



# Code7Crusaders

Software Development Team

Analisi per la scelta delle tecnologie backend

**Membri del Team:**

Enrico Cotti Cottini, Gabriele Di Pietro, Tommaso Diviesti  
Francesco Lapenna, Matthew Pan, Eddy Pinarello, Filippo Rizzolo

**Data:** 12 Febbraio 2025

## Indice

<b>1</b>	<b>Obiettivo</b>	<b>2</b>
<b>2</b>	<b>Analisi tecnologie backend scelte</b>	<b>2</b>
2.1	Flask . . . . .	2
2.2	LangChain . . . . .	3
<b>3</b>	<b>Possibili alternative a Flask</b>	<b>4</b>
3.1	FastAPI . . . . .	4
3.2	Django REST Framework . . . . .	4
<b>4</b>	<b>Possibili alternative a LangChain</b>	<b>5</b>
4.1	Haystack . . . . .	5
<b>5</b>	<b>Confronto alternative a Flask</b>	<b>6</b>
<b>6</b>	<b>Confronto tra LangChain e Haystack</b>	<b>7</b>
<b>7</b>	<b>Conclusioni su Flask</b>	<b>7</b>
<b>8</b>	<b>Conclusioni su LangChain</b>	<b>7</b>

## Elenco delle tabelle

1	Tabella di confronto tecnico tra Flask, Django e FastAPI . . . . .	6
2	Tabella di confronto tra LangChain e Haystack . . . . .	7

# 1 Obiettivo

Questo documento si pone l'obiettivo di confrontare le diverse tecnologie backend in modo da avere un'idea precisa e prendere una decisione sicura per la scelta di esse e per la loro integrazione all'interno del nostro progetto. Il confronto considera aspetti tecnici, caratteristiche, vantaggi e svantaggi dei vari framework discussi. Le nostre scelte finali sono l'utilizzo di Flask e di Langchain per diverse motivazioni chiarite nei paragrafi successivi.

## 2 Analisi tecnologie backend scelte

### 2.1 Flask

Flask è un *microframework* web per Python, il che significa che fornisce gli strumenti di base necessari per sviluppare un'applicazione web, lasciando allo sviluppatore la possibilità di aggiungere altre funzionalità secondo le necessità. È una tecnologia open source e gratuita che fornisce un modo semplice per creare e distribuire applicazioni web dinamiche, offrendo molta libertà e controllo sullo sviluppo dell'applicazione.

#### Vantaggi

- **Semplicità e flessibilità:** si può iniziare con un'applicazione piccola e espanderla facilmente man mano che cresce
- **Personalizzazione:** permette di costruire un'applicazione che si adatti perfettamente alle proprie esigenze specifiche
- **Documentazione completa e community attiva:** è molto facile trovare aiuto quando si ha qualche tipo di problema
- **Facile integrazione con altre tecnologie:** è compatibile con una vasta gamma di tecnologie, incluse basi di dati, sistemi di autenticazione e altri ancora
- **Sviluppo rapido ed efficiente:** Garantisce di portare a termine l'applicazione velocemente e in modo efficace
- **Scalabilità:** è possibile adattare la propria applicazione a un numero maggiore di utenti e carichi di lavoro senza dover riscrivere il codice da zero
- **Tempo di caricamento ridotto:** è leggero ed efficiente in termini di risorse, il che significa che le pagine web si caricano rapidamente e senza problemi

#### Svantaggi

- **Maggiore quantità di codice:** Flask richiede più codice personalizzato rispetto a framework più completi
- **Manutenzione e aggiornamenti:** molte funzionalità devono essere aggiunte tramite estensioni o pacchetti di terze parti
- **Dipendenza da terze parti:** molte funzionalità avanzate per costruire applicazioni web complete devono essere aggiunte tramite librerie esterne

## 2.2 LangChain

LangChain è una libreria progettata per facilitare l'integrazione di Large Language Models nei sistemi software. Consente di gestire conversazioni, memoria contestuale e connettori a database vettoriali, rendendo l'interazione con i modelli più strutturata e personalizzabile. Grazie a queste funzionalità, LangChain è particolarmente utile per lo sviluppo di chatbot avanzati, sistemi di generazione automatica di testo e applicazioni basate su intelligenza artificiale.

### Vantaggi

- **Facile integrazione con LLMs:** semplifica la connessione con modelli di linguaggio di grandi dimensioni, permettendo una rapida implementazione di chatbot e assistenti virtuali.
- **Gestione della memoria contestuale:** consente di mantenere il contesto delle conversazioni, migliorando l'interattività e la coerenza delle risposte.
- **Compatibilità con database vettoriali:** supporta l'integrazione con FAISS, Milvus e altre tecnologie per la ricerca semantica efficiente.
- **Modularità:** offre componenti riutilizzabili che possono essere combinati per costruire flussi di interazione complessi.
- **Flessibilità:** permette di personalizzare il comportamento dell'applicazione in base alle esigenze specifiche dell'utente.
- **Community attiva e in crescita:** grazie a una comunità attiva, è possibile trovare risorse, documentazione e supporto per lo sviluppo.

### Svantaggi

- **Dipendenza dagli LLMs:** le prestazioni e l'affidabilità dipendono dalla qualità e dalla disponibilità dei modelli di linguaggio.
- **Complessità di configurazione:** richiede una certa esperienza per configurare e ottimizzare le pipeline di elaborazione del linguaggio.
- **Costi di esecuzione:** l'uso di modelli di linguaggio avanzati può comportare costi elevati in termini di risorse computazionali e API a pagamento.
- **Integrazione con servizi esterni:** alcune funzionalità necessitano di dipendenze esterne, aumentando la complessità della manutenzione del progetto.

**Gestione delle Run con LangSmith** LangSmith è un tool che facilita la gestione e l'analisi delle esecuzioni dei chatbot basati su LangChain. Consente di ottenere informazioni dettagliate sulle metriche delle run, tra cui:

- **ID:** identificativo univoco della run.
- **Nome:** nome assegnato alla sessione.
- **Input:** dati di input elaborati dal modello.
- **Tempo di inizio:** timestamp di avvio dell'esecuzione.
- **Tempo di fine:** timestamp di completamento della run.
- **Errore:** eventuali errori incontrati durante l'esecuzione.

- **Output:** risultato generato dal modello.
- **Total Token:** numero totale di token utilizzati nella run.
- **Costo totale:** stima dei costi basata sul consumo di token.

questo strumento ci sarà utile per realizzare l'esportazione dei dati delle metriche dalla dashboard dedicata dell'amministratore.

## 3 Possibili alternative a Flask

### 3.1 FastAPI

FastAPI è un framework web moderno, ad alte prestazioni, per la costruzione di API RESTful con Python. È progettato per essere facile da usare, veloce da sviluppare, e altamente efficiente, sfruttando le potenzialità delle moderne caratteristiche di Python. Inoltre, possiede la capacità di produrre API estremamente veloci e sicure, con un focus sull'efficienza sia in termini di prestazioni che di sviluppo rapido.

#### Vantaggi

- **Prestazioni elevate:** FastAPI è uno dei framework più veloci in Python grazie al supporto nativo per la programmazione asincrona
- **Supporto per la programmazione asincrona:** permette di gestire richieste asincrone in modo molto efficiente, riducendo il tempo di attesa e migliorando la gestione delle risorse durante le operazioni I/O intensive
- **Facilità di testing:** facilita il testing delle API grazie alla sua struttura di testing integrata e alla capacità di generare automaticamente mock di richieste e risposte
- **Validazione automatica dei dati:** i dati inviati tramite le richieste (come JSON) sono automaticamente validati contro i modelli definiti, riducendo il codice necessario per la gestione degli errori e migliorando la sicurezza

#### Svantaggi

- **Curva di apprendimento:** FastAPI, pur essendo molto potente, può avere una curva di apprendimento ripida per chi non è familiare con Python e con la programmazione asincrona
- **Documentazione e community:** ha una community più piccola rispetto ad altri framework più consolidati come Django o Flask. Questo significa che potrebbe esserci meno documentazione, risorse online o librerie di terze parti da utilizzare
- **Meno funzionalità predefinite:** fornisce solo le funzionalità essenziali per la creazione di API. Funzionalità come autenticazione, gestione dei permessi, e admin panel devono essere implementate tramite estensioni o personalizzazioni

### 3.2 Django REST Framework

Django è un framework di alto livello, open source, costruito su Python che incoraggia uno sviluppo rapido e un design pulito e pragmatico. Si basa sul paradigma MTV, ossia *"Model-Template-View"*. In questo esplicita la sua natura full-stack, in quanto gestiamo in modo olistico le interazioni tra la parte back-end (i modelli) e la parte front-end (i template) tramite viste (view).

## Vantaggi

- **Serializzazione dei dati:** fornisce un sistema di serializzazione che consente di convertire facilmente i dati tra formati (come JSON o XML) e oggetti Python, semplificando la gestione dei dati tra il client e il server
- **Autenticazione e permessi avanzati:** include supporto nativo per vari metodi di autenticazione (come token-based o sessione) e per la gestione dei permessi
- **Paginazione:** offre un sistema di paginazione per gestire grandi quantità di dati nelle risposte API, migliorando le prestazioni quando ci sono molte risorse
- **Struttura sicura:** si basa su diversi meccanismi integrati che aiutano a proteggere le applicazioni API da vulnerabilità comuni e garantire una gestione sicura dei dati e degli accessi
- **Comunità e supporto:** ha una grande comunità di sviluppatori, documentazione completa e numerosi esempi

## Svantaggi

- **Overhead:** in alcuni casi, per progetti molto semplici o con API leggere, questa completezza potrebbe introdurre overhead non necessario, aumentando la complessità e i tempi di sviluppo
- **Performance:** Poiché gestisce molte operazioni, come la serializzazione, la validazione e la gestione dei permessi, in alcuni casi può essere meno performante rispetto a soluzioni più leggere per API molto grandi o ad alte prestazioni
- **Apprendimento complesso:** apprendimento che coinvolge concetti avanzati, abilità e conoscenze che richiedono tempo, sforzo e un certo livello di comprensione approfondita

# 4 Possibili alternative a LangChain

## 4.1 Haystack

Haystack è un framework progettato per la ricerca di informazioni e la costruzione di agenti conversazionali avanzati. Fornisce strumenti per il recupero di documenti, la generazione di risposte e l'integrazione con modelli di linguaggio.

## Vantaggi

- **Focalizzato sulla ricerca di informazioni:** offre funzionalità avanzate per il recupero di documenti pertinenti.
- **Supporto per agenti conversazionali:** include strumenti per creare chatbot con memoria contestuale.
- **Integrazione con più modelli:** compatibile con LLMs come GPT, BERT e T5 per migliorare le risposte.
- **Supporto per pipeline personalizzate:** consente di costruire flussi di elaborazione complessi con componenti modulari.

### Svantaggi

- **Maggiore complessità:** richiede più configurazioni rispetto a LangChain per essere utilizzato efficacemente.
- **Possibili costi elevati:** alcune integrazioni con modelli avanzati possono risultare costose in ambienti di produzione.
- **Dipendenza da componenti esterni:** necessita di più strumenti per sfruttare appieno le sue capacità.

## 5 Confronto alternative a Flask

Aspetto	Flask	Django	FastAPI
<b>Ambito di applicazione</b>	Microframework per applicazioni leggere	Framework completo per applicazioni web	Ottimo per API ad alte prestazioni
<b>Sicurezza</b>	Funzionalità di base, personalizzabile	Sicurezza integrata (autenticazione, CSRF)	Sicurezza per API, richiede librerie esterne
<b>Flessibilità</b>	Altamente flessibile, poche convenzioni	Struttura rigida con molte convenzioni	Flessibile, ma con focus su API asincrone
<b>Prestazioni</b>	Molto buone	Prestazioni medio-alte, non ottimizzato per async	Eccellenti, soprattutto per operazioni asincrone
<b>Velocità di apprendimento</b>	Facile, ma richiede più lavoro per funzionalità avanzate	Curva di apprendimento più ripida	Facile per chi conosce async/await e type hints
<b>Community</b>	Molto ampia e matura	Grande e attiva	In rapida crescita, ma più piccola

Tabella 1: Tabella di confronto tecnico tra Flask, Django e FastAPI

## 6 Confronto tra LangChain e Haystack

Aspetto	LangChain	Haystack
<b>Ambito di applicazione</b>	Framework per costruire chatbot avanzati con memoria contestuale e gestione dei flussi conversazionali	Progettato per la ricerca di informazioni e il recupero documentale
<b>Integrazione con LLM</b>	Nativo, supporta OpenAI, Hugging Face e altri	Supporta diversi LLM ma con configurazioni più complesse
<b>Memoria contestuale</b>	Supporto avanzato con gestione della memoria a lungo termine	Limitato alla gestione delle interazioni immediate
<b>Facilità d'uso</b>	API intuitive e modulari	Richiede più configurazione e tuning
<b>Gestione delle conversazioni</b>	Strutturato per la creazione di chatbot con azioni e memoria	Più focalizzato sulla ricerca di documenti che sulle conversazioni
<b>Community e supporto</b>	Grande community e supporto attivo	Community in crescita, meno risorse rispetto a LangChain

Tabella 2: Tabella di confronto tra LangChain e Haystack

## 7 Conclusioni su Flask

**Flask** è stato scelto per la sua leggerezza e semplicità nel creare API RESTful. Essendo un micro-framework, permette di sviluppare rapidamente un backend senza imporre dei vincoli rigidi. Inoltre, la sua ampia documentazione e la sua flessibilità lo rendono ideale per prototipi e progetti in evoluzione. Per quanto riguarda **Langchain**, invece, si tratta di una libreria progettata per facilitare l'integrazione dei *Large Language Models (LLMs)* nei sistemi Software. Esso permette di gestire conversazioni, memoria contestuale e connettori a database vettoriali, rendendo l'interazione con i modelli più strutturata e personalizzabile.

## 8 Conclusioni su LangChain

LangChain è stato scelto per il nostro progetto in quanto offre un'integrazione ottimale con modelli OpenAI, semplifica la creazione di chatbot con memoria contestuale e permette la gestione di flussi conversazionali avanzati. Grazie a strumenti come LangSmith, è possibile monitorare le metriche delle esecuzioni, migliorare i prompt e ottimizzare il comportamento del chatbot. Inoltre, la modularità di LangChain consente una rapida estensione delle funzionalità, rendendolo la scelta ideale per il nostro caso d'uso.