ATLAS: underwriting determinista de deep-tech

La IA razona y extrae; los números los calcula un motor auditable — «underwrite the physics»

Adrián Anchuela

Proyecto ATLAS · Agosto de 2026 · Despliegue de referencia: atlas.mlmarketlens.com

***Resumen*—** Evaluar la viabilidad de una startup de *deep-tech* (baterías, electrónica de potencia, captura de carbono, robótica, manufactura espacial) exige traducir un *data room* técnico —químicas, *specs*, BOM, hitos— en un modelo financiero ajustado por riesgo. Es un trabajo escaso: pocos analistas combinan criterio de ingeniería y de inversión, y las herramientas de *due diligence* actuales se limitan a resumir documentos. Un enfoque ingenuo con grandes modelos de lenguaje (LLM) agrava el problema: el modelo alucina cifras financieras que parecen precisas pero no son auditables. Presentamos **ATLAS**, una plataforma que adopta un principio de diseño estricto: **la IA razona y extrae; los números los calcula un motor determinista**. Un LLM clasifica el vertical, extrae parámetros técnicos y emite juicios calibrados mediante *salida estructurada*; un motor en Python convierte esos parámetros en economía unitaria, escenarios, TIR y un score compuesto, con aritmética reproducible y trazable. Cada afirmación del informe conserva su base, confianza y origen. Documentamos el problema, el principio de separación, la arquitectura, el modelo financiero multivertical, el modelo de *scoring* ponderado y calibrable, la capa de procedencia, el modelo BYOK y las visualizaciones que genera el sistema. En una validación con casos públicos, el modelo determinista cae en banda en 11/12 métricas y la extracción alcanza el 96–100 % de fidelidad; el banco de evaluación detectó y permitió corregir un error real de la IA. El resultado hace del underwriting de deep-tech un proceso auditable, no una caja negra.

Palabras clave— underwriting, deep-tech, análisis tecnoeconómico, LLM, salida estructurada, curva de aprendizaje, TIR, LCOS, TRL, auditabilidad, sistemas deterministas, visualización.

1. Introducción y problema

La inversión en *deep-tech* —empresas cuyo riesgo dominante es físico y de ingeniería, no de mercado— crece con fuerza, pero su evaluación sigue siendo artesanal. Decidir si una startup de baterías o de captura de carbono es financiable requiere responder a una pregunta que pocos saben plantear: *¿cuál es la economía unitaria de esta tecnología a escala, y qué riesgo técnico separa el laboratorio de la primera planta comercial?* Responderla exige leer un *data room* denso (química de celda, BOM, curvas de coste, hitos de proceso) y traducirlo a un modelo financiero: coste por unidad a escala, coste nivelado, TIR de la primera planta, márgenes por escenario.

Este trabajo choca con tres obstáculos. **Primero, escasez de criterio combinado:** el análisis correcto vive en la intersección de ingeniería, finanzas e industria, un perfil raro y caro. **Segundo, herramientas superficiales:** el *tooling* de *due diligence* existente resume PDFs y extrae cláusulas, pero no *modela la física* —no convierte una química en una curva de coste ni un riesgo de proceso en una probabilidad de fallo del escalado—. **Tercero, y crítico, la tentación de la IA generativa:** pedir directamente a un LLM «dame la TIR y el coste a escala de este proyecto» produce cifras fluidas y verosímiles pero *alucinadas*: no se pueden auditar, no son reproducibles y colapsan la distinción entre un dato extraído y un número calculado.

La tesis de ATLAS —resumida en el lema *«underwrite the physics»*— es que el underwriting de deep-tech puede sistematizarse si se separan con rigor dos tareas de naturaleza distinta: el **juicio** (clasificar, extraer, ponderar cualitativamente), que es donde un LLM aporta valor, y el **cálculo** (economía unitaria, escenarios, TIR, score), que debe ser determinista, explícito y auditable.

2. Contexto y trabajo relacionado

2.1 *Análisis tecnoeconómico*

El núcleo cuantitativo bebe del análisis tecnoeconómico clásico. La **curva de aprendizaje** (ley de Wright[1]) describe cómo el coste unitario cae un porcentaje fijo por cada duplicación del volumen acumulado; en hardware, tasas del 80–90 % (10–20 % de bajada por duplicado) son habituales. El **coste nivelado** —LCOE en generación, **LCOS** en almacenamiento[2]— amortiza el capex de sistema sobre la energía o las unidades entregadas a lo largo de la vida útil, permitiendo comparar tecnologías heterogéneas en una única unidad ($/kWh, $/t CO_2, $/kg…).

2.2 *Madurez y riesgo de escalado*

La escala **TRL** (*Technology Readiness Level*, 1–9)[3] ancla la madurez técnica; el salto entre el TRL de laboratorio y el necesario para la primera planta —el riesgo *first-of-a-kind* (FOAK) — es, en deep-tech, el determinante real de la financiabilidad. ATLAS lo modela explícitamente a partir de la severidad y probabilidad de los riesgos de proceso.

2.3 *LLM y salida estructurada*

Los LLM modernos permiten *salida estructurada*: forzar la respuesta a un esquema tipado (aquí, un modelo Pydantic) en lugar de texto libre. ATLAS usa esta capacidad[4] no para que el modelo calcule, sino para que *rellene un formulario de parámetros* con la disciplina de un esquema. Esta restricción es la que hace viable la separación IA/motor: el LLM entrega *inputs* tipados y acotados, nunca resultados financieros.

3. Principio de diseño: IA que razona, motor que calcula

El principio rector es una separación de responsabilidades estricta (Fig. 1). La IA interviene una sola vez —para clasificar el vertical, extraer parámetros técnicos y emitir juicios cualitativos— y entrega un objeto tipado. El coste a escala, la TIR, el LCOS y el score los calcula código determinista.

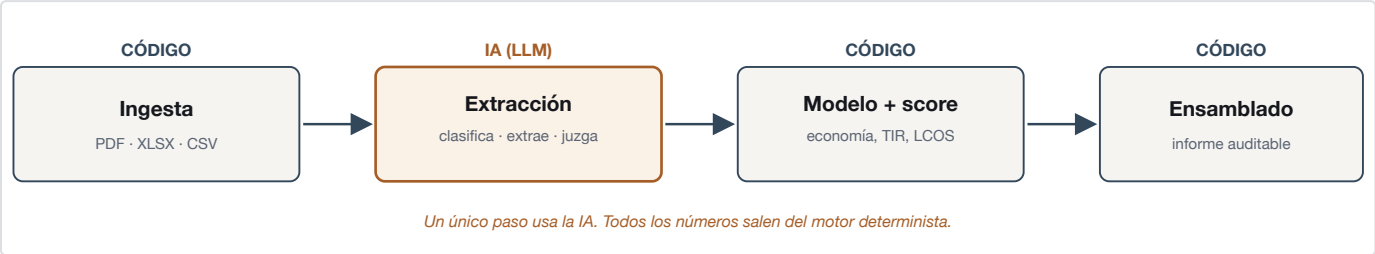

Fig. 1. Pipeline de un expediente. La IA sólo interviene en la extracción; el coste a escala, la TIR, el LCOS y el score los calcula código determinista.

Las consecuencias de esta separación son el corazón de la propuesta: **auditabilidad** (todo número procede de una fórmula explícita sobre *inputs* conocidos), **reproducibilidad** (los mismos parámetros dan siempre el mismo resultado; sin varianza de *sampling* en las cifras), **antialucinación por construcción** (el esquema no tiene campos de salida financiera; el LLM no *puede* inventar una TIR) y **calibrabilidad** (la lógica de puntuación es un artefacto editable, no un comportamiento emergente opaco).

> **REGLA DE ORO DEL SISTEMA**
> El LLM **da inputs, no resultados**. Si en algún punto el modelo «calcula el resultado», el diseño ha fallado. El esquema de salida y el *prompt* están construidos para hacer esa frontera infranqueable.

4. Arquitectura del sistema

ATLAS es un monorepo con dos servicios (Fig. 2). La **web** es una aplicación Next.js (App Router, TypeScript) que sirve la interfaz y actúa de *proxy* hacia el motor. El **motor** es una API FastAPI (Python) que orquesta el *pipeline* y persiste en SQLite. La web nunca habla con el LLM ni con la base de datos: toda la lógica de dominio vive en el motor, una única fuente de verdad. La clave de la IA es *bring-your-own-key* (BYOK): viaja del navegador al motor por cabecera y nunca se almacena en el servidor.

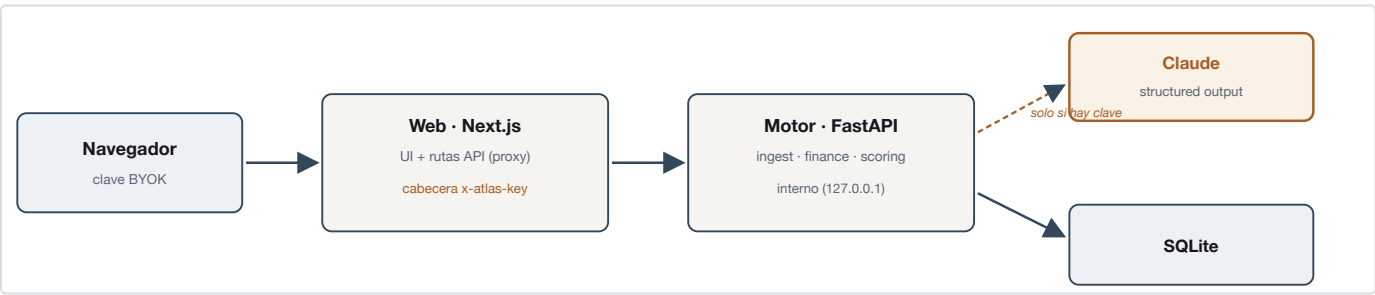

Fig. 2. Arquitectura de servicios. El motor sólo llama a Claude en modo *live*; sin clave opera en «modo demo» (extracción simulada, cálculo real). En producción, un reverse proxy Caddy termina TLS y el motor no se expone a Internet.

El *pipeline* de un análisis es: **ingesta** (extracción de texto de PDF, XLSX y CSV) → **extracción** (LLM con salida estructurada, o *fallback* determinista) → **modelo financiero** + **scoring** → **ensamblado** del informe → **persistencia**. El informe resultante es un JSON con nombres que coinciden 1:1 con los tipos del frontend, de modo que el mismo objeto describe un análisis en vivo o uno sembrado.

5. Implementación

5.1 *Capa de extracción*

La extracción usa `messages.parse(output_format=ExtractedParams)`, donde `ExtractedParams` es un modelo Pydantic que define exactamente qué puede devolver el LLM: el vertical (enumerado), los costes en la unidad del vertical (`current/target/benchmark_unit_cost`), TRL actual y objetivo, tasa de aprendizaje, capex de la primera planta, capacidad de planta, cinco juicios cualitativos por dimensión (0–100 con nota) y una lista de riesgos (severidad y probabilidad 1–5). **No existe ningún campo de salida financiera** (coste a escala real, TIR, márgenes): esa ausencia es deliberada. La calidad del modo en vivo depende de un *prompt* de sistema diseñado como **rúbrica**: (i) la regla de oro «das inputs, no resultados»; (ii) «fundamentar antes que inventar»; (iii) reglas de coherencia (`current ≥ target`) y rango del *learning rate* (0,80–0,90); (iv) anclas de la escala TRL; y (v) una **calibración explícita de los juicios 0–100** en bandas, para que el modelo discrimine en vez de amontonar todo en 70–85. Si no hay clave o falla, la extracción cae a un **modo demo determinista** que fabrica parámetros verosímiles y variados por proyecto; el cálculo posterior es siempre real, y la clave nunca se registra.

5.2 *Motor financiero determinista*

Toda la aritmética es explícita y parametrizada por el **perfil del vertical** (Tabla 1). Para cada uno de los tres escenarios se calcula:

```
ingresos = salida·benchmark  ·  margen = (ingresos
             − salida·coste)/ingresos

        ebitda = (margen − opex)·ingresos
```

La **TIR de la primera planta** resuelve la tasa que anula el VAN del flujo `[−capex, ebitda, …, ebitda]` por **bisección** (100 iteraciones, rango −90 %…+200 %), robusta y sin derivadas. Los **escenarios** aplican factores comunes: el coste a escala se multiplica por 1,16 / 1,00 / 0,86 (pesimista/base/optimista) y el capex por 1,05 / 1,00 / 0,85; el año FOAK se desplaza ±1 sobre una base de 2028. La métrica nivelada es propia del vertical: para almacenamiento, el **LCOS** amortiza el capex de sistema (×1,5) sobre el *throughput* de ciclos (profundidad 0,90; *round-trip* 0,88) más O&M y carga; para el resto, un coste nivelado que añade la amortización del capex por unidad de salida.

Tabla 1. Perfiles por vertical. La unidad y la métrica nivelada difieren; la estructura de cálculo es común.

Vertical	Unidad	Bench.	Opex	Vida	Cpx×
Almacenamiento	$/kWh	80	6 %	15	1,0
Elec. potencia	$/kW	65	7 %	12	1,0
Captura CO_2 (DAC)	$/t	550	10 %	20	2,5
Robótica	$/ud	45.000	14 %	10	1,1
Manuf. espacial	$/kg	900	12 %	10	2,0
Genérico	$/ud	100	8 %	12	1,2

5.3 *Modelo de scoring*

El score compuesto (0–100) combina siete dimensiones (Tabla 2). Tres son **derivadas por el modelo**: madurez (`trl_actual/objetivo`), riesgo de escalado (peor pareja severidad×probabilidad) y economía unitaria, `clamp(40 + arista·124)` con `arista = (benchmark − coste_base)/benchmark`. Cuatro son **juicios de la IA** (IP, cadena, equipo, mercado). El reparto resultante es revelador: **62 % modelo determinista / 38 % juicio de IA** —coherente con «underwrite the physics»—. Tres **pilares** resumen el informe: técnico (0,6·TRL + 0,4·escalado), financiero (= economía) y riesgo (0,7·escalado + 0,3·cadena).

Tabla 2. Dimensiones del score, peso y procedencia.

Dimensión	Peso	Origen
Madurez técnica (TRL)	0,22	Modelo
Riesgo de escalado / FOAK	0,20	Modelo
Economía unitaria	0,20	Modelo
Defensibilidad (IP)	0,12	IA
Cadena de suministro	0,10	IA
Equipo	0,09	IA
Mercado / demanda	0,07	IA

5.4 *Calibración y banco de pruebas*

Como los pesos son un artefacto explícito, la interfaz los expone como **preferencia editable** (Fig. 8). Recalibrar **re-puntúa toda la cartera al vuelo**: la operación es aritmética pura sobre las puntuaciones crudas ya almacenadas (el score crudo no cambia; sólo su peso y el compuesto), por lo que es **reversible** y no vuelve a llamar a la IA. Un **banco de pruebas** ejecuta el modelo financiero sobre parámetros arbitrarios, mostrando cómo reaccionan la TIR, el LCOS y el *score* de economía al instante.

5.5 *Procedencia, BYOK e interfaz*

Cada informe conserva un conjunto de **supuestos** (afirmación, base, confianza 0–100, origen). La vista «Fuentes» los agrega en un libro de evidencia ordenable y en un resumen que separa lo que **calcula el motor** de lo que **viene del data room** (Fig. 6). **BYOK** implica que el servidor no custodia secretos y que el **modo demo es gratuito** (fabrica parámetros localmente, sin llamada al LLM). La capa de presentación adopta una estética de «instrumento» —tema oscuro, acento cobre, rejilla *blueprint*— coherente con la seriedad de una herramienta de underwriting.

6. Las visualizaciones generadas

Un objetivo explícito de ATLAS es que el resultado no sea un volcado de números, sino un **informe legible** que un comité de inversión pueda leer. A partir del JSON del motor, el sistema genera un conjunto de visualizaciones; a continuación se describe cada una y el cálculo que representa. Las figuras 3–8 reproducen esas visualizaciones para el caso de referencia «Celda LFP».

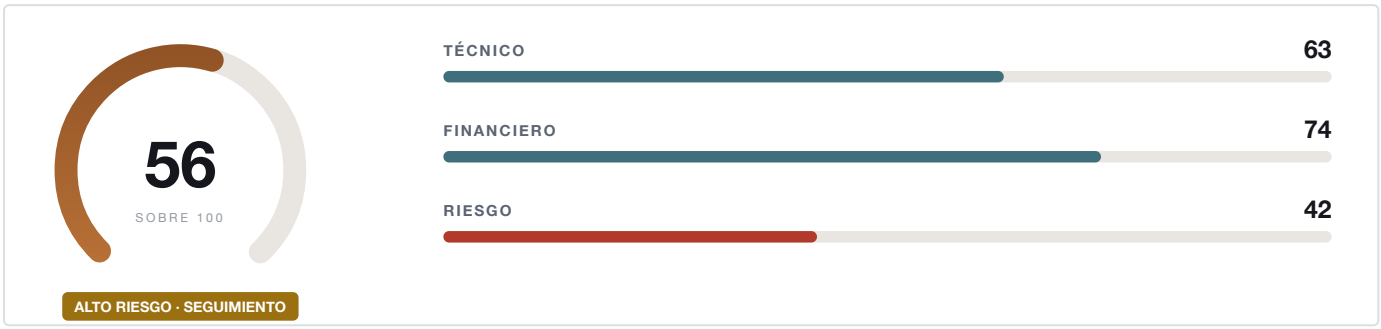

Fig. 3. Veredicto del informe. El **arco de score** (0–100) resume el juicio técnico-financiero y su etiqueta codifica por color la financiabilidad; los tres **pilares** (técnico, financiero, riesgo) son combinaciones deterministas de las siete dimensiones. La tesis y la recomendación (texto) son cualitativas, de la IA.

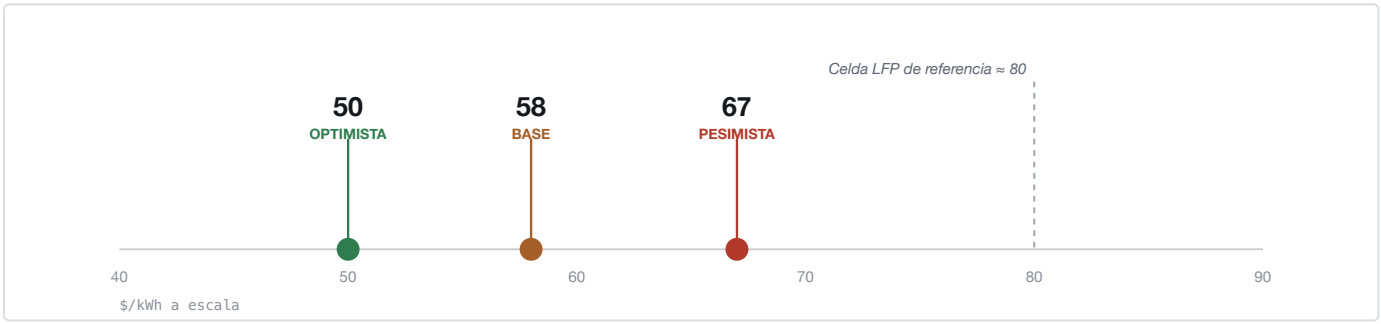

Fig. 4. Gráfico de escenarios. Sitúa el coste a escala en los tres escenarios sobre un eje **escala-consciente** (se adapta a \$/kWh, \$/t CO_2, \$/kg…), frente al *benchmark* del vertical (línea discontinua). La tesis «vive o muere» en el eje del coste.

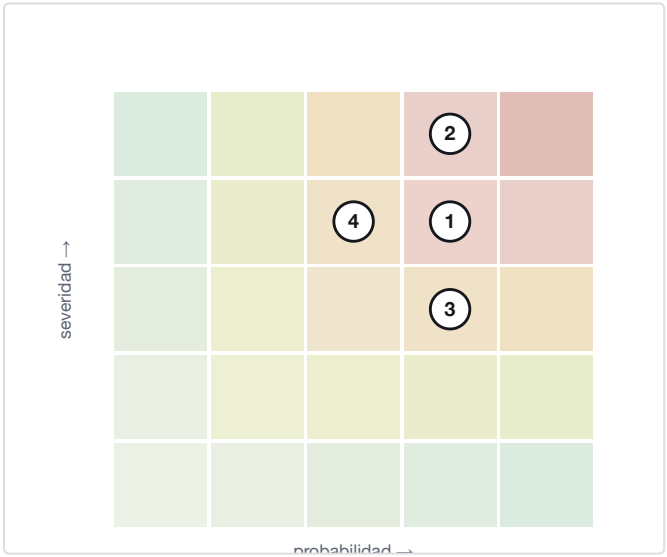

Fig. 5. Matriz de riesgos severidad×probabilidad (5×5). Cada riesgo técnico se ubica y colorea por nivel; los dos peores pares alimentan el score de escalado.

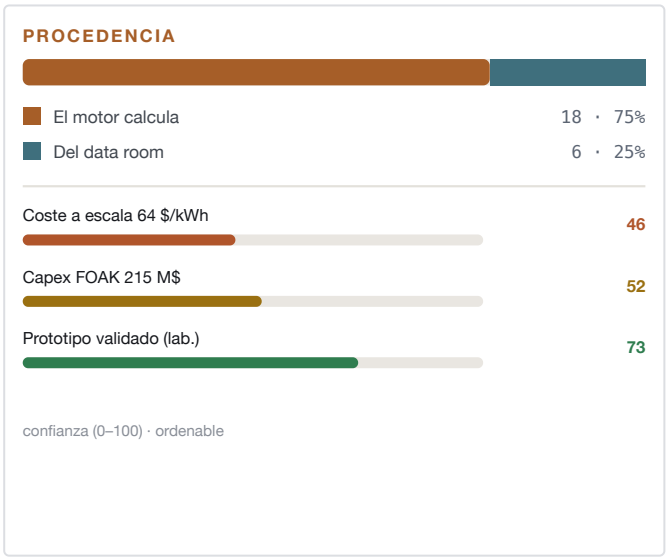

Fig. 6. Rastro de evidencia («Fuentes»). Arriba, la procedencia agregada (qué fracción de las afirmaciones **calcula el motor** vs. **viene del data room**). Debajo, el libro de evidencia: cada afirmación con su barra de confianza, ordenable para sacar a flote lo peor respaldado.

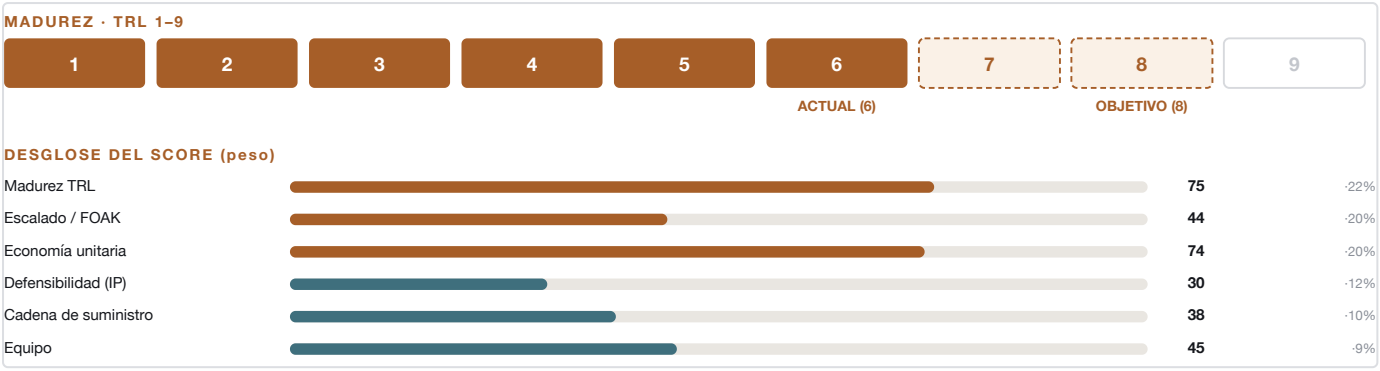

Fig. 7. Escalera TRL (arriba) y desglose ponderado del score (abajo). La escalera marca el nivel actual y el objetivo; el salto es el riesgo FOAK. En el desglose, cada dimensión con su barra y su peso: en cobre las derivadas del modelo, en teal los juicios de la IA. Es la transparencia del veredicto.

Fig. 8. Puesto de control del motor. El reparto **62 % / 38 %** (modelo determinista vs. juicio de la IA) hace visible la tesis «underwrite the physics»; los pesos son una preferencia calibrable.

7. Resultados

7.1 *Ejemplo ilustrativo de extremo a extremo*

La Tabla 3 traza el caso «Celda LFP» a través del motor (cifras internamente consistentes con las fórmulas). Entradas extraídas por la IA: coste hoy 95, objetivo 58, benchmark 80 $/kWh; capacidad 2 GWh/año; capex FOAK 180 M$; vida 4.000 ciclos. Derivados por el motor: LCOS base ≈ 0,072 $/kWh; *score* de economía = 40 + 0,275·124 ≈ 74; compuesto ≈ 56 («alto riesgo · seguimiento»).

Tabla 3. Traza del caso LFP: salida del motor por escenario.

Magnitud	Pesim.	Base	Optim.
Coste a escala ($/kWh)	67	58	50
Margen bruto	16 %	28 %	38 %
TIR 1ª planta	3 %	17 %	33 %
Año FOAK	2029	2028	2027

7.2 *Validación con casos públicos*

Construimos un banco de evaluación con casos reales y de dominio público (celda LFP, sodio-ion, DAC) con parámetros de referencia de fuentes públicas. El modelo determinista, corriendo sobre esos parámetros, cae dentro de la banda pública en **11 de 12** métricas (Tabla 4); las baterías validan limpiamente (8/8) y el DAC captura correctamente su economía adversa de primera planta (TIR negativa —la planta destruye valor a 550 $/t sin créditos de carbono altos—).

Tabla 4. Modelo sobre parámetros de referencia (escenario base) vs. banda pública.

Caso	Coste	TIR	Econ	Banda
LFP	58 $/kWh	17,4 %	74	**4/4**
Sodio-ion	45 $/kWh	29,6 %	94	**4/4**
DAC	230 $/t	–10,3 %	100	3/4

La **fidelidad de extracción** en vivo (Claude Opus 5) fue del **96 %** en la primera corrida —desviación que precisamente **detectó un error real**: la IA convirtió la capacidad de DAC de `20 kt/año` a `20.000`, corrompiendo el coste nivelado y la TIR —. Tras aclarar la unidad en el esquema y el prompt, la re-ejecución alcanzó el **100 % (27/27)**. Que el error fuera *detectable y localizable* comparando el cálculo sobre la referencia frente al cálculo sobre lo extraído es la ventaja concreta de separar extracción y cálculo.

8. Discusión y limitaciones

ATLAS es una contribución de **sistema e ingeniería**, no un estudio empírico. Sus límites son explícitos: no hay validación con *outcomes* reales (que tardan años); los parámetros del modo demo son sintéticos; el modelo financiero es de primera planta, deliberadamente sencillo (una curva de aprendizaje de un parámetro, factores de escenario comunes); y el LCOS omite financiación y degradación, quedando en el lado optimista frente al LCOS de sistema completo. Los pesos por defecto encarnan una tesis razonable pero no óptima —por eso son calibrables—. Como **trabajo futuro**: validación con expedientes reales; benchmarks por vertical con fuentes citadas; modelos de despliegue multiplanta; y la discriminación del score frente a resultados o juicio experto.

Conclusión

El underwriting de deep-tech falla cuando se confunde el juicio con el cálculo. ATLAS los separa: un LLM disciplinado por un esquema tipado **razona y extrae**, y un motor determinista **calcula** toda la economía, con cada cifra trazable a una fórmula y cada afirmación a su origen. El resultado no es un resumidor de documentos más, sino un instrumento de underwriting **auditable, calibrable, desplegado y validado**. La lección trasciende el dominio: para tareas donde la corrección importa, el patrón «la IA razona, el código calcula» convierte la elocuencia del modelo, de riesgo, en valor.

Referencias

1. T. P. Wright, «Factors Affecting the Cost of Airplanes», *J. Aeronautical Sciences*, vol. 3, n.º 4, 1936 — curva/ley de aprendizaje.
2. Coste nivelado de almacenamiento (LCOS): capex de sistema amortizado sobre el *throughput* de ciclos más O&M y energía de carga; cf. Lazard, *LCOS*.
3. J. C. Mankins, «Technology Readiness Levels», *NASA White Paper*, 1995.
4. Anthropic, «Structured outputs», Claude Developer Platform.
5. Pilas tecnológicas: Next.js, FastAPI y Pydantic, SQLite, Docker, Caddy (HTTPS vía ACME/Let's Encrypt).

Reproducibilidad. Todas las constantes citadas (pesos, factores de escenario, constantes de LCOS, perfiles por vertical) son la fuente de verdad del propio motor y se exponen dentro del producto. El banco de evaluación es reproducible con `eval/run_model.py` (sin clave) y `eval/run_extract.py` (BYOK).