION DE USO)




CREACION DE INDEX (HTML) "PAGINA WEB"



# QUÉ SE HA CONSTRUIDO (DEMO)

## SPRINT 2–

**Resultado del Sprint**

- Comunicación funcional entre frontend y backend
- Validación de que la arquitectura del proyecto funciona

### Conexión Frontend – Backend

Implementación básica entre ambas capas del sistema:

- Uso de ***fetch*** para realizar peticiones al backend.
- Uso del endpoint de prueba creado en el Sprint anterior.
- Verificación de respuesta correcta desde el navegador

**Definición del modelo de datos Task**

Se define la estructura básica de una tarea:

- id
- título
- estado

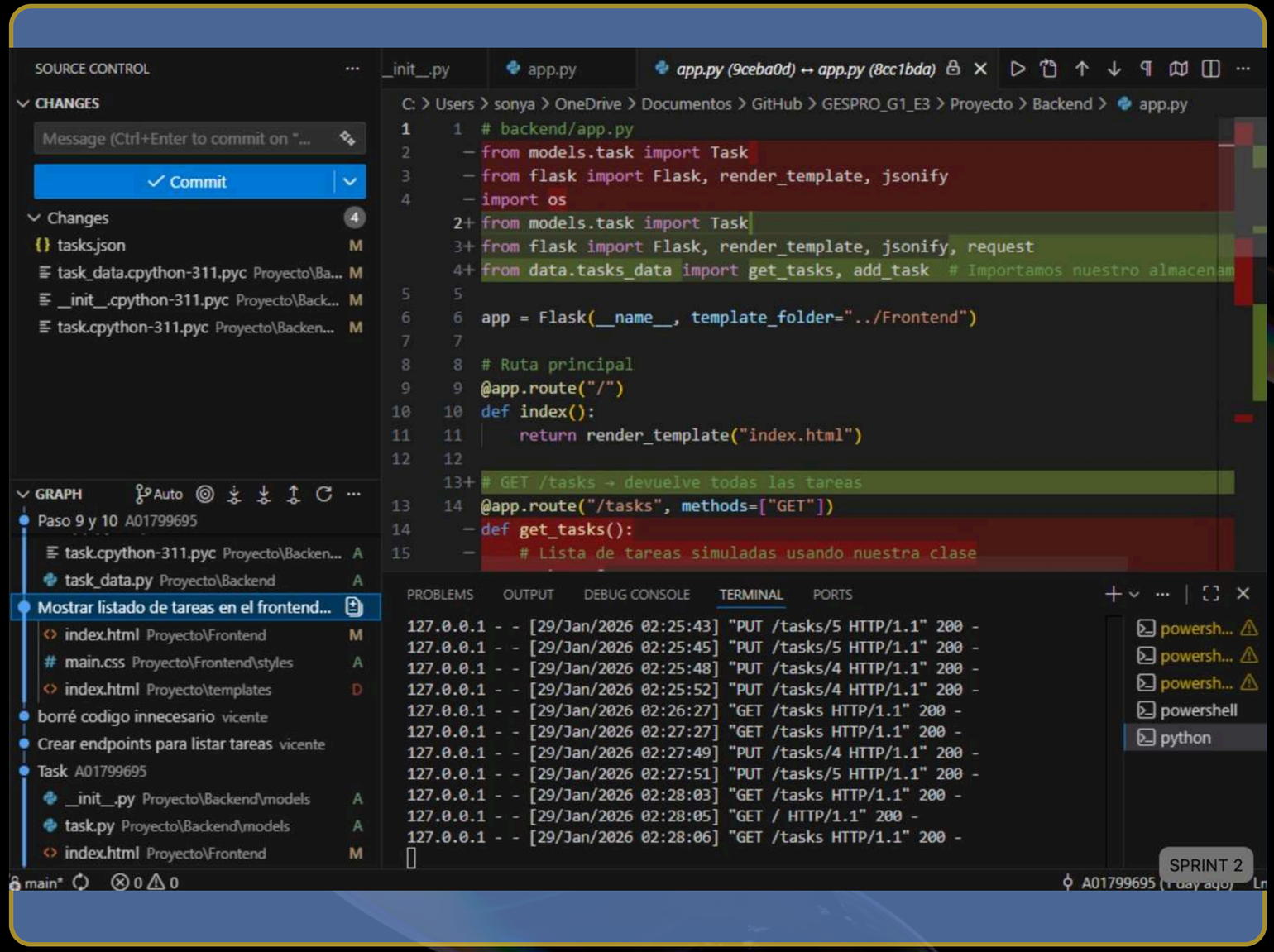
**Crear endpoint GET /tasks**

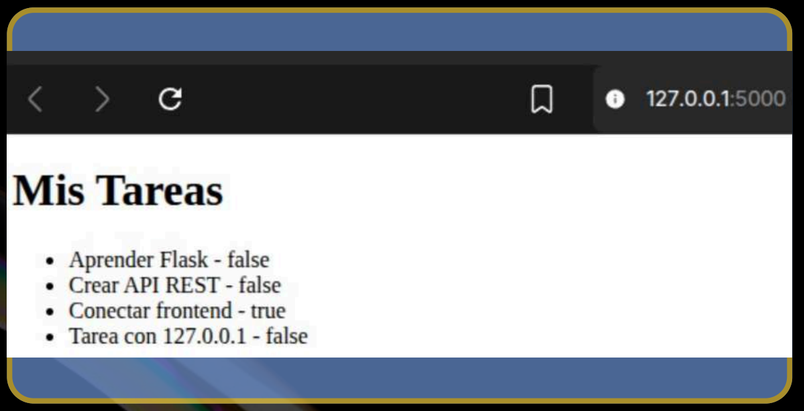
- Devuelve lista de tareas
- Respuesta en formato JSON
- Datos simulados inicialmente

**Renderizado en el frontend**

- Captura de la respuesta del servidor
- Inserción dinámica del contenido en el HTML
- Confirmación del flujo completo:

**Servidor > Cliente**




# CODIGOS DE RESULTADOS DEL SPRINT 2

# QUÉ SE HA CONSTRUIDO (DEMO)

## SPRINT 3 – CONFIGURACIÓN Y BASE DEL PROYECTO

**Crear endpoint POST/ tasks**

**Se implementa la creación de nuevas tareas desde el backend:**

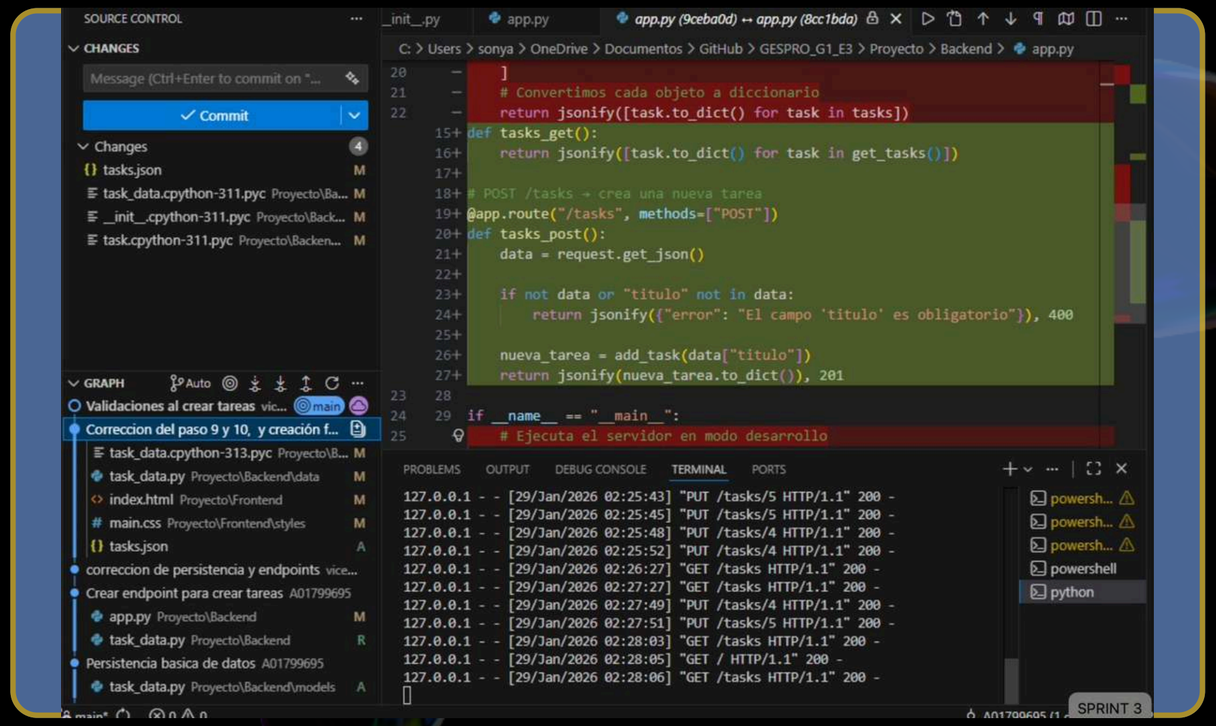
- Recepción de datos en formato JSON.
- Generación automática de id.
- Agregado a lista en memoria.

**Formulario en el frontend**

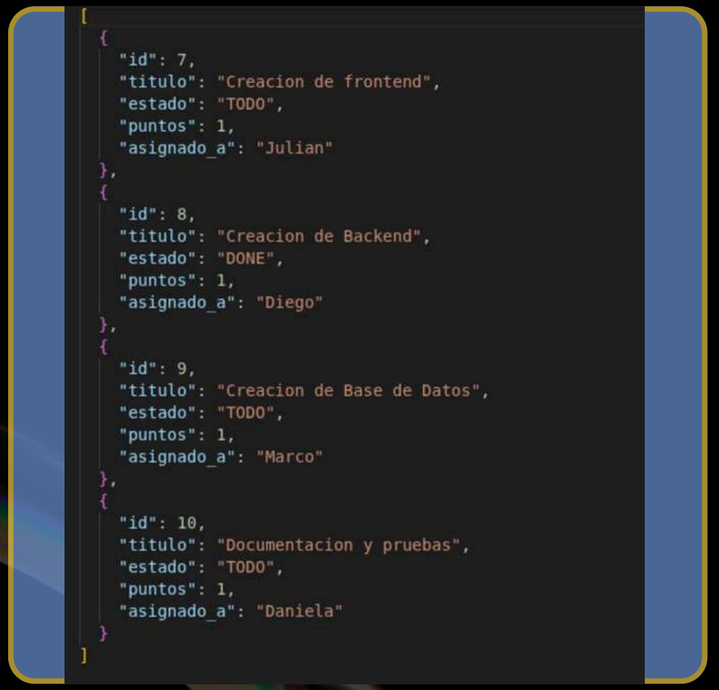
- Input para título de tarea
- Botón de envío
- Envío mediante fetch (POST)

**Actualización automática de la lista**

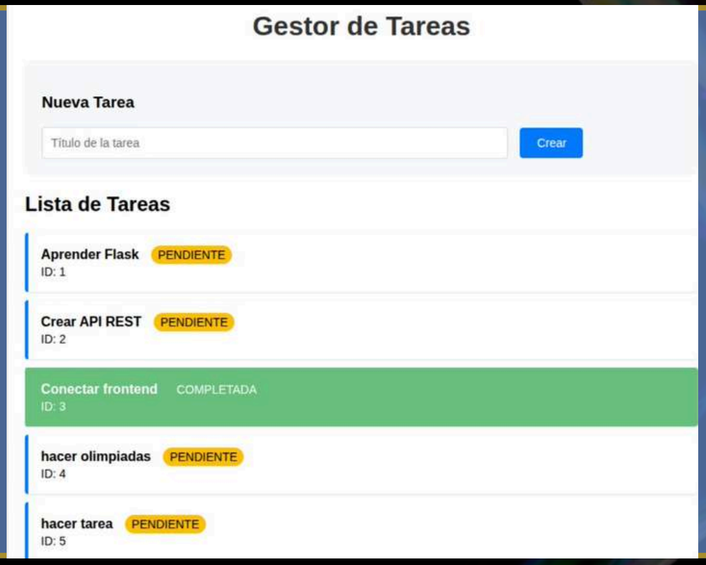
- Refresh de las tareas tras su creación
- Mostrar nueva tarea sin recargar página



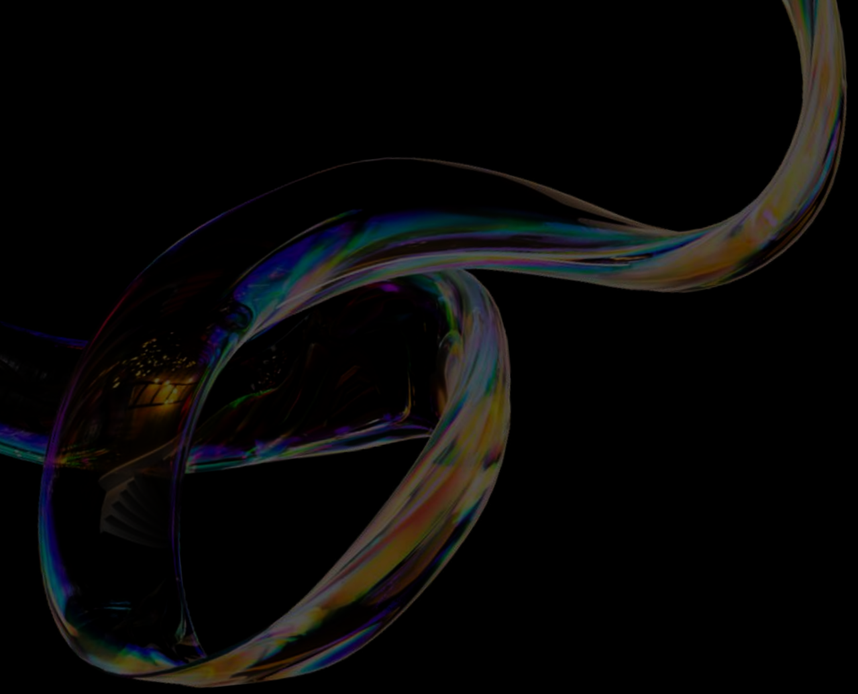

CREACION DE METODO “POST” EN LA RUTA “/”




ALMACENAMIENTO DE DATOS EN JSON




RESULTADO DEL SPRINT



# QUÉ SE HA CONSTRUIDO (DEMO)

## SPRINT 4 – MEJORAR LA FUNCIONALIDAD Y USABILIDAD DEL GESTOR DE TAREAS

### Validaciones al crear tareas

- Evitar títulos vacíos
- Mostrar mensajes de error

### Estado de las tareas

- Implementación de estados (pendiente, en progreso, completada)

### Mejora del frontend

- Interfaz más clara
- Organización visual por columnas

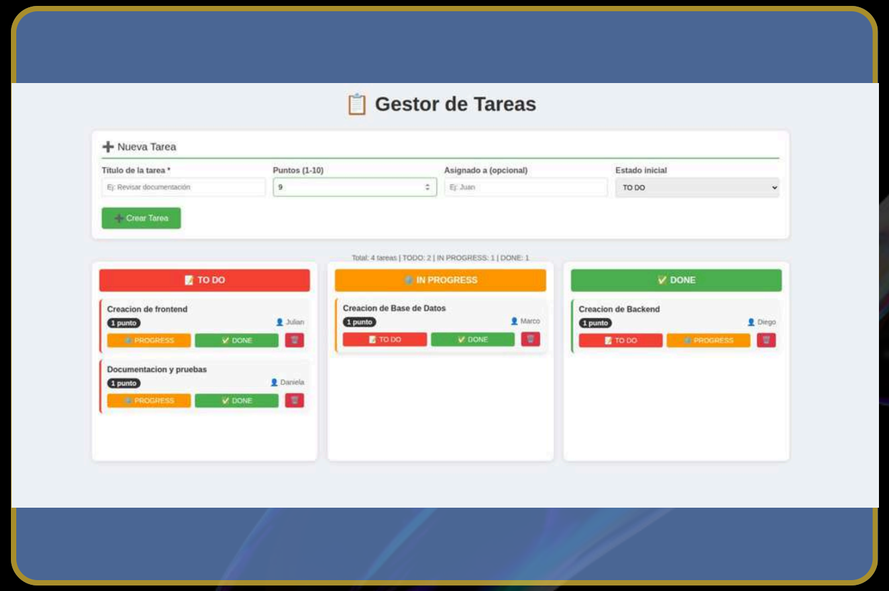
### Mover tareas entre columnas

- Cambio de estado reflejado dinámicamente

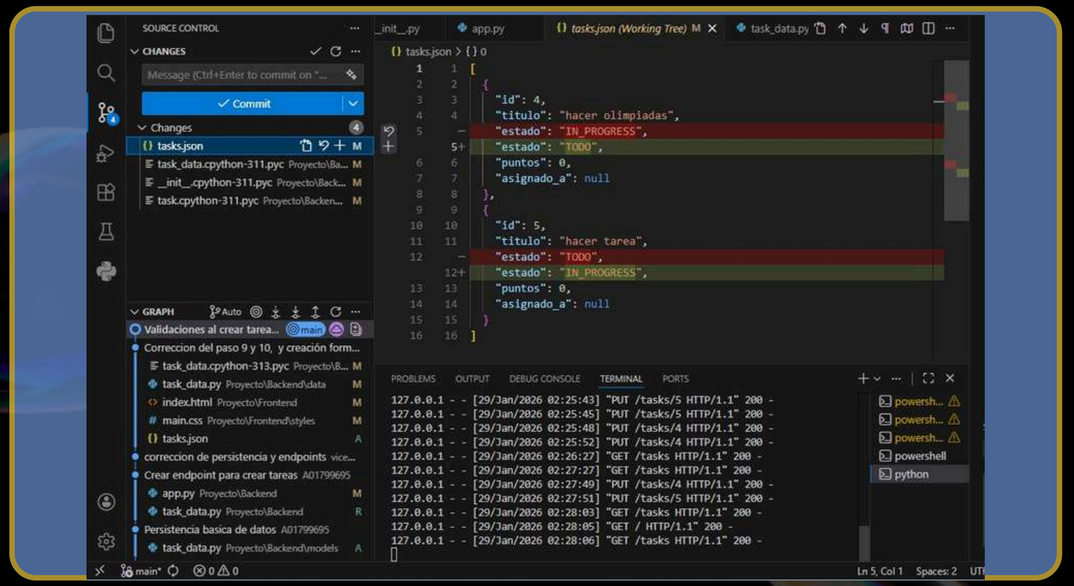
### Eliminar tareas

- Botón para eliminar tareas
- Actualización automática en pantalla

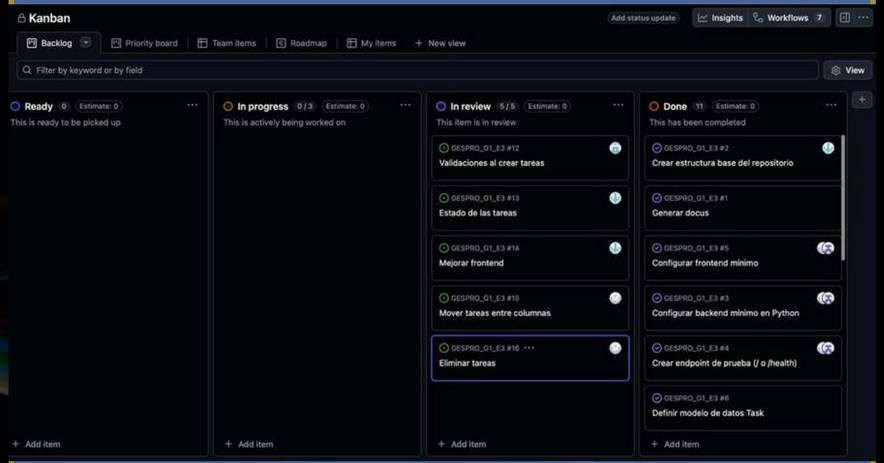
## ESTRUCTURA DEL PROYECTO EN VS CODE

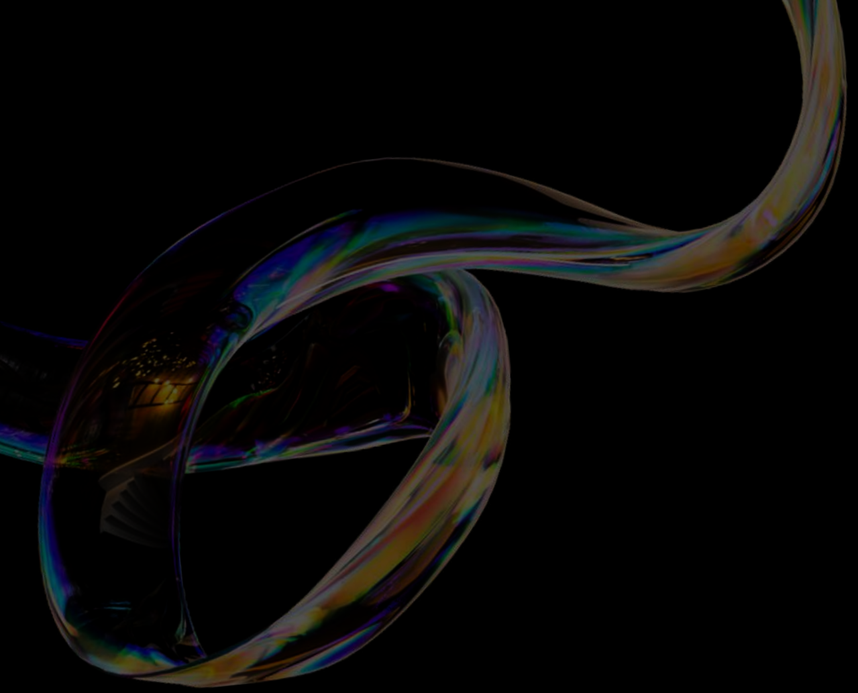



## ESTRUCTURA DEL PROYECTO EN VS CODE




## KANBAN SPRINT 4





# QUÉ SE HA CONSTRUIDO (DEMO)

## SPRINT 5

### Mejora de estructura y organización

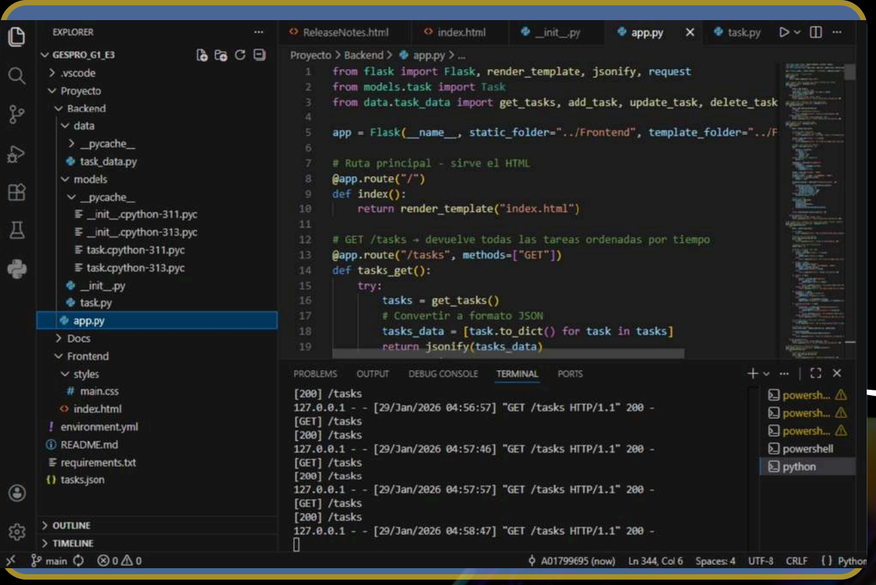
- Limpieza de código
- Separación de lógica
- Revisión de commits

### Documentación inicial

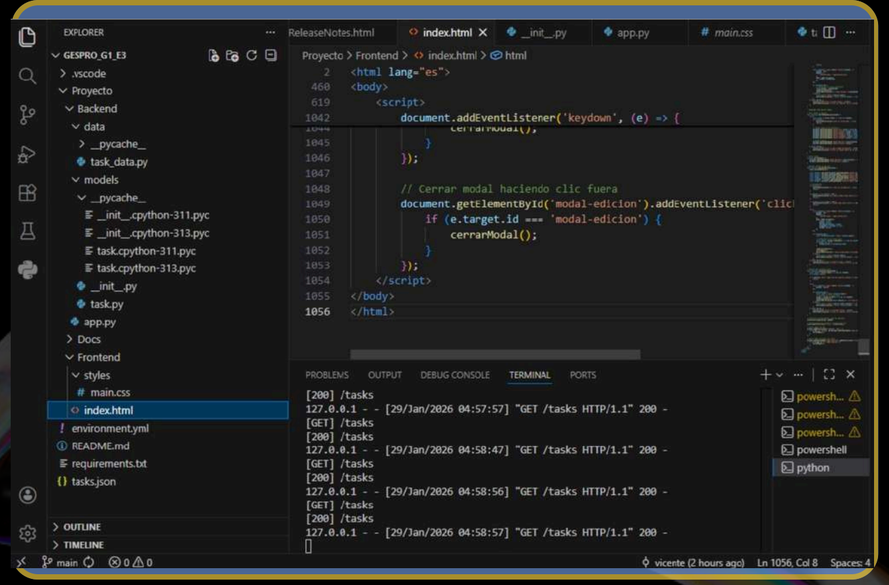
- Creación de README.md
- Instrucciones para ejecutar el proyecto en local
- Descripción de funcionalidades implementadas

**Se implementan reglas de negocio simples:**

- Evitar títulos vacíos
- Control de errores en backend
- Mensajes de error al usuario



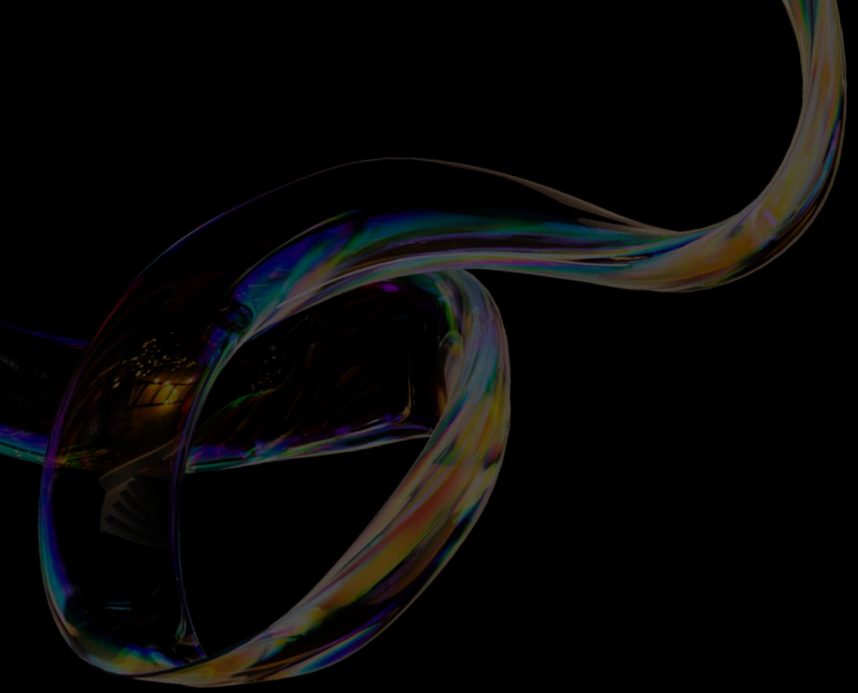

APP . PY




HTML


TABLERO INICIAL


EDICIÓN Y GESTIÓN DE TAREA



# QUÉ SE HA CONSTRUIDO (DEMO)

## SPRINT 6

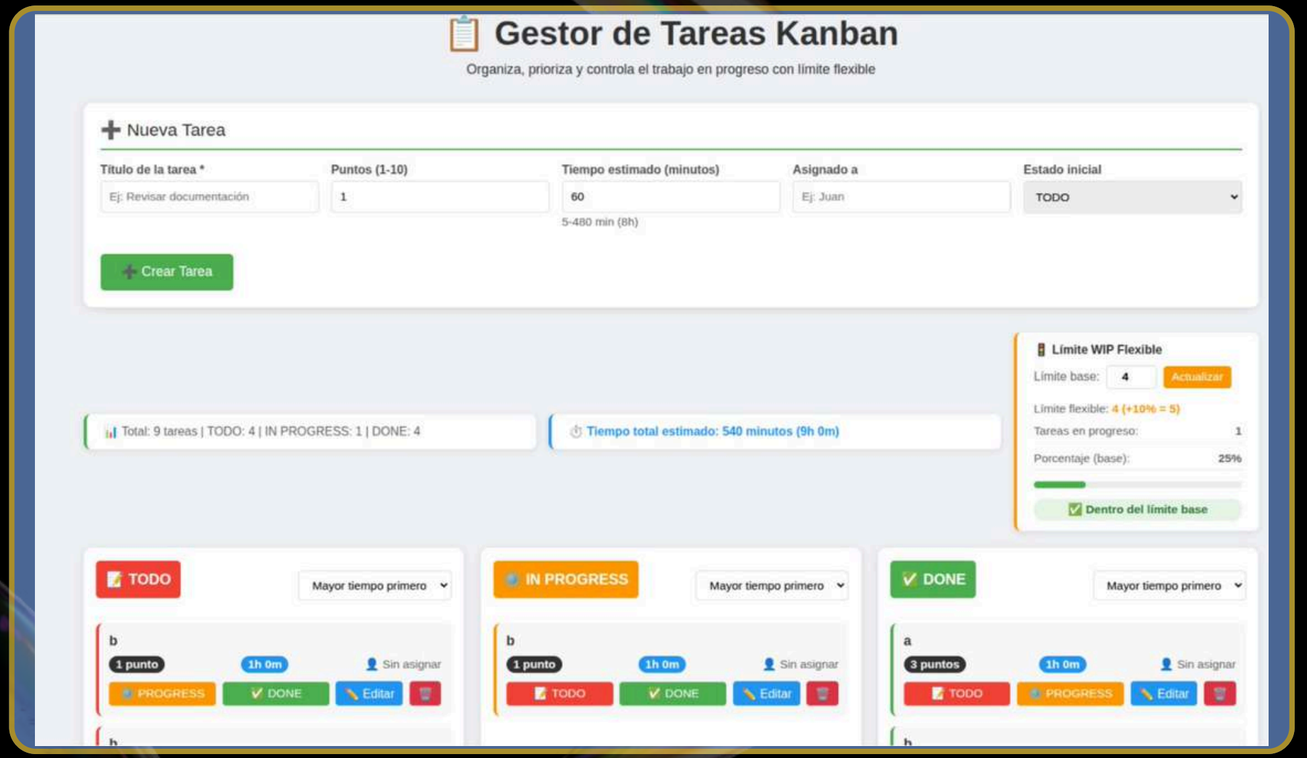
**Persistencia de tareas en fichero JSON**

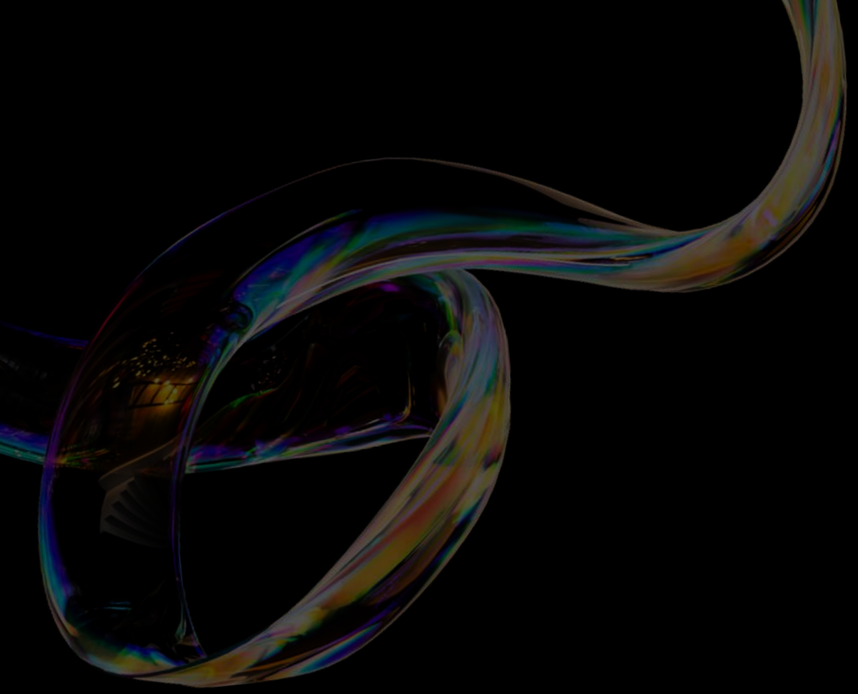
- bbdd/tareas.jsn
- gitignore
- Límite para WIP
- Flexibilidad en el límite WIP

```html
<!-- Contadores y WIP -->
<div class="contadores-container">
    <div class="contador-tareas" id="contador-tareas">
        📊 Total: 0 tareas | TODO: 0 | IN PROGRESS: 0 | DONE: 0
    </div>
    <div class="tiempo-total" id="tiempo-total">
        ⏱️ Tiempo total estimado: 0 minutos
    </div>
    <div class="wip-container" id="wip-container">
        <div class="wip-header">
            <span>🚦 Límite WIP Flexible</span>
        </div>
        <div class="wip-limite">
            <span>Límite base:</span>
            <input type="number" id="wip-limit" class="wip-input" min="1" max="20" value="5">
            <button class="btn-wip" onclick="actualizarWIP()">Actualizar</button>
        </div>
        <div class="wip-info">
            <span>Límite flexible: <span id="wip-flexible" class="wip-flexible">5 (+10% = 5)</span></span>
        </div>
        <div class="wip-actual">
            <span>Tareas en progreso:</span>
            <span id="wip-actual" class="wip-value">0</span>
        </div>
        <div class="wip-actual">
            <span>Porcentaje (base):</span>
            <span id="wip-porcentaje" class="wip-value">0%</span>
        </div>
        <div class="wip-progress-bar">
            <div id="wip-progress-fill" class="wip-progress-fill" style="width: 0%"></div>
```

CÓDIGO

SALIDA DEL SPRINT





# QUÉ SE HA CONSTRUIDO (DEMO)

## SPRINT 7

**Añadir fichero con user/pass/nombre**

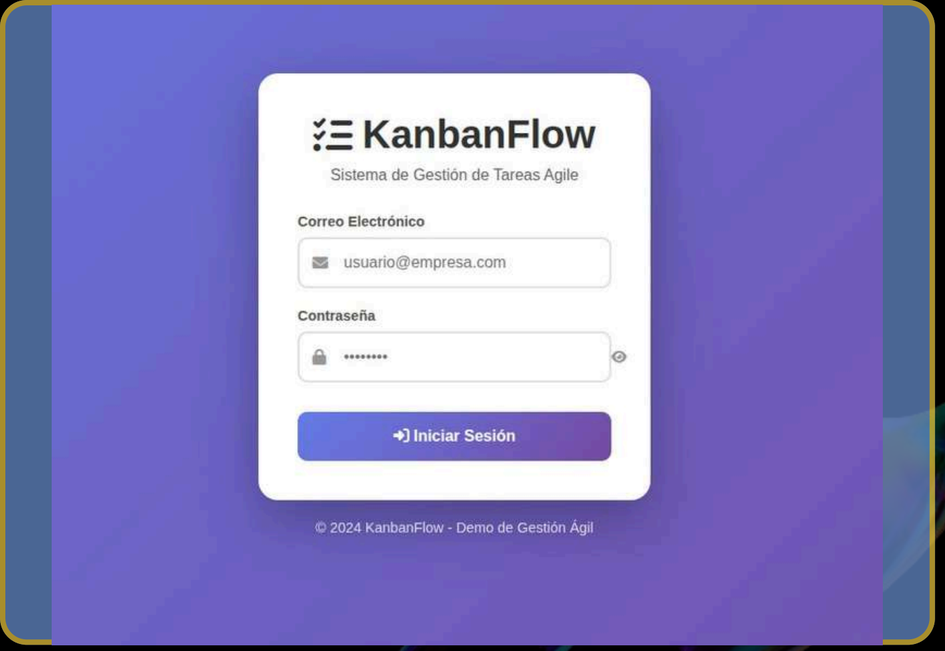
- bbdd/usuarios.txt

**Modificar frontend con desplegable usuarios/equipo**

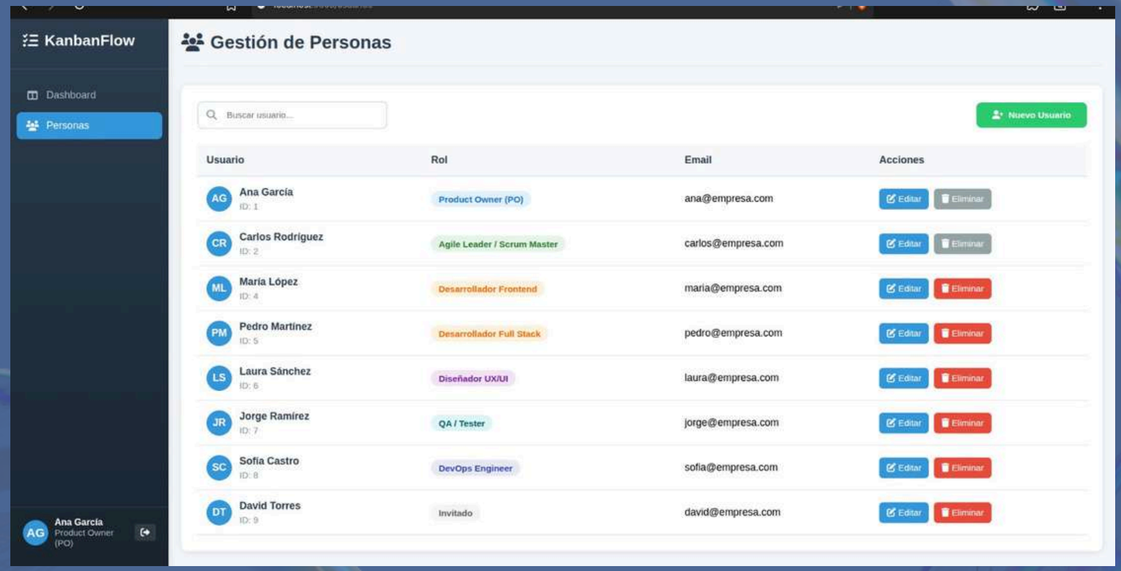
**Añadir roles al equipo**

- product owner
- normal
- invitado

**Inicio de sesión, usuario**



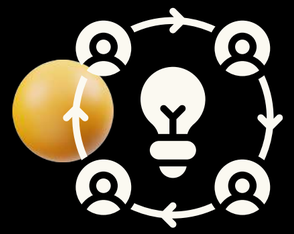

VISUALIZACIÓN INICIO DE SESIÓN




TABLERO INICIAL

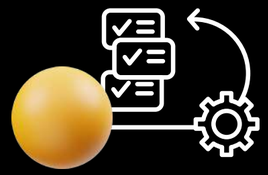
# METODOLOGÍA DE TRABAJO

## ¿Cómo se ha trabajado con Scrum?



### Gestión en GitHub

- Uso de GitHub Projects para organizar tareas
- Control de versiones mediante commits frecuentes
- Colaboración en repositorio compartido
- Documentación en README y Wiki



### Trabajo con Scrum

- Trabajo en 5 sprints de 1h 10m
- Sprint Planning al inicio de cada iteración
- Daily Scrums breves durante el desarrollo
- Sprint Review con demostración del incremento
- Sprint Retrospective para identificar mejoras

# QUÉ DIFICULTADES HAN ENCONTRADO.

La parte de la programación resultó especialmente complicada, ya que solo una persona comprendía con claridad qué acciones debían realizarse y cómo estructurar el desarrollo. Esto generó una alta dependencia y dificultades para avanzar en equipo, además de que el código presentaba fallas frecuentes, lo que provocó retrabajo constante y retrasos en el progreso del proyecto.

# GRACIAS