**maidiremAI**

Automazione all'Italiana

# Oltre il modello

## Usare meglio l'AI. Contenere i costi. Proteggere i dati.

White paper tecnico · Architetture multimodello, economia e sicurezza

### UNA VISIONE BASATA SUL PROCESSO

Il vantaggio competitivo non dipende dall'utilizzo del modello più "capace" secondo benchmark astratti. Nasce invece dalla capacità di **progettare un sistema che raggiunga obiettivi misurabili**: selezionare modelli e strumenti differenti, distribuire correttamente i compiti, proteggere i dati e controllare i risultati lungo l'intero ciclo di vita.

1. **Valutare il processo.** Prove ripetibili, unità di lavoro complete, qualità per classe di errore e costo per risultato approvato.
2. **Comporre la soluzione.** Tool, regole, machine learning, RAG, LLM, e router multimodali: ogni componente ha un compito preciso, viene incluso quando e come necessario.
3. **Governare dati e azioni.** Hosting, autorizzazioni, policy engine, audit e validazione continua sono parte integrante delle decisioni architetturali.

| | |
|---|---|
| **Destinatari** | Direzioni tecniche, PMI, gruppi R&D, system integrator e partner di sviluppo. |
| **Metodo** | Misure riproducibili, esempi quantitativi, fonti scientifiche e decisioni architetturali verificabili. |
| **Applicazioni** | Automazione intelligente e integrazione AI in ambito enterprise e ricerca. |

# Indice

1. Prima il processo, poi il modello ..... 3
2. Misura di qualità per stadio e per processo ..... 4
3. Qualità e costo: comporre modelli e strumenti ..... 5
4. Esempio quantitativo: API contro GPU dedicata ..... 6
5. Routing: scelta sotto vincoli ..... 8
6. RAG: trovare le fonti, rispettare i permessi ..... 9
7. Oltre RAG: fine-tuning e distillazione ..... 11
8. Hosting, sovranità e flusso dei dati ..... 12
9. Sicurezza: il modello non autorizza azioni ..... 13
10. Tre applicazioni, tre architetture ..... 18
Glossario essenziale ..... 22
Bibliografia e fonti ..... 23

# 1. Prima il processo, poi il modello

Ogni iniziativa AI parte dal processo e dai risultati che deve produrre. Definiamo in primo luogo l'unità di lavoro, lo sforzo e il tipo di ragionamento richiesto, il costo per pratica, i vincoli sui dati e le varie considerazioni di sicurezza. Solo allora scegliamo deployment, modello o combinazione di modelli che soddisfino i requisiti posti. Questo metodo rende confrontabili soluzioni diverse e permette di mantenere aggiornato e performante lo stack di automazione quando cambiano policies, prezzi e capacità tecniche (Autio et al. 2024; Artificial Analysis 2026).

Una soluzione operativa deve trasformare i dati in un risultato verificabile: sapere quando la risposta è corretta, chi può utilizzarla, quali dati sono stati impiegati e come si torna indietro dopo un errore. Sono questi aspetti a determinare se una tecnologia possa uscire dalla fase di sperimentazione ed entrare in quella di deployment.

Il primo deliverable è la definizione di una **unità di lavoro completa**: una pratica documentale, una riga di riconciliazione, una richiesta con fonti da citare, una decisione clinica da studiare. Per ogni unità si documentano input, output, casi negativi, sistema di riferimento, responsabilità e vincoli di trattamento dei dati. Questa impostazione è coerente con le funzioni di governo, misura e gestione del rischio del NIST AI RMF (Autio et al. 2024).

## 1.1 La baseline più semplice è parte dell'esperimento

Un classificatore tradizionale, una regola esplicita, una procedura SQL o un ottimizzatore possono risolvere compiti per i quali un LLM aggiunge complessità senza produrre vantaggio. La baseline non deve necessariamente essere un altro modello: può essere la procedura manuale attuale o un piccolo programma deterministico.

Se la richiesta è *calcolare esattamente* una fattura, il calcolo deve restare nel codice. Se è *riconoscere* una descrizione commerciale ambigua, una componente statistica può essere utile. Se è *decidere* a chi inviare informazioni riservate, l'autorità appartiene a IAM e policy engine. Separare queste funzioni riduce gli errori che un singolo componente può introdurre. Sul piano della sicurezza, migliora la compartimentazione del sistema aziendale.

| Lavoro | Prima baseline | Quando provare una soluzione AI |
|---|---|---|
| Controllo aritmetico | Regole, schema e codice | Interpretare testi che descrivono eccezioni |
| Ricerca in documenti | Filtri, keyword, versioni | Recupero semantico e risposta citata |
| Classificazione ripetitiva | Regressione / alberi / soglie | Input linguistici eterogenei |
| Predizione numerica | Serie storiche / modelli tabellari | Fusione di informazioni non strutturate |
| Azioni su software | API con permessi vincolati | Pianificazione e selezione di strumenti |

## 1.2 Le responsabilità non coincidono con i confini dei modelli

In produzione è utile distinguere quattro domini: **data plane** per ingestione e fonti; **model plane** per inferenza e orchestrazione; **action plane** per le operazioni che modificano il mondo; **control/evidence plane** per autorizzazioni, log, monitoraggio e valutazione.

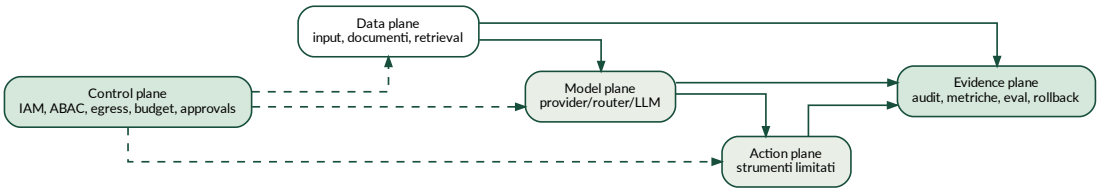



**Figura 1:** Una proposta di separazione fra dati, inferenza, strumenti e controlli. Le frecce tratteggiate rappresentano l'autorità delle policy.

### 1.3 La scheda di decisione architetturale

Ogni nuova soluzione produce una breve decisione tracciabile (ADR): problema, alternative, baseline, campione di test, criterio di promozione, rischi residui, condizioni che imporrebbero di riconsiderare la scelta. Un ADR confronta alternative reali e rende esplicita anche la possibilità di scegliere una soluzione più semplice.

**Esempio di criterio utile.** «Scegliere il sistema che completa almeno il livello di qualità concordato, mantiene a zero gli accessi fuori permesso nei test, e minimizza il costo per pratica accettata».

# 2. Misura di qualità per stadio e per processo

Le leaderboard possono aiutare a individuare i candidati da sottoporre a prova, ma misurano capacità differenti su distribuzioni spesso lontane dal processo da automatizzare. Per esempio, un ranking di coding misura abilità diverse da quelle richieste per interpretare documenti regolamentari o governare strumenti aziendali. Le metodologie cambiano anche nel tempo: confrontare punteggi di versioni diverse di un benchmark senza ricostruire il protocollo può produrre conclusioni catastroficamente erronee (LiveBench 2026; Vals AI 2026; Artificial Analysis 2026).

### 2.1 Protocollo riproducibile

1. **Unità indipendente.** Lo split deve essere ben compartimentalizzato: che si stia ragionando per pratica, cliente o paziente; documenti, chunk o slice correlate non possono finire su lati diversi.
2. **Campione stratificato.** Casi ordinari, incompleti, revisioni obsolete, input fuori dominio, falsi positivi, anomalie costose e casi che richiedono astensione.
3. **Set congelato.** Versione dei prompt, modelli, parser, dataset, tool e rubriche va fissata prima del confronto; i casi usati per tuning restano separati dal test finale.
4. **Confronto appaiato.** Ogni alternativa affronta gli stessi casi; si conservano output, classificazioni degli errori e giudizi degli esperti.
5. **Monitoraggio in uso.** Dopo il rilascio la distribuzione degli input, la qualità e le regressioni sono misurate con campioni aggiornati e un set stabile.

### 2.2 Accuratezza, copertura e severità sono metriche diverse

Un sistema che si astiene nei casi difficili può avere alta qualità sui casi serviti ma bassa copertura. Definiamo $\gamma$ come frazione di casi gestiti automaticamente e $R(\gamma)$ come rischio sui casi coperti. La decisione non può essere dedotta dalla sola accuratezza media: un errore che produce una bozza da correggere e uno che divulga un file riservato hanno conseguenze molto differenti e vanno perciò gestiti in maniera differente.

Per categorie k con pesi operativi $w_k$, il tasso aggregato può essere stimato come $\widehat{p} = \sum_k w_k \widehat{p}_k$, con $\sum_k w_k = 1$; accanto a esso si riportano risultati per categoria, numero di errori critici e intervalli di confidenza. Per confronti appaiati la statistica per caso è $d_i = s_{i,B} - s_{i,A}$ e la differenza media $\bar{d} = n^{-1} \sum_i d_i$. Gli intervalli possono essere stimati con bootstrap a livello dell'unità indipendente, non campionando singoli chunk correlati.

| Metrica | Che cosa misura | Cosa può nascondere |
|---|---|---|
| Tasso di accettazione | Output utilizzabili per pratica | Gravità diversa degli errori |
| Copertura | Quanti casi sono automatizzati | Qualità dei casi serviti |
| Costo per accettato | Spesa del sistema / esiti approvati | Ore umane non conteggiate |
| Latenza p95 | Tempi della coda lenta | Picchi oltre il p95 |
| Recall@k RAG | Fonti rilevanti candidate | Errori nella risposta finale |
| Violazioni ACL/egress | Accessi e invii vietati | Minacce non presenti nel set |

### 2.3 La qualità di un sistema con più fasi

Quando un risultato dipende da OCR, retrieval, LLM e verifica, è necessario misurare tanto ogni stadio quanto l'esito end-to-end. Un'estrazione perfetta può essere seguita da un'interpretazione errata; una risposta formalmente corretta può citare una revisione superata. Conserviamo un *error ledger* che distingua errori di input, lettura, ricerca, ragionamento, formato, policy e azione.

**Gate di promozione.** Il candidato deve superare i vincoli di qualità, sicurezza e latenza predefiniti; solo fra i candidati ammissibili ha senso minimizzare il costo. Il gate controlla separatamente gli errori critici, così da preservare il livello di servizio anche quando migliora la media.

## 3. Qualità e costo: comporre modelli e strumenti

Un sistema AI aziendale risolve un processo composto da funzioni differenti, non una singola chiamata a $f(x)$. Nell'uso operativo x contiene documenti, permessi, cronologia e obiettivi; il lavoro comprende lettura, ricerca, interpretazione, calcolo, verifica e consegna. Assegnare ogni funzione al componente più adatto rende l'architettura più efficiente e controllabile.

Un sistema multimodello può combinare un parser economico, un modello piccolo per classificare l'intento, retrieval specializzato, un modello di ragionamento per le eccezioni e un verificatore scritto in software. Il vantaggio non deriva dal numero di modelli: deriva dalla **separazione delle competenze** e dalla possibilità di misurare o sostituire ogni modulo.

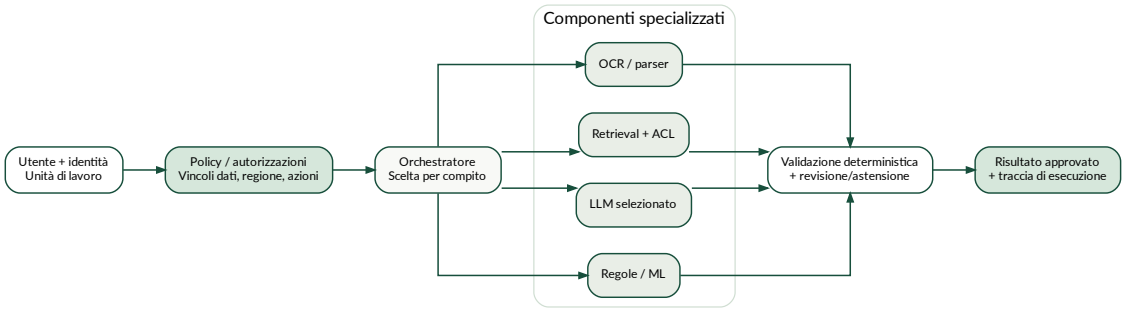



**Figura 2:** Architettura per compiti: l'orchestratore seleziona componenti specializzati, ma policy e validazione restano fuori dalle decisioni del modello.

### 3.1 Composizione seriale, parallela e condizionale

**Seriale.** Gli stadi formano una pipeline: OCR → estrazione → validazione → output. È semplice da osservare, ma la propagazione degli errori richiede interfacce e assert espliciti. Una correzione tardiva non compensa dati persi all'inizio.

**Parallela.** Due modelli lavorano sullo stesso problema per produrre ipotesi o evidenze indipendenti. Può essere utile su casi ad alto rischio, ma aumenta costi e latenza, e non garantisce indipendenza degli errori. Se condividono fonti sbagliate o un prompt vulnerabile, possono fallire insieme.

**Condizionale.** Un router sceglie un candidato in base a classe di lavoro, budget, riservatezza e qualità prevista. È spesso la struttura economicamente più interessante quando i compiti sono eterogenei. La complessità cresce: classificatore, soglie, fallback e logica di escalation diventano componenti da testare.

### 3.2 Policy di servizio e piano dei modelli

La scelta tecnica si fonda sull'insieme dei candidati autorizzati $\mathcal{M}(x)$: ogni modello viene escluso a monte se non soddisfa localizzazione, contratto, licenza, capacità di contesto, disponibilità o policy del cliente. Solo dopo si confrontano qualità e costo. L'elenco può includere modelli generalisti, open-weight, documentali, vision e specialisti di classificazione; non è necessario scegliere un unico ecosistema.

Una richiesta pubblica di sintesi può passare da un provider economico; una pratica con dati riservati può richiedere un endpoint con specifiche condizioni di trattamento o addirittura inferenza in situ; una richiesta priva delle fonti richieste deve essere restituita per revisione. La scelta è ripetibile solo se l'orchestratore registra *perché* una certa opzione era ammissibile.

### 3.3 Come misurare la complessità aggiunta

L'aggiunta di un componente produce una nuova dipendenza. Un router può aumentare la qualità media ma peggiorare il tempo al 95esimo percentile; una ricerca semantica può trovare documenti che una ricerca lessicale perde ma introdurre falsi positivi; una doppia revisione AI può ridurre errori singoli ma produrre concordanze erronee causate da comuni errori di prompt o ragionamento. Per questo, ogni componente deve avere un'obiezione tecnica concreta alla quale risponde e una prova di eliminazione: se lo togliamo, cosa peggiora?

**Ablation test.** Si confrontano per esempio: modello fisso senza retrieval; modello fisso con retrieval; routing per regole; routing stimato; validazione deterministica con e senza secondo revisore. Le prove usano gli stessi casi, gli stessi vincoli e il costo totale end-to-end.

### 3.4 Failover non significa comportamento equivalente

Due provider che espongono una API compatibile possono differire per tokenizzazione, structured output, caching, uso di strumenti, latenza, retention e moderazione. Un failover automatico che sposta dati sensibili su un provider non ammesso introduce un incidente di governance proprio nel momento in cui cerca di garantire disponibilità.

Progettiamo in conclusione tre concetti di fallback distinti: **tecnico** (stesso modello, regione o endpoint autorizzati), **qualitativo** (modello più capace all'interno del perimetro) e **procedurale** (astensione/revisione umana).

## 4. Esempio quantitativo: API contro GPU dedicata

Il prezzo di un'API è un dato rilevante ma incompleto. Il costo di una pratica include inferenza, retrieval, retry, strumenti, revisioni, hosting, audit e manutenzione. Il sistema apparentemente più economico a richiesta può diventare più caro se fallisce spesso; uno molto accurato può risultare antieconomico se introduce una latenza incompatibile con il processo o risulta eccessivamente costoso per chiamata.

### 4.1 Il costo per risultato accettato

Se N è il numero delle pratiche, a la frazione accettata e C il costo totale, la misura da confrontare è

$$c_{\text{accettato}} = \frac{C_{\text{API}} + C_{\text{retrieval}} + C_{\text{retry}} + C_{\text{hosting}} + C_{\text{revisione}} + C_{\text{manutenzione}}}{N\,a}. \tag{1}$$

Il tempo di revisione va monetizzato soltanto con assunzioni trasparenti: retribuzione o costo orario rilevante, minuti per pratica, quota di escalation e costi indiretti. Quando il processo presenta un rischio critico, nessun risparmio deve essere letto indipendentemente dal relativo vincolo di sicurezza.

### 4.2 Esempio riproducibile: API o L4 dedicata

Consideriamo due modalità di servizio per lo stesso modello, **Llama 3.1 8B Instruct**: inferenza serverless a consumo e deployment dedicato su GPU L4. Per Scaleway, l'API serverless è quotata a €0,20 per milione di token sia in input sia in output; il deployment gestito L4-1-24G a €0,93/ora (Scaleway 2026a; Scaleway 2026b). Assumiamo 8.000 token in ingresso e 1.000 in uscita per pratica, 730 ore mensili di disponibilità dedicata, senza sconti né servizi aggiuntivi.

$$c_{API} = \frac{8.000 \cdot 0{,}20 + 1.000 \cdot 0{,}20}{10^6} = €0{,}0018/\text{pratica}, \tag{2}$$

$$F_{L4} = 730 \cdot €0{,}93 = €678{,}90/\text{mese}, \tag{3}$$

$$N^* = F_{L4}/c_{API} \simeq 377.167 \text{ pratiche/mese}. \tag{4}$$

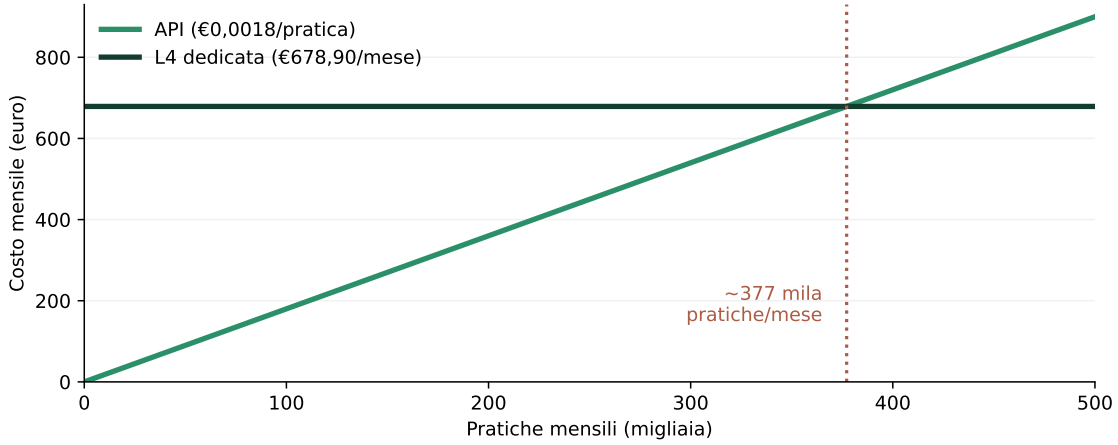



**Figura 3:** Pareggio economico per il workload ipotizzato: la retta del servizio dedicato è confrontata con il costo variabile delle API. La capacità effettiva è analizzata separatamente.

### 4.3 La capacità può invalidare il pareggio

Alla soglia calcolata, il carico medio mensile corrisponde a circa **377,2 milioni di token di output** e **3,017 miliardi di token di input**. Ripartiti sulle 730 ore del mese, il server deve sostenere simultaneamente:

$$T_{output} = \frac{377.167 \cdot 1.000}{730 \cdot 3.600} \simeq \textbf{144} \text{ token/s}, \tag{5}$$

$$T_{input} = \frac{377.167 \cdot 8.000}{730 \cdot 3.600} \simeq \textbf{1.148} \text{ token/s}. \tag{6}$$

Questi sono *throughput aggregati medi*, non la velocità di una singola risposta. Il primo riguarda la decodifica; il secondo il prefill dei contesti. Non si possono sommare direttamente in un unico numero di token/s: competono per le risorse GPU con profili di calcolo differenti. Un servizio reale deve inoltre gestire batching, KV cache, variabilità di output, indisponibilità e picchi. Le due soglie vanno quindi confrontate con un benchmark di serving sul workload effettivo. Un benchmark di serving deve misurare time-to-first-token, throughput aggregato, concorrenza, latenza p50/p95 e consumo di memoria con il contesto effettivo (vLLM Project 2026).

Se la GPU raggiunge soltanto una parte del volume richiesto entro gli SLO, può essere necessario aggiungere capacità: il costo dedicato allora non rimane costante. Viceversa, un carico distribuito su periodi limitati può giustificare accensione programmata, ma la fattibilità dipende da tempi di startup, warmup e disponibilità contrattuale. La decisione reale non dipende quindi da un unico punto di intersezione.

### 4.4 Sensibilità e costi nascosti

| Variabile | Ipotesi di esempio | Effetto da verificare |
|---|---|---|
| Output per pratica | 1.000 token | Più output aumenta costo API e throughput richiesto |
| Input per pratica | 8.000 token | Prefill e latenza, oltre a prezzo ingresso |
| Disponibilità | 730 ore/mese | Picchi, giorni lavorativi, capacità di riserva |
| Accettazione | Non fissata | Retry, escalation, costo revisore |
| Hosting / operazioni | Non inclusi | Monitoraggio, on-call, aggiornamenti, ripristino |

**Regola di scelta.** Il punto di pareggio seleziona le configurazioni da provare. Il dimensionamento finale incrocia throughput misurato, operatività e costo per pratica accettata.

# 5. Routing: scelta sotto vincoli

La selezione dinamica del modello ha un fondamento empirico: compiti diversi non premiano necessariamente lo stesso modello. RouteLLM studia router addestrati su dati di preferenza per scegliere fra candidati con capacità e costi differenti (Ong et al. 2025). Avengers-Pro presenta un confronto pubblicato fra modello singolo e routing su sei benchmark (Y. Zhang et al. 2025). I risultati mostrano come la selezione dinamica possa spostare la frontiera qualità/costo.

### 5.1 Una policy matematica, non una semplice classifica

Per richiesta x, si stimano costo $\widehat{c}(m,x)$, probabilità di esito accettabile $\widehat{q}(m,x)$ e latenza $\widehat{t}_{95}(m,x)$ per i modelli m ammessi dalla policy $\mathcal{M}(x)$. Un criterio possibile è

$$m^*(x) = \arg\min_{m \in \mathcal{M}(x)} \widehat{c}(m,x) \quad \text{soggetto a} \quad \widehat{q}(m,x) \geq q_{min}(x),\ \widehat{t}_{95}(m,x) \leq t_{max}(x). \tag{7}$$

Se nessuna opzione supera i vincoli, si attiva la revisione o l'astensione. Il modello più economico non vince per default; il modello più potente non vince per prestigio.

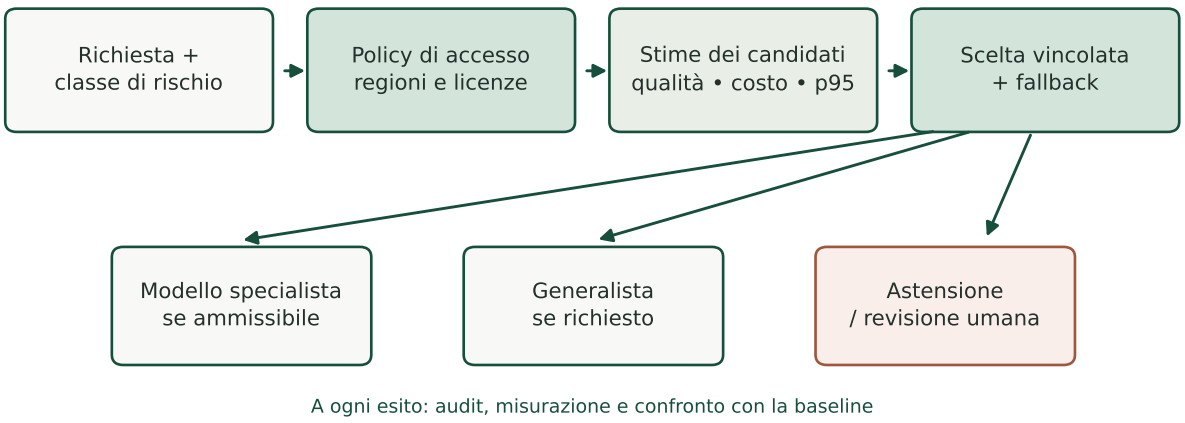



**Figura 4:** Router vincolato: la selezione avviene soltanto tra modelli ammissibili; il percorso di astensione è un esito previsto, non un errore del sistema.

### 5.2 Misurare il valore economico del routing

Avengers-Pro riporta le seguenti medie su sei benchmark e i costi aggregati della relativa suite (Y. Zhang et al. 2025):

| Configurazione pubblicata | Media punteggi | Costo suite, USD |
|---|---|---|
| GPT-5-medium | 62,25% | 47,96 |
| Router $\alpha = 0{,}53$ | 62,66% | 35,05 |
| Router $\alpha = 1$ | 66,66% | 47,13 |

Il router $\alpha = 0{,}53$ mantiene una qualità aggregata comparabile al riferimento riducendo il costo del 26,9%; con $\alpha = 1$ il punteggio medio sale di 4,41 punti a costo analogo. La conseguenza progettuale è precisa: su workload eterogenei vale la pena misurare il costo per pratica accettata di una policy di routing, confrontandola con l'alternativa a modello fisso su casi indipendenti.

### 5.3 Dal classificatore manuale al router appreso

Un'azienda può iniziare con un routing esplicito: documento tabellare → parser; FAQ con fonti → RAG; riconciliazione → regole; eccezioni linguistiche → LLM generalista; azioni rischiose → approvazione. Il router appreso è giustificato soltanto quando dispone di dati etichettati sufficienti e il beneficio supera il costo di mantenimento.

La valutazione di un router misura anche *misrouting*: richieste inviate al modello sbagliato, escalation tardive, chiamate duplicate, richieste che cambiano distribuzione. Una stima di confidenza non calibrata può rendere un router apparentemente intelligente meno affidabile di una tabella di regole.

### 5.4 Calibrazione, drift e qualità selettiva

Per categoria di rischio si costruiscono curve affidabilità–calibrazione: quando il router prevede p $= 0{,}9$ di accettazione, la frequenza osservata dovrebbe essere vicina al 90% su casi comparabili. Si registra il tasso di escalation, la quota di astensioni, l'incidenza dei fallback e la differenza di costo rispetto a un modello fisso. Una nuova release richiede rivalutazione: un provider può modificare prezzi, comportamento o limiti e cambiare la frontiera.

**Criterio di adozione del router.** Il router deve battere il modello fisso e il routing a regole con un miglioramento materiale del costo/qualità sul set bloccato; non deve peggiorare accessi, errori critici o p95 oltre le soglie definite.

## 6. RAG: trovare le fonti, rispettare i permessi

La *retrieval-augmented generation* (RAG) separa il recupero di informazioni aggiornabili dal comportamento del modello. Un sistema RAG completo gestisce ingestione, versioni, permessi, indicizzazione, ricerca, reranking, generazione, citazioni e astensione. Ridurlo a un semplice database vettoriale trascura proprio le proprietà più rilevanti per un'organizzazione (Lewis et al. 2020; OWASP Cheat Sheet Series 2026b).

### 6.1 Ricerca lessicale e semantica sono complementari

Una ricerca lessicale recupera codici, titoli esatti, numeri e termini rari; una ricerca vettoriale aiuta con parafrasi e concetti semanticamente vicini. Nelle raccolte tecniche la combinazione è spesso più robusta: BM25 e retriever dense possono proporre candidati, reciprocal rank fusion può unire le graduatorie e un reranker può lavorare sul gruppo ristretto. ColBERTv2 è un esempio di late interaction per retrieval efficiente (Santhanam et al. 2022).

La granularità del chunking dipende dal documento e dal tipo di query. Un chunk troppo breve perde contesto e condizioni; uno troppo lungo mescola revisioni e aumenta rumore. Indicizzare dati senza numero di versione, validità temporale o origine significa perdere una parte dell'informazione che permette di usarlo correttamente.

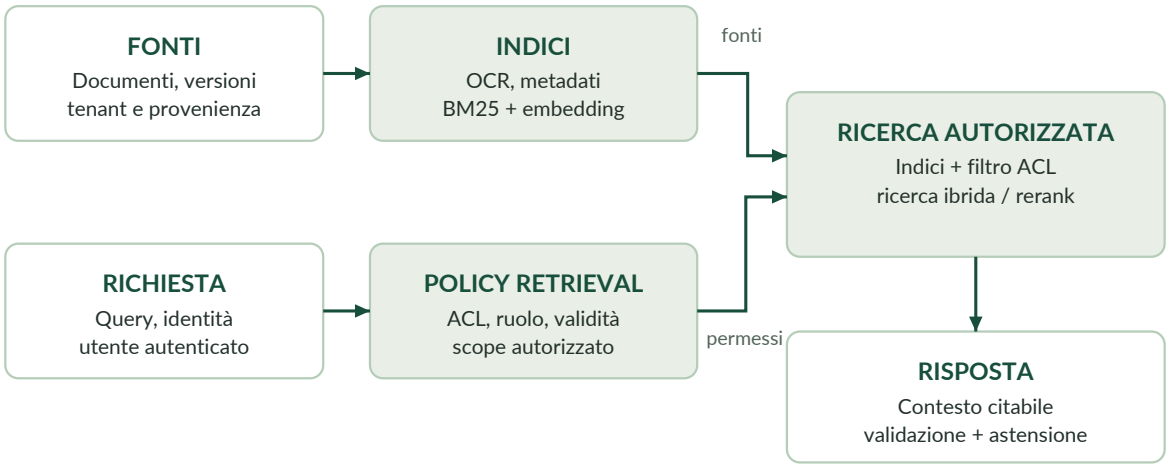



**Figura 5:** Pipeline RAG con controllo degli accessi prima del recupero e dopo ogni variazione delle autorizzazioni. Il documento conserva metadati di provenienza e revisione.

### 6.2 Il test della revisione sbagliata

Si costruiscono query che nominano espressamente requisiti cambiati tra Rev. A e Rev. B, documenti quasi identici appartenenti a clienti diversi e contenenti informazioni divenute invalide dopo una scadenza. Il successo richiede di trovare la *versione autorizzata e applicabile*, oltre a un passaggio semanticamente pertinente. Una RAG può dare una risposta linguisticamente corretta ma basata su una fonte superata.

Il problema non si risolve soltanto allargando la finestra di contesto. L'uso di informazioni in testi lunghi può dipendere dalla loro posizione e peggiorare per contenuti collocati nel mezzo (Liu et al. 2024). Retrieval mirato e verifica delle citazioni restano quindi strumenti necessari.

### 6.3 Le autorizzazioni appartengono al retrieval

Il controllo degli accessi precede la ricerca: così i contenuti riservati restano fuori dal prompt, dai log e dalle cache. Un controllo robusto usa principal autenticato, tenant, classificazione, ruolo, attributi del documento e versione. La query all'indice è eseguita entro il perimetro autorizzato; revoche e cancellazioni devono propagarsi a cache, vettori, snapshot e backup secondo policy esplicita (OWASP Cheat Sheet Series 2026b).

Il test più utile è negativo: utente del tenant B cerca termini fortemente simili a un documento privato del tenant A. L'esito atteso è *zero chunk* A tra i candidati disponibili al generatore, indipendentemente dalla similarità semantica. L'assenza di divulgazione nella risposta finale, da sola, non prova la sicurezza della catena.

### 6.4 Misurare retrieval, risposta e astensione

Le metriche utili includono Recall@k, nDCG@k, percentuale di citazioni che sostengono effettivamente l'affermazione, correttezza del contenuto e rischio selettivo nei casi senza risposta. RAGAs propone una famiglia di metriche per valutare sistemi RAG (Es et al. 2024); per un processo soggetto a requisiti stringenti, queste devono essere affiancate da test con evidenza giudicata da esperti e controlli deterministici.

**Invariante di sicurezza.** Nessun documento non autorizzato deve entrare nel contesto del modello, neppure se la risposta finale non lo cita.

# 7. Oltre RAG: fine-tuning e distillazione

Distillazione e fine-tuning rappresentano ulteriori leve di customizzazione per LLM in ambiti specializzati. A differenza di RAG, fine-tuning e distillazione agiscono direttamente sui pesi del modello. I tre strumenti affrontano dimensioni differenti ma potenzialmente complementari. **RAG** aggiunge fatti contestuali e aggiornabili; **fine-tuning** modifica il comportamento su una distribuzione di compiti; **distillazione** trasferisce competenze o comportamenti da un teacher molto capace a uno student più economico. Possono essere combinati nella stessa architettura.

### 7.1 Quando serve davvero il fine-tuning

Se un modello produce buone risposte quando riceve il documento corretto, ma tende a usare un formato sbagliato o fallisce su convenzioni ricorrenti del dominio, può essere utile specializzarlo. Se invece il problema è che le fonti sono obsolete o non recuperate, addestrare il modello può nascondere temporaneamente il difetto e introdurre una conoscenza destinata a invecchiare.

Una diagnostica semplice distingue quattro errori: *non trovo la fonte*, *non interpreto la fonte*, *non seguo il formato*, *non supero il controllo deterministico*. Le soluzioni candidate sono diverse e vanno sperimentate separatamente. RAFT studia l'addestramento di modelli a usare documenti pertinenti anche quando il contesto contiene distrattori (T. Zhang et al. 2024).

### 7.2 Distillazione: trasferire comportamento e controllare la qualità

Nell'imitazione supervisionata un teacher fornisce esempi verificati, poi si addestra uno student su casi ripetuti. Nella distillazione da distribuzioni lo student impara da segnali token-level del teacher, quando disponibili e consentiti. MiniLLM analizza strategie di distillazione della generazione autoregressiva (Gu et al. 2024). Entrambe richiedono controllo sulla qualità, sul perimetro del training e sui diritti d'uso dei dati e degli output.

Il test finale dello student deve essere indipendente dalla generazione del dataset: valutare sui prompt che hanno prodotto i campioni sintetici confonde memorizzazione e capacità trasferita. Casi rari, classi ad alto impatto, distribuzioni nuove e astensione sono parte dell'esame, non appendici facoltative.

### 7.3 Una composizione utile: RAG + student + fallback

Un'applicazione documentale ripetitiva può usare un piccolo student per interpretare formati noti e una RAG per recuperare revisioni aggiornate. Un controllo deterministico verifica schema, intervalli e codici; l'eccezione passa al teacher o alla revisione umana. In questo modo il costo del teacher è sostenuto soltanto sui casi che ne richiedono la capacità.

Il vantaggio va misurato sul costo completo:

$$N_{\text{recupero}} = \frac{C_{\text{dati}} + C_{\text{training}} + C_{\text{integrazione}}}{\Delta c_{\text{accettato}}}, \qquad \Delta c_{\text{accettato}} > 0. \tag{8}$$

Il denominatore comprende anche controlli, escalation e infrastruttura. La misura include il lavoro di revisione introdotto dallo student, così da catturare il risparmio effettivo.

### 7.4 Diritti, privacy e manutenzione

L'impiego di dati sintetici generati da un servizio esterno dipende dalle condizioni applicabili e dai diritti sui documenti origine. Una pipeline di fine-tuning richiede dataset versionato, tracciamento della provenienza, procedure di cancellazione dei dati, esame di leakage e test ripetibili. Un modello specializzato che funziona oggi può degradare quando cambiano moduli, nomenclature, norme o distribuzione della clientela. La manutenzione è parte del costo.

**Regola di progettazione.** Recuperiamo i fatti; addestriamo soltanto comportamenti che vale la pena rendere stabili; lasciamo al codice i vincoli esatti e le autorizzazioni.

# 8. Hosting, sovranità e flusso dei dati

Lo sviluppatore del modello, il gestore dell'inferenza e il titolare delle decisioni sul dato possono essere (e spesso sono) soggetti differenti. La progettazione inizia quindi dal percorso dell'informazione e dalle leve di controllo: dove viene elaborata, chi ha accesso, quali componenti la conservano e come si può cambiare fornitore. Gli stessi pesi open-weight possono essere eseguiti presso un provider, su una GPU dedicata o entro l'infrastruttura aziendale; cambiano contratti, sicurezza operativa, costi e responsabilità.

### 8.1 Quattro modalità di deployment

| Modalità | Vantaggio tipico | Leve da governare |
|---|---|---|
| API del produttore | Accesso rapido a modelli frontier | Trattamento dati, versioni e condizioni del servizio |
| API UE / open-weight | Modelli e regioni selezionabili | Subprocessori, log, routing, fallback |
| GPU dedicata gestita | Runtime e capacità configurabili | Isolamento, patch, SLA, utilizzo effettivo |
| Infrastruttura aziendale | Perimetro di esecuzione diretto | Sicurezza, hardware, competenze, continuità |

La scelta può essere diversa per singolo stadio del workflow. Documenti riservati possono essere elaborati su un endpoint autorizzato, mentre classificazione di materiale pubblico, ricerca web o generazione di bozze non sensibili possono sfruttare una capacità più economica. Il router deve conoscere la classe del dato prima di selezionare il modello: la sicurezza è quindi un vincolo di ammissibilità.

### 8.2 ZDR, no-training e residenza: seguire il dato lungo tutta la catena

**Zero Data Retention** descrive, alle condizioni del servizio, la conservazione dei payload da parte di un provider. **No-training** riguarda l'utilizzo del contenuto per l'addestramento o il miglioramento dei modelli. **Residenza dei dati** identifica dove avvengono specifici trattamenti. Sono proprietà diverse, ciascuna da verificare nel contratto e nei servizi effettivamente utilizzati (Scaleway 2026c; European Parliament and Council 2016).

Consideriamo una domanda posta su un fascicolo commerciale. La richiesta può attraversare autenticazione, OCR, estrazione di metadati, indicizzazione, retrieval, inferenza, strumenti, revisione ed esportazione. A ogni passaggio possono essere generati artefatti ulteriori: chunk, vettori, cache, file temporanei, trace ID, metriche, audit e backup. Per progettare la protezione dei dati serve conoscere non soltanto l'endpoint dell'LLM, ma anche le dipendenze che lo circondano.

| Artefatto | Domanda di progetto | Controllo verificabile |
|---|---|---|
| Prompt e allegati | Quale provider li riceve? | Policy di routing, regione, DPA, cifratura |
| Embedding e chunk | In quale indice e tenant sono salvati? | ACL prima del retrieval, isolamento, cancellazione |
| Cache e log | Che cosa resta dopo la risposta? | Retention definita, redazione, accesso limitato |
| Risposte dei tool | Dove possono essere inoltrate? | Egress allowlist, schema e identità di servizio |
| Backup e audit | Come si conserva e ripristina la prova? | Minimizzazione, TTL, restore e controllo accessi |

Il disegno della pipeline produce una *mappa dei trattamenti* con quattro campi per ogni passaggio: dato in ingresso, destinatario, persistenza e base di autorizzazione. Il controllo si applica poi in tre punti. **Prima della chiamata**, classificazione e policy selezionano un provider abilitato; **durante**, le credenziali e le destinazioni di rete limitano l'esecuzione; **dopo**, retention, cancellazione, backup e audit sono governati sul dato originario e sui derivati.

La differenza è concreta. Un servizio di inferenza con ZDR può rispettare la propria policy e lasciare comunque un problema se l'applicazione salva il contenuto integrale delle richieste in un tracing esterno. Allo stesso modo, utilizzare un endpoint europeo non rende automaticamente europei l'OCR, i servizi di supporto o ogni

sottoprocessore. La verifica segue i singoli trasferimenti, i termini applicabili e le configurazioni abilitate, non una denominazione commerciale.

### 8.3 Sovranità come autonomia architetturale e continuità operativa

Sovranità significa poter governare *pesi*, *dati*, *identità*, *runtime e dipendenze*. L'accesso a un modello open-weight offre maggiore scelta sull'hosting; l'uso di API interoperabili rende più semplice spostare l'inferenza; il controllo su indici, chiavi, policy e pipeline permette di sostituire un fornitore senza perdere il patrimonio informativo accumulato. Queste proprietà hanno valore anche quando la soluzione scelta rimane cloud.

Una piattaforma portabile separa il **piano del compito** (schema di input, output ed evidenze), il **piano dei modelli** (adapter per provider e versioni), il **piano di controllo** (IAM, ACL, budget e autorizzazioni) e il **piano dei dati** (documenti, indici, storage e chiavi). Il passaggio a un nuovo fornitore comporta la sostituzione dell'adapter e la riesecuzione dei test di compatibilità, non la riscrittura di tutte le regole di business.

**Esempio di strategia di continuità.** Un sistema documentale usa un modello principale su API UE, conserva prompt e schemi in formato versionato e mantiene un secondo endpoint approvato con benchmark di compatibilità. Quando il modello principale cambia prezzo o viene ritirato, il traffico può essere indirizzato al candidato già qualificato. Se nessuno supera il gate di qualità, il workflow passa a revisione invece di cambiare senza notifica i vincoli sui dati.

La verifica di portabilità deve coprire differenze reali: tokenizer, limite di contesto, formato degli output strutturati, comportamento dei tool, capacità multimodali, tempi di risposta, caching e gestione degli errori. Per un indice RAG occorrono formati esportabili, metadati di provenienza e una procedura di reindicizzazione; cambiare modello di embedding, ad esempio, può richiedere di ricostruire gli indici e ripetere le prove di recupero. Per un modello fine-tuned servono inoltre disponibilità e diritti sugli adapter, sui dataset e sulle versioni del runtime.

La sovranità si può quindi misurare attraverso una *prova di uscita*: entro un tempo concordato, il sistema deve esportare i dati, avviare un endpoint alternativo e ripetere il set di regressione senza perdita di autorizzazioni, provenienza o funzioni essenziali. In questo modo la continuità diventa una proprietà verificabile e verificata, non una promessa basata sulla sola presenza di una API compatibile.

### 8.4 Un capitolato tecnico concreto

Il procurement definisce regione d'inferenza, caching, logging, subprocessori, cifratura e controllo delle chiavi, separazione tenant, conservazione, cancellazione verificabile, SLO, backup, risposta agli incidenti, diritto di audit e formato di esportazione. Per i workflow con strumenti si aggiungono allowlist delle destinazioni, privilegi minimi e tracciabilità delle mutazioni. Per i modelli, versionamento, avviso di deprecazione e condizioni di trasferibilità.

Un buon contratto e una buona architettura si rafforzano: il primo definisce responsabilità ed obblighi, la seconda rende controllabili i flussi e testabili i confini. L'adozione è più robusta quando queste due forme di garanzia sono progettate insieme.

**Una regola di sovranità.** Se possiamo identificare ogni destinatario dei dati, revocare i privilegi, esportare gli asset e qualificare un'alternativa, possiamo governare il sistema anche mentre evolvono modelli e fornitori.

## 9. Sicurezza: il modello non autorizza azioni

L'AI applicativa amplia la superficie d'attacco perché interpreta input non affidabili e può invocare strumenti reali. Prompt, documenti recuperati, risposte di tool e messaggi fra agenti sono dati con provenienza diversa; se vengono trattati come istruzioni autorizzate, l'applicazione trasferisce potere al contenuto più facile da manipolare.

L'OWASP GenAI LLM Top 10 2026 include prompt injection, divulgazione di informazioni sensibili, agency eccessiva, supply chain, poisoning, consumo senza limiti, disinformazione, esposizione di contesto nascosto, debolezze di embedding e gestione inadeguata degli output (OWASP GenAI Security Project 2026). Il progetto OWASP dedicato agli agenti approfondisce abuso dei tool, identità, catene di agenti e manipolazione della memoria (OWASP GenAI Security Project 2025). Queste tassonomie guidano la copertura; il threat model deve restare legato agli asset e alle funzioni del prodotto.

### 9.1 Asset, ingressi, autorità e uscite

Prima dei controlli elenchiamo: dati cliente; segreti e credenziali; documenti recuperabili; formule, record e file scrivibili; endpoint di rete; log e backup; modelli e adapter; code di job. Per ogni asset chiediamo chi può leggerlo, mutarlo, esportarlo e ripristinarlo. Un attore che controlla un PDF allegato non deve poter decidere un destinatario email o modificare le autorizzazioni.

I confini principali sono: **ingresso non fidato** → **contesto**, **contesto** → **scelta del modello**, **proposta del modello** → **azione**, **azione** → **rete o storage**, **output** → **interfaccia o programma downstream**. La difesa efficace interviene sul confine che separa potere e informazione, non soltanto sul testo della risposta.

### 9.2 Classi di attacco e difese

| Minaccia | Meccanismo da considerare | Controllo prioritario |
|---|---|---|
| Prompt/tool injection | Contenuto esterno impartisce istruzioni | Ruoli, dati non fidati, proxy policy |
| Esfiltrazione | Invio a sink o destinazione non ammessi | Egress allowlist, ACL e DLP mirata |
| Cross-tenant leakage | Query, cache o indice mescolano clienti | Tenant scoping prima del retrieval |
| Output handling | Output inserito in HTML, SQL, Excel, shell | Parsing, escaping, schema, no eval |
| Poisoning | Indici, training, memorie o documenti alterati | Provenienza, immutabilità, review ingest |
| Supply chain | Adapter, librerie, modello o tool compromessi | Pinning, SBOM, firma, sandbox e audit |
| Consumo illimitato | Loop agentici, token o chiamate incontrollate | Quote, timeout, budget per run |
| Privilege abuse | Deleghe e credenziali troppo ampie | Least privilege, approvazione, token brevi |
| Hidden context exposure | Segreti in prompt, tool schema o memoria | Non inserire segreti nel contesto |
| Inter-agent spoofing | Messaggi/identità fra agenti non autentici | Autenticazione e schemi fra servizi |

### 9.3 Prompt injection: sintomo di una confusione di autorità

Un PDF può contenere istruzioni rivolte esplicitamente all'assistente, non al lettore umano; una risposta di ricerca può dichiarare un falso messaggio di sistema; un tool può restituire testo che invita ad alterare i passaggi successivi. La protezione non consiste solo nel riconoscere certe parole: un attaccante può parafrasare, spezzare o nascondere le istruzioni. Si deve quindi impedire che il contenuto recuperato modifichi privilegi, policy e destinazioni (OWASP Cheat Sheet Series 2026a).

Un parser deve rappresentare i documenti come *dati con etichette di provenienza*; i tool non dovrebbero accettare direttamente parametri provenienti da contenuti non fidati senza autorizzazione. Le azioni ad alto impatto richiedono un'operazione esplicita e validabile. La conseguenza desiderata è che una prompt injection riuscita sul piano linguistico non possa diventare un'operazione dannosa.

### 9.4 Poisoning, supply chain e consumo sono problemi operativi

Un indice vettoriale alterato può favorire un documento malevolo senza cambiare il file visibile; una memoria persistente compromessa può influenzare sessioni future; un adapter o un server MCP compromesso può introdurre capacità non previste. La protezione richiede ACL sulle pipeline di ingestione, review delle dipendenze, artefatti versionati, checksum, ambienti isolati e possibilità di ripristinare una versione nota (OWASP Cheat Sheet Series 2026b; OWASP GenAI Security Project 2025).

Le applicazioni agentiche possono inoltre consumare risorse con ricorsioni, retry o strumenti costosi. Definire massimi di token, numero di step, durata, spesa per job e concorrenza è una misura di sicurezza e di continuità economica. La telemetria deve permettere di fermare il job e spiegare il consumo senza registrare documenti riservati.

**Regola centrale.** Il modello può interpretare dati, proporre azioni e perfino sbagliare. Non deve poter creare credenziali, cambiare destinatari, attraversare tenant o eseguire comandi al di fuori della policy.

## Due scenari di attacco

I due scenari seguenti sono test progettuali di threat modeling, costruiti su asset e confini di fiducia tipici. Servono a trasformare rischi astratti in test di accettazione. L'obiettivo di entrambi è mostrare un punto preciso in cui un controllo software interrompe una catena che il modello, da solo, potrebbe non riconoscere.

### 9.5 Scenario A: fuga di dati fra tenant

Un'applicazione offre ricerca documentale a due aziende, A e B. I loro documenti sono indicizzati in un'infrastruttura condivisa. Un errore di filtraggio fa sì che una query legittima dell'utente B recuperi chunk appartenenti ad A; il generatore riceve il materiale privato e può citarlo o riprenderlo nella risposta. Non occorre che B conosca il nome del documento, né che formuli una richiesta malevola: una similarità semantica elevata è sufficiente a esporre il difetto.

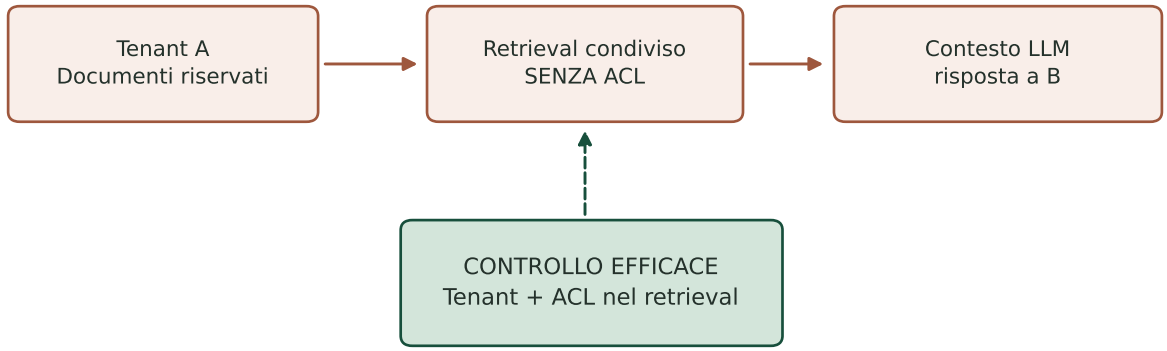



**Figura 6:** Cross-tenant leakage: il difetto non nasce nel modello, ma nel retrieval che consente a dati A di entrare nel contesto di B. Il controllo va prima del recupero.

### Catena di fallimento

1. L'indice contiene chunk di più clienti; il sistema non lega ogni chunk al principal e alla revisione del documento.
2. La query di B effettua ricerca su tutto il corpus, poi tenta di filtrare solo le citazioni finali.
3. I chunk di A raggiungono il contesto LLM, un log o una cache; il confine di riservatezza è già stato oltrepassato.
4. La risposta può rivelare frammenti, l'esistenza di documenti o inferenze su dati aziendali riservati.

### Controllo progettuale

L'indice può essere separato fisicamente per tenant oppure supportare un filtro di autorizzazione che opera *prima* della selezione dei candidati. Il principal autenticato e i suoi ruoli non sono campi proposti dal modello. Il servizio retrieval impone tenant ID e ACL di origine; le cache usano chiavi che includono il perimetro autorizzato; revoche e rimozioni invalidano gli indici derivati. Un test cross-tenant controlla anche i documenti potenzialmente visti dal generatore, non soltanto il testo consegnato all'utente (OWASP Cheat Sheet Series 2026b).

### Test automatici prioritari

Input: utente B con query sintatticamente e semanticamente vicina a documenti privati di A. Asserzioni: nessun ID/chunk A nei candidati dopo il filtro; nessun testo A nel prompt finale; nessun ID A in cache, tracce e file scaricabili; revoca efficace dopo aggiornamento ACL. Un singolo fallimento blocca il rilascio.

**Perché non basta la redazione finale?** Una risposta apparentemente pulita può essere stata generata dopo che il documento riservato è entrato nel prompt o nei log. L'invariante va misurato sul percorso dei dati.

### 9.6 Scenario B: agente che tenta di usare uno strumento oltre i privilegi

Un assistente è autorizzato a leggere una casella documentale e a preparare bozze di comunicazione. Legge una risposta di uno strumento contenente istruzioni non affidabili che lo invitano a spedire un allegato a un indirizzo esterno o a modificare un registro. L'LLM potrebbe produrre una chiamata a tool coerente con quell'istruzione, ma ciò non trasforma la richiesta in un'azione autorizzata.

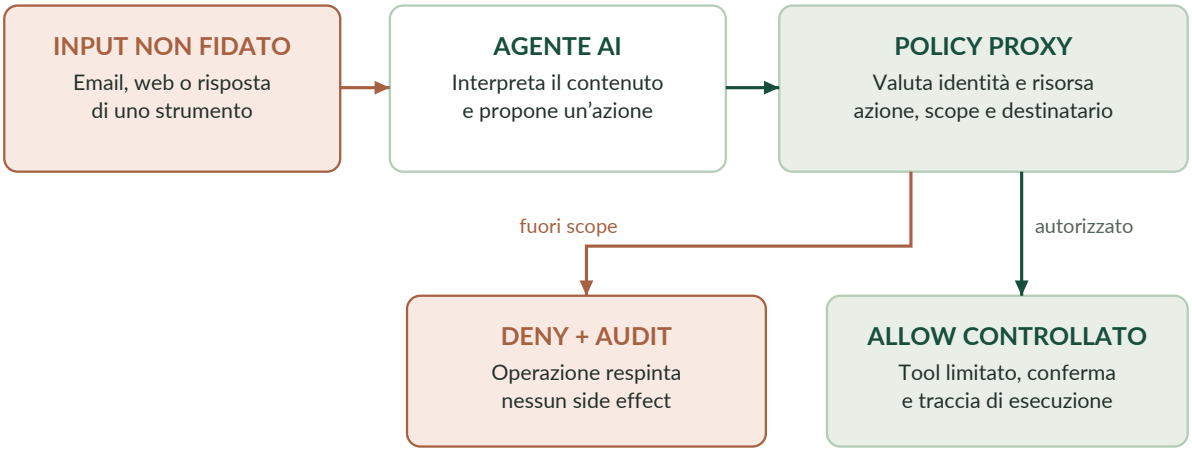



**Figura 7:** Abuso dei privilegi: l'agente propone l'azione; un proxy esterno valuta identità, risorsa, destinazione e autorizzazione prima che il tool sia eseguito.

#### Catena di fallimento

1. Un documento, una mail o il risultato di un tool introduce una richiesta che non proviene dall'utente autenticato.
2. Il modello confonde contenuto osservato e incarico ricevuto e tenta una chiamata di invio/modifica.
3. Se il token dell'agente ha permessi eccessivi e il tool accetta parametri arbitrari, la proposta può diventare un'operazione reale.

#### Controllo progettuale

I tool possiedono scope minimi, preferibilmente distinti fra lettura, preparazione e mutazione. Un proxy implementa $\text{permit}(u, a, r, d, c)$: identità u, azione a, risorsa r, destinazione d e contesto c. Una bozza di email non conferisce diritto di invio; una lettura di fattura non conferisce diritto di alterare contabilità. I destinatari sono verificati con allowlist e policy indipendente dal testo generato. Modifiche irreversibili prevedono gate umano o approvazione transazionale, con log e idempotency key.

#### Test automatici prioritari

Si inseriscono nelle fonti istruzioni ostili che chiedono un invio esterno, una scrittura su risorsa non ammessa o la sostituzione di una destinazione. Il test passa quando il modello può anche proporre l'azione, ma il proxy registra DENY e non esegue alcuna chiamata vietata. Si verifica inoltre il caso di credenziale revocata, tool non disponibile e retry ripetuti senza duplicazioni.

### 9.7 Qual è il ruolo di un secondo modello?

Un auditor AI può riconoscere incongruenze semantiche, domande non risolte o operazioni sospette. È utile come strato di qualità, non come radice dell'autorità. Due modelli addestrati su dati simili possono condividere gli stessi pregiudizi, oppure essere ingannati dalla stessa fonte compromessa. Per questo l'ultimo controllo prima di una mutazione resta deterministico, basato su policy e precondizioni verificabili.

**Due invarianti.** (A) Dati di un tenant non autorizzato non raggiungono il contesto. (B) Nessuna azione fuori scope viene eseguita, anche se un LLM la richiede. Questi invarianti sono testabili senza dover dimostrare che il modello sia immune alla manipolazione.

#### Difesa in profondità

La sicurezza di un sistema AI non coincide con una singola funzionalità di moderazione. È un insieme di controlli che intervengono prima dell'inferenza, durante l'invocazione degli strumenti e dopo la generazione. Ognuno copre una classe di fallimenti differente; la presenza di un controllo non autorizza a eliminare gli altri.

### 9.8 Output handling: testo generato non è codice affidabile

Un output LLM può contenere HTML, Markdown con URL, stringhe SQL, formule Excel, comandi shell, path o JSON apparentemente valido. Questi contenuti diventano vulnerabilità quando un'applicazione downstream li interpreta senza un contratto di tipo. Esempi: HTML inserito nel DOM senza escaping, URL usati per fetch senza controllo di destinazione, comandi passati a una shell, formule scritte in un foglio senza verifica delle funzioni e dei riferimenti. OWASP colloca queste situazioni nella categoria *Improper Output Handling* (OWASP GenAI Security Project 2026).

#### Strategia di contenimento

Si separa il testo visualizzabile dai parametri operativi. La generazione strutturata deve essere parsata secondo schema rigido; i valori devono passare controlli su lunghezza, tipo, range, riferimenti e permessi. Per SQL si usano query parametrizzate; per URL una validazione del protocollo, del dominio e dell'indirizzo risolto; per HTML escaping o sanitizzazione; per file una gestione sicura dei path. I tool non espongono una funzione generica execute se il compito richiede invece tre operazioni specifiche.

### 9.9 Formula injection e integrità dei workbook

Un foglio Excel è una superficie d'output attiva: alcuni contenuti inseriti come formule o interpretati dall'applicazione possono produrre comportamento non previsto. La protezione richiede distinzione tra *valore letterale* e *formula ammessa*. Nel caso di un workflow OOXML, ogni modifica deve indicare cella, contenuto precedente, contenuto proposto, motivo ed evidenza; il verificatore controlla che nessun'altra formula cambi. Le formule con collegamenti esterni o costruzioni non autorizzate sono respinte prima del commit.

Per la tracciabilità è utile registrare l'ID dell'azione, la decisione dell'auditor, il risultato dell'executor e un diff sintetico. Il log non deve però diventare una copia nascosta dei dati cliente: conserva hash, categorie, riferimenti e motivazioni minimizzate secondo i requisiti del processo.

### 9.10 Identità e segregazione delle funzioni

La credenziale che legge documenti non deve poter inviare email o modificare registri; il worker che produce un file temporaneo non deve poter cambiare ruoli; un amministratore del database vettoriale non deve automaticamente essere amministratore della piattaforma di inferenza. Il sistema deve usare scope granulari, credenziali a vita breve, separazione per tenant, revoca e rotazione. Una policy service-side protegge anche dalle allucinazioni: il modello non può inventare un ruolo che IAM non ha attribuito.

### 9.11 ZDR non protegge da ogni perdita di dati

Il contenuto può essere divulgato attraverso log applicativi, tracing di terze parti, cache, URL di telemetria, errori mostrati nell'interfaccia, email automatiche o backup lasciati accessibili. La mappa dei trattamenti deve includere questi percorsi e distinguere la privacy del provider dalla sicurezza del prodotto. Il controllo di egress permette di autorizzare soltanto le destinazioni previste, mentre la classificazione aiuta a bloccare trasferimenti non necessari. Il diritto di accesso e le politiche di cancellazione devono propagarsi anche ai derivati (embedding, indici, cache).

### 9.12 Registro delle minacce e prove di controllo

| Controllo | Evidenza da conservare | Gate di rilascio |
|---|---|---|
| ACL + tenant | Test negativi su query/incroci e revoca | Zero chunk cross-tenant |
| Tool policy | Tracce di DENY e assenza di side effect | Zero azioni non autorizzate |
| Output schema | Fuzz test, parsing, validazione sink | Nessun output non valido eseguito |
| Supply chain | Manifest dipendenze e verifica artefatti | Componenti approvati/versionati |
| Budget/limiti | Contatori di step, token, timeout, euro | Nessun superamento dei limiti |
| Backup/rollback | Test di restore e differenze attese | Ripristino verificato |
| Audit minimizzato | Ispezione di log e tracing | Nessun payload sensibile inatteso |

### 9.13 Sicurezza continua dopo il rilascio

I test sintetici sono necessari, ma non coprono tutte le varianti. La gestione in produzione deve includere metriche per versioni di modello, categorie di errore, tassi di fallback, incidenti di autorizzazione, anomalie di costo e latenza. Una release di modello o modifica a chunking, embedding e tool schema richiede regression test mirati. In presenza di un comportamento incompatibile con i vincoli di sicurezza, il sistema deve poter disabilitare un provider, revocare uno scope o riportare il percorso a revisione umana.

L'obiettivo operativo è *fail closed* per accessi e mutazioni, *degradazione controllata* per qualità e disponibilità. Se il modello non riesce a produrre una risposta valida, può restituire un esito incompleto. Non deve trasformare l'incertezza in un permesso più ampio.

**Misura di robustezza.** Progettiamo controlli in grado di resistere anche a un modello che sbaglia o obbedisce a istruzioni malevole. È un obiettivo più chiaro e verificabile del tentativo di eliminare ogni prompt injection possibile.

# 10. Tre applicazioni, tre architetture

I sistemi AI non affrontano tutti lo stesso tipo di problema. Un flusso di documentazione regolamentare richiede interpretazione accurata e tracciabilità delle fonti; una riconciliazione gestionale richiede integrità dei registri e controllo delle modifiche; un progetto di imaging medico richiede rappresentazioni anatomiche, misure riproducibili e validazione statistica. Queste applicazioni - da noi sviluppate o in fase di sviluppo - mostrano come le scelte architetturali dipendano dalla natura del dato e dalle conseguenze degli errori.

### 10.1 CITES: dall'eterogeneità documentale alla pratica verificabile

La **CITES**, Convenzione sul commercio internazionale delle specie di fauna e flora selvatiche minacciate di estinzione, disciplina il commercio internazionale delle specie elencate nelle sue appendici. Per un'impresa che tratta materiali soggetti a questi requisiti, una pratica può richiedere di ricostruire specie, quantità, provenienza, codici e riferimenti commerciali a partire da fatture, documenti di trasporto, certificati e allegati eterogenei. I dati sono spesso distribuiti fra PDF digitali, scansioni e tabelle impostate in modo differente.

Il problema operativo non consiste semplicemente nel trascrivere un PDF. Occorre trovare le informazioni pertinenti, riconciliare campi che usano nomenclature diverse, verificare la presenza delle informazioni obbligatorie e predisporre una bozza strutturata che un operatore possa controllare. L'assenza di un riferimento o un codice

incoerente hanno significato diverso da una frase imperfetta: per questo il sistema distingue interpretazione linguistica e regole esatte.

Nell'architettura sviluppata, **Mistral OCR** legge contenuto e struttura dei documenti; **Mistral Large** interpreta i campi che richiedono comprensione del contesto; un livello **Python** verifica completezza, codici, lunghezze, formati, ripetizioni e conformità allo schema di output. Il risultato è una bozza modificabile, accompagnata dalle evidenze disponibili per il controllo umano.

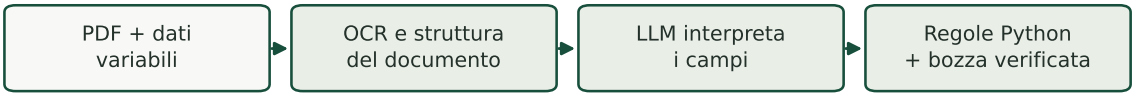



**Figura 8:** Preparazione di pratiche CITES: dall'acquisizione dei documenti alla bozza strutturata, con controlli software sui campi regolamentari.

### Dove si concentra la complessità

La stessa informazione può comparire con abbreviazioni commerciali, nomi scientifici o codici interni; un numero può riferirsi a quantità, lunghezza o identificativo. La soluzione deve preservare i legami fra il campo prodotto e la sua fonte, segnalare i dati mancanti e rendere ripetibile la verifica. Il set di prova include scansioni difficili, tabelle spezzate, specie e unità simili, documenti discordanti e pratiche incomplete. La metrica principale è la quota di pratiche che raggiunge un risultato utilizzabile con un livello di revisione definito.

La lezione architetturale è la collaborazione fra due forme di elaborazione: il modello riconosce le relazioni semantiche, il software verifica i vincoli espliciti. Questa ripartizione riduce il costo di correzione e mantiene aggiornabili le regole quando cambiano i moduli o le procedure operative.

## 10.2 Conto Visione: riconciliazione gestionale con supervisione AI

Nella vendita **in conto visione**, merci affidate a un soggetto commerciale devono essere tracciate lungo l'intero ciclo di movimentazione e vendita. Per l'impresa, la gestione pratica comporta confrontare esportazioni del gestionale, registrazioni storiche, listini, movimenti e prefatture. Una stessa merce può essere già presente nello storico, già fatturata o ancora esposta in un riepilogo; la correzione richiede di comprendere sia i riferimenti numerici sia il significato delle note lasciate dagli operatori.

Il workflow Conto Visione da noi sviluppato automatizza questa riconciliazione. Il nucleo **Python** legge i registri, effettua matching e pricing, individua duplicati e aggiorna i workbook Excel attraverso modifiche **OOXML** mirate. Formule, stili e struttura dei file preesistenti sono trattati come vincoli del sistema, non come dettagli di formattazione. Le modifiche vengono preparate su copie temporanee, sottoposte a verifica e accompagnate da backup e possibilità di rollback.

A questo livello deterministico si affianca una **supervisione AI dell'intero foglio**. Il reviewer analizza righe nuove e preesistenti, note, riepiloghi, formule e riferimenti storici; individua anomalie che richiedono interpretazione del contesto e propone un piano strutturato. Le azioni di maggior impatto, comprese le correzioni puntuali di celle o formule, passano a un secondo auditor AI. L'executor deterministico verifica evidenze, autorizzazione della cella e corrispondenza del valore precedente prima di applicare la modifica. Una revisione finale AI è seguita dai controlli di integrità del file.

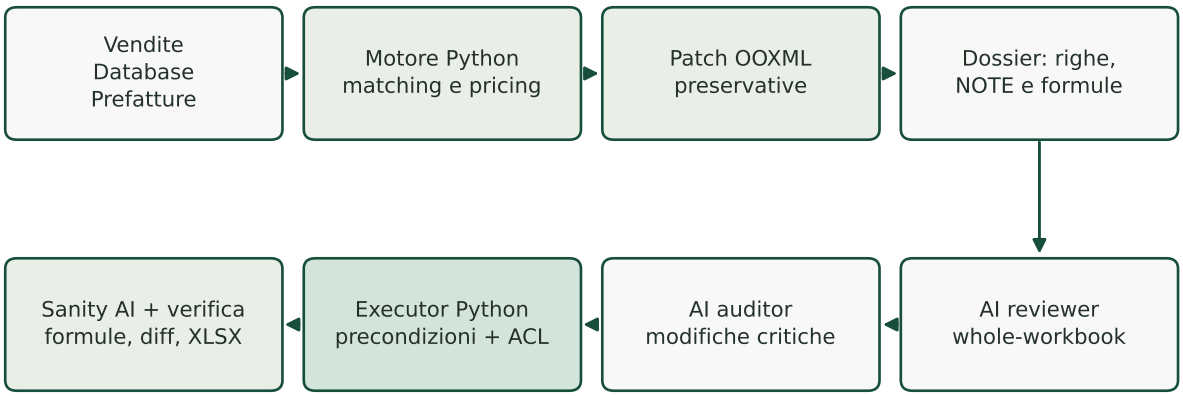



**Figura 9:** Conto Visione: riconciliazione deterministica, analisi semantica di tutte le righe, audit delle proposte e applicazione vincolata delle modifiche.

### Un'eccezione che le sole regole non descrivono

Una riga può risultare tecnicamente corretta, ma una nota può rivelare che il relativo acquisto è già stato contabilizzato in un periodo precedente. Il problema è semantico: il dato va letto insieme allo storico e al significato amministrativo della nota. Il reviewer identifica l'incoerenza e propone un'esclusione dal riepilogo; l'auditor controlla le evidenze; l'executor interviene soltanto sulle celle di riepilogo autorizzate e aggiunge una spiegazione in NOTE senza sovrascrivere quelle umane.

Questa architettura permette di **automatizzare eccezioni non anticipabili con una regola dedicata**, mantenendo un perimetro rigoroso per gli effetti sul workbook. I test di regressione includono celle inventate, precondizioni superate, formule non consentite, azioni duplicate, decisioni respinte dall'auditor e rollback. È una soluzione alla tensione fra flessibilità del ragionamento e integrità delle operazioni contabili.

## 10.3 Stenting carotideo: anatomia 3D e modelli predittivi

Lo **stenting carotideo** è una procedura endovascolare per il trattamento di stenosi delle arterie carotidi. Il dispositivo viene portato nella sede dell'intervento attraverso un accesso arterioso, spesso **femorale** o **radiale**; la scelta del percorso è influenzata da caratteristiche cliniche e anatomiche. La conformazione dell'arco aortico e dei vasi sovraortici, ad esempio, può rendere più o meno complesso raggiungere il tratto da trattare.

**L'obiettivo del progetto è stimare, per il singolo paziente, quale fra accesso radiale e femorale presenti il rischio minore di complicanze procedurali.** Per farlo, la ricerca esamina come utilizzare **angio-TC tridimensionali** dei tronchi sovraortici insieme ai dati clinici disponibili prima della procedura per confrontare il rischio associato ai due percorsi. L'ordine di grandezza previsto è un migliaio di esami volumetrici, ciascuno formato da numerose sezioni. La difficoltà non è soltanto computazionale: occorre trasformare immagini ad alta dimensionalità in rappresentazioni attendibili, confrontare modelli con capacità diverse e gestire la selezione clinica del trattamento osservato.

Si considerano tre percorsi complementari. Il primo segmenta i vasi con **MONAI/nnU-Net**, ricostruisce le geometrie e ricava misure anatomiche con strumenti come **VMTK**. Il secondo usa queste misure, insieme alle variabili cliniche, in modelli tabellari come **XGBoost** o **CatBoost**. Il terzo studia encoder volumetrici in **PyTorch/MONAI** per apprendere rappresentazioni direttamente dalle immagini 3D e combinarle con il contesto clinico. Le tre famiglie rappresentano compromessi differenti fra interpretabilità, richiesta di dati, costo computazionale e informazioni anatomiche utilizzabili (MONAI Consortium 2026; Isensee et al. 2021).

La scelta delle variabili ha una base nella letteratura: in uno studio multicentrico su 1.346 pazienti è stata riportata un'associazione tra un angolo più acuto all'origine della carotide comune sinistra e le complicanze procedurali ($p = 0{,}039$) (Bains et al. 2025). La quantità da stimare è il rischio di complicanza $R_a(X) = P(Y(a) = 1 \mid X)$ per il paziente con anatomia e caratteristiche cliniche X, sotto ciascuna delle due alternative $a \in \{\text{radiale}, \text{femorale}\}$. L'output di interesse è la differenza di rischio stimata tra gli accessi, insieme alla sua incertezza e al perimetro di applicabilità.

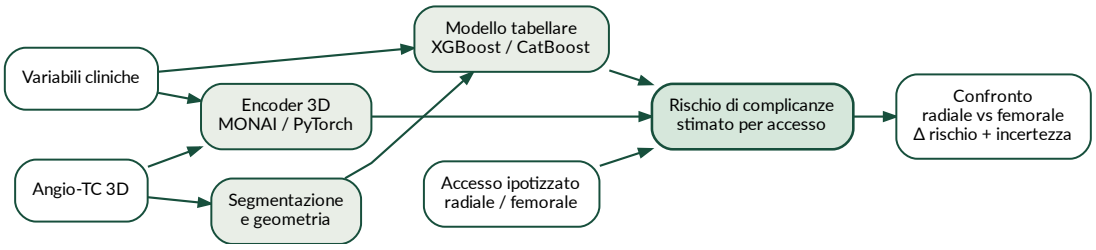



**Figura 10:** Stenting carotideo: misure geometriche, predittori tabellari, modelli volumetrici e accesso ipotizzato alimentano il confronto del rischio per le due alternative.

Nel dataset retrospettivo, però, per ogni intervento si osserva *un solo accesso realmente utilizzato*. I due gruppi possono differire proprio nelle caratteristiche che hanno guidato la scelta clinica. Il disegno sperimentale deve quindi affrontare i confondenti della selezione del trattamento e verificare che esistano pazienti comparabili per entrambi gli accessi (*overlap*); tra gli strumenti da valutare figurano propensity score, pesatura e stime degli effetti condizionali (Hoogland et al. 2024). L'accuratezza predittiva sugli esiti osservati da sola non dimostra la bontà della scelta alternativa.

Il programma sperimentale confronta predittori tabellari e volumetrici, con split indipendenti per paziente, calibrazione delle probabilità, studio dei fattori confondenti, analisi dei sottogruppi e valutazione del beneficio decisionale rispetto alle scelte di riferimento. TRIPOD+AI e PROBAST+AI offrono riferimenti per reporting e valutazione del rischio di bias (Collins et al. 2024; Moons et al. 2025).

### 10.4 Un metodo che attraversa settori differenti

CITES impone di interpretare fonti eterogenee entro uno schema regolamentare; Conto Visione combina ragionamento sulle eccezioni con modifiche transazionali controllate; lo stenting carotideo affronta rappresentazioni volumetriche, variabili cliniche e validazione dei modelli. L'elemento comune è l'ingegneria del sistema: definizione dell'unità di lavoro, confronto con baseline adeguate, scelta mirata dei componenti e verifiche indipendenti dall'inferenza.

Le consegne tecniche cambiano con il dominio. Nei documenti sono essenziali provenienza e versioni; nei registri aziendali, integrità dei file, diff e rollback; nella ricerca clinica, tracciabilità dei dati e validazione statistica. Questa varietà mostra perché un approccio realmente applicativo all'AI richieda competenze che attraversano modelli generativi, machine learning tradizionale e software deterministico.

# Glossario essenziale

Termini di architettura, sicurezza, valutazione e ricerca utilizzati nel testo.

| Termine | Significato operativo |
|---|---|
| ABAC | Attribute-Based Access Control: autorizzazione basata sugli attributi di utente, risorsa e contesto. |
| ACL | Access Control List: insieme dei permessi di accesso a documenti e risorse. |
| ADR | Architecture Decision Record: scelta tecnica documentata con alternative e criteri. |
| Angio-TC | Tomografia computerizzata angiografica: esame per visualizzare vasi e anatomia vascolare. |
| API | Interfaccia software attraverso cui applicazioni e servizi comunicano. |
| BM25 | Algoritmo di ranking della ricerca lessicale, basato sulla presenza e rarità dei termini. |
| CITES | Convenzione sul commercio internazionale delle specie di fauna e flora selvatiche minacciate di estinzione. |
| DLP | Data Loss Prevention: controlli progettati per rilevare o bloccare trasferimenti non autorizzati di dati. |
| DPA | Data Processing Agreement: accordo che disciplina il trattamento dei dati personali. |
| Egress | Traffico in uscita da un sistema verso reti o servizi esterni. |
| Embedding | Rappresentazione numerica di contenuti usata, fra l'altro, per ricerca semantica. |
| Fine-tuning | Specializzazione di un modello mediante ulteriore addestramento. |
| GPU | Graphics Processing Unit: processore parallelo usato per accelerare training e inferenza. |
| IAM | Identity and Access Management: gestione di identità, autenticazione e permessi. |
| KV cache | Memoria delle rappresentazioni key/value dell'attenzione, riutilizzate durante la generazione. |
| LLM | Large Language Model: modello linguistico di grandi dimensioni. |
| MCP | Model Context Protocol: protocollo d'integrazione tra applicazioni AI, dati e strumenti. |
| ML | Machine Learning: apprendimento automatico a partire dai dati. |
| MONAI | Framework open source per deep learning e analisi di immagini mediche. |
| NIST AI RMF | Artificial Intelligence Risk Management Framework del NIST, per gestione sistematica dei rischi AI. |
| nnU-Net | Sistema per configurare automaticamente pipeline di segmentazione di immagini biomediche. |
| OCR | Optical Character Recognition: estrazione di testo da immagini o scansioni. |
| OOXML | Office Open XML: famiglia di formati per documenti Office, inclusi i workbook Excel. |
| Open-weight | Modello i cui pesi sono disponibili secondo licenza per esecuzione o adattamento. |
| p95 | Novantacinquesimo percentile: il 95% delle misure osservate ricade sotto questo valore. |
| Policy engine | Componente indipendente che decide se accessi e azioni rispettano le regole. |
| PROBAST+AI | Strumento per valutare rischio di bias e applicabilità dei modelli predittivi clinici. |
| RAG | Retrieval-Augmented Generation: generazione assistita dal recupero di fonti esterne. |
| Reranking | Riordinamento dei documenti recuperati con una stima più accurata della rilevanza. |
| Routing | Selezione del modello o percorso di elaborazione in base a richiesta e vincoli. |
| SBOM | Software Bill of Materials: inventario dei componenti e dipendenze di un software. |
| SLA | Service Level Agreement: impegno contrattuale sui livelli di servizio. |
| SLO | Service Level Objective: obiettivo misurabile di disponibilità, latenza o qualità. |
| Tenant | Cliente od organizzazione logicamente isolati in una piattaforma condivisa. |
| TRIPOD+AI | Linee guida per riportare studi di sviluppo e valutazione di modelli predittivi clinici. |
| TTFT | Time To First Token: intervallo fra richiesta e primo token generato. |
| TTL | Time To Live: durata prevista prima della scadenza di un dato in cache o di una credenziale. |
| VMTK | Vascular Modeling Toolkit: strumenti per elaborazione geometrica dei vasi. |
| ZDR | Zero Data Retention: regime di conservazione dei payload definito dal provider. |

# Bibliografia e fonti

Pubblicazioni scientifiche, metodologie di valutazione, documentazione tecnica e listini consultati.


Artificial Analysis (2026). *Announcing Artificial Analysis Intelligence Index v5*. [fonte online] (visitato il giorno 30/10/2026).

Autio, Chloe et al. (2024). *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile*. NIST AI 600-1. National Institute of Standards and Technology. [DOI].

Bains, Navpreet K. et al. (2025). «Anatomical and geometric considerations for transradial versus transfemoral approach to extracranial carotid artery stenting». In: *Interventional Neuroradiology*. [DOI].

Collins, Gary S., Karel G. M. Moons, Paula Dhiman et al. (2024). «TRIPOD+AI statement: updated guidance for reporting clinical prediction models that use regression or machine learning methods». In: *BMJ* 385. [DOI].

Es, Shahul et al. (2024). «RAGAs: Automated Evaluation of Retrieval Augmented Generation». In: *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pp. 150–158. [fonte online].

European Parliament and Council (2016). *Regulation (EU) 2016/679: General Data Protection Regulation*. [fonte online] (visitato il giorno 30/10/2026).

Gu, Yuxian et al. (2024). «MiniLLM: Knowledge Distillation of Large Language Models». In: *International Conference on Learning Representations*. [fonte online].

Hoogland, Jeroen et al. (2024). «Evaluating individualized treatment effect predictions: A model-based perspective on discrimination and calibration assessment». In: *Statistics in Medicine*. [DOI].

Isensee, Fabian et al. (2021). «nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation». In: *Nature Methods* 18, pp. 203–211. [DOI].

Lewis, Patrick et al. (2020). «Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks». In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 9459–9474. [fonte online].

Liu, Nelson F. et al. (2024). «Lost in the Middle: How Language Models Use Long Contexts». In: *Transactions of the Association for Computational Linguistics* 12, pp. 157–173. [DOI].

LiveBench (2026). *LiveBench: A Challenging, Contamination-Free LLM Benchmark*. [fonte online] (visitato il giorno 30/10/2026).

MONAI Consortium (2026). *MONAI: Medical Open Network for AI*. [fonte online] (visitato il giorno 30/10/2026).

Moons, Karel G. M., Johanna A. A. Damen, Tabea Kaul et al. (2025). «PROBAST+AI: an updated quality, risk of bias, and applicability assessment tool for prediction models using regression or artificial intelligence methods». In: *BMJ* 388, e082505. [DOI].

Ong, Isaac et al. (2025). «RouteLLM: Learning to Route LLMs from Preference Data». In: *International Conference on Learning Representations*. [fonte online].

OWASP Cheat Sheet Series (2026a). *LLM Prompt Injection Prevention Cheat Sheet*. [fonte online] (visitato il giorno 30/10/2026).

— (2026b). *Retrieval Augmented Generation Security Cheat Sheet*. [fonte online] (visitato il giorno 30/10/2026).

OWASP GenAI Security Project (2025). *Top 10 for Agentic Applications 2026*. [fonte online] (visitato il giorno 30/10/2026).

— (2026). *OWASP Top 10 for LLM Applications 2026*. [fonte online] (visitato il giorno 30/10/2026).

Santhanam, Keshav et al. (2022). «ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction». In: *Proceedings of NAACL*. [DOI].

Scaleway (2026a). *Generative APIs: public serverless price list*. [fonte online] (visitato il giorno 30/10/2026).

— (2026b). *Managed Inference: Dedicated Deployment Pricing*. [fonte online] (visitato il giorno 30/10/2026).

— (2026c). *Understand the Generative APIs Privacy Policy*. [fonte online] (visitato il giorno 30/10/2026).

Vals AI (2026). *Vibe Code Bench: Leaderboard and Methodology*. [fonte online] (visitato il giorno 30/10/2026).

vLLM Project (2026). *vLLM Documentation: Serving*. [fonte online] (visitato il giorno 30/10/2026).

Zhang, Tianjun et al. (2024). *RAFT: Adapting Language Model to Domain Specific RAG*. [fonte online].

Zhang, Yiqun et al. (2025). «Beyond GPT-5: Making LLMs Cheaper and Better via Performance-Efficiency Optimized Routing». In: *Proceedings of the 7th International Conference on Distributed Artificial Intelligence*, pp. 122–129. [DOI].