SIGMA n=18

Sigma18Unipd@gmail.com

Incontri di formazione con Var Group S.p.A.

Responsabile	Matteo Marangon
Redattore	Carmelo Russello
Verificatore	Marco Egidi

Versione	1.0.1
Tipo	Verbale Esterno
Destinatari	*Var Group S.p.A.* *Prof. Tullio Vardanega* *Prof. Riccardo Cardin*

Abstract dei contenuti:

Resoconto degli incontri di formazione con l'azienda Var Group S.p.A. svolti in data 06 maggio 2025, 08 maggio 2025, 13 maggio 2025, 19 maggio 2025 e 20 maggio 2025.

Indice

1. Riferimenti generali

Il seguente documento contiene il resoconto degli incontri di formazione del gruppo *Sigma18* proposti dall'azienda *Var Group S.p.A* sulle tecnologie ***AWS***$_{GL}$, ***Backend***$_{GL}$, ***Frontend***$_{GL}$ e ***Gen AI***$_{GL}$ Le sessioni sono state svolte in modalità mista, con incontri in presenza e incontri remoti.

- Presso la sede di Guizza (Padova) rispettivamente nelle date:
  - **6 maggio 2025**
  - **8 maggio 2025**
  - **13 maggio 2025**

della durata di **12 ore** totali.

- In modalità remota sulle tecnologie di modalità nelle date:
  - **19 maggio 2025**
  - **20 maggio 2025**

della durata di **6 ore** totali.

Nell'incontro sono stati trattati i punti descritti nella sezione *ordine del giorno*.

1.1. Partecipanti

1.1.1. Interni

- Alessandro Bernardello
- Mirco Borella
- Pietro Crotti
- Marco Egidi
- Matteo Marangon
- Aleena Mathew
- Carmelo Russello

1.1.2. Esterni

- Alessandro Baldissera (*Var Group S.p.A.*)
- Alessandro Dindinelli (*Var Group S.p.A.*)
- Francesco Battistella (*Var Group S.p.A.*)
- Stefano Sommariva (*Var Group S.p.A.*)
- Vittorio Corrizzato (*Var Group S.p.A.*)

2. Ordine del giorno

Durante gli incontri di formazione i dipendenti di *Var Group S.p.A.* sopracitati hanno presentato le tecnologie che l'azienda utilizza per lo sviluppo software, con particolare attenzione a ciò che può trovare applicazione nel progetto del gruppo *Sigma18*. Le spiegazioni sono state corredate da slide presentative ed esempi pratici, mostrando l'interfaccia web di alcuni applicativi o esempi di codice.

Le sessioni di formazione sono state organizzate in modo da coprire le seguenti aree tematiche:

2.1. *AWS*

La sessione di formazione su *AWS*, svoltasi il **6 maggio 2025** e tenuta dal referente aziendale Alessandro Dindinelli (Var Group S.p.A.), ha affrontato i seguenti argomenti:

2.1.1. Introduzione a *AWS*

È stata presentata una panoramica di *AWS* (Amazon Web Services), la piattaforma di servizi cloud di *Amazon*, che offre una vasta gamma di servizi per l'elaborazione, l'archiviazione, il networking e altro ancora.
Nello specifico si é discusso delle ***Regions and Availability Zones***, aree geografiche isolate in cui sono distribuiti i data center di *AWS* progettati per fornire alta disponibilità e ridurre la latenza in ogni parte del mondo e del concetto di ***Identity and Access Management (IAM)***, servizio base che consente di gestire gli accessi e le autorizzazioni degli utenti e dei servizi all'interno dell'ambiente *AWS*.

2.1.2. Panoramica dei servizi di *AWS*

È stata in seguito presentata una panoramica dei principali servizi di *AWS*, tra cui:

2.1.2.1. *Amazon EC2 (Elastic Compute Cloud)*

Un servizio di cloud computing che permette di avviare e gestire server virtuali, chiamati istanze, nel cloud.
Il servizio permette di affittare macchine virtuali sulle cui é possibile eseguire applicazioni, servizi web o altri carichi di lavoro. In base alle esigenze dell'utente è possibile scegliere le risorse di calcolo, memoria e *storage* necessarie per le varie istanze.

2.1.2.2. *Amazon DynamoDB*

Un servizio di database *NoSQL serverless* che consente di sviluppare applicazioni moderne su qualsiasi scala. Le caratteristiche messe in evidenza dall'azienda proponente sono la sua natura *on demand*, che consente di pagare solo per le risorse effettivamente utilizzate, l'ampia gamma di controlli di sicurezza che offre e la sua integrazione con altri servizi di *AWS*.

2.1.2.3. *Amazon S3*

Servizio di archiviazione di oggetti che consente di archiviare e recuperare qualsiasi quantità di dati in modo semplice e scalabile.
Il servizio è stato presentato come una soluzione ideale per l'archiviazione dei dati in modo scalabile e per le sue opzioni di sicurezza come il *backup* e la gestione dei disastri che garantiscono la protezione dei dati in caso di guasti o errori umani.

Per l'archiviazione dei dati (oggetti) nel cloud é necessaria la creazione di *bucket* (contenitori), con nome univoco per ogni regione di *AWS*.

2.1.2.4. *AWS Lambda*

Un servizio di calcolo *serverless* che consente agli sviluppatori di eseguire codice in risposta a eventi senza dover gestire l'infrastruttura sottostante.
La sua natura *on demand*, la sua semplicità di utilizzo e la sua integrazione con altri servizi di *AWS* sono state evidenziate come caratteristiche principali.
Un inconveniente messo in evidenza dal referente è il suo tempo di esecuzione limitato (15 minuti), che può essere un problema per alcune applicazioni che richiedono un'elaborazione prolungata.

2.2. *Backend*

La sessione di formazione sul *Backend*, svoltasi l'**8 maggio 2025** e tenuta dal referente aziendale Alessandro Baldissera (Var Group S.p.A.), ha affrontato i seguenti argomenti:

2.2.1. *Container* e *Docker*

È stata presentata una panoramica dei **container**, un metodo per impacchettare un'applicazione e le sue dipendenze in un'unità standardizzata che può essere eseguita in qualsiasi ambiente.
I container sono isolati l'uno dall'altro e dal sistema operativo sottostante, inoltre possono essere costruiti in modo ripetibile per eseguire qualsiasi *software*.

È stato introdotto il concetto di ***Docker***, una piattaforma *open source* che consente agli sviluppatori di creare, implementare, eseguire, aggiornare e gestire i container.
I container *Docker* vengono descritti dichiarando un file di configurazione chiamato ***Dockerfile*** che contiene le istruzioni per costruire l'ambiente in cui verrà eseguita l'applicazione.

2.2.2. *Rest*

Rest è uno stile architetturale per la progettazione di servizi web che definisce le modalità di interazione tra client e server per lo scambio di informazioni. È impiegato principalmente per la realizzazione di ***API (Application Programming Interface)***$_{GL}$.

Rest è stato consigliato dalla azienda proponente per la sua semplicità, scalabilità, flessibilità (supporta vari formati di dati), performance e, soprattutto, per la sua ampia diffusione, che facilita l'accesso a risorse e supporto tecnico.

2.2.3. *Environment Management*

È stato illustrato il concetto di ***Environment Management***, un approccio che consente di gestire gli ambienti di sviluppo tramite dei parametri tramite un file di configurazione *.env* che contiene le variabili di ambiente necessarie per l'esecuzione dell'applicazione.
È stato spiegato che le variabili di ambiente possono essere utilizzate per configurare l'applicazione in modo dinamico, ad esempio per specificare le credenziali di accesso al database o le chiavi API.

2.2.4. *NestJS*$_{GL}$

NestJS è un *framework* per lo sviluppo di applicazioni *Node.js* basato su *Typescript* utilizzato soprattutto per realizzare *API* e per la costruzione di applicazioni complesse che richiedono un'architettura scalabile e manutenibile.

È stato presentato come un *framework* dotato di una vasta *community* e di un ampio ecosistema, che facilitano la ricerca di risorse per approfondire le proprie conoscenze sull'argomento.

Il referente ha reso il gruppo consapevole che, nell'utilizzo di *NestJS*, é necessaria una conoscenza estesa di altri strumenti e librerie; inoltre, ha evidenziato come la sua configurazione possa diventare complessa su larga scala.

2.2.5. *MongoDB*

MongoDB è un database *NoSQL* (non relazionale) orientato ai documenti e facilmente scalabile.

È stato spiegato come un database in *MongoDB* segue tre concetti principali:
- ***Database***: un contenitore per uno o più collezioni.
- ***Collezione***: Un insieme di documenti
- ***Documento***: Un insieme di coppie chiave-valore identificati da una proprietà speciale *id*, simile a un oggetto *JSON*.

MongoDB è stato consigliato dalla azienda proponente per il suo ricco linguaggio di *query*, la sua scalabilità e per la sua flessibilità nella definizione della struttura dei documenti.

2.3. *Frontend*

La sessione di formazione sul *Frontend*, svoltasi il **13 maggio 2025** e tenuta dal referente aziendale Vittorio Corrizzato (Var Group S.p.A.), ha affrontato i seguenti argomenti:

2.3.1. Introduzione a $React_{GL}$

È stato presentata una panoramica della libreria ***React***, una libreria *JavaScript* creata dall'azienda *Meta* per lo sviluppo di interfacce utente.
Sono stati inoltre illustrati i prerequisiti utili per l'utilizzo di *React*, come una conoscenza di base di progettazione di pagine *web* (*HTML* e *CSS*) e una familiarità con *Typescript*, un linguaggio che estende la sintassi di *JavaScript* .

2.3.2. Concetti chiave di *React*

Sono stati presentati i concetti chiave di *React*, tra cui:

2.3.2.1. *DOM*

***DOM* (*Document Object Model*)**, un'*API* per i documenti *HTML* e *XML*, definisce la struttura logica dei documenti e di come possono essere manipolati.

il principale problema relativo al *DOM* consiste nel fatto che la maggior parte dei framework *JavaScript* tendono ad aggiornarlo più del necessario, con conseguente rallentamento delle prestazioni delle applicazioni web.

È stato introdotto il concetto di ***Virtual DOM***, una rappresentazione in memoria del *DOM* che *React* utilizza ogni volta che viene renderizzato un elemento *JavaScript* comparandolo con il *DOM* reale per applicare solo le modifiche necessarie, migliorando così le prestazioni in modo significativo.

2.3.2.2. *JSX* e Componenti

Il referente ha introdotto ***JSX***, un'estensione della sintassi *JavaScript* usata per descrivere l'aspetto dell'interfaccia utente

Per evitare di dividere in modo artificiale il codice vengono creati degli elementi debolmente accoppiati chiamati **componenti**, degli elementi *JavaScript* riutilizzabili che accettano dati in input e ritornano elementi *React* che descrivono ciò che dovrebbe apparire a schermo.

È stato spiegato che i componenti possono essere di due tipi: le classi componente e le funzioni componente.

- Le **classi componente** sono definiti come classi *JavaScript*, possono gestire uno stato interno (*stateful*), reagire agli eventi del ciclo di vita e non supportano l'uso degli *hooks*.
- Le **funzioni componente** sono funzioni *JavaScript* che non hanno uno stato (*stateless*), accettano tutti i tipi di dato restituendo elementi *HTML* e supportano gli *hooks*.

il modo migliore per scrivere componenti in *React* è utilizzare *JSX*, che permette di scrivere codice simile a *HTML* all'interno del codice in *JavaScript*.

2.3.3. Hooks in *React*

L'ultimo argomento presentato in questa sessione è stato quello degli *hooks*, funzioni speciali che permettono di "ancorarsi" allo stato e al ciclo di vita dei componenti *React* e di utilizzare lo stato e altre funzionalità di *React* senza dover scrivere una classe.
Gli *hooks* hanno le seguenti caratteristiche:

- Gli hooks sono retrocompatibili e permettono di:
  - Riutilizzare logica *stateful* senza dover cambiare la gerarchia dei componenti
  - Dividere un componente in funzioni più piccole basate sui pezzi che sono correlati
  - Utilizzare più funzioni di *React* senza dover ricorrere alle classi
- Gli *hooks* seguono delle regole generali:
  - Gli hooks vanno richiamati al livello più alto. Non possono essere richiamati all'interno di cicli, condizioni o funzioni nidificate
  - Gli hooks possono essere richiamati solo da funzioni componente di React

2.4. *Gen AI*

Le sessioni di formazione su *Gen AI*, svolte negli incontri del **19 maggio 2025** e **20 maggio 2025**, tenute dal referente aziendale Stefano Sommariva (*Var Group S.p.A.*), hanno affrontato i seguenti argomenti:

2.4.1. Introduzione a *Gen AI*

La sessione di formazione su *Gen AI* é iniziata con una introduzione generale sull'intelligenza artificiale generativa, concentrandosi su due concetti chiave:

2.4.1.1. Reti neurali

Sono modelli di intelligenza artificiale ispirati al funzionamento del cervello umano tramite dei nodi interconnessi fra di loro.
Il referente ha spiegato come le reti neurali sono composte da tre strati:

- **Strato di *input***: ogni nodo corrisponde a una e una sola caratteristica dell'input

- **Strati nascosti**: ogni nodo rappresenta una combinazione di caratteristiche provenienti dallo strato precedente
- **Strato di *output***: il numero di nodi corrisponde al numero di possibili valori di output per la rete

2.4.1.2. *LLM (Large Language Model)*$_{GL}$

Le applicazioni di intelligenza artificiale generativa si fondano sul concetto di *LLM*, dei modelli neurali avanzati che riescono a generare e comprendere il linguaggio naturale. I *LLM* sono addestrati su quantità enormi di dati, il che li rende estremamente efficaci ma anche molto costosi dal punto di vista computazionale.

È stato illustrato il funzionamento dell'architettura *Transformer*, architettura su cui si basano i *LLM*, che si basa su due componenti principali:

- ***Encoder***: una rete neurale che converte le parole in linguaggio naturale in rappresentazioni numeriche, permettendo al modello di elaborarle.
- ***Decoder***: una rete neurale che utilizza le rappresentazioni numeriche fornite dall'*encoder* per generare nuove parole in linguaggio naturale.

2.4.2. *OpenAI*

Il gruppo ha ricevuto una breve introduzione a *OpenAI*, in particolare sui modelli della famiglia *GPT*.

Particolare attenzione é stata posta sui modelli specializzati per diverse applicazioni come *Dall-E* per la generazione di immagini e *Whisper* per la trascrizione automatica del parlato.

La sezione di *OpenAI* si é conclusa con una panoramica sulla partnership con *Microsoft*, che ha portato all'integrazione dei modelli di *OpenAI* in prodotti come *Azure OpenAI* e *Copilot*.

2.4.3. *Azure OpenAI*

È stato presentato il servizio *Azure OpenAI*, una piattaforma che consente di accedere ai modelli di intelligenza artificiale generativa di *OpenAI* tramite l'infrastruttura cloud di *Microsoft Azure*.

L'azienda ha elencato alcuni concetti chiave di *Azure OpenAI*:

2.4.4. *Tokens*

I ***tokens*** sono sequenze comuni di caratteri con i quali i modelli di *Azure OpenAI* leggono e interpretano il testo.

I token possono essere singoli caratteri, una parola o una parte di una parola, il ruolo dei modelli di *Azure OpenAI* è quello di imparare le relazioni statistiche tra i diversi *tokens* per generare il *token* successivo in sequenza.

2.4.5. *Prompts*

I ***prompts*** rappresentano le richieste effettuate in *input* al modello.

È stato puntualizzato come la loro progettazione sia fondamentale per ottenere risposte precise e pertinenti; questo processo di progettazione é detto ingegneria del *prompt* caratterizzato da un attento studio delle istruzioni e del contesto fornito al modello per ottenere risposte ottimali.

2.4.6. *Endpoints*

Gli *endpoints* consentono di interagire con i modelli di *OpenAI*, permettendo di ricevere un prompt in input e di ottenere in output una risposta generata dal modello.
È stato mostrata l'esistenza di tre tipologie di *endpoints*:

- ***Completions***: utilizzati per generare testo in risposta a un prompt.
- ***ChatCompletion***: utilizzati per gestire conversazioni in linguaggio naturale, consentendo interazioni più complesse e contestualizzate.
- ***Embeddings***: utilizzati per generare rappresentazioni numeriche di testo, utili per compiti come la ricerca semantica e la classificazione del testo.

3. Conclusioni

Tutto il gruppo *Sigma18* ringrazia l'azienda per il tempo fornito e per la loro disponibilità.
L'azienda si è resa disponibile a fornire supporto e chiarimenti in caso di necessità.

Firma dell'azienda