Glossario

Indice

A 3
B 4
C 5
D 6
E 8
F 9
G 10
H 13
I 14
J 15
K 16
L 17
M 18
N 19
O 20
P 21
Q 22
R 23
S 24
T 25
U 27
V 28
W 29
X 30
Y 31

Z 32

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

Vedi Variabili d'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. È un'interfaccia per accedere a funzionalità o dati specifici di un'applicazione, servizio o piattaforma, facilitando l'integrazione e l'interoperabilità tra sistemi diversi.

API Token

Chiave unica talvolta necessaria per accedere a una API.

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

Processo isolato per le componenti di un'applicazione. Un container è:

- autocontenuto, nel senso che non si appoggia su dipendenze della macchina su cui corre
- isolato, garantendo maggiore sicurezza per le applicazioni. A differenza di una macchina virtuale la separazione non avviene a livello kernel.
- gestito indipendentemente dagli altri container
- portatile, nel senso che lo stesso container può correre allo stesso modo su macchine diverse

Si ottiene eseguendo una Immagine (Docker) tramite Docker.

Context Prompt

CVE

CVSS

CWE

D

Database non relazionale

Database relazionale

Database vettoriale

Dev

Vedi Developer.

Developer

Il Developer 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

Piattaforma per sviluppare e condividere applicazioni in modo che siano separate dalla loro infrastruttura (vedi Container). Facilita la riproducibilità della costruzione di software nonché il loro sviluppo e la scalabilità.

Utilizza un'architettura client-server, dove il demone `dockerd` (locale o remoto) e il client (ad es. docker compose) comunicano tramite API REST tramite socket o rete.

docker compose

Comando che permette di eseguire Container multipli secondo una configurazione specificata in un file YAML, permettendo la configurazione ad esempio di reti, volumi e variabili d'ambiente.

Dockerfile

File contenente le istruzioni per la costruzione di un'Immagine (Docker). A seguire istruzioni comuni[1]:

- `FROM`: definisce l'immagine di partenza
- `WORKDIR`: definisce la *working directory* all'interno del filesystem dummy dell'immagine
- `COPY <host-path> <image-path>`: copia tutti i file/directory presenti a `<host-path>` (dell'utente) in `<wd>/<image-path>` (dell'immagine). Qui, `<wd>` è definita dal comando `WORKDIR`
- `RUN <cmd>`: esegue `<cmd>` da riga di comando. Si noti che
  1. la cache non viene invalidata in build successive, salvo l'esecuzione di `docker build --no-cache` in fase di build
  2. dal momento che `RUN` genera un nuovo layer, effettuare un'operazione che influenza(ad esempio) solamente la shell corrente non influenzerà i comandi successivi. Ad esempio, se entro attivo un `virtualenv` di python con un comando `RUN`, e successivamente eseguo `RUN pip install...`, il pacchetto non verrà installato all'interno dell'ambiente virtuale.
- `ENV <name> <value>`: assegna `<name>=<value>` a livello di Ambiente
- `EXPOSE <port>`: indica la porta che vorresti esporre. In realtà è solo documentativa: il comando in sé non espone la porta, ma lo comunica a chi eseguirà poi il container. Starà a quella persona poi pubblicare la porta ad esempio utilizzando la flag `-p` in `docker run -p<host>:<container>/<protocol>`. In tal caso, il comando specifica che avviene port-forwarding dalla porta `<host>` dell'host a quella `<container>` del container, utilizzando protocollo `<protocol>` che può essere `udp` o `tcp`.
- `USER <user/uuid>`: specifica l'utente per le istruzioni successive
- `CMD ["<command>", "<arg1>"]`: specifica il comando che il container eseguirà ogni volta che viene fatto correre. Ce ne può essere solamente uno per container e deve essere l'ultima istruzione presente nel file.

Documentazione

[1] https://docs.docker.com/reference/dockerfile

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

Immagine (Docker)

Un pacchetto in sola lettura che contiene tutti i file, le librerie e le configurazioni per eseguire un Container. Un'immagine è composta da vari strati (*layer*) che rappresentano le modifiche applicate a un'immagine di partenza. La composizioni in strati permette una costruzione di container più rapida e il risparmio di spazio, dal momento che sono condivisibili.

Immagini docker possono essere trovate su `https://hub.docker.com/`.

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

NVD

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.

OWASP Top 10

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 obbiettivi 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[2].

I tool si dividono in:

- **Extensions**: collegano l'Agente a API esterne in modo autonomo, definendo quando, come e cosa aspettarsi dalla risposta.

[2] https://docs.langchain.com/oss/python/langchain/tools

- **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.

Trivy

TypeScript

U

V

Variabili d'Ambiente

Un insieme di variabili di sistema (`NAME=VALUE`) che un processo può leggere, modificando il proprio comportamento. Una variabile d'ambiente nota è `PATH`, che fa riferimento a una lista di cartelle contenenti file eseguibili (che quindi possono essere eseguiti per nome da riga di comando). Nel nostro caso, tramite Docker si possono indicare dei file che andranno a definire l'ambiente di un container. In questo modo si può facilmente (utilizzando la libreria `dotenv`) evitare di scrivere dei segreti all'interno di file che saranno caricati e condivisi pubblicamente. Mai condividere un `.env` file.

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