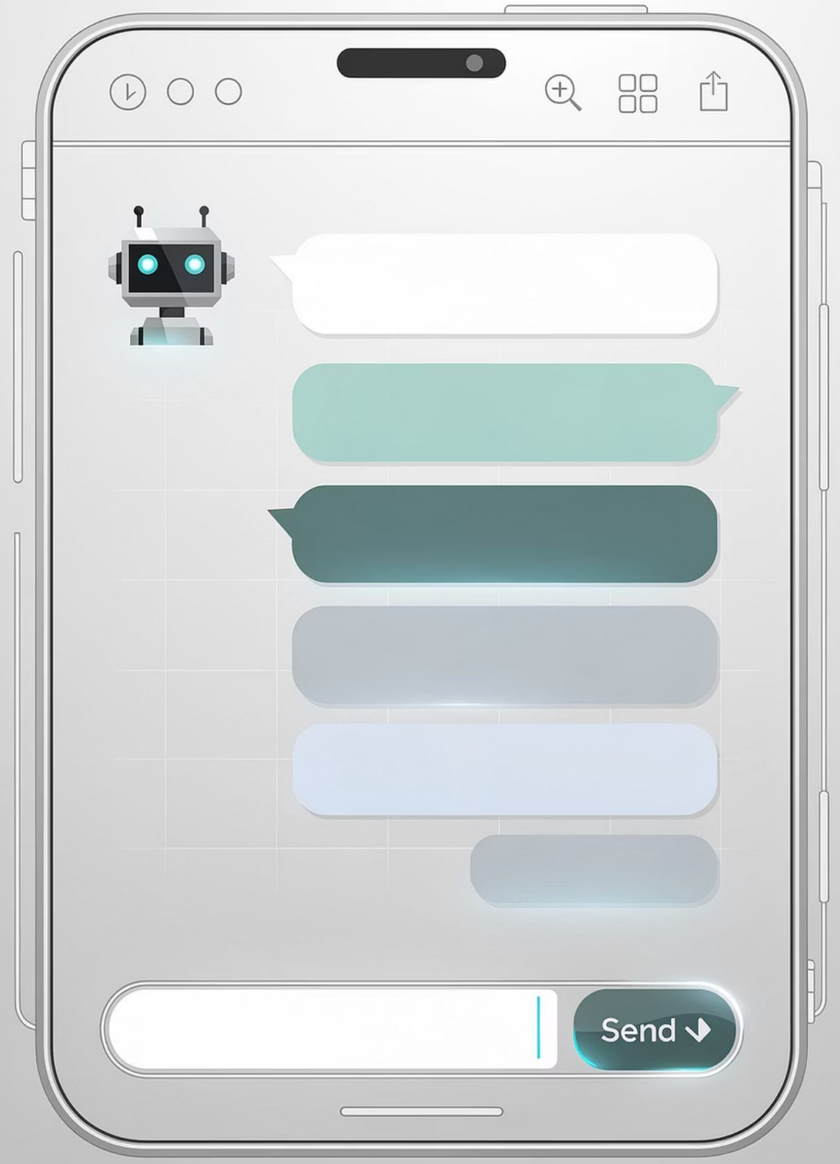


Chatbot Intelligenti e Large Language Models

Dalla teoria alla pratica

Il futuro dell'interazione digitale

Questo corso rappresenta un viaggio attraverso l'evoluzione dei chatbot, dalle semplici regole predefinite all'intelligenza artificiale avanzata basata su Large Language Models. Esploreremo come queste tecnologie stanno trasformando il modo in cui cittadini e pubbliche amministrazioni comunicano.

Costruiremo le fondamenta necessarie per comprendere e implementare chatbot moderni.

Argomenti

- Evoluzione dei chatbot
- Architetture LLM moderne
- Strategie implementative
- Casi d'uso concreti

L'Agenda della sezione 1

Evoluzione dei Chatbot

Dalle regole semplici ai modelli di linguaggio avanzati

Large Language Models

Architetture, capacità e limitazioni dei LLM moderni

Locale vs Cloud

Analisi comparativa delle strategie di deployment

Casi d'Uso PA

Applicazioni concrete nella Pubblica Amministrazione

Architettura Moderna

Componenti e pattern architetturali

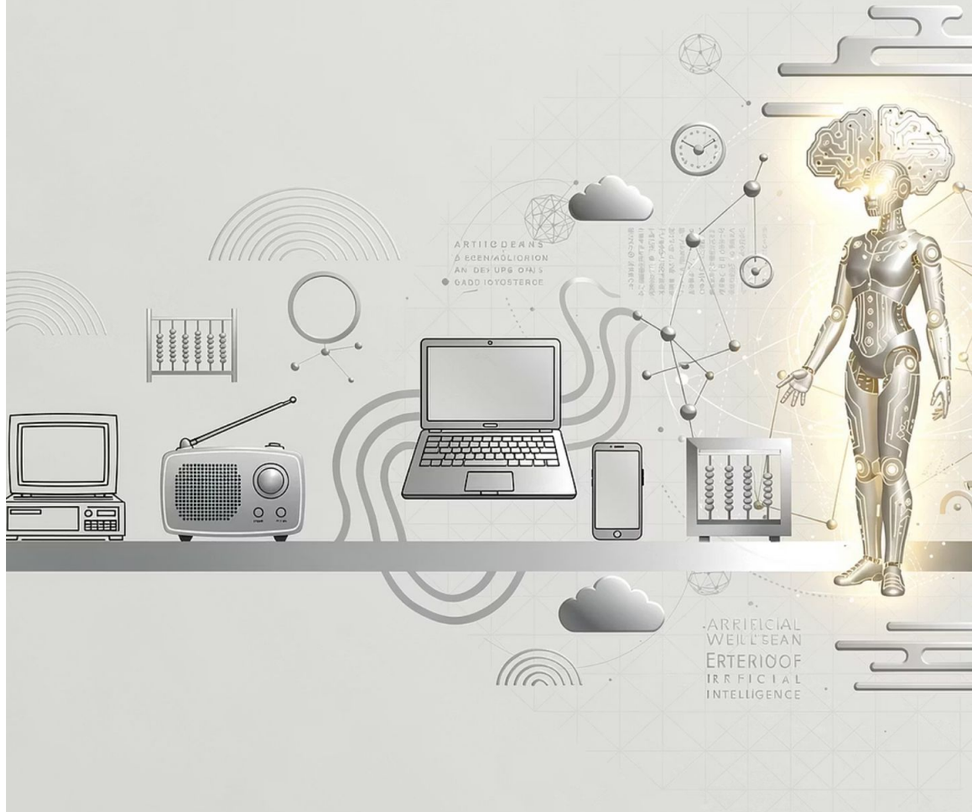
Roadmap Implementativa

Dal prototipo alla produzione

L'Evoluzione dei Chatbot

La storia dei chatbot è un racconto di crescente sofisticazione tecnologica. Dalle primitive regole if-then degli anni '90, siamo arrivati a sistemi capaci di comprendere il linguaggio naturale, mantenere conversazioni complesse e integrarsi con ecosistemi aziendali.

Questa evoluzione non è solo tecnologica: rappresenta un cambiamento fondamentale nel modo in cui concepiamo l'interazione uomo-macchina.



Le Tre Generazioni dei Chatbot

Prima Generazione: Rule-Based

1

Anni '90-2000 · Pattern matching rigidi e alberi decisionali predefiniti.

Seconda Generazione: NLP Tradizionale

2

2010-2020 · Introduzione di intenti ed entità attraverso machine learning supervisionato.

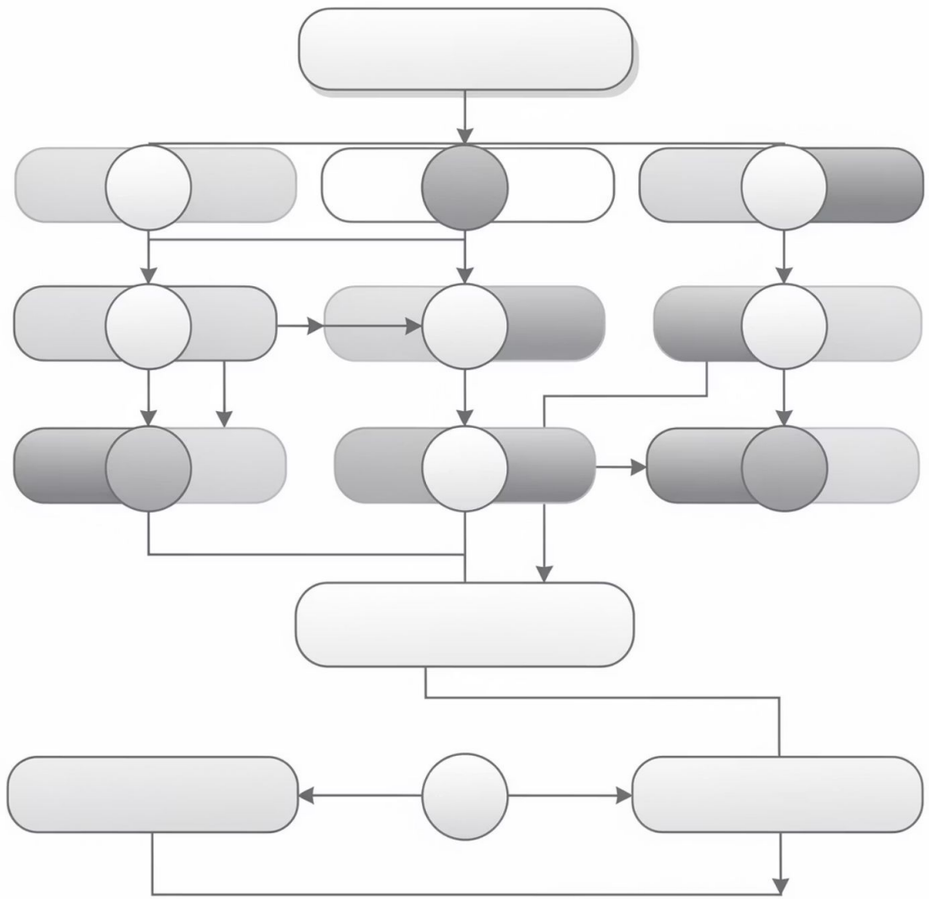
Si diffondono maggiormente gli assistenti virtuali aziendali.

Terza Generazione: LLM-Based

3

2020-presente · Large Language Models pre-addestrati con comprensione contestuale avanzata, ragionamento e capacità generative.

Prima Generazione: Chatbot Rule-Based



Caratteristiche Principali

I chatbot di prima generazione operavano su logiche deterministiche estremamente rigide. Ogni possibile input dell'utente doveva essere previsto e mappato manualmente a una risposta predefinita.

- Pattern matching basato su parole chiave
- Alberi decisionali con percorsi fissi
- Zero comprensione semantica
- Manutenzione manuale costante

Esempio: se l'utente scriveva "orari apertura", il sistema rispondeva con gli orari. Ma "quando siete aperti?" non veniva riconosciuto.

Seconda Generazione: l'era del NLP tradizionale

Intenti ed Entità

I sistemi hanno imparato a riconoscere **cosa** vuole l'utente (intento) e **su cosa** (entità), permettendo maggiore flessibilità linguistica.

Machine Learning Supervisionato

Training su dataset etichettati manualmente per migliorare il riconoscimento, ma con costi elevati di preparazione dati.

Piattaforme Dedicate

Dialogflow, Rasa, BotPress e Microsoft Bot Framework hanno democratizzato lo sviluppo di chatbot aziendali.

Nonostante i progressi, questi sistemi richiedevano dataset estesi, faticavano con variazioni linguistiche impreviste e mantenevano una gestione limitata del contesto conversazionale a lungo termine.

Terza Generazione: LLM

L'avvento dei Large Language Models ha rappresentato un salto quantico nella capacità dei chatbot. Modelli come GPT-4, Claude e Llama non sono semplicemente "più intelligenti": operano secondo paradigmi completamente diversi, con comprensione contestuale profonda e capacità di ragionamento emergenti.

Zero-Shot Learning

Funzionano immediatamente senza training specifico sul dominio

Gestione Contestuale

Mantengono coerenza attraverso conversazioni articolate e prolungate

Multilingua Nativo

Supporto naturale per decine di lingue senza configurazione aggiuntiva

Ragionamento Avanzato

Capacità di inferenza logica e generazione creativa di soluzioni

Confronto Pratico: stesso scenario

Scenario: Un cittadino chiede "Quando posso ritirare il documento che ho richiesto la settimana scorsa?"

Rule-Based

Cerca "ritirare" + "documento" → risposta generica predefinita. Non capisce "la settimana scorsa" e non può verificare lo stato specifico.

NLP Tradizionale

Riconosce intento RICHIESTA_INFO_RITIRO ed entità "documento". Risponde solo se questo scenario è stato previsto nel training. Ignora il riferimento temporale.

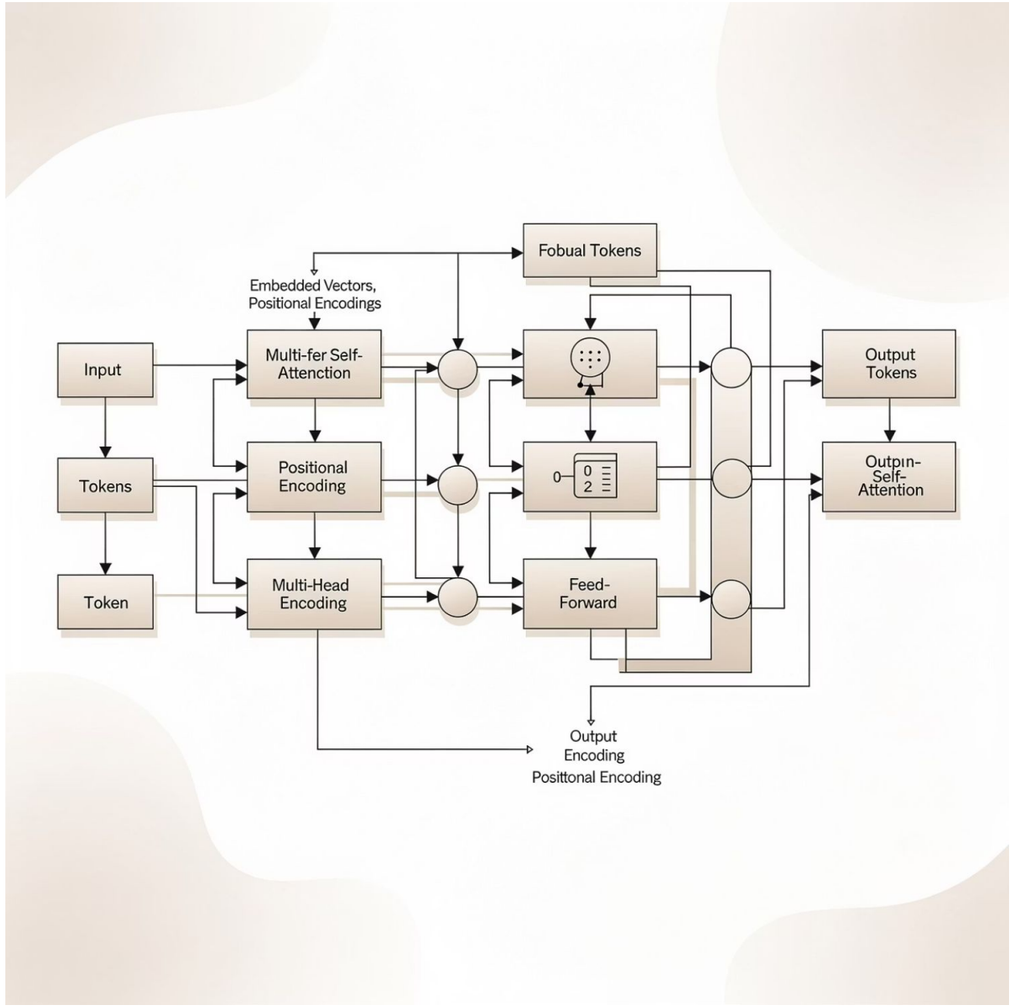
LLM-Based

Comprende il contesto completo, identifica il riferimento temporale, può interrogare il database per verificare lo stato della pratica specifica e fornire una risposta personalizzata e accurata.

Large Language Models: Fondamenti

I Large Language Models rappresentano una delle innovazioni più significative nell'intelligenza artificiale moderna. Addestrati su enormi quantità di testo, questi modelli apprendono pattern linguistici complessi, conoscenza enciclopedica e persino capacità di ragionamento emergenti.

La loro potenza deriva dall'architettura Transformer, introdotta nel paper "**Attention Is All You Need**" del 2017, che ha rivoluzionato il modo in cui le macchine elaborano il linguaggio naturale.



L'Architettura

Attention Mechanism

Permette al modello di "prestare attenzione" alle parti rilevanti dell'input, pesando dinamicamente l'importanza di ogni elemento in relazione agli altri.

Parallelizzazione

A differenza delle reti ricorrenti, i Transformer elaborano sequenze intere simultaneamente, accelerando drasticamente il training.

Scalabilità

L'architettura scala efficacemente con l'aumento di parametri e dati, portando a miglioramenti continui nelle performance.

Varianti Architetturali

Decoder-Only

Esempi: GPT, Llama, Mistral

Specializzati nella generazione di testo. Predicono il token successivo basandosi sul contesto precedente. Ideali per chatbot, scrittura creativa e completamento di codice.

Encoder-Only

Esempi: BERT, RoBERTa

Ottimizzati per comprendere e classificare testo. Eccellono in task come sentiment analysis, question answering e named entity recognition.

Encoder-Decoder

Esempi: T5, BART

Combinano comprensione e generazione. Perfetti per traduzione automatica, summarization e trasformazioni complesse di testo.

Le Principali Famiglie di LLM

OpenAI

GPT-5.2: Il modello più avanzato, con ragionamento superiore e contesto esteso.

Anthropic

Claude 4.5: Famiglia con varianti Opus (massima intelligenza), Sonnet (bilanciato) e Haiku (veloce). Enfasi su sicurezza e allineamento etico.

Meta

Llama 3.2: Modelli completamente open source, disponibili per download e esecuzione locale. Ottima base per fine-tuning personalizzato.

Mistral AI

Mixtral: Architettura mixture-of-experts eccezionalmente efficiente. Prestazioni paragonabili a modelli molto più grandi.

Google

Gemini 3: Modello multimodale nativo, integrato profondamente con Google Cloud. Eccelle nell'elaborazione di immagini e video.

AWS

Nova: Modelli micro, lite e PRO. Disponibili solo su Amazon Bedrock

xAI

Grok 4: Modelli Fast, Expert e Thinking.

Accesso ai modelli

Self-hosting

Open source (Llama, Mistral, Qwen)

✓ Controllo totale dei dati

✓ Personalizzazione estrema

✗ Gestione infrastruttura complessa

✗ Costi operativi alti

API Ufficiali

OpenAI, Anthropic, Google

✓ Accesso ai modelli più recenti

✓ Zero operazioni infrastruttura

✗ Dati fuori dall'azienda

✗ Vendor lock-in

Broker / Multi-provider

AWS Bedrock, Azure AI

✓ Accesso a più modelli

✓ Governance centralizzata

✓ Portabilità tra provider

✗ Meno controllo sul modello specifico

Capacità dei Large Language Models

Comprensione Linguistica Avanzata

Analisi semantica profonda, riconoscimento di sfumature, ironia e contesto culturale. Gestione naturale di ambiguità e riferimenti impliciti.

Ragionamento Logico

Capacità di inferenza, problem solving e ragionamento step-by-step. Limitato ma in continuo miglioramento con tecniche come Chain-of-Thought.

Adattamento Rapido

Prompt engineering permette di specializzare il modello su nuovi domini senza riaddestramento, con esempi few-shot.

Generazione Coerente

Produzione di testo fluido, contestualmente appropriato e stilisticamente coerente su argomenti complessi e articolati.

Supporto Multilingua

Funzionamento nativo in decine di lingue con traduzione automatica di alta qualità e comprensione cross-linguistica.

Function Calling

Integrazione con strumenti esterni: API, database, calcolatrici. Il modello decide quando e come usare gli strumenti disponibili.

Limitazioni e sfide dei LLM

Allucinazioni

I LLM possono generare informazioni plausibili ma completamente false con sicurezza apparente. Questo è uno dei problemi più critici per applicazioni mission-critical.

Conoscenza Statica

I dati di training hanno una data di cutoff. GPT-3.5, ad esempio, non conosce eventi posteriori al suo training. Non può accedere a informazioni in tempo reale senza integrazioni esterne.

Costi Computazionali

L'esecuzione locale richiede GPU potenti (8GB+ VRAM per modelli medio-piccoli). I modelli cloud hanno costi pay-per-token che possono accumularsi rapidamente.

Privacy e Sicurezza

I modelli cloud processano dati su server esterni, sollevando preoccupazioni GDPR. I dati di training potrebbero contenere informazioni sensibili memorizzate involontariamente.

Latenza Variabile

I tempi di risposta dipendono dal carico del sistema, dalla lunghezza dell'input/output e dalla complessità della richiesta. Imprevedibilità problematica per servizi real-time.

Limiti di Contesto

Ogni modello ha un limite massimo di token processabili simultaneamente (4K, 8K, 128K). Conversazioni molto lunghe o documenti estesi richiedono strategie di chunking.

Strategie di Mitigazione

RAG contro le Allucinazioni

Retrieval-Augmented Generation àncora le risposte a documenti verificati, fornendo fonti citabili e riducendo drasticamente le informazioni inventate.

Aggiornamento Continuo

Combinazione di RAG per conoscenza aggiornata e fine-tuning periodico su dati recenti per mantenere il modello al passo con cambiamenti normativi e procedurali.

Modelli Locali Efficienti

Ollama e LM Studio permettono esecuzione di modelli quantizzati che richiedono meno risorse, con deployment su hardware consumer-grade.

Deployment On-Premise

Per dati sensibili della PA, soluzioni self-hosted garantiscono conformità GDPR completa con controllo totale sui dati.

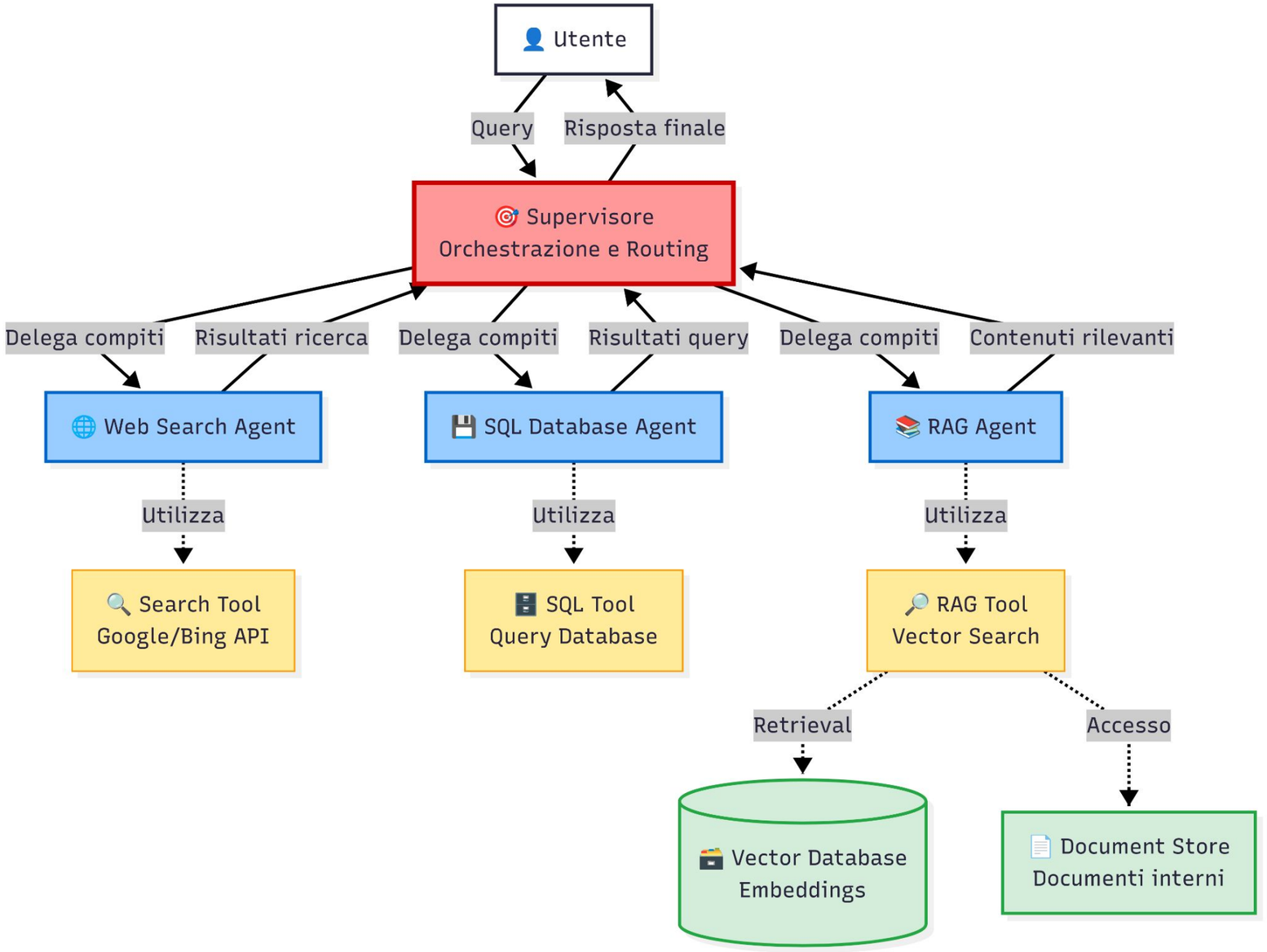
Ottimizzazioni Performance

Caching intelligente di risposte comuni, batching di richieste, uso di modelli più piccoli per task semplici e streaming per ridurre latenza percepita.

Gestione Contesto

Summarization automatica di conversazioni lunghe, architetture gerarchiche con retrieval selettivo e chunking intelligente di documenti estesi.

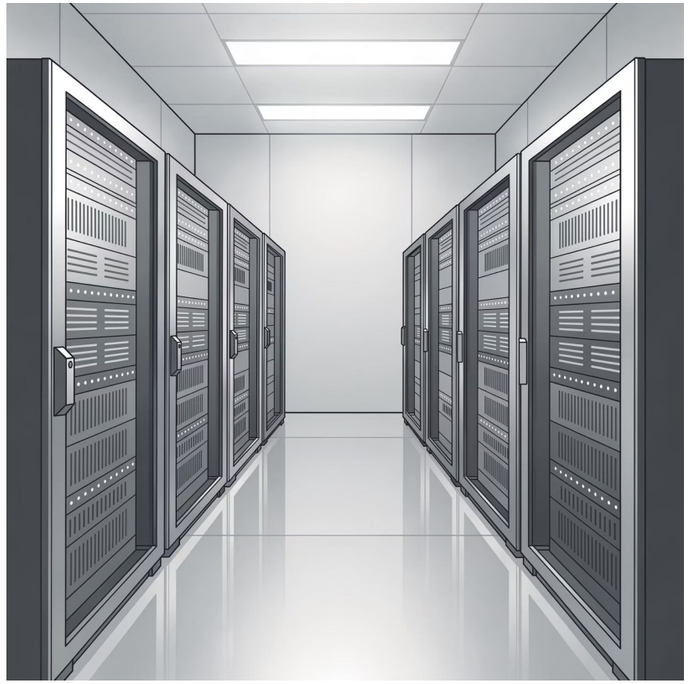
RAG e architettura multi agente



LLM Locali vs Cloud

- La scelta tra deployment locale e cloud è una delle decisioni architetturali più importanti.
- Non esiste una risposta universale: la soluzione ottimale dipende da requisiti specifici di privacy, budget, competenze tecniche e requisiti prestazionali.
- Per la Pubblica Amministrazione, questa scelta ha implicazioni particolarmente significative in termini di conformità normativa e protezione dei dati dei cittadini.

LLM Locali: controllo



Quando Scegliere il Locale

- Dati sensibili o riservati (dati sanitari, giudiziari)
- Requisiti GDPR stringenti
- Necessità di controllo completo
- Budget prevedibile a lungo termine
- Volumi di utilizzo molto elevati

Privacy Assoluta
I dati non lasciano mai l'infrastruttura controllata. Nessun rischio di leak attraverso provider terzi.

Costi Fissi
Investimento iniziale in hardware, poi costi operativi prevedibili. Nessuna sorpresa da pay-per-token.

Personalizzazione
Fine-tuning completo senza limitazioni. Controllo totale su versioni e configurazioni.

Indipendenza
Nessuna dipendenza da fornitori esterni, rate limits o modifiche unilaterali di servizio.

LLM Locali: sfide

1

Requisiti Hardware Significativi

GPU moderne con almeno 8GB VRAM per modelli da 7B parametri. Modelli più grandi richiedono 24GB+ o configurazioni multi-GPU. Costi hardware iniziali non trascurabili.

2

Complessità Setup

Configurazione ambiente CUDA, gestione dipendenze Python, ottimizzazione performance. Curva di apprendimento ripida per team senza esperienza ML.

3

Performance Inferiori

Modelli locali sono generalmente più piccoli e meno capaci dei flagship cloud.

La latenza può essere maggiore senza ottimizzazioni dedicate.

4

Responsabilità Operativa

Aggiornamenti di sicurezza, monitoring, backup, disaster recovery interamente a carico del team interno.

Richiede competenze DevOps/MLOps.

Strumenti per LLM Locali

Ollama

✓

Soluzione più semplice per eseguire LLM localmente. Download modelli con un comando, API compatibile OpenAI. Ideale per prototipazione rapida e sviluppo.

LM Studio

Interfaccia grafica user-friendly per scaricare, testare e servire modelli. Ottimo per esplorare diversi modelli senza codice.

vLLM

Server di inferenza ottimizzato per throughput elevato. Production-ready per deployment scalabile.

Text Generation Inference

Soluzione Hugging Face per serving modelli in produzione. Ottimizzazioni avanzate, streaming, batching dinamico. Containerizzato e cloud-native.

Modelli Consigliati per Deployment Locale

Llama 3 8B

Parametri: 8 miliardi · **VRAM:** ~6GB (quantizzato)

Eccellente bilanciamento qualità/risorse. Ottimo per italiano. Ideale come punto di partenza per la maggior parte dei casi d'uso.

✓ Llama 3.2 - 3B

Mistral 7B

Parametri: 7 miliardi · **VRAM:** ~5GB (quantizzato)

Estremamente efficiente grazie a sliding window attention. Prestazioni superiori alla categoria di dimensione. Ottimo supporto multilingua.

Phi-3 Mini

Parametri: 3.8 miliardi · **VRAM:** ~3GB (quantizzato)

Microsoft, sorprendentemente capace nonostante dimensioni ridotte. Perfetto per hardware limitato o deployment edge.

Qwen 2

Parametri: varie dimensioni · **VRAM:** variabile

Alibaba, eccellente comprensione multilingua. Particolarmente forte su lingue asiatiche ma ottimo anche per italiano.

LLM Cloud

Prestazioni di Punta

Accesso immediato ai modelli più avanzati (GPT-5.2, Claude Opus) con inferenza ottimizzata da infrastrutture dedicate.

Scalabilità Automatica

Il sistema gestisce automaticamente picchi di carico senza configurazione. Paghi solo per l'uso effettivo.

Time-to-Market Rapido

Setup in minuti con semplici API calls. Nessuna configurazione hardware o gestione infrastruttura.

Aggiornamenti Continui

Accesso automatico a nuovi modelli e features senza migrazione. Provider gestiscono ottimizzazioni e sicurezza.

LLM Cloud: considerazioni

Costi Variabili

Un progetto di successo con alto volume può diventare molto costoso.

Privacy dei Dati

I dati transitano su server esterni.

OpenAI e Anthropic *affermano di non usare dati API per training*.

Dipendenza da Terzi

Richiede connessione internet stabile. Il servizio dipende dalla disponibilità del provider. Rate limits possono limitare scalabilità in momenti critici.

Compliance Complessa

GDPR richiede Data Processing Agreements.

Data residency europea non sempre garantita.

Audit di sicurezza più complessi con terze parti.

Provider Cloud Principali

OpenAI API

Accesso diretto a GPT-5.x. API semplice e ben documentata. Pricing trasparente. Non garantisce data residency europea.

Anthropic

Claude 4.5 via API diretta o AWS Bedrock. Enfasi su sicurezza e allineamento. Contesto esteso fino a 200K token.

Amazon Bedrock

Accesso unificato a Claude, Llama, Titan, Mistral. Integrazione nativa con ecosistema AWS. Data residency controllabile*.

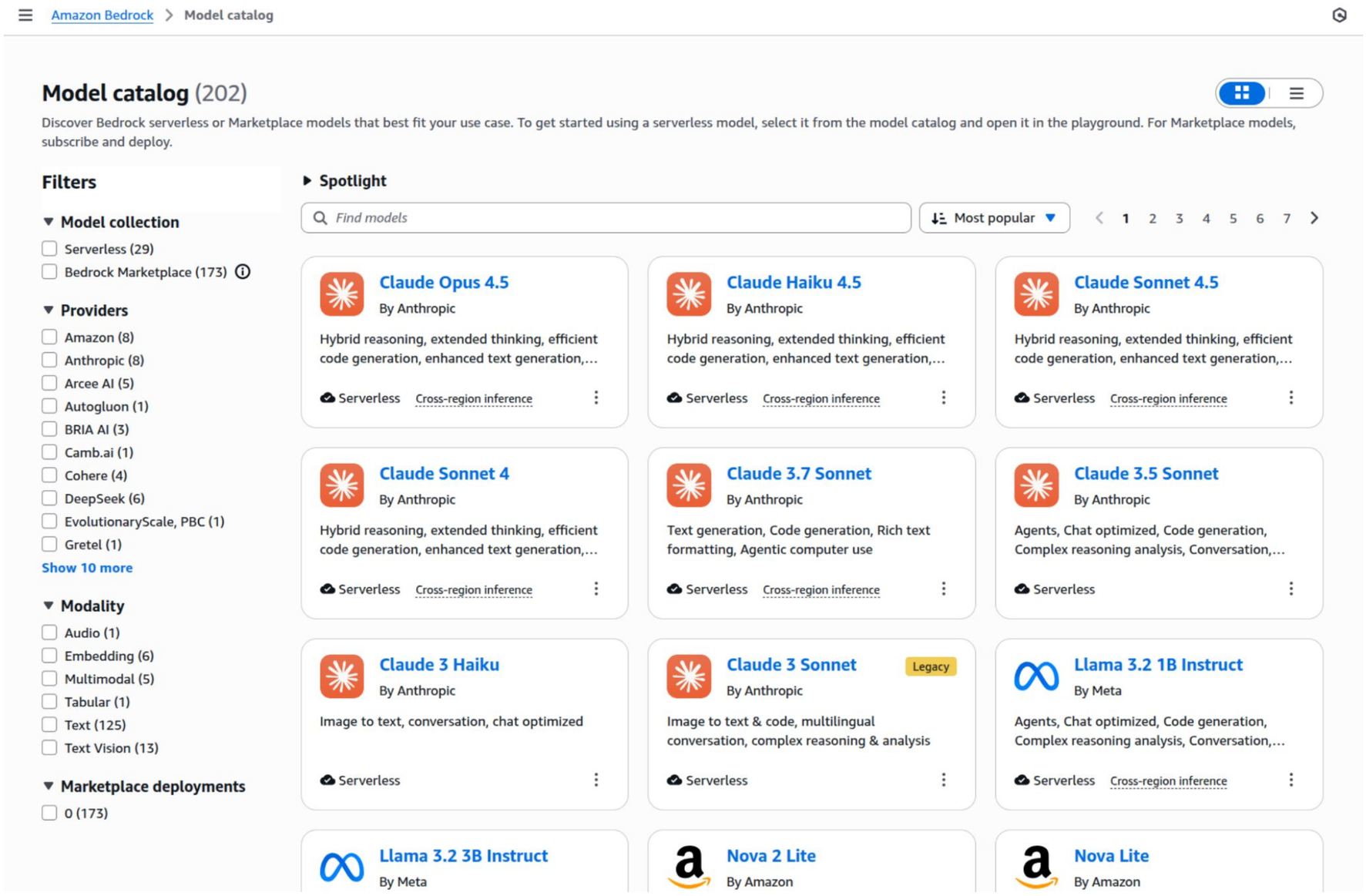
Google Cloud Vertex AI

Gemini Pro. Forte integrazione con Google Cloud. Vertex AI Search per RAG gestito. Compliance certificazioni ISO.

Azure OpenAI

GPT-5.x tramite Microsoft Azure. Data residency europea garantita. Integrazione Azure Active Directory. SLA enterprise-grade.

Amazon Bedrock



Strategia Ibrida

Approccio Raccomandato

La soluzione ottimale non è "o locale o cloud", ma una strategia ibrida che sfrutta i vantaggi di entrambi approcci in base al contesto specifico.

Sviluppo e Prototipazione

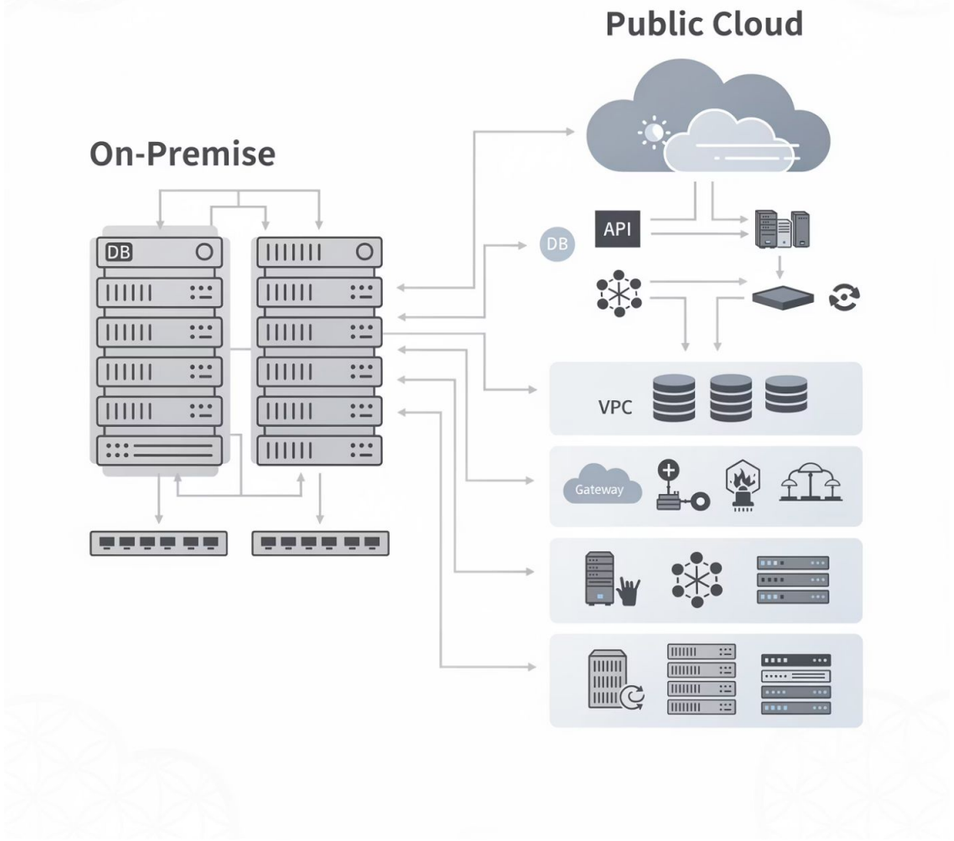
Ollama locale per iterazione rapida, privacy totale durante sviluppo, costi zero

Valutazione Caso per Caso

Dati sensibili → locale/on-premise; Dati pubblici → cloud per prestazioni

Fallback e Ridondanza

Cloud come backup quando locale non disponibile, bilanciamento carico in picchi



Casi d'Uso nella Pubblica Amministrazione

La Pubblica Amministrazione italiana gestisce milioni di interazioni con cittadini ogni anno. I chatbot intelligenti possono trasformare radicalmente questa esperienza, offrendo supporto immediato, accurato e disponibile 24/7. I casi d'uso spaziano dall'assistenza al cittadino all'automazione di processi interni, con benefici misurabili in termini di efficienza, soddisfazione utente e riduzione costi operativi.

Assistenza ai cittadini: primo punto di contatto

FAQ Automatizzate

Risposte immediate a domande frequenti su servizi comunali, documenti necessari, orari uffici. Il chatbot gestisce il 60-70% delle richieste comuni senza intervento umano.

Supporto Pre-Compilazione

Guida step-by-step nella compilazione di moduli complessi. Verifica dati in tempo reale, suggerisce correzioni, riduce errori e tempi di elaborazione.

Informazioni su Pratiche

Verifica stato di pratiche in corso, prossime scadenze, documenti mancanti. Integrazione con sistemi gestionali per informazioni real-time personalizzate.

Orientamento Servizi

Indirizzamento intelligente verso il servizio corretto. Comprende richieste generiche e suggerisce il percorso amministrativo appropriato.

Vantaggi misurabili per cittadini e PA

24/7

Disponibilità continua

Nessun vincolo di orari d'ufficio. I cittadini ottengono risposte quando ne hanno bisogno.

<60s

Tempo di risposta

Risposte immediate vs ore/giorni di attesa per email o giorni per appuntamento di persona.

50%

Riduzione carico call center

Il chatbot gestisce autonomamente la maggioranza delle richieste semplici, liberando operatori per casi complessi.

40%

Riduzione Costi Operativi

Automazione di interazioni ripetitive con ROI positivo in 12-18 mesi.

La chiave è progettare l'integrazione chatbot-operatore in modo fluido: escalation automatica per casi complessi, handoff contestuale con storico conversazione completo.

Supporto Interno: efficienza per i dipendenti

- **Help Desk Interno**

  Supporto tecnico IT first-level automatizzato: password reset, problemi comuni software, troubleshooting guidato

- **Knowledge Base Aziendale**

  Accesso conversazionale a procedure interne, policy, regolamenti. Ricerca semantica vs keyword matching

- **Onboarding Dipendenti**

  Guida interattiva per nuovi assunti: procedure, strumenti, cultura organizzativa. Riduce tempo formazione

- **Training Continuo**

  Supporto formativo just-in-time su nuove normative, aggiornamenti procedurali, best practices

Automazione Processi Documentali

Classificazione Automatica

Categorizzazione intelligente documenti in arrivo: tipologia, priorità, reparto competente.

Estrazione Dati

Parsing automatico di moduli, email, PDF. Estrazione campi strutturati da testo libero. Popolamento database automatico.

Routing Intelligente

Instradamento richieste al reparto/persona corretta basato su contenuto. Riduce errori di smistamento e tempi di lavorazione.

Report Automatici

Generazione sintesi da dataset complessi. Evidenziazione trend, anomalie, insight. Formato narrativo human-friendly.

Standard di qualità per il servizio

Multilingua inclusivo

Supporto italiano standard e lingue minoritarie riconosciute.

Trasparenza algoritmica

Indicazione chiara quando si interagisce con AI vs operatore umano. Spiegabilità decisioni. Possibilità di richiedere revisione umana.

Accountability e Audit

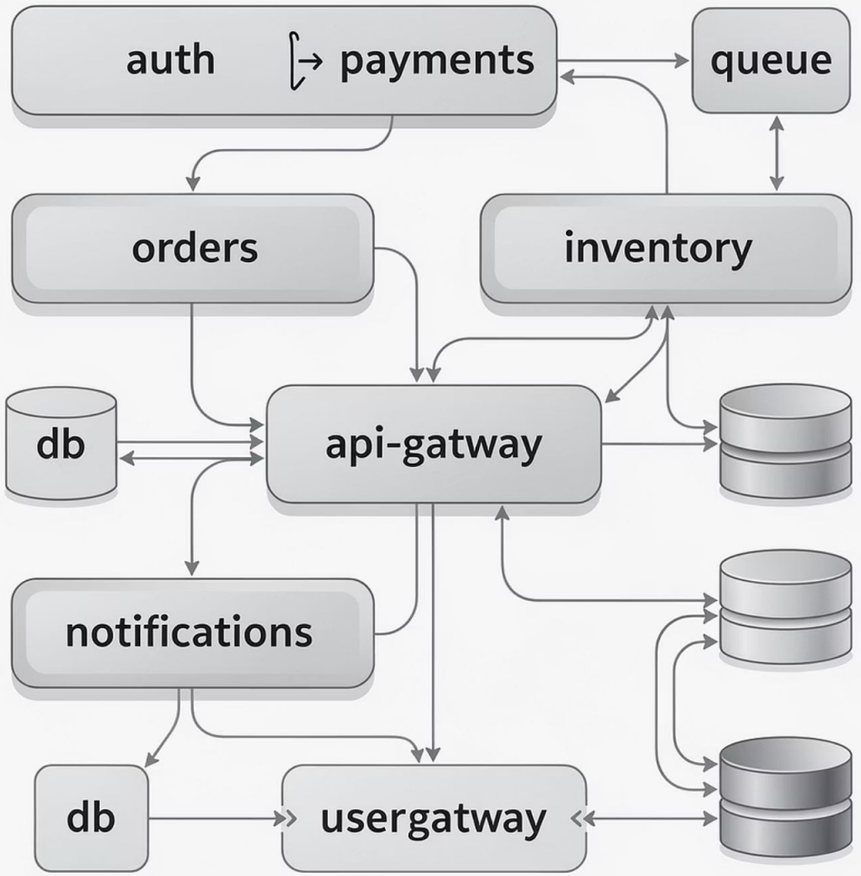
Logging completo conversazioni con timestamp. Tracciabilità decisioni. Possibilità audit su richiesta cittadino o autorità.

Sicurezza e Privacy

Crittografia end-to-end. Autenticazione forte per dati sensibili. Minimizzazione raccolta dati. Retention policy conforme.

Compliance Normativa

GDPR compliance completa. Rispetto Codice Amministrazione Digitale. Data residency italiana/europea.



Architettura Moderna di un Chatbot Intelligente

Un chatbot moderno non è un monolite, ma un ecosistema di componenti che collaborano. L'architettura deve bilanciare semplicità implementativa, scalabilità futura, manutenibilità e integrazione con sistemi esistenti. Una progettazione solida è fondamentale per evitare debito tecnico e garantire evoluzione nel tempo.

Componenti Architetturali Fondamentali

Interfaccia Utente

Layer di presentazione multi-canale: web, mobile, Telegram, WhatsApp, interfacce vocali. Adattamento automatico a device e contesto.

Orchestratore Conversazionale

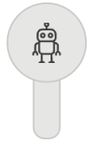
Cervello del sistema. Gestisce stato conversazione, memoria, routing messaggi, decisioni strategiche su come processare ogni input.

LLM Core

Motore di generazione risposte. Può essere locale o cloud. Interfaccia astratta per supportare switch tra provider.

RAG System

Retrieval-Augmented Generation. Vector database con la documentazione aziendale, retrieval semantico, injection contesto nei prompt.



Agent Framework

Esecuzione azioni: query database, creazione ticket, invio email, chiamate API. LLM decide quando e come usare ogni tool.

Sistemi Esterni

Integrazione con CRM, Help Desk, database legacy, API terze parti. Connector layer per disaccoppiamento e manutenibilità.

Flusso di Elaborazione di una Richiesta

1. Ricezione e Preprocessing

Input utente arriva via canale (web/Telegram/etc). Normalizzazione testo, pulizia, estrazione metadati (utente ID, sessione, timestamp, lingua).

2. Gestione Memoria

Recupero storia conversazione da database. Caricamento contesto utente. Aggiornamento working memory con nuovo messaggio.

3. Decisione Strategia

L'orchestratore classifica richiesta: RAG (serve conoscenza documentale), Agent (serve azione esterna), o LLM diretto (conversazione generica).

4. Elaborazione Specifica

Se RAG: embedding query, search vector DB, retrieve documenti rilevanti.

Se Agent: planning task, selezione tools, esecuzione azioni.

Se diretto: costruzione prompt con contesto.

5. Generazione Risposta

Chiamata LLM con prompt costruito. Post-processing: validazione formato, sanitization, applicazione policy organizzative, guardrails.

6. Integrazione e Logging

Se necessario: creazione ticket Help Desk, update CRM, query database per arricchimento. Logging completo per audit e analytics.

7. Risposta Utente

Formattazione per canale specifico (markdown, HTML, plain text). Invio tramite interfaccia appropriata. Update memoria conversazione.

Pattern Architetturali Chiave

RAG Pattern

Quando: Domande su documenti, policy, knowledge base

Flow: Query → Embedding → Vector Search → Context Injection → LLM → Risposta + Citations

Agent Pattern

Quando: Richieste azioni su sistemi esterni

Flow: Intent Analysis → Planning → Tool Selection → Execution → Result Synthesis → Response

Memory Pattern

Quando: Conversazioni multi-turno con contesto

Flow: History Retrieval → Summarization → Context Window Management → LLM with Memory

Multi-Agent Pattern

Quando: Task complessi con specializzazione

Flow: Task Decomposition → Agent Assignment → Parallel Execution → Result Aggregation

Obiettivi per Sicilia Digitale

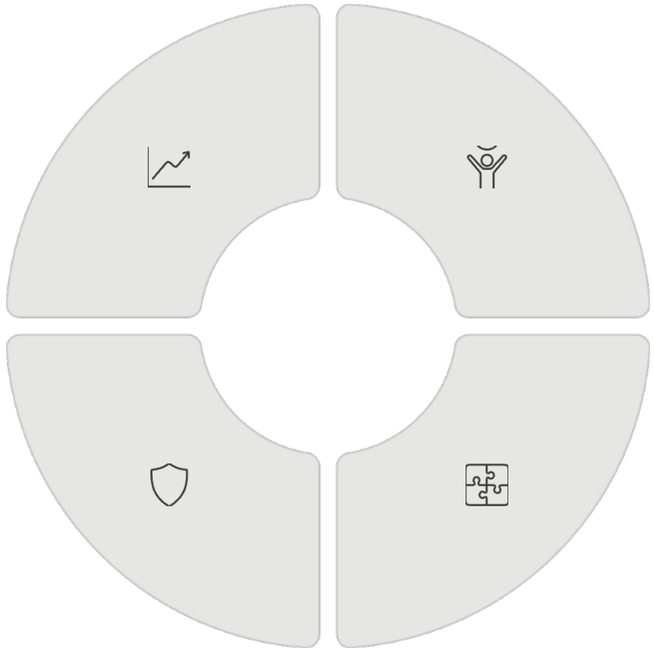
L'implementazione di chatbot intelligenti per Sicilia Digitale non è solo un progetto tecnologico, ma una trasformazione del modo in cui l'organizzazione serve cittadini e supporta i propri dipendenti. Gli obiettivi devono essere specifici, misurabili e allineati con la missione di digitalizzazione della PA siciliana.

Riduzione Carico

Automatizzare 40-50% richieste comuni

Esperienza Utente

Disponibilità 24/7, risposte immediate, supporto multilingua



Compliance

GDPR, audit trail, sicurezza dati sensibili garantita

Integrazione

Connessione fluida con Help Desk e database esistenti

Metriche di successo: numeri veri?

70%

Tasso di Risoluzione Autonoma

Percentuale richieste risolte dal chatbot senza escalation a operatore umano.

85%

Soddisfazione Utente

Survey post-interazione con scala 1-5. Target: 4.2+ (85%+ soddisfatti). Monitoraggio sentiment analysis conversazioni.

40%

Riduzione Ticket Help Desk

Diminuzione richieste manuali per FAQ comuni. Liberazione capacity per problemi complessi che richiedono expertise umana.

80%

Accuracy Risposte

Percentuale risposte corrette verificate tramite audit campionario. Importante per fiducia e adoption utenti.

Roadmap di Implementazione

Fase 1: Prototipo

Durata: Durante questo corso

Chatbot base con LLM locale (Ollama). RAG su documenti FAQ.

Mock o reale integrazione con Help Desk/Database. Proof of concept funzionante.

Fase 2: Pilot

Durata: 3-6 mesi

Deploy interno (rete locale o intranet). Integrazione reale Help Desk.

User testing con gruppo limitato. Raccolta feedback.

Iterazione basata su metriche.

Fase 3: Produzione

Durata: 6-12 mesi

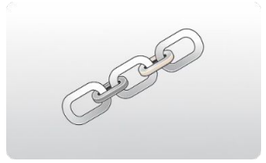
Rollout completo multi-canale (Telegram? WhatsApp? WEB? Mobile?).

Integrazione ecosistema completo.

Fine-tuning su casi d'uso specifici.

Scaling su cloud se necessario.

Tecnologie e Framework



LangChain

Framework principale per orchestrazione. Gestione prompt templates, chains, memory, agent. Astrazione multi-LLM. Ecosistema ricco di integrazioni.

LangSmith

Piattaforma monitoring e debugging. Tracing completo conversazioni. Ottimizzazione prompt iterativa. Analytics performance e costi.

Chroma / Qdrant

Vector database per RAG. Chroma: semplice, Python-native, ottimo per prototipazione. Qdrant: production-ready, features avanzate, scalabile.

Python 3.12+

Linguaggio principale dell'ecosistema LLM. Jupyter Notebook per prototipazione. pip per dependency management.

Ollama

Execution engine per LLM locali. Download modelli con un comando. API compatibile OpenAI. Ideale per sviluppo privacy-first.

Docker

Containerizzazione per deployment riproducibile. Docker Compose per orchestrazione multi-container. Semplifica staging e produzione.

Prossimi Passi...

Questo modulo teorico ha gettato le fondamenta. Abbiamo ora una comprensione solida dell'evoluzione dei chatbot, delle architetture LLM, delle scelte implementative e dei casi d'uso specifici.

Setup Ambiente Pratico

Installazione Ollama e Open WebUI con Docker.

Configurazione e download dei modelli in locale.

Verifica setup e primo test.

Primi Esperimenti LLM

Interazione diretta con Llama locale.

Prompt engineering basics.

Comprensione temperature e parametri.

Introduzione LangChain

Chains, Prompt Templates, Memory.

Costruzione primo chatbot conversazionale semplice.

Implementazione RAG

Ingestion documenti, embedding, vector search.

Chatbot che risponde basandosi su knowledge base.

Agent e Tools

Creazione agent che interagisce con API/database.

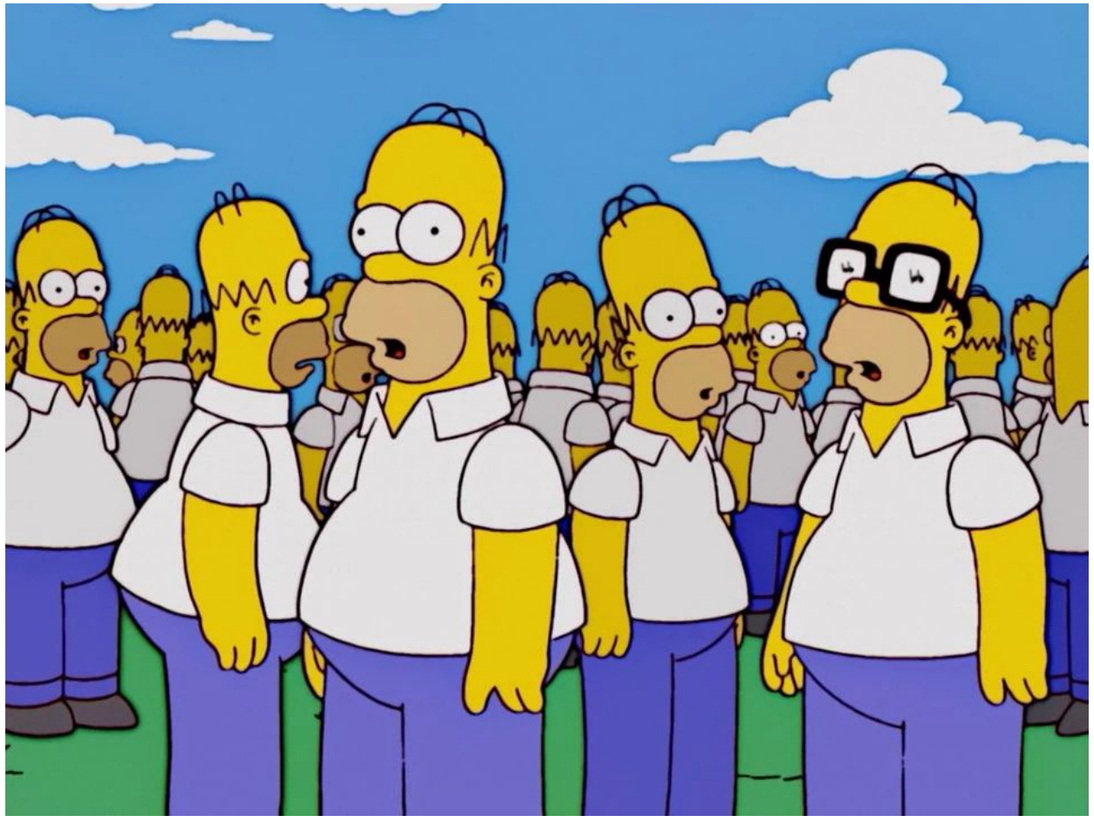
Function calling ed MCP.

Orchestrazione task complessi.

Amazon Bedrock & Bedrock AgentCore

Introduzione ad Amazon Bedrock.

Panoramica delle funzionalità di Bedrock AgentCore



git clone https://github.com/giovanniconte-dt/sicilia-digitale.git