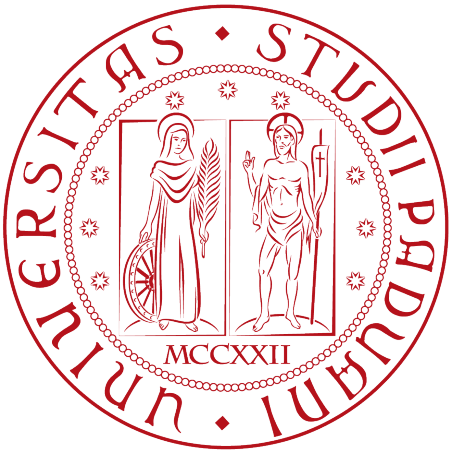
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



SportAI: un assistente *chatbot* per lo sport

Tesi di Laurea triennale

Relatore
Prof. Vardanega Tullio

Laureando
Ferraioli Francesco
Matricola 2003606

Anno Accademico 2023/2024

"Per Aspera ad Astra"

— Lucio Anneo Seneca.

Ringraziamenti

Desidero innanzitutto esprimere la mia più sincera gratitudine al Prof. Vardanega Tullio, tutor del mio tirocinio curricolare e relatore della mia tesi di laurea, per il costante supporto e la disponibilità dimostrati durante il corso "Ingegneria del Software" da lui tenuto, nonché durante il periodo di tirocinio e la stesura della tesi.

Un ringraziamento speciale va alla mia famiglia: Gerardo, Giovanna e Marilena, e tutti i miei parenti per il loro continuo sostegno e per la fiducia che hanno sempre riposto in me.

Ringrazio il team Wavelop per avermi accolto durante il tirocinio in un ambiente sereno e stimolante.

Ringrazio il team Coding Cowboys, gruppo di progetto di Ingegneria del Software, per il sostegno e l'impegno mostrato durante la realizzazione del progetto più impegnativo del corso di studi.

Infine, ringrazio di cuore i miei amici e compagni di studi per i momenti di crescita condivisi, e per i legami che abbiamo costruito durante questi anni di studio.

Padova, Luglio 2024 Ferraioli Francesco

Sommario

Questo documento descrive i processi, gli strumenti e le metodologie impiegate nello sviluppo di una *Web App*[1], ovvero un'applicazione accessibile via *web*, progettata per la creazione di schede di allenamento per il fitness e per lo sport.

Struttura del testo

Il corpo del documento è suddiviso in quattro capitoli principali:

Primo capitolo delinea il contesto di svolgimento del tirocinio curricolare, concludendo con una riflessione sul rapporto tra l'azienda ospitante e l'innovazione nei processi e strumenti aziendali.

Secondo capitolo spiega le motivazioni che hanno portato le esigenze mie e dell'azienda ospitante a convergere in un progetto vantaggioso per entrambe le parti.

Terzo capitolo descrive i processi, gli strumenti, le tecnologie e le modalità di esecuzione delle attività di tirocinio, nonché i risultati ottenuti.

Quarto capitolo offre una retrospettiva sul tirocinio, evidenziando le competenze acquisite durante il percorso e quelle utilizzate derivanti dal corso di studi.

Alla fine del documento si trovano le seguenti sezioni:

- **Acronimi e abbreviazioni**: contiene i collegamenti ai relativi termini nel **Glossario**.
- **Glossario**: fornisce le definizioni dei termini specifici del settore, con i collegamenti ai relativi acronimi o abbreviazioni (se presenti). Dopo ogni definizione, sono disponibili collegamenti alle pagine in cui tali termini sono utilizzati.
- **Bibliografia**: elenca le fonti delle informazioni utilizzate per definire concetti, indicando eventuali *link*, porzioni di testo in cui sono state citate e termini ai quali si riferiscono.

[1] *Web App*

Convenzioni tipografiche

Per la stesura del testo, sono state adottate le seguenti convenzioni tipografiche:

- Gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune sono definiti nel capitolo **Glossario**:
  - Al primo utilizzo di uno di questi termini, verrà fornita una breve definizione.
  - Solo il primo utilizzo di tali termini sarà accompagnato da una nota a piè di pagina contenente il riferimento al termine nel capitolo **Glossario**.
- Stile *corsivo* per i termini in lingua straniera, nomi propri e termini tecnici.
- Stile **grassetto** per il nome dell'azienda ospitante del tirocinio, i nomi dei capitoli del documento e i termini chiave delle attività di tirocinio.
- Stile **grassetto** per le parole rilevanti in sezioni di testo ampie.
- Stile **grassetto** per i termini che definiscono da soli le voci di un elenco (sia esso numerato o puntato).
- Colore rosso per ogni collegamento a pagine *web*.
- Le fonti delle immagini sono inserite come nota a piè di pagina, contenente il *link* della pagina *web* principale di appartenenza.

Indice

1 Contesto di svolgimento delle attività **1**
- 1.1 Introduzione all'azienda ospitante . . . 1
- 1.2 Prodotti e servizi . . . 2
- 1.3 Processi interni . . . 3
- 1.4 Rapporto con l'innovazione . . . 6

2 Motivazioni alla base del tirocinio **8**
- 2.1 Strategia aziendale . . . 8
- 2.2 Sfide tecnologiche . . . 9
- 2.3 Obiettivi . . . 11
- 2.4 Vincoli . . . 12
- 2.5 Pianificazione . . . 12
- 2.6 Scelta del tirocinio . . . 14

3 Elementi caratterizzanti del progetto **17**
- 3.1 Stile lavorativo . . . 17
- 3.2 Strumenti utilizzati . . . 18
  - 3.2.1 Strumenti di sviluppo . . . 18
  - 3.2.2 Strumenti di versionamento . . . 23
  - 3.2.3 Strumenti di documentazione . . . 23
- 3.3 Analisi dei requisiti . . . 24
  - 3.3.1 Casi d'uso . . . 24
- 3.4 Progettazione . . . 27
  - 3.4.1 LLM a confronto . . . 27
  - 3.4.2 Scelta del database per RAG . . . 30
  - 3.4.3 Architettura del sistema . . . 31
- 3.5 Codifica . . . 34
  - 3.5.1 Problematiche riscontrate . . . 35
  - 3.5.2 Strumenti utilizzati . . . 36
  - 3.5.3 Best practices seguite . . . 36
- 3.6 Verifica . . . 37
- 3.7 Validazione . . . 38
- 3.8 Risultato finale . . . 39
  - 3.8.1 Statistiche qualitative e quantitative . . . 39

4 Retrospettiva delle attività **42**
- 4.1 Raggiungimento degli obiettivi prefissati . . . 42

4.1.1 Obiettivi aziendali 42
4.1.2 Obiettivi personali 43
4.2 Competenze e conoscenze acquisite 44
4.3 Competenze curricolari e lavorative 45

Acronimi e abbreviazioni **46**

Glossario **47**

Bibliografia **50**

Elenco delle figure

1.1 Sito internet di *Wavelop* 1
1.2 Web App sviluppata da *Wavelop* 2
1.3 Attività di sviluppo con metodologia agile 5
1.4 Corso di formazione al quale parteciperà il *team Wavelop* 6

2.1 Schema sintesi di una rete neurale alla base di un *LLM* 9
2.2 Schema sintesi del concetto di *black-box* 10
2.3 Diagramma di *Gantt* di suddivisione delle attività 14
2.4 Pianificazione interfaccia del progetto *SportAI* 15

3.1 Diagramma di collegamento delle tecnologie *frontend* 19
3.2 Schema di funzionamento di *LangChain* 20
3.3 Schema di funzionamento di *ChromaDb* 21
3.4 Diagramma di collegamento delle tecnologie *backend* 22
3.5 Diagramma di collegamento delle tecnologie di supporto 23
3.6 Diagramma dei casi d'uso UC-8 e UC-9. 25
3.7 Diagramma del caso d'uso UC-10. 26
3.8 Schema di funzionamento di *SportAI* 27
3.9 Piano offerto da *Google* per *Gemini 1.5 Flash* 28
3.10 Modello di sviluppo del prodotto adottato 38
3.11 Pagina di *Login* di *SportAI* 40
3.12 Pagina di *Chat* di *SportAI* 40

Elenco delle tabelle

2.1 Consuntivo delle ore di lavoro 14

3.1 Tabella riassuntiva dei requisiti soddisfatti 39

3.2 Statistiche quantitative sul prodotto 39

4.2 Obiettivi di tirocinio - desiderabili 43
4.3 Obiettivi di tirocinio - facoltativi 43
4.4 Obiettivi di tirocinio - personali 44

Capitolo 1

Contesto di svolgimento delle attività

Questo capitolo introduce ***Wavelop***, azienda ospitante del tirocinio, e fornisce informazioni relative al suo settore lavorativo, al suo rapporto con l'innovazione e con il miglioramento di processi e strumenti già in uso ed ai processi in essa utilizzati. Le informazioni riportate sono soggettive basate sulla esperienza personale.

1.1 Introduzione all'azienda ospitante

Wavelop è una *software house*[2], ovvero un'azienda che si occupa dello sviluppo di *software*, specializzata nella consulenza e nello sviluppo di soluzioni personalizzate.

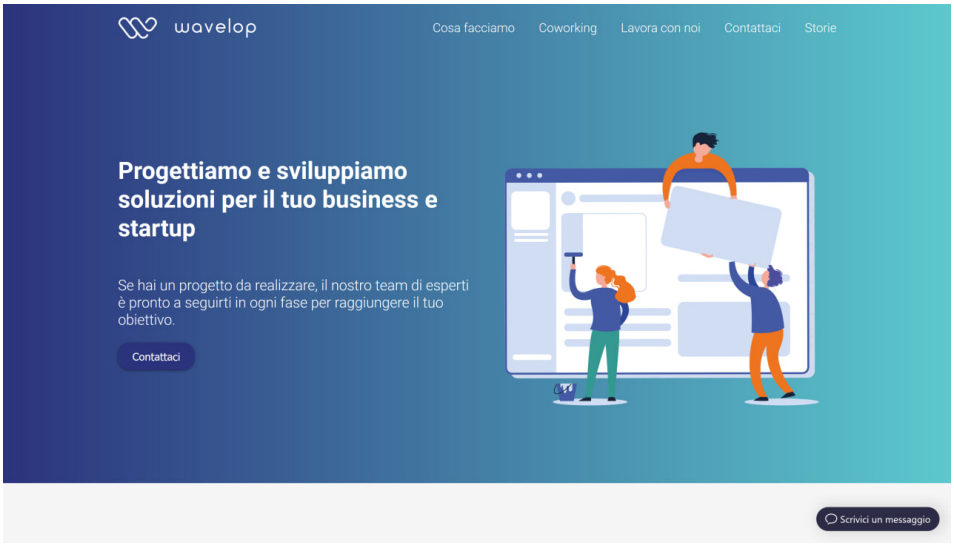


Figura 1.1: Sito internet di ***Wavelop***[3]

L'azienda, fondata nel 2018, è ubicata nel centro di *Treviso*,
dispone all'incirca di una decina di dipendenti *IT*[4] (informatici) tra i 20 e i 30 anni, creando un ambiente molto giovane e dinamico.

[2] *Software house*
[3] Fonte: https://wavelop.com/it/
[4] *Information Technology* (IT)

1.2 Prodotti e servizi

Wavelop offre servizi di consulenza e sviluppo di soluzioni *web* e *mobile* personalizzate per i propri clienti. Infatti, tutti i prodotti realizzati dall'azienda sono creati ad *hoc* per adempiere appieno alle reali necessità dei clienti.
I prodotti e i servizi offerti dall'azienda sono:

- ***Web development***: realizzazione di siti internet personalizzati per i clienti, tra cui anche siti di *e-commerce b2b*[5], curando sia la parte di *frontend*[6], mediante tecnologie come *Javascript, Angular, React*, sia quella di *backend*[7], utilizzando tecnologie come *MongoDB, Node.js, Fastify*;
- ***Mobile application development***: realizzazione di applicazioni *web* personalizzate per i clienti e ibride, cioè fruibili sia da *Android* che da *IOS*, mediante tecnologie come *React Native* e *Ionic*, che permettono la distribuzione su entrambe le piattaforme senza necessità di sviluppare due applicazioni separate;
- ***IoT***[8] ***application development***: realizzazione di applicazioni per dispositivi *IoT* personalizzate, come collegamenti di sensoristica industriale per aggiornare le industrie ai nuovi standard di industria 4.0 e assistenti vocali come *Amazon Alexa* per gli usi privati dei clienti.

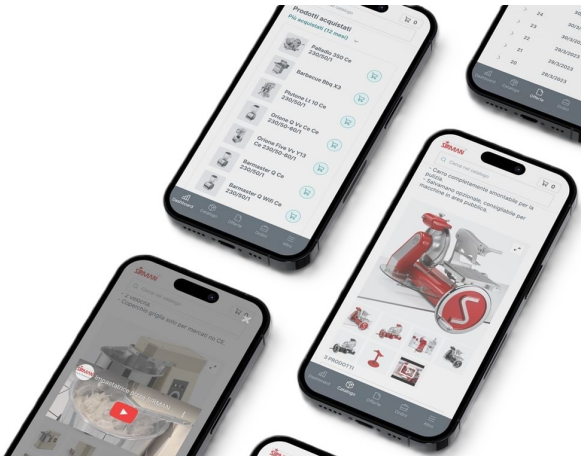


Figura 1.2: Web app sviluppata da ***Wavelop***

La clientela a cui ***Wavelop*** si rivolge è composta da piccole e medie imprese, start-up e privati che necessitano di soluzioni *software* personalizzate per le proprie esigenze. Così facendo, l'azienda si distingue per la qualità dei prodotti e dei servizi offerti, per la flessibilità e la personalizzazione delle soluzioni proposte e per la capacità di adattarsi alle esigenze dei clienti, a differenza dei prodotti generalisti realizzati da studi più grandi.

[5] E-commerce
[6] *Frontend*
[7] *Backend*
[8] IoT

1.3 Processi interni

Il modello di lavoro per la realizzazione dei progetti dei quali ***Wavelop*** viene incaricata è detto *agile*[9], cioè un metodo di pianificazione e di esecuzione orientato all'ottimizzazione del flusso di lavoro, evitando tempi morti e consentendo una risposta rapida alle variazioni delle esigenze del cliente anche in stadi avanzati dello sviluppo.

- **Gestione di progetto**: per quanto concerne la gestione di progetto, ho potuto assistere e partecipare alle seguenti attività:
  - **Definizione degli obiettivi e delle risorse**: è una discussione con il cliente per definire i requisiti del progetto e le risorse necessarie per portarlo a termine. Inoltre, si determinano anche vincoli temporali e di budget;
  - **Pianificazione**: avviene a partire dalla definizione degli obiettivi e delle risorse di progetto, basandosi su esperienze pregresse e sulla disponibilità di capitale umano e risorse economiche. Si dividono le varie attività da svolgere e si assegnano priorità alle varie fasi del progetto, distribuendo il lavoro tra i membri del *team* per ottenere un valore di *business* rapidamente, sfruttando anche la possibilità di svolgere le attività in parallelo;
  - **Comunicazione con gli *stakeholders***[10]: la comunicazione con le parti interessate, ovvero coloro che hanno un interesse diretto nel successo del progetto, avviene attraverso colloqui e rappresenta un elemento fondamentale per dimostrare un progresso tangibile in conformità con i tempi e le modalità stabilite, o per giustificare eventuali scostamenti dalla pianificazione.
- **Sviluppo**: per quanto concerne il ciclo di un progetto, ho potuto assistere e partecipare alle seguenti fasi:
  - **Analisi dei requisiti**: la prima fase del metodo *agile* consiste nell'analisi dei requisiti, nella quale i bisogni espressi dai clienti vengono trasposti prima in casi d'uso mediante l'uso di grafici *UML*[11] in modo da avere una visione generale delle funzionalità e degli attori coinvolti nel progetto, e successivamente in *user-stories*, cioè delle descrizioni del comportamento o della funzionalità attesa da ogni sezione del *software* da parte del cliente. Ogni *story* è composta da un titolo, che permette di identificarla all'interno del sistema di gestione dei *ticket*.
    Un punteggio chiamato *story-points* che indica la complessità della *story* per facilitare la distribuzione del carico di lavoro tra i membri del *team*.
    Una descrizione dettagliata della funzionalità attesa con relativi sottopunti per rendere divisibile il lavoro;

[9] Agile
[10] *Stakeholder*
[11] UML

  - **Progettazione**: è la fase successiva all'analisi dei requisiti e consiste nel definire come il prodotto dovrà essere realizzato, in termini di architettura, tecnologie e librerie da utilizzare. In questa fase lo sviluppatore incaricato esegue anche delle ricerche ed effettua esperimenti necessari per comprendere se le tecnologie individuate siano adeguate. Infine, se necessario, stila un documento di analisi dei costi/benefici per valutare se la soluzione proposta sia conveniente per il cliente;
  - **Codifica**: è la fase in cui si scrive il codice sorgente del prodotto, seguendo le linee guida e l'architettura definite nella fase di progettazione.
    Il codice viene prodotto mediante l'uso di *IDE* e relativi *plugin* per facilitare la scrittura e la manutenzione del codice. Inoltre, si utilizzano strumenti di *linting* per controllare che il codice scritto rispetti le norme di codifica stabilite dall'azienda;
  - **Verifica**: è la fase in cui si controlla che il codice scritto rispetti le linee guida e le norme di codifica stabilite dall'azienda e che il prodotto funzioni correttamente secondo i requisiti definiti nella fase di analisi.
    Questa fase viene eseguita mediante test automatici e manuali. Inoltre, si utilizzano strumenti di *code-review* per controllare che il codice scritto sia corretto e rispetti le norme;
  - **Validazione**: è la fase in cui si verifica che la funzionalità sviluppata soddisfi i requisiti definiti nella fase di analisi e che sia pronta per essere rilasciata nel progetto corrente;
  - **Manutenzione**: è la fase in cui si correggono eventuali errori o bug riscontrati mediante un sistema di *ticketing* con il quale il cliente, o uno sviluppatore, segnala i problemi del prodotto.
    Inoltre, in questa fase si prendono decisioni per quanto concerne l'evoluzione del progetto in base a nuove esigenze o a nuove tecnologie disponibili.

- **Tecnologie**: per quanto concerne il ciclo di un progetto, si usano le seguenti tecnologie:

  - **GitLab**: per la gestione del codice sorgente e delle *pipeline*[12] di *CI/CD*[13]. *GitLab* è fondamentale in numerose fasi del progetto, infatti tramite la funzionalità di *issue tracking* è possibile tenere traccia dello stato dei *ticket*. Inoltre, fornisce un sistema di versionamento del codice sorgente e di *merge-request* per incorporare le modifiche apportate da altri membri del *team*;
  - **Diagrammi UML**: per la progettazione del prodotto, fornisce un metodo rapido per consultare le funzionalità attese dal prodotto mediante l'uso di riferimenti grafici piuttosto che letterali;
  - **Discord**: per la comunicazione interna tra i membri del *team* e per tenere i giornalieri *stand up meeting*[14].

[12] Pipeline
[13] *Continuous Integration/Continuous Delivery* (CI/CD)
[14] Stand-up meeting

Discord, infatti è una piattaforma di comunicazione che permette di creare canali di comunicazione testuali e vocali per discutere di problemi e soluzioni in modo rapido e diretto;

- **Google Meet**: è una infrastruttura semplice da usare, ben consolidata e professionale che permette di tenere delle riunioni con i clienti;
- **Google Drive**: per la condivisione di documenti e *file* riguardanti metodologie di lavoro, strumenti da utilizzare e pratiche aziendali, tra i membri del *team*;
- **Clockify**: per la gestione del tempo di lavoro e la fatturazione.
  Questo strumento è fondamentale per una azienda che fa dello *smart-working* e della flessibilità dei propri dipendenti un punto di forza; infatti permette di tenere traccia delle ore lavorate e di fatturare in modo trasparente e preciso.

L'utilizzo di queste tecnologie permette di mantenere un flusso di lavoro continuo e di avere un controllo costante sullo stato di avanzamento del progetto, garantendo così un lavoro di tipo *agile* e facilitando le attività di rendicontazione.

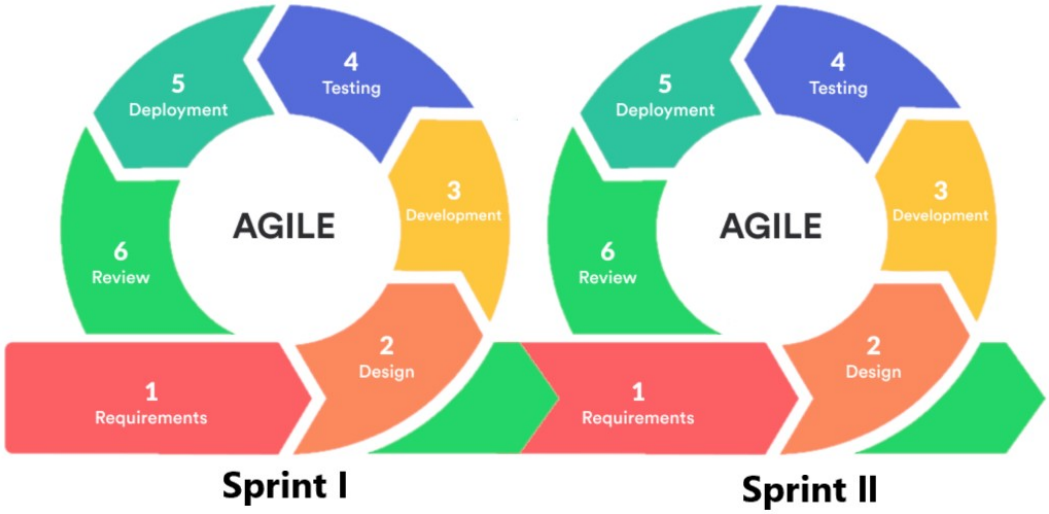


Figura 1.3: Attività di sviluppo con metodologia agile[15]

[15] Fonte: https://indevlab.com

1.4 Rapporto con l'innovazione

In base alle mie osservazioni durante l'esperienza di tirocinio, il rapporto con l'innovazione[16] (ovvero l'introduzione di novità e il miglioramento di processi e tecnologie impiegati) dell'azienda è costituito da:

- Formazione: l'azienda partecipa a corsi di formazione e seminari per tenersi aggiornata sulle nuove tecnologie e metodologie di sviluppo *software*.
  Questo permette di acquisire nuove competenze, di migliorare i processi interni e di offrire ai clienti soluzioni innovative e all'avanguardia;
- Sperimentazione: prima di iniziare un progetto, gli sviluppatori interessati nel progetto sono incoraggiati a sperimentare nuove tecnologie e strumenti per valutarne l'efficacia e l'efficienza per usi futuri.

La sperimentazione è fortemente sostenuta anche dai tirocini curricolari: il tempo a disposizione dei tirocinanti è usato anche per eseguire operazioni di ricerca e prototipazione, valutando più o meno in profondità strumenti da poter adottare per futuri progetti e acquisendo un minimo di esperienza nell'ambito. Nel caso specifico, ho realizzato un documento di analisi dei costi e benefici per valutare l'adozione delle tecnologie implementate nel progetto di stage.

Figura 1.4: Corso di formazione al quale parteciperà il *team* ***Wavelop***

[16] Innovazione

Capitolo 2

Motivazioni alla base del tirocinio

Questo capitolo discute le motivazioni che hanno portato alla scelta di svolgere il tirocinio curricolare, dal punto di vista dell'azienda ospitante e dal mio punto di vista, descrivendo gli obiettivi e le esigenze che il progetto punta a soddisfare.

2.1 Strategia aziendale

Da ciò che ho potuto osservare e da ciò che mi è stato riferito durante il periodo di *stage*, la strategia di gestione dei tirocini dell'azienda persegue i seguenti obiettivi:

- **Ricerca**: come riportato nella sezione §1.4, l'azienda ha modo di effettuare sperimentazione, prototipazione e ricerca di nuove tecnologie anche senza avere un cliente che le richiede, a basso costo sia orario che economico grazie al lavoro dei tirocinanti, i quali non essendo dipendenti, hanno la possibilità di lavorare con i propri tempi ed effettuare ricerche senza avere il rischio di creare ritardi alla produzione. Ciò permette all'azienda di poter offrire in futuro servizi innovativi a partire da quelle che sono state le sperimentazioni effettuate dai tirocinanti;

- **Creazione di nuovi prodotti**: i prodotti realizzati durante le attività di tirocinio, nonostante possano risultare grezzi o incompleti, hanno comunque un valore di business, per cui possono essere utilizzati e sistemati dall'azienda nel caso un cliente richieda quell'applicazione specifica, in modo da poter risparmiare;

- **Valutazione delle competenze**: il periodo di tirocinio è il periodo fondamentale nel quale uno studente si avvicina al contesto aziendale e più generalmente al mondo del lavoro: In questo periodo infatti ho compreso come effettivamente funziona un'azienda di sviluppo software, quali processi si seguono, la visione aziendale e che con che metodologie si lavora.
  L'azienda ospitante fornisce una formazione di base e valuta quali sono le reazioni alle difficoltà, alle sfide e agli stimoli in funzione dell'inserimento del tirocinante nel *team* aziendale, in ottica di possibile assunzione.

2.2 Sfide tecnologiche

Gli *LLM*[17], cioè i Modelli linguistici di grandi dimensioni, e in generale le intelligenze artificiali, sono un ambito nuovo e in continua evoluzione nel settore informatico, per cui la ricerca e lo sviluppo di nuove tecnologie e metodologie è un aspetto fondamentale per le aziende.
Le intelligenze artificiali disponibili al pubblico sono, per motivo intrinseco della loro creazione, troppo generiche in quanto esse vengono addestrate caricando miliardi di documenti di carattere, lingua e contenuto differente.
Ciò comporta che questi LLM siano decisamente buoni per intrattenere delle conversazioni, in quanto i modelli a LLM funzionano a concatenamento di parole, cioè in base alla frase che si formula, valutando quale è la parola più adatta da inserire in seguito, ed essendo addestrati mediante dei testi, la loro capacità di dialettica è elevata.

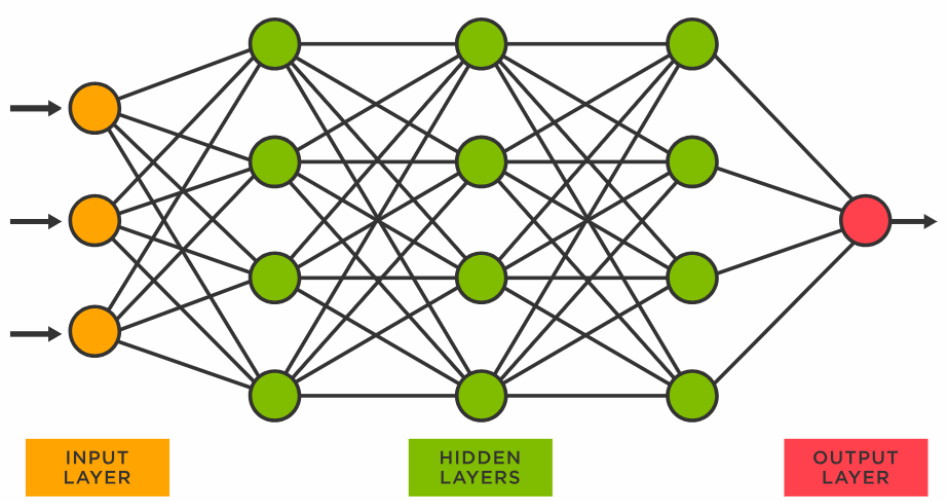


Figura 2.1: Schema sintesi di una rete neurale alla base di un *LLM* [18]

Gli LLM però non riescono a soddisfare di per sé le esigenze di un sistema nel quale l'intelligenza artificiale deve essere in grado di comprendere e rispondere a domande su specifici argomenti o documenti. In un contesto pratico ciò si traduce con il prodotto che deve essere in grado di rispondere a domande su documenti di contesto specifici, come ad esempio un documento di un regolamento aziendale, un documento di un contratto, un documento sportivo, e fornire informazioni unicamente su quell'argomento.
Per Risolvere questo problema, è necessario utilizzare un sistema di *RAG*[19], cioè un sistema di ottimizzazione del LLM in modo che il modello possa rispondere a domande specifiche su documenti di contesto.
La sfida principale del progetto consiste nel far generare ad un'intelligenza artificiale commerciale addestrata tramite documenti di carattere generale, delle informazioni

[17] Large Language Model
[18] Fonte: https://medium.com/
[19] Retrieval Augmented Generation

in particolare contenute nei documenti di contesto caricati, senza che essa possa inventarsi delle soluzioni.
Infatti, si potrebbe vedere la IA[20] come una scatola nera, nella quale si danno in pasto delle informazioni in *input*, e si ottengono degli *output*, senza però avere accesso IA metodi di lavorazioni dei dati al suo interno, è per cui necessario fornire informazioni in ingresso più specifiche e raffinate possibili.

Figura 2.2: Schema sintesi del concetto di *black-box* [21]

Nelle immagini 2.1 e 2.2 si può associare lo strato *Hidden Layer* alla *black-box* della IA, cioè la parte interna della IA che non è accessibile dall'esterno.
Vi è anche la sfida di trovare un modo rapido, efficiente e affidabile con il quale fornire solo i dati necessari all' IA per rispondere alle domande poste.
Dato il carattere di *chat* del *bot*, si ha la necessità che esso sia inoltre capace di capire il contesto di conversazione e rispondere di conseguenza.
Un'altra grossa sfida di questo progetto è il dover utilizzare tecnologie molto nuove, (certe di esse non hanno nemmeno raggiunto la prima versione *major*) e per cui spesso non esistono guide o documentazioni esaustive, per cui è necessario sperimentare e testare molto per capire come funzionano e come utilizzarle al meglio.
Infine, un'altra sfida consiste nel separare il contesto di conoscenze e di conversazione, in modo che il *bot* possa rispondere in modo coerente e corretto alle domande poste da utenti diversi.

[20] Intelligenza Artificiale
[21] Fonte: https://www.quora.com

2.3 Obiettivi

Riporto le notazioni utilizzate in seguito per identificare gli obiettivi delle attività:

- **O** per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
- **D** per i requisiti desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
- **F** per i requisiti facoltativi, rappresentanti valore aggiunto non strettamente competitivo.

Di seguito, la lista degli obiettivi:

- **Obbligatori**
  - **O01**: Apprendimento delle tecnologie di sviluppo React, Node.js, Fastify;
  - **O02**: Apprendimento delle tecnologie Git e utilizzo per il versionamento del codice;
  - **O03**: Analisi e scelta della soluzione IA da utilizzare;
  - **O04**: Apprendimento delle tecnologie e servizi IA per l'interfacciamento con la soluzione scelta;
  - **O05**: Sviluppo di un'interfaccia grafica (simil chat) per la comunicazione con l'IA;
  - **O06**: Sviluppo di un sistema di database per conservare le conversazioni, le schede e gli utenti;
  - **O07**: Sviluppo di un sistema di generazione di schede a partire dalle chat.
- **Desiderabili**
  - **D01**: Sviluppo di una soluzione *back-office* per l'inserimento di documenti per l'apprendimento dell'IA;
  - **D02**: Gestione utenti dell'applicativo.
  - **D03**: Sviluppo di un sistema per la gestione delle schede create.
- **Facoltativi**
  - **F01**: Sviluppo di una soluzione di *feedback* per l'apprendimento dell'IA;
  - **F02**: Sviluppo di una soluzione di tag per migliorare l'*user experience*;
  - **F03**: Sviluppo di un sistema *role-based* per l'accesso all'applicativo.

2.4 Vincoli

Si impongono dei vincoli alle tecnologie da usare nel progetto, in modo da garantire la similitudine con i prodotti già esistenti della azienda e permettere agli altri membri del *team* di poter aiutare lo studente in caso di necessità:

- Il prodotto deve essere sviluppato usando il *framework React* per il *frontend*;
- Il prodotto deve essere sviluppato usando il *framework Fastify* per il *backend*;
- Il prodotto deve essere sviluppato usando Typescript, il quale migliora la qualità del codice e la manutenibilità dello stesso;
- La sezione grafica deve essere sviluppata utilizzando il *framework Material-UI* e *styled-components*;
- Il prodotto deve essere una *Web App* (applicazione *web*);
- Uso di REST API per la comunicazione tra il *frontend* e il *backend*;
- Il prodotto deve venire versionato tramite *GitLab*;

2.5 Pianificazione

Ho rispettato la pianificazione delle attività, redatta anticipatamente rispetto all'inizio del progetto e disponibile di seguito

- **Prima settimana (40 ore)**
  - Incontro con persone coinvolte nel progetto per discutere i requisiti e le richieste relativamente al sistema da sviluppare;
  - Introduzione alla Cultura Aziendale;
  - Presa visione dell'infrastruttura esistente e assegnazione dei relativi strumenti necessari;
  - Formazione sulle tecnologie adottate.
- **Seconda settimana - (40 ore)**
  - Analisi dei requisiti;
- **Terza settimana - (40 ore)**
  - Progettazione Architetturale;
  - Analisi e definizione storie per il backlog dello Sprint[22] successivo assieme al referente.

[22] Sprint

- **Quarta settimana - (40 ore)**
  - Sviluppo delle storie assegnate: Realizzazione API della Chat;
  - *Sprint Review* con il referente e le altre persone coinvolte nel progetto;
  - Analisi e definizione storie per il backlog dello Sprint successivo assieme al referente;
  - Stesura documentazione.
- **Quinta settimana - (40 ore)**
  - Sviluppo delle storie assegnate: Realizzazione API della Chat;
  - *Sprint Review* con il referente e le altre persone coinvolte nel progetto;
  - Analisi e definizione storie per il backlog dello Sprint successivo assieme al referente;
  - Stesura documentazione.
- **Sesta settimana - (40 ore)**
  - Sviluppo delle storie assegnate: Creazione di una interfaccia grafica per la chat, inserimento dei documenti per l'apprendimento dell'IA;
  - *Sprint Review* con il referente e le altre persone coinvolte nel progetto;
  - Analisi e definizione storie per il backlog dello Sprint successivo assieme al referente;
  - Stesura documentazione.
- **Settima settimana - (40 ore)**
  - Sviluppo delle storie assegnate: Gestione delle sessioni multiple e degli utenti;
  - *Sprint Review* con il referente e le altre persone coinvolte nel progetto;
  - Analisi e definizione storie per il backlog dello Sprint successivo assieme al referente;
  - Stesura documentazione.
- **Ottava settimana - Conclusione (40 ore)**
  - Sviluppo delle storie assegnate: Creazione di un sistema di generazione di schede, creata *API* per inserire documenti e *Fix* di *bug*;
  - *Sprint Review* con il referente e le altre persone coinvolte nel progetto;
  - Collaudo finale e incontro finale con le persone coinvolte;
  - Stesura documentazione.

Le ore di lavoro sono state distribuite come segue:

Durata in ore	Descrizione dell'attività
30	**Formazione e sperimentazione sulle tecnologie**
40	**Analisi dei requisiti**
40	**Progettazione**
200	**Sviluppo**
	Realizzazione di codice Stesura documentazione Verifica e validazione del codice *Bug Fixing*
10	**Collaudo finale**
	Collaudo Stesura documentazione finale Incontro con gli *stakeholders* (nel mio caso, i miei tutor aziendali) per presentazione finale lavoro svolto
Totale ore	**320**

Tabella 2.1: Consuntivo delle ore di lavoro

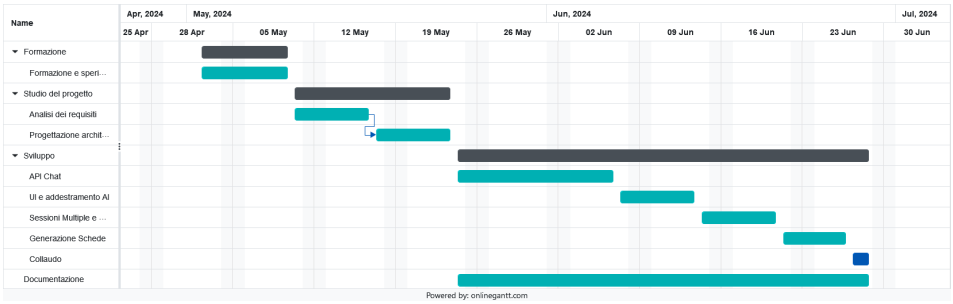


Figura 2.3: Diagramma di *Gantt* di suddivisione delle attività

2.6 Scelta del tirocinio

Le motivazioni che mi hanno spinto a scegliere il progetto di tirocinio offerto da ***Wavelop*** sono molteplici:

- **Introduzione al mondo del lavoro come informatico**: Durante la mia carriera scolastica, ho avuto modo di svolgere un tirocinio, ma in quanto studente di un istituto tecnico, indirizzo elettronico, non ho avuto modo di svolgere un tirocinio in un'azienda informatica, per cui ho deciso di svolgere il tirocinio per capire come funziona un'azienda di sviluppo software;
- **Sviluppo *fullstack***: In quanto prima esperienza vera di lavoro ho preferito optare per un progetto che mi permettesse di sviluppare sia il *frontend* che il *backend* di un'applicazione, in modo da poter capire come funzionano entrambi e poter scegliere in futuro quale dei due ruoli mi piace di più, se non entrambi;

- **Vicinanza**: La ricerca e la scelta di un'azienda in cui svolgere il tirocinio è stata condizionata dalla vicinanza al luogo di residenza, in modo da ridurre i tempi di spostamento e poter dedicare più tempo allo studio e al lavoro, inoltre questa ricerca mi ha permesso di scoprire che nella mia zona di residenza ci sono molte più aziende informatiche di quanto pensassi;

Queste considerazioni mi hanno consentito di definire una serie di obiettivi personali per quanto riguarda la scelta dell'azienda, il tema delle attività e l'acquisizione di conoscenze e competenze:

- Realizzare una *Web App* da zero, partendo dall'analisi dei requisiti fino alla realizzazione e collaudo finale;
- Progettare e sviluppare un'interfaccia grafica semplice, intuitiva, funzionale e accessibile ma allo stesso tempo accattivante e moderna;
- Realizzare il prodotto adoperando modelli di progettazione e pattern architetturali, in particolare come gestire la divisione tra *frontend* e *backend* mediante l'uso di *API*;
- Approfondire le conoscenze per le nuove tecnologie di intelligenza artificiale, in particolare per quanto riguarda gli *LLM* e il *RAG*;
- Completare gli obiettivi aziendali posti per il progetto dando la priorità a quelli obbligatori;
- Partecipare attivamente a *stand-up meeting* e *sprint review* per capire come funzionano e poter migliorare la mia capacità di comunicazione e di lavoro in *team*;
- Realizzare controlli automatici di qualità del codice, in modo da garantire la qualità del prodotto finale;

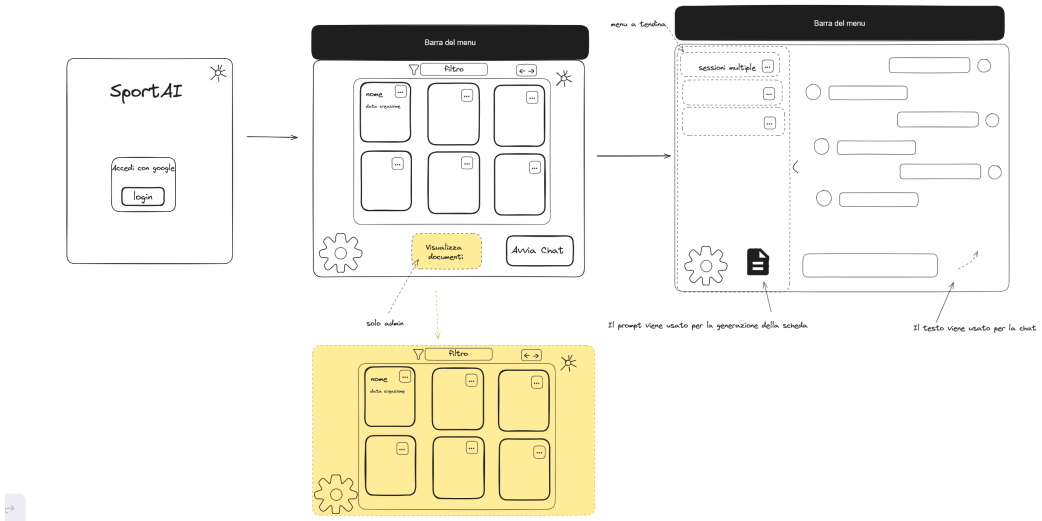


Figura 2.4: Pianificazione interfaccia del progetto *SportAI*, oggetto del tirocinio

Capitolo 3

Elementi caratterizzanti del progetto

Questo capitolo illustra gli strumenti e le tecnologie impiegati durante il tirocinio, delineando in modo chiaro le attività di sviluppo del progetto di stage. Inoltre, offre una panoramica dei risultati ottenuti, evidenziando la documentazione prodotta, il codice sviluppato e gli obiettivi raggiunti.

3.1 Stile lavorativo

I riti aziendali di ***Wavelop*** prevedono l'adozione di metodologie *Agile* per lo sviluppo dei progetti, nello specifico, prevedono che, ogni settimana si tenga una riunione tra ogni singolo componente del *team*, il proprietario e lo *scrum master*. Per cui con i tutor *Nicola Capovilla* (Proprietario) e *Matteo Granzotto* (*Scrum Master*) abbiamo concordato di svolgere un incontro ogni venerdì pomeriggio. Gli obiettivi di queste riunioni sono i seguenti:

- Presentare i progressi compiuti durante la settimana;
- Effettuare una retrospettiva sulla settimana passata;
- Valutare lo stato di avanzamento del progetto rispetto alle aspettative;
- Pianificare le attività da svolgere nello *sprint* successivo.

Durante la settimana comunque era possibile contattare i tutor per qualsiasi dubbio o problema, in qualsiasi momento, allo *stand-up meeting* del mattino, o di persona o tramite *Discord* se si trovavano in *smart working*.
Durante tutto il periodo di stage, ho ricevuto supporto dai membri con più esperienza del *team*, in particolare per quanto riguarda la parte di collegamento tra *frontend* e *backend*.

3.2 Strumenti utilizzati

3.2.1 Strumenti di sviluppo

HTML

Versione: 5

Descrizione: è il linguaggio *standard* di *markup* utilizzato per la creazione di pagine *web*. Serve a strutturare i contenuti di una pagina *web*, definendo la disposizione di testo, immagini, *link*, tabelle, e altri elementi multimediali.
HTML, sviluppato dal *World Wide Web Consortium* (W3C), è il fondamento su cui si basa il *web*, consentendo ai *browser* di interpretare e visualizzare i documenti *web*. È un linguaggio basato su *tag*.

CSS

Versione: 3

Descrizione: è un linguaggio di stile utilizzato per descrivere l'aspetto e la formattazione di un documento scritto in HTML.
CSS, sviluppato dal *World Wide Web Consortium* (W3C), permette di separare il contenuto dalla presentazione, migliorando così l'accessibilità e la personalizzazione. CSS consente di applicare stili in modo efficiente e coerente su più pagine web.

TypeScript

Versione: 5.1.6

Descrizione: è un linguaggio di programmazione *open-source* sviluppato da *Microsoft*. È una versione "*superset*" di *JavaScript*, il che significa che aggiunge nuove funzionalità e tipizzazione statica al linguaggio *JavaScript*.

React

Versione: 18.2.0

Descrizione: è un *framework open-source* per la creazione di interfacce utente, cioè la parte di software con la quale l'utente interagisce.
Questa libreria è sviluppata da *Meta* (per riferimento, i creatori di *Facebook*); *React* si occupa del *rendering* a video dei componenti che compongono l'interfaccia utente, questi componenti sono riutilizzabili e vengono realizzati in *JavaScript* (.jsx) o *TypeScript* (.tsx).

Material UI

Versione: 5.16.0

Descrizione: è una libreria di componenti *React* che implementa il *Material Design*, uno stile di design sviluppato da *Google*.
Questa libreria offre una serie di componenti predefiniti e pronti all'uso che possono essere utilizzati per creare interfacce utente moderne e intuitive, inoltre ogni componente è personalizzabile.

i18next

Versione: 22.5.0

Descrizione: è una libreria di internazionalizzazione per *JavaScript* che consente di gestire la traduzione di testi all'interno di un'applicazione.
i18next offre funzionalità avanzate per la gestione delle traduzioni, come la gestione di plurali, la sostituzione di variabili e la gestione di traduzioni dinamiche.

Styled components

Versione: 6.1.11

Descrizione: è una libreria *open-source* per *React* che consente di scrivere stili *CSS* all'interno dei componenti *React*.
Styled components permette di creare componenti *React* con stili personalizzati, rendendo il codice più leggibile e manutenibile.

Redux

Versione: 1.9.5

Descrizione: è una libreria di gestione dello stato per applicazioni *JavaScript*. *Redux* è utilizzato per gestire lo stato dell'applicazione in modo centralizzato, rendendo più semplice la gestione dello stato e la comunicazione tra i componenti.

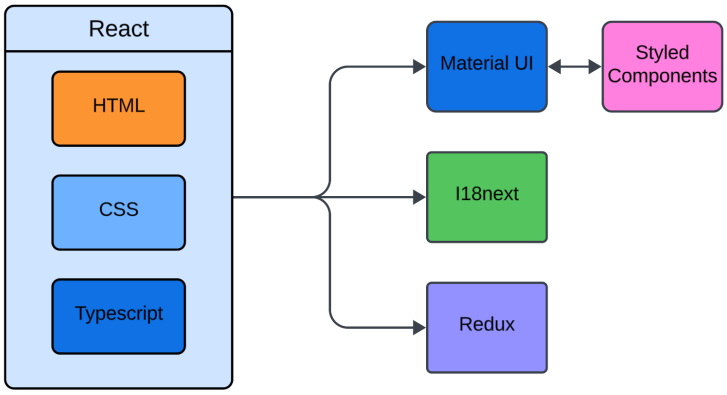


Figura 3.1: Diagramma di collegamento delle tecnologie *frontend*

Excalidraw

Versione: 0.17.6

Descrizione: è uno strumento di disegno *open-source* che consente di creare diagrammi, e schizzi in modo semplice e intuitivo.
Excalidraw è stato utilizzato per la prototipazione dell'interfaccia utente del prodotto e per la realizzazione dello schema di funzionamento del *software*.

Node.js

Versione: 21.7.3

Descrizione: è un ambiente di esecuzione *JavaScript open-source* che consente di eseguire codice *JavaScript* lato server.
Node.js è basato sul motore *V8* di *Google Chrome* ed è utilizzato per creare applicazioni *web* scalabili e performanti.

Fastify

Versione: 4.17.0

Descrizione: è un *framework web* italiano *open-source* per *Node.js* che consente di creare applicazioni *web* veloci e scalabili.
Fastify è progettato per essere estremamente performante, con tempi di risposta molto bassi e un basso utilizzo di risorse.

Langchain

Versione: 0.2.4

Descrizione: è un *framework open source* per lo sviluppo di applicazioni che utilizzano modelli linguistici di grandi dimensioni (LLM). Gli strumenti e le *API* di LangChain sono disponibili nelle librerie basate su Python e JavaScript e semplificano il processo di creazione di applicazioni basate su *LLM* come *chatbot* e agenti virtuali.

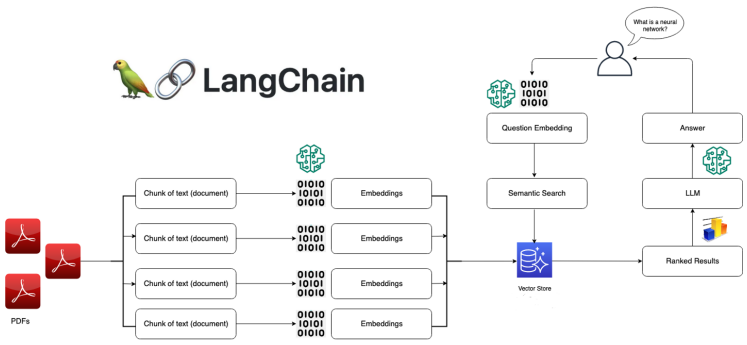


Figura 3.2: Schema di funzionamento di *LangChain*

ChromaDb

Versione: 1.8.1

Descrizione: è un *database open-source* ottimizzato per l'archiviazione e la gestione di dati vettoriali, utilizzato principalmente in applicazioni di intelligenza artificiale e apprendimento automatico. ChromaDB è progettato per supportare la ricerca e l'indicizzazione di dati vettoriali, rendendolo ideale per scenari come la ricerca di similarità, il clustering e la classificazione di dati complessi. Offre un'integrazione fluida con vari *framework* di *machine learning* e consente una gestione efficiente dei dati per applicazioni su larga scala.

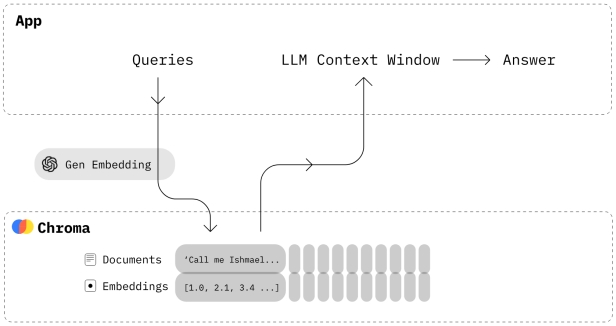


Figura 3.3: Schema di funzionamento di *ChromaDb*

MongoDb

Versione: 6.4.0

Descrizione: è un *database open-source* progettato per la gestione di dati non strutturati e semi-strutturati.
MongoDB è basato su un modello di dati flessibile e scalabile, che consente di memorizzare dati in documenti *JSON* e di eseguire query complesse su di essi.

MinIO / aws-sdk

Versione: 2.1467.0

Descrizione: *MinIO* è un server di archiviazione di oggetti *open-source* che consente di memorizzare grandi quantità di dati in modo scalabile ed efficiente.
MinIO è compatibile con l'*API* di *Amazon S3* e può essere utilizzato come alternativa economica ai servizi di archiviazione di oggetti basati su cloud.
aws-sdk è un *framework open-source* per *Node.js* che consente di interagire con i servizi *Amazon Web Services* (*AWS*).

Ollama

Versione: 0.1.48

Descrizione: è un *framework open-source* per la creazione di applicazioni di intelligenza artificiale e apprendimento automatico.
Ollama offre funzionalità avanzate per la creazione di modelli di apprendimento automatico, la gestione dei dati e la valutazione delle prestazioni dei modelli.

Docker

Versione: 23.0.3

Descrizione: è una piattaforma di *containerizzazione open-source* che consente di creare, distribuire e gestire applicazioni in *container*.
Docker offre funzionalità avanzate per la creazione di *container*, la gestione delle risorse e la distribuzione delle applicazioni in ambienti *cloud* e *on-premise*.

Gemini API

Versione: 1.5 Flash

Descrizione: è una *API* per accedere al modello di linguaggio di grandi dimensioni (*LLM*) sviluppato da *Google* che consente di generare testi di alta qualità in modo automatico.

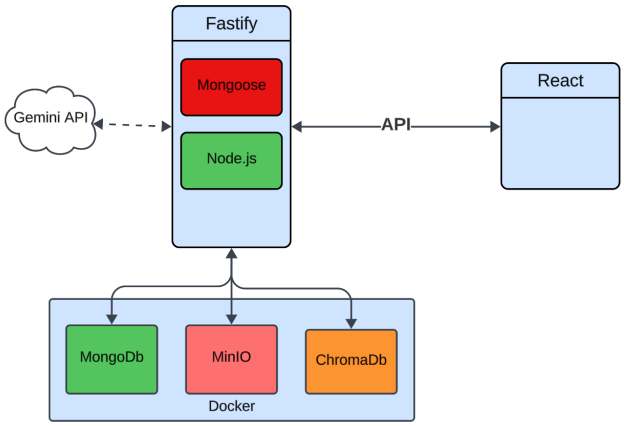


Figura 3.4: Diagramma di collegamento delle tecnologie *backend*

StarUML

Versione: 6.1.2

Descrizione: è uno strumento di modellazione *open-source* che consente di creare diagrammi *UML* e di modellare il software in modo visuale.
StarUML offre una vasta gamma di funzionalità per la modellazione di software, tra cui diagrammi di classi, diagrammi di sequenza e diagrammi di attività.

Visual Studio Code

Versione: 1.91.0

Descrizione: è un *editor* di codice *open-source* sviluppato da *Microsoft* per *Windows*, *macOS* e *Linux*.
Visual Studio Code offre funzionalità avanzate per la scrittura di codice, come il completamento automatico, il *debugging* integrato e la gestione dei *repository Git*.

3.2.2 Strumenti di versionamento

Git

Versione: 2.45.2

Descrizione: è un sistema di controllo delle versioni distribuito *open-source* che consente di tenere traccia delle modifiche apportate al codice sorgente di un progetto.
Git è ampiamente utilizzato nello sviluppo di *software* per gestire il codice sorgente, coordinare il lavoro tra i membri del *team* e mantenere una cronologia delle modifiche.

GitLab

Versione: 17.1.1

Descrizione: è una piattaforma di gestione del codice sorgente basata su *Git* che consente di collaborare, gestire e distribuire il codice sorgente di un progetto.
GitLab offre funzionalità avanzate per la gestione del codice sorgente, come il *repository Git*, il *CI/CD* e il *code review*.

3.2.3 Strumenti di documentazione

Documenti Google

Descrizione: è una suite di applicazioni di produttività basata su *cloud* sviluppata da *Google*. *Documenti Google* offre una vasta gamma di strumenti per la creazione, la modifica e la condivisione di documenti, fogli di calcolo e presentazioni.

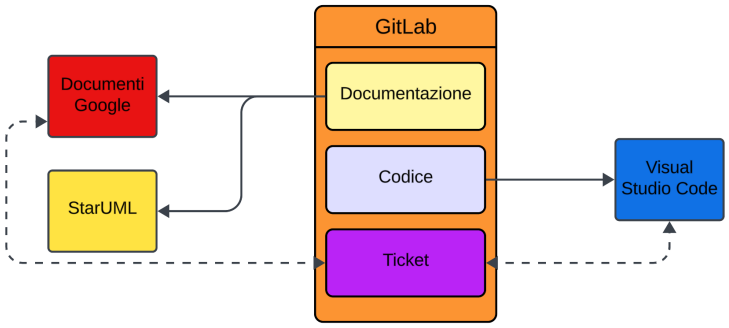


Figura 3.5: Diagramma di collegamento delle tecnologie di supporto

3.3 Analisi dei requisiti

L'**analisi dei requisiti** ha come obiettivo la comprensione dei bisogni espressi dal cliente, al fine di definire i requisiti del prodotto da sviluppare.
Dopo una discussione con i *tutor* circa il funzionamento e l'aspetto del prodotto, ho stilato un documento di analisi dei requisiti, che riporta i casi d'uso e i requisiti del prodotto. I casi d'uso una volta stilati sono stati controllati, raffinati e infine approvati dai tutor per procedere con lo sviluppo del prodotto.

3.3.1 Casi d'uso

Con "caso d'uso" si intende una descrizione con punto di vista macroscopico sul funzionamento del sistema, per definire le interazioni tra gli attori, che possono essere utenti o altri *software*, e il sistema che interagisce con gli attori.
I casi d'uso vengono rappresentati in maniera schematica tramite diagrammi, in modo da rendere più chiara e veloce la comprensione del funzionamento del sistema. Ogni caso d'uso può ricevere ulteriori specifiche mediante l'uso di sotto-casi d'uso dipendenti che catturano dettagli specifici.
Di seguito riporterò i casi d'uso principali del prodotto, con relativo nome e descrizione.

Nomenclatura

I casi d'uso sono identificati da una sigla così composta:

UC-[Codice]

- **UC**: abbreviativo di "*Use Case*";
- **Codice**: numero del caso d'uso, i sottocasi d'uso vengono rappresentati con la forma **[Codice].[SottoCodice]**.

Attori primari

Sono presenti tre tipologie di attori primari:

- ***User* non autenticato**: cioè un utente che non ha effettuato l'autenticazione alla piattaforma; ha accesso unicamente alla pagina di *login*;
- ***User* autenticato**: cioè un utente che ha effettuato l'autenticazione alla piattaforma; ha accesso a tutte le funzionalità per user base;
- ***Admin***: cioè un utente che ha accesso a tutte le funzionalità della piattaforma, comprese quelle di amministrazione come quella di poter caricare documenti.

Attori secondari

Sono presenti due tipologie di attori secondari:

- ***Google***: è un attore secondario che permette l'autenticazione tramite servizi di autenticazione di *Google*;

- ***LLM***: è un attore secondario che permette la generazione di testi tramite modelli di linguaggio di grandi dimensioni.
  A sua volta questo attore può essere interpretato sia da Gemini (LLM di Google) che da Ollama (LLM *open-sorce*).

Lista dei principali casi d'uso

UC-8: Visualizza risposta

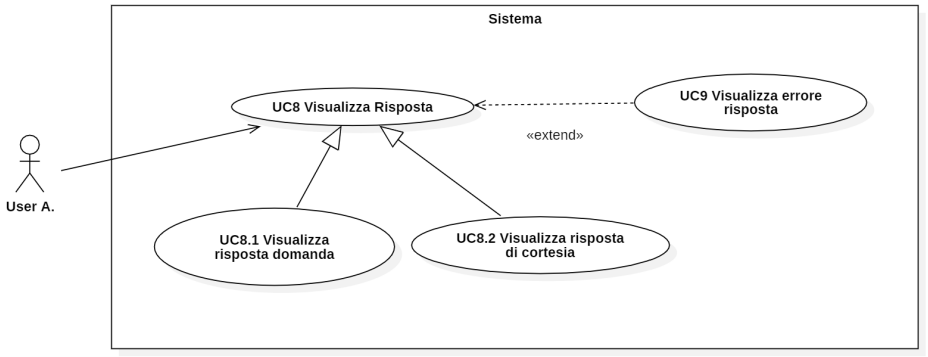


Figura 3.6: Diagramma dei casi d'uso UC-8 e UC-9.

- **Descrizione:** L'utente autenticato vuole visualizzare la risposta del chatbot.
- **Estensioni:** UC-8.1: Visualizza risposta alla domanda, UC-8.2 visualizza risposta di cortesia, UC-9 visualizza errore.
- **Attore principale:** utente autenticato;
- **Precondizioni:** l'utente è autenticato;
- **Postcondizioni:** l'utente visualizza la risposta del *chatbot*.

Questo caso d'uso rappresenta la funzionalità principale del prodotto, ovvero la visualizzazione della risposta del *chatbot* da parte dell'utente autenticato. Nel caso la domanda non risultasse pertinente, scatterà il sottocaso d'uso che farà in modo che l'utente riceva una risposta di cortesia nella quale viene spiegato che la domanda non è stata compresa.

UC-10: Visualizzazione della chat history

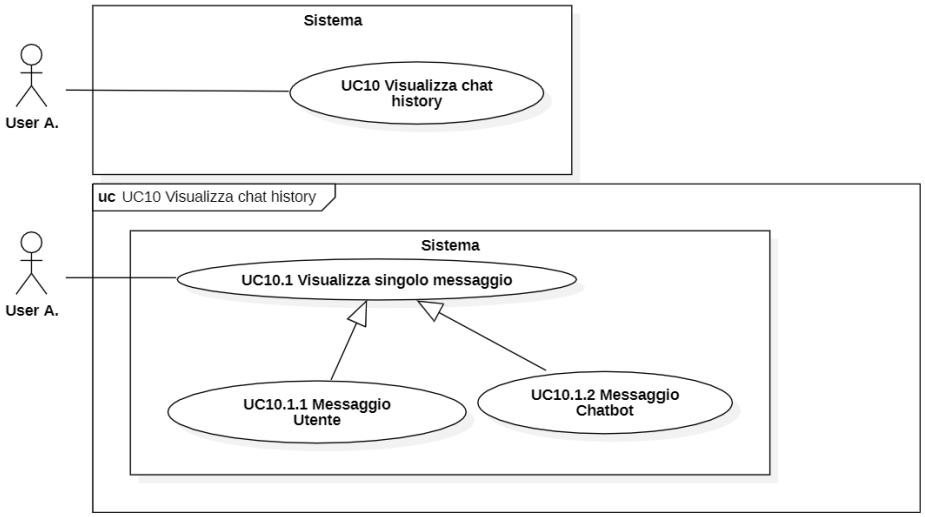


Figura 3.7: Diagramma del caso d'uso UC-10.

- **Descrizione:** L'utente autenticato vuole visualizzare la *chat history*.
- **Estensioni:** UC-10.1: visualizza singolo messaggio. UC-10.1.1: messaggio utente; UC-10.1.2: messaggio *chatbot*.
- **Attore principale:** utente autenticato;
- **Precondizioni:** L'utente è autenticato e nella pagina di *chat* in una conversazione;
- **Postcondizioni:** L'utente visualizza la *chat history*.

Questo caso d'uso rappresenta la funzionalità di visualizzazione della *chat history* da parte dell'utente autenticato. La *chat history* è composta da una serie di messaggi scambiati tra l'utente e il *chatbot*, e viene visualizzata in ordine cronologico. Come indicato poi nei sottocasi d'uso 10.1.1 e 10.1.2, i messaggi possono essere di due tipologie: messaggi inviati dall'utente (cioè le domande) e messaggi inviati dal *chatbot* (cioè le risposte), questi due tipologie di messaggi vengono visualizzati in modo diverso.

3.4 Progettazione

La progettazione getta delle basi solide per lo sviluppo del prodotto, definendo l'architettura del sistema, i dettagli implementativi e l'aspetto del prodotto.
Di seguito, si approfondiscono le scelte effettuate per:

- La scelta di diversi LLM;
- Database da utilizzare per RAG.
- Scelta dell'architettura del sistema.;

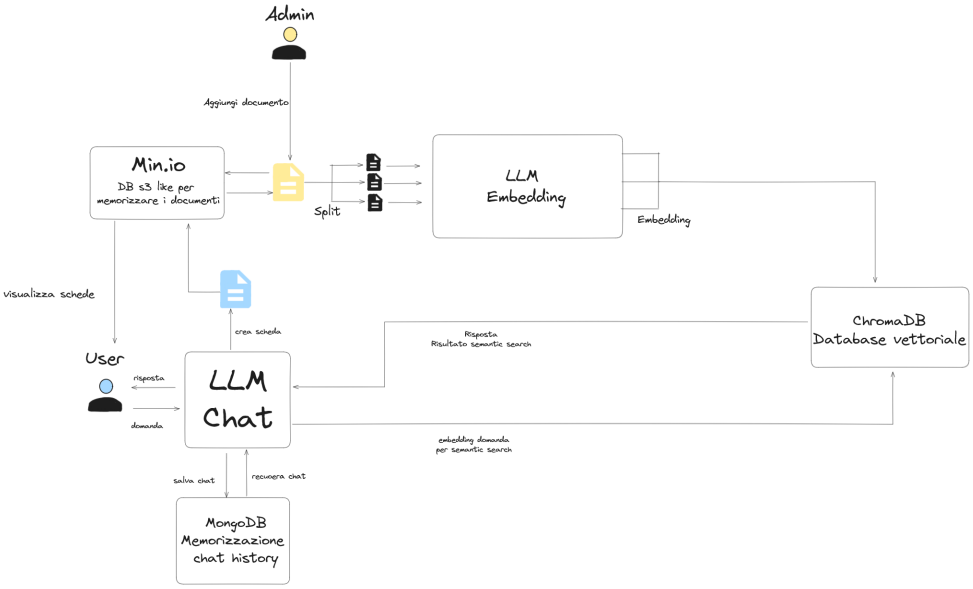


Figura 3.8: Schema di funzionamento di *SportAI*

3.4.1 LLM a confronto

Ogni modello LLM differisce per dei parametri che ne definiscono la qualità e la potenza, è bene quindi analizzare nel dettaglio le soluzioni principali che sono state analizzate:

- **Costo**: Questo indice rappresenta il costo di utilizzo del modello di LLM. Il costo è calcolato in base al numero di token generati dal modello di LLM.
- **Potenza**: Questo indice rappresenta la potenza del modello di LLM. si misura in Miliardi di Parametri, ovvero il numero di parametri che il modello di LLM ha.
- **Flusso di token**: Questo indice rappresenta il flusso di token che il modello di LLM può generare o che può ricevere.

La scelta finale nel progetto è stata quella di realizzare un selettore multiplo, che dà la possibilità all'utente di scegliere il modello di LLM che preferisce utilizzare,

inserendo perciò le *API* key dei modelli selezionati.
Infatti, gli LLM sono stati inseriti nel codice tramite pattern *Factory* e *singleton* per permettere la scelta del modello da utilizzare senza comportare cambiamenti nel codice.
Nonostante questa scelta sia stata fatta, e che il modello Gemini è quello preferito (vedasi motivazioni sottostanti) è bene fare un confronto tra i modelli di LLM più utilizzati per capire le differenze tra di essi.

Gemini 1.5 Flash

Vantaggi :

- È un modello di LLM di Google, ed è quindi un modello potente e preciso ma inferiore alla versione PRO;
- *API* di facile utilizzo e apprendimento;
- Flusso di token elevatissimo, a livello numerico, questo parametro è più di 10 volte la concorrenza.
- possibilità di utilizzo di una *API* Key gratuita (con limitazioni).
- Bassa aleatorietà nelle risposte.

Svantaggi :

- Non è un modello *open-source*;
- Non è il modello più potente sul mercato;
- Costo elevato se si superano i limiti della versione gratuita;

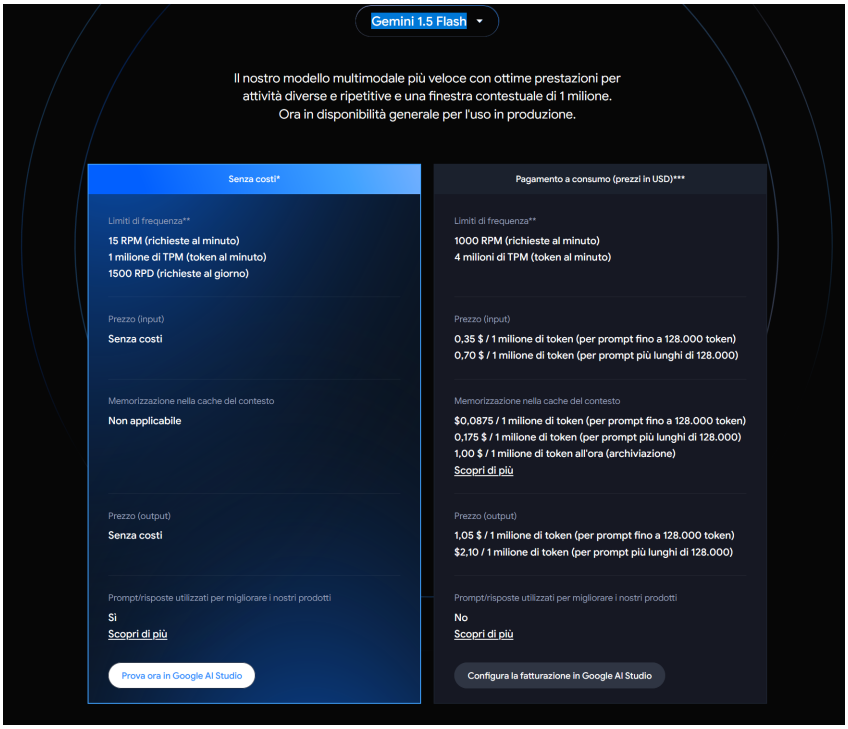


Figura 3.9: Piano offerto da *Google* per *Gemini 1.5 Flash*[23]

ChatGPT 3.5

Vantaggi :

- È un modello di LLM di OpenAI, ed è quindi uno dei modelli più potenti e precisi attualmente disponibili;
- *API* di facile utilizzo e apprendimento;
- Risposte di alta qualità;
- Bassa aleatorietà nelle risposte.

Svantaggi :

- Non è un modello *open-source*;
- Costo elevato dei piani a pagamento;
- Flusso di token limitato rispetto alla concorrenza;
- Non è possibile utilizzare una *API* Key gratuita.

Ollama

Vantaggi :

- È un modello di LLM *open-source*;
- *API* di facile utilizzo e apprendimento;
- È possibile utilizzare un LLM tra i centinaia disponibili a seconda la potenza della macchina che si possiede;

Svantaggi :

- Richiede una macchina potente per poter essere utilizzato;
- Flusso di token limitato rispetto alla concorrenza;
- aleatorietà elevata nelle risposte;
- Modelli piccoli sono poco performanti rispetto alla concorrenza.

Sono stati presi comunque in considerazioni anche altre soluzioni come *Amazon Bedrock* e *Hugging Face*, ma non sono stati selezionati perché ritenuti peggiori rispetto ai modelli sopra citati.

[23] Fonte: www.ai.google.dev

3.4.2 Scelta del database per RAG

La tecnologia del RAG, come indicato in precedenza, è di fatto la funzionalità fondamentale del prodotto, infatti tramite essa è possibile:

- **Estrarre le informazioni necessarie**: tramite *query* a database vettoriali è possibile estrarre le informazioni necessarie per la risposta senza dover passare l'intero contesto del documento, operazione onerosa in termini di tempo e risorse;
- **Embedding dei documenti**: i documenti vengono *embeddizzati* in modo da poter essere utilizzati per fare *query* e ottenere informazioni mediante vicinanza vettoriale tra domanda e contenuto del documento;
- **Rispondere in modo più pertinente**: il RAG permette di rispondere in modo più pertinente alle domande, in quanto il modello utilizzerà le conoscenze "esterne", cioè quelle ricevute dai documenti.
- **Raffinamento *prompt***: il RAG permette di raffinare il *prompt* in modo da forzare il modello a rispondere solo e unicamente con le informazioni presenti nei documenti.

A seguito di una fase di ricerca e di studio è stato scelto di utilizzare il database **ChromaDB**, di seguito riporto i motivi della scelta confrontando ChromaDB con altri database vettoriali:

ChromaDB

Vantaggi :

- È un database *open-source*;
- È possibile utilizzarlo mediante *Docker*;
- È possibile utilizzarlo in locale;
- È un database unicamente vettoriale, perciò ottimizzato e dotato di funzionalità specifiche che permettono di effettuare *query* in modo efficiente;

Svantaggi :

- Tecnologia molto giovane, perciò poco documentata;
- La curva di apprendimento è molto ripida;
- La visualizzazione dei dati all'interno del database non è molto intuitiva.

PineCone

Vantaggi :

- È il database vettoriale più potente sul mercato;
- È supportato da una vasta documentazione;
- È utilizzabile in diversi linguaggi di programmazione;

- È un database unicamente vettoriale, perciò ottimizzato e dotato di funzionalità specifiche che permettono di effettuare *query* in modo efficiente;

Svantaggi :

- È un database a pagamento, e i piani gratuiti sono molto limitati;
- La curva di apprendimento è molto ripida;
- Non è possibile utilizzarlo in locale per testare le funzionalità.

MongoDB Atlas

Vantaggi :

- È un database che permette di memorizzare sia vettori che dati non vettoriali, e pertanto utilizzabile come unico database per il progetto;
- È utilizzabile in diversi linguaggi di programmazione;
- È uno tra i database più utilizzati al mondo, e pertanto è molto documentato;

Svantaggi :

- È un database a pagamento, e i piani gratuiti sono molto limitati;
- Non essendo unicamente un database vettoriale, le funzionalità di *query* sono limitate e meno efficienti rispetto a database vettoriali puri;
- Non è possibile utilizzarlo in locale per testare le funzionalità.

3.4.3 Architettura del sistema

A seguito di una fase di studio e di analisi è stata scelta l'architettura *three-tier* per il progetto, di seguito riporto i motivi della scelta:

- **Scalabilità**: L'architettura *three-tier* permette di scalare i singoli livelli in modo indipendente. Ad esempio, è possibile aumentare le risorse del livello applicativo senza dover necessariamente potenziare il livello di presentazione o il livello dei dati. Questo permette una gestione più efficiente delle risorse e una risposta più rapida alle esigenze di crescita del sistema.
- **Sicurezza**: La separazione dei livelli consente di implementare misure di sicurezza specifiche per ciascun *tier*. Ad esempio, il livello dei dati può essere protetto da un firewall dedicato, mentre il livello applicativo può includere meccanismi di autenticazione e autorizzazione robusti. Inoltre, isolando i dati dall'accesso diretto dell'utente finale, si riduce il rischio di compromissione.
- **Manutenibilità**: La divisione dell'architettura in tre livelli distinti facilita la manutenzione del sistema. Gli sviluppatori possono apportare modifiche a un livello senza influenzare gli altri, riducendo così il rischio di introdurre errori. Ad esempio, è possibile aggiornare l'interfaccia utente senza modificare la logica di business o la struttura dei dati sottostante.

- **Flessibilità**: L'architettura *three-tier* supporta la flessibilità nell'adozione di nuove tecnologie e nella sostituzione di componenti obsoleti. Ad esempio, il livello di presentazione può essere aggiornato per utilizzare un nuovo *framework* JavaScript senza necessità di cambiare il livello applicativo o il database. Questa flessibilità consente al sistema di evolvere e adattarsi rapidamente alle nuove esigenze del mercato.

- **Facilità di sviluppo**: La suddivisione del sistema in tre livelli distinti semplifica lo sviluppo, permettendo ai team di lavorare in parallelo sui diversi componenti. Gli sviluppatori possono specializzarsi in specifici *tier* (*frontend, backend, database*), migliorando l'efficienza e la qualità del codice prodotto. Inoltre, il riutilizzo del codice è facilitato, poiché la logica di business può essere condivisa tra diverse interfacce utente.

Descrizione dei livelli

- **Livello di presentazione (Presentation Tier)**:
  Questo livello è responsabile dell'interfaccia utente e dell'interazione con l'utente finale. Comprende le pagine web, i moduli di input, i componenti grafici e le funzionalità di navigazione.
  Il livello di presentazione in sostanza si occupa del rapporto visivo tra l'utente e il sistema.
  Le tecnologie utilizzate in questo livello sono: HTML, CSS, TypeScript e *framework* di sviluppo *frontend* React.

- **Livello applicativo (Application Tier)**:
  Conosciuto anche come livello di logica di business, questo *tier* gestisce la logica applicativa, le regole di business e il flusso di dati tra il livello di presentazione e il livello dei dati.
  Qui risiedono i server applicativi e le *API* che elaborano le richieste degli utenti.
  Le tecnologie utilizzate in questo livello sono: Node.js, TypeScript e *framework* di sviluppo *backend* Fastify.

- **Livello dei dati (Data Tier)**: Questo livello è dedicato alla gestione dei dati, mediante le operazioni *CRUD*, cioè di creazione, lettura, aggiornamento ed eliminazione dei dati. Comprende database vettoriale (ChromaDb), database NoSQL (MongoDB) e database a oggetti (MinIO). Il livello dei dati garantisce l'integrità e la sicurezza delle informazioni archiviate e fornisce accesso rapido e affidabile ai dati richiesti dal livello applicativo. Per facilitare le operazioni di gestione dei dati, per la comunicazione con MongoDb è stato utilizzato il *framework* Mongoose, che è un *Object Data Modeling* (ODM) per MongoDB e Node.js.

Comunicazione tra i livelli

La comunicazione tra i tre livelli avviene tramite protocolli e interfacce ben definiti. Il livello di presentazione comunica con il livello applicativo tramite *API* REST, mentre il livello applicativo interagisce con il livello dei dati utilizzando query che possono essere di diverso tipo a seconda di quale dei tre database si sta comunicando. Questa

separazione delle responsabilità e dei canali di comunicazione assicura una maggiore modularità e facilita il debug e la risoluzione dei problemi, oltre che garantisce una maggiore sicurezza in quanto i dati sensibili non vengono esposti direttamente al livello di presentazione.

Pattern architetturali utilizzati

Nell'implementazione dell'architettura *three-tier*, sono stati utilizzati diversi pattern architetturali per garantire modularità, efficienza e manutenibilità tra i quali:

- **Modularità dei Componenti React**: Nel livello di presentazione, l'utilizzo di React permette di creare componenti modulari e riutilizzabili. Ogni componente gestisce una parte specifica dell'interfaccia utente e può essere combinato con altri componenti per formare interfacce complesse. Questo approccio migliora la manutenibilità e facilita l'aggiornamento dell'interfaccia utente, poiché i cambiamenti possono essere isolati a singoli componenti senza influenzare l'intero sistema.
- **Singleton**: Nel livello applicativo, il pattern Singleton viene utilizzato per garantire che una classe abbia una sola istanza e fornisca un punto di accesso globale a tale istanza. Questo è particolarmente utile per la gestione di risorse condivise come connessioni al database, configurazioni globali, o logiche di business critiche che devono essere uniche attraverso l'intera applicazione.
- **Factory**: Il pattern Factory viene impiegato per creare oggetti senza dover specificare la classe esatta dell'oggetto che verrà creato. Questo pattern è utile per la creazione di oggetti complessi o dipendenti da configurazioni, poiché incapsula la logica di creazione e permette di cambiare le implementazioni senza modificare il codice che utilizza questi oggetti. Nel livello applicativo, il Factory Pattern facilita la gestione delle dipendenze e migliora la testabilità del codice.

Vantaggi e svantaggi dell'architettura *three-tier*

Come ogni approccio architetturale, l'architettura *three-tier* presenta vantaggi e svantaggi che vanno valutati attentamente in base alle esigenze specifiche del progetto:

- **Vantaggi**:
  - **Modularità**: La suddivisione in livelli consente di isolare le problematiche e risolverle in modo più mirato.
  - **Riutilizzo del codice**: La logica di business e i componenti di accesso ai dati possono essere riutilizzati su diverse piattaforme di presentazione.
  - **Indipendenza della tecnologia**: Ogni livello può essere sviluppato e aggiornato utilizzando le tecnologie più appropriate e recenti.
- **Svantaggi**:
  - **Complessità**: La gestione di tre livelli distinti può aumentare la complessità del sistema.

  - **Performance**: La comunicazione tra i livelli può introdurre latenza, richiedendo ottimizzazioni specifiche per mantenere alte prestazioni.
  - **Costo**: La necessità di risorse dedicate per ogni livello può incrementare i costi operativi e di sviluppo.

L'adozione di un'architettura *three-tier* offre numerosi vantaggi in termini di scalabilità, sicurezza, manutenibilità, flessibilità e facilità di sviluppo. Nonostante alcune potenziali complessità e costi aggiuntivi, i benefici superano di gran lunga gli svantaggi, rendendo questa architettura una scelta solida per lo sviluppo di applicazioni moderne e scalabili.

3.5 Codifica

La **codifica** ha come obiettivo l'implementazione del *software* in base a quelle che sono state le decisioni in sede di **progettazione**. Durante la stesura del codice, ho seguito alcune direttive che mi sono state fornite dal tutor aziendale, tra le quali:

- **Utilizzo di *Arrow function***: Le *arrow function* offrono una sintassi più compatta per la scrittura di funzioni anonime e mantengono il contesto del ***this*** del blocco esterno, semplificando la gestione delle funzioni *callback* e dei metodi all'interno delle classi. Ad esempio:

```
const add = (a, b) => a + b;
```

- **Utilizzo di *TypeScript***: TypeScript è stato scelto per migliorare la robustezza del codice, grazie alla sua capacità di effettuare controlli statici sui tipi. Questo riduce significativamente gli errori di *runtime* e migliora l'autocompletamento e il *refactoring* del codice nei moderni IDE.
- **Utilizzo del *PascalCase*** per la definizione delle variabili e delle funzioni: *PascalCase* è stato adottato per garantire una maggiore coerenza e leggibilità nel codice.
- **Creazione di un file .tsx per ogni componente React**: Questo approccio facilita i cambiamenti senza dover modificare il codice in più file. Ogni componente React ha il proprio file .tsx, migliorando la modularità e la manutenibilità del codice. Ad esempio, un componente 'Button' sarà definito in 'Button.tsx'.
- **Divisione del codice in diversi file**: La suddivisione del codice in file distinti rende più facile la manutenzione e la comprensione. Di seguito riporto la struttura delle cartelle del progetto:

apps

- **sport-ai-api**
  - **config**: Configurazioni generali dell'applicazione (es. variabili di ambiente, impostazioni del database).
  - **src**
    - **controllers**: Gestori delle richieste HTTP, contengono la logica di business legata ai vari endpoint.
    - **core**: Componenti core dell'applicazione, come middleware, autenticazione e autorizzazione.
    - **libs**: Librerie e utilità comuni utilizzate nell'applicazione.
    - **models**: Definizioni dei modelli di dati utilizzati nell'applicazione.
    - **routes**: Definizione delle rotte dell'applicazione.
    - **services**: Logica di business e interazioni con altri sistemi o servizi.
    - **types**: Definizioni dei tipi TypeScript utilizzati nell'applicazione.
    - **utils**: Funzioni utilitarie e *helper* generici.
    - **index.ts**: Punto di ingresso principale dell'applicazione.
- **sport-ai-ui**
  - **public**: File pubblici accessibili dall'esterno (es. index.html, favicon).
  - **src**
    - **apis**: Interfacce per le chiamate *API*.
    - **components**: Componenti React riutilizzabili.
    - **hooks**: Custom hooks per gestire logiche riutilizzabili.
    - **libs**: Librerie e utilità comuni utilizzate nell'applicazione *frontend*.
    - **models**: Definizioni dei modelli di dati utilizzati nell'applicazione *frontend*.
    - **pages**: Componenti delle pagine principali dell'applicazione.
    - **stores**: Gestione dello stato globale dell'applicazione (es. Redux, Context *API*).
    - **themes**: Temi e stili globali dell'applicazione.

3.5.1 Problematiche riscontrate

Durante la fase di codifica sono emerse diverse problematiche che sono state affrontate e risolte grazie alla collaborazione con i tutor aziendali:

- **Integrazione con API**: Uno dei principali problemi riscontrati è stata l'utilizzo delle *API* per la comunicazione con il backend, in quanto esse vanno definite seguendo metodologie precise in modo da essere utilizzate in modo corretto e coerente.
  Più di qualche volta, infatti, mi è capitato di posizionare una *API* nella sezione errata di gestione delle chiamate (operazione che non compromette il funzionamento ma che inficia la qualità del codice per eventuali manutenzioni da parte di altri sviluppatori).

- **Gestione dello stato globale**: Un'altra sfida è stata la gestione dello stato globale nell'applicazione *frontend*. Utilizzando Redux, ho potuto centralizzare la gestione dello stato, ma è stato necessario un notevole sforzo per strutturare correttamente questa tecnologia a causa della ripida curva di apprendimento.
- **Mantenere la coerenza stilistica del codice**: In un team di sviluppo è essenziale mantenere la coerenza del codice. Ho utilizzato strumenti come ESLint e Prettier per assicurare che tutto il codice fosse conforme agli stessi standard di stile e qualità e che fosse quindi correttamente formattato e privo di errori.

3.5.2 Strumenti utilizzati

Durante lo sviluppo del progetto, sono stati utilizzati vari strumenti per migliorare la produttività e la qualità del codice:

- **Visual Studio Code**: L'IDE principale utilizzato per lo sviluppo, grazie alla sua vasta gamma di estensioni e alla sua integrazione con TypeScript e altri strumenti di sviluppo.
- **ESLint**: Strumento di *linting* per identificare e correggere problemi nel codice JavaScript/TypeScript.
- **Prettier**: Formattatore di codice per mantenere uno stile coerente in tutto il progetto.
- **Husky**: Utilizzato per eseguire *script pre-commit* e *pre-push* per garantire che il codice sia conforme agli standard prima di essere committato.
- **Lerna**: Utilizzato per la gestione dei pacchetti e delle dipendenze tra i vari moduli del progetto.
- **MongoDB Compass**: Strumento che permette di visualizzare e interagire con i dati presenti nel database MongoDB, per verificare il corretto funzionamento del software.

3.5.3 Best practices seguite

Nel corso della codifica, sono state seguite diverse best practices per garantire la qualità e la manutenibilità del codice:

- **Code reviews**: Ogni *merge request* è stata sottoposta a code review da parte del tutor per assicurare la qualità e la conformità agli standard del progetto.
- **Documentazione**: Tutte le principali funzioni e moduli sono stati adeguatamente documentati, sia tramite commenti nel codice che con documentazione esterna.
- **Refactoring costante**: È stato effettuato un *refactoring* costante del codice, sotto i consigli dei tutor, per migliorare la leggibilità e ridurre la complessità, assicurando che il codice rimanesse pulito e facile da mantenere.

- **Utilizzo di design patterns**: Sono stati applicati design patterns consolidati, come il Singleton e il Factory prima descritti, per risolvere problemi comuni di progettazione e migliorare la struttura del codice.

Queste pratiche, combinate con l'adozione di una struttura modulare e l'utilizzo di tecnologie moderne, hanno contribuito a sviluppare un software robusto, scalabile e manutenibile.

3.6 Verifica

La **verifica** ha come obiettivo che il risultato del lavoro svolto sia conforme alle necessità espresse all'interno del ticket collegato all'attività e che il nuovo codice non abbia introdotto nuovi errori.

Esistono due tipologie di verifica, che vanno applicate in parallelo durante il ciclo di vita del *software*:

- **Analisi statica**: è una tecnica di valutazione del codice sorgente di un software senza eseguirlo. Questo tipo di analisi viene effettuato durante la fase di sviluppo e prima dell'esecuzione del programma. Gli strumenti di analisi statica esaminano il codice per identificare potenziali errori, vulnerabilità di sicurezza, e violazioni delle best practices di programmazione. L'obiettivo è migliorare la qualità del codice e prevenire bug che potrebbero causare problemi in fase di *runtime*.

- **Analisi dinamica**: è una tecnica di valutazione del software che viene eseguita durante l'esecuzione del programma. Questo tipo di analisi coinvolge l'osservazione del comportamento del software in un ambiente di *runtime* per identificare errori che si manifestano solo durante l'esecuzione, come problemi di gestione della memoria, condizioni di gara, e difetti logici. Gli strumenti di analisi dinamica monitorano vari aspetti del comportamento del programma, inclusi l'uso della CPU, della memoria, e l'interazione con altri sistemi.

Analisi statica

Ho svolto attività di **analisi statica** nel periodo di tirocinio:
Per quanto concerne la parte di documentazione sono stato coadiuvato dai sistemi di controllo ortografico e di sintassi integrati in *Google Docs*. Nonostante questi aiuti automatici, ho svolto attività di verifica manuale per garantire la correttezza e la coerenza della documentazione.
Per quanto concerne invece la parte di codice, ho utilizzato *ESLint* e *Prettier* per garantire la coerenza e la qualità del codice sorgente. Anche in questo caso, prima di ogni *commit* avveniva una verifica manuale di tutto il contenuto che desideravo committare.

Analisi dinamica

Ho svolto attività di *analisi dinamica* del *modello a V* nel periodo di tirocinio:

- **Test di sistema**: è un *test* che verifica il comportamento del sistema rispetto ai requisiti funzionali e non funzionali specificati nella analisi dei requisiti.
- **Test di integrazione**: è un *test* che verifica che i componenti del sistema funzionino correttamente quando vengono integrati insieme.
- **Test di unità**: è un *test* che verifica il corretto funzionamento di una singola unità di codice, come una funzione o un metodo.

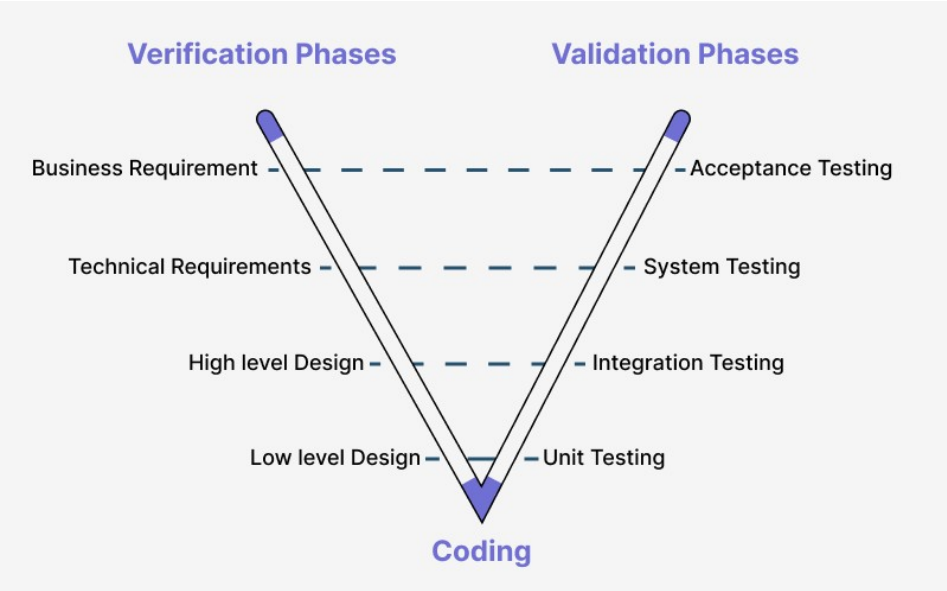


Figura 3.10: Modello di sviluppo adottato[24]

A seguito di una iniziale discussione con i tutor aziendale, si è deciso che i *test* di analisi dinamica venissero svolti in toto manualmente dato il poco tempo per lo sviluppo dovuto alla notevole mole di ricerca e studio necessaria per l'utilizzo di tecnologie per l'intelligenza artificiale e la mia inesperienza con il *framework Jest*, che sarebbe stato utilizzato per l'automazione dei *test*.

3.7 Validazione

La **validazione** ha come obiettivo accertarsi che il prodotto rispecchi le richieste del proponente. Durante gli ultimi due giorni di tirocinio, ho svolto attività di collaudo del progetto, in particolare ho verificato che il prodotto fosse conforme ai requisiti e che svolgesse tutte le funzionalità che mi erano state richieste e che ho sviluppato. A tal proposito per verificare la correttezza del prodotto, abbiamo avviato il *software* tramite un computer diverso da quello su cui era stato sviluppato, in modo da verificare che non ci fossero problemi di compatibilità.

[24] Fonte: https://codegym.cc/it/

3.8 Risultato finale

Il risultato finale del progetto è stato un *software* con le seguenti caratteristiche:

3.8.1 Statistiche qualitative e quantitative

Requisiti soddisfatti

Il prodotto realizzato non completa tutti i casi d'uso previsti nella iniziale analisi, ma soddisfa tutti gli Obiettivi obbligatori desiderabili, e qualche requisito opzionale. Questo è dovuto al fatto che era previsto che il progetto non fosse completabile nelle ore previste dallo stage, dato il suo carattere molto esplorativo delle nuove tecnologie, e che esso fosse portato avanti e infine completato dallo stagista successivo.

Tipologia	Quantità	Soddisfatti
Requisiti	46	31

Tabella 3.1: Tabella riassuntiva dei requisiti soddisfatti

I requisiti che non sono riuscito a soddisfare, erano quelli etichettati con priorità bassa e che gestivano la visualizzazione e gestione dei documenti e delle schede, di questa sezione se ne occuperà lo stagista successivo.

Quantità di prodotti

Metrica	Valore
Documenti prodotti	3
Componenti *React*	17
API	8
Linee di codice	4806
Numero di file	156

Tabella 3.2: Statistiche quantitative sul prodotto

Figura 3.11: Pagina di *Login* di ***SportAI***

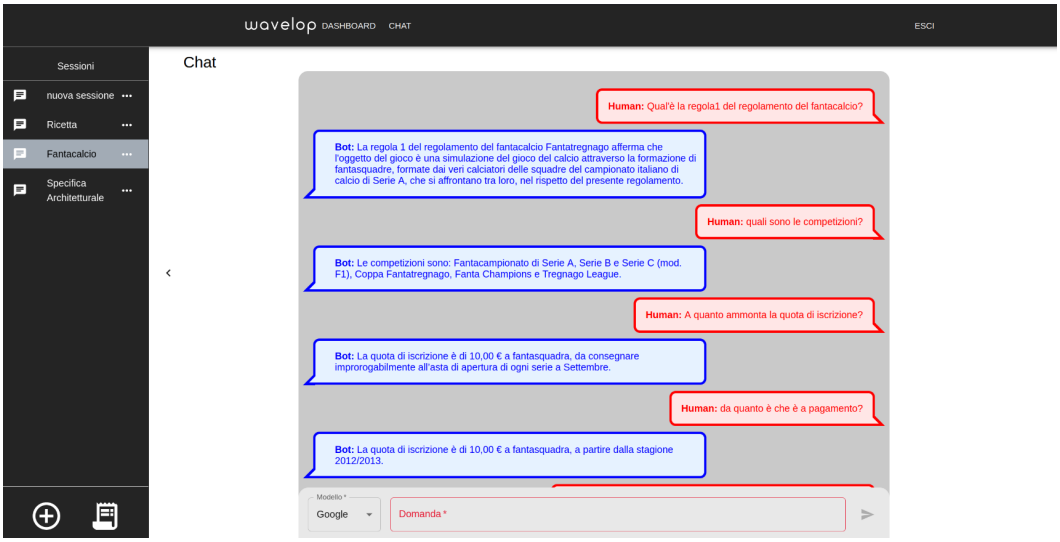


Figura 3.12: Pagina di *Chat* di ***SportAI***

Capitolo 4

Retrospettiva delle attività

Questo capitolo contiene una valutazione retrospettiva sulle attività svolte ed il risultato ottenuto, mettendo a confronto:

- Aspettative e risultati raggiunti;
- Competenze acquisite durante le attività e competenze erogate dal corso di studi.

4.1 Raggiungimento degli obiettivi prefissati

4.1.1 Obiettivi aziendali

Vengono riportati gli obiettivi aziendali indicati nella sezione §2.3.

Obiettivi obbligatori

Obiettivo	Soddisfatto	Fonte
O01: Apprendimento delle tecnologie di sviluppo React, Node.js, Fastify.	SÌ	3.2.1 Strumenti di sviluppo.
O02: Apprendimento delle tecnologie Git e utilizzo per il versionamento del codice.	SÌ	§3.2.2 Strumenti di versionamento.
O03: Analisi e scelta della soluzione IA da utilizzare.	SÌ	§3.4.1 LLM a confronto
O04: Apprendimento delle tecnologie e servizi IA per l'interfacciamento con la soluzione scelta.	SÌ	§3.2.1 Strumenti di sviluppo
O05: Sviluppo di un'interfaccia grafica (simil chat) per la comunicazione con l'IA.	SÌ	3.9 Dashboard
O06: Sviluppo di un sistema di database per conservare le conversazioni, le schede e gli utenti.	SÌ	3.4.3 Livello dei Dati
O07: Sviluppo di un sistema di generazione di schede a partire dalle chat.	SÌ	Realizzato e implementato.

Obiettivi desiderabili

Obiettivo	Soddisfatto	Fonte
D01: Sviluppo di una soluzione back-office per l'inserimento di documenti per l'apprendimento dell'IA.	Parziale	Ho realizzato l'API per l'inserimento
D02: Gestione utenti dell'applicativo.	SÌ	3.8 Login
D03: Sviluppo di un sistema per la gestione delle schede create.	Parziale	Vengono salvate in MinIO

Tabella 4.2: Obiettivi di tirocinio - desiderabili

Gli obiettivi **D01** e **D03** sono stati soddisfatti parzialmente: in entrambi i casi ho implementato la parte di *backend* necessaria per il funzionamento, ma non ho sviluppato l'interfaccia grafica, per cui per il caricamento dei documenti si usa *Postman*, mentre per la gestione delle schede, si possono visualizzano all'interno del *Database* MinIO.

Obiettivi facoltativi

Obiettivo	Soddisfatto	Fonte
F01: Sviluppo di una soluzione di feedback per l'apprendimento dell'IA	NO	-
F02: Sviluppo di una soluzione di tag per migliorare l'*user experience*	NO	-
F03: Sviluppo di un sistema *role-based* per l'accesso all'applicativo.	Parziale	3.8 Login

Tabella 4.3: Obiettivi di tirocinio - facoltativi

Gli obiettivi **F01** e **F02** non sono stati soddisfatti in quanto creati ad inizio progetto, ma non sono stati ritenuti necessari per il funzionamento dell'applicativo. L'obiettivo **F03** è stato soddisfatto parzialmente: ho implementato il sistema di *login* con due ruoli, ma non è stato implementato il sistema *role-based* nel *frontend*.

4.1.2 Obiettivi personali

Vengono riportati gli obiettivi personali indicati nella sezione §2.6.

Obiettivo	Soddisfatto	Fonte
Realizzare una Web App da zero, dall'analisi al collaudo.	SÌ	§3.8 Risultato
Progettare e sviluppare un'interfaccia grafica semplice, intuitiva, funzionale e accessibile ma allo stesso tempo accattivante e moderna.	SÌ	3.9 Dashboard
Realizzare il prodotto adoperando modelli di progettazione e pattern architetturali.	SÌ	§3.4.3 Architettura

Obiettivo	Soddisfatto	Fonte
Apprendimento delle tecnologie per l'intelligenza artificiale: LLM e il RAG	SÌ	3.5 Schema funzionamento
Completare gli obiettivi aziendali posti per il progetto dando la priorità a quelli obbligatori	SÌ	§4.1.1 Obiettivi obbligatori
Partecipare attivamente a stand-up meeting e sprint review	SÌ	§3.1 Stile lavorativo
Realizzare controlli automatici di qualità del codice, in modo da garantire la qualità del prodotto finale	NO	-

Tabella 4.4: Obiettivi di tirocinio - personali

Ho soddisfatto tutti gli obiettivi personali, ad eccezione dell'ultimo, in quanto non ho avuto il tempo di implementare controlli automatici di qualità del codice.

Commento

Valuto la mia attività di stage in maniera positiva, in quanto sono riuscito a portare a termine la maggior parte degli obiettivi assegnatemi e quelli personali, mentre gli obiettivi non raggiunti risultavano marginali o non necessari per il funzionamento dell'applicativo.

4.2 Competenze e conoscenze acquisite

Durante il periodo di stage ho avuto modo di acquisire una serie di competenze e conoscenze che hanno arricchito il mio bagaglio professionale e personale:

Ambito professionale

A livello professionale, ho avuto l'opportunità di approfondire le mie conoscenze in *JavaScript* e *TypeScript*, due linguaggi fondamentali per lo sviluppo di applicazioni web moderne. Questa esperienza di stage si è rivelata un'esperienza completa nel campo dello sviluppo web, coprendo sia l'aspetto *frontend* che quello *backend*. Per quanto riguarda il *frontend*, ho sviluppato competenze avanzate nell'uso del framework *React*, comprendendo tutte le convenzioni stilistiche, gli stili e le tecnologie ad esso legate. Ho imparato a progettare interfacce utente reattive ed intuitive, gestendo lo stato dell'applicazione con *Redux* e migliorando le performance delle applicazioni attraverso tecniche di ottimizzazione.
Sul versante *backend*, ho approfondito la gestione dei dati, le richieste API e tutto ciò che ruota intorno all'ecosistema *Node.js* e *Fastify*. Ho acquisito una solida comprensione di come gestire server efficienti, implementando logiche di business complesse e integrando database non relazionali.
Inoltre, ho esplorato il campo del *machine learning*, concentrandomi in particolare sugli *LLM* (Large Language Models) e sulla metodologia *RAG* (*Retrieval-Augmented Generation*). Infine, ho affinato le mie capacità nell'uso di *Git* per il controllo delle versioni e nella gestione di progetti software, seguendo pratiche di sviluppo agile come

Scrum. Questo mi ha permesso di collaborare efficacemente con il team, assicurando un flusso di lavoro fluido e la consegna di progetti di alta qualità.

Ambito personale

A livello personale, l'esperienza di stage mi ha permesso di migliorare significativamente le mie *soft skills*, che sono essenziali per il successo in qualsiasi ambiente lavorativo. Ho migliorato la mia capacità di lavorare in team, imparando a collaborare efficacemente con colleghi di diverse competenze e background. La comunicazione è stata un elemento chiave, e ho sviluppato abilità nel presentare idee in modo chiaro e nel ricevere e dare feedback costruttivi. La gestione del tempo e delle priorità è un'altra area in cui ho visto notevoli miglioramenti. Ho imparato a organizzare il mio lavoro in modo efficiente, impostando obiettivi chiari e rispettando le scadenze, il che ha aumentato la mia produttività. Inoltre, ho potuto potenziare le mie capacità di *problem solving* e di analisi. L'affrontare problemi complessi e situazioni nuove mi ha permesso di sviluppare un approccio analitico e metodico, migliorando la mia capacità di identificare rapidamente le cause dei problemi e di trovare soluzioni efficaci. Queste competenze personali non solo hanno migliorato la mia performance lavorativa, ma mi hanno anche aiutato a crescere come persona, rendendomi più pronto per delle nuove sfide.

4.3 Competenze curricolari e lavorative

Nonostante l'iniziale pessimismo riguardo la preparazione ricevuta durante il corso di studi, ho potuto constatare che le competenze acquisite durante il percorso universitario sono state sufficienti per affrontare le attività di stage. Infatti, ho potuto notare come questi aspetti siano stati trattati durante i corsi di studio:

- **Programmazione**: durante il corso dei vari corsi universitari che ho frequentato, ho potuto apprendere i concetti fondamentali della programmazione, come la programmazione ad oggetti, la programmazione funzionale e la programmazione procedurale, che mi hanno permesso di affrontare con successo lo sviluppo dell'applicativo; Ho dovuto però approfondire le mie conoscenze in *JavaScript* e Node.js, in quanto trattati solo marginalmente durante il corso di studi;
- **Gestione di progetto**: durante il corso dei vari progetti didattici svolti, ho potuto constatare di aver colto gli elementi fondamentali per la buona riuscita di un progetto, come l'utilizzo di metodologie agili, il versionamento del codice e la gestione del tempo;
- **Ricerca e studio autonomo**: durante il corso di studi ho appreso l'importanza di saper cercare e studiare autonomamente nuove tecnologie e strumenti, competenza che mi è stata molto utile durante lo stage per apprendere le tecnologie necessarie per lo sviluppo dell'applicativo;
- **Comunicazione**: durante i progetti didattici ho avuto modo di capire l'importanza di una comunicazione chiara e celere con i colleghi in modo da poter lavorare in modo efficiente, inoltre ho potuto migliorare le mie capacità di presentazione e di scrittura;

Acronimi e abbreviazioni

CI/CD *Continuous Integration/Continuous Delivery*. 4, 46

IA *Intelligenza Artificiale*. 46, 47

IoT *Internet of Things*. 48

IT *Information Technology*, acronimo usato per indicare persone o cose attinenti all'ambito informatico. 1

LLM *Large Language Model*. 46, 48

RAG *Retrieval Augmented Generation*. 46, 48

REST *Representational State Transfer*. 48

UML *Unified Modeling Language*. 49

WA *Web App*. 49

Glossario

Agile in informatica con il termine *"agile"* si intende un insieme di metodi di sviluppo del *software* basati su processi iterativi e incrementali, dove i requisiti e le soluzioni si evolvono attraverso la collaborazione tra team auto-organizzati e multidisciplinari.
Agile. URL: https://www.atlassian.com/it/agile . 3

Backend con il termine *"backend"* si intende la parte non visibile all'utente di un programma, che elabora e gestisce i dati generati dall'interfaccia grafica.
Backend. URL: https://it.wikipedia.org/wiki/Front-end_e_back-end . 2

Black box con il termine *"black box"* si intende un dispositivo con riferimento alle sole caratteristiche esterne, in particolare alle funzioni di trasferimento tra grandezze di ingresso o di uscita, ignorando cioè del tutto la costituzione interna.
Black box. URL: https://www.treccani.it/vocabolario/scatola/ . 10

E-commerce in informatica con il termine *"e-commerce"* si intende la compravendita di beni e servizi tramite Internet.
ecommerce. URL: https://www.treccani.it/enciclopedia/e-commerce/?search=e-commerce%2F . 2

Frontend con il termine *"frontend"* si intende la parte visibile all'utente di un programma e con cui egli può interagire, tipicamente un'interfaccia utente.
Frontend. URL: https://it.wikipedia.org/wiki/Front-end_e_back-end . 2

Intelligenza Artificiale in informatica con il termine *"intelligenza artificiale"* si intende una tecnologia che consente a computer e macchine di simulare l'intelligenza e la capacità di risoluzione dei problemi degli esseri umani. *Artificial Intelligence*. URL: https://www.ibm.com/it-it/topics/artificial-intelligence. 10

Innovazione con il termine "innovazione" si intende l'atto e l'effetto dell'innovare, cioè dell'introdurre concetti, metodi, strumenti nuovi.
Innovazione. URL: https://www.treccani.it/enciclopedia/innovazione_(Dizionario-delle-Scienze-Fisiche)/ . 6

IoT in informatica con il termine *"Internet of Things"* si intende la rete di dispositivi fisici, veicoli, elettrodomestici e altri oggetti incorporati con sensori, software e altre tecnologie che connettono e scambiano dati con altri dispositivi e sistemi su Internet.
IoT. URL: https://www.treccani.it/vocabolario/internet-of-things_res-7e9f1ee2-7aa2-11eb-9a1f-00271042e8d9_(Neologismi)/ . 2, 46

Large Language Model in informatica con il termine *"Large Language Model"* si intende un modello di *machine learning* che utilizza una grande quantità di dati per apprendere e generare testi in linguaggio naturale.
Large Language Model. URL: https://www.hpe.com/emea_europe/en/what-is/large-language-model.html . 9

Pipeline in informatica con il termine *"pipeline"* si intende un insieme di passaggi che vengono eseguiti in sequenza per completare un'operazione.
Pipeline. URL: https://www.redhat.com/it/topics/devops/what-cicd-pipeline. 4

Retrieval Augmented Generation in informatica La Retrieval-Augmented Generation (RAG) è il processo di ottimizzazione dell'output di un modello linguistico di grandi dimensioni, in modo che faccia riferimento a una base di conoscenza autorevole al di fuori delle sue fonti di dati di addestramento prima di generare una risposta. I modelli linguistici di grandi dimensioni (LLM) vengono addestrati su vasti volumi di dati e utilizzano miliardi di parametri per generare output originali per attività come rispondere a domande, tradurre lingue e completare frasi. La RAG estende le capacità già avanzate degli LLM a domini specifici o alla knowledge base interna di un'organizzazione, il tutto senza la necessità di riaddestrare il modello. È un approccio conveniente per migliorare l'output LLM in modo che rimanga pertinente, accurato e utile in vari contesti. *RAG*. URL: https://aws.amazon.com/it/what-is/retrieval-augmented-generation/. 9

REST in informatica con il termine *Representational State Transfer* si intende uno stile architetturale di comunicazione tra *software* dotato dei seguenti principi:

1. **Assenza di stato**: ogni richiesta del *client* al *server* deve contenere tutte le informazioni necessarie per comprendere e elaborare la richiesta.
   Lo stato del *client* non è memorizzato sul *server* tra le richieste;
2. **Architettura *client-server***: il sistema è diviso in due parti indipendenti: il *client*, che si occupa dell'interfaccia utente e delle interazioni dell'utente, e il *server*, che gestisce la logica aziendale e conserva le risorse;
3. **Possibilità di salvare in *cache* le risposte**: le risposte del *server* devono essere esplicitamente contrassegnate come *cacheable* o *non cacheable*. Ciò consente ai *client* di memorizzare in modo efficiente le risorse e migliorare le prestazioni complessive del sistema;
4. **Interfaccia uniforme**: l'interfaccia tra il *client* e il *server* per l'ottenimento della stessa risorsa deve essere uniforme, non importa da dove proviene la richiesta;

5. **Sistema stratificato**: l'architettura può essere suddivisa in livelli, con ogni livello che svolge un ruolo specifico.

Representational State Transfer. URL: https://www.ibm.com/topics/rest-apis . 12, 46

Software house con il termine *"software house"* si intende un'azienda specializzata nella produzione di *software* il cui obiettivo è quello di sviluppare applicazioni informatiche personalizzate per i propri clienti, che possano soddisfare le loro esigenze specifiche; si occupa dell'intero processo di sviluppo: dalle attività di analisi e progettazione, alla scrittura del codice, alla messa in produzione e manutenzione.
Software house. URL: https://www.businesscompetence.it/cose-una-software-house/. 1

Sprint con il termine *"Sprint"* si intende un'unità di tempo, tipicamente di durata fissa, in cui un *team* di sviluppo lavora per completare un insieme di attività.
Sprint. URL: https://www.atlassian.com/it/agile/scrum/sprints . 12

Stakeholder con il termine *"stakeholder"* (it. portatore di interessi) si intendono tutti i soggetti, individui od organizzazioni, attivamente coinvolti in un'iniziativa economica (progetto, azienda), il cui interesse è negativamente o positivamente influenzato dal risultato dell'esecuzione, o dall'andamento, dell'iniziativa e la cui azione o reazione a sua volta influenza le fasi o il completamento di un progetto o il destino di un'organizzazione.
Stakeholder. URL: https://www.treccani.it/enciclopedia/stakeholder/ . 3

Stand-up meeting in informatica con il termine *"stand-up meeting"* si intende una riunione giornaliera in cui i membri di un *team* espongono brevemente lo stato dei lavori, i problemi riscontrati e le attività che svolgeranno durante la giornata.
Standup. URL: https://www.atlassian.com/it/agile/scrum/standups. 4

UML in ingegneria del *software UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma a oggetti. L'*UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico.
Unified Modeling Language. URL: https://it.wikipedia.org/wiki/Unified_Modeling_Language . 3, 46

Web App in informatica con il termine *"Web App"* (it. applicazione web) si indica un'applicazione sviluppata utilizzando tecnologie per lo sviluppo *web*, come un sito *internet*, può funzionare su piattaforme e dispositivi diversi utilizzando un unico codice sorgente; *Web App*. URL: https://culturedigitali.eu/marketing/web-app/. iii, 12, 46

Bibliografia

Siti web consultati

Agile. URL: https://www.atlassian.com/it/agile (cit. a p. 47).

Application Programming Interface. URL: https://www.treccani.it/enciclopedia/api_%28Lessico-del-XXI-Secolo%29/.

Artificial Intelligence. URL: https://www.ibm.com/it-it/topics/artificial-intelligence (cit. a p. 47).

Backend. URL: https://it.wikipedia.org/wiki/Front-end_e_back-end (cit. a p. 47).

Black box. URL: https://www.treccani.it/vocabolario/scatola/ (cit. a p. 47).

ecommerce. URL: https://www.treccani.it/enciclopedia/e-commerce/?search=e-commerce%2F (cit. a p. 47).

Frontend. URL: https://it.wikipedia.org/wiki/Front-end_e_back-end (cit. a p. 47).

Innovazione. URL: https://www.treccani.it/enciclopedia/innovazione_(Dizionario-delle-Scienze-Fisiche)/ (cit. a p. 47).

IoT. URL: https://www.treccani.it/vocabolario/internet-of-things_res-7e9f1ee2-7aa2-11eb-9a1f-00271042e8d9_(Neologismi)/ (cit. a p. 48).

Large Language Model. URL: https://www.hpe.com/emea_europe/en/what-is/large-language-model.html (cit. a p. 48).

Pipeline. URL: https://www.redhat.com/it/topics/devops/what-cicd-pipeline (cit. a p. 48).

RAG. URL: https://aws.amazon.com/it/what-is/retrieval-augmented-generation/ (cit. a p. 48).

Representational State Transfer. URL: https://www.ibm.com/topics/rest-apis (cit. a p. 49).

Software house. URL: https://www.businesscompetence.it/cose-una-software-house/ (cit. a p. 49).

Sprint. URL: https://www.atlassian.com/it/agile/scrum/sprints (cit. a p. 49).

Stakeholder. URL: https://www.treccani.it/enciclopedia/stakeholder/ (cit. a p. 49).

Standup. URL: https://www.atlassian.com/it/agile/scrum/standups (cit. a p. 49).

Unified Modeling Language. URL: https://it.wikipedia.org/wiki/Unified_Modeling_Language (cit. a p. 49).

Web App. URL: https://culturedigitali.eu/marketing/web-app/ (cit. a p. 49).