



## Glossario

## Indice

A	3
B	4
C	5
D	6
E	7
F	8
G	9
H	11
I	12
J	13
K	14
L	15
M	16
N	17
O	18
P	19
Q	20
R	21
S	22
T	23
U	25
V	26
W	27
X	28
Y	29



---

# A

---

## Agente

Un agente è un'entità autonoma in grado di: ragionare, pianificare, interagire con API e strumenti esterni, modificando l'ambiente esterno in cui opera per raggiungere obiettivi specifici.

I principali componenti di un agente sono:

- **Model:** modello generativo (es. [LLM](#)) usato dall'agente. La sua efficacia dipende dalla qualità e quantità dei dati su cui è stato addestrato.  
Diversamente dal modello puro, l'agente può ampliare le proprie capacità tramite interazioni con strumenti esterni.
- **Reasoning Loop:** processo iterativo di pensiero e decisione dell'agente., in cui distingue i passaggi necessari e decide quali azioni eseguire.  
Può essere implementato con framework come ReAct.
- **Tools:** strumenti esterni con cui l'agente interagisce con l'ambiente esterno per ottenere informazioni o eseguire azioni. Vedi [Tool](#).

## Amazon Web Services

### Ambiente

### Analisi dei Requisiti

### API

Un'API (Application Programming Interface) è un insieme di regole e protocolli che consente a diverse applicazioni software di comunicare tra loro. Fornisce un'interfaccia standardizzata per accedere a funzionalità o dati specifici di un'applicazione, servizio o piattaforma, facilitando l'integrazione e l'interoperabilità tra sistemi diversi.

### API Token

### AWS

---

## B

---

### **Bedrock**

### **Board**

Una Board è una bacheca in stile Kanban che permette di monitorare lo stato di avanzamento del progetto, attraverso l'organizzazione visuale delle attività (issue) da svolgere (Backlog, se non è ancora da prendere in carico - Ready, quando la issue è pronta per essere svolta), in corso (In Progress), in revisione (in Review, in attesa di verifica e approvazione) e terminate (Done).

### **Branch**

Un branch (ramo) è una linea di sviluppo indipendente (una copia separata del codice sorgente) all'interno di un sistema di controllo versione (VCS), come Git. Permette agli sviluppatori di lavorare su funzionalità, correzioni di bug o esperimenti senza influenzare direttamente il ramo principale.

---

## C

---

**Chain-of-thought prompting**

**Cognito**

**Container**

**Context Prompt**

---

## D

---

**Database non relazionale**

**Database relazionale**

**Database vettoriale**

**Dev**

**Developer**

Il Developer (Dev) è la figura responsabile di progettare, scrivere, testare e mantenere il codice necessario a realizzare le funzionalità di un progetto software. Traduce i requisiti forniti da Project Manager e Tech Lead in soluzioni tecniche concrete, contribuendo allo sviluppo, all'evoluzione e alla qualità del prodotto.

Le sue principali responsabilità sono:

- **Sviluppo del codice:** implementa nuove funzionalità, corregge bug e realizza le componenti software richieste.
- **Collaborazione tecnica:** interagisce con Tech Lead e altri sviluppatori per garantire coerenza e qualità.
- **Test e qualità:** scrive test, verifica il funzionamento del proprio codice e contribuisce alla stabilità del sistema.
- **Documentazione:** documenta funzionalità, API, processi e soluzioni implementate.
- **Manutenzione:** aggiorna, ottimizza e riorganizza il codice esistente quando necessario.

**Docker**

**Dockerfile**

**Documentazione**

E



---

## F

---

**Few-shot prompting**

**FM**

**Foundation Model**

**Framework**

---

## G

---

### Git

Git è un sistema di controllo versione distribuito che consente di tracciare le modifiche apportate ai file e coordinare il lavoro tra più sviluppatori.

I principali comandi di git sono:

- **git clone <url>**: Clona un repository remoto in locale.
- **git add <file>**: Aggiunge file specifici all'area di staging.
- **git add .**: Aggiunge tutti i file modificati all'area di staging.
- **git rm <file>**: Rimuove file specifici dal repository e dall'area di staging
- **git commit -m "messaggio"**: Crea un commit con i file nell'area di staging e un messaggio descrittivo.
- **git push origin <branch>**: Invia i commit locali al repository remoto sul branch specificato.
- **git pull origin <branch>**: Recupera e integra le modifiche dal repository remoto al branch locale.
- **git status**: Mostra lo stato dei file nel repository, inclusi quelli modificati, aggiunti o non tracciati.
- **git diff**: Mostra le differenze tra i file modificati e l'ultima versione committata.
- **git branch**: Elenca, crea o elimina branch nel repository.
- **git checkout <branch>**: Passa a un branch specifico.
- **git switch <branch>**: Alternativa a git checkout per cambiare branch.
- **git fetch**: Recupera gli aggiornamenti dal repository remoto senza integrarli automaticamente.
- **git merge <branch>**: Unisce un branch specifico nel branch corrente.
- **git rebase <branch>**: Integra le modifiche di un branch specifico riscrivendo la cronologia dei commit.

### GitHub

GitHub è una piattaforma di hosting per lo sviluppo software e il versionamento basato su Git che permette di:

- collaborare allo sviluppo di software in modo distribuito,
- gestire issue, pull request, documentazione
- automatizzare processi tramite GitHub Actions

È ampiamente utilizzata per progetti open-source e collaborativi.

## Gitflow

Gitflow è un workflow per Git basato su branch multipli con ruoli specifici. Permette una chiara separazione tra sviluppo, rilascio e manutenzione.

La struttura principale include:

- **master**: contiene solo versioni pronte al rilascio
- **develop**: rappresenta lo stato corrente dello sviluppo
- **feature branches**: uno per ogni nuova funzionalità
- **release branches**: preparazione della versione
- **hotfix branches**: correzioni urgenti sul master

I principali comandi di gitflow sono:

- **git flow init**: Inizializza un repository Gitflow.
- **git flow feature start <nome-feature>**: Crea e passa a un nuovo branch di funzionalità.
- **git flow feature finish <nome-feature>**: Unisce il branch di funzionalità nel branch develop e lo elimina.
- **git flow feature publish <nome-feature>**: Pubblica il branch di funzionalità sul repository remoto.
- **git flow feature pull origin <nome-feature>**: Recupera e integra le modifiche dal branch di funzionalità remoto.
- **git flow feature track <nome-feature>**: Crea un branch di funzionalità locale che traccia il branch remoto.
- **git flow release start release [BASE] <versione>**: Crea un nuovo branch di release a partire dal branch specificato (di default develop).
- **git flow feature release publish <versione>**: Pubblica il branch di release sul repository remoto.
- **git flow release finish <versione>**: Unisce il branch di release nei branch master e develop, crea un tag per la versione e elimina il branch di release.
- **git flow hotfix start <nome-hotfix> [BASE]**: Crea un nuovo branch di hotfix a partire dal branch specificato (di default master).
- **git flow hotfix finish <nome-hotfix>**: Unisce il branch di hotfix nei branch master e develop, crea un tag per la versione e elimina il branch di hotfix.

## GitHub Actions

GitHub Actions è il sistema di automazione di GitHub che consente di eseguire workflow automatizzati direttamente all'interno del repository. Permette di creare processi personalizzati per la compilazione, il test, il rilascio e la distribuzione del codice, nonché per altre attività legate allo sviluppo software, in risposta ad eventi del repository, come push o pull request.

## GitHub Linguist

# H

Human Prompt

---

**I**

---

**IaaS****IAM****Identity and Access Management****Infrastructure as a Service****Issue**

Un'issue è uno strumento di GitHub utilizzato per segnalare bug, proporre nuove funzionalità, discutere idee o tracciare attività.

---

## J

---

JavaScript

JSON

# K

---

## L

---

**Langchain**

**Langgraph**

**Large Language Model**

**LaTeX**

LaTeX Linguaggio di marcatura (mark-down) compilato per la realizzazione di documenti.

**Linguist**

**LLM**



---

## M

---

### Milestone

Una milestone è un contenitore che raggruppa issue e pull request sotto un obiettivo comune, di solito legato a una versione, uno sprint o una fase del progetto.

Permette di:

- monitorare i progressi verso obiettivi specifici (percentuale di completamento),
- organizzare le attività verso una scadenza condivisa,
- facilitare la pianificazione e la comunicazione all'interno del team.

### MongoDB

### MongoDB Atlas

---

## N

---

**NestJS**

**NodeJS**

**NoSQL**

**Norme di Progetto**

**npm**

---

## O

---

### One-shot prompting

#### Orchestratore Autocratico

L'orchestratore è il componente centrale di un sistema ad agenti che coordina e gestisce le operazioni degli agenti per raggiungere obiettivi specifici in modo efficiente. In un sistema ad agenti con orchestratore autocratico, l'orchestratore prende decisioni centralizzate e dirige le azioni degli agenti senza consultare o coinvolgere gli agenti stessi nel processo decisionale. In questo sistema gli agenti svolgono compiti molto specifici.

---

## P

---

### PaaS

### Platform as a Service

### Project Manager

Il Project Manager (PM) è la figura responsabile della gestione completa del ciclo di vita di uno o più progetti, garantendone la realizzazione nel rispetto di tempi, costi, qualità e obiettivi stabiliti. Pur non essendo necessariamente un esperto tecnico di sviluppo software, coordina persone, attività e risorse, fungendo da punto di contatto tra team di sviluppo, organizzazione e cliente.

Le sue principali responsabilità sono:

- **Gestione del Progetto:** Definire obiettivi, scope, tempistiche e budget del progetto, assicurando che vengano rispettati durante tutto il ciclo di vita del progetto
- **Pianificazione e Organizzazione:** Definisce il piano di progetto, stabilisce scadenze, milestones e priorità operative.
- **Comunicazione con il Cliente:** Raccoglie requisiti, chiarisce aspettative, condivide stato avanzamento e gestisce richieste o modifiche.
- **Assegnazione delle Risorse/Assegnazioni (chi fa cosa):** Decide a quali Developer assegnare i vari progetti di cui è responsabile.
- **Monitoraggio dell'Avanzamento:** Controlla lo stato dei lavori, interviene in caso di rischi o ritardi, assicura il rispetto degli standard.
- **Qualità e Conformità:** Definisce standard e requisiti per documentazione e deliverable.

### Piano di Progetto

### Piano di Qualifica

### PMD

### PoC

### Proof of Concept

Q

Query

## R

**Raccolta**

**React**

**Report**

**REST**

**Role Prompt**

---

## S

---

**SaaS**

**Sarif**

**Schema**

**semgrep**

**Serverless**

**Sistema ad agenti**

Un sistema ad agenti è un insieme di entità autonome, chiamate agenti, che interagiscono tra loro per raggiungere obiettivi specifici all'interno di un ambiente condiviso. Ogni agente possiede capacità di percezione, decisione e azione, permettendo loro di adattarsi e collaborare in modo dinamico.

**Software as a Service**

**SonarQube**

**Sprint**

Uno sprint è una piccola porzione di tempo (due settimane nel nostro caso) durante il quale un team porta a termine una determinata quantità di lavoro. Gli sprint prevedono

- una riunione iniziale, per pianificare gli obiettivi e lo svolgimento dello sprint
- check-in giornalieri, per controllare l'andamento dello sprint e reagire tempestivamente a eventuali problematiche
- una riunione retrospettiva al termine dello sprint, per pianificare e migliorare gli sprint futuri

**Swagger**

**System Prompt**

---

## T

---

### Tag

### Tag raccolta

### Tech Lead

Un tech lead (o technical lead) è il supervisore tecnico di più progetti. È la figura che unisce visione tecnica e capacità di coordinamento, fungendo da ponte tra il Project Manager e il team di sviluppo.

Garantisce la solidità dell'architettura, la qualità del codice e la corretta applicazione delle tecnologie, supportando il team nella risoluzione dei problemi più complessi.

Le sue principali responsabilità sono:

- **Supporto e Mentoring:** fornire supporto tecnico al team di sviluppo e al Project Manager, aiutando a risolvere problemi complessi e guidando le decisioni tecniche riguardo alle tecnologie di cui si occupa.
- **Standard Tecnici:** Assicurare che il codice e le pratiche di sviluppo rispettino gli standard di qualità (guarda i risultati dei test e il test coverage, per valutare la qualità complessiva del codice prodotto), sicurezza (identifica le vulnerabilità o criticità del prodotto) e performance definiti. Mantiene aggiornate librerie e dipendenze e monitora i rischi legati alle tecnologie adottate.
- **Architettura e Design:** Revisionare e approvare le decisioni architetturali, contribuendo alla definizione della direzione tecnica dei progetti.
- **Monitoraggio Aggregato:** Analizza dati provenienti da più repository e progetti (test results, test coverage, stato di sicurezza, indicatori di qualità) per identificare trend, criticità e colli di bottiglia.

### Test coverage

### Tool

Un tool è uno strumento esterno che estende le capacità di un [Agente](#), permettendo ad esempio l'esecuzione di codice e recuperare informazioni aggiornate<sup>1</sup>.

I tool si dividono in:

- **Extensions:** collegano l'[Agente](#) a [API](#) esterne in modo autonomo, definendo quando, come e cosa aspettarsi dalla risposta.
- **Function calling:** usata quando l'[API](#) è "segreta"; guida gli input, ma la chiamata vera la fa un sistema esterno.
- **Data store / Database vettoriale:** permette all'agente di cercare informazioni in database, documenti o siti.

---

<sup>1</sup><https://docs.langchain.com/oss/python/langchain/tools>



Trivy

TypeScript

U

---

## V

---

### VCS

Un Version Control System (VCS) è un sistema software che gestisce e traccia tutte le modifiche effettuate a un insieme di file, come documenti, programmi o siti web. Ogni modifica viene registrata come una revisione identificata da un numero o codice, accompagnata da timestamp e autore.

Un VCS permette di confrontare versioni, ripristinare stati precedenti, condividere cambiamenti tra più utenti e gestire attività collaborative tramite funzionalità come branching, merging e tracciamento delle modifiche.

---

## W

---

### **Workflow**

Il workflow è il processo o modello che stabilisce come vengono gestite le modifiche nello sviluppo del codice all'interno di un'organizzazione o di un progetto.

Ogni workflow definisce come si creano i branch, come si integrano le modifiche e come si collabora.

Il tipo di workflow dipende dal VCS utilizzato e dalle esigenze del team di sviluppo.

### **Way of Working**

**X**

# Y

## YAML

---

## Z

---

**Zero-short prompting**