

Task Forge

Sistema de Gestión de Tareas y Proyectos
con Estructura de Datos Basada en Árboles

Desarrollado por: Santiago Lozano
Universidad de la Costa
Fecha: 22 de Noviembre, 2025

Introducción

Contexto del Proyecto

Desarrollo de una aplicación que implementa estructuras de datos basadas en árboles para la gestión eficiente de tareas y proyectos, permitiendo organización jerárquica y escalable.

Objetivos

- Implementar una estructura de árbol para representar relaciones padre-hijo entre nodos
- Desarrollar operaciones CRUD completas sobre tareas y proyectos
- Crear una interfaz gráfica intuitiva y funcional
- Aplicar arquitectura de software limpia y escalable

Definición: Un nodo puede ser tanto una tarea individual como un proyecto de mayor alcance, ambos con atributos completos y capacidad de anidamiento.

Requerimientos del Sistema

Funcionalidades de Gestión

- Creación de tareas y proyectos
- Eliminación de tareas y proyectos
- Búsqueda de tareas y proyectos

Atributos por Nodo

- Título
- Descripción
- Fecha de vencimiento
- Prioridad (Alta, Media, Baja)
- Etiquetas (tags)
- Notas adicionales
- Identificador de nodo padre
- Estado de completitud

Interfaz Gráfica

Implementar una interfaz gráfica amigable que mejore la experiencia del usuario y facilite la visualización de la estructura jerárquica.

Arquitectura Tecnológica

Frontend

Flutter Framework

- Lenguaje: Dart
- UI: Material Design 3
- Estado: Provider Pattern
- HTTP: Cliente nativo
- Formato de fechas: intl

Backend

Python Flask

- Framework: Flask 3.0
- CORS: Flask-CORS
- API: RESTful
- Persistencia: In-memory (MVP)
- Formato: JSON

Patrón Arquitectónico: Cliente-Servidor con comunicación mediante API REST. Separación clara de responsabilidades entre presentación y lógica de negocio.

Implementación de Estructura de Árbol

Concepto Teórico

Cada nodo contiene un campo **parent_id** que referencia a su nodo padre, formando una estructura de árbol mediante relaciones implícitas.

Ejemplo de Estructura: Proyecto A (id: 1, parent_id: null) |— Tarea 1 (id: 2, parent_id: 1) |— Sub-Proyecto B (id: 3, parent_id: 1) | |— Tarea 2 (id: 4, parent_id: 3) | |— Tarea 3 (id: 5, parent_id: 3) |— Tarea 4 (id: 6, parent_id: 1)

Operaciones sobre el Árbol

| Operación | Implementación | Complejidad |
|-------------------|---------------------------------|-------------|
| Insertión | Asignar parent_id al crear nodo | O(1) |
| Búsqueda de hijos | Filtrar por parent_id | O(n) |
| Eliminación | Recursiva en cascada | O(n) |

Backend: API REST en Flask

Estructura del Proyecto

app.py (130 líneas)

- Almacenamiento en memoria mediante lista de diccionarios
- Funciones helper para operaciones sobre el árbol
- Endpoints RESTful para todas las operaciones
- Configuración de CORS para comunicación cross-origin

Endpoints Implementados

| | |
|--------|---------------------|
| GET | /api/tasks |
| POST | /api/tasks |
| PUT | /api/tasks/:id |
| DELETE | /api/tasks/:id |
| GET | /api/search?q=query |

Conclusiones

Cumplimiento de Objetivos

- Se implementó exitosamente una estructura de árbol mediante relaciones parent-child
- Se desarrollaron todas las operaciones CRUD requeridas
- La interfaz gráfica cumple con estándares de usabilidad modernos
- La arquitectura permite escalabilidad y mantenimiento futuro
- El sistema está funcional y desplegado en la nube

Aprendizajes Técnicos

- Aplicación práctica de estructuras de datos basadas en árboles
- Diseño e implementación de APIs RESTful
- Desarrollo de aplicaciones móviles multiplataforma
- Gestión de estado en aplicaciones reactivas

Gracias

Task Forge
Sistema de Gestión de Tareas y Proyectos

¿Preguntas?