Glossario

Autore	Alessandro Morabito, Giulia Romanato
Verificatore	Alessandro Morabito, Giulia Romanato
Approvazione	Alessandro Morabito

Registro delle versioni

Versione	Data	Autore	Verificatore	Descrizione delle modifiche
1.0.0	24/02/2026	Giulia Romanato	Alessandro Morabito	Documento comprendente terminologia fino a revisione RTB.
0.8.0	24/02/2026	Alessandro Morabito	Giulia Romanato	Aggiunti: Attore, Casi d'uso, Norme di Progetto, Owner, Piano di Progetto, Piano di Qualifica, Scenario.
0.7.0	22/02/2026	Alessandro Morabito	Giulia Romanato	Aggiunti: Code Coverage, Langchain, Langgraph, LLM, MongoDB, PoC, Report, SBOM, semgrep, syft.
0.6.0	20/02/2026	Alessandro Morabito	Giulia Romanato	Aggiunti: Framework, JavaScript, JSON, NestJS, NodeJS, npm, React, REST, TypeScript.
0.5.0	18/01/2026	Alessandro Morabito	Giulia Romanato	Aggiunti: Owasp TOP 10.
0.4.0	17/01/2026	Alessandro Morabito	Giulia Romanato	Aggiunti: CIA, CVE, CVSS, CWE, Debolezza, NVD, Vulnerabilità.
0.3.0	10/01/2026	Alessandro Morabito	Giulia Romanato	Aggiunti: Analisi dei Requisiti, Analisi dei Requisiti (Processo), Documentazione, Glossario, Requisito.
0.2.0	08/01/2026	Alessandro Morabito	Giulia Romanato	Aggiunti: API token, Container, Dev, Docker, docker compose, Dockerfile, Immagine (Docker), Variabili d'Ambiente.
0.1.0	02/12/2025	Giulia Romanato	Alessandro Morabito	Aggiunti: API, Agente, Board, Branch, Developer, Git, GitHub, Gitflow, GitHub Actions, Issue, LaTeX, Milestone, Orchestratore Autocratico, Project Manager, Sistema ad Agenti, Tech Lead, VCS, Workflow.

Indice

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.
- **Tool**: strumenti esterni con cui l'agente interagisce con l'ambiente esterno per ottenere informazioni o eseguire azioni. Vedi Tool.

Ambiente

Vedi Variabili d'Ambiente

Analisi dei Requisiti

Documento che formalizza i requisiti funzionali e non funzionali di CodeGuardian.

Analisi dei Requisiti (Processo)

Processo di studio dei bisogni degli utenti per definire un Requisito.

API

Application Programming Interface. Un'API 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.

Attore

Un Attore è un'entità esterna al sistema che interagisce con esso per scambiare informazioni o attivare funzionalità. Un attore può essere un essere umano, un componente hardware o un altro sistema software. Gli attori vengono classificati in primari, se danno inizio al caso d'uso per raggiungere un obiettivo, o secondari, se forniscono un servizio al sistema durante l'esecuzione dello scenario.

B

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

Caso d'uso

Un Caso d'uso rappresenta una descrizione coerente delle funzionalità che il sistema offre per soddisfare un obiettivo specifico di un utente. Formalmente, consiste in un insieme di scenari (principali, alternativi o di errore) che descrivono le interazioni tra gli attori e il sistema stesso, finalizzate al raggiungimento di un risultato di valore per l'interessato.

CIA

Confidentiality, Integrity, Availability. Triade che rappresenta i principi chiave della sicurezza informatica.

- **Confidentiality**: limita l'accesso a informazioni sensibili
- **Integrity**: assicura l'accuratezza e l'affidabilità di dati e ed equipaggiamenti
- **Availability**: assicura accesso tempestivo a dati ed equipaggiamenti

Code coverage

Metrica che misura la percentuale di codice sorgente eseguito durante l'esecuzione di una suite di test. In particolare, include:

- Statement coverage
- Branch coverage
- Function coverage
- Path coverage

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.

CVE

Common Vulnerabilities and Exposures. Un glossario di Vulnerabilità formalmente identificate. Utilizziamo il termine per fare riferimento più in particolare ai *CVE Record*: una entry di tale glossario, ovvero dati strutturati riguardanti una certa vulnerabilità. Ciascun record possiede un identificatore univoco, e può essere in uno dei seguenti stati:

- **Reserved**: il codice associato è riservato e non può essere utilizzato
- **Published**: i dati relativi al record sono stati popolati, e il record è stato pubblicato
- **Rejected**: il record non deve essere più considerato, e non viene rimosso per comunicarne lo stato

A ciascun record è inoltre associato un punteggio CVSS.
Dopo che un record viene aggiunto alle CVE, diventa rapidamente disponibile anche nel NVD, che provvederà ad arricchirlo.

CVSS

Common Vulnerability Scoring System. Una metrica che associa un valore numerico compreso tra 0 (bassa gravità) e 10 (alta gravità) a una determinata vulnerabilità. A un valore numerico è associata anche una stringa (*CVSS vector string*) che specifica i valori utilizzati per ottenerlo. Attualmente in versione `4.0`[1].

CWE

Common Weaknesses Enumeration. Una lista di debolezze software e hardware. A ciascuna Debolezza in CWE è associato:

- **Nome** descrittivo
- **Descrizione breve** riassuntiva
- **Descrizione lunga**, destinata a chi può non comprendere per quale motivo possa essere un problema
- **Modalità di introduzione**
- **Potenziali mitigazioni** del problema
- **Consequenze comuni** che si verificano quando un attaccante sfrutta la debolezza
- **Piattaforme applicabili**: linguaggi di programmazione, sistemi operativi, architetture e tecnologie
- **Esempi dimostrativi** che illustrano la vulnerabilità tramite codice, testo e, talvolta, diagrammi
- **Esempi osservati**, ad es. record CVE
- **Relazioni** con altre CWE
- **Riferimenti**, nel senso di citazioni sotto forma di URL, documenti, slide o video

[1] Specifica CVSSv4.0: https://www.first.org/cvss/v4.0/specification-document

D

Database non relazionale

Tipologia di base di dati che permette il salvataggio di dati strutturati, non strutturati e semi strutturati. Possono assumere varie forme come grafi oppure documenti. I database non relazionali sono anche detti NoSQL.

A differenza di un Database relazionale sono facilmente scalabili orizzontalmente, ma non possiedono un linguaggio di interrogazione standard e hanno un supporto limitato per le transazioni (non aderiscono, generalmente, alle proprietà ACID[2]).

Database relazionale

Base di dati basata sul modello relazionale. Salva i dati in formato strutturato all'interno di tabelle (relazioni) composte da righe (tuple) e colonne (attributi) ben definite. La comunicazione con un database relazionale avviene tramite RDBMS (*Relational DataBase Management System*) che aderiscono alle proprietà ACID per garantire l'affidabilità delle transazioni. La comunicazione avviene tramite SQL.

I database relazionali permettono un'ottimizzazione formale della ridondanza e dipendenza dei dati e di effettuare interrogazioni complesse tra tabelle sfruttando la proprietà di integrità referenziale. Sono però complessi da scalare orizzontalmente, sono poco flessibili e inadatti a gestire dati non strutturati o gerarchici.

Database vettoriale

Base di dati che permette il salvataggio di dati sotto forma di vettori (*embedding*), che vengono raggruppati in base alla similarità. Le interrogazioni effettuate non si basano sulle corrispondenze esatte per recuperare i risultati rilevanti, bensì sulla somiglianza dei dati.

DBMS

DataBase Management System. Sistema software progettato per consentire la creazione, la manipolazione e l'interrogazione di una o più basi di dati in modo corretto ed efficiente.

Debolezza

Condizione in un software o componente che può contribuire all'introduzione di Vulnerabilità.

Dev

Vedi Developer.

[2] ACID: https://database.guide/what-is-acid-in-databases/

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[3]:

- `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

[3] Documentazione: https://docs.docker.com/reference/dockerfile

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. Valore solo documentativo: 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

Repository, con sito accessibile al link https://byte-holders.github.io/Documentazione/, contenente tutti i documenti formali prodotti dal gruppo **Byte Holders**.

I documenti pubblicati:

- Analisi dei Requisiti
- Norme di Progetto
- Piano di Progetto
- Piano di Qualifica
- Glossario
- Verbali esterni
- Vebali interni
- Diari di bordo

E

F

Framework

Codice esterno che controlla il flusso del programma. Un framework permette all'utilizzatore di scrivere del codice all'interno di una struttura predefinita. Successivamente, il codice dell'utente verrà chiamato dal framework sfruttando l'*Inversion of Control*, ed eseguito secondo i suoi termini.
CodeGuardian utilizza NestJS come framework per il backend.

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 esistente in modo che possa usare 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.

Glossario

Questo documento, che ha lo scopo di disambiguare ed essere un riferimento condiviso all'interno del gruppo.

H

I

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/`.

Issue

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

J

JavaScript

Linguaggio ad alto livello, generalmente compilato *just-in-time*, dinamicamente tipato. Multi-paradigma, supporta programmazione orientata agli oggetti, agli eventi e funzionale. Nasce per rendere dinamico il frontend, successivamente utilizzato anche nel backend tramite NodeJS.

JSON

JavaScript Object Notation. Formato basato sulla sintassi di oggetti ed array di JavaScript. Può descrivere dati serializzati, anche complessi, come testo leggibile. Utilizzato da API REST.

K

L

Langchain

Framework open source per lo sviluppo di applicazioni basate su LLM. Consente l'esecuzioni di workflow semplici.

Langgraph

Framework open source per lo sviluppo di applicazione basate su LLM. Di basso livello, è concentrato principalmente sull'orchestrazione di agenti. I workflow sono compilati a partire da un grafo, personalizzabile.

Large Language Model

Tipologia di modello di *machine learning* addestrato per *Natural Language Processing*, particolarmente per la generazione di testo. Alla base dei chatbot.

LaTeX

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

LLM

Vedi Large Language Model.

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

Un Database non relazionale orientato ai documenti BSON (estensione del JSON, salva dati in binario). Permette anche la creazione di Database vettoriale.

N

NestJS

Framework per la costruzione del lato server di applicazioni che utilizzano NodeJS per il backend. Supporta sia JavaScript che TypeScript.

NodeJS

Sistema runtime orientato agli eventi per l'esecuzione di codice JavaScript lato server.

NoSQL

Altro nome per Database non relazionale.

Norme di Progetto

Documento che fornisce le linee guida per la documentazione, comunicazione, gestione delle versioni e le responsabilità all'interno del gruppo Byte Holders.

npm

node package manager. Gestore di pacchetti per NodeJS, permette di installare e gestire le dipendenze da riga di comando. Informazioni riguardo il *range* di versioni accettabile sono specificate all'interno del file `package.json`. In fase di installazione delle dipendenze, queste ottengono una versione "fissa" (quella effettivamente utilizzata), e vengono aggiunte all'interno del file `package-lock.json`. Questo rende le installazioni ripetibili pur se su sistemi diversi.

NVD

National Vulnerability Database. Un database americano che raccoglie e aggiunge informazioni (CVSS, CWE) alle CVE tramite i riferimenti e altri dati pubblici. I dati possono variare se vengono pubblicate informazioni rilevanti.

O

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

Documento che contiene le 10 categorie di maggior rischio per le applicazioni web. È redatto da OWASP, un'organizzazione no profit, aperta e non affiliata ad aziende tecnologiche. La OWASP Top 10 viene utilizzata dalle organizzazioni come standard per la sicurezza di applicazioni ed è basata sull'analisi di dati e sondaggi comunitari. A ciascuna categoria è associato:

- **CWE**: numero di CWE mappate alla categoria. È stato impostato dal team OWASP un limite di circa 40 CWE per ciascuna categoria per questione di praticità.
- **Tasso di incidenza**: percentuale di applicazioni con Vulnerabilità associate ad almeno una CWE mappata alla categoria, tra le applicazioni testate
- **Sfruttabilità (pesata)**: media pesata del campo *exploitability* del punteggio CVSS (v2 o v3) di CVE associato alle CWE della categoria, normalizzato in una scala da 0 a 10
- **Impatto (pesato)**: media pesata del campo *impact* del punteggio CVSS (v2 o v3) di CVE associato alle CWE della categoria, normalizzato in una scala da 0 a 10
- **Copertura**: percentuale di applicazioni testate contro cui sono stati effettuati test per una specifica CWE di categoria, rispetto al numero totale di applicazioni testate
- **Occorrenze totali**: numero di applicazioni che hanno presentato una o più CWE di categoria
- **CVE**: numero di CVE con un punteggio calcolato con CVSSv2 o CVSSv3 nel NVD mappate a CWE di categoria
- **Punteggio di rischio**[4]: calcolato utilizzando i dati soprastanti. Definisce la posizione nel ranking della Top 10

[4]Formula per il calcolo del punteggio del rischio OWASP: https://owasp.org/Top10/2025/0x02_2025-What_are_Application_Security_Risks/

Owner

Ruolo che fa riferimento al creatore di uno workspace. Gode degli stessi permessi di un Project Manager e non può essere rimosso dagli workspace che ha creato.

P

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, milestone 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

Documento che definisce la pianificazione temporale, l'allocazione delle risorse e le tempistiche per la realizzazione di CodeGuardian. Fornisce una guida strutturata per il controllo dell'avanzamento di progetto e la mitigazione dei rischi all'interno del ciclo di vita.

Piano di Qualifica

Documento di riferimento per il controllo continuo della qualità del prodotto e dei processi di ciclo di vita. Definisce strategie, standard e metriche per garantire la piena soddisfazione dei requisiti, nonché la correttezza e la buona qualità del prodotto.

PoC

Vedi Proof of Concept.

Proof of Concept

Semplice applicazione volta a dimostrare la fattibilità dell'obbiettivo prefissato. Utile a fissare una *baseline* tecnologica, quindi RTB. Le tecnologie utilizzate all'interno del Proof of Concept sono:

- NestJS
- React
- MongoDB
- Langgraph
- semgrep
- Syft
- GitHub REST API

Q

R

React

Libreria JavaScript per la creazione di interfacce utente. Utilizza un'architettura a componenti e permette di aggiornare il contenuto della pagina web in modo efficiente.

Report

Prodotto finale di un Sistema ad agenti. Comprende sezioni per:

- analisi statica del codice (semgrep)
- Code coverage
- analisi della documentazione (utilizzando un LLM dedicato)
- informazioni generali sul repo (GitHub REST API)
- dipendenze (Syft)

Requisito

Una condizione che un sistema o componente deve rispettare, soddisfacendo una specifica definita nell'Analisi dei Requisiti.

REST

REpresentational State Transfer. Stile architetturale che si basa sui seguenti principi:

- *interfaccia uniforme*: utilizza metodi HTTP standard (GET, POST, PUT, DELETE) e URI per le risorse
- *client-server*: impone la separazione dei compiti
- *stateless*: una richiesta deve contenere tutte le informazioni necessarie, e il server non deve contenere lo stato della sessione
- *cacheability*: le risposte devono indicare se possono essere messe in *cache*, diminuendo il carico lato server
- *sistema a strati*: il client non deve conoscere la struttura interna del server, ma solamente la specifica

S

SBOM

Inventario formale e strutturato dei componenti software che vengono utilizzati da un sistema.

Scenario

Uno Scenario è una specifica sequenza di eventi e azioni che descrive un singolo percorso di esecuzione all'interno di un Caso d'uso. Esso elenca i passi logici compiuti dagli attori e le risposte fornite dal sistema. Ogni caso d'uso comprende tipicamente uno Scenario Principale (il percorso ideale di successo) e uno o più Scenari Alternativi (che gestiscono varianti o condizioni di errore).

semgrep

Strumento di analisi statica del codice. Utilizza regole definite all'interno di file YAML per rilevare vulnerabilità di sicurezza e bug. Offre una *community edition* gratuita.

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.

Syft

Tool open source per la creazione di SBOM a partire da Immagine (Docker), filesystem e binari.

Sprint

Uno sprint è una piccola porzione di tempo (una o 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

SQL

Structured Query Language. Linguaggio utilizzato per interrogare un RDBMS (*Relational DataBase Management System*).

T

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

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.
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.

TypeScript

Linguaggio di programmazione ad alto livello fortemente tipato. Viene compilato a JavaScript tramite compilatori come *TypeScript compiler* e *Babel*.

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.

Vulnerabilità

Istanza di una Debolezza che può essere sfruttata, impattando negativamente la CIA.

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.

X

Y

Z