Tarjetas — Sistema de Análisis de Complejidad con LLMs

Resumen del proyecto
El sistema aceptará descripciones en lenguaje natural o pseudocódigo, reconocerá estructuras algorítmicas (ciclos, condicionales, recursiones, llamadas a procedimientos y estructuras de datos) y producirá un análisis detallado de complejidad computacional que incluya razonamientos formales (árboles de recursión, relaciones de recurrencia, DP, heurísticas, voraces, etc.). Además se incorporarán LLMs para traducir NL a pseudocódigo, ayudar en el razonamiento paso a paso, clasificar patrones, validar análisis y generar diagramas de ejecución y documentación con coste por instrucción (microsegundos y tokens).

Tarjeta: 01 — Definir alcance y requisitos funcionales

- **Objetivo:** Establecer requisitos funcionales y no funcionales del sistema.
- **Descripción:** Reunir stakeholders, redactar historias de usuario, definir límites del MVP (qué estructuras y análisis cubrir en la primera versión).
- **Subtareas:**
    - Workshop con interesados.
    - Lista de casos de uso (entradas/salidas).
    - Definir métricas de calidad (precisión, latencia, coste por análisis).
- **Criterios de aceptación:** Documento de requisitos aprobado; backlog inicial con prioridades.
- **Dependencias:** Ninguna.
- **Prioridad:** Alta.
- **Estimación:** 5 pts.

Tarjeta: 02 — Diseño de arquitectura general

- **Objetivo:** Crear la arquitectura (frontend, backend, motores de análisis, almacenamiento) y seleccionar tecnologías.
- **Descripción:** Diagrama de componentes, flujos de datos, puntos de integración con LLM y servicios externos.
- **Subtareas:**
    - Diagrama de alto nivel.
    - Decisiones sobre LLM (local vs API), formatos de intercambio (JSON, protobuf).
    - Definir API interna para el motor de análisis.
- **Criterios de aceptación:** Diagrama + documento de decisiones de arquitectura.
- **Dependencias:** Tarjeta 01.
- **Prioridad:** Alta.
- **Estimación:** 8 pts.

Tarjeta: 03 — Parser de algoritmos y normalizador de entrada

- **Objetivo:** Implementar un parser capaz de interpretar pseudocódigo estructurado y representar ASTs.
- **Descripción:** Soporte inicial para: asignaciones, for/while, if/else, return, llamadas a procedimientos, recursión, arreglos/vectores, estructuras tipo objeto; tokenización y manejo de ambigüedades.
- **Subtareas:**
    - Definir gramática de pseudocódigo.
    - Implementar lexer & parser (ANTLR/pegjs/menhir o librería elegida).
    - Convertir a una representación intermedia (AST normalizada).
    - Tests unitarios con ejemplos simples y complejos.
- **Criterios de aceptación:** Parser pasa suite de tests (mínimo 95% de casos definidos); AST documentado.

- **Dependencias:** Tarjeta 02.
- **Prioridad:** Muy alta.
- **Estimación:** 13 pts.

Tarjeta: 04 — Integración LLM: traducción NL → pseudocódigo

- **Objetivo:** Permitir que el sistema reciba descripciones en lenguaje natural y genere pseudocódigo estructurado.
- **Descripción:** Diseñar prompts, pipeline de postprocesado y validación sintáctica del pseudocódigo generado.
- **Subtareas:**
    - Selección del modelo (coste/latencia/privacidad).
    - Creación de prompt templates y few-shot examples.
    - Post-procesado para mapear la salida del LLM al AST del parser.
    - Tests de calidad (BLEU/EM/accuracy contra dataset).
- **Criterios de aceptación:** Traducción correcta en ≥ 85% de casos del set de pruebas; fallback cuando no se puede generar código válido.
- **Dependencias:** Tarjeta 03.
- **Prioridad:** Muy alta.
- **Estimación:** 13 pts.

Tarjeta: 05 — Motor de análisis de complejidad (secuencial)

- **Objetivo:** Implementar reglas y algoritmos para estimar complejidad temporal y espacial en código secuencial.
- **Descripción:** Analizador que recorre el AST, cuenta operaciones, identifica patrones básicos (O(1), O(n), O(n^2), etc.) y produce justificativos (sumatorias, simplificación).

- **Subtareas:**
    - Definir catálogo de costes por instrucción.
    - Reglas de agregación (composición de costes en secuencia y anidamiento).
    - Generación de explicación matemática (notación Theta/Omega/O).
    - Tests con ejemplos clásicos (búsqueda lineal, ordenamiento simple, etc.).
- **Criterios de aceptación:** Resultados correctos en el conjunto de pruebas; explicación matemática legible y trazable.
- **Dependencias:** Tarjeta 03.
- **Prioridad:** Alta.
- **Estimación:** 13 pts.

Tarjeta: 06 — Análisis de recursión: relaciones de recurrencia y árboles de recursión

- **Objetivo:** Resolver relaciones de recurrencia frecuentes y construir árboles de recurrencia cuando aplique.
- **Descripción:** Implementar estrategia para detectar recursividad, extraer relación de recurrencia y resolverla (Master theorem, sustitución, expansión, memoización/DP cuando aplique).
- **Subtareas:**
    - Detectar tipo de recursión (lineal, divide & conquer, múltiple llamada).
    - Generador de árboles de recurrencia con pasos intermedios.
    - Implementar solución por casos (teorema maestro, recurrencia lineal, caso general con aproximación).
    - Tests y validación numérica.
- **Criterios de aceptación:** Capacidad de resolver y justificar al menos los tipos comunes de recurrencia; producción de árbol con pasos.
- **Dependencias:** Tarjeta 03, 05.

- **Prioridad:** Alta.
- **Estimación:** 13 pts.

Tarjeta: 07 — Programación dinámica y heurísticas (reconocimiento y reformulación)

- **Objetivo:** Detectar subestructura óptima y reformular recursiones en DP cuando sea posible; aplicar heurísticas y aproximaciones.
- **Descripción:** Identificar estados, transiciones y posibilidad de memoización; proponer versión DP y computar su complejidad.
- **Subtareas:**
    - Detector de sobreposición de subproblemas.
    - Transformador recursión → DP (top-down/bottom-up) en pseudocódigo.
    - Evaluación comparativa (complejidad antes/después).
- **Criterios de aceptación:** Transformaciones correctas en ejemplos estándar (Fibonacci, knapsack, caminos en matriz).
- **Dependencias:** 03, 06.
- **Prioridad:** Media-Alta.
- **Estimación:** 8 pts.

Tarjeta: 08 — Clasificación automática de patrones algorítmicos

- **Objetivo:** Identificar patrones como divide & conquer, greedy, backtracking, dynamic programming, bruteforce, graph algorithms.
- **Descripción:** Módulo que etiqueta el algoritmo con patrón(s) conocidos y explica por qué.
- **Subtareas:**
    - Definir taxonomía de patrones.

    - Entrenar o reglas basadas en heurísticas/LLM para clasificación.
    - Tests con ejemplos etiquetados.
- **Criterios de aceptación:** Precisión ≥ 90% en set control de patrones clásicos.
- **Dependencias:** 03, 04.
- **Prioridad:** Media.
- **Estimación:** 8 pts.

Tarjeta: 09 — Verificación matemática del análisis

- **Objetivo:** Proveer validación formal o numérica del análisis (relaciones, límites, pruebas básicas).
- **Descripción:** Integrar herramientas simbólicas (SymPy u otras), checks numéricos y cross-check con LLM para contrastar deducciones.
- **Subtareas:**
    - Implementar representación simbólica de sumatorias/recurrencias.
    - Automatizar pruebas de equivalencia y límites asintóticos.
    - Generar contraejemplos numéricos cuando aplique.
- **Criterios de aceptación:** Validación automática para al menos el 80% de las derivaciones generadas.
- **Dependencias:** 05, 06.
- **Prioridad:** Alta.
- **Estimación:** 8 pts.

Tarjeta: 10 — Diagrama y visualizador de trazas de ejecución

- **Objetivo:** Generar diagramas paso a paso que muestren la ejecución del pseudocódigo y el consumo de recursos.

- **Descripción:** Visualizador web que muestre llamadas, pilas de recursión, estados de estructuras, contadores de operaciones y puntos de coste.
- **Subtareas:**
    - Diseño de UX para trazas.
    - Motor que genere eventos de ejecución a partir del AST.
    - Render de diagramas (D3, SVG o librería elegida).
- **Criterios de aceptación:** Visualizaciones interactivas que reproduzcan correctamente el flujo para ejemplos de test.
- **Dependencias:** 03, 05, 06.
- **Prioridad:** Media.
- **Estimación:** 13 pts.

Tarjeta: 11 — Costeo por instrucción (microsegundos y tokens)

- **Objetivo:** Estimar coste por instrucción en tiempo (microsegundos) y coste en tokens por invocación de LLM.
- **Descripción:** Construir una tabla de coste por operación y medir/estimar latencias; integrar cálculo de tokens para prompts y completions.
- **Subtareas:**
    - Medir tiempos de operaciones representativas en entorno objetivo.
    - Definir costes por instrucción y por estructura (por ejemplo, coste de acceso a arreglo, asignación, comparación).
    - Instrumentar pipeline LLM para contabilizar tokens (prompts y respuestas) y coste por llamada.
- **Criterios de aceptación:** Reportes de coste reproducibles y estimación de tokens por análisis.
- **Dependencias:** 02, 04.
- **Prioridad:** Media.

- **Estimación:** 8 pts.

Tarjeta: 12 — Pipeline de tests y benchmark (suite algorítmica)

- **Objetivo:** Crear suite de pruebas con algoritmos conocidos para validar precisión y rendimiento del sistema.
- **Descripción:** Base de casos (pequeños y grandes), métricas de evaluación y scripts de ejecución automatizada.
- **Subtareas:**
    - Recolectar y curar dataset de algoritmos y descripciones NL.
    - Tests unitarios y de integración.
    - Benchmarks de latencia y consumo de tokens.
- **Criterios de aceptación:** Suite ejecutable y reproducible; reportes de resultados.
- **Dependencias:** 03, 04, 05.
- **Prioridad:** Muy alta.
- **Estimación:** 8 pts.

Tarjeta: 13 — Interfaz de usuario (Web) — entrada y visualización de análisis

- **Objetivo:** Diseñar e implementar UI donde el usuario suba descripción NL o pseudocódigo y vea el análisis, trazas y costes.
- **Descripción:** Formulario de entrada, panel de resultados (complejidad, explicación, diagrama), historial y exportar resultados.
- **Subtareas:**
    - Mockups y flujo UX.
    - Implementación React + llamadas API.

    - Exportar análisis (PDF/Markdown/JSON).

- **Criterios de aceptación:** Interfaz usable con los flujos principales; tests de usabilidad básicos.
- **Dependencias:** 02, 10.
- **Prioridad:** Alta.
- **Estimación:** 13 pts.

Tarjeta: 14 — API pública y backend

- **Objetivo:** Exponer funcionalidades como API REST/GraphQL para uso programático.
- **Descripción:** Endpoints para enviar descripción, obtener pseudocódigo, análisis, trazas y métricas de coste.
- **Subtareas:**
    - Diseñar contrato API.
    - Implementar autenticación y rate-limiting.
    - Logging y métricas.
- **Criterios de aceptación:** Documentación de la API y endpoints operativos con pruebas básicas.
- **Dependencias:** 02, 03, 05.
- **Prioridad:** Alta.
- **Estimación:** 8 pts.

Tarjeta: 15 — Almacenamiento y dataset de ejemplos

- **Objetivo:** Diseñar repositorio de ejemplos (NL ↔ pseudocódigo ↔ análisis) para entrenar y validar.

- **Descripción:** Esquema de datos, pipelines de ingestión, etiquetado y versionado.
- **Subtareas:**
    - Definir esquema JSON de ejemplo.
    - Implementar base de datos (Postgres/NoSQL) y UI de curator.
    - Estrategia de versionado y exportación para benchmarks.
- **Criterios de aceptación:** Repositorio inicial poblado con ejemplos y API para consultarlos.
- **Dependencias:** 12, 14.
- **Prioridad:** Media.
- **Estimación:** 5 pts.

Tarjeta: 16 — Seguridad, privacidad y gobernanza del uso de LLM

- **Objetivo:** Garantizar privacidad de las descripciones y cumplimiento de políticas de uso de modelos.
- **Descripción:** Políticas de retención, anonimización, control de prompts sensibles y opt-out para datos.
- **Subtareas:**
    - Definir política de datos.
    - Implementar anonimización y cifrado en tránsito/descanso.
    - Revisar TOS del proveedor LLM.
- **Criterios de aceptación:** Política redactada y controles técnicos implementados mínimos.
- **Dependencias:** 02, 04.
- **Prioridad:** Alta.
- **Estimación:** 5 pts.

Tarjeta: 17 — Optimización y escalado del motor de análisis

- **Objetivo:** Mejorar rendimiento y paralelización del análisis para grandes inputs y lotes.
- **Descripción:** Perfilado, cache de resultados, paralelización segura, optimizaciones en memoria.
- **Subtareas:**
    - Medir cuellos de botella.
    - Introducir cache de sub-análisis.
    - Pruebas de carga.
- **Criterios de aceptación:** Reducción de latencia en x% (según baseline) y estabilidad bajo carga.
- **Dependencias:** 05, 11.
- **Prioridad:** Media.
- **Estimación:** 8 pts.

Tarjeta: 18 — Documentación, tutoriales y ejemplos educativos

- **Objetivo:** Crear documentación para usuarios y para desarrolladores, con ejemplos paso a paso.
- **Descripción:** Guías: cómo interpretar análisis, cómo proveer descripciones NL efectivas, formato de pseudocódigo, FAQ.
- **Subtareas:**
    - Guía de usuario (MVP features).
    - Tutorial de 5 ejemplos clásicos.
    - Documentación técnica API y arquitectura.
- **Criterios de aceptación:** Documentación accesible y ejemplos reproducibles.

- **Dependencias:** 03, 05, 10.
- **Prioridad:** Media.
- **Estimación:** 5 pts.

Tarjeta: 19 — CI/CD, despliegue y monitorización

- **Objetivo:** Automatizar pruebas, builds y despliegue; monitorizar uso y errores.
- **Descripción:** Pipelines para tests, despliegue en staging/production, dashboards de métricas.
- **Subtareas:**
    - Configurar pipelines (GitHub Actions/GitLab CI).
    - Scripts de despliegue.
    - Dashboard de métricas (latencia, errores, tokens consumidos).
- **Criterios de aceptación:** Pipelines en funcionamiento y despliegue reproducible.
- **Dependencias:** 14, 12.
- **Prioridad:** Media.
- **Estimación:** 5 pts.

Tarjeta: 20 — Evaluación final y plan de roadmap

- **Objetivo:** Ejecutar evaluación con stakeholders
- **Descripción:** Uso de resultados de la suite de pruebas, feedback de usuarios y definir próximos hitos.
- **Subtareas:**
    - Recolección de métricas y feedback.
    - Priorizar backlog para siguientes sprints.

    - Plan de hitos y recursos.

- **Criterios de aceptación:** Roadmap aprobado y backlog priorizado.
- **Dependencias:** Todas las anteriores.
- **Prioridad:** Alta.
- **Estimación:** 3 pts.