ANALISI FUNZIONALE

Framework per la Realizzazione di
SITI WEB AGENTICI

Versione 1.0
Gennaio 2026

Documento tecnico ad uso interno

Indice

1. Introduzione e Obiettivi

1.1 Contesto

Il presente documento definisce l'analisi funzionale per la realizzazione di un framework modulare destinato alla creazione di siti web agentici. Un sito agentico rappresenta l'evoluzione del paradigma web tradizionale: non più un insieme di pagine statiche o dinamiche che l'utente deve navigare attivamente, ma un sistema intelligente capace di comprendere le esigenze dell'utente, adattare dinamicamente i contenuti e compiere azioni autonome per suo conto.

Il framework si propone come soluzione architetturale generica, indipendente dal settore di applicazione, che consenta ai team di sviluppo di implementare rapidamente siti web con capacità agentiche complete.

1.2 Obiettivi del Framework

- Fornire un'architettura modulare e estensibile per la creazione di siti agentici
- Astrarre la complessità dell'integrazione con modelli LLM e sistemi di Retrieval-Augmented Generation (RAG)
- Offrire componenti riutilizzabili per le funzionalità agentiche più comuni
- Garantire sicurezza, scalabilità e manutenibilità delle soluzioni implementate
- Supportare l'integrazione con sistemi esterni (CRM, ERP, calendari, ecc.)
- Predisporre il supporto per gli standard emergenti del web agentico (llms.txt, MCP, A2A)

1.3 Scope del Documento

Questo documento copre i requisiti funzionali e non funzionali del framework, l'architettura ad alto livello e la descrizione dei componenti principali. Non include dettagli implementativi specifici, scelte tecnologiche vincolanti o stime di effort, che saranno oggetto di documenti successivi.

2. Requisiti del Framework

2.1 Requisiti Funzionali

RF-01: Gestione Conversazionale

Il framework deve fornire un motore conversazionale che permetta agli utenti di interagire con il sito attraverso linguaggio naturale. Il sistema deve mantenere il contesto della conversazione, gestire ambiguità e guidare l'utente verso il completamento dei propri obiettivi.

RF-02: Personalizzazione Dinamica dei Contenuti

Il framework deve consentire la generazione e l'adattamento dei contenuti in tempo reale basandosi su: profilo utente, storico interazioni, contesto della sessione corrente, dati provenienti da sistemi integrati. La personalizzazione deve operare sia a livello di contenuto testuale che di struttura della pagina.

RF-03: Sistema RAG (Retrieval-Augmented Generation)

Il framework deve integrare un sistema RAG che consenta di ancorare le risposte dell'AI a fonti di conoscenza aziendali verificate. Deve supportare l'indicizzazione di documenti in vari formati (PDF, DOCX, HTML, database) e fornire meccanismi per l'aggiornamento incrementale della knowledge base.

RF-04: Esecuzione di Azioni Autonome

Il framework deve permettere all'agente di eseguire azioni concrete per conto dell'utente, previo consenso esplicito quando richiesto. Le azioni includono: invio email, creazione appuntamenti, generazione documenti, interrogazione di API esterne, aggiornamento di record in sistemi integrati.

RF-05: Orchestrazione Workflow

Il framework deve fornire un motore di orchestrazione per definire e gestire flussi di lavoro complessi. I workflow devono supportare: condizioni, branching, gestione errori, timeout, retry automatici, e integrazione con servizi esterni.

RF-06: Integrazione con Sistemi Esterni

Il framework deve offrire un layer di astrazione per l'integrazione con sistemi esterni comuni: CRM (Salesforce, HubSpot), ERP, calendari (Google Calendar, Outlook), sistemi di ticketing, database aziendali. Deve supportare pattern di integrazione sia sincroni che asincroni.

RF-07: Motore Decisionale

Il framework deve includere un motore decisionale configurabile che consenta di definire regole di business per: routing delle richieste, prioritizzazione dei contenuti, qualificazione lead, raccomandazioni prodotto. Le regole devono poter combinare logica deterministica e inferenza AI.

RF-08: Gestione Multi-canale

Il framework deve supportare la delivery dell'esperienza agentica su canali multipli: web (SPA/SSR), widget embeddabili, API per integrazioni terze. Il comportamento dell'agente deve essere coerente tra i canali, con adattamenti specifici per le caratteristiche di ciascuno.

RF-09: Analytics e Tracciamento

Il framework deve fornire strumenti per il tracciamento delle interazioni utente-agente, inclusi: metriche di conversazione (durata, completamento task), funnel di conversione, analisi delle query non risolte, feedback utente. I dati devono essere esportabili verso sistemi di analytics esterni.

RF-10: Supporto Standard Emergenti

Il framework deve predisporre il supporto per gli standard emergenti del web agentico: generazione automatica di file llms.txt, esposizione di endpoint MCP (Model Context Protocol), predisposizione per protocolli A2A (Agent-to-Agent).

2.2 Requisiti Non Funzionali

ID	Categoria	Descrizione
RNF-01	Performance	Tempo di risposta dell'agente < 3 secondi per il 95% delle richieste. Supporto minimo di 100 sessioni concorrenti per istanza.
RNF-02	Scalabilità	Architettura orizzontalmente scalabile. Possibilità di scaling automatico basato sul carico. Separazione tra componenti stateless e stateful.
RNF-03	Sicurezza	Autenticazione e autorizzazione integrate. Protezione contro prompt injection. Audit log delle azioni agentiche. Conformità GDPR per trattamento dati.
RNF-04	Affidabilità	Disponibilità target 99.5%. Gestione graceful degradation in caso di fallimento servizi AI. Meccanismi di fallback per funzionalità critiche.
RNF-05	Manutenibilità	Architettura modulare con componenti sostituibili. Documentazione API completa. Test coverage minimo 80% per componenti core.
RNF-06	Estensibilità	Sistema di plugin per estensioni custom. Hook points per personalizzazione comportamenti. Supporto per provider LLM multipli.
RNF-07	Osservabilità	Logging strutturato. Metriche esportabili (Prometheus/OpenTelemetry). Distributed tracing per debug flussi complessi.
RNF-08	Portabilità	Deployment containerizzato (Docker/Kubernetes). Indipendenza da cloud provider specifici. Supporto per deployment on-premise.

3. Architettura del Framework

3.1 Panoramica Architetturale

L'architettura del framework segue un modello a layer con separazione delle responsabilità. Ogni layer comunica esclusivamente con i layer adiacenti attraverso interfacce ben definite, garantendo modularità e sostituibilità dei componenti.

Il design architetturale si basa sui seguenti principi:

- Separazione tra logica di presentazione, business logic e accesso ai dati
- Componenti stateless dove possibile per facilitare lo scaling
- Event-driven architecture per operazioni asincrone e disaccoppiamento
- Plugin architecture per estensibilità senza modifiche al core
- Fail-safe design con graceful degradation

3.2 Layer Architetturali

L'architettura è organizzata in sei layer principali, dal più vicino all'utente al più profondo:

Layer	Responsabilità	Componenti Principali
Presentation	Interfaccia utente e rendering. Gestione input/output conversazionale. Adattamento multi-canale.	UI Components, Chat Widget, Content Renderer, Channel Adapters
Agent	Orchestrazione comportamento agentico. Gestione contesto e stato conversazione. Routing verso capabilities appropriate.	Agent Orchestrator, Context Manager, Conversation Handler, Intent Router
Capabilities	Funzionalità atomiche eseguibili dall'agente. Interfaccia verso servizi esterni. Logica di business specifica.	Action Executors, Content Generators, Tool Adapters, Business Rules
Intelligence	Comprensione linguaggio naturale. Generazione risposte. Recupero informazioni da knowledge base.	LLM Gateway, RAG Engine, Embedding Service, Prompt Manager
Integration	Connessione a sistemi esterni. Normalizzazione dati. Gestione autenticazione verso terze parti.	Connector Registry, API Clients, Data Transformers, Auth Manager
Infrastructure	Servizi trasversali. Persistenza. Caching. Event bus. Sicurezza.	Event Bus, Cache Manager, Session Store, Security Layer, Config Manager

3.3 Componenti Core

3.3.1 Agent Orchestrator

Componente centrale che coordina il ciclo di vita delle interazioni utente. Riceve le richieste dal layer di presentazione, determina l'intento dell'utente attraverso il layer Intelligence, seleziona e invoca le capabilities appropriate, e compone la risposta finale.

Responsabilità principali:

- Gestione del ciclo request-response agentico
- Coordinamento tra capabilities multiple per task complessi
- Gestione del contesto conversazionale tra turni di dialogo
- Applicazione delle policy di sicurezza e governance
- Logging e tracciamento delle interazioni per audit

3.3.2 Context Manager

Gestisce lo stato della sessione e il contesto accumulato durante l'interazione. Mantiene la memoria della conversazione, le preferenze dell'utente espresse, e le informazioni raccolte da sistemi integrati.

Il Context Manager opera su tre livelli di contesto:

- Session Context: informazioni relative alla sessione corrente (turni di conversazione, intent rilevati, azioni eseguite)
- User Context: profilo utente persistente, storico interazioni, preferenze salvate
- System Context: configurazione attiva, capabilities disponibili, stato dei sistemi integrati

3.3.3 LLM Gateway

Astrazione per l'accesso ai modelli di linguaggio. Fornisce un'interfaccia unificata indipendente dal provider specifico (OpenAI, Anthropic, modelli open-source). Gestisce retry, rate limiting, fallback tra provider, e caching delle risposte dove appropriato.

Funzionalità chiave:

- Provider abstraction per supporto multi-LLM
- Prompt templating e management
- Token counting e cost tracking
- Response streaming per latenza percepita ridotta
- Fallback chain in caso di indisponibilità provider primario

3.3.4 RAG Engine

Motore di Retrieval-Augmented Generation che collega le capacità generative dell'LLM alle fonti di conoscenza aziendali. Comprende pipeline di indicizzazione documenti, vector store per similarità semantica, e logica di retrieval context-aware.

Pipeline di funzionamento:

- Document Ingestion: parsing multi-formato, chunking intelligente, estrazione metadata

- Embedding Generation: vettorizzazione dei chunk tramite embedding models
- Retrieval: ricerca semantica con ranking e filtering
- Augmentation: composizione del prompt arricchito con contesto recuperato
- Generation: invocazione LLM con prompt augmentato

3.3.5 Action Executor

Componente responsabile dell'esecuzione delle azioni autonome richieste dall'agente. Implementa il pattern Command per incapsulare le operazioni, con supporto per undo dove possibile e logging dettagliato per audit.

Ogni azione è definita da:

- Preconditions: condizioni che devono essere verificate prima dell'esecuzione
- Permission requirements: livello di autorizzazione richiesto
- Execution logic: implementazione effettiva dell'azione
- Rollback logic: procedura di annullamento se disponibile
- Post-conditions: stato atteso dopo l'esecuzione

3.3.6 Workflow Engine

Motore per l'orchestrazione di flussi multi-step. Permette di definire sequenze di operazioni con condizioni, branching, parallelismo e gestione errori. Supporta sia workflow definiti staticamente che workflow generati dinamicamente dall'agente.

Caratteristiche:

- Definizione workflow tramite DSL dichiarativo o visual editor
- Supporto per task sincroni e asincroni
- Gestione stato persistente per workflow long-running
- Retry policy configurabili per step
- Human-in-the-loop per approvazioni su step critici

3.3.7 Decision Engine

Motore per la valutazione di regole di business e decisioni complesse. Combina regole deterministiche (if-then) con inferenza AI per scenari che richiedono flessibilità. Utilizzato per routing, personalizzazione, qualificazione lead e raccomandazioni.

Modalità di decisione supportate:

- Rule-based: regole deterministiche configurate dall'amministratore
- ML-based: modelli di machine learning per pattern recognition
- Hybrid: combinazione di regole e ML con override espliciti
- LLM-assisted: delega all'LLM per decisioni che richiedono comprensione semantica

3.3.8 Connector Registry

Registro centralizzato dei connettori verso sistemi esterni. Fornisce discovery, lifecycle management, e health monitoring dei connettori. Ogni connettore implementa un'interfaccia standard che astrae le specificità del sistema target.

Connettori previsti nel framework base:

- CRM Connector: integrazione con Salesforce, HubSpot, Dynamics
- Calendar Connector: Google Calendar, Outlook, CalDAV
- Email Connector: SMTP, SendGrid, provider transazionali
- Storage Connector: S3, Azure Blob, Google Cloud Storage
- Database Connector: interfaccia generica per database relazionali e documentali

4. Modello dei Dati

4.1 Entità Principali

Il framework definisce le seguenti entità core che costituiscono il modello dati fondamentale:

Session

Rappresenta una sessione di interazione utente-agente. Contiene il riferimento all'utente, il canale di interazione, lo stato corrente, e il timestamp di inizio/fine. Le sessioni possono essere anonime o autenticate.

Conversation

Sequenza ordinata di turni di dialogo all'interno di una sessione. Ogni conversazione ha un obiettivo implicito o esplicito e può risultare in zero o più azioni eseguite.

Turn

Singolo scambio request-response. Include: input utente (testo, azione UI), intent rilevato, contesto utilizzato, risposta generata, azioni eseguite, metriche (latenza, token).

Intent

Rappresentazione dell'intenzione dell'utente derivata dall'analisi del suo input. Include: tipo di intent, confidenza, entità estratte, slot da riempire per completare la richiesta.

Action

Record di un'azione eseguita dall'agente. Include: tipo di azione, parametri, risultato, timestamp, eventuale utente che ha autorizzato. Costituisce l'audit trail delle operazioni agentiche.

Knowledge Item

Unità di conoscenza indicizzata nel sistema RAG. Include: contenuto testuale, embedding vettoriale, metadata (fonte, data, categoria), riferimenti al documento originale.

User Profile

Profilo utente con informazioni persistenti: identificativi, preferenze espresse, storico interazioni aggregate, segmenti di appartenenza, permessi.

5. Flussi di Interazione

5.1 Flusso Conversazionale Standard

Il flusso tipico di un'interazione conversazionale segue questi step:

1. L'utente invia un messaggio attraverso il layer di Presentation
2. L'Agent Orchestrator riceve la richiesta e carica il contesto dal Context Manager
3. Il layer Intelligence analizza l'input per determinare intent ed entità
4. Se necessario, il RAG Engine recupera informazioni rilevanti dalla knowledge base
5. L'Orchestrator seleziona le capabilities appropriate in base all'intent
6. Le capabilities vengono eseguite, eventualmente interagendo con sistemi esterni
7. L'LLM genera la risposta finale integrando i risultati delle capabilities
8. Il Context Manager aggiorna lo stato della sessione
9. La risposta viene inviata all'utente attraverso il layer di Presentation

5.2 Flusso di Esecuzione Azione

Quando l'agente deve eseguire un'azione con effetti sul mondo esterno:

- L'Orchestrator identifica la necessità di un'azione dall'analisi dell'intent
- Il Decision Engine verifica le precondizioni e i permessi
- Se richiesto, viene sollecitata conferma esplicita all'utente
- L'Action Executor prepara e valida i parametri dell'azione
- L'azione viene eseguita attraverso il connettore appropriato
- Il risultato viene loggato e il contesto aggiornato
- L'utente riceve conferma dell'azione completata

5.3 Flusso RAG

Il processo di Retrieval-Augmented Generation:

- La query utente viene preprocessata ed eventualmente riformulata
- L'Embedding Service genera il vettore della query
- Il Vector Store esegue ricerca per similarità
- I risultati vengono filtrati e re-ranked per rilevanza
- I chunk più rilevanti vengono estratti con il loro contesto
- Il Prompt Manager compone il prompt arricchito
- L'LLM genera la risposta basandosi sul contesto fornito
- La risposta viene post-processata (citazioni, formattazione)

6. Interfacce e API

6.1 API Pubbliche

Il framework espone le seguenti categorie di API per l'integrazione:

Conversation API

Endpoint per la gestione delle conversazioni: creazione sessione, invio messaggi, recupero storico, chiusura sessione. Supporta sia modalità request-response che streaming per risposte progressive.

Management API

API per la configurazione e amministrazione: gestione knowledge base, configurazione workflow, definizione regole di business, gestione connettori, monitoring.

Webhook API

Sistema di notifiche push per eventi significativi: nuova conversazione, azione eseguita, errore critico, threshold superato. Configurabile per evento e destinazione.

MCP Endpoint

Implementazione del Model Context Protocol per esporre le capabilities del sito a agenti AI esterni. Permette discovery automatico delle funzionalità e invocazione standardizzata.

6.2 Interfacce Interne

Per garantire modularità, i componenti comunicano attraverso interfacce formalizzate:

- ILLMProvider: interfaccia per provider di modelli linguistici
- IConnector: interfaccia per connettori verso sistemi esterni
- ICapability: interfaccia per definizione di capabilities agentiche
- IDecisionRule: interfaccia per regole del decision engine
- IWorkflowStep: interfaccia per step di workflow custom

7. Considerazioni sulla Sicurezza

7.1 Minacce Identificate

Il framework deve proteggere contro le seguenti categorie di minacce:

- Prompt Injection: tentativi di manipolare il comportamento dell'agente attraverso input malevoli
- Data Exfiltration: estrazione non autorizzata di dati dalla knowledge base o sistemi integrati
- Privilege Escalation: ottenimento di permessi superiori a quelli autorizzati
- Action Abuse: esecuzione di azioni dannose o non autorizzate
- Denial of Service: sovraccarico del sistema attraverso richieste eccessive

7.2 Contromisure Architetturali

Input Sanitization Layer

Tutti gli input utente passano attraverso un layer di sanitizzazione che rileva e neutralizza pattern di prompt injection noti, limita la lunghezza degli input, e normalizza il contenuto.

Action Authorization Framework

Ogni azione richiede autorizzazione esplicita. Il framework implementa un modello di permessi granulare con supporto per: ruoli, capability-level permissions, rate limiting per azione, approval workflow per azioni critiche.

Output Filtering

Le risposte dell'LLM vengono filtrate per rimuovere eventuale contenuto sensibile (PII, credenziali) che potrebbe essere stato incluso erroneamente. Pattern matching e classificazione ML identificano contenuto da redarre.

Audit Trail Completo

Ogni interazione, decisione e azione viene loggata con dettaglio sufficiente per ricostruire il flusso completo. I log sono immutabili e conservati secondo policy di retention configurabili.

Governance Layer

Un layer trasversale applica policy di governance definite dall'amministratore: limiti di utilizzo, contenuti vietati, azioni proibite, escalation automatica per scenari critici.

8. Glossario

Termine	Definizione
Agentic Website	Sito web intelligente capace di adattare contenuti e funzionalità in risposta alle richieste dell'utente, compiendo azioni autonome per suo conto.
Capability	Funzionalità atomica che l'agente può invocare per compiere un'azione o recuperare informazioni.
Graceful Degradation	Capacità del sistema di continuare a funzionare in modo ridotto quando alcuni componenti falliscono.
Intent	L'obiettivo o intenzione sottostante alla richiesta dell'utente, derivato dall'analisi del linguaggio naturale.
LLM	Large Language Model - modello di intelligenza artificiale addestrato su grandi quantità di testo.
MCP	Model Context Protocol - standard emergente per l'interazione tra agenti AI e servizi web.
Prompt Injection	Attacco in cui input malevoli tentano di manipolare il comportamento del modello AI.
RAG	Retrieval-Augmented Generation - tecnica che combina ricerca documentale con generazione AI.
Turn	Singolo scambio di messaggi tra utente e agente in una conversazione.
Vector Store	Database specializzato per memorizzare e ricercare embedding vettoriali.

— Fine Documento —