Verbale della Riunione

Atlas

team9.atlas@gmail.com

Data: 2025/10/20
Luogo: Chiamata Zoom
Versione: v1.0
Tipo: Esterno

Partecipanti

Nome	Presenza
Stefano Dindo	SI (**Var Group**)
Michele Massaro	SI (**Var Group**)
Andrea Difino	SI
Federico Simonetto	SI
Riccardo Valerio	NO
Francesco Marcolongo	SI
Michele Tesser	NO
Giacomo Giora	SI
Bilal Sabic	SI

Registro delle modifiche

Versione	**Data**	**Autore**	**Verificatore**	**Descrizione**
v1.0	2025/10/21	Giacomo Giora	Federico Simonetto	Sistemazione contenuto
v0.1	2025/10/20	Giacomo Giora		Prima stesura

Indice

1 Abstract **3**

2 Ordine del Giorno **3**

3 Domande e risposte **3**

3.1 Framework da utilizzare per l'architettura 3

3.2 Struttura e funzionalità della dashboard 3

3.3 Formato delle proposte di remediation 4

3.4 Metriche per l'analisi 4

3.5 Data layer 4

3.6 Demo live e documentazione tecnica 5

4 Conclusioni **5**

1 Abstract

In questo verbale vengono riportate le domande poste e le risposte ricevute durante il primo meeting effettuato dal team Atlas con l'azienda **Var Group**, riguardante il capitolato C2 del corso di Ingegneria del Sofware. L'incontro si è tenuto da remoto nel giorno 2025/10/20 dalle 16:30 alle 16:50.

2 Ordine del Giorno

- Chiarimenti sul capitolato
- Esposizione domande riguardanti il progetto

3 Domande e risposte

3.1 Framework da utilizzare per l'architettura

Domanda: Nel capitolato sono citati diversi framework per la creazione dell'architettura a multi agenti con un orchestratore centrale. Noi dovremo usare uno di quei framework o ci verrà fornito successivamente?

Risposta: L'obiettivo è creare un nostro framework. Attualmente esistono già strumenti in grado di effettuare test di unità e analisi statica. Noi vogliamo utilizzare l'AI per orchestrare questi strumenti in una procedura automatizzata di rilascio. Quindi ad esempio, quando arriva un commit, l'orchestratore attiva un agente dedicato all'analisi statica. Se è necessario effettuare un controllo di sicurezza, l'orchestratore attiva un altro agente (ad esempio un LLM specializzato). Ogni agente ha un compito specifico, e l'orchestratore si occupa di delegare e raccogliere i risultati.

3.2 Struttura e funzionalità della dashboard

Domanda: Nella presentazione del capitolato viene mostrato uno schema concettuale con una dashboard come componente finale. Essa come dovrà essere strutturata? Cosa dovrebbe permettere all'utente di poter fare? Si può visualizzare l'analisi solo delle proprie repository o anche di repository "generali" per esempio su una sezione trending come c'è su GitHub?

Risposta: La struttura della dashboard verrà definita più precisamente durante la sessione di design thinking. L'idea è di avere un processo di drill down, che permetta all'utente di partire da una visione d'insieme e con un click approfondire i dettagli tecnici di ogni singola repository. Le funzionalità previste sono: visualizzazione di linguaggi di programmazione e librerie utilizzate, analisi di livello di uniformità delle librerie usate nei vari progetti, possibilità di ottenere suggerimenti di miglioramento, ad esempio: se in un progetto è presente una libreria in versione 2.0 ma l'ultima versione è la 3.0, l'interfaccia suggerirà l'aggiornamento. e anche una dashboard progettata per essere interattiva e fornire indicazioni utili per ottimizzare la qualità del codice e la gestione dei progetti. Possiamo avere due tipi di utenti: l'utente standard che può visualizzare e analizzare solo le proprie repository e il superadmin, che può visualizzare e analizzare tutte le repository.

3.3 Formato delle proposte di remediation

Domanda: Le proposte di remediation devono essere fornite come report, per esempio come documento con i risultati dell'analisi, oppure in tempo reale per esempio come differenza visuale (diff) del codice, come avviene su GitHub?

Risposta: L'ideale sarebbe avere la possibilità di visualizzare le differenze nel codice (diff) direttamente nella piattaforma, permettendo così una visualizzazione più chiara e immediata dei suggerimenti di modifica, ma in fase di design thinking si definirà meglio se conviene concentrarsi su questa parte oppure dare priorità ad altri aspetti del progetto.

3.4 Metriche per l'analisi

Domanda: Nel capitolato è specificato che come metriche per superare l'analisi sulla sicurezza di un progetto si utilizza lo standard OWASP, cos'è precisamente questo standard? Per quanto riguarda l'analisi sul fatto che un progetto è aggiornato, documentato, e con test, ci verrebbero fornite anche lì delle metriche oggettive?

Risposta: Lo standard OWASP è un insieme di best practices e linee guida rivolte alla sicurezza del software che sono facilmente approfondibili nel loro sito ufficiale. Per quanto riguarda invece la parte relativa ad aggiornamenti, documentazione e test, non esiste uno standard analogo già definito. Sarà l'orchestratore a lanciare le verifiche in corrispondenza di un commit, per esempio tramite l'uso di un tag, e da lì far partire le attività di controllo. Verranno utilizzate metriche come una soglia minima di code coverage, la verifica della presenza di documentazione API tramite strumenti come Swagger, l'analisi delle firme e dei tipi di ritorno dei metodi, e la disponibilità di una documentazione architetturale che spieghi le scelte progettuali. L'obiettivo è avere un processo automatizzato e oggettivo che controlli sicurezza, aggiornamento e qualità del codice.

3.5 Data layer

Domanda: Come deve funzionare il data layer che comunica con l'orchestratore? Con che linguaggio deve essere implementato?

Risposta: Il data layer è lo strato in cui risiedono il database e il sistema di code, e ha il compito di gestire lo scambio di messaggi tra l'orchestratore e i vari agenti. Per esempio, quando un agente esegue un'analisi statica e un altro deve riceverne i risultati, è necessario un canale di comunicazione affidabile ed efficiente. Questo scambio può avvenire attraverso un sistema a code oppure tramite API intermedie che fungono da punto di passaggio per i dati. Non è richiesto un linguaggio specifico per la sua implementazione: la scelta dipenderà dal design complessivo e dagli strumenti che verranno adottati. L'aspetto più importante è garantire l'interoperabilità tra gli agenti e un'integrazione fluida con l'orchestratore.

3.6 Demo live e documentazione tecnica

Domanda: In cosa consiste la demo live? Ci viene spiegato come produrre la documentazione tecnica?

Risposta: Per la documentazione tecnica: il primo step sarà una sessione di design thinking, durante la quale verrà definito il perimetro e l'esigenza di business da realizzare. Successivamente verranno fornite indicazioni tecniche su come strutturare la documentazione. La collaborazione potrà avvenire tramite: accesso a una nostra repo dedicata, oppure una repo creata direttamente dai proponenti, oppure tramite strumenti di comunicazione come Teams. Le modalità operative non sono ancora state decise, anche perché i sistemi potrebbero essere piuttosto complessi per voi studenti. Invece per la demo live: si mostrerà la parte applicativa sviluppata, in particolare il cruscotto degli agenti. Durante la presentazione, verrà simulato un commit e sarà possibile vedere l'interazione tra tutti i sistemi. Quindi l'obiettivo della demo è dimostrare il funzionamento concreto dell'architettura multi-agente e della pipeline automatizzata.

4 Conclusioni

Sono state poste diverse domande riguardanti il capitolato C2 e sono state fornite risposte dettagliate dai rappresentanti dell'azienda Var Group. Le risposte sono state esaustive e hanno chiarito vari aspetti tecnici e funzionali del progetto, inclusi l'architettura multi-agente, la dashboard, le metriche di analisi, il data layer e le modalità di presentazione della demo live e della documentazione tecnica.

Approvazione dell'azienda
Il proponente,

_____