



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.



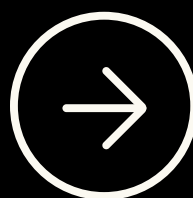
**Aplicar reglas de
negocio realistas.**



**Trabajar siguiendo los
principios y eventos de
Scrum.**



QUÉ SE HA CONSTRUIDO (DEMO)

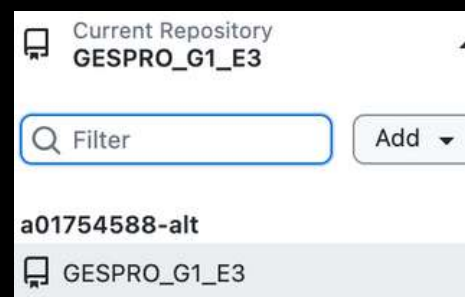


INICIO

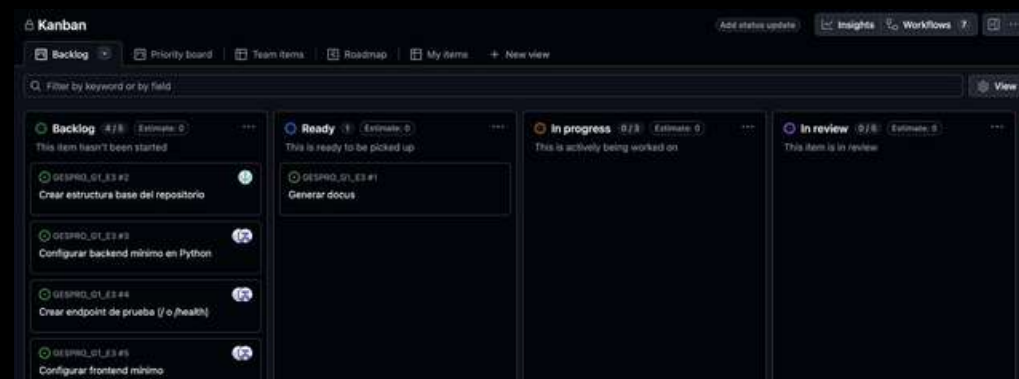
- Configuración del entorno



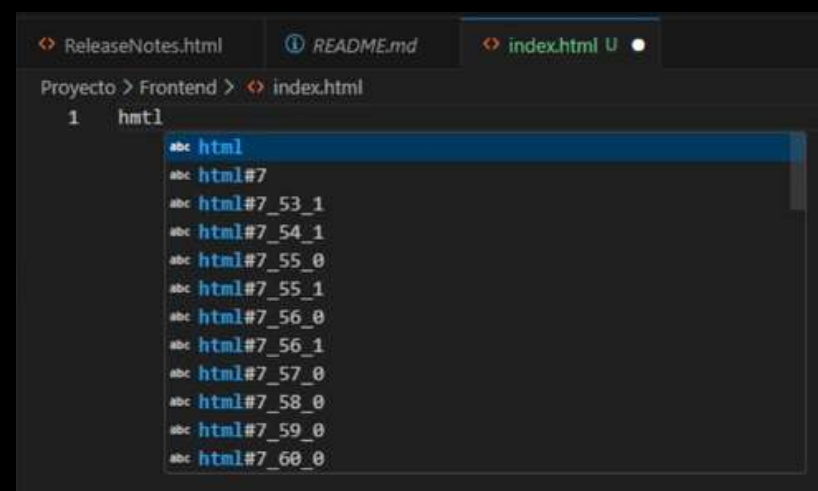
- Crear repository en GitHub y agregar colaboradores



- 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

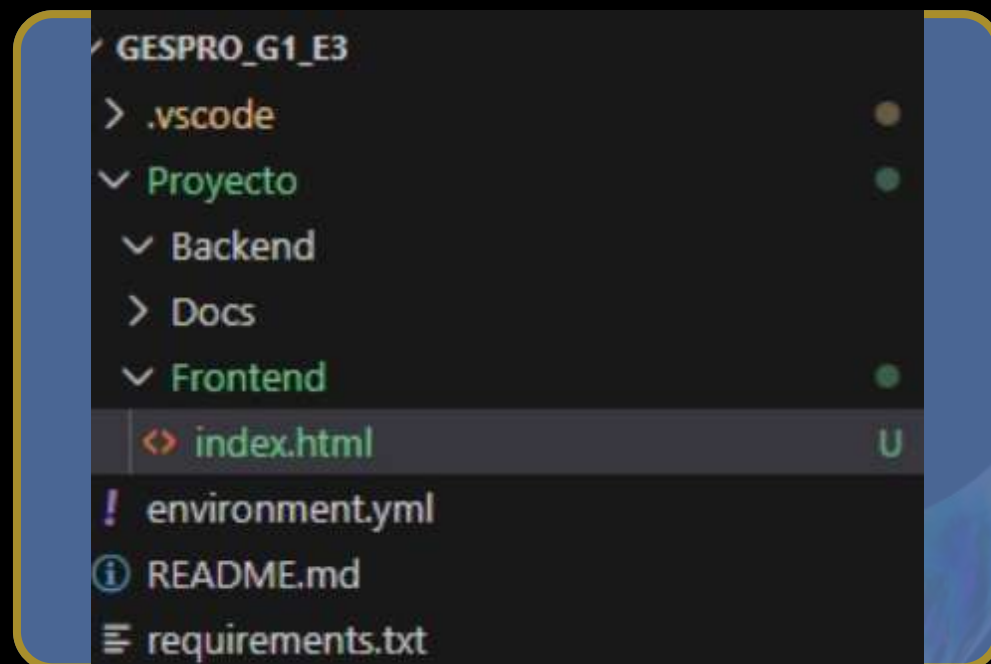
Accesible desde navegador

Frontend

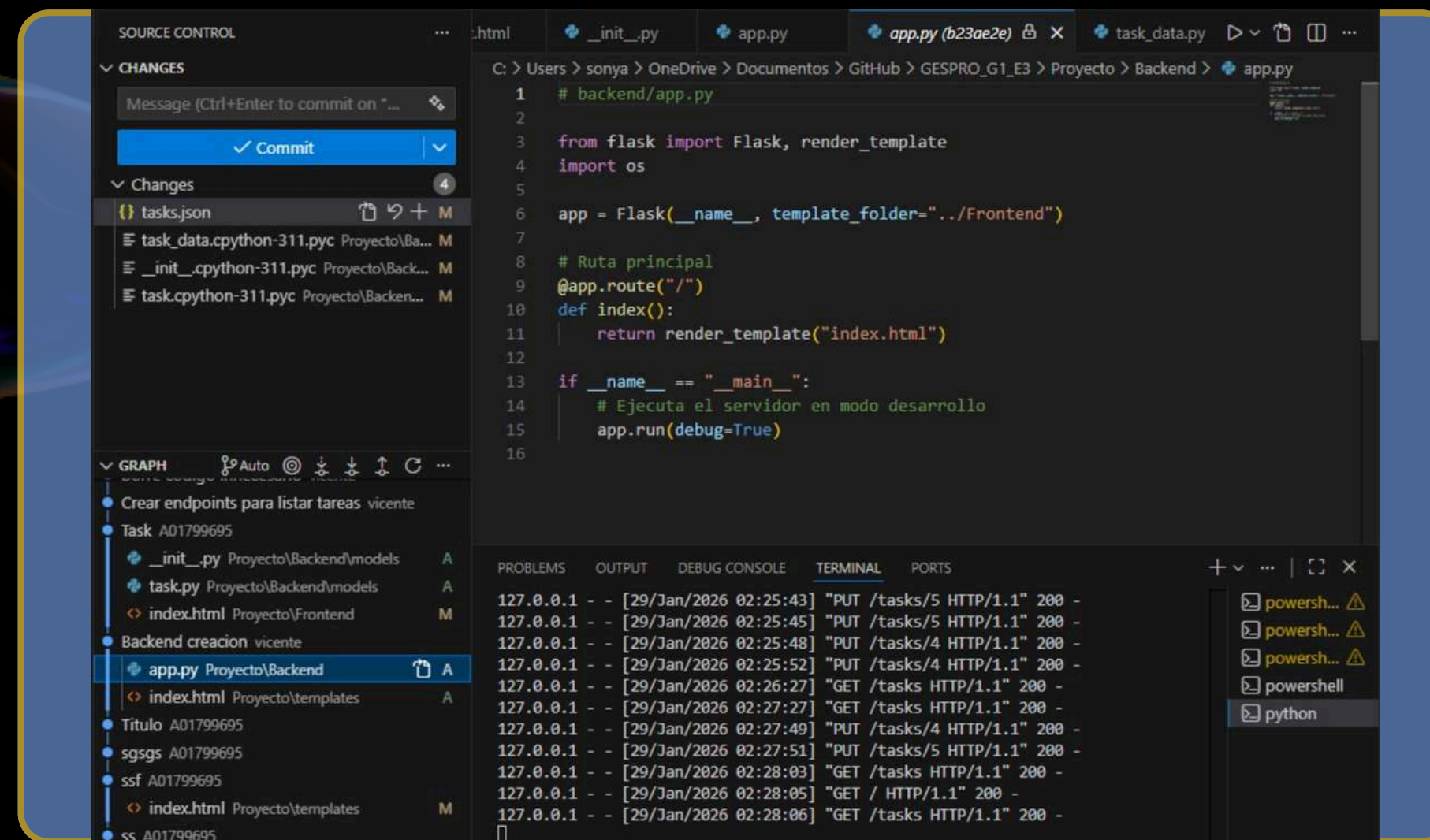
Llamada básica al backend

Mostrar en pantalla el texto devuelto por el endpoint

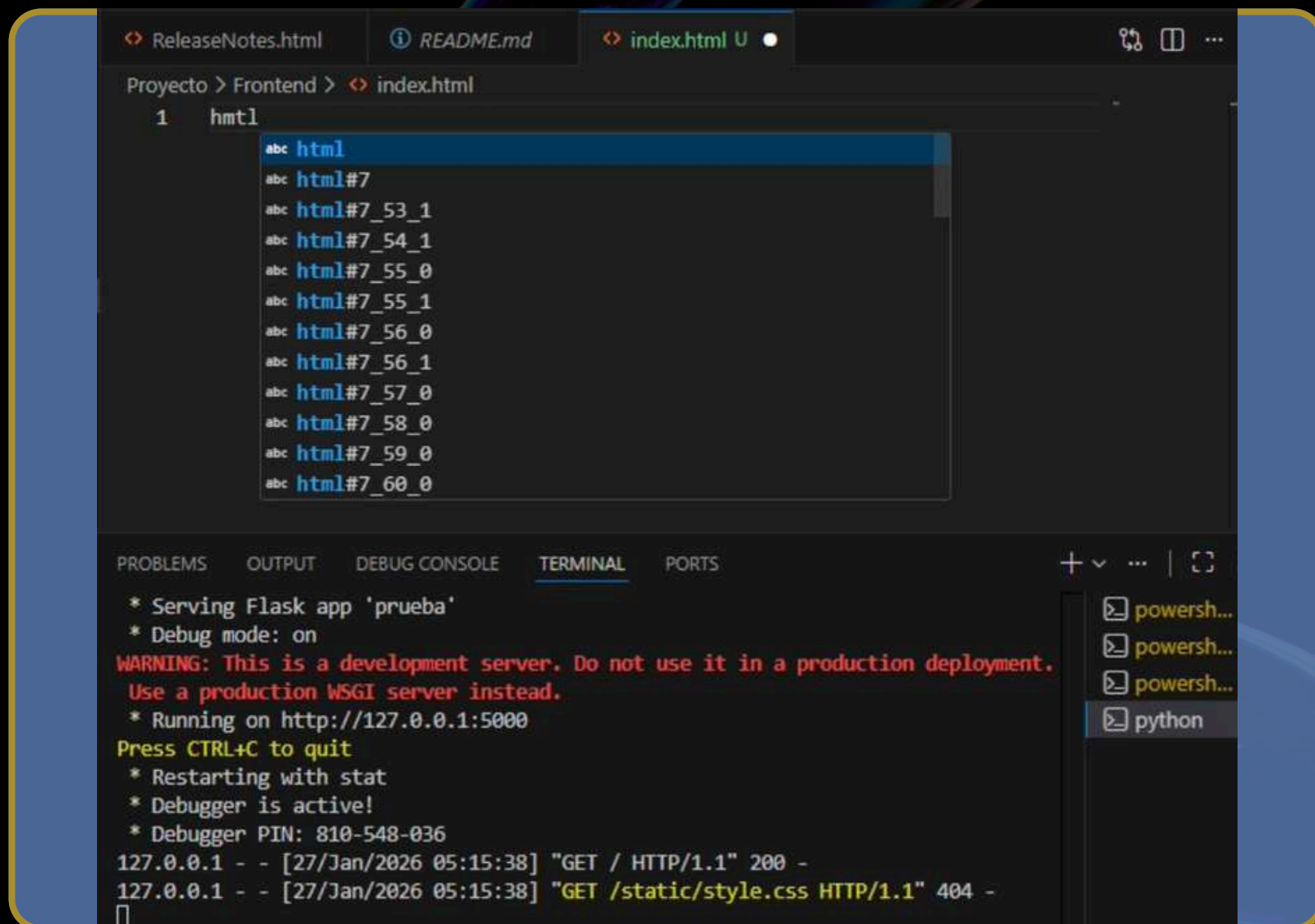




ESTRUCTURA DEL PROYECTO EN VS CODE



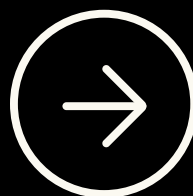
CREACION DE APP . PY (APLICACION DE USO)



CREACION DE INDEX (HTML) "PAGINA WEB"



QUÉ SE HA CONSTRUIDO (DEMO)



SPRINT 2–

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

Resultado del Sprint

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

SOURCE CONTROL

CHANGES

Message (Ctrl+Enter to commit on "...")

Commit

Changes

tasks.json

task_data.cpython-311.pyc Proyecto\Ba...

init.cpython-311.pyc Proyecto\Back...

task.cpython-311.pyc Proyecto\Backen...

GRAPH

Paso 9 y 10 A01799695

task.cpython-311.pyc Proyecto\Backen...

task_data.py Proyecto\Backend

Mostrar listado de tareas en el frontend...

index.html Proyecto\Frontend

main.css Proyecto\Frontend\styles

index.html Proyecto\templates

borré código innecesario vicente

Crear endpoints para listar tareas vicente

Task A01799695

init.py Proyecto\Backend\models

task.py Proyecto\Backend\models

index.html Proyecto\Frontend

app.py

app.py (9ceba0d) ↔ app.py (8cc1bda)

C:\> Users > sonya > OneDrive > Documentos > GitHub > GESPRO_G1_E3 > Proyecto > Backend > app.py

```
1 1 # backend/app.py
2 - from models.task import Task
3 - from flask import Flask, render_template, jsonify
4 - import os
2+ from models.task import Task
3+ from flask import Flask, render_template, jsonify, request
4+ from data.tasks_data import get_tasks, add_task # Importamos nuestro almacenam
5
6 app = Flask(__name__, template_folder="../Frontend")
7
8 # Ruta principal
9 @app.route("/")
10 def index():
11     return render_template("index.html")
12
13+ # GET /tasks → devuelve todas las tareas
14 @app.route("/tasks", methods=["GET"])
15 - def get_tasks():
16     # Lista de tareas simuladas usando nuestra clase
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

127.0.0.1 - - [29/Jan/2026 02:25:43] "PUT /tasks/5 HTTP/1.1" 200 -

127.0.0.1 - - [29/Jan/2026 02:25:45] "PUT /tasks/5 HTTP/1.1" 200 -

127.0.0.1 - - [29/Jan/2026 02:25:48] "PUT /tasks/4 HTTP/1.1" 200 -

127.0.0.1 - - [29/Jan/2026 02:25:52] "PUT /tasks/4 HTTP/1.1" 200 -

127.0.0.1 - - [29/Jan/2026 02:26:27] "GET /tasks HTTP/1.1" 200 -

127.0.0.1 - - [29/Jan/2026 02:27:27] "GET /tasks HTTP/1.1" 200 -

127.0.0.1 - - [29/Jan/2026 02:27:49] "PUT /tasks/4 HTTP/1.1" 200 -

127.0.0.1 - - [29/Jan/2026 02:27:51] "PUT /tasks/5 HTTP/1.1" 200 -

127.0.0.1 - - [29/Jan/2026 02:28:03] "GET /tasks HTTP/1.1" 200 -

127.0.0.1 - - [29/Jan/2026 02:28:05] "GET / HTTP/1.1" 200 -

127.0.0.1 - - [29/Jan/2026 02:28:06] "GET /tasks HTTP/1.1" 200 -

powerash...

powerash...

powerash...

powershell

python

SPRINT 2

127.0.0.1:5000

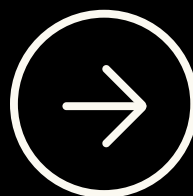
Mis Tareas

- Aprender Flask - false
- Crear API REST - false
- Conectar frontend - true
- Tarea con 127.0.0.1 - false

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:

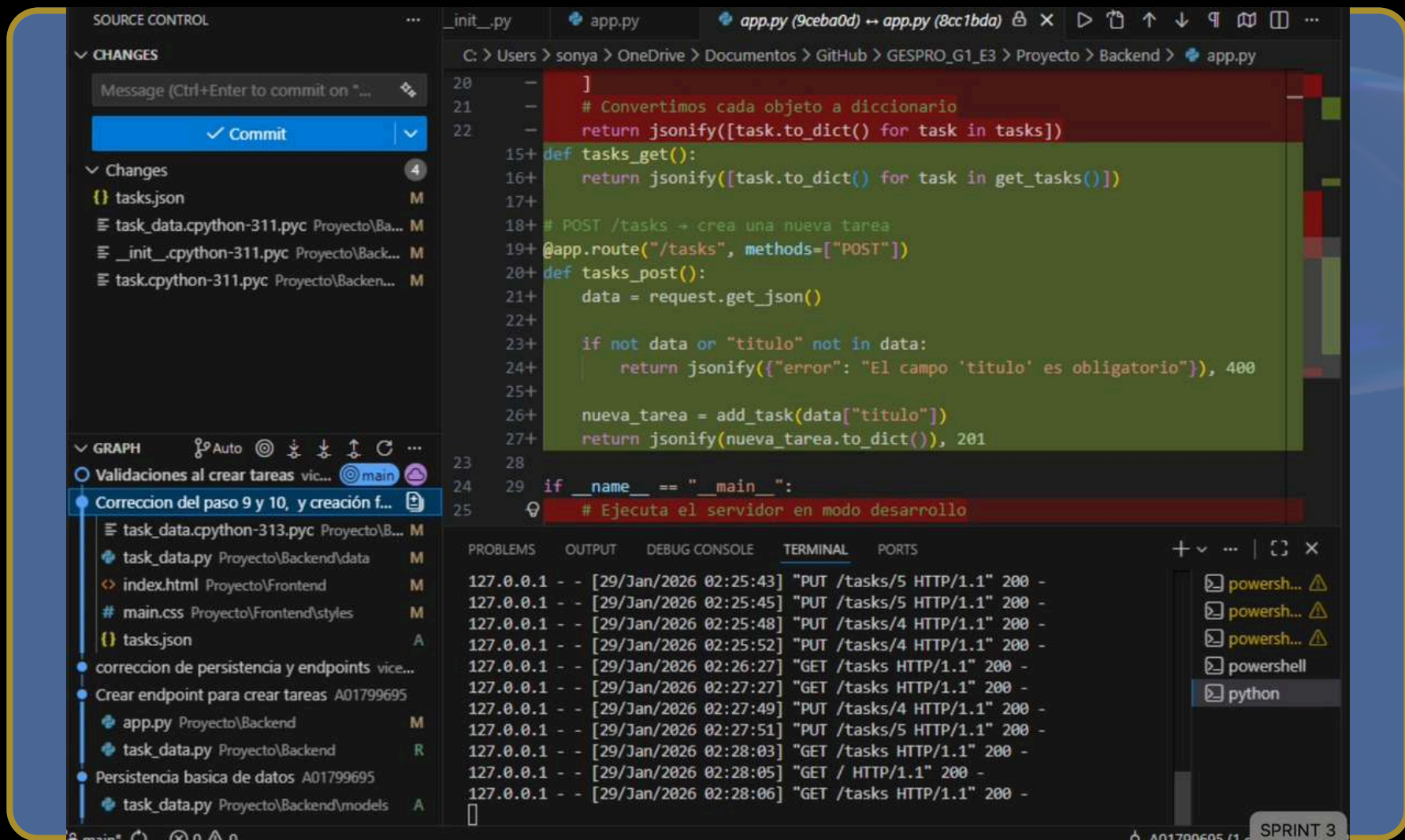
- Recepción de datos en formato JSON.
- Generación automática de id.
- Agregado a lista en memoria.

Formulario en el frontend

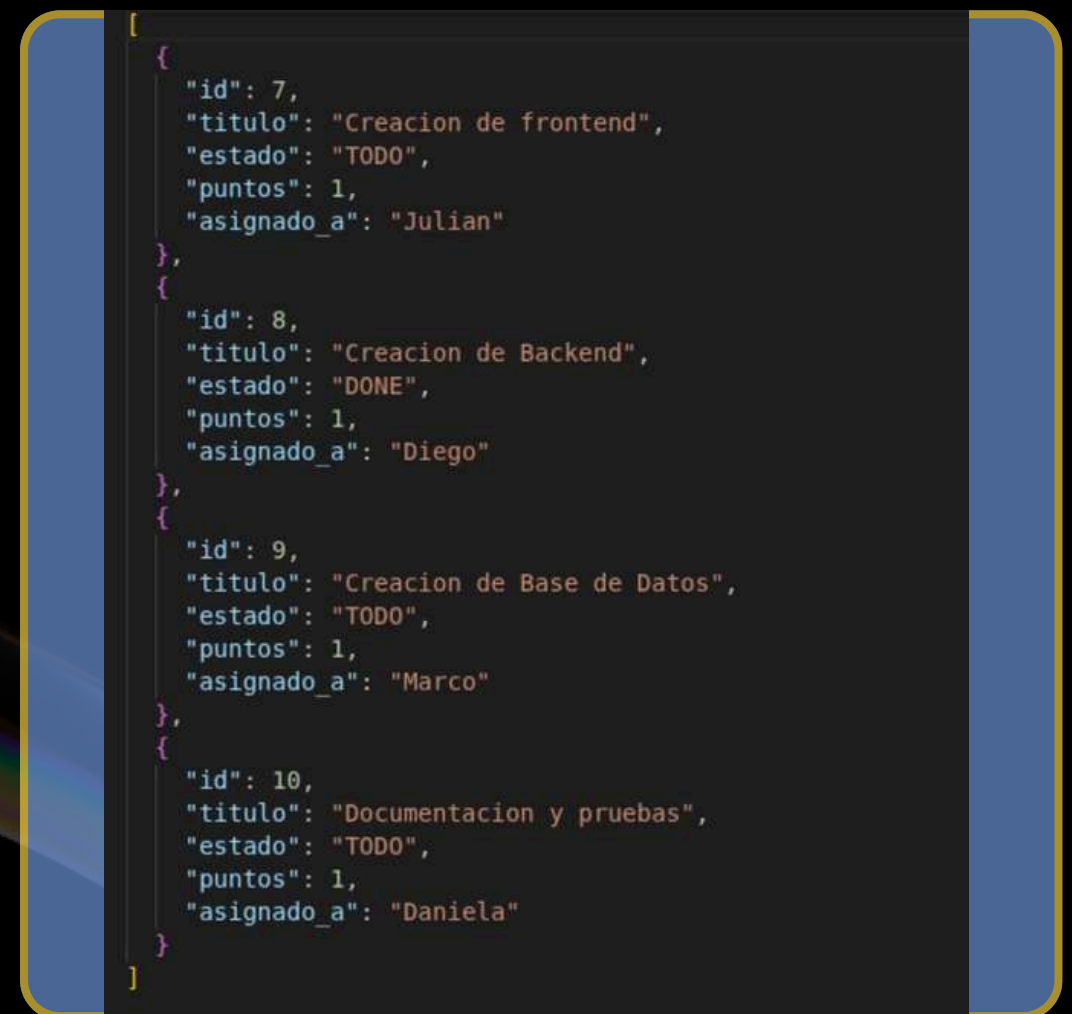
- Input para título de tarea
- Botón de envío
- Envío mediante fetch (POST)

Actualización automática de la lista

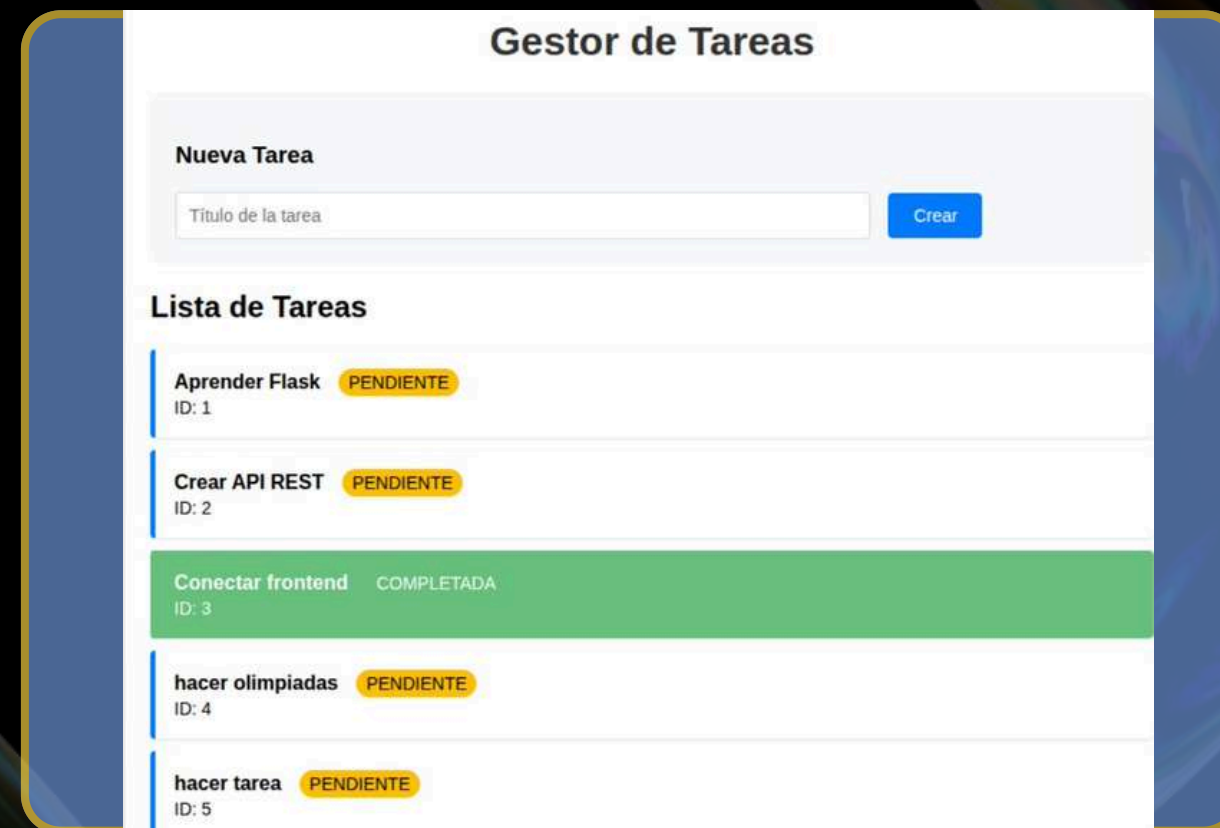
- Refresh de las tareas tras su creación
- Mostrar nueva tarea sin recargar página



CREACION DE METODO "POST" EN LA RUTA "/"



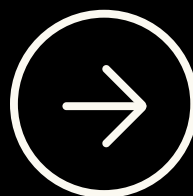
ALMACENAMIENTO DE DATOS EN JSON



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

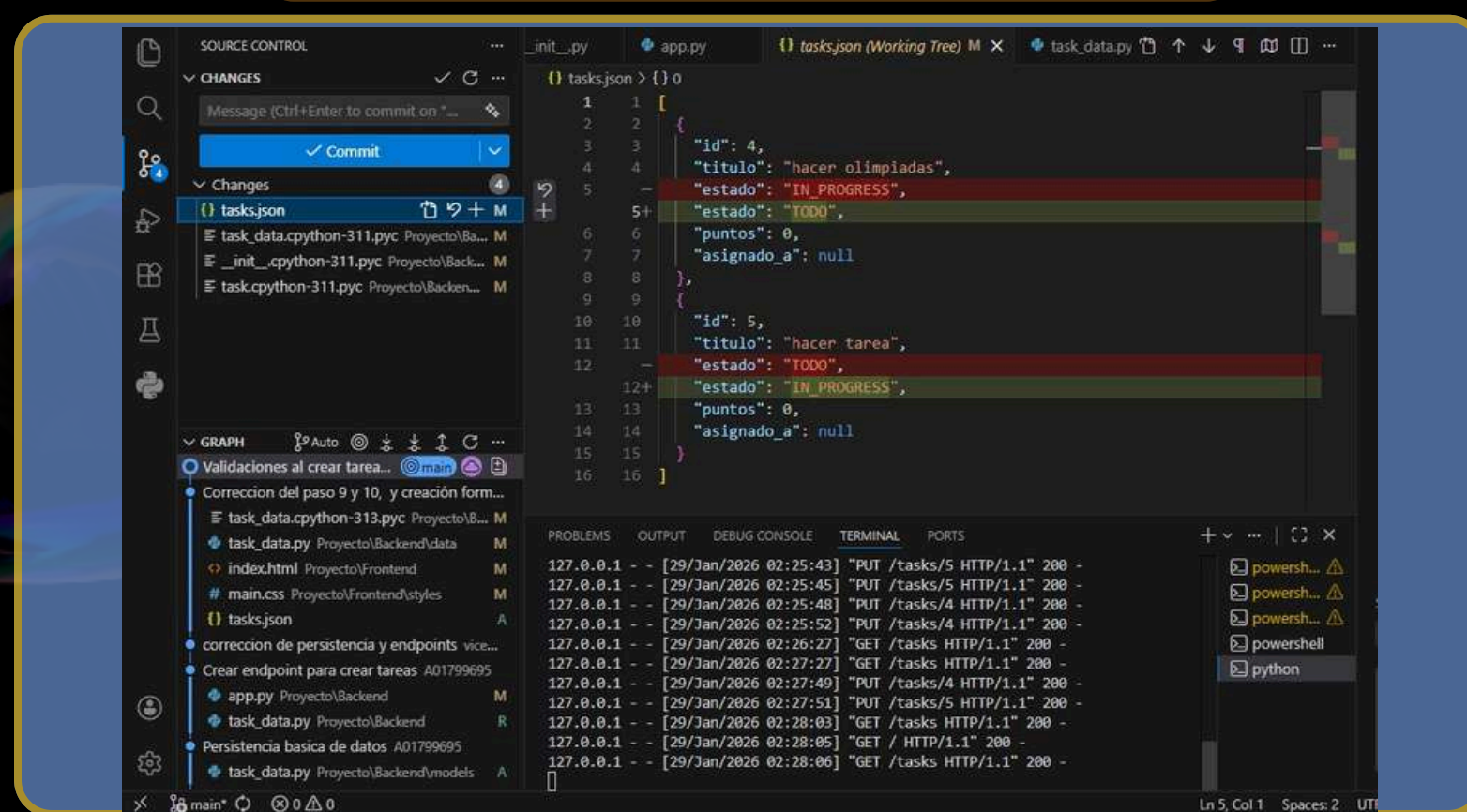
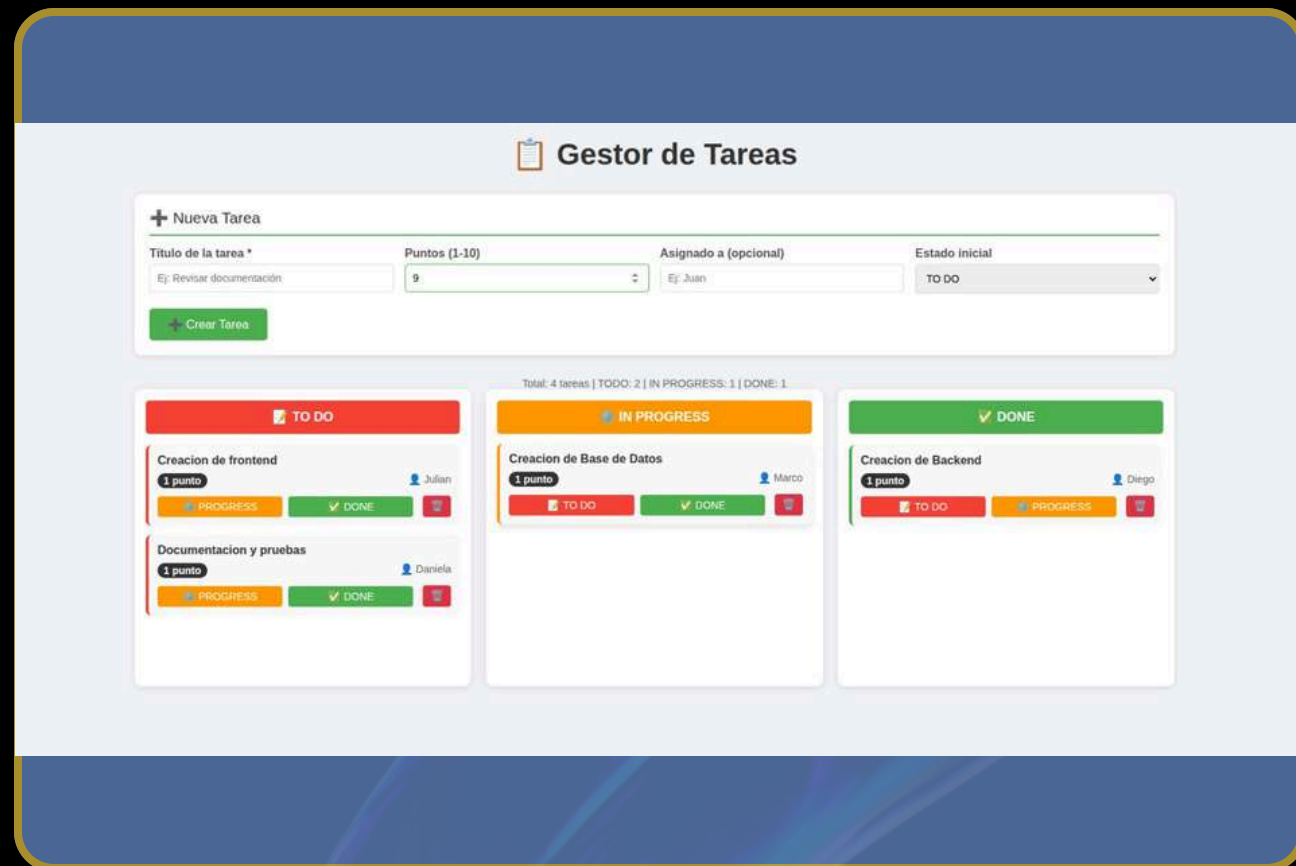
- Cambio de estado reflejado dinámicamente

Eliminar tareas

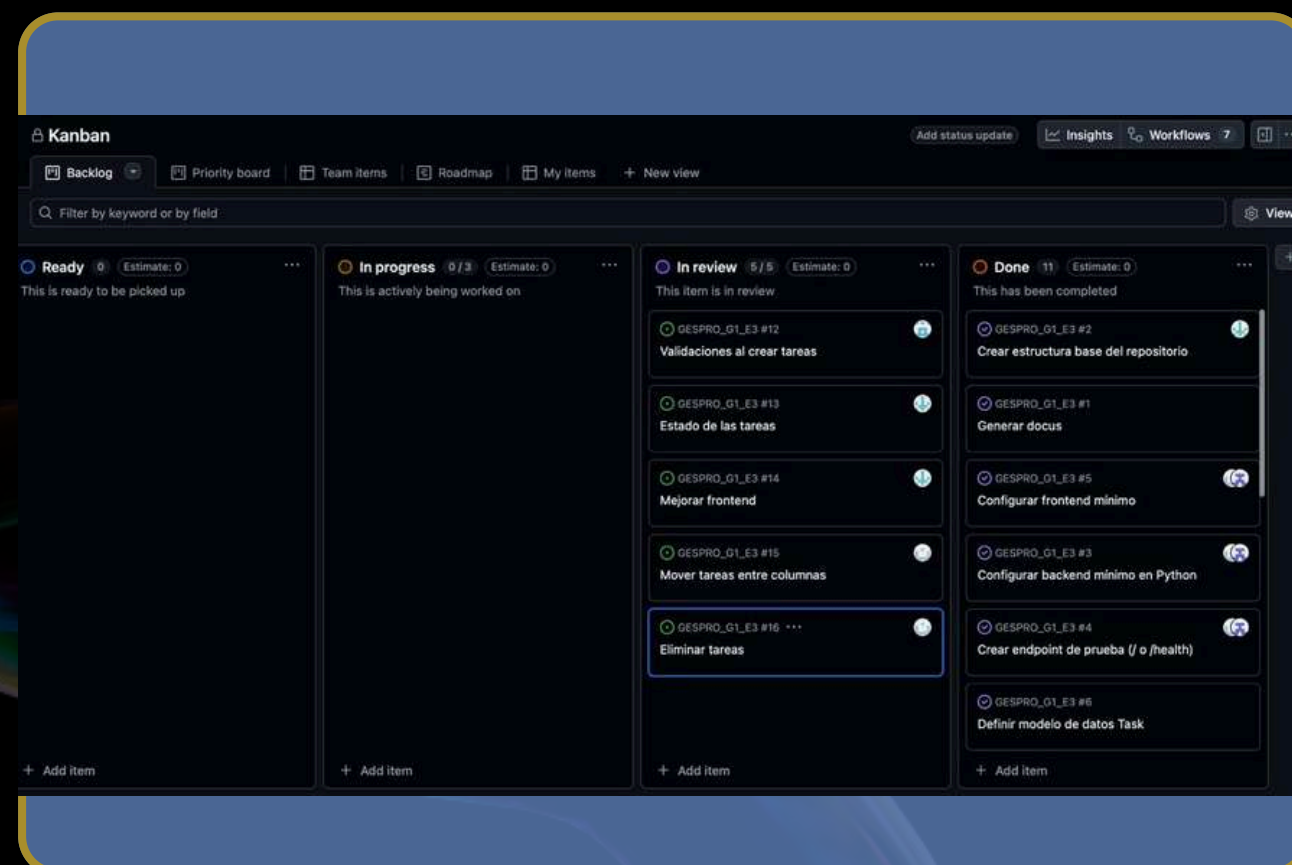
- Botón para eliminar tareas
- Actualización automática en pantalla

ESTRUCTURA DEL PROYECTO EN VS CODE

ESTRUCTURA DEL PROYECTO EN VS CODE

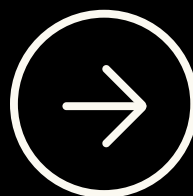


KANBAN SPRINT 4





QUÉ SE HA CONSTRUIDO (DEMO)



SPRINT 5

Mejora de estructura y organización

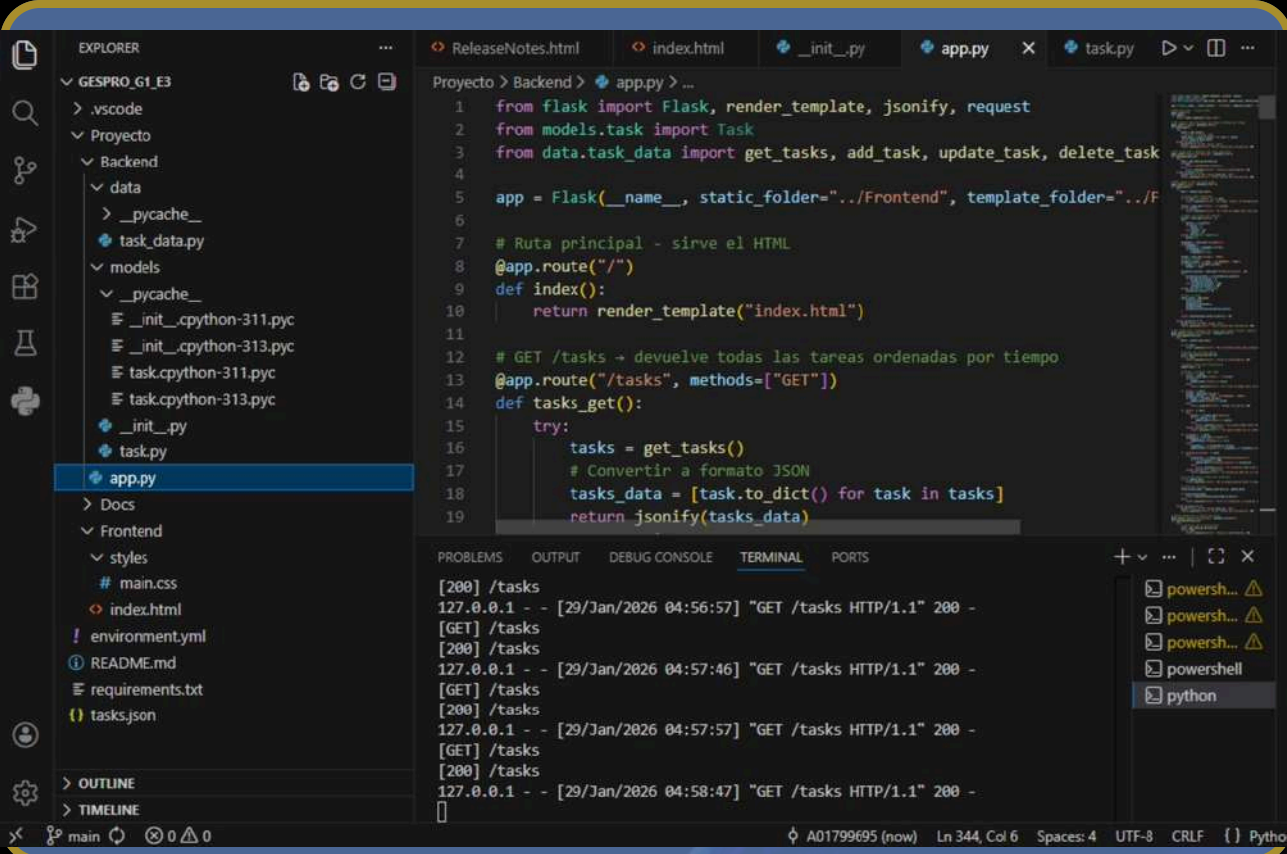
- Limpieza de código
- Separación de lógica
- Revisión de commits

Documentación inicial

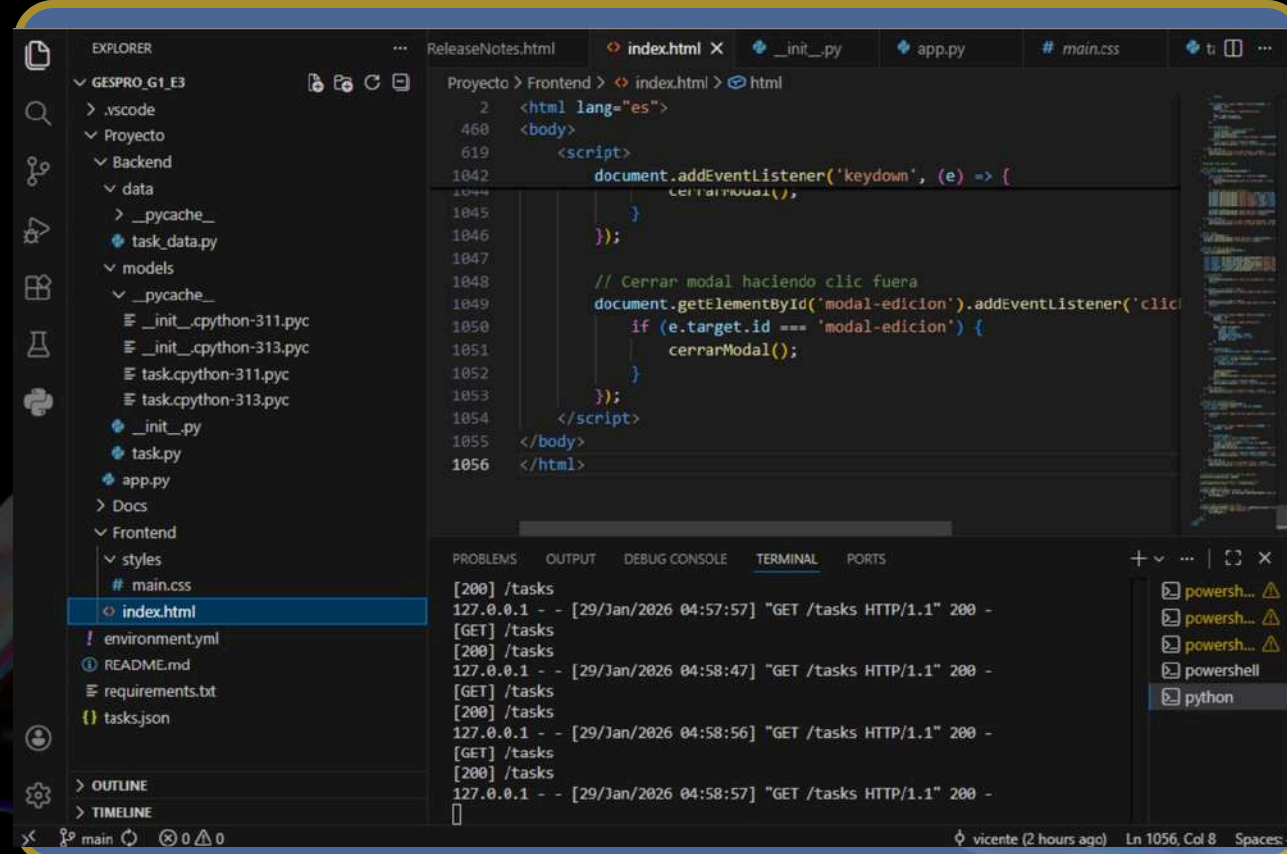
- Creación de README.md
- Instrucciones para ejecutar el proyecto en local
- Descripción de funcionalidades implementadas

Se implementan reglas de negocio simples:

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



APP.PY



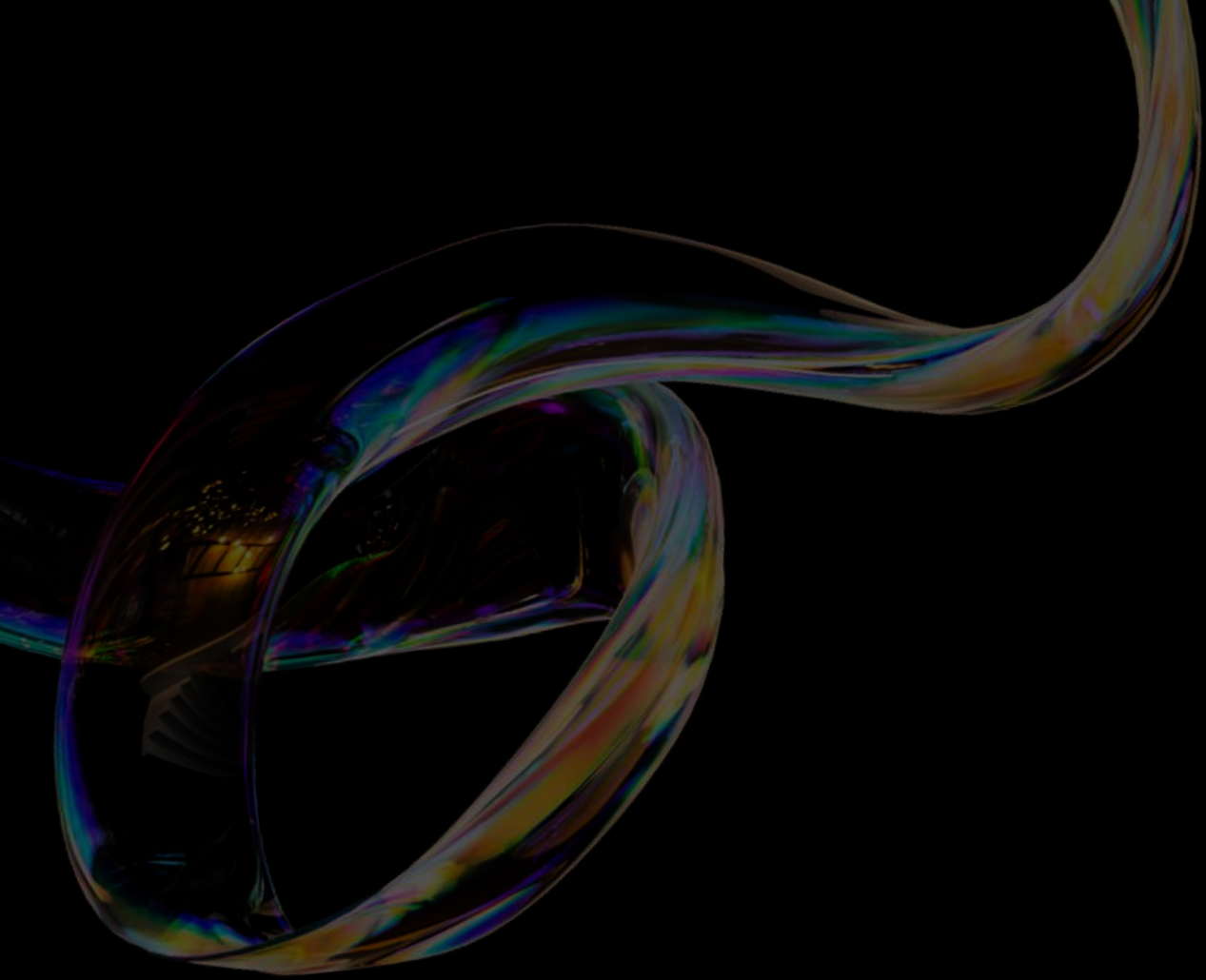
HTML



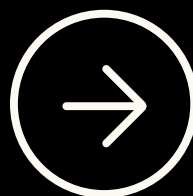
EDICIÓN Y GESTIÓN DE TAREA



TABLERO INICIAL



QUÉ SE HA CONSTRUIDO (DEMO)



SPRINT 6

Persistencia de tareas en fichero JSON

- bdd/tareas.json
- gitignore
- Límite para WIP
- Flexibilidad en el límite WIP

CÓDIGO

Gestor de Tareas Kanban

Organiza, prioriza y controla el trabajo en progreso con límite flexible

Total: 9 tareas | **TODO:** 4 | **IN PROGRESS:** 1 | **DONE:** 4

Tiempo total estimado: 540 minutos (9h 0m)

Límite WIP Flexible

Límite base: Actualizar

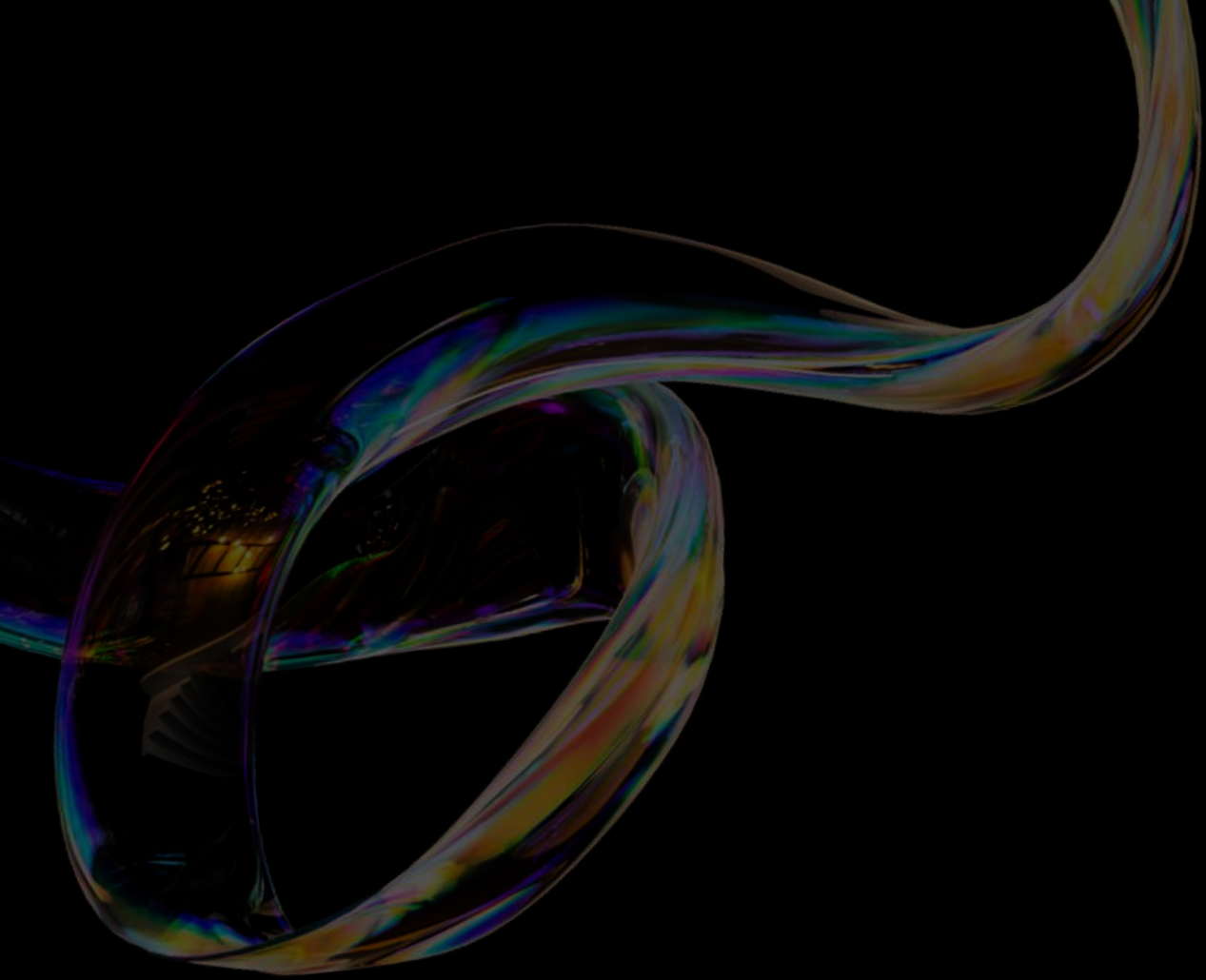
Límite flexible: 4 (+10% = 5)

Tareas en progreso: 1

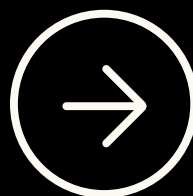
Porcentaje (base): 25%

✓ Dentro del límite base

The diagram illustrates three Kanban boards representing different stages of a workflow: TODO, IN PROGRESS, and DONE. Each board has a header with the column name and a dropdown menu set to 'Mayor tiempo primero'. The TODO board shows a task 'b' with 1 point, 1h 0m, and 'Sin asignar' status. The IN PROGRESS board shows a task 'b' with 1 point, 1h 0m, and 'Sin asignar' status. The DONE board shows a task 'a' with 3 points, 1h 0m, and 'Sin asignar' status. Each task has a row of buttons: TODO, IN PROGRESS, DONE, Editar, and a trash icon.



QUÉ SE HA CONSTRUIDO (DEMO)



SPRINT 7

Añadir fichero con user/pass/nombre

- bdd/usuarios.txt

Modificar frontend con desplegable usuarios/equipo

Añadir roles al equipo

- product owner
- normal
- invitado

Inicio de sesión, usuario

KanbanFlow

Sistema de Gestión de Tareas Agile

Correo Electrónico

usuario@empresa.com

Contraseña

••••••••

➔ Iniciar Sesión

© 2024 KanbanFlow - Demo de Gestión Ágil

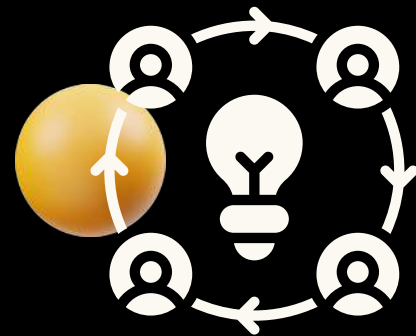
VISUALIZACIÓN INICIO DE SESIÓN

Gestión de Personas			
<div>Buscar usuario...</div> <div>Nuevo Usuario</div>			
Usuario	Rol	Email	Acciones
<div>AG</div> Ana García ID: 1	Product Owner (PO)	ana@empresa.com	<div>Editar</div> <div>Eliminar</div>
<div>CR</div> Carlos Rodríguez ID: 2	Agile Leader / Scrum Master	carlos@empresa.com	<div>Editar</div> <div>Eliminar</div>
<div>ML</div> María López ID: 4	Desarrollador Frontend	maria@empresa.com	<div>Editar</div> <div>Eliminar</div>
<div>PM</div> Pedro Martínez ID: 5	Desarrollador Full Stack	pedro@empresa.com	<div>Editar</div> <div>Eliminar</div>
<div>LS</div> Laura Sánchez ID: 6	Diseñador UX/UI	laura@empresa.com	<div>Editar</div> <div>Eliminar</div>
<div>JR</div> Jorge Ramírez ID: 7	QA / Tester	jorge@empresa.com	<div>Editar</div> <div>Eliminar</div>
<div>SC</div> Sofía Castro ID: 8	DevOps Engineer	sofia@empresa.com	<div>Editar</div> <div>Eliminar</div>
<div>DT</div> David Torres ID: 9	Invitado	david@empresa.com	<div>Editar</div> <div>Eliminar</div>

TABLERO INICIAL

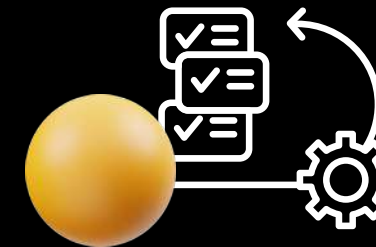
METODOLOGÍA DE TRABAJO

¿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

