■■ Asistente de Planificación Diaria

con LangGraph + Llama 3.2

Sistema Agéntico para Optimización de Horarios Diarios

Generado: 04 de November de 2025

1. Descripción General del Proyecto

Este proyecto implementa un asistente inteligente de planificación diaria que utiliza LangGraph para orquestar un flujo de trabajo compuesto por tres nodos especializados. El sistema recibe descripciones naturales de las actividades que el usuario desea realizar durante el día y genera un horario personalizado, validando automáticamente que el plan incluya descansos adecuados después de actividades intensivas.

2. Objetivos del Proyecto

• **Análisis Inteligente:** Utilizar Llama 3.2 para extraer actividades de forma semántica, no solo mediante palabras clave.

• **Generación de Horarios:** Crear horarios personalizados que se adapten a las actividades específicas del usuario.

• **Validación de Salud:** Rechazar planes que incluyan actividades intensivas sin descansos adecuados.

• **Modularidad:** Implementar un flujo modular con LangGraph para facilitar el mantenimiento y expansión.

3. Arquitectura del Sistema

El proyecto está diseñado como un grafo dirigido acíclico (DAG) con tres nodos principales que se ejecutan secuencialmente. Cada nodo procesa el estado de la solicitud y lo transmite al siguiente,

permitiendo una separación clara de responsabilidades.

Nodo	Función	Tecnología
Analizador	Extrae actividades del texto de entrada	Llama 3.2 + JSON
Planificador	Genera horario personalizado 07:00-22:00+	Lógica determinística
Validador	Valida descansos e intensidad del plan	Llama 3.2 + Lógica híbrida

4. Flujo de Ejecución

START → Analizador (Llama 3.2) → Planificador → Validador (Llama 3.2) → END El flujo es determinístico y secuencial. El estado se transmite entre nodos como un TypedDict que contiene: entrada del usuario, actividades extraídas, plan generado, validez, y mensaje de validación.

5. Implementación Técnica

5.1 Nodo Analizador

Utiliza Llama 3.2 con un prompt estructurado que instrye al modelo a extraer actividades en formato JSON. El prompt incluye un mapeo explícito de palabras clave (cine, pasear, ejercicio, etc.) a tipos de actividad. Maneja excepciones gracefully asignando una actividad genérica si el JSON es inválido.

5.2 Nodo Planificador

Define un horario base (07:00-22:00+) y marca las actividades del usuario con checkmarks (✓). Incluye un mapeo de actividades a franjas horarias (estudio: 13:30-15:30, ejercicio: 16:00-17:30, etc.). Genera recomendaciones personalizadas basadas en las actividades detectadas.

5.3 Nodo Validador

Implementa una lógica híbrida que combina reglas determinísticas con evaluación de Llama 3.2. Las reglas son:

• Si SOLO hay actividades relajadas (cine, lectura, pasear, estudio) → APROBADO
• Si hay actividades intensivas (clases, ejercicio, trabajo) Y el horario incluye descansos (Pausa, Relax, Cena) → APROBADO
• Si hay actividades intensivas SIN descansos en el horario → RECHAZADO
• Fallback a Llama 3.2 si no coincide ninguna regla anterior

6. Ejemplo de Ejecución

Entrada del usuario:

```
"Hoy quiero salir a pasear con mis hijos, estudiar para un examen e ir al gimnasio"
```

Salida generada:

```
ACTIVIDADES DETECTADAS: pasear, estudio, ejercicio HORARIO: 07:00-08:00 → Desayuno
13:30-15:30 → ✓ Estudio 15:30-16:00 → Pausa 16:00-17:30 → ✓ Ejercicio 17:30-19:00
→ ✓ Pasear con familia VALIDACIÓN: ■ PLAN APROBADO - Plan equilibrado con descansos
adecuados JSON: {"user_input": "...", "extracted_activities": [...], "is_valid":
true, ...}
```

7. Tecnologías y Dependencias

Tecnología	Versión	Función
LangGraph	0.0.84	Orquestación de flujos agénticos
LangChain	0.1.11	Framework para LLMs
Ollama	0.1.0	Runtime local para Llama 3.2
Llama 3.2	N/A	Modelo de lenguaje para análisis y validación
Python	3.8+	Lenguaje de programación

8. Tipos de Validación

Validación de Actividades Intensivas:
Se considera intensiva: clases, ejercicio, trabajo
Se considera relajada: cine, lectura, personal, descanso, estudio, pasear

Validación de Descansos:
El sistema verifica explícitamente en el horario generado la presencia de:
• Pausa (15:30-16:00)
• Almuerzo (12:00-13:30)
• Cena (19:00-20:00)
• Relax (20:00-22:00)
• Desayuno (07:00-08:00)

Criterios de Aprobación:
■ Plan APROBADO si: (a) solo actividades relajadas, (b) actividades intensivas con descansos
■ Plan RECHAZADO si: actividades intensivas sin descansos
■■ Plan ADVERTENCIA si: pocas pausas con múltiples actividades

9. Características Clave

✓ Análisis semántico de entrada natural con Llama 3.2

✓ Generación de horarios personalizados y dinámicos

✓ Validación inteligente de descansos basada en actividades

✓ Mapeo flexible de actividades a franjas horarias

✓ Recomendaciones personalizadas por actividad

✓ Salida estructurada en JSON para integración programática

✓ Manejo robusto de excepciones y fallbacks

✓ Modularidad mediante LangGraph para fácil extensión

10. Casos de Uso

Caso 1: Día Relajado
Entrada: "Ir al cine, estudiar un poco"
Resultado: ■ APROBADO - Día relajado y equilibrado

Caso 2: Día Intenso con Descansos
Entrada: "Clases de IA, gimnasio, estudiar para examen"
Resultado: ■ APROBADO - Plan equilibrado con descansos adecuados

Caso 3: Día Sobrecargado
Entrada: "Clases por la mañana, trabajo por la tarde, más trabajo por la noche"
Resultado: ■ RECHAZADO - Actividades intensivas sin descanso

11. Conclusiones

El proyecto demuestra la aplicación práctica de LangGraph para orquestar flujos de procesamiento de lenguaje natural complejos. La combinación de análisis semántico (Llama 3.2) con lógica determinística permite crear un sistema robusto y predecible que: • Entiende contexto y requiere validación inteligente de planes diarios • Genera recomendaciones personalizadas basadas en actividades específicas • Valida automáticamente la salud y equilibrio del plan • Produce salida estructurada para integración con otros sistemas La arquitectura modular permite agregar nuevas actividades, ajustar horarios o mejorar la validación sin modificar la estructura base del sistema.

12. Repositorio y Acceso

GitHub Repository:
https://github.com/ricdex/utec-ai/tree/main/tarea_langgraph

Archivos Principales:
• **daily_planner_with_llama.py** - Código principal (~240 líneas)

• **requirements.txt** - Dependencias (langgraph, langchain, ollama)
• **README.md** - Documentación de instalación y uso

Instrucciones de Ejecución:
1. Instalar Ollama desde ollama.ai
2. Descargar modelo: `ollama pull llama3.2`
3. Instalar dependencias: `pip install -r requirements.txt`
4. Ejecutar servidor: `ollama serve`
5. En otra terminal: `python daily_planner_with_llama.py`

Documento generado: 04/11/2025 - 10:51 | Proyecto Académico LangGraph